Summary Slides

Safety and Robustness Verification of Neural Networks

Presented by
Zhong Yuyi

Introduction

- Artificial neural networks are one of the most investigated and well-established Machine Learning techniques
- Application in a wide range of research and engineering domains
- Key challenge: to ensure the NNs behave as intended
- Two goals: safety and robustness

Safety Property

- Safety **Property**: given every possible input value, its output is guaranteed to range within specific bounds
  - *The first time in which formal verification of a ML technique is investigated*
  - *Pulina L, etc. An abstraction-refinement approach to verification of artificial neural networks. CAV 2010.*
- Application: neural network to control PUMA 500 industrial manipulator
  - *Build an MLP predicting the final position of the end-effector*
  - *Never emit a prediction that exceeds the safe region*
  - *Safety verification*

Robustness Property

- Recent research showing that even highly accurate NNs are vulnerable to adversarial examples.
- Adversarial examples: typically obtained by <u>slightly perturbing</u>
  - an input that is correctly classified by the network
  - such that the network misclassifies the perturbed input.
- Various perturbations
  - Brightened pixels are marked in yellow and darkened pixels are marked in purple
  - FGSM: adding to it a particular noise vector multiplied by a small number ϵ
  - Brightening: changing all pixels above the threshold $1 - \delta$ to the brightest possible value

Attack	Original	Perturbed	Diff
FGSM [12], $\epsilon = 0.3$			
Brightening, $\delta = 0.085$			

Robustness Property

- Adversarial examples can be especially problematic for neural networks
- To mitigate these issues, reasoning about neural network robustness, and in particular on local robustness.
- Local robustness (or robustness, for short): all samples in the neighbourhood of a given input are classified with the same label
- Consider the image of the digit 8 and to prove that no matter how we brighten the value of pixels with intensity above $1 - \delta$, the network will still classify the image as 8

Brightening, $\delta = 0.085$			

Robustness Verification

- Huang X, etc. Safety verification of deep neural networks. CAV 2017.
  - Layer by layer analysis based on SMT
- Gehr T, etc. Ai2: Safety and robustness certification of neural networks with abstract interpretation. IEEE 2018.
  - Abstract transformer to capture the behaviour, leverage existing generic abstract domain, zonotope
- Singh G, etc. An abstract domain for certifying neural networks. POPL 2019.
  - A novel abstract domain specifically tailored to the setting of neural networks

Development on Abstract Interpretation

- Pulina L, etc. An abstraction-refinement approach to verification of artificial neural networks. CAV **2010**.
  - Application: network with only 6 neurons
  - Abstract domain: intervals
- Gehr T, etc. Ai2: Safety and robustness certification of neural networks with abstract interpretation. IEEE **2018**.
  - Convolutional neural network; zonotope
- Singh G, etc. An abstract domain for certifying neural networks. POPL **2019**.
  - Even larger network; polyhedra with interval
- More Scalable and use more precise abstract domain (more precise result)

Interval Abstract Domain

Pulina L, etc. An abstraction-refinement approach to verification of artificial neural networks. CAV 2010.

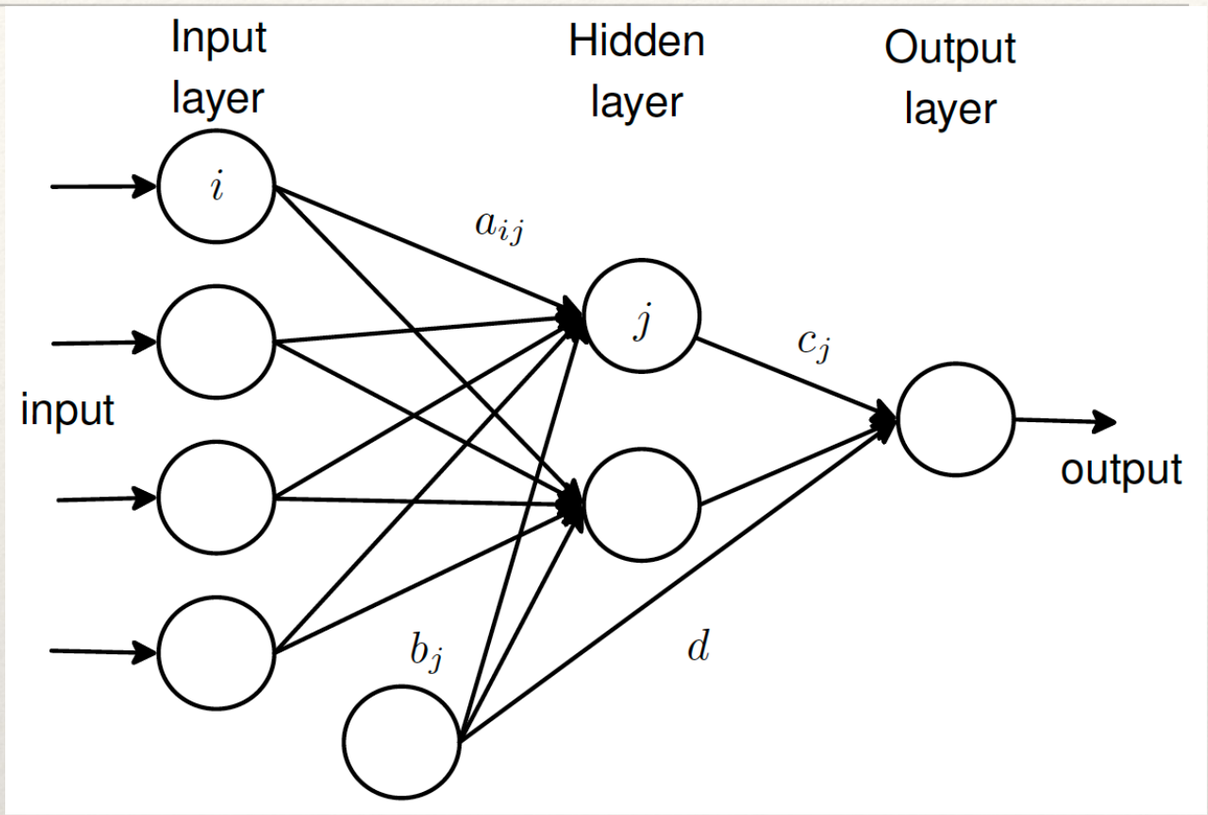
- Verifying safety on MLPs: given every possible input value, its output is guaranteed to range within specific bounds
- MLP: a specific kind of feed-forward neural network known as Multi-Layer Perceptron
- Abstraction yields over-approximations of concrete networks
- Counterexample triggered abstract refinement (CETAR):
  - Control the "coarseness" of the abstraction through a numeric parameter p
  - Start with coarse abstraction
  - CEX: Modify p to refine the abstraction and retry the verification

Contributions

1. The abstraction of MLPs
2. A strategy for automating MLP repair
   - modifications of the MLP synthesis attempting to correct its misbehaviours
   - strong empirical evidence in support of the fact that adding spurious counterexamples to the training set yields MLPs which are safer than the original ones.

MLP

- Multi-Layer Perceptron (MLP)
  - neurons are represented by circles
  - connections are represented by arrows
  - the i-th input value is denoted by x_i
  - hidden neuron j
  - a_{ij} the weight of the connection
  - b_j, d are constant, bias of neuron
  - Input received by neuron j



- Verifying on MLP v, given n inputs (eg: 6 joint angles) and a single output (eg: coordinate in meters)
- Given any set $X \subseteq R$, abstraction is $\alpha(X) = [min\{X\}, max\{X\}]$
- Concretization $\gamma([a, b]) = \{x \mid x \in [a, b]\}$
- Input set $X = \{\bar{x}_1, \ldots, \bar{x}_k\}$, each $\bar{x}_i$ has 6 components, $\alpha(X)$ is the vector has n intervals.

$$\alpha(X) = \langle [\min_{1 \le j \le k} \{x_{1j}\}, \max_{1 \le j \le k} \{x_{1j}\}], \ldots, [\min_{1 \le i \le k} \{x_{nj}\}, \max_{1 \le j \le k} \{x_{nj}\}] \rangle$$

Verification on MLP

- Input received by neuron j

$$y_j = \sum_{i=1}^{n} a_{ji} x_i + b_j$$

- After activation function: logistic func

$$\sigma(r) = \frac{1}{1 + \exp(-r)}, \; r \in \mathbb{R}$$

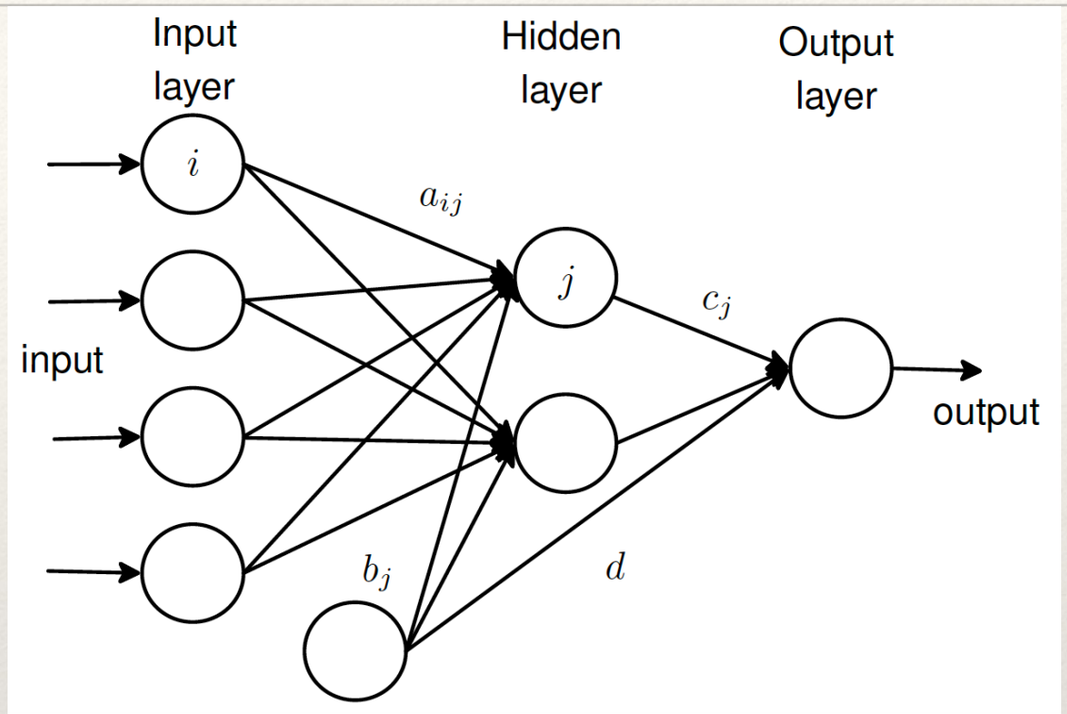
- The output of MLP:

$$\nu(\underline{x}) = \sum_{j=1}^{m} c_j \sigma(y_j) + d$$

- Abstract Network

$$\tilde{\nu}_p([\underline{x}]) = \sum_{j=1}^{m} c_j \tilde{\sigma}_p(\tilde{y}_j([\underline{x}])) + d$$

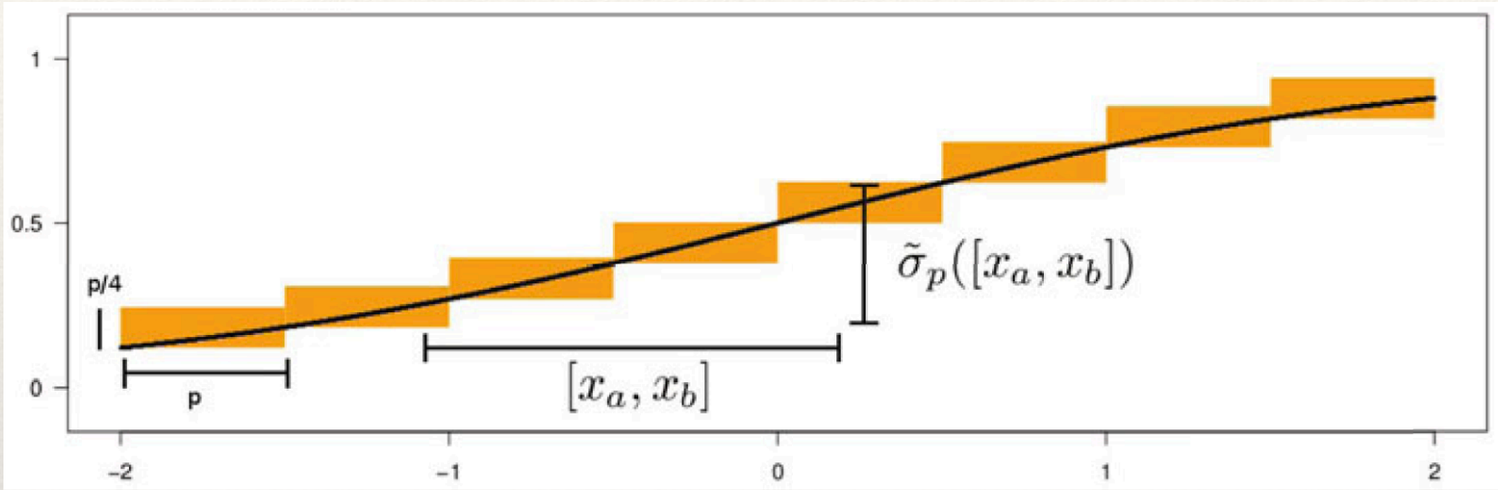
- Abstract $\tilde{y}_j$:
  - Products and sums on intervals



