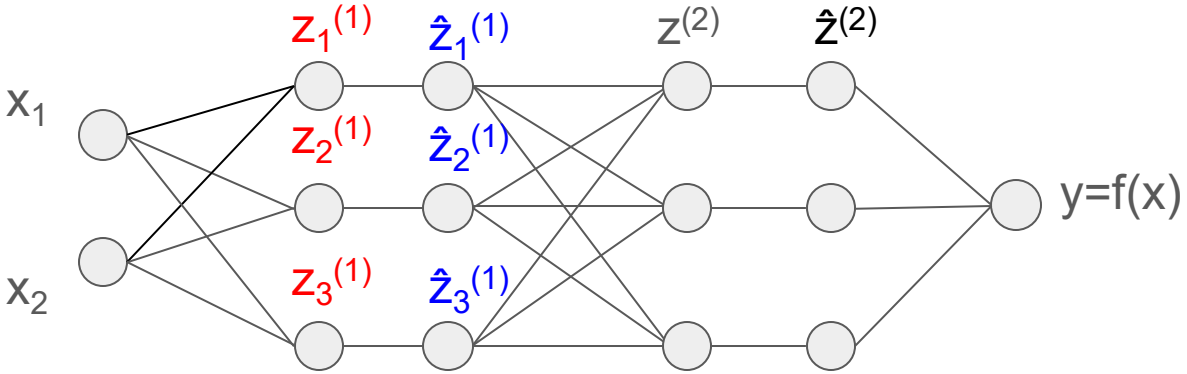
Neural Network Verification
Integer and Linear Programming Formulations

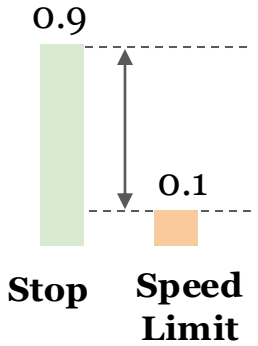
Prof. Sayan Mitra

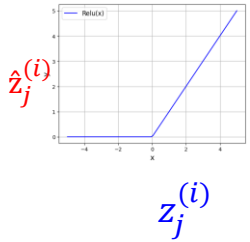
mitras@Illinois.edu

Slides adapted from Prof. Huan Zhang's lectures, Spring 2025

Review: Neural network verification as a satisfiability problem







To verify for all $x \in S$, $y > 0 \wedge y = f(x)$ we solve
Does there exist x, s.t. $x \in S \wedge y \leq 0 \wedge y = f(x)$

Input domain under consideration

Negation of the desired property

Defines the neural network

Verification of neural networks

Satisfiability problem: $\exists\ x \in S\ \wedge\ y \leq 0\ \wedge\ y = f(x)$

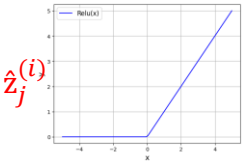
$x_i \leq u_i\ \wedge x_i \geq l_i$ for each dimension of x

$((z_j^{(i)} \geq 0 \wedge \hat{z}_j^{(i)} = z_j^{(i)})\ \vee\ (z_j^{(i)} < 0 \wedge \hat{z}_j^{(i)} = 0))$ for each ReLU neuron

$z_1 = W^{(1)} x \quad \wedge \quad z^{(2)} = W^{(2)} \hat{z}^{(1)} \quad \wedge \quad y = w^{(3)T} \hat{z}^{(2)} \quad \wedge \quad y \leq 0$

Add all clauses to the formula and solve using DPLL(T) with Linear Real Arithmetic.

In general this is very slow!



Today: More efficient algorithms for NN verification

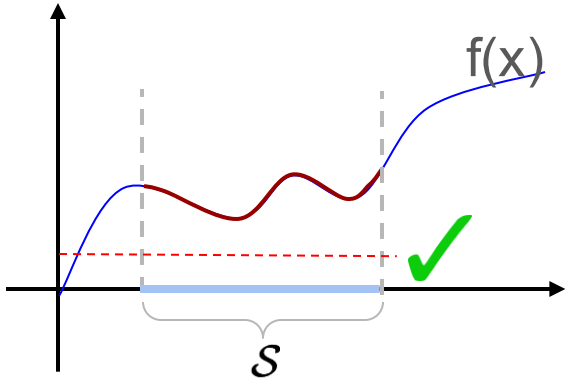
Solving neural network verification using SMT solvers

Solving neural network verification using optimization (MIP/LP)

Solving neural network verification using **bound propagation**

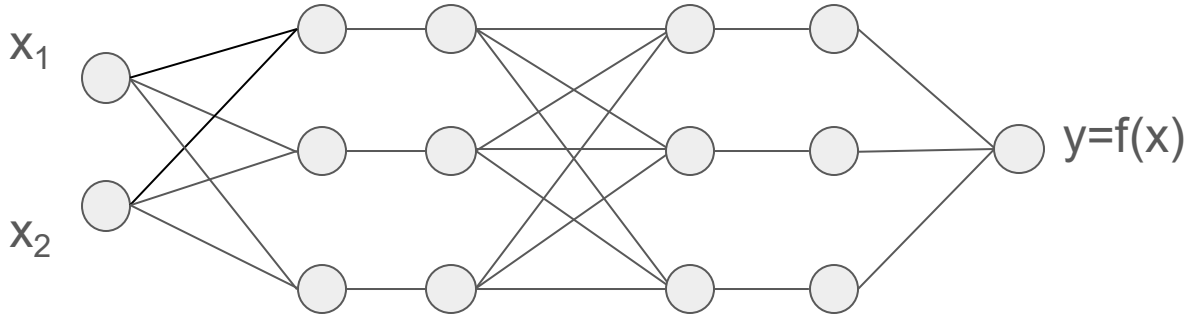
- Interval bound propagation (IBP)
- Linear (symbolic) bound propagation (CROWN)

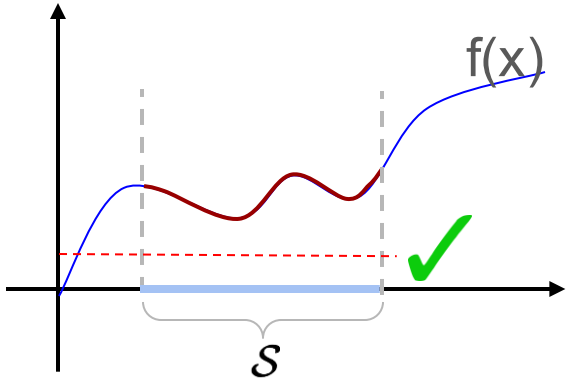
Efficient methods are typically incomplete (solving a lower bound, as tight as possible)



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

Any faster ways to calculate the bounds on f(x)?

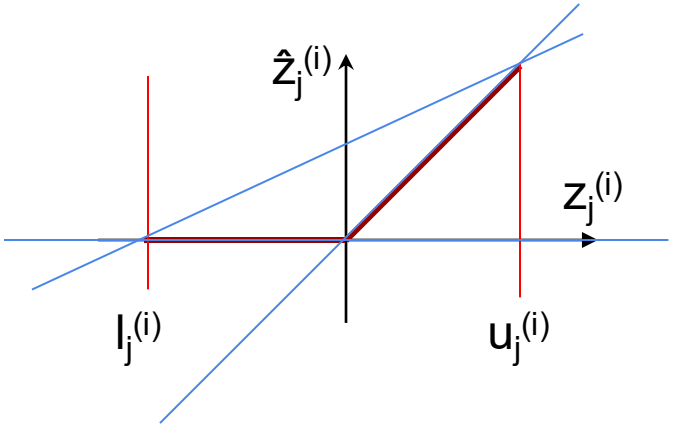


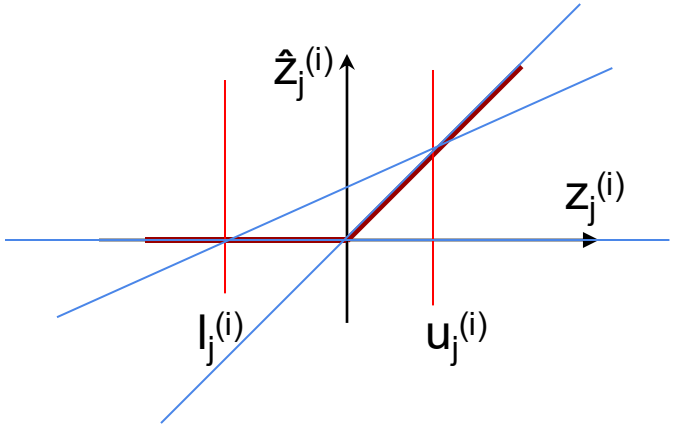


A closer look at the linear programming relaxation

MILP: solutions are constrained on ReLU function
Linear programming: solutions are constrained on the triangle

We want the triangle to be as small as possible! So tight pre-activation bounds are necessary.

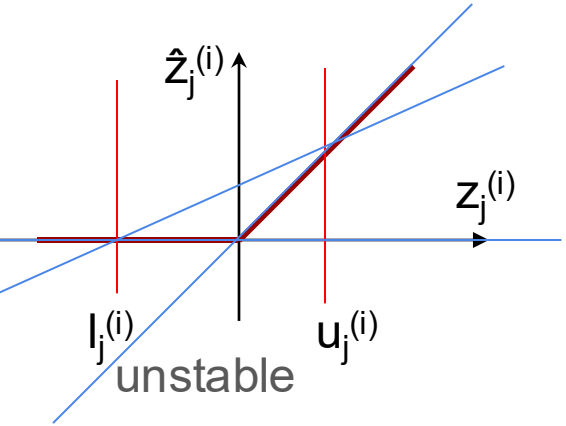


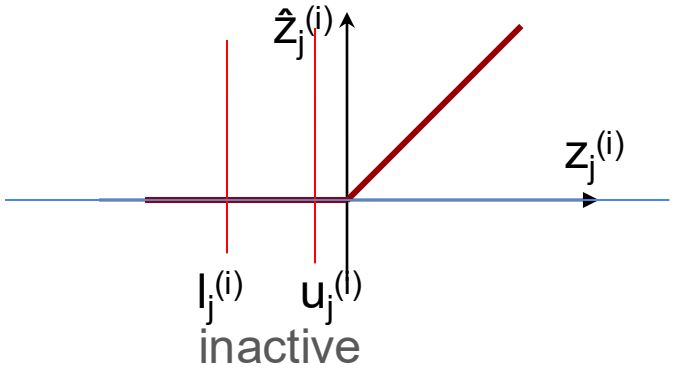


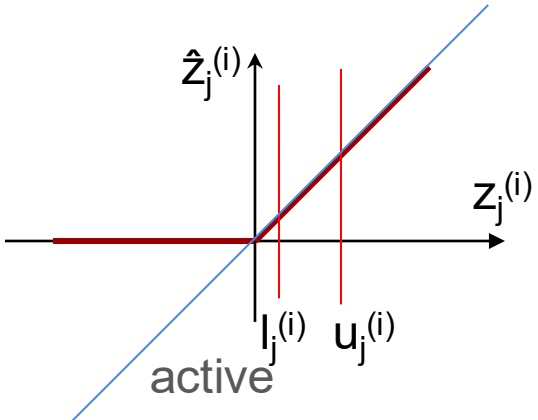
Stable vs. unstable neurons

If bounds are tight enough so $l_j^{(i)} \geq 0$ or $u_j^{(i)} \leq 0$, it is called stable neurons (active or inactive); otherwise it is a unstable neuron

In stable neuron cases, a binary variable in MILP or relaxation in LP is not needed - ReLU becomes a linear function.





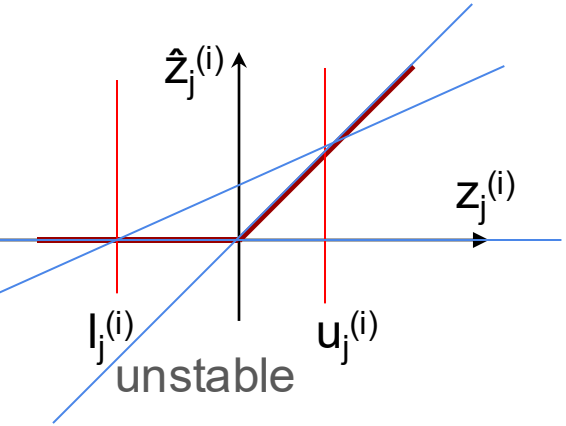


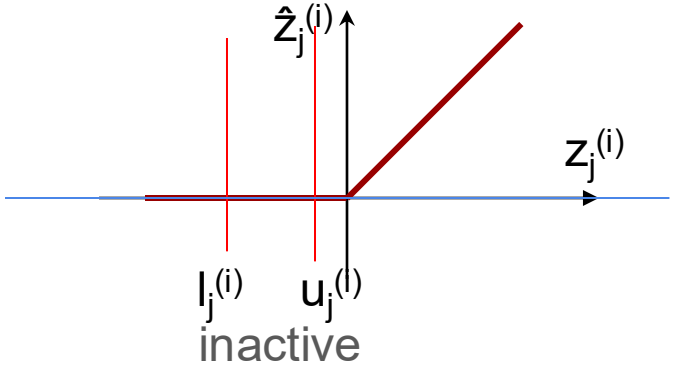
Stable vs. unstable neurons

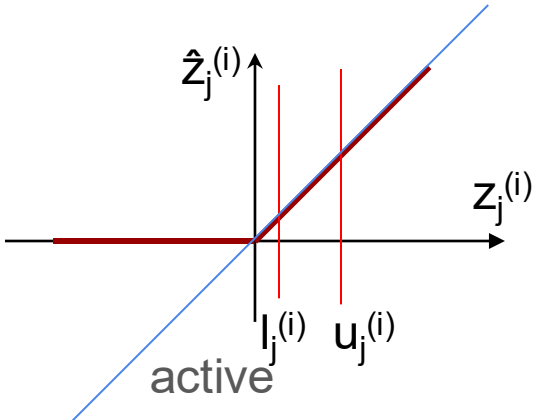
$$\hat{z}_j^{(i)} \leq z_j^{(i)} - l_j^{(i)}(1 - p_j^{(i)})$$
$$\hat{z}_j^{(i)} \leq u_j^{(i)} p_j^{(i)}$$
$$\hat{z}_j^{(i)} \geq z_j^{(i)}$$
$$\hat{z}_j^{(i)} \geq 0$$

$$\hat{z}_j^{(i)} = 0$$

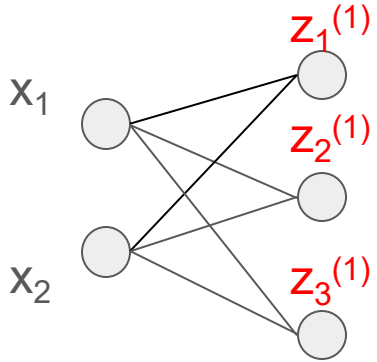
$$\hat{z}_j^{(i)} = z_j^{(i)}$$







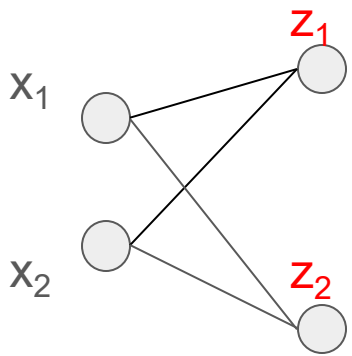
Let’s look at one layer first and interval bounds on input



Given bounds on x, can we calculate the bounds on z?

$$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$$

Let’s look at one layer first



Given bounds on x, can we calculate the bounds on z?

$$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$$

As an illustration, suppose we have

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$

Can you infer bounds on z given bounds on x?

Interval Bound Propagation (IBP)

$$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$$

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$

Lower bounds

$$\underline{z}_1 = -1 - 1 = -2$$

$$\underline{z}_2 = -1 \times 2 - 1 = -3$$

Upper bounds

$$\overline{z}_1 = 2 - (-2) = 4$$

$$\overline{z}_2 = 2 \times 2 - (-2) = 6$$

Interval Bound Propagation (IBP)

$x_1 \in [-1, 2],\ x_2 \in [-2, 1] \qquad z_1 = x_1 - x_2 \qquad z_2 = 2x_1 - x_2$

$$\underline{z}_1 = -1 - 1 = -2 \qquad \overline{z}_1 = 2 - (-2) = 4$$

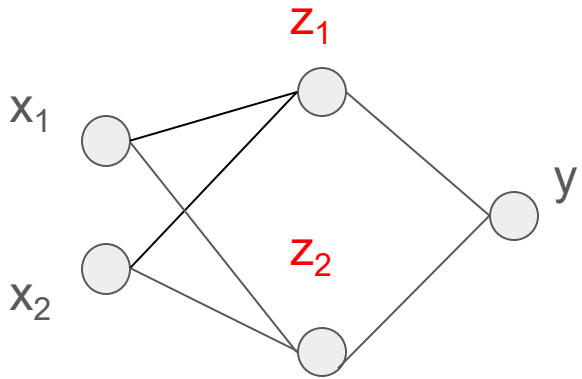
$$\underline{z}_2 = -1 \times 2 - 1 = -3 \qquad \overline{z}_2 = 2 \times 2 - (-2) = 6$$

In general:

$$\sum_{i \in \{i | w_i \geq 0\}} w_i l_i + \sum_{i \in \{i | w_i < 0\}} w_i u_i \leq \sum_i w_i x_i \leq \sum_{i \in \{i | w_i \geq 0\}} w_i u_i + \sum_{i \in \{i | w_i < 0\}} w_i l_i$$

Elements lower and upper bounds of x

Interval Bound Propagation: continue to the next layer



Let's say $y = z_1 - z_2$

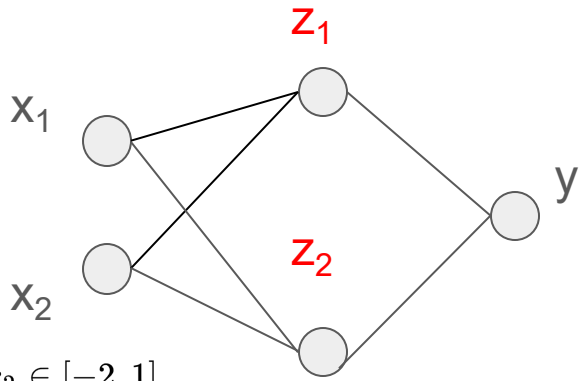
We also know that:

$z_1 \in [-2, 4]$ $z_2 \in [-3, 6]$

Then what can we conclude about y?

$$y \in [-8, 7]$$

Interval Bound Propagation: limitations



$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$

Apply IBP we obtain $y \in [-8, 7]$ for this simple linear network.

However observe that

$$z_1 = x_1 - x_2$$

$$z_2 = 2x_1 - x_2$$

$$y = z_1 - z_2$$

$$y = x_1 - x_2 - (2x_1 - x_2) = -x_1$$

The actual bounds is [-2, 1], much tighter than [-8, 7]

A Better Idea: Keep the correlations between x and z

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$
$$y = z_1 - z_2$$
$$y = x_1 - x_2 - (2x_1 - x_2) = -x_1$$

The actual bounds is [-2, 1], much tighter than [-8, 7]

It is important to keep the correlations between z and x to obtain this tighter result!

We treat z as a **symbolic function of x**, rather than intervals

A Better Idea: linear bound propagation

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$
$$y = z_1 - z_2$$
$$y = x_1 - x_2 - (2x_1 - x_2) = -x_1$$

The actual bounds is [-2, 1], much tighter than [-8, 7]

It is important to keep the correlations between z and x to obtain this tighter result!

We treat z as a **linear function of x**, rather than intervals

A Better Idea: linear bound propagation

$$y = z_1 - z_2 \Longrightarrow y = x_1 - x_2 - (2x_1 - x_2) = -x_1$$

Plug in

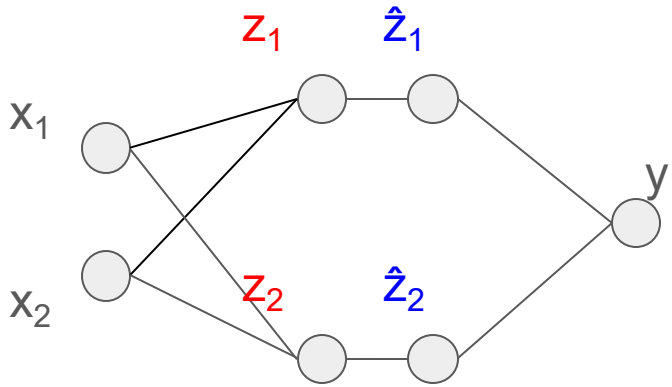
$$z_1 = x_1 - x_2$$

$$z_2 = 2x_1 - x_2$$

We treat z as a **linear function of x**, rather than concrete intervals.

After we plug in linear functions (z w.r.t. x), we still get a linear function (y w.r.t. x)

Bound propagation: how about nonlinear functions?



Can we improve IBP using symbolic linear bounds?

Instead of $y = z_1 - z_2$

Now we have $y = \text{ReLU}(z_1) - \text{ReLU}(z_2)$

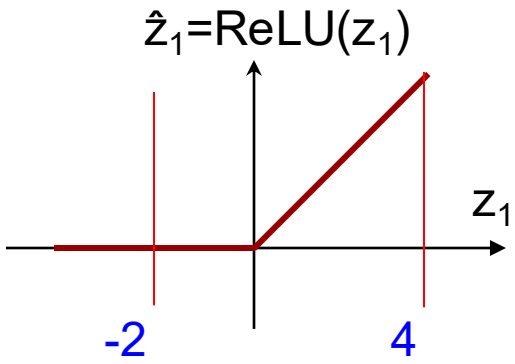
From IBP we already know that

$z_1 \in [-2, 4]$, $z_2 \in [-3, 6]$,

$\text{ReLU}(z_1) \in [0, 4]$, $\text{ReLU}(z_2) \in [0, 6]$

$y \in [-6, 4]$

Linear bound propagation for ReLU function (CROWN)



Instead of $y = z_1 - z_2$

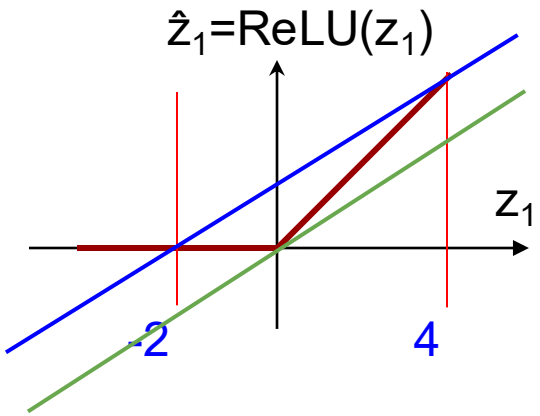
Now we have $y = \text{ReLU}(z_1) - \text{ReLU}(z_2)$

We already know that

$z_1 \in [-2, 4]$, $z_2 \in [-3, 6]$,

(Preactivation bounds)

Linear bound propagation for ReLU function (CROWN)

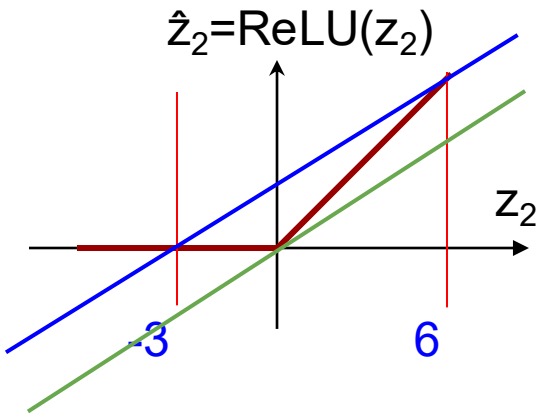


Linear upper bound (same as the one of triangle relaxation in LP)

Linear lower bound (actually not unique)

$$\frac{2}{3}z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

Linear bound propagation for ReLU function (CROWN)



ReLU(z_2) can be bounded using linear functions similarly.

Now let's consider y = ReLU(z_1) - ReLU(z_2). How to bound it using linear functions of z_1 and z_2?

$$\frac{2}{3}z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

$$\frac{2}{3}z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

Linear bound propagation for ReLU function (CROWN)

$$\frac{2}{3}z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3} \qquad \frac{2}{3}z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

Negative coefficient, take upper bound

positive coefficient, take lower bound

$$\frac{2}{3}z_1 - \left(\frac{2}{3}z_2 + 2\right) \leq$$

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

$$\leq \left(\frac{2}{3}z_1 + \frac{4}{3}\right) - \frac{2}{3}z_2$$

Linear bound propagation for ReLU function (CROWN)

$$\frac{2}{3}z_1 - (\frac{2}{3}z_2 + 2) \leq y \leq (\frac{2}{3}z_1 + \frac{4}{3}) - \frac{2}{3}z_2$$

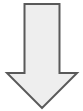
Now we have linear inequalities for y w.r.t. z!

Next step we can simply plug in, as in the linear ($y=z_1-z_2$) case.

Linear bound propagation for ReLU function (CROWN)

$$\frac{2}{3}z_1 - (\frac{2}{3}z_2 + 2) \leq y \leq (\frac{2}{3}z_1 + \frac{4}{3}) - \frac{2}{3}z_2$$

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$



Plug in

$$-\frac{2}{3}x_1 - 2 \leq y \leq -\frac{2}{3}x_1 + \frac{4}{3}$$

Linear bound propagation for ReLU function (CROWN)

We now have symbolic linear bounds for y w.r.t. x

$$-\frac{2}{3}x_1 - 2 \leq y \leq -\frac{2}{3}x_1 + \frac{4}{3}$$

$$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$$

Take lower bound given x

Take upper bound given x

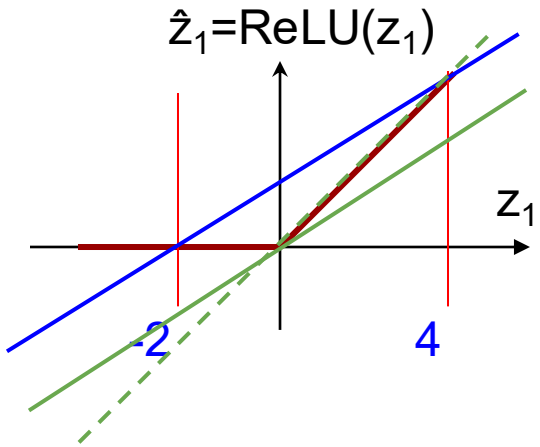
$$y \in \left[-\frac{10}{3}, 2\right]$$

Concrete interval bounds

A lot more tighter than IBP bounds y ∈ [-6, 4]

Can we do even better?

Let's recall that when we linearly bound the ReLU function, there are some flexibilities



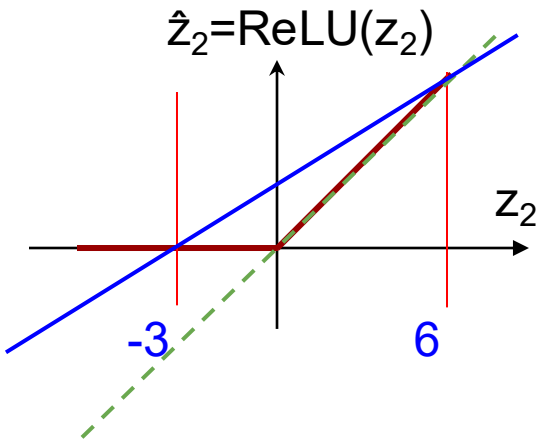
Linear upper bound (same as the one of triangle relaxation in LP)

Linear lower bound (actually not unique)

$$\frac{2}{3}z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

Also valid: $z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$

Choosing different linear bounds (α-CROWN)



$$z_1 \leq \text{ReLU}(z_1) \leq \frac{2}{3} z_1 + \frac{4}{3}$$

$$z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3} z_2 + 2$$

Now what are the linear bounds of
$y = \text{ReLU}(z_1) - \text{ReLU}(z_2)$?

$$z_1 - (\frac{2}{3} z_2 + 2) \leq y \leq (\frac{2}{3} z_1 + \frac{4}{3}) - z_2$$

Choosing different linear bounds (α-CROWN)

$$z_1 - (\tfrac{2}{3}z_2 + 2) \leq y \leq (\tfrac{2}{3}z_1 + \tfrac{4}{3}) - z_2$$

$$z_1 = x_1 - x_2$$
$$z_2 = 2x_1 - x_2$$

Plug in

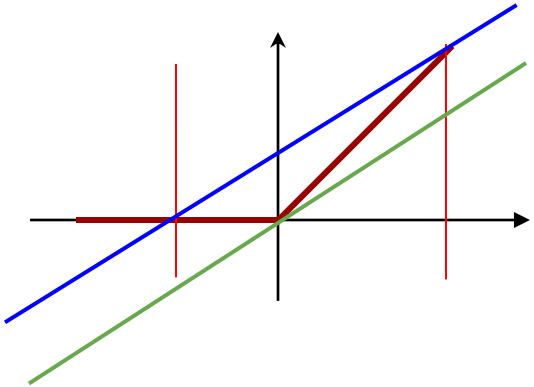
$$-\tfrac{1}{3}x_1 - \tfrac{1}{3}x_2 - 2 \leq y \leq -\tfrac{4}{3}x_1 + \tfrac{1}{3}x_2 + \tfrac{4}{3}$$

$$x_1 \in [-1, 2],\ x_2 \in [-2, 1]$$

Concretize

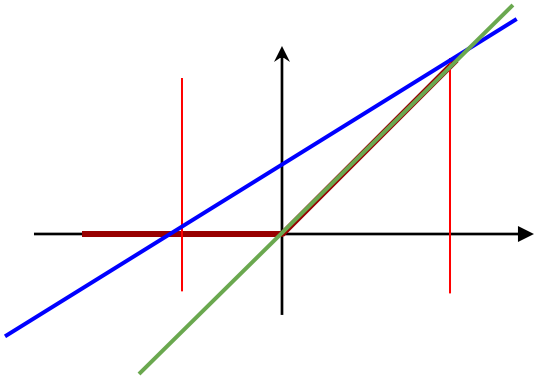
$$y \in [-3, 3]$$

Linear lower bounds for ReLU function matters!



$$\frac{2}{3}z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$
$$\frac{2}{3}z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

$$y \in [-\frac{10}{3}, 2]$$



$$z_1 \leq \mathrm{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$
$$z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

$$y \in [-3, 3]$$

Which one is correct?

Linear lower bounds for ReLU function matters!

Both results are correct! But we want the bounds to be as tight as possible! So best result is **$y \in [-3, 2]$**

In general, the slope of the linear lower bound for every ReLU neuron can be optimized to find the best result.

Linear lower bounds for ReLU function matters!

In general, the slope of the linear lower bound for every ReLU neuron can be optimized to find the best result.

$$\alpha_1 z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3} z_2 + \frac{4}{3}$$

$$\alpha_2 z_2 \leq \mathrm{ReLU}(z_2) \leq \frac{2}{3} z_2 + 2$$

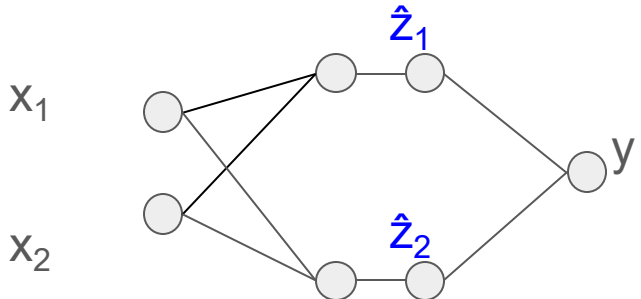
For optimal lower bound of y, set α_1=1, α_2=1

For optimal upper bound of y, set α_1=⅔, α_2=⅔

(note that the optimal α_1 and α_2 do not equal in general)

Linear bound propagation method (CROWN)

1. Obtain all pre-activation bounds (can be done via CROWN recursively)
2. Start from the output layer, form the initial linear (in)equality y = y
3. Recursively propagate linear inequality $y <= a^T z + b$ through each layer:
   a. For a linear layer, z=Wz', directly plug in $a^T z + b$ to get a linear bound of z'
   b. For a non-linear layer (e.g., z=ReLU(z')), we first form the linear inequalities to bound the nonlinear layer itself. Then multiply either the lower or upper bound based on the sign of element in a
4. When the linear inequality propagates to the input layer, we can concretize the linear bound using bounds on input layer.



Example: MLP with 1 Layer

Input: $x \in R^2$ with bounds $x_1, x_2 \in [0,1]$

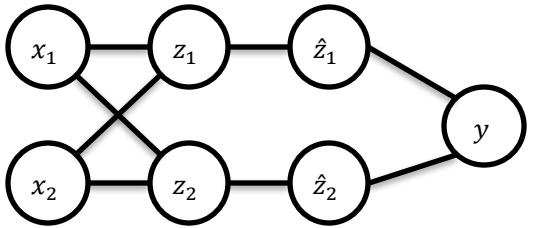
Layer 1: $z = W_1 x + b_1$

- $W_1 = \begin{bmatrix} 1 & -1 \\ 0.5 & 1 \end{bmatrix}$ $b_1 = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$

ReLU: $\hat{z}_i = ReLU(z_i)$

Output: $y = {w_2}^{\mathrm{T}} \hat{z} + b_2$

- $w_2 = \begin{bmatrix} 1.2 \\ -0.7 \end{bmatrix}$ $b_2 = 0.3$



Forward Propagation of Interval Bounds

Neuron 1: $z_1 = x_1 - x_2 + 0.1$

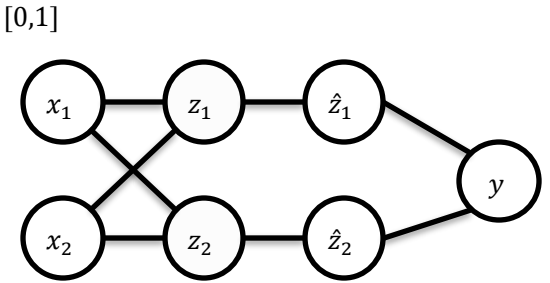
- $x_1, x_2 \in [0,1]$
- $\Rightarrow z_1 \in [-0.9, 1.1]$
- $\Rightarrow \hat{z}_1 \in [0, 1.1]$

Neuron 2: $z_2 = 0.5x_1 + x_2 - 0.2$

- $\Rightarrow z_2 \in [-0.2, 1.3]$
- $\Rightarrow \hat{z}_2 \in [0, 1.3]$

Output: $y = 1.2\,\hat{z}_1 - 0.7\,\hat{z}_2 + 0.3$

- $y \in [-0.61, 1.62]$



3. Backward Propagation of Linear bounds (Crown)

$z_1 \in [-0.9, 1.1] = [\ell_1, u_1]$

$\hat{z}_1 \leq \frac{1.1}{2}(z_1 + 0.9) = 0.55(z_1 + 0.9)$

$z_2 \in [-0.2, 1.3]$

$\hat{z}_2 \leq -\frac{1.3}{1.5}(z_2 + 0.2) = 0.866(z_2 + 0.2)$

$y = 1.2\,\hat{z}_1 - 0.7\,\hat{z}_2 + 0.3$

For each ReLU relaxation, depending on the sign of the coefficient, choose either the **upper** or **lower** bounding inequality:

For positive coefficient (1.2), use the **upper bound on** $\hat{z}_1 \leq 0.55(z_1 + 0.9)$

For negative coefficient (-0.7), to minimize y, use the **lower bound** $\hat{z}_2 \geq z_2$ or $\hat{z}_2 \geq 0$.

$y \leq 1.2 \times 0.55\,(z_1 + 0.9) + 0.3$

Then substitute $z = Wx + b$ to get

$y \leq 0.66x_1 - 0.66x_2 + 0.6$

For $x_1, x_2 \in [0,1]$ by maximizing over this the bound it concretized to $y \leq 1.26$

$$\hat{z}_i \geq z_i$$
$$\hat{z}_i \geq 0$$
$$\hat{z}_i \leq \frac{u_i}{u_i - l_i}(z_i - l_i)$$

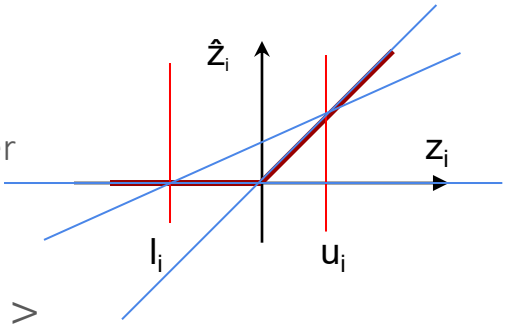
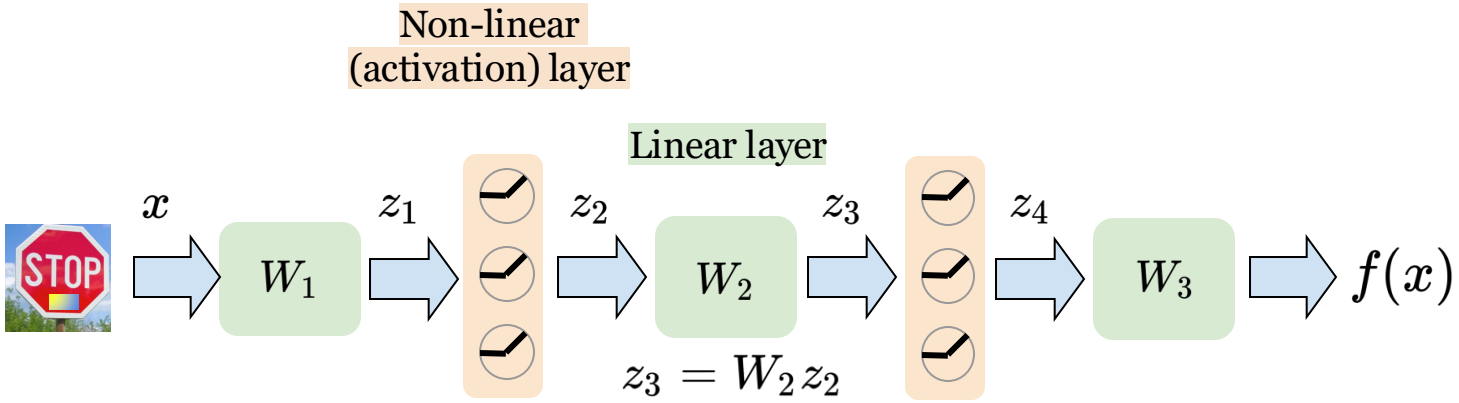
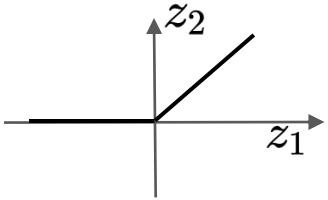


Illustration: Linear bound propagation process



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

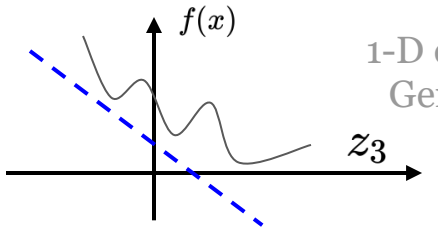


Steps:

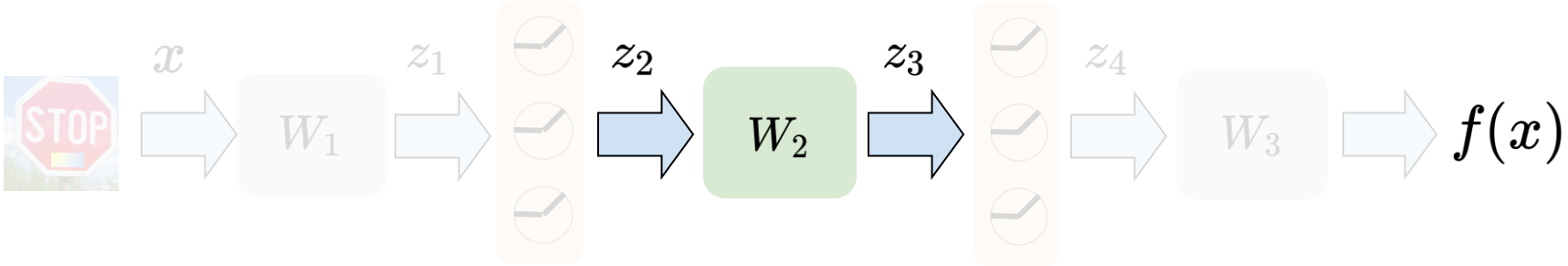
- Propagate bounds through linear layers
- Propagate bounds through non-linear layers

Illustration: Linear bound propagation process

A linear lower bound for an intermediate layer

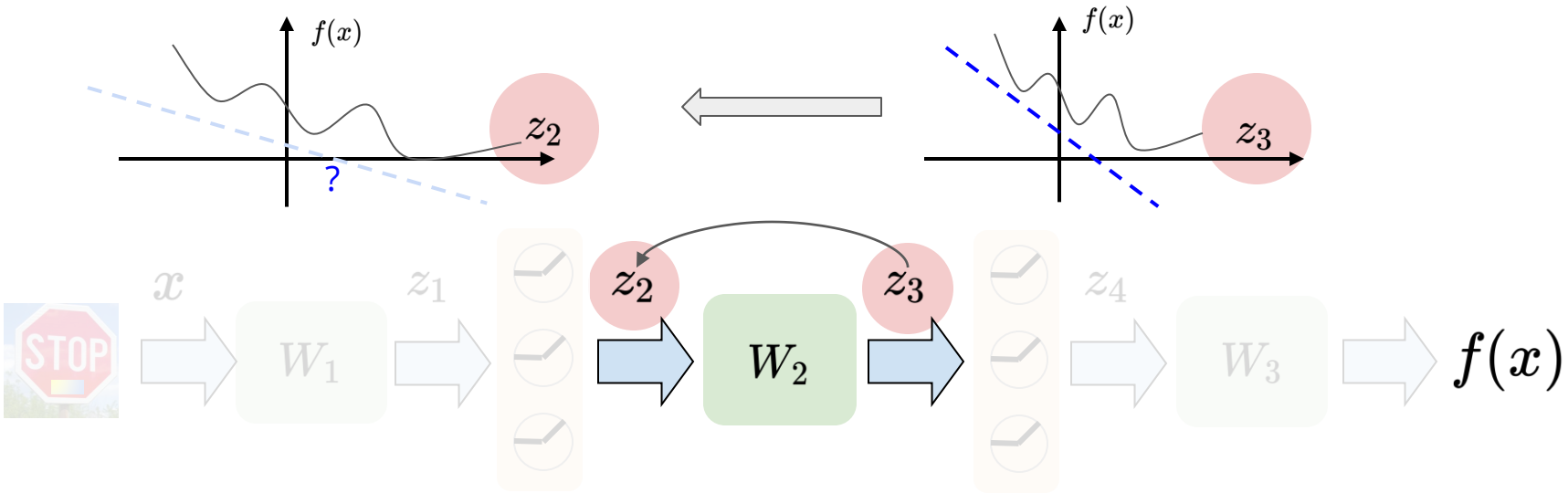


1-D case for illustration. Generally it's a linear hyperplane



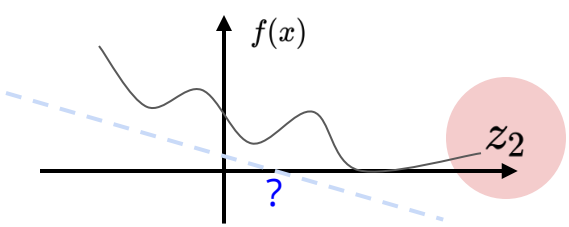
$$f(x) \geq a^\top z_3 + b$$

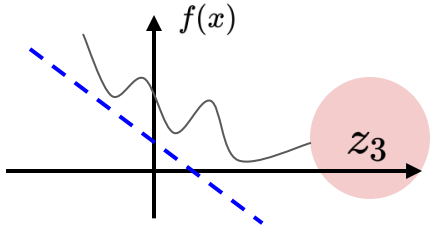
Illustration: Linear bound propagation process

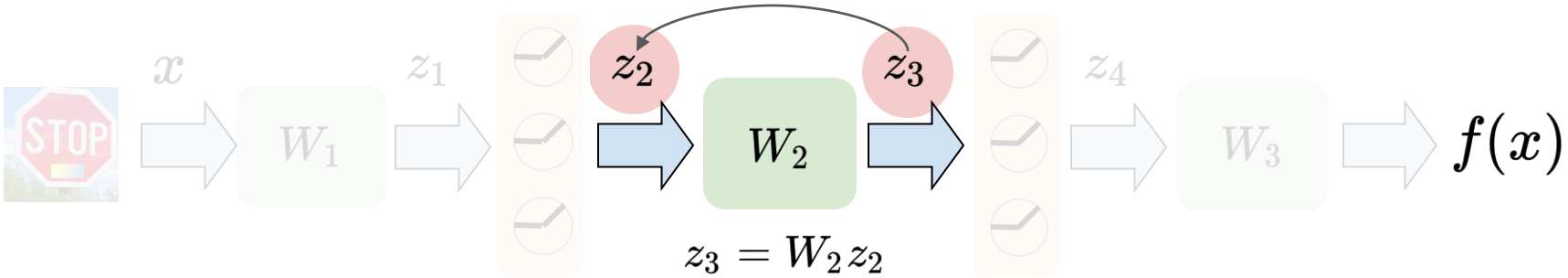


Propagate it to one layer before,
while keeping the lower bound valid

Illustration: Linear bound propagation process



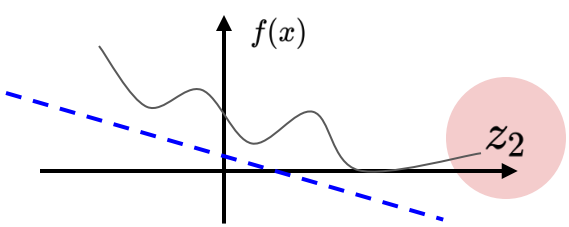


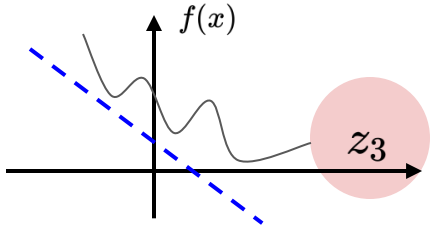


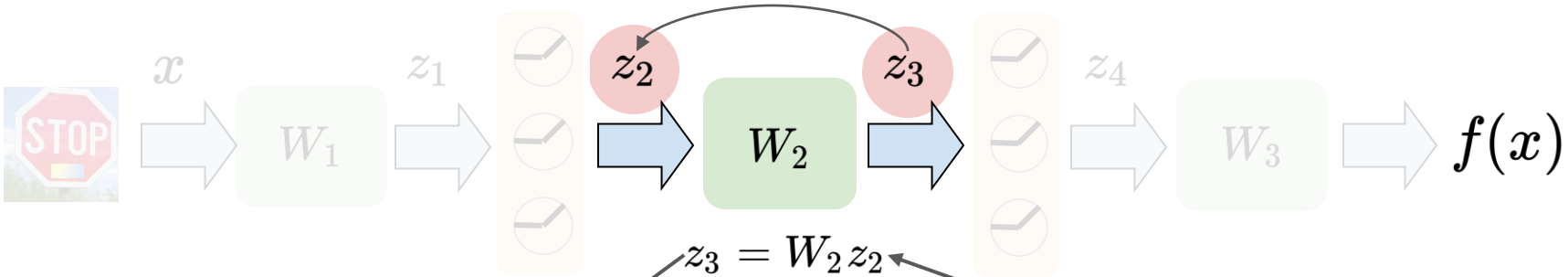
$$z_3 = W_2 z_2$$

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

Illustration: Linear bound propagation process







$$z_3 = W_2 z_2$$

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

Inequality updated after propagation

Illustration: Linear bound propagation process

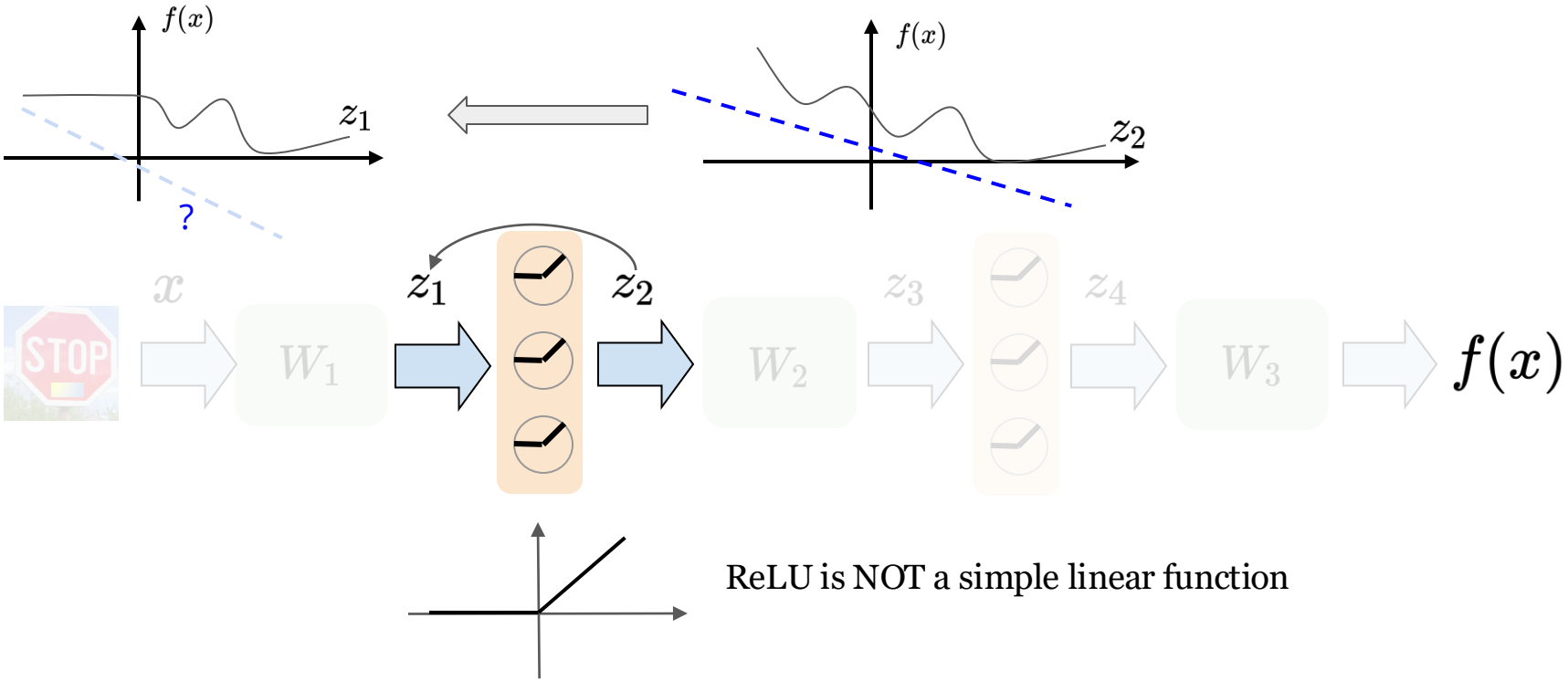
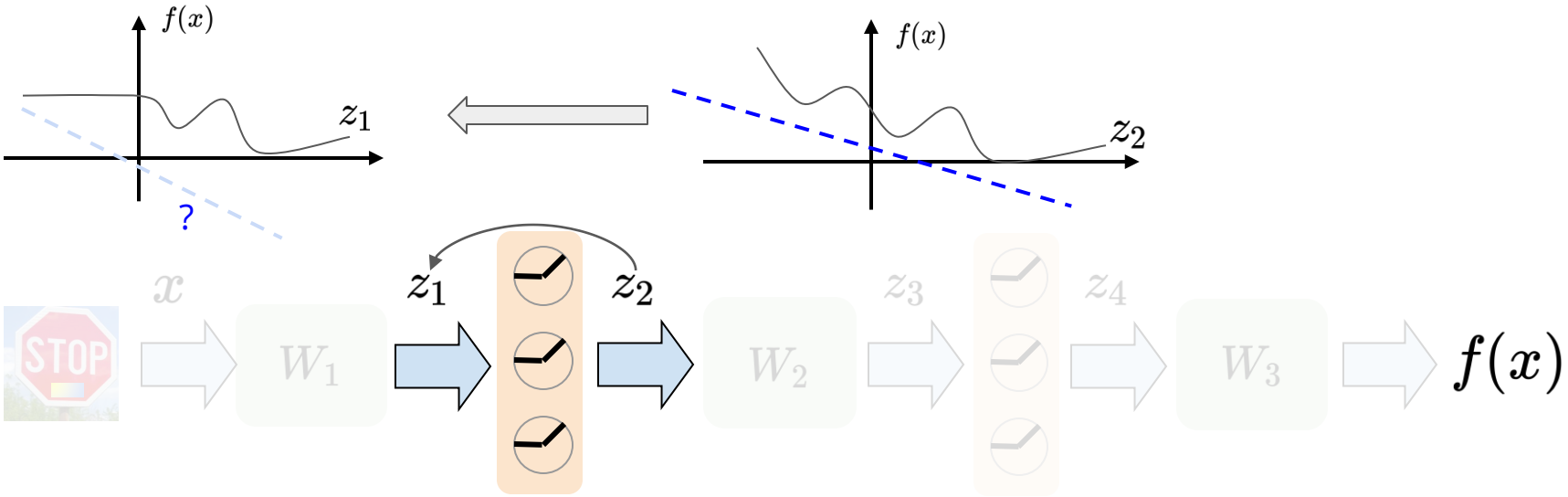
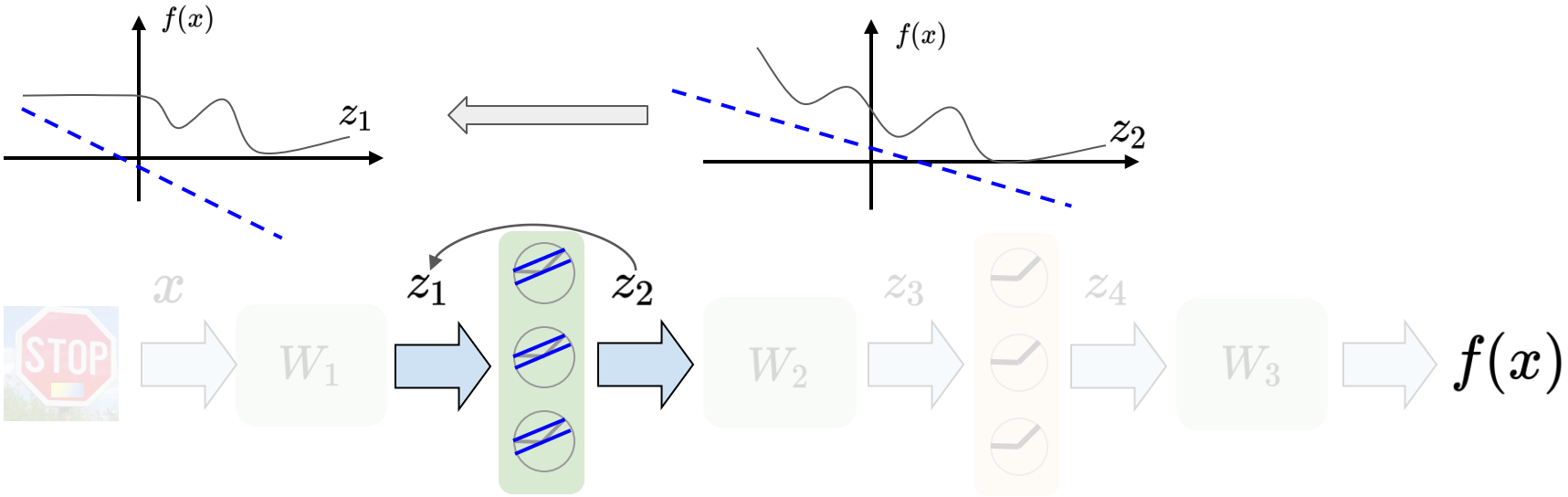


Illustration: Linear bound propagation process



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

Illustration: Linear bound propagation process

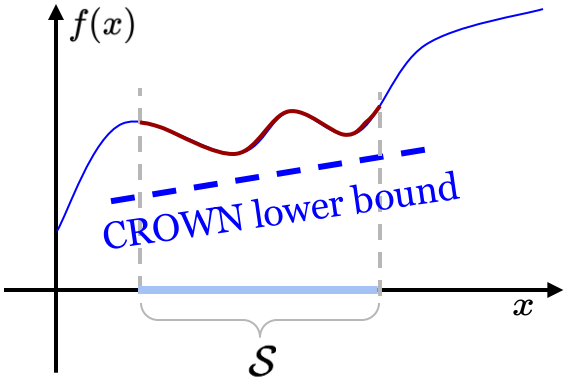


Theorem (informal): we can efficiently find D, b' such that:

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

[Z*W*CHD NeurIPS 2018]

Use Linear Bounds to Prove Robustness



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

$x_1 \in \mathcal{S}$

$x_2 \in \mathcal{S}$

$x_3 \in \mathcal{S}$

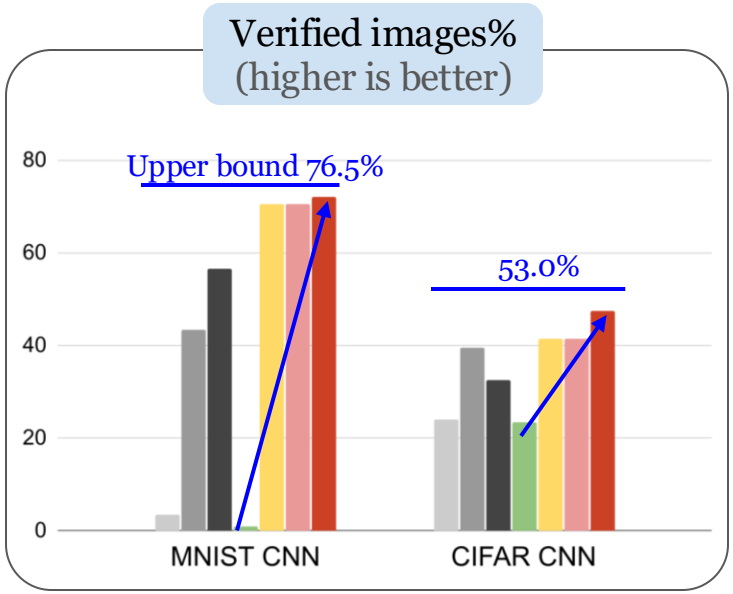
...

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

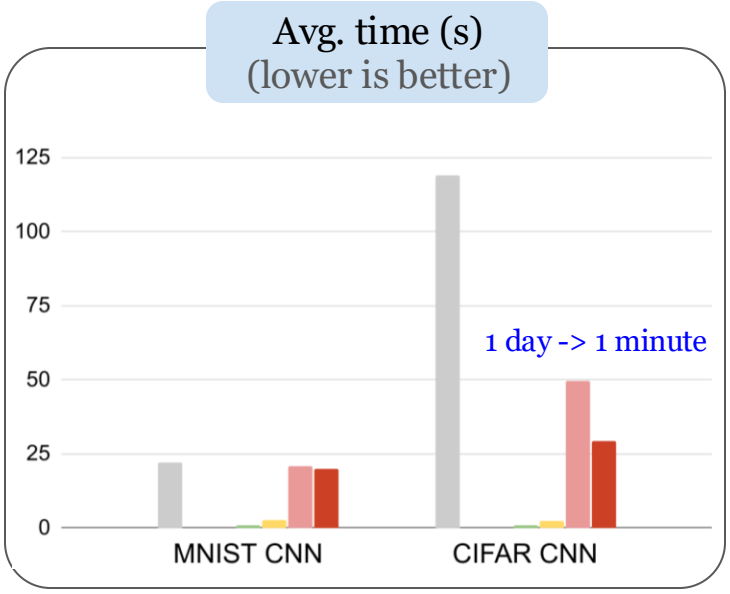
Benchmarks: CROWN-family bound propagation algorithms

- 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

Pixel perturbation magnitude: 0.3 for MNIST, 2/255 for CIFAR



Model size: ~5k neurons



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

Key enablers: specialized bound propagation solver + GPU acceleration + BaB

MILP/LP vs Bound Propagation

Bound propagation:

- Scalable and fast propagation
- GPU friendly
- Incomplete verification (can be made complete with partitioning the input domain)
- Bounds are looser compared to LP; much looser compared to MILP

MILP/LP:

- Tighter solution
- Does no scale (MILP ~10k neurons, LP ~100k neurons)
- Much slower; cannot utilize GPU

Example branching heuristic

$S = \{x_1 \in [-1, 1], x_2 \in [-1, 1]\}$ =>

$S_1 = \{x_1 \in [-1, 0], x_2 \in [-1, 1]\}$, $S_2 = \{x_2 \in [0, 1], x_2 \in [-1, 1]\}$

OR

$S_1 = \{x_1 \in [-1, 1]\ x_2 \in [-1, 0]\}$, $S_2 = \{x_2 \in [-1, 1], x_2 \in [0, 1]\}$

We can estimate the impact on lower bound given changes on x_1 and x_2

Given the CROWN linear bound $y >= a_1 x_1 + a_2 x_2 + c$, we branch on dimension i where $|a_i|$ is largest.

`auto_LiRPA`: Verification Library for General Computation Graphs

Colab Demo:

http://PaperCode.cc/AutoLiRPA-Demo

The auto_LiRPA library on GitHub:

http://PaperCode.cc/AutoLiRPA