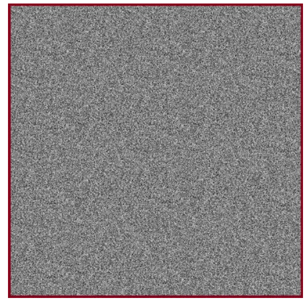
Multi-path Back-propagation for Neural Network Verification

Authors: Zheng Ye, Shi Xiaomu, Liu Jiaxiang

Shenzhen University

Neural Network Verification

+



=

x_1 prediction: Stop

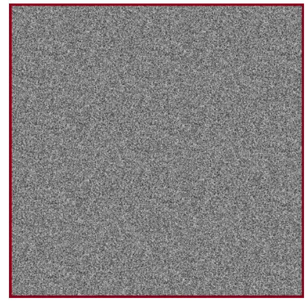
δ small perturbation

$x_1 + \delta$ prediction: 80km/h

Image source: https://www.businessinsider.com/why-are-stop-signs-red

Neural Network Verification

$+$

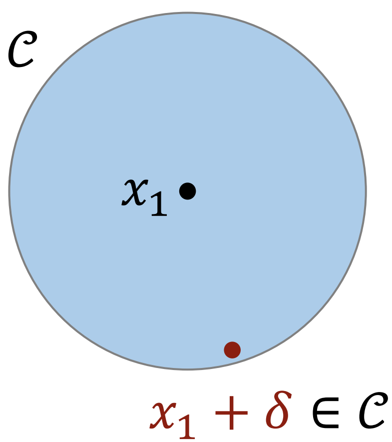


$=$

x_1 prediction: Stop

δ small perturbation

$x_1 + \delta$ prediction: 80km/h



$f(x_1) = \text{STOP}$

$f(x_1 + \delta) = 80\text{km/h}$

Verify: Given f and $\mathcal{C}$, $\forall x \in \mathcal{C}$

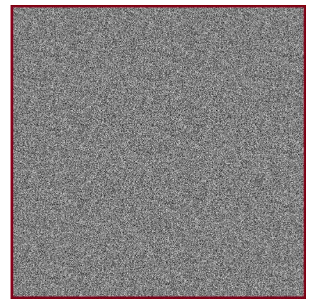
$$f(x) = \text{STOP}$$

Image source: https://www.businessinsider.com/why-are-stop-signs-red

Neural Network Verification

x_1 prediction: Stop

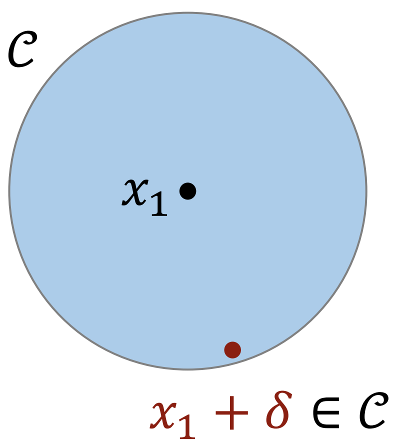
+



δ small perturbation

=

$x_1 + \delta$ prediction: 80km/h



$f(x_1) = \text{STOP}$

$f(x_1 + \delta) = 80\text{km/h}$

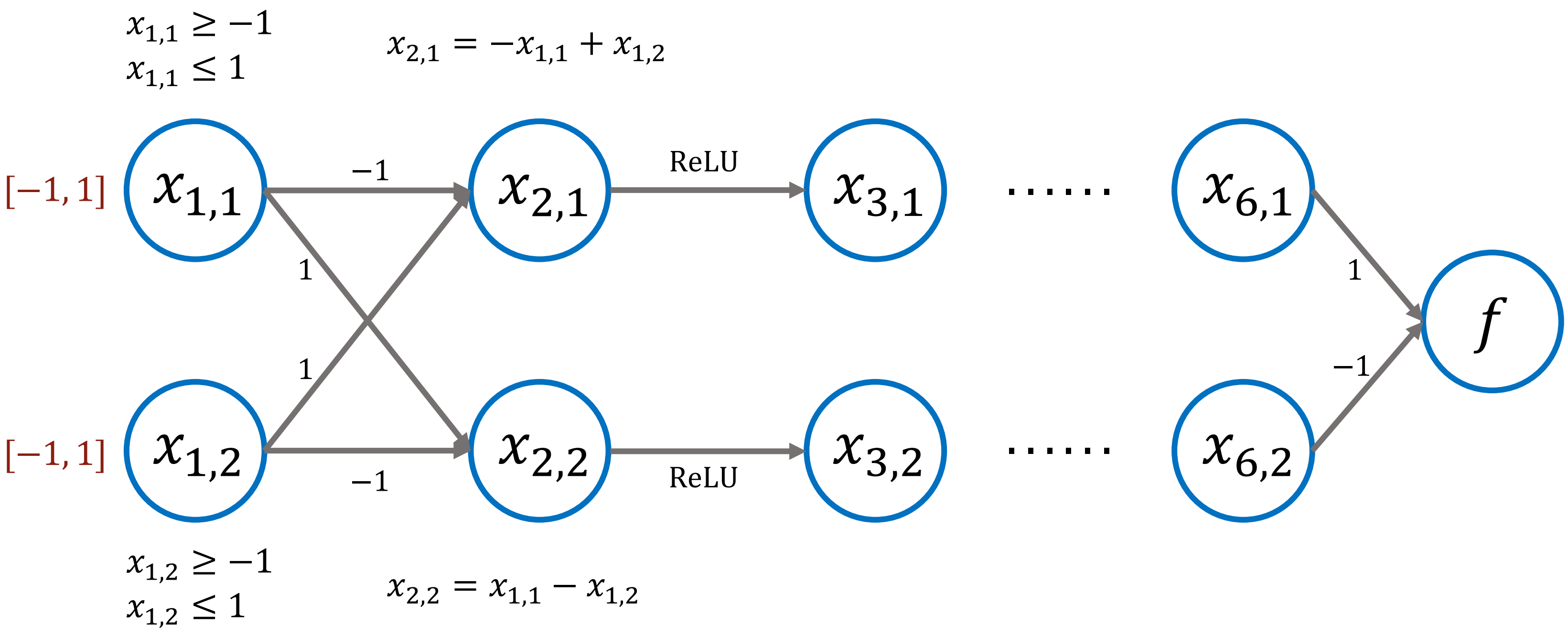
Verify: Given f and $\mathcal{C}$, $\forall x \in \mathcal{C}$

$$f(x) \geq 0$$

Image source: https://www.businessinsider.com/why-are-stop-signs-red

- One method: solve optimization problems

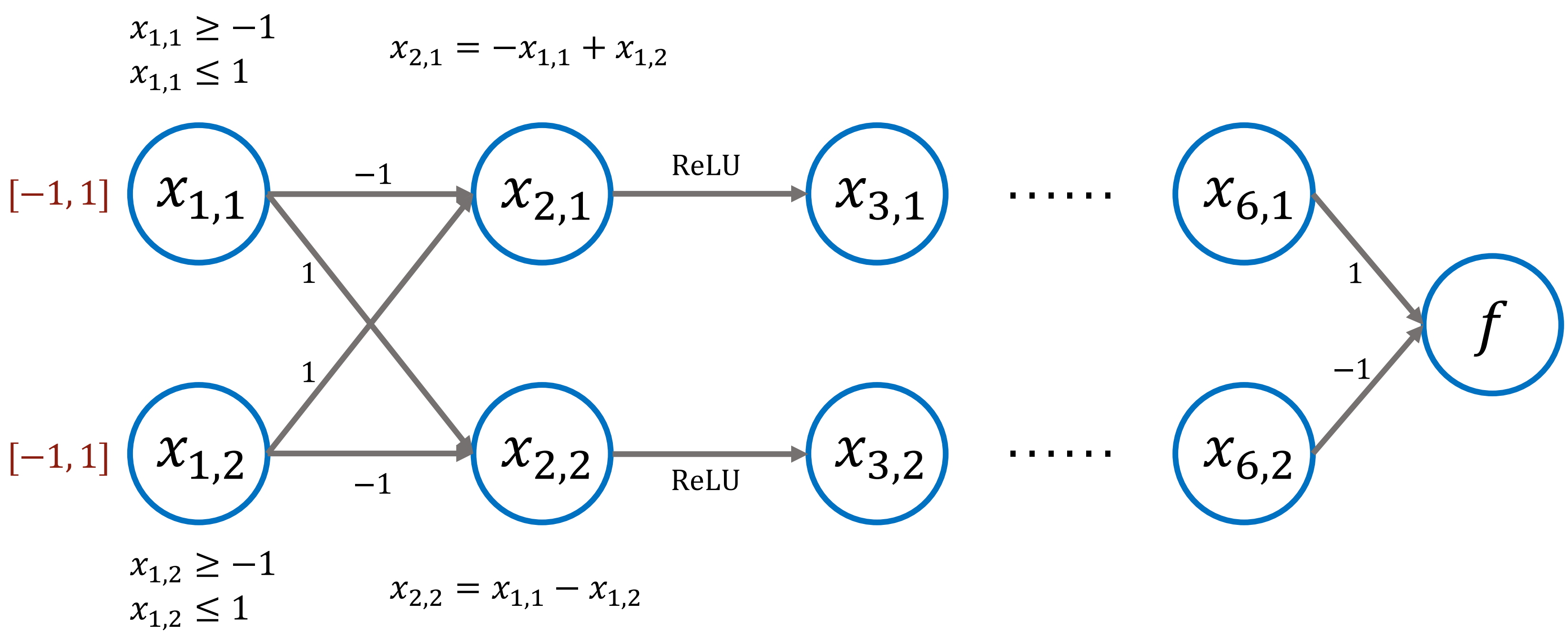
Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



Back-propagation for NN Verification

- One method: solve optimization problems (low efficiency)
- Back-propagate to calculate the lower and upper bounds of each node

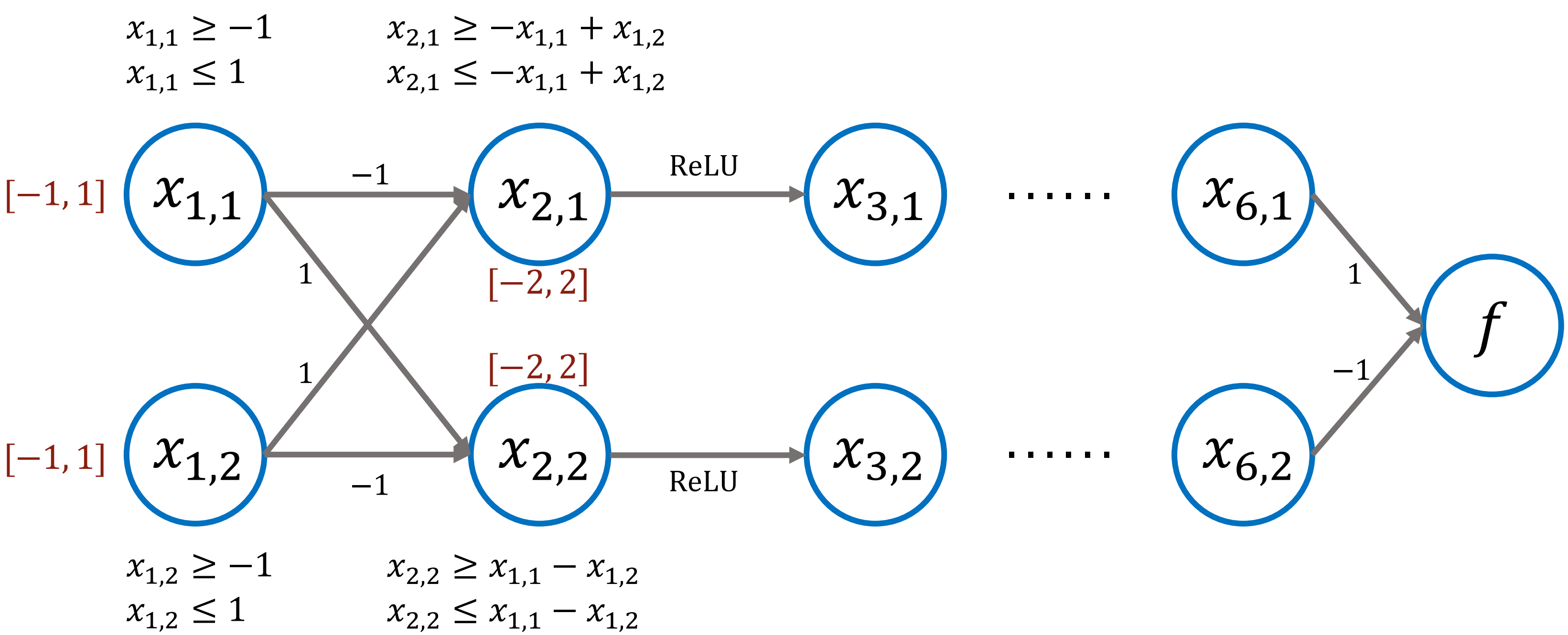
Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



Back-propagation for NN Verification

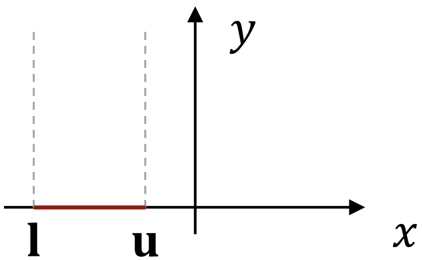
- One method: solve optimization problems (low efficiency)
- Back-propagate to calculate the lower and upper bounds of each node

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$

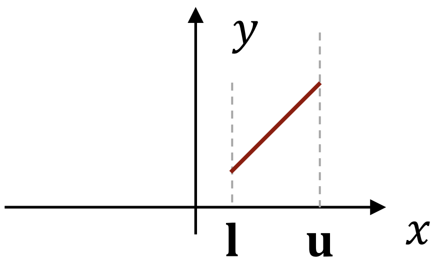


Back-propagation for NN Verification

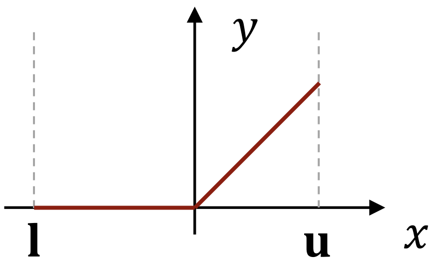
- ReLU neurons have three cases depending their input bounds:



$\mathbf{u} \leq 0$: zero



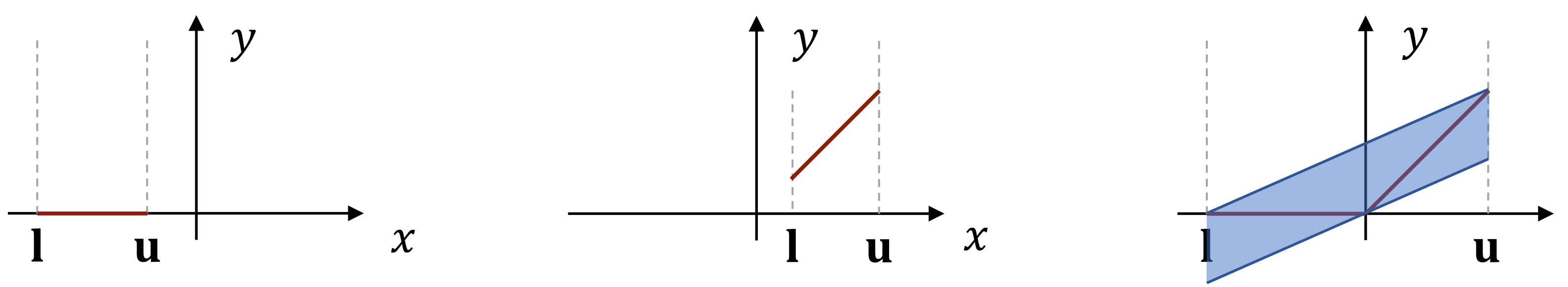
$\mathbf{l} \geq 0$: linear



$\mathbf{l} \leq 0 \leq \mathbf{u}$: piecewise

Back-propagation for NN Verification

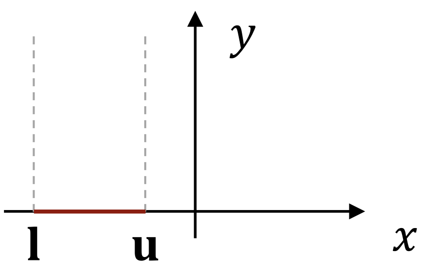
- ReLU neurons have three cases depending their input bounds:
- Linear bounds for non-linear ReLU:



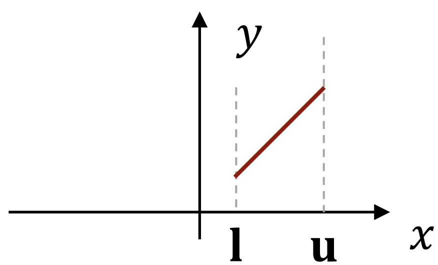
$\mathbf{u} \leq 0$: zero $\mathbf{l} \geq 0$: linear $\mathbf{l} \leq 0 \leq \mathbf{u}$: piecewise

Back-propagation for NN Verification

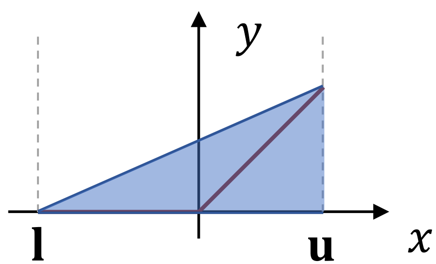
- ReLU neurons have three cases depending their input bounds:
- Linear bounds for non-linear ReLU:



$\mathbf{u} \leq 0$: zero



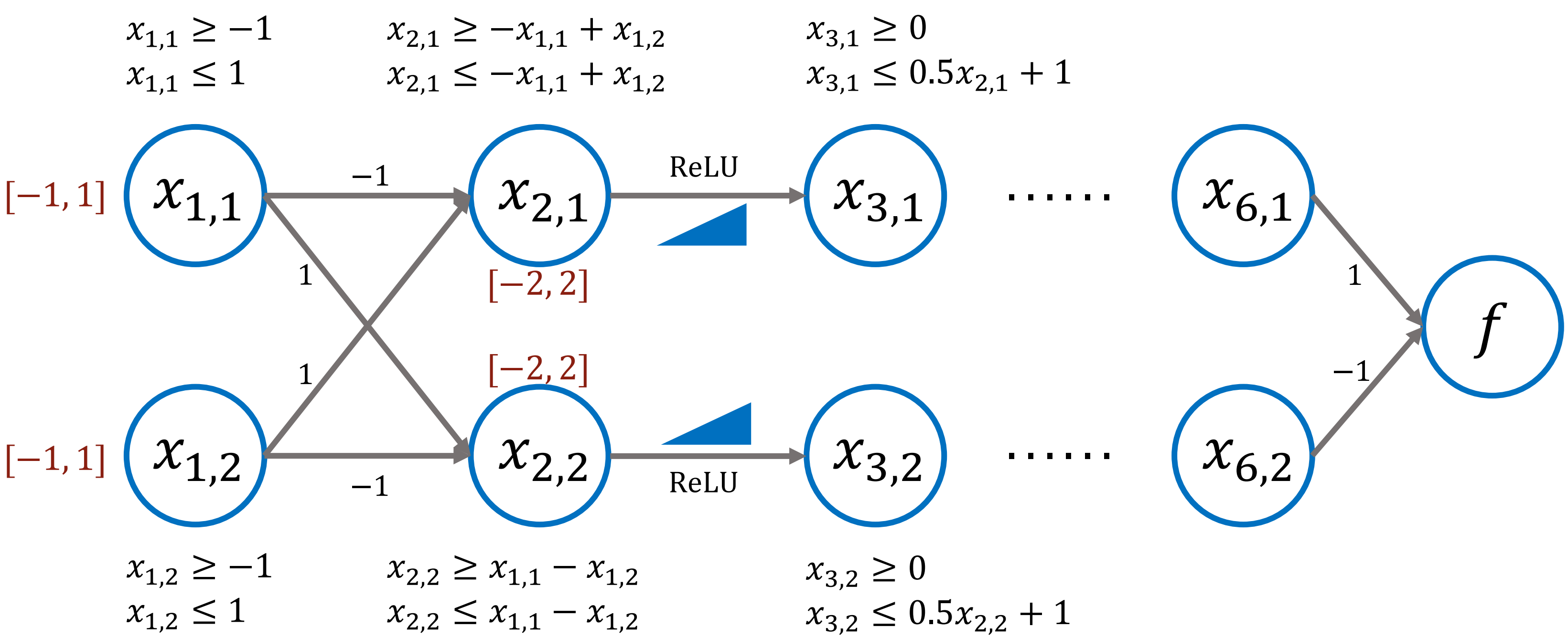
$\mathbf{l} \geq 0$: linear



$\mathbf{l} \leq 0 \leq \mathbf{u}$: piecewise

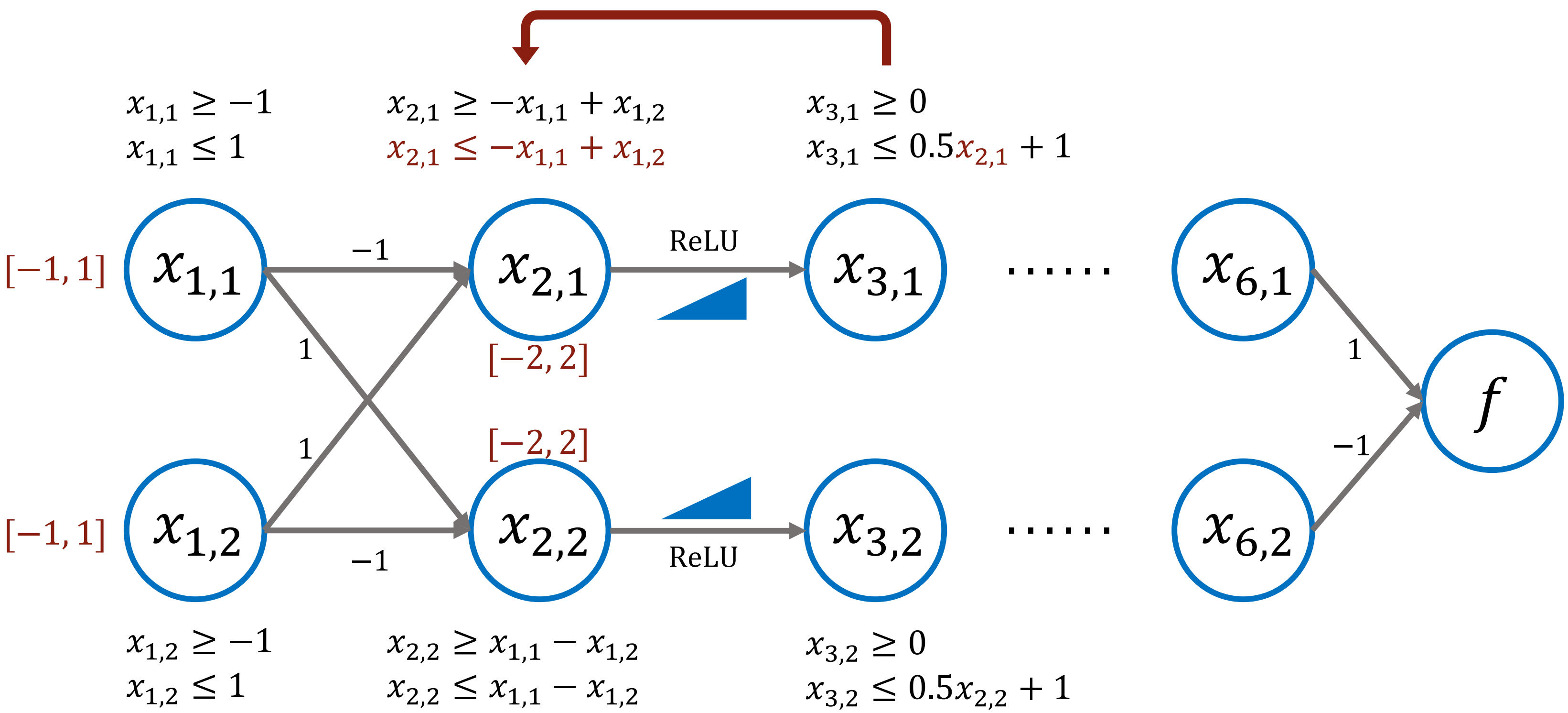
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



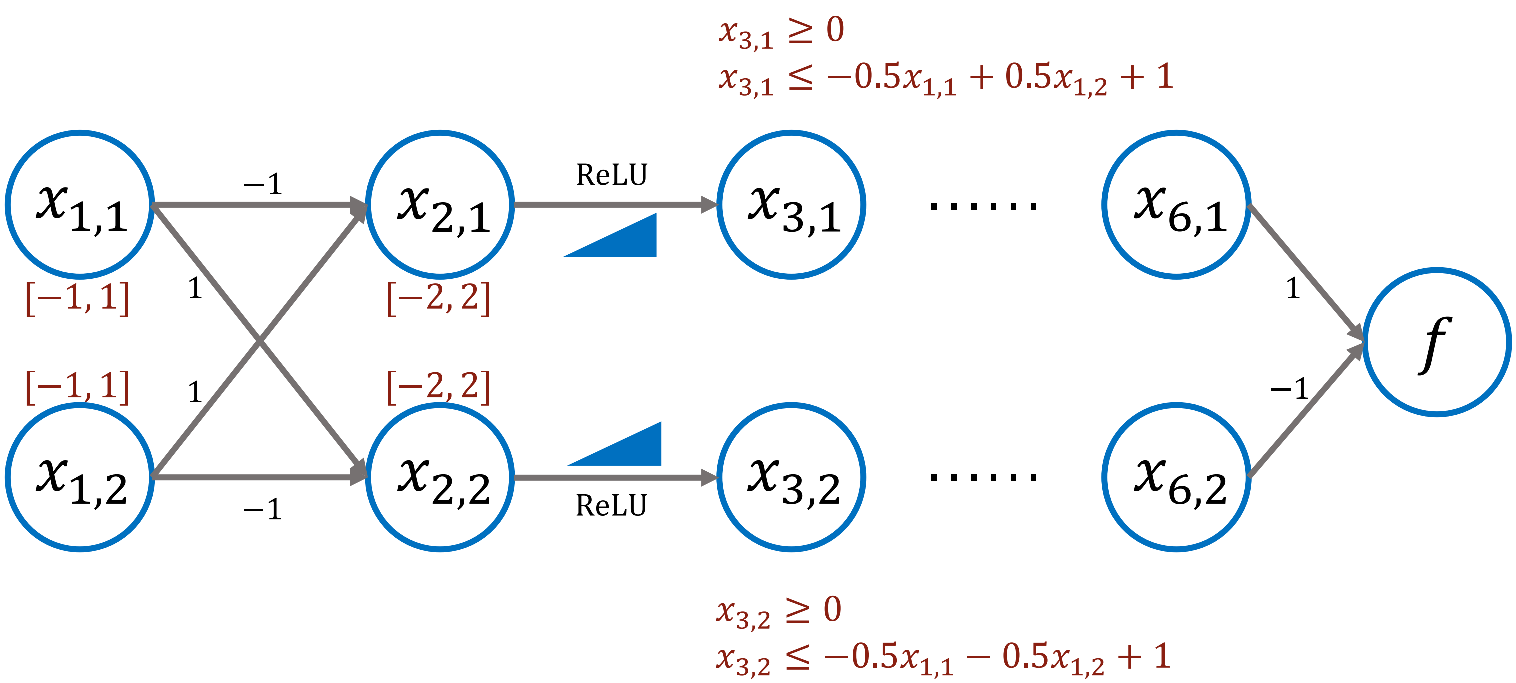
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



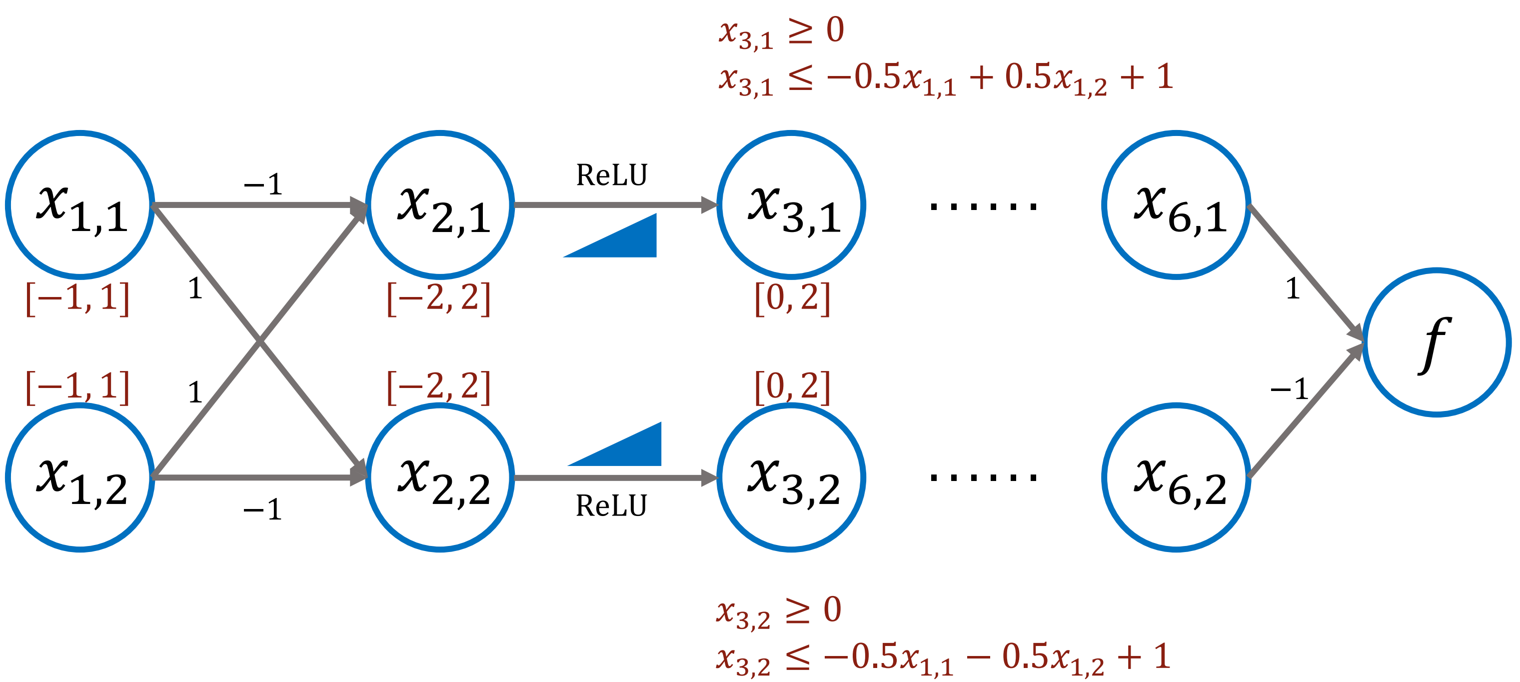
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



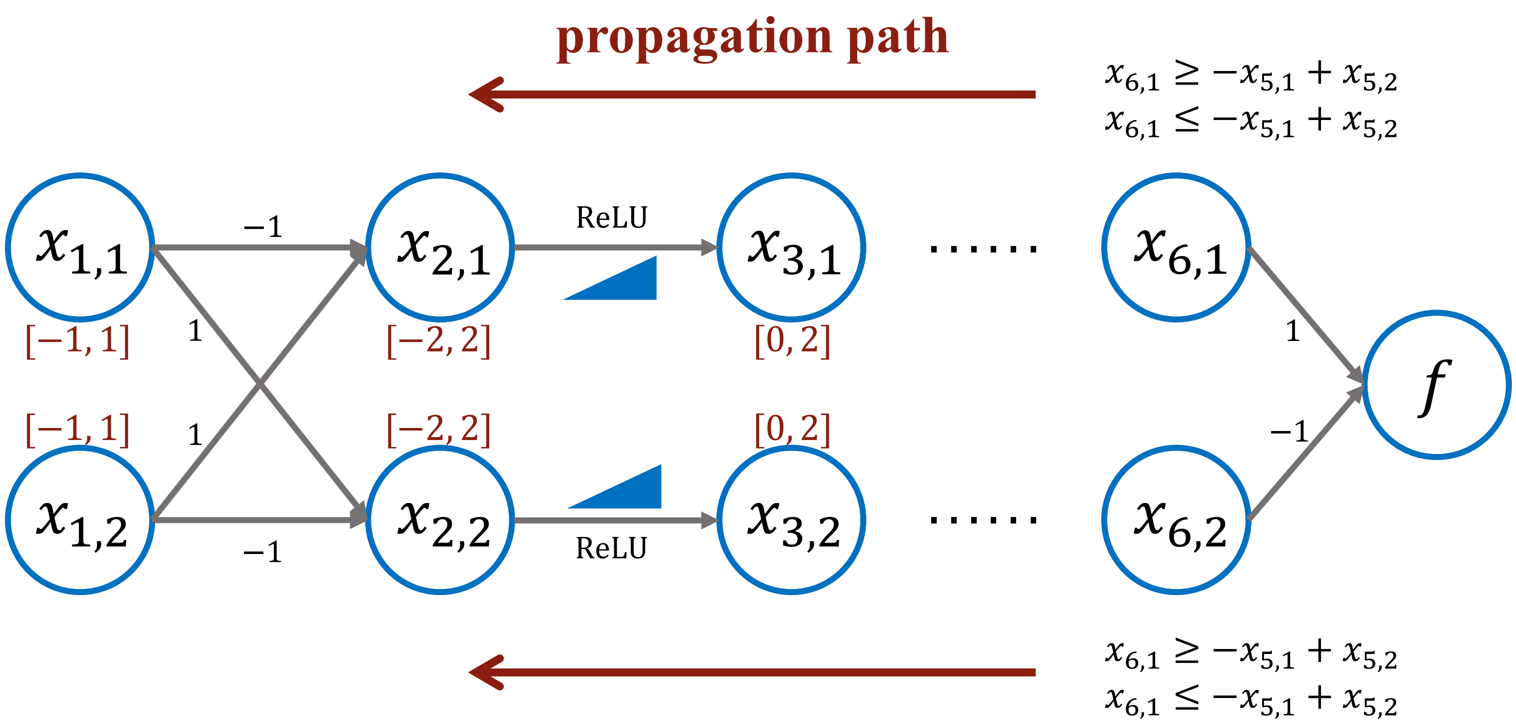
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



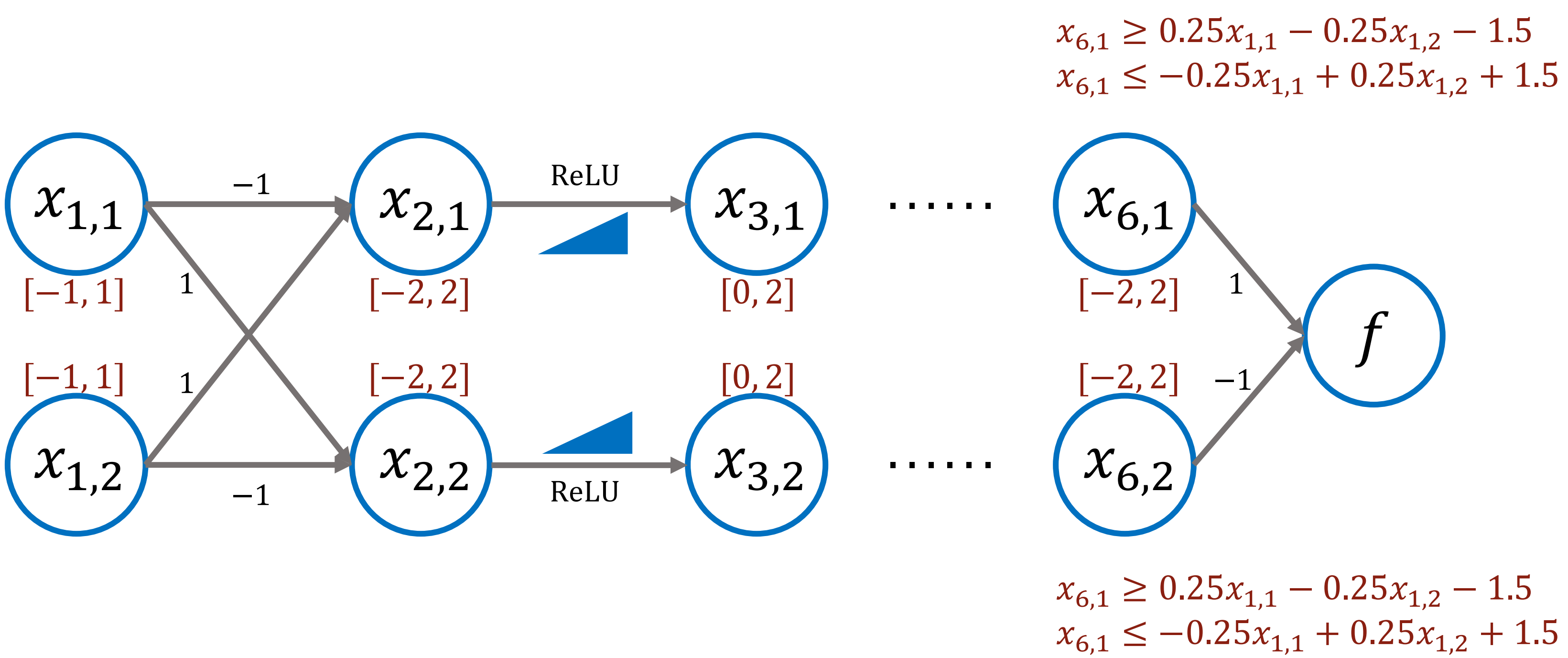
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



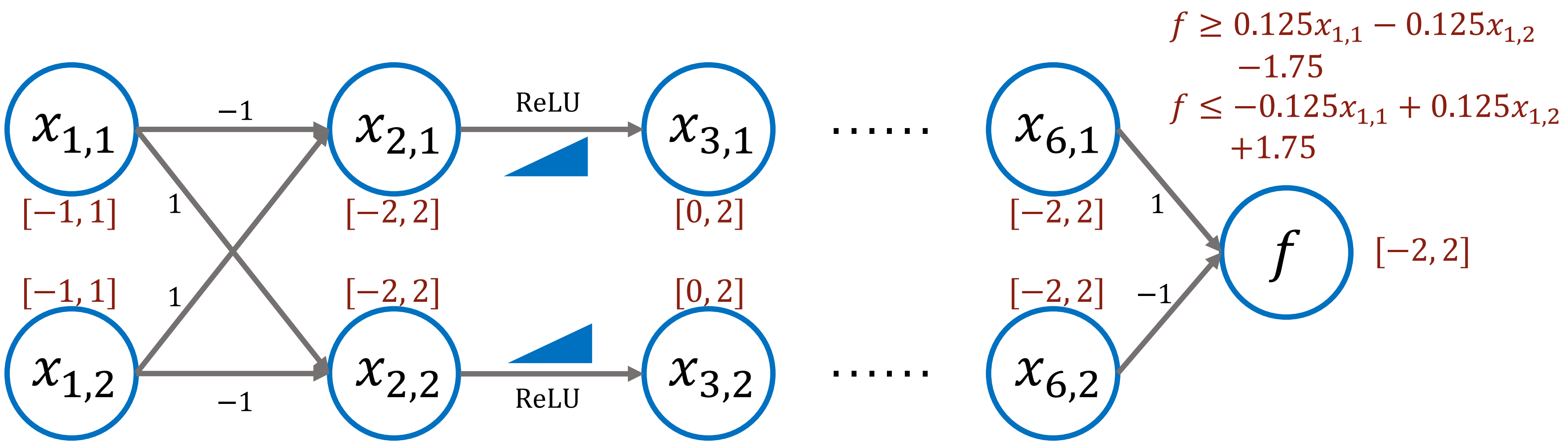
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



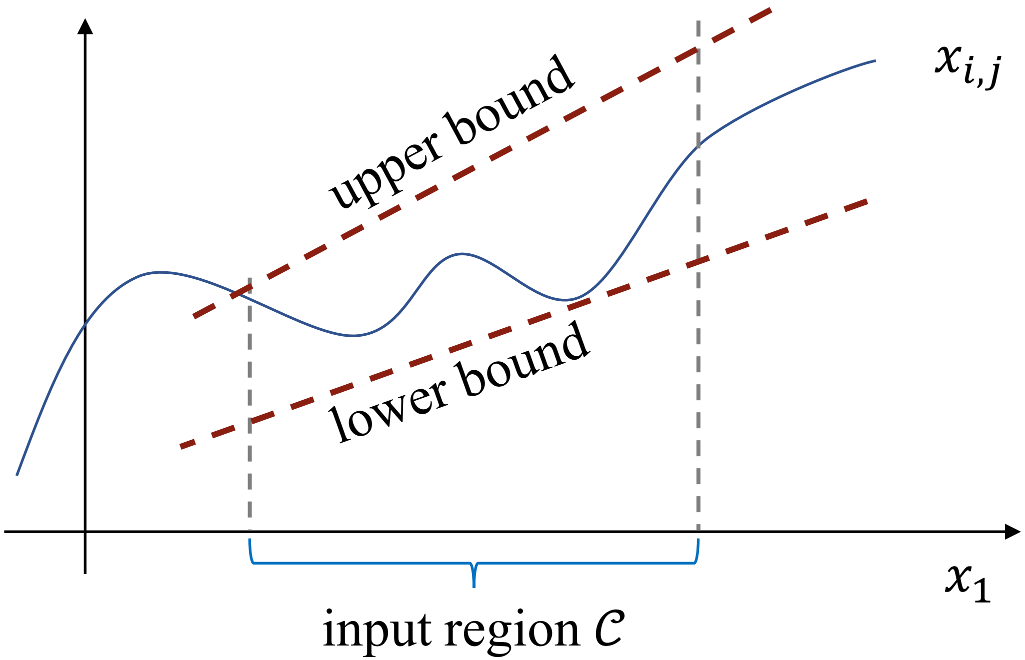
Back-propagation for NN Verification

Goal: $\forall \mathbf{x}_1 \in \mathcal{C},\ f(\mathbf{x}_1) \geq 0$



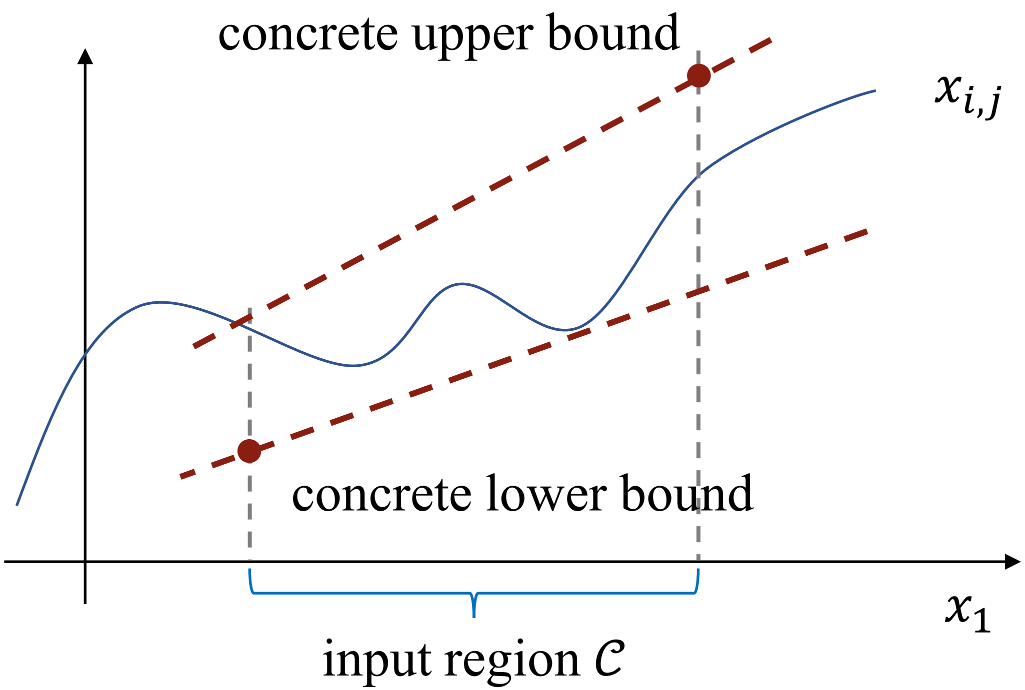
Back-propagation for NN Verification

What propagation methods do? (one dimension example)



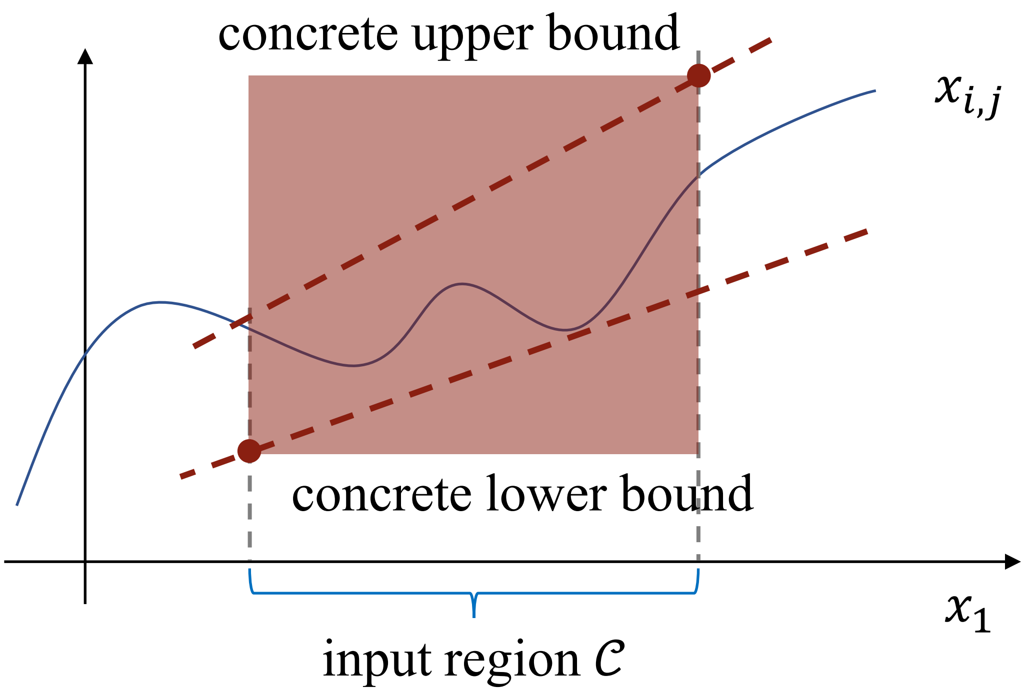
Back-propagation for NN Verification

What propagation methods do? (one dimension example)



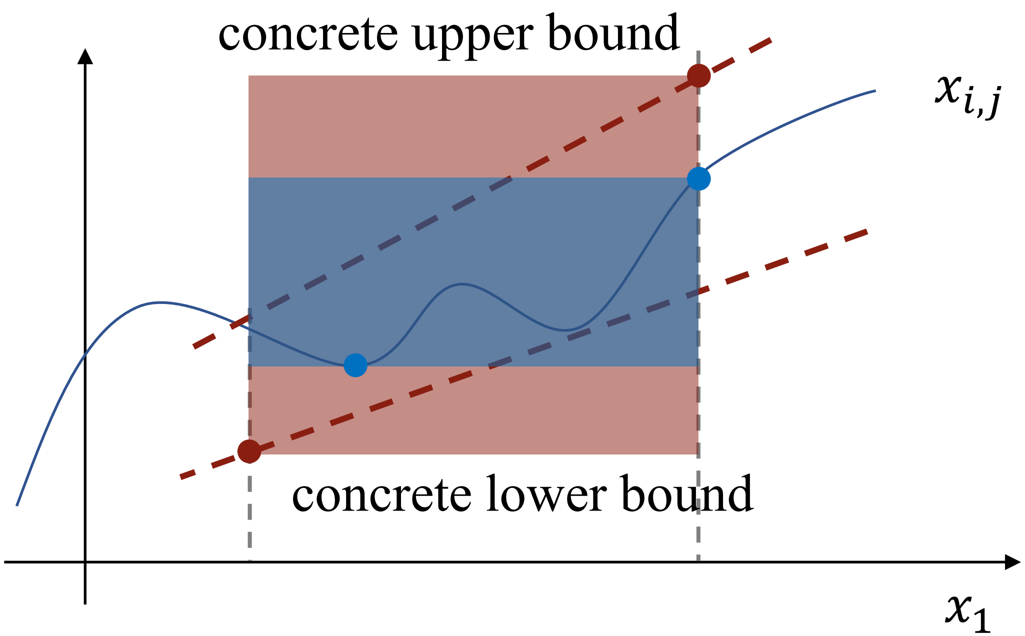
Back-propagation for NN Verification

What propagation methods do? (one dimension example)



Back-propagation for NN Verification

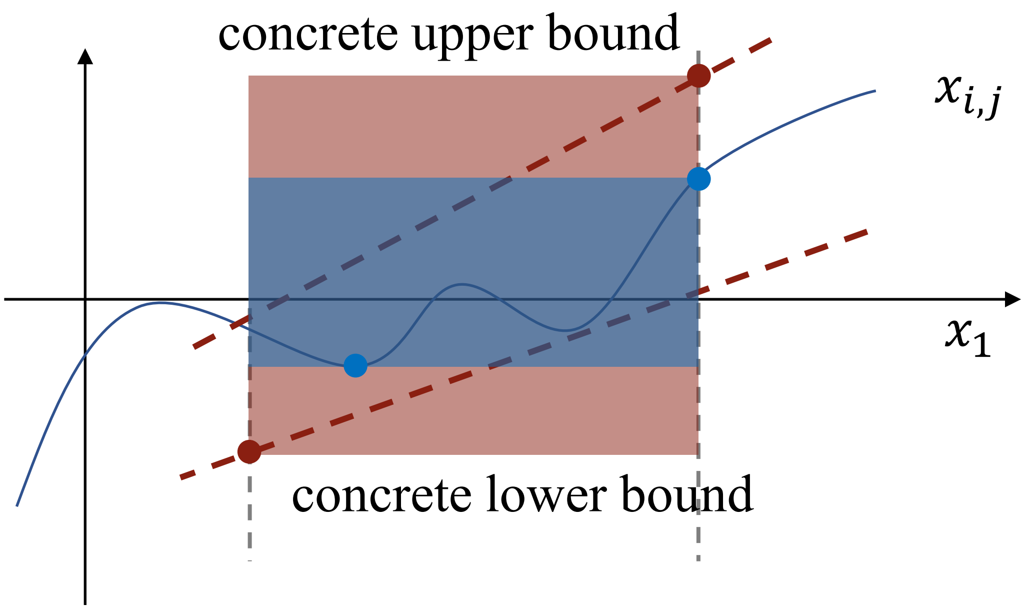
What propagation methods do? (one dimension example)

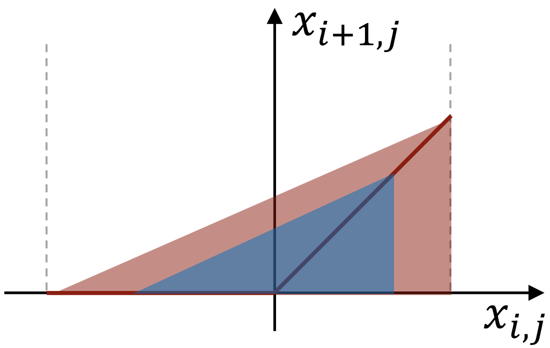


- Sound but not complete
- Tighter bounds bring better verification results

Back-propagation for NN Verification

What propagation methods do? (one dimension example)





- Sound but not complete
- Tighter bounds bring better verification results

Multi-path Back-propagation

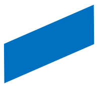
- Representative methods: DeepPoly, Fast-Lin, CROWN
- Specific cases of one path, being very fast but with loose bounds
- This work improves the bounds of back-propagation methods

Main idea:

More propagation paths will get more bounds

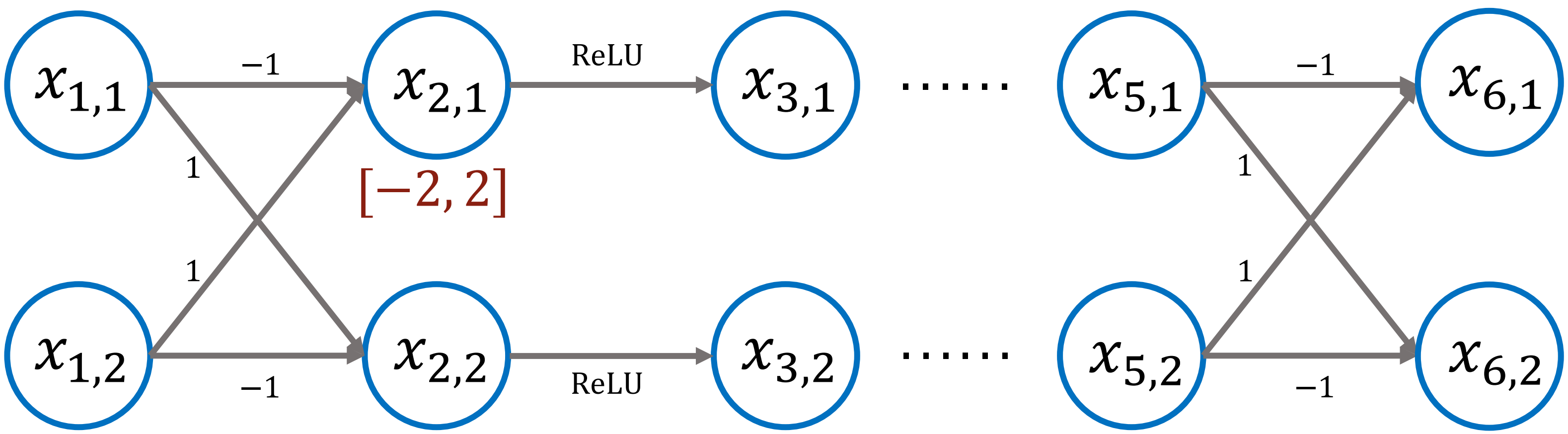
The union of these sound bounds is also a sound bound

Two-path Example

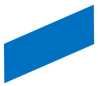


$$x_{3,1} \geq 0.5 x_{2,1}$$
$$x_{3,1} \leq -0.5 x_{2,1} + 1$$

$$x_{3,1} \geq 0$$
$$x_{3,1} \leq -0.5 x_{2,1} + 1$$

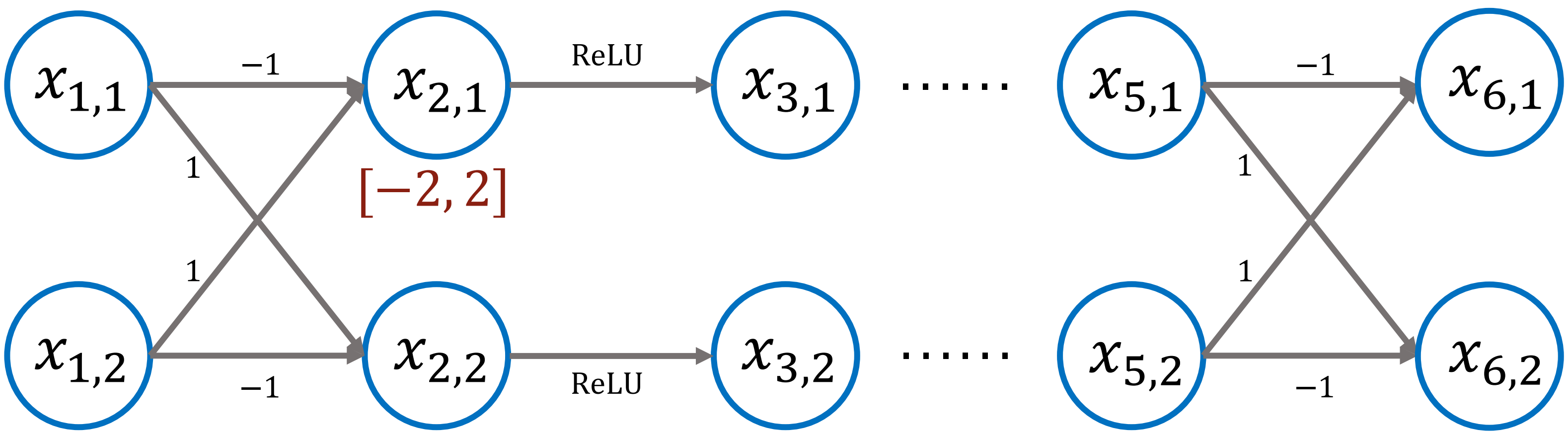


Two-path Example



$x_{3,1} \geq 0.5x_{2,1}$
$x_{3,1} \leq -0.5x_{1,1} + 0.5x_{1,2} + 1$

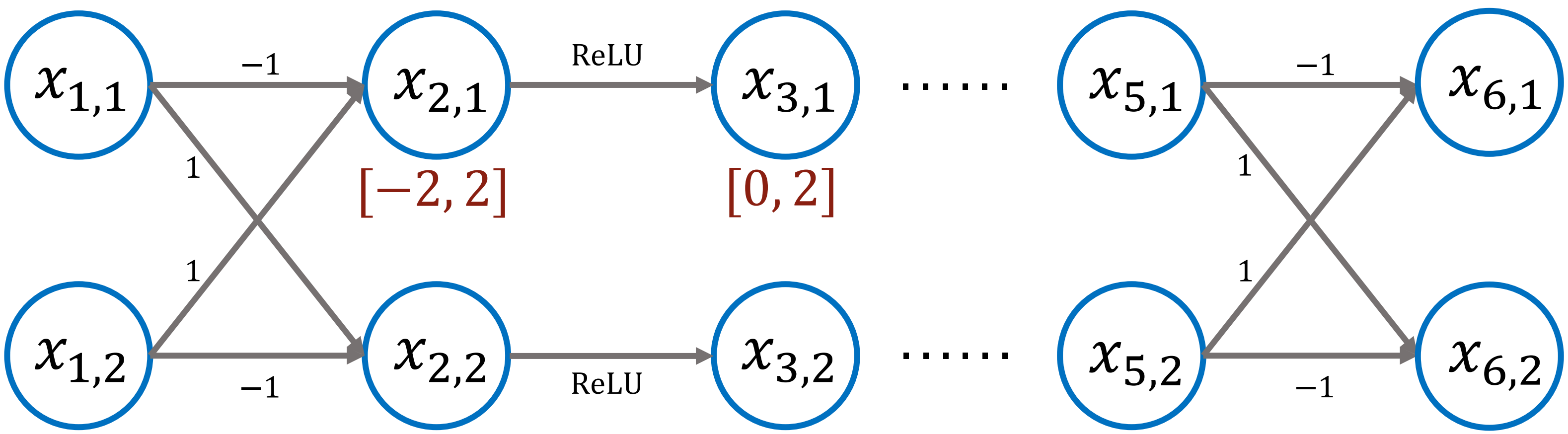
$x_{3,1} \geq 0$
$x_{3,1} \leq -0.5x_{1,1} + 0.5x_{1,2} + 1$



Two-path Example

$[-1, 2]$

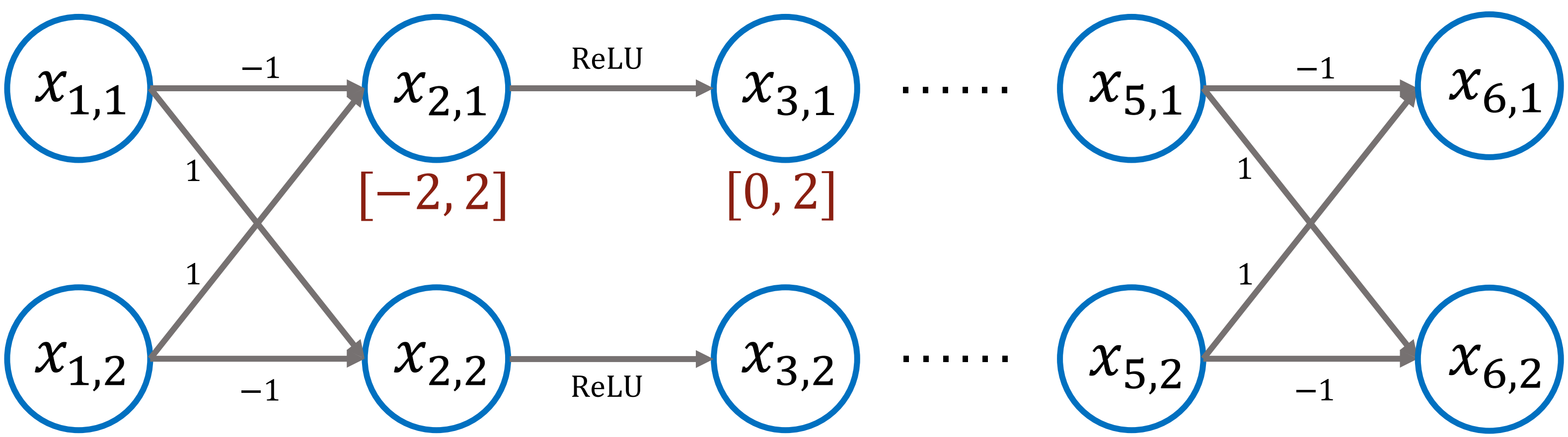
$[0, 2]$



Two-path Example

$$x_{5,1} \geq -0.5x_{1,1} + 0.5x_{1,2} - 0.5$$
$$x_{5,1} \leq -0.5x_{1,1} + 0.5x_{1,2} + 1.5$$

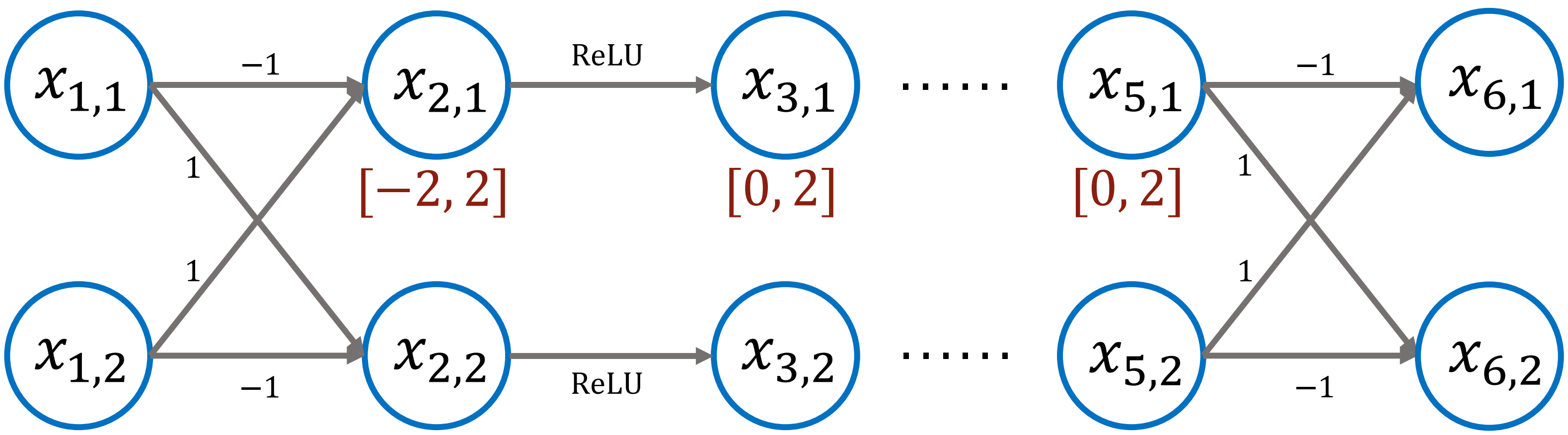
$$x_{5,1} \geq 0$$
$$x_{5,1} \leq -0.25x_{1,1} + 0.25x_{1,2} + 1.5$$



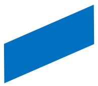
Two-path Example

[−1.5, 2.5]

[0, 2]

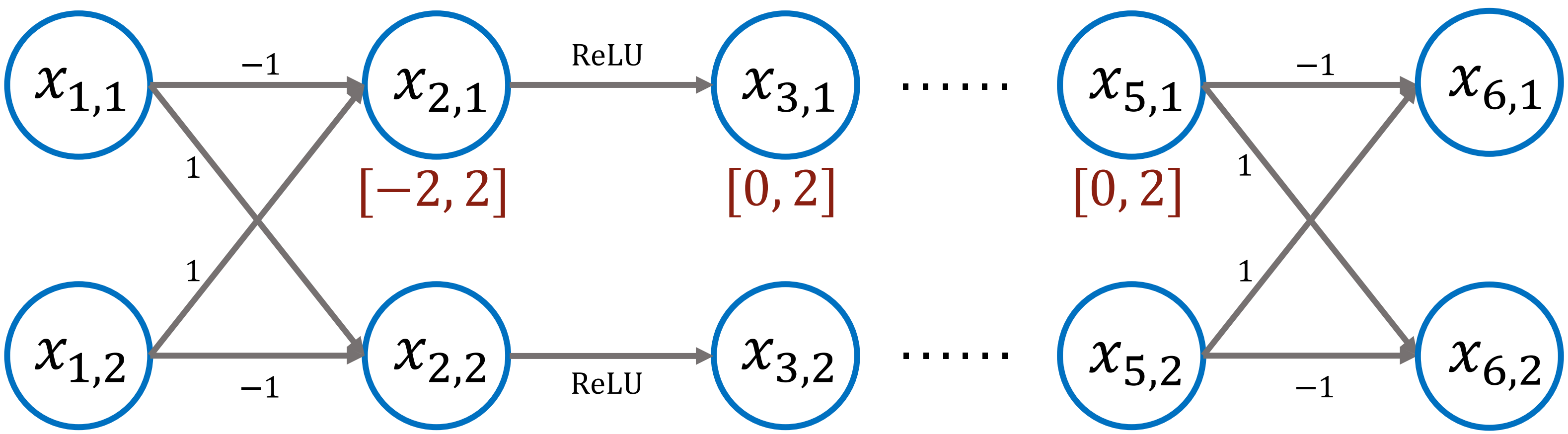


Two-path Example



$x_{6,1} \geq -1$
$x_{6,1} \leq 1$

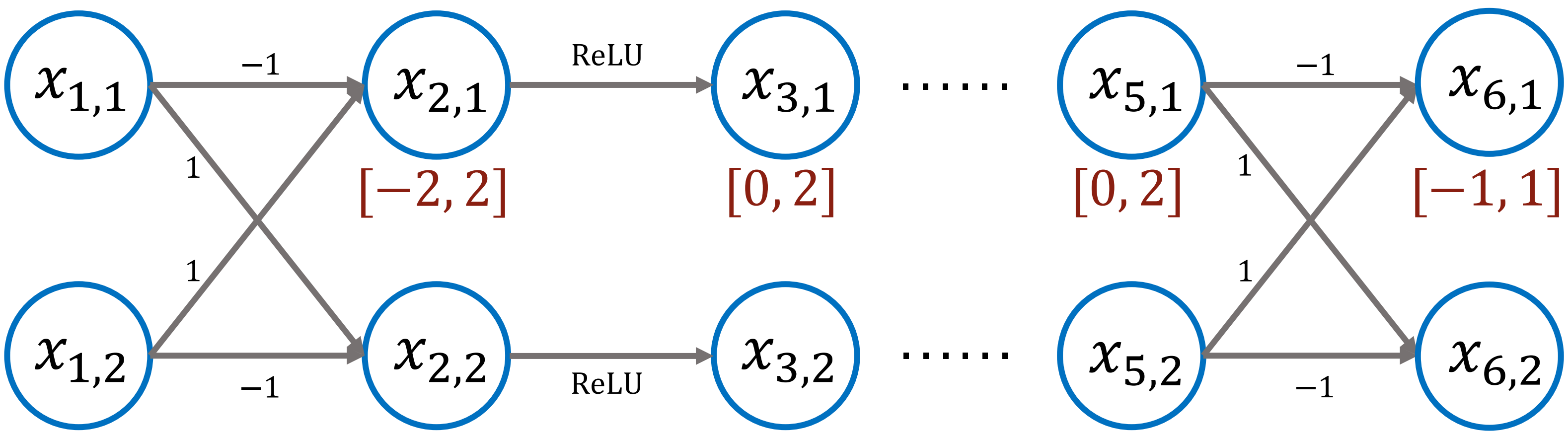
$x_{6,1} \geq 0.25x_{1,1} - 0.25x_{1,2}$
-1.5
$x_{6,1} \leq -0.25x_{1,1} + 0.25x_{1,2}$
$+1.5$



Two-path Example

$[-1, 1]$

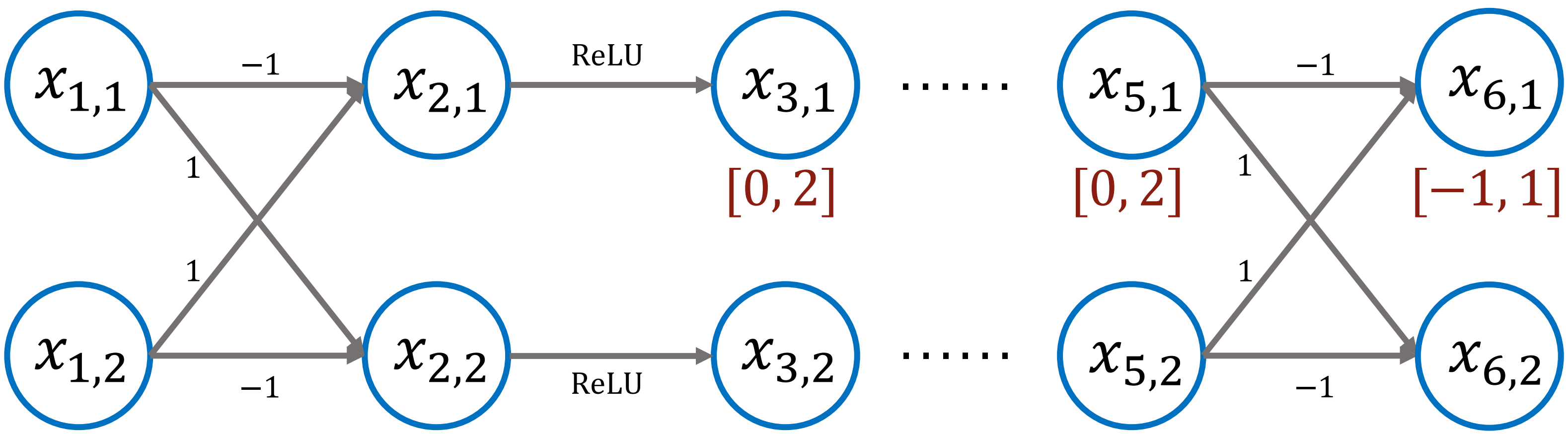
$[-2, 2]$



Two-path Example

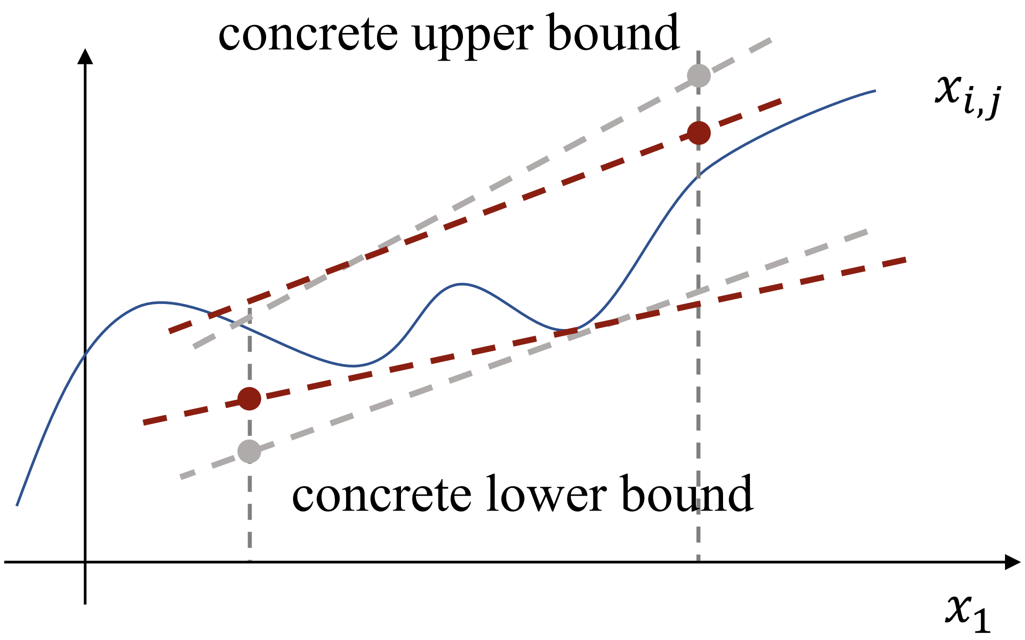
√

√ √