- Abstract activation function $\tilde{\sigma}_p$
- Parametrised by $p \in R^+$
- Obtained by considering the maximum increment of σ over intervals of length p
- σ is a monotonically increasing function, and its <u>first derivative</u> is <u>maximum</u> in the origin
- Origin has slope 1/4

Verification on MLP

- The max slope (increment rate) is 1/4 $\forall x \in \mathbb{R} : 0 \leq \sigma(x+p) - \sigma(x) \leq \frac{p}{4}$



- Large values of p correspond to coarse-grained abstractions

$$\tilde{\sigma}_p([x_a, x_b]) = \begin{cases} [0, p/4] & \text{if } x_b \leq x_0 \\ [0, \sigma(\lfloor \frac{x_b}{p} \rfloor) + \frac{p}{4}] & \text{if } x_a \leq x_0 \text{ and } x_b < x_1 \\ [\sigma(\lfloor \frac{x_a}{p} \rfloor), \sigma(\lfloor \frac{x_b}{p} \rfloor) + \frac{p}{4}] & \text{if } x_0 < x_a \text{ and } x_b < x_1 \\ [\sigma(\lfloor \frac{x_a}{p} \rfloor), 1] & \text{if } x_0 < x_a \text{ and } x_1 \leq x_b \\ [1 - p/4, 1] & \text{if } x_a \geq x_1 \end{cases}$$

Verification on MLP

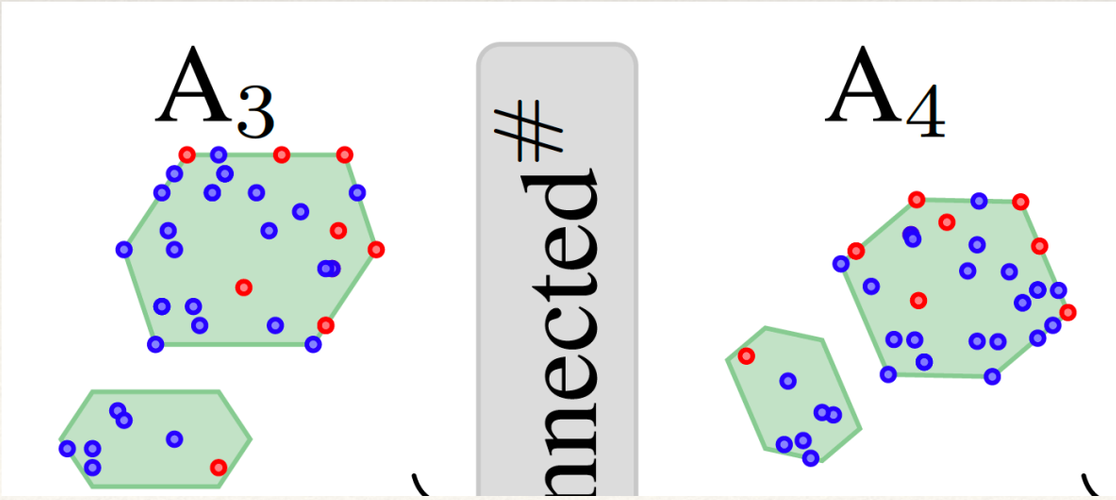
- Leverage the previous definitions to provide a complete abstraction-refinement algorithm to prove MLP safety.

```
NEVER(Δ, Π, [l, h], p, r)
  isSafe ← FALSE; isFeasible ← FALSE
  ν ← TRAIN(Δ, Π)
  repeat
    ν̃_p ← ABSTRACT(ν, p)
    s̃ ← NIL; isSafe ← CHECKSAFETY(ν̃_p, [l, h], s̃)
    if (not isSafe) then
      isFeasible ← CHECKFEASIBILITY(ν, s̃)
      if (not isFeasible) then
        p ← p / r
  until isSafe or (not isSafe and isFeasible)
  return isSafe
```

Zonotope Abstract Domain

Gehr T, etc. Ai2: Safety and robustness certification of neural networks with abstract interpretation. IEEE **2018**

- Verifying robustness on fully connected and convolutional neural network layers, as well as max pooling layers.
- Robustness requires that all samples in the neighbourhood of a given input are classified with the same label
- AI^2: Abstract Interpretation for AI.
- Abstract domain: logical formulas that capture certain shapes (e.g., zonotope, centrally symmetric polyhedra)

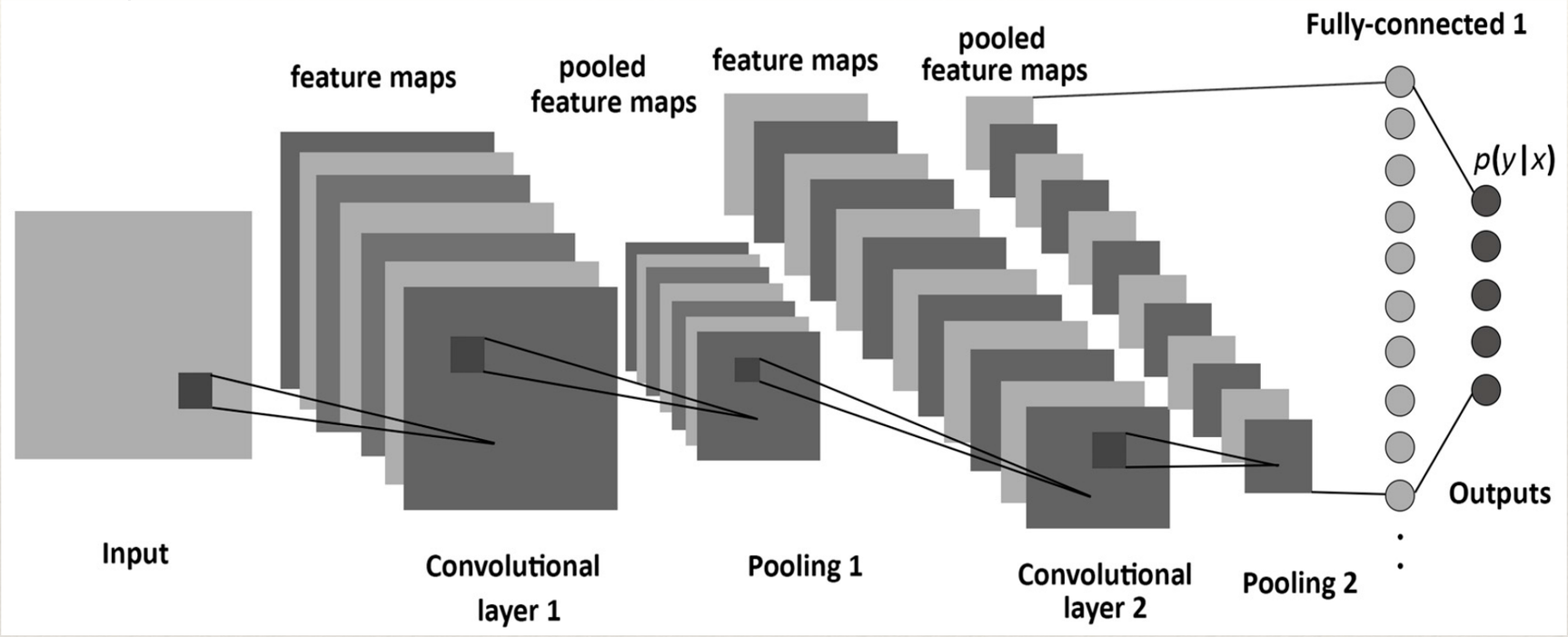


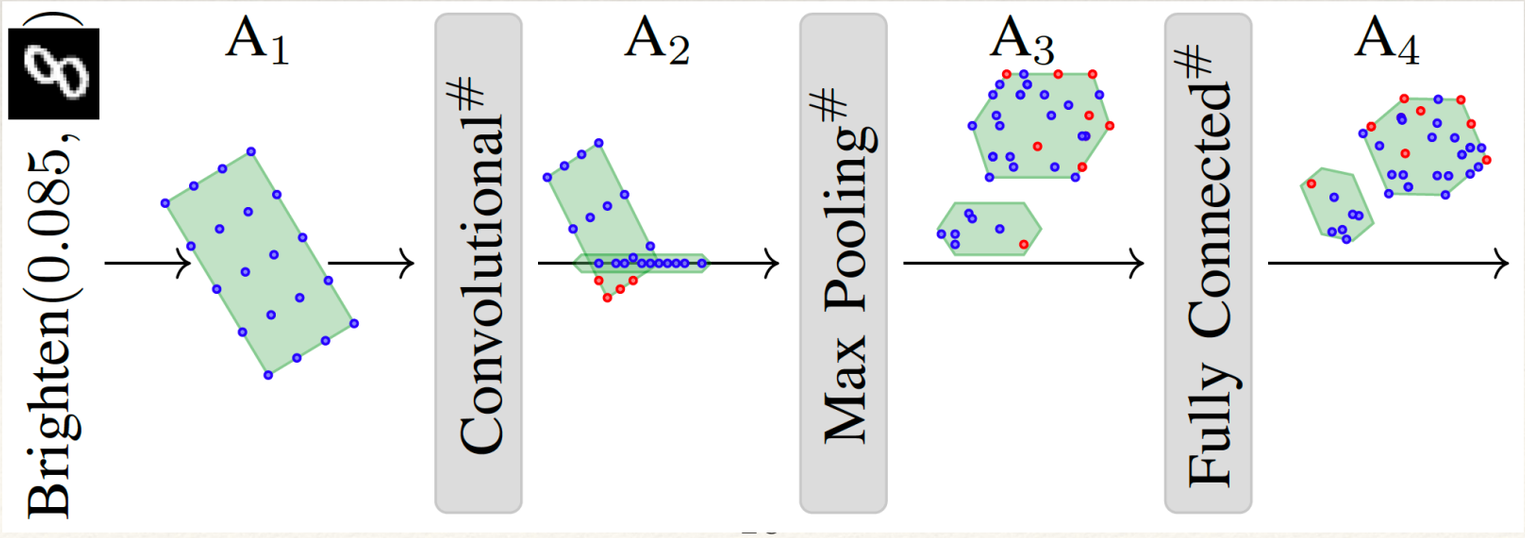
Contributions

- Using more precise abstract domain, therefore it is precise enough to prove useful specifications (e.g., robustness),
- AI2 can handle deep convolutional networks, which are beyond the reach of existing methods
- AI2 is significantly faster than existing analysers based on symbolic analysis, which often take hours to verify simple fully connected networks

Key to Success

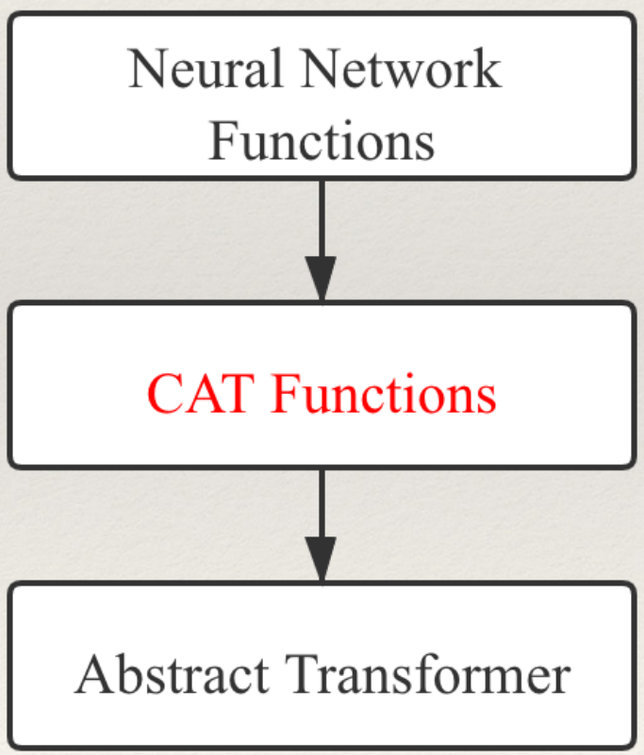
- Defining sound and precise, yet scalable abstract transformers for different layers in NNs.





Abstraction Flow

1. Transform the neural networks into a representation (*conditional affine transformations*, CAT) amenable to abstract interpretation.
2. Define the abstract transformer for CAT functions.

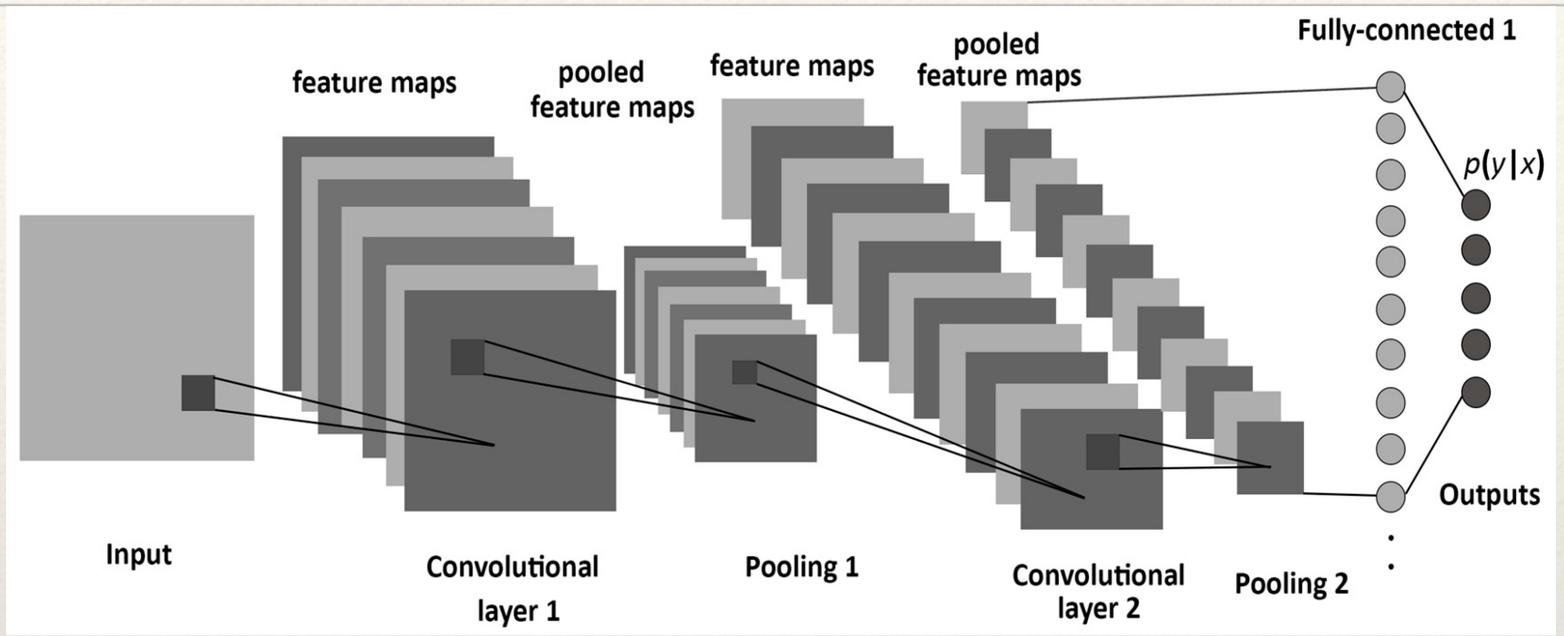


Conditional Affine Transformations

1. Normal affine transformation
2. Sequences of conditions, returns $f_i(\bar{x})$ for the first E_i satisfied by $\bar{x}$.
3. Any composition of CAT functions is a CAT function.
   $f'' \circ f' \Leftrightarrow f(\bar{x}) = f''(f'(\bar{x}))$
4. Consist of function $f : R^m \rightarrow R^n$

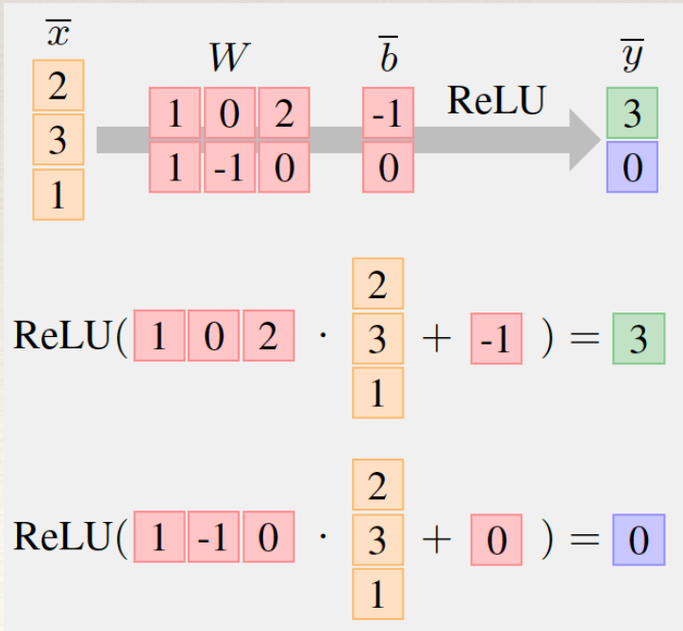
$$
\begin{array}{rcl}
f(\overline{x}) & ::= & W \cdot \overline{x} + \overline{b} \\
 & | & \textbf{case } E_1 : f_1(\overline{x}), \ldots, \textbf{case } E_k : f_k(\overline{x}) \\
 & | & f(f'(\overline{x})) \\
E & ::= & E \wedge E \mid x_i \geq x_j \mid x_i \geq 0 \mid x_i < 0
\end{array}
$$

Elements in Neural Networks



<u>Fully Connected</u>: *m outputs pass into n neurons, computation based on weights and bias*

- Activation function in the end $\mathrm{FC}_{W,\overline{b}}(\overline{x}) = \mathrm{ReLU}(W \cdot \overline{x} + \overline{b})$ $ReLU(x) = max(0,x)$



ReLU to CAT

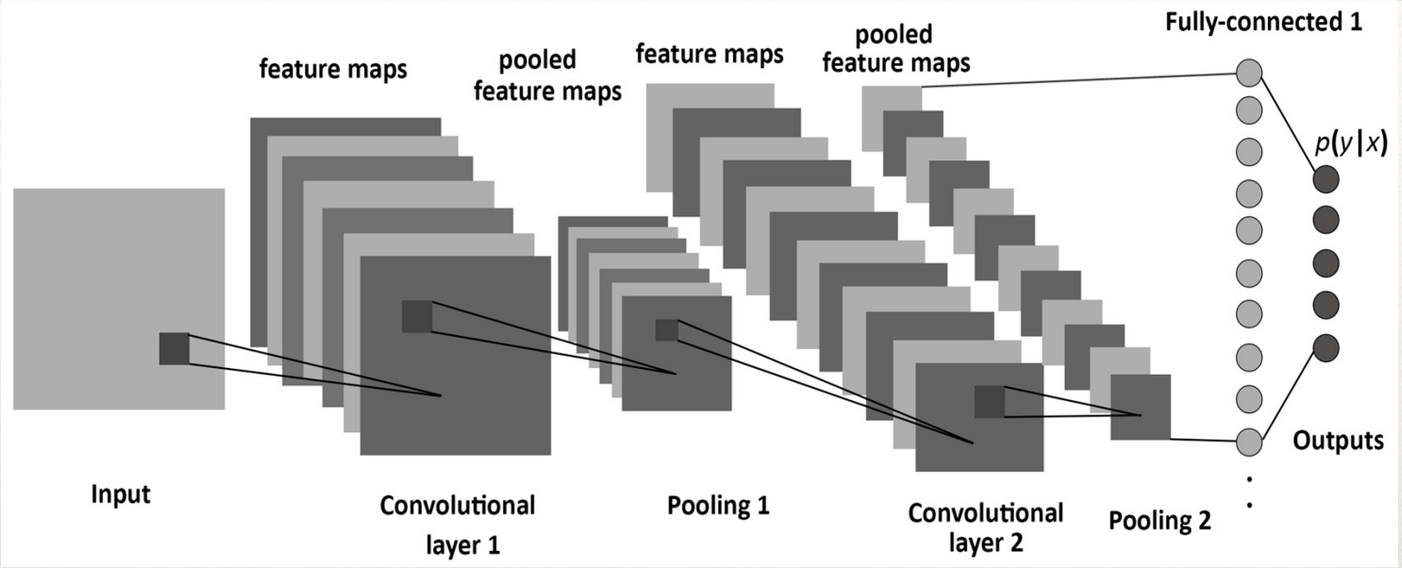
<u>Fully Connected</u>: $ReLU(W \cdot \bar{x} + \bar{b}) \Leftrightarrow ReLU \circ W \cdot \bar{x} + \bar{b}$

ReLU activation function as $ReLU = ReLU_n \circ \ldots \circ ReLU_1$ where $ReLU_i$ processes the i-th entry of the input x and is given by

$$\begin{aligned} \mathrm{ReLU}_i(\overline{x}) \;=\; & \textbf{case } (x_i \geq 0)\colon \overline{x}, \\ & \textbf{case } (x_i < 0)\colon I_{i \leftarrow 0} \cdot \overline{x}. \end{aligned}$$

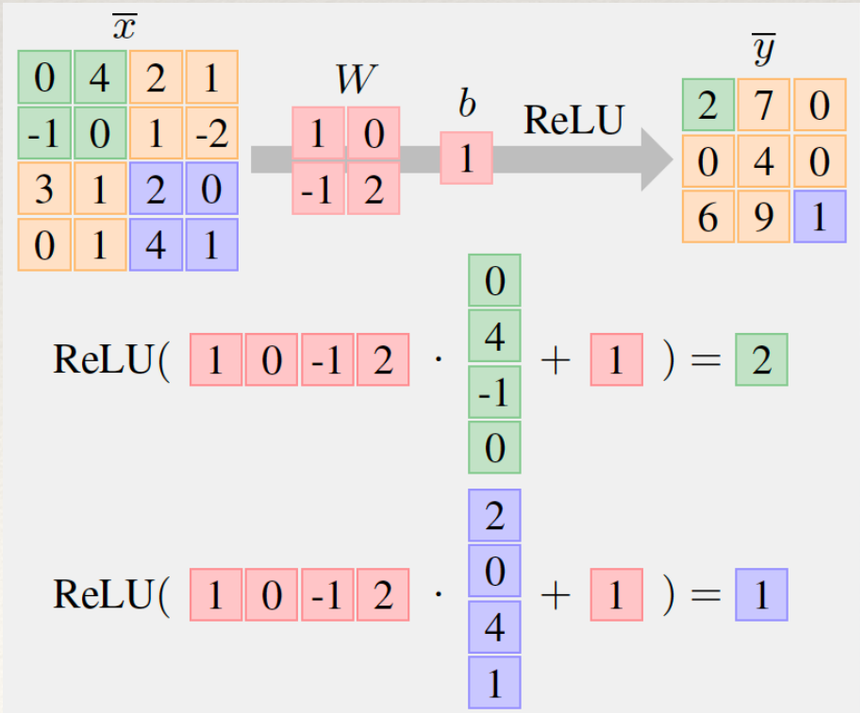
$I_{i \leftarrow 0}$ is the identity matrix with the i-th row replaced by zeros.

Elements in Neural Networks



<u>Convolutional</u>: *defined by filter, take a three-dimensional array and return 2-d array*

- Sliding the filter along the height and width of the input
- Each time read a slice of size $p \times q \times r$
- Compute dot product with W, add b
- Apply activation function



Convolutional to CAT

Reshape of Inputs: $\bar{x} \in R^{m \times n \times r} \rightarrow \bar{x}^v \in R^{m \cdot n \cdot r}$, first by depth, then column, finally row

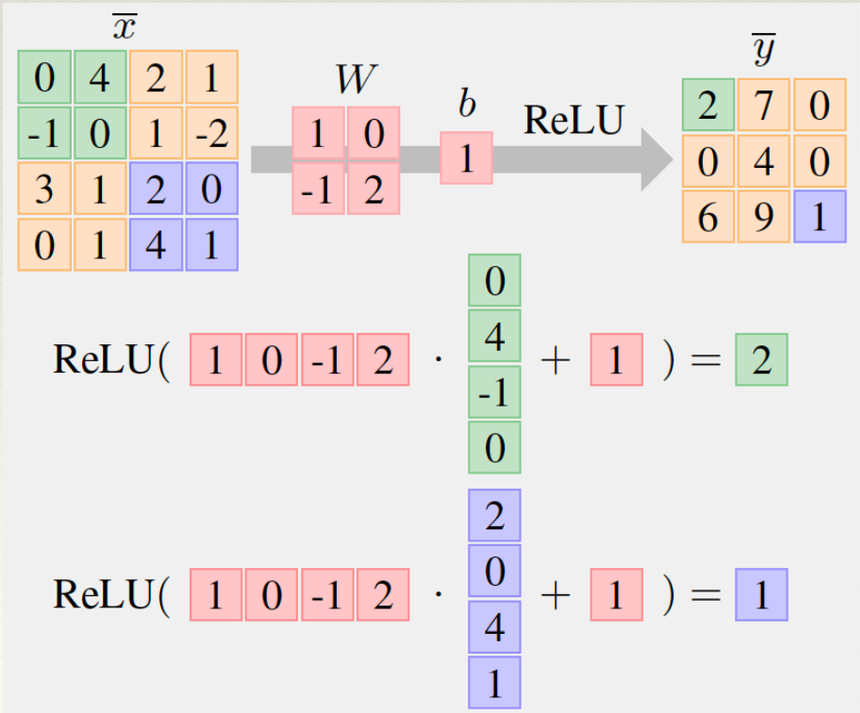
$$\overline{x}^v = (x_{1,1,1} \ldots x_{1,1,r}\ x_{1,2,1} \ldots x_{1,2,r} \ldots x_{m,n,1} \ldots x_{m,n,r})^T$$

Goal: $\bar{y}^v = W^F \cdot \bar{x}^v + \bar{b}^F$

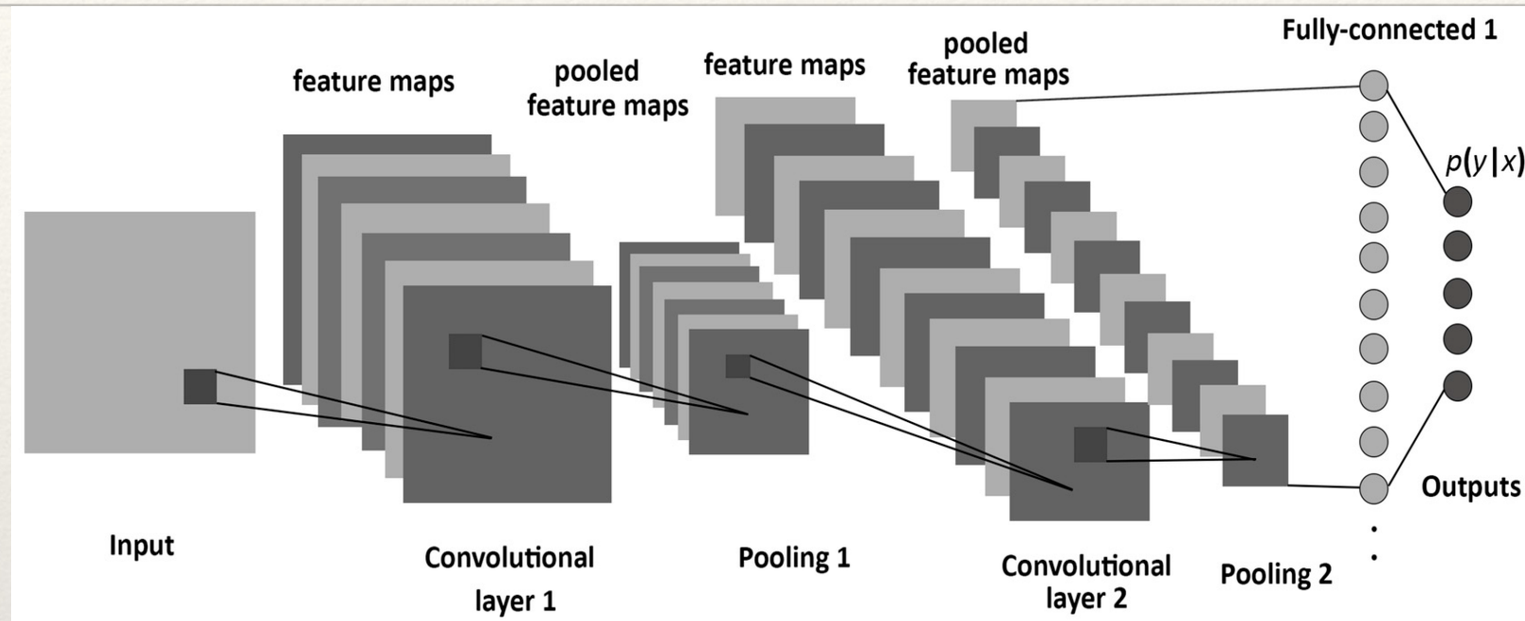
Example: $y_{1,1} = ReLU(W_{1,1} \cdot x_{1,1} + W_{1,2} \cdot x_{1,2} + W_{2,1} \cdot x_{2,1} + W_{2,2} \cdot x_{2,2} + b)$

- After reshaping, four entries $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}$ will be in $x_1^v, x_2^v, x_5^v, x_6^v$
- $y_{1,1}$ will be found in y_1^v
- First row in W^F such that its 1st, 2nd, 5th, and 6th entries are $W_{1,1}, W_{1,2}, W_{2,1}, W_{2,2}$
- First entry of $\bar{b}^F$ is b

$$W^F = \begin{pmatrix}
\mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{-1} & \mathbf{2}
\end{pmatrix} \qquad \bar{b}^F = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

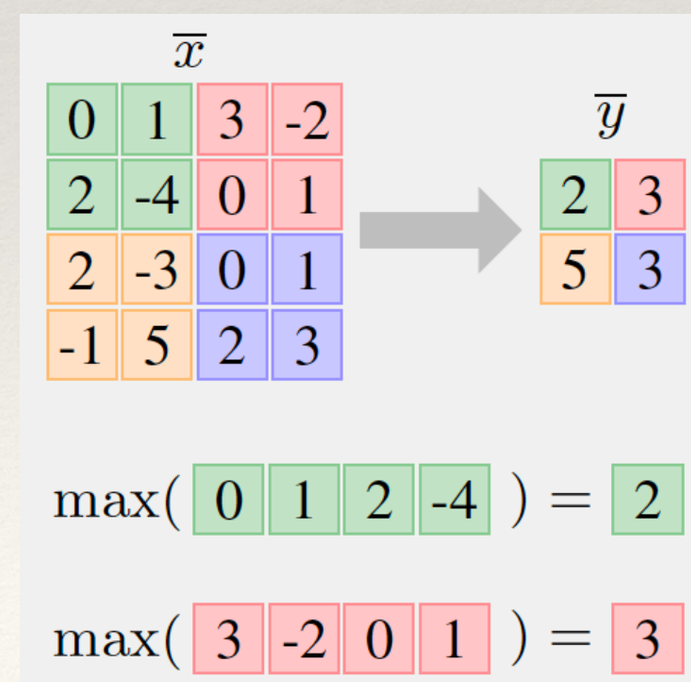


Elements in Neural Networks



Max Pooling: *take disjoint sub-rectangles and return the maximal value in the rectangle*

- Reduces the height m of $\bar{x}$ by a factor of p and
- the width n of $\bar{x}$ by a factor of q
- $R^{m \times n \times r} \rightarrow R^{\frac{m}{p} \times \frac{n}{q} \times r}$

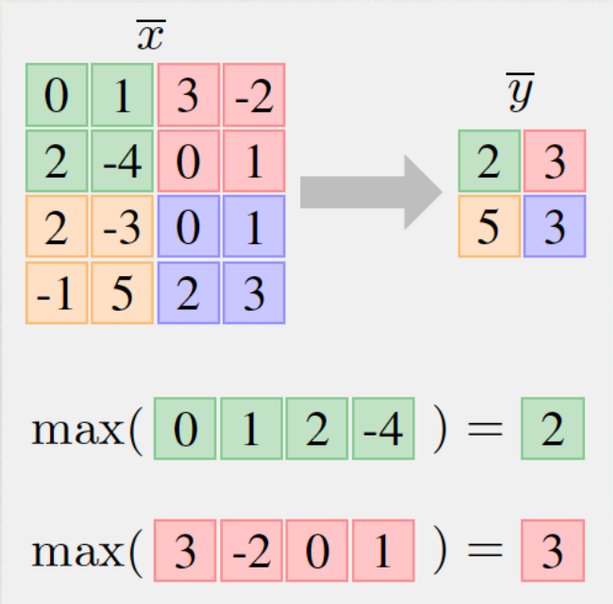


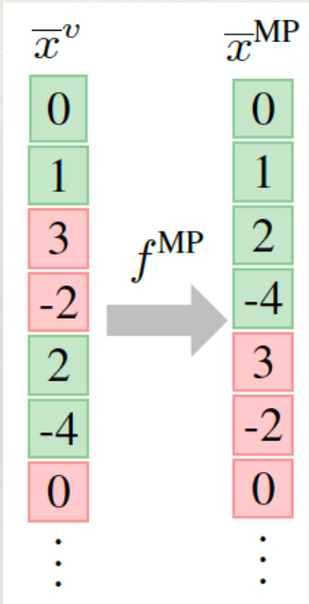
Max Pooling to CAT

Reshape of Inputs: $\bar{x} \in R^{m \times n \times r} \rightarrow \bar{x}^v \in R^{m \cdot n \cdot r}$, first by depth, then column, finally row

$$\overline{x}^v = (x_{1,1,1} \ldots x_{1,1,r}\ x_{1,2,1} \ldots x_{1,2,r} \ldots x_{m,n,1} \ldots x_{m,n,r})^T$$

- Use permutation matrix W^{MP} to make the values in the sub-rectangle continuous.





$$W^{\mathrm{MP}} = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

- Then for each rectangle, a matrix W^{f_i} is defined to pick up the maximal value (skipping the detail definition of W^{f_i})
- CAT function: $MaxPool(\bar{x}^v) = W^{f_1} \cdot W^{f_2} \cdot W^{f_3} \cdot W^{f_4} \cdot W^{MP} \cdot \bar{x}^v$

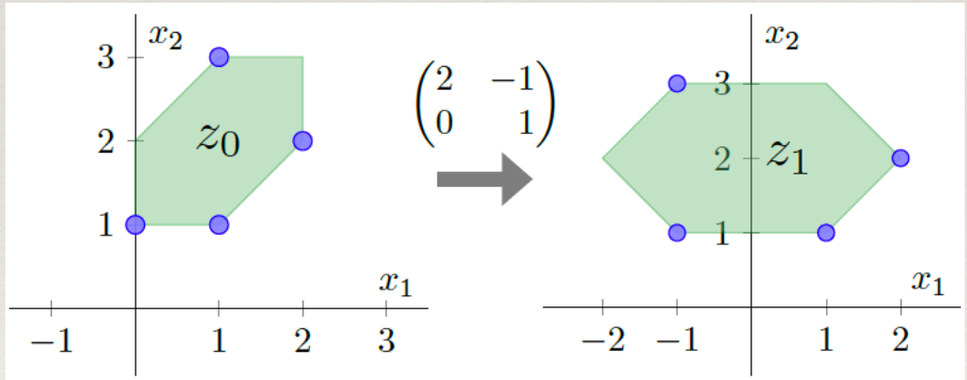
Abstraction Interpretation for CAT

- All layers are represented using CAT
- A neural network $N = f'_l \circ \ldots \circ f'_1$ is a composition of multiple CAT functions f'_i
- Only define abstract transformers for two kinds of CAT (affine func and case func)
- AF for NN $T^{\#}_N = T^{\#}_{f'_l} \circ \ldots \circ T^{\#}_{f'_1}$
- How to over-approximate CAT function with AI???

$$
\begin{array}{rcl}
f(\overline{x}) & ::= & W \cdot \overline{x} + \overline{b} \\
 & | & \textbf{case } E_1 : f_1(\overline{x}), \ldots, \textbf{case } E_k : f_k(\overline{x}) \\
 & | & f(f'(\overline{x})) \\
E & ::= & E \wedge E \mid x_i \geq x_j \mid x_i \geq 0 \mid x_i < 0
\end{array}
$$

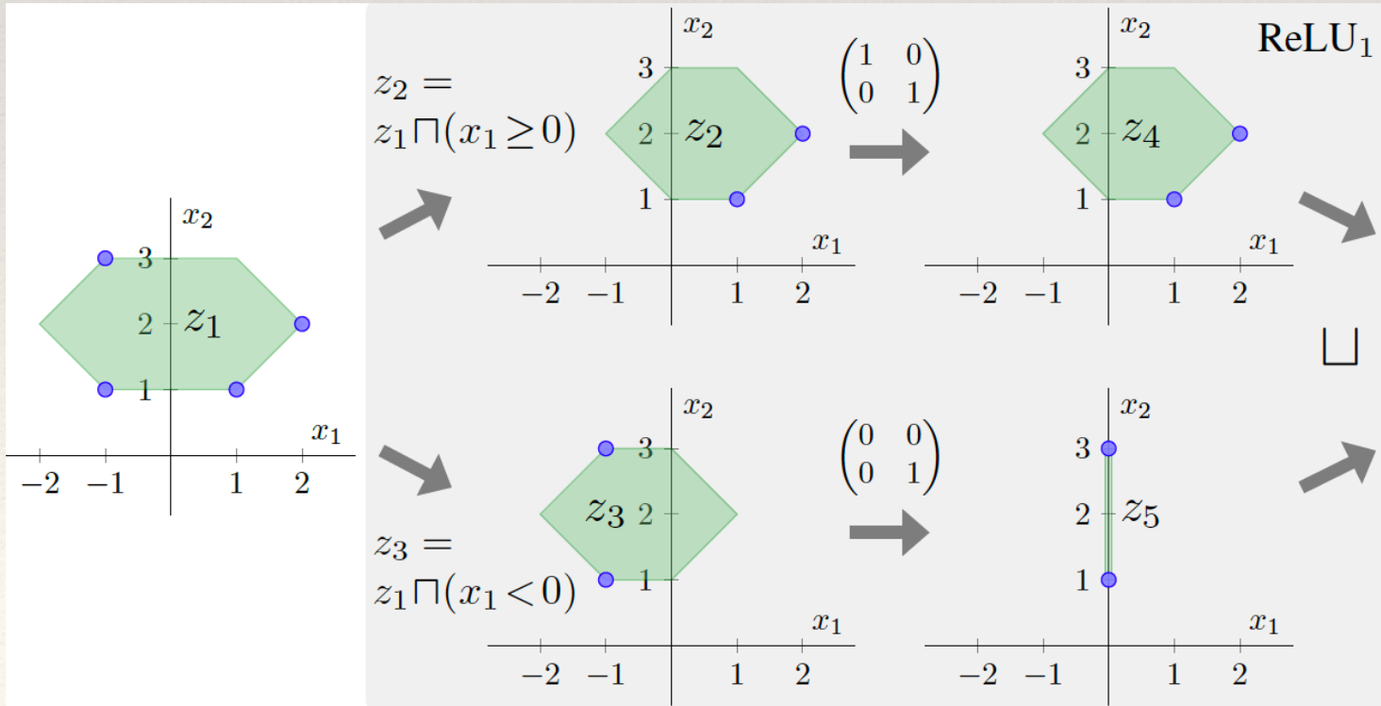
AI for Affine Function

- $f(\bar{x}) = W \cdot \bar{x} + \bar{b}$
- Affine operation for Zonotope.
- Example: single fully connected layer $f(\overline{x}) = \mathrm{ReLU}_2\left(\mathrm{ReLU}_1\left(\left(\begin{smallmatrix} 2 & -1 \\ 0 & 1 \end{smallmatrix}\right) \cdot \overline{x}\right)\right)$
- Input $X = \{(0,1), (1,1), (1,3), (2,2)\}$



AI for Case Function

- $f(\bar{x}) =$ case $E_1 : f_1(\bar{x}), \ldots,$ case $E_k : f_k(\bar{x})$
- First split the abstract element a into different cases (each defined by one of the expressions E_i), resulting in k abstract elements $a_1, \ldots, a_k$
- Compute the result $T^{\#}_{f_i}(a_i)$ for each a_i
- Unify the result in a single abstract element
- Example: $f(\overline{x}) = \mathrm{ReLU}_2\left(\mathrm{ReLU}_1\left(\left(\begin{smallmatrix} 2 & -1 \\ 0 & 1 \end{smallmatrix}\right) \cdot \overline{x}\right)\right)$
- $z_1 = \begin{pmatrix} 2 & -1 \\ 0 & 1 \end{pmatrix}^{\#} \cdot \bar{x}$, split into two zonotope
- $z_2 = z_1 \sqcap (x_1 \geq 0)$ and $z_3 = z_1 \sqcap (x_1 < 0)$
- The way to compute a meet between a zonotope and a linear constraint is omitted in the paper.

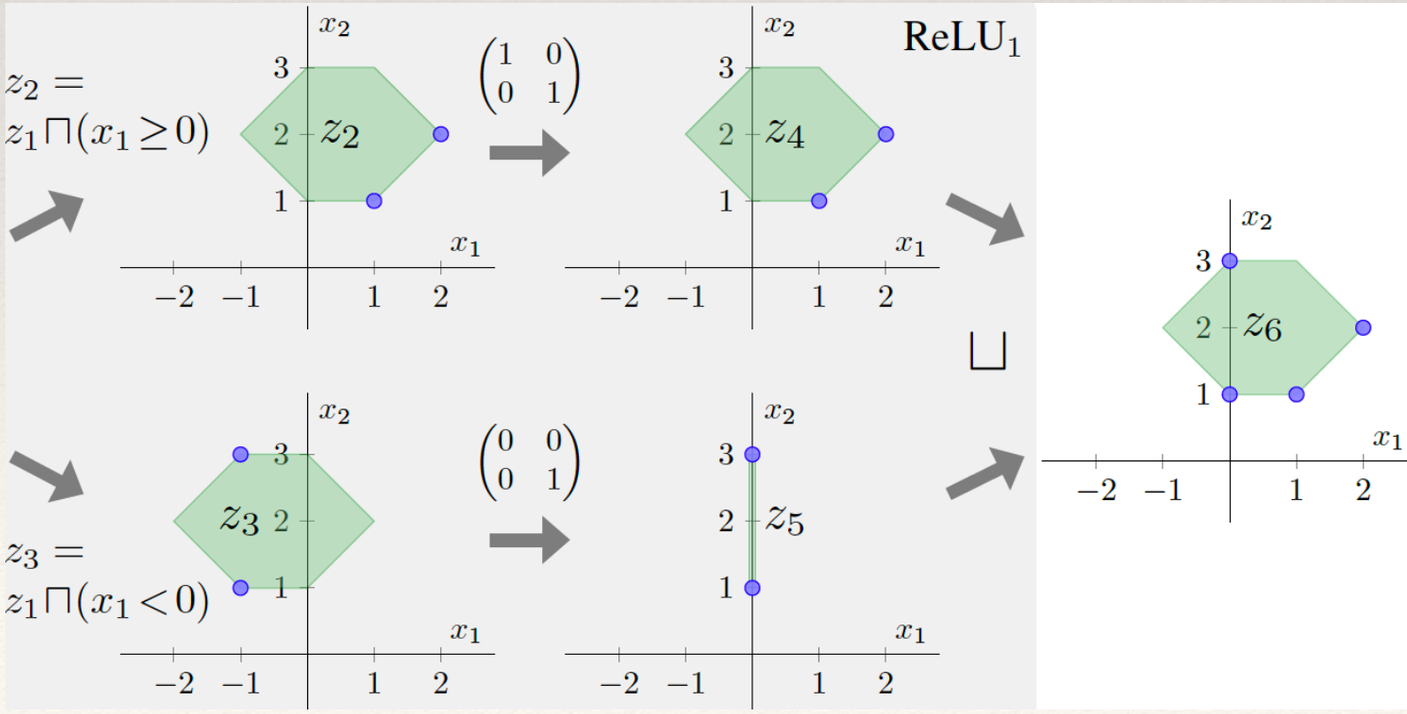


AI for Case Function

- Example: $f(\overline{x}) = \mathrm{ReLU}_2\left(\mathrm{ReLU}_1\left(\left(\begin{smallmatrix} 2 & -1 \\ 0 & 1 \end{smallmatrix}\right) \cdot \overline{x}\right)\right)$
- $z_2 = z_1 \sqcap (x_1 \geq 0)$ and $z_3 = z_1 \sqcap (x_1 < 0)$
- the two cases of $ReLU_1$ are processed separately

$$\begin{aligned} \mathrm{ReLU}_i(\overline{x}) \;=\; & \textbf{case } (x_i \geq 0)\colon \overline{x}, \\ & \textbf{case } (x_i < 0)\colon I_{i \leftarrow 0} \cdot \overline{x}. \end{aligned}$$

- Resulting z_4, z_5, these are then joined to obtain a single zonotope z_6
- Then z_6 is passed to $ReLU_2$ to get final result

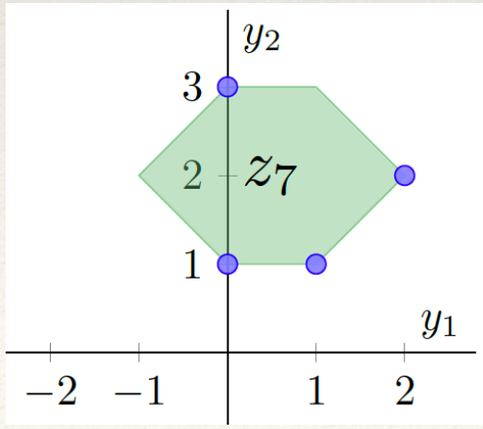


Robustness Proofs using AI

- This is a property (X, C_L) where X is a robustness region and C_L contains the outputs that describe the same label L:

$$C_L = \left\{ \overline{y} \in \mathbb{R}^n \;\middle|\; \underset{i \in \{1,\ldots,n\}}{\arg\max}\,(y_i) = L \right\}$$

- Typically, we will want to check that there is some label L for which (X, C_L) holds.
- Suppose the abstract result is a, the following condition is sufficient to prove that N satisfies (X, C_L): $a \sqcap \neg C_L = \bot$
- In the example, we want to check that all inputs are classified as 2. C_L is presented as $y_2 \geq y_1$, successfully verified
- Fail to prove $y_1 \geq 0$



Evaluation of AI2

- Datasets: MNIST and CIFAR-10
- MNIST: 60 000 grayscale images of handwritten digits
- CIFAR consists of 60 000 coloured photographs with 3 colour channels, whose resolution is 32 32 pixels. The images are partitioned into 10 different classes
- AI2 is parameterized by the numerical abstract domain, therefore they can explore the precision-scalability trade-off by plugging in different abstract domains: Box, Zonotope and Polyhedra
- Example NN: LeNet convolutional architecture which consists of the following sequence of layers: 2 convolutional, 1 max pooling, 2 convolutional, 1 max pooling, and 3 fully connected layers.

Evaluation of AI2

- Adversarial examples: brightening the pixels of an image.

- Important findings:
  - More precise abstract domains enable AI2 to prove stronger robustness properties
  - More precise domains increase AI2's running time.
  - They observe that AI2 takes less than 10 seconds on average to verify a property with the Zonotope domain.
  - In Polyhedra domain, which takes several days to terminate for our smallest networks. (It is very important for us to see how the 2019 paper tackle this running time issue)

Application of AI2

- Evaluating and comparing neural network defences.
- Defence: an algorithm whose goal is to reduce the effectiveness of a certain attack against a specific network
- A natural metric to compare defences is the average "size" of the robustness region on some test set.
- Robustness region can be represented as a zonotope and can thus be precisely captured by AI2.

Polyhedra Abstract Domain

Singh G, etc. An abstract domain for certifying neural networks. POPL **2019**.

- Key technical insight: a new abstract domain
- Equipped with abstract transformers specifically tailored to the setting of neural networks
- Experimental results indicate that DeepPoly is more precise than prior work while scaling to large networks.
- Abstract domain is a combination of floating-point polyhedra with intervals

Adversarial Region

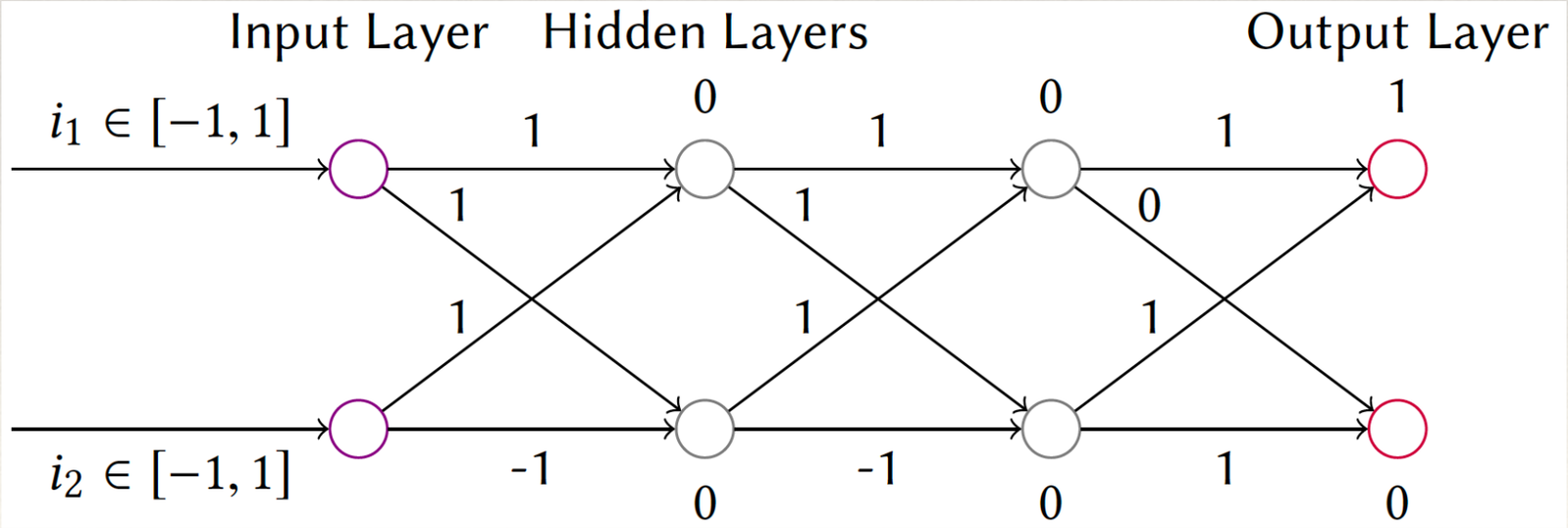
- the set $X \subseteq R^n$ of possible perturbations for a given input is called an <u>adversarial region</u>.
- They focus on verifying robustness for <u>adversarial regions</u> that can be represented using <u>a set of interval constraints</u> $X = \bigtimes_{i=1}^{m} [l_i, u_i]$ for $l_i, u_i \in \mathbb{R} \cup \{-\infty, +\infty\}$
- Prove robustness given the adversarial region

Presentation Flow

- Overview of the design on a small illustrative example
- Formal detail of abstract domain and transformer (in the real arithmetic)
- Adoption to floating point arithmetic
  - floating-point polyhedra with intervals
  - weights at each layer are usually detailed floating-point numbers

Illustrative Example

- A fully connected feedforward network with ReLU activation.
  - four layers: an input layer, two hidden layers, and an output layer with two neurons each.
  - learned bias for each neuron is shown above or below it.
  - ReLU, which outputs v, if v > 0, and 0 otherwise
  - the perturbation is such that it places both pixels in the range [−1, 1]
  - prove that one of the output neurons is always greater than the output at the other one, for any possible input of two pixels in the range [−1, 1]



Constraints Definition

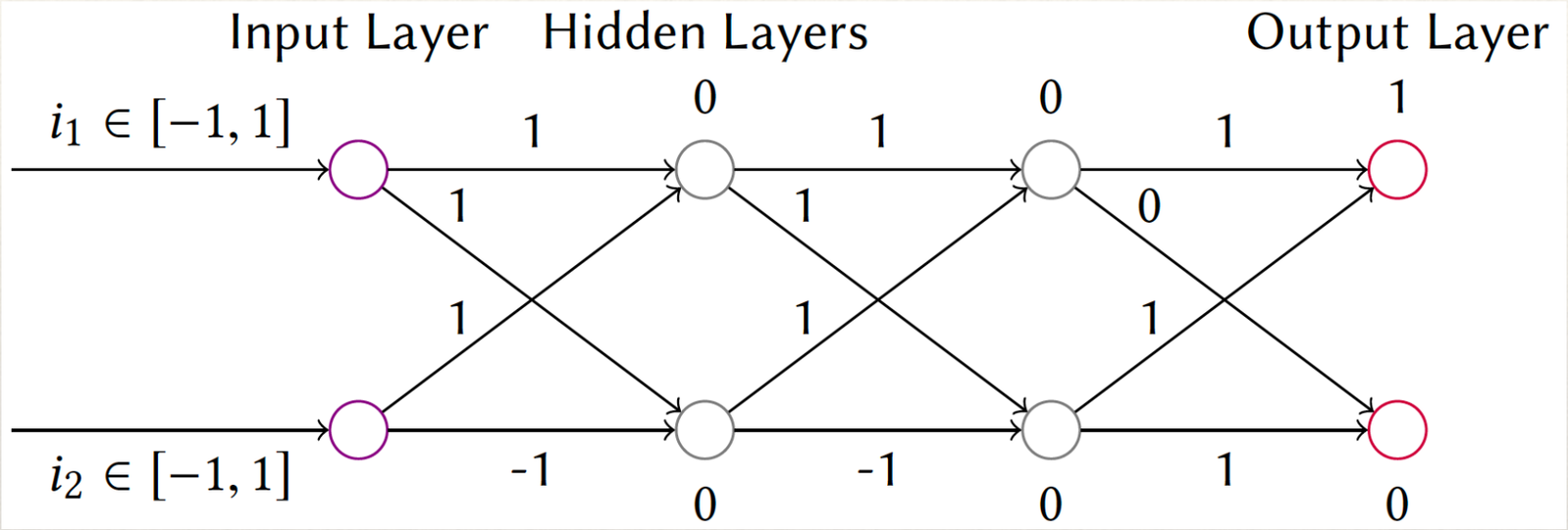
- The lower ($a_i^{\leq}$) and upper ($a_i^{\geq}$) relational polyhedral constraints associated with x_i have the form

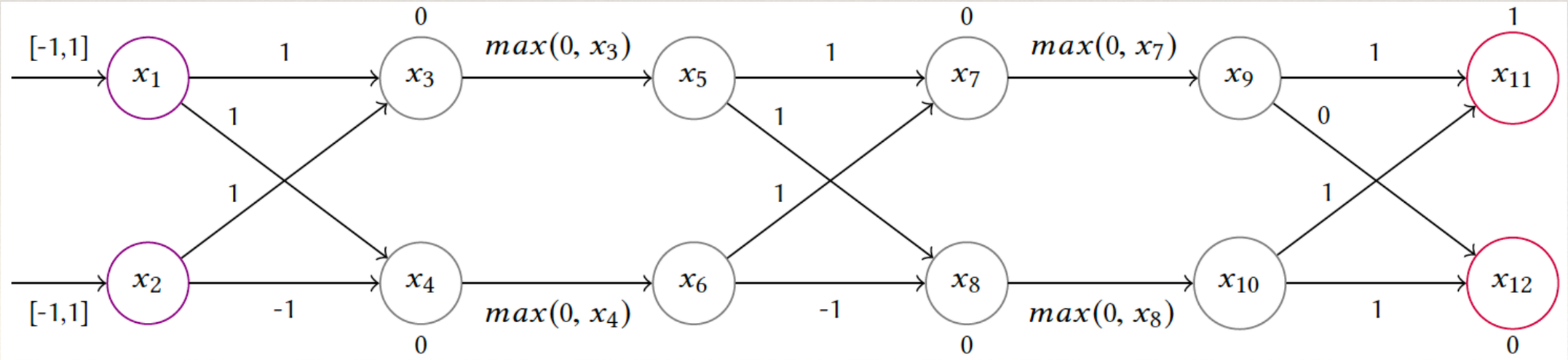
$$v + \sum_j w_j \cdot x_j \text{ where } v \in \mathbb{R} \cup \{-\infty, +\infty\}, w \in \mathbb{R}^n, \forall j \geq i.\, w_j = 0.$$

- Constraint only refers to variables before x_i in the network
- The concrete lower and upper bounds of x_i , we use $l_i, u_i \in R \cup \{-\infty, +\infty\}$

Rewrite Network

- Expand each neuron into two nodes: one for associated affine transformation, one for activation function (ReLU)





Propagation

1. Input $a_1^{\leq} = a_2^{\leq} = -1$, $a_1^{\geq} = a_2^{\geq} = 1$, $l_1 = l_2 = -1$, and $u_1 = u_2 = 1$
2. Use the affine transformation to updates the constraints for x_3, x_4 (the weight)

$$x_1 + x_2 \leq x_3 \leq x_1 + x_2$$

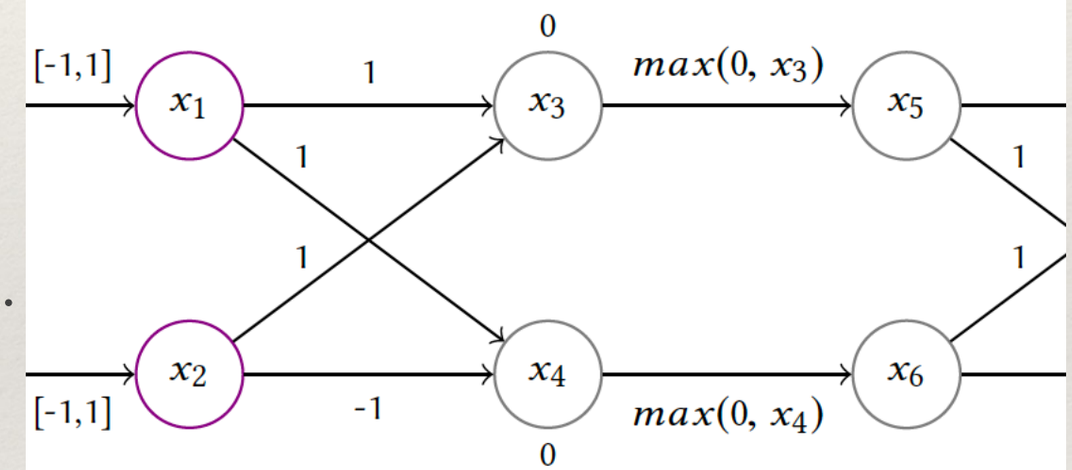
$$x_1 - x_2 \leq x_4 \leq x_1 - x_2$$

3. Uses the concrete bound for x_1, x_2 to compute $l_3 = l_4 = -2$ and $u_3 = u_4 = 2$
4. The transformer for the ReLU activation is applied. The output x_j of the ReLU activation on variable x_i is $x_j := max(0, x_i)$
   - If $u_i \leq 0$, then $0 \leq x_j \leq 0$, $l_j = u_j = 0$
   - If $l_i \geq 0$, then $x_i \leq x_j \leq x_i$, $l_j = l_i$, $u_j = u_i$
   - Transformer is exact at above cases

$\langle x_1 \geq -1,$ $x_1 \leq 1,$ $l_1 = -1,$ $u_1 = 1\rangle$

$\langle x_3 \geq x_1 + x_2,$ $x_3 \leq x_1 + x_2,$ $l_3 = -2,$ $u_3 = 2\rangle$

$\langle x_5 \geq 0,$ $x_5 \leq 0.5 \cdot x_3 + 1,$ $l_5 = 0,$ $u_5 = 2\rangle$



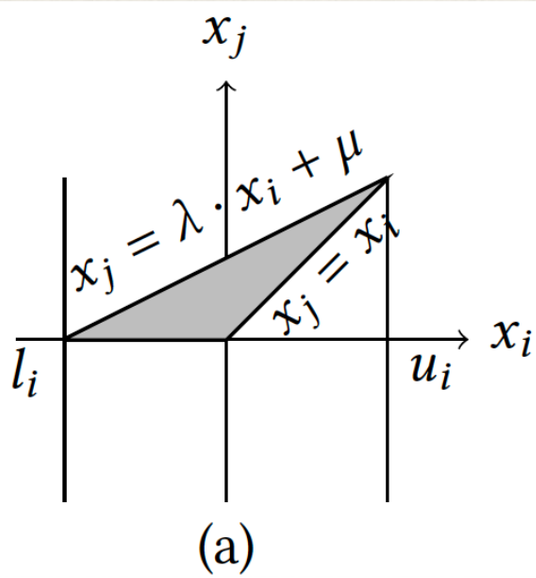
$\langle x_2 \geq -1,$ $x_2 \leq 1,$ $l_2 = -1,$ $u_2 = 1\rangle$

$\langle x_4 \geq x_1 - x_2,$ $x_4 \leq x_1 - x_2,$ $l_4 = -2,$ $u_4 = 2\rangle$

$\langle x_6 \geq 0,$ $x_6 \leq 0.5 \cdot x_4 + 1,$ $l_6 = 0,$ $u_6 = 2\rangle$

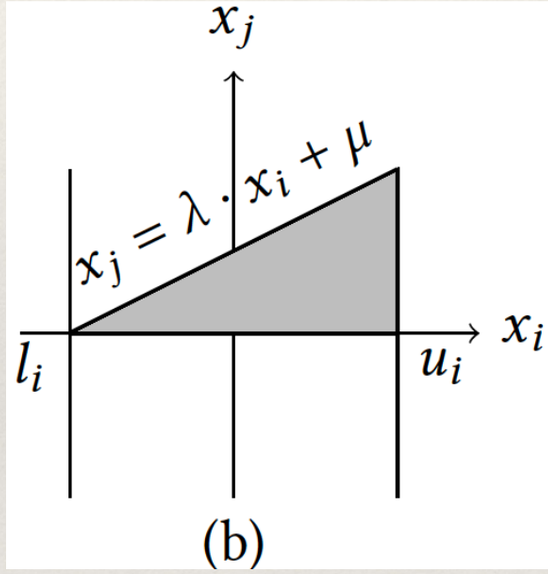
Approximation for Two Polyhedra

If $u_i > 0$ or $l_i < 0$ the result cannot be captured exactly by the abstraction and need to decide how to lose information.
Approximation in (a) contains two lower polyhedra constraints, which is NOT allowed, due to potential blowup in the number of constraints as the analysis proceeds



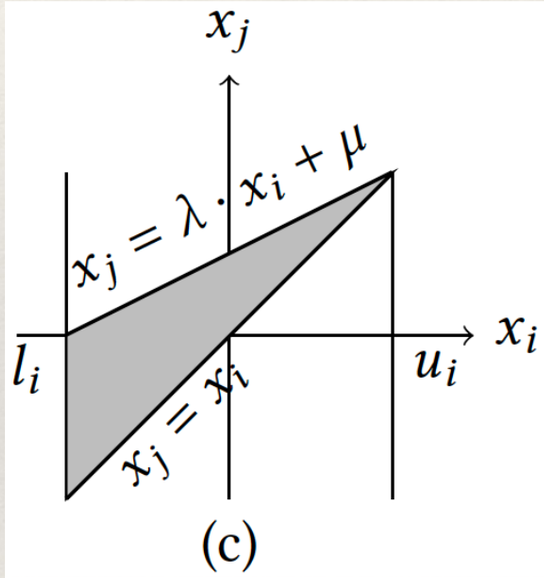
$x_i \leq x_j, 0 \leq x_j,$
$x_j \leq u_i(x_i - l_i)/(u_i - l_i).$
$l_j = 0, u_j = u_i.$

the convex approximation with the minimum area in the input-output plane



$0 \leq x_j \leq u_i(x_i - l_i)/(u_i - l_i),$
$l_j = 0, u_j = u_i.$

consider both and choose the one with the least area



$x_i \leq x_j \leq u_i(x_i - l_i)/(u_i - l_i),$
$l_j = l_i, u_j = u_i.$

less expressive than the Polyhedra domain

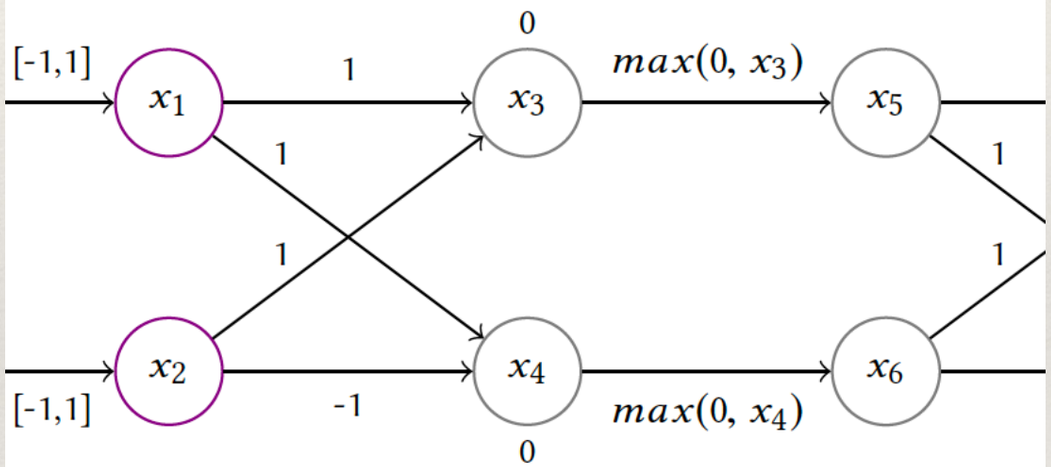
Propagation

In the example, they choose

$$0 \leq x_j \leq u_i(x_i - l_i)/(u_i - l_i),$$
$$l_j = 0, u_j = u_i.$$

Constraint for x_5: $x_5 \leq 2(x_3 + 2)/(2 + 2)$

$\langle x_1 \geq -1,$ $\langle x_3 \geq x_1 + x_2,$ $\langle x_5 \geq 0,$
$x_1 \leq 1,$ $x_3 \leq x_1 + x_2,$ $x_5 \leq 0.5 \cdot x_3 + 1,$
$l_1 = -1,$ $l_3 = -2,$ $l_5 = 0,$
$u_1 = 1\rangle$ $u_3 = 2\rangle$ $u_5 = 2\rangle$



$\langle x_2 \geq -1,$ $\langle x_4 \geq x_1 - x_2,$ $\langle x_6 \geq 0,$
$x_2 \leq 1,$ $x_4 \leq x_1 - x_2,$ $x_6 \leq 0.5 \cdot x_4 + 1,$
$l_2 = -1,$ $l_4 = -2,$ $l_6 = 0,$
$u_2 = 1\rangle$ $u_4 = 2\rangle$ $u_6 = 2\rangle$

Propagation

Again, abstract affine transformation to updates the constraints for x_7, x_8 (the weight)

$$x_5 + x_6 \leq x_7 \leq x_5 + x_6,$$
$$x_5 - x_6 \leq x_8 \leq x_5 - x_6.$$

How to compute the <u>concrete bound</u>??

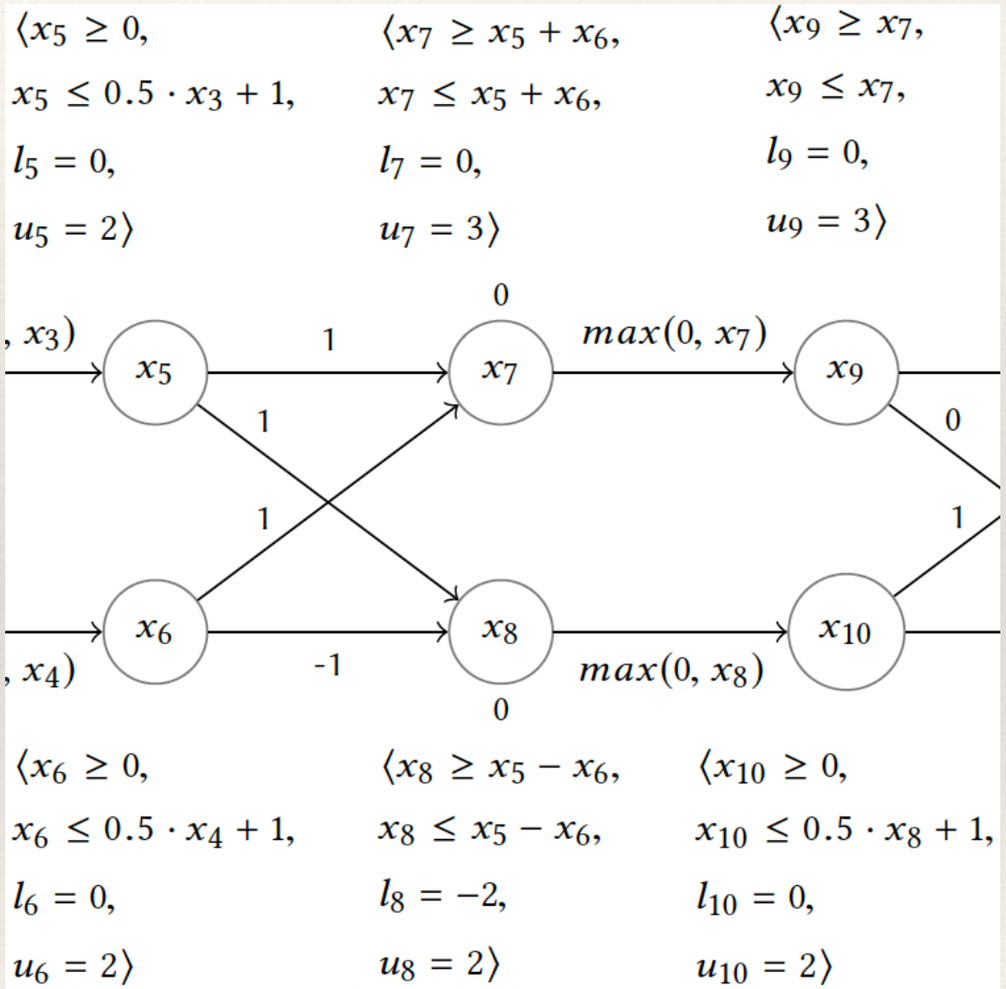
- Imprecise! Substituting the concrete bounds for x_5 and x_6
- Better bounds! Recursively substituting the polyhedral constraints until the <u>bounds only depend on the input variables</u> for which we then use their concrete bounds.

1. substitute the relational constraints for x_5, x_6 from x_3, x_4

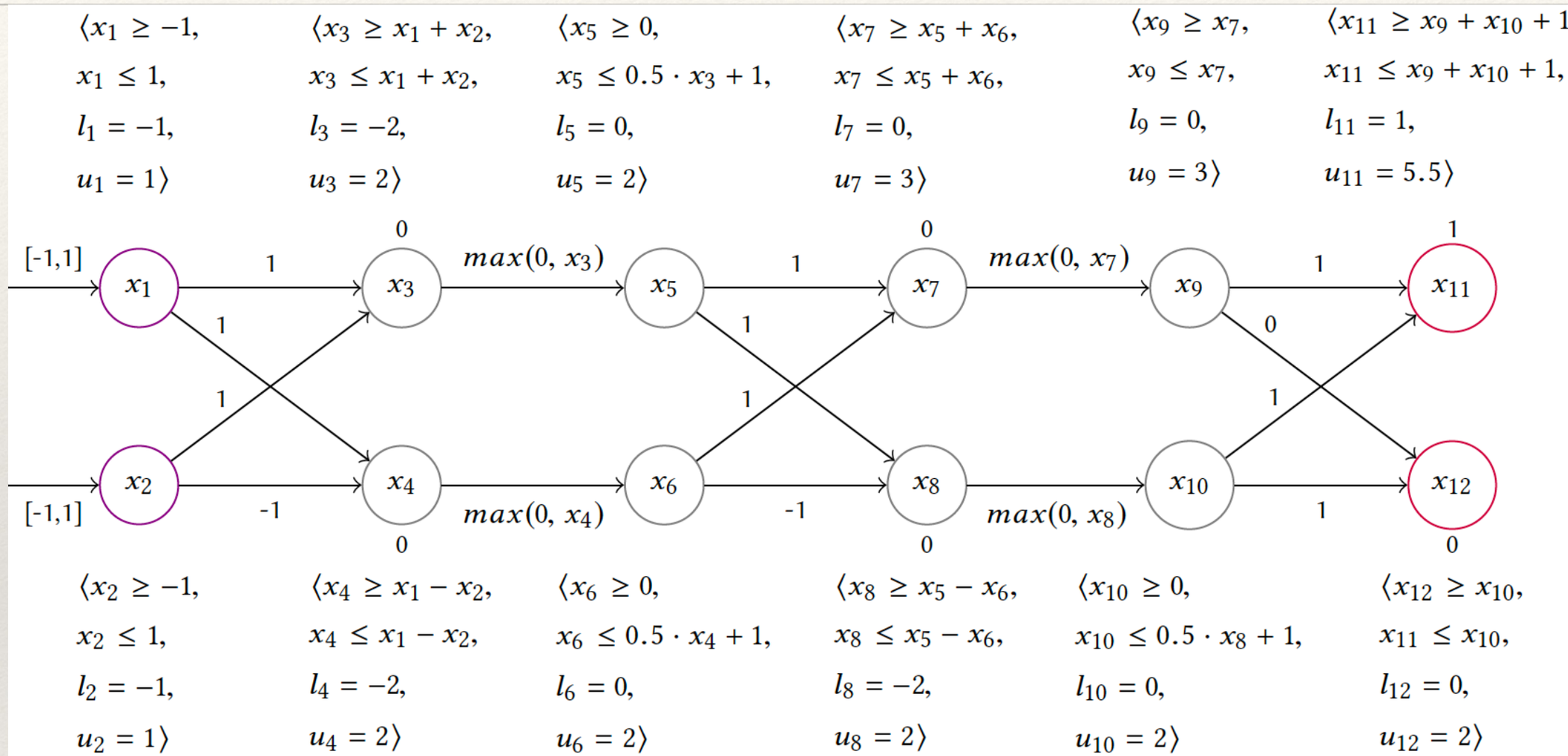
$$0 \leq x_7 \leq 0.5 \cdot x_3 + 0.5 \cdot x_4 + 2,$$
$$-0.5 \cdot x_4 - 1 \leq x_8 \leq 0.5 \cdot x_3 + 1.$$

2. Replacing x_3, x_4 with the constraints from x_1, x_2

$$0 \leq x_7 \leq x_1 + 2,$$
$$-0.5 \cdot x_1 + 0.5 \cdot x_2 - 1 \leq x_8 \leq 0.5 \cdot x_1 + 0.5 \cdot x_2 + 1.$$



Propagation

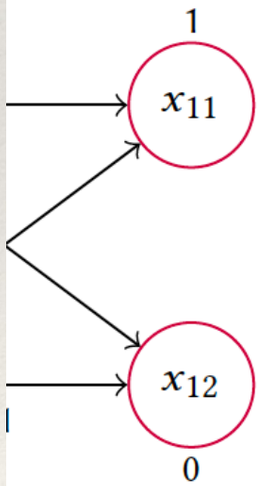


The transformers work point-wise, i.e., they are independent for different variables since they only read constraints and bounds from the previous layers. This makes it possible to parallelize the analysis

Property Checking

- Whether $x_{11} - x_{12} > 0$ or $x_{12} - x_{11} > 0$ holds given the abstract results computed by our analysis.
- Create a new temporary variable and apply our abstract transformer for the assignment $x_{13} := x_{11} - x_{12}$
- $x_{11} - x_{12} \leq x_{13} \leq x_{11} - x_{12}$
- Recursively substituting the constraints, get $l_{13} = 1$, $u_{13} = 4$
- Robustness specification proved

$\langle x_{11} \geq x_9 + x_{10} + 1,$
$x_{11} \leq x_9 + x_{10} + 1,$
$l_{11} = 1,$
$u_{11} = 5.5\rangle$

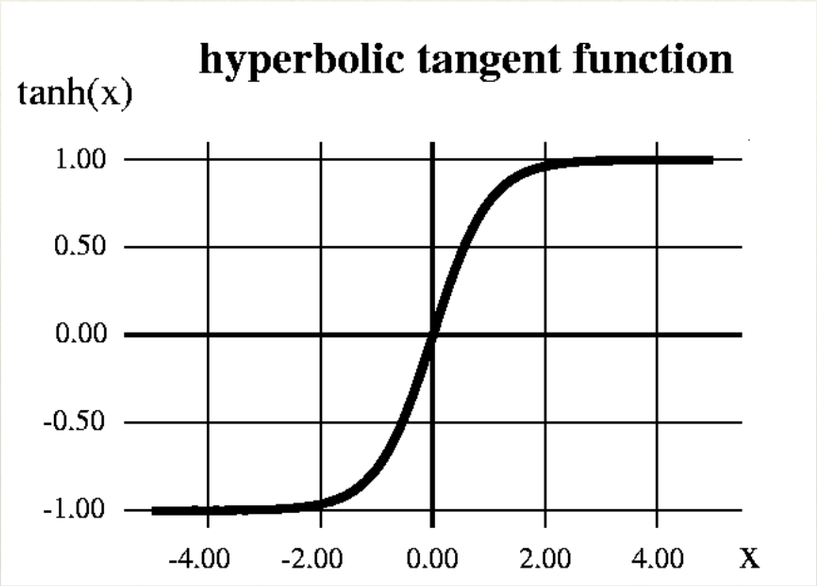


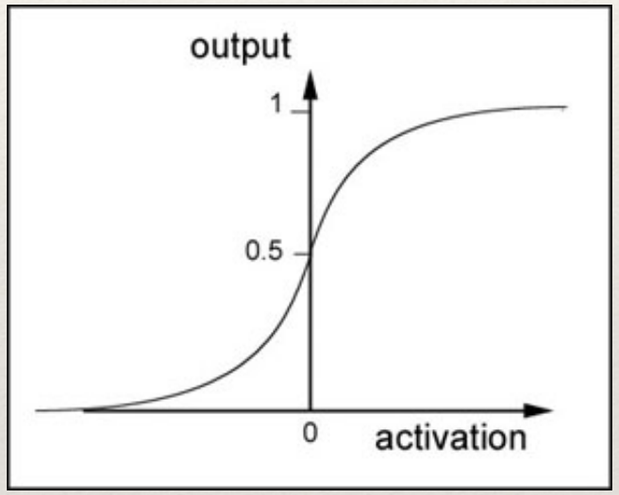
$\langle x_{12} \geq x_{10},$
$x_{11} \leq x_{10},$
$l_{12} = 0,$
$u_{12} = 2\rangle$

Abstract Domain and Transformers

Contributions: introduce new transformers for affine transforms, the rectified linear unit (ReLU), <u>sigmoid, tanh</u>, and maxpool functions

- Sigmoid and Tanh Abstract Transformers $x_i = T_f^{\#}(x_j)$





The function is $g(x)$, $l_i = g(l_j)$, $u_i = g(u_j)$. How to compute $a_i^{\leq}$ and $a_i^{\geq}$?

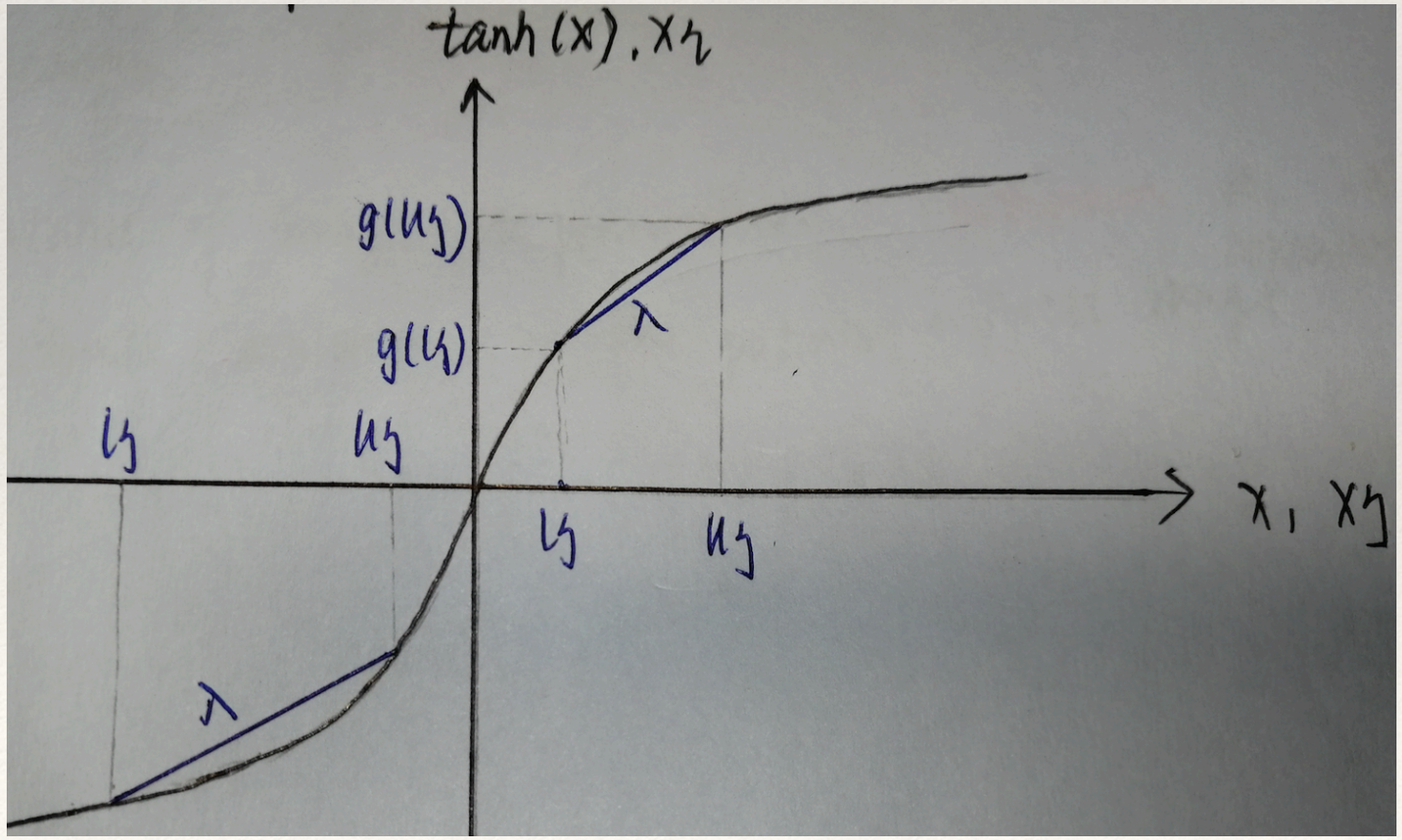
- If $l_j = u_j$, $a_i^{\leq} = a_i^{\geq} = g(l_j)$
- Otherwise, compute $a_i^{\leq}$ *and* $a_i^{\geq}$ separately

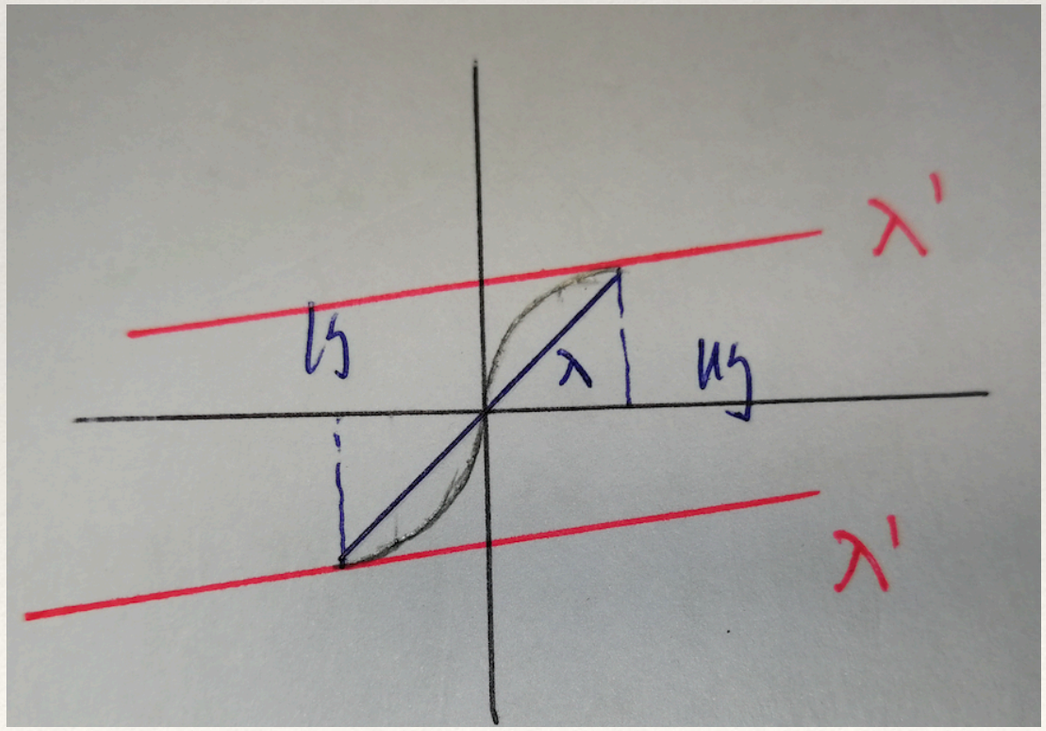
Abstract Domain and Transformers

Let $\lambda = (g(u_j) - g(l_j))/(u_j - l_j)$. The slope of the line between $(l_j, g(l_j))$ *and* $(u_j, g(u_j))$

$\lambda' = min(g'(l_j), g'(u_j))$. The min slope at point $(l_j, g(l_j))$ *and* $(u_j, g(u_j))$

- If $l_j > 0$, then $a_i^{\leq} = g(l_j) + \lambda \cdot (x_j - l_j)$
- Else $a_i^{\leq} = g(l_j) + \lambda' \cdot (x_j - l_j)$
- If $u_j \leq 0$, then $a_i^{\geq} = g(u_j) + \lambda \cdot (x_j - u_j)$
- Else $a_i^{\geq} = g(u_j) + \lambda' \cdot (x_j - u_j)$





Soundness under Floating Point

The abstract domain and its transformers above are sound under real arithmetic but unsound under floating point arithmetic if one does not take care of the rounding errors.

Floating-point numbers have a limited number of digits, they cannot represent all real numbers accurately: when there are more digits than the format allows, the leftover ones are omitted - the number is rounded.

For a real constant c, use $c^-, c^+ \in F$ to denote the floating point representation of c with rounding towards $-\infty$ and $+\infty$ respectively

Then the computation $(+, -, \times, /)$ for real number c now becomes interval $[c^-, c^+]$ addition, subtraction, multiplication, and division

Eg: tanh(x): $\lambda = (g(u_j) - g(l_j))/(u_j - l_j)$

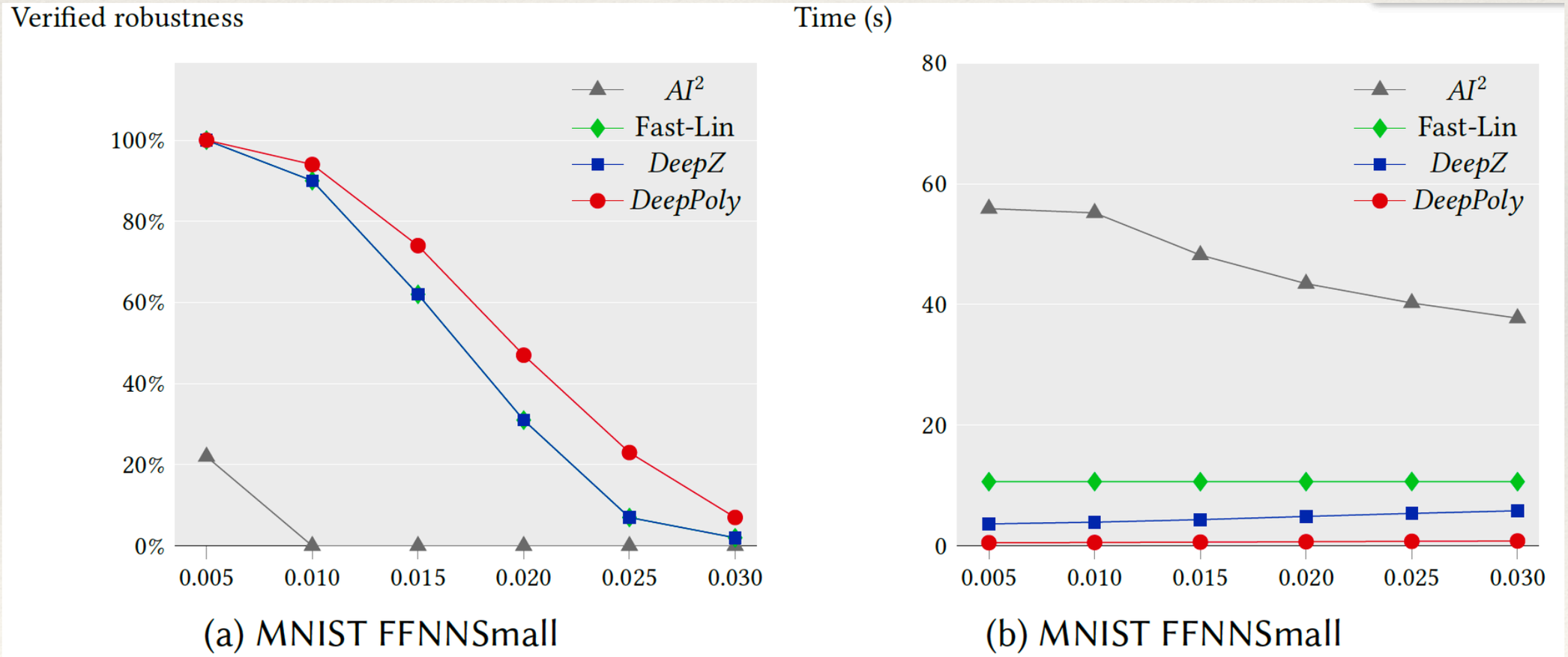
$[\lambda^-, \lambda^+] = ([g(u_j)^-, g(u_j)^+] \ominus_f [g(l_j)^-, g(l_j)^+]) \oslash_f ([u_j^-, u_j^+] \ominus_f [l_j^-, l_j^+])$

Experimental Evaluation

Same dataset as in AI^2: MNIST and CIFAR-10

Compared to AI2, Fast-Lin and DeepZ (both in 2018)

- Sequential running: most precise and fastest



Experimental Evaluation

Compared to DeepZ (both in 2018) (AI2 cannot be efficiently parallelized)

- Parallel running:
  - For feedforward NN, more precise and faster than DeepZ
  - For convolutional NN, proves slightly more properties than DeepZ, but is slower

Further Discussion

The first time that images perturbed by complex transformations such as rotations employing linear interpolation are considered.

But I couldn't understand this part yet…

Rotation perturbation part is in Sec 5, page 17-19

DeepPoly available on http://safeai.ethz.ch/

Thank you!

–Presented by Zhong Yuyi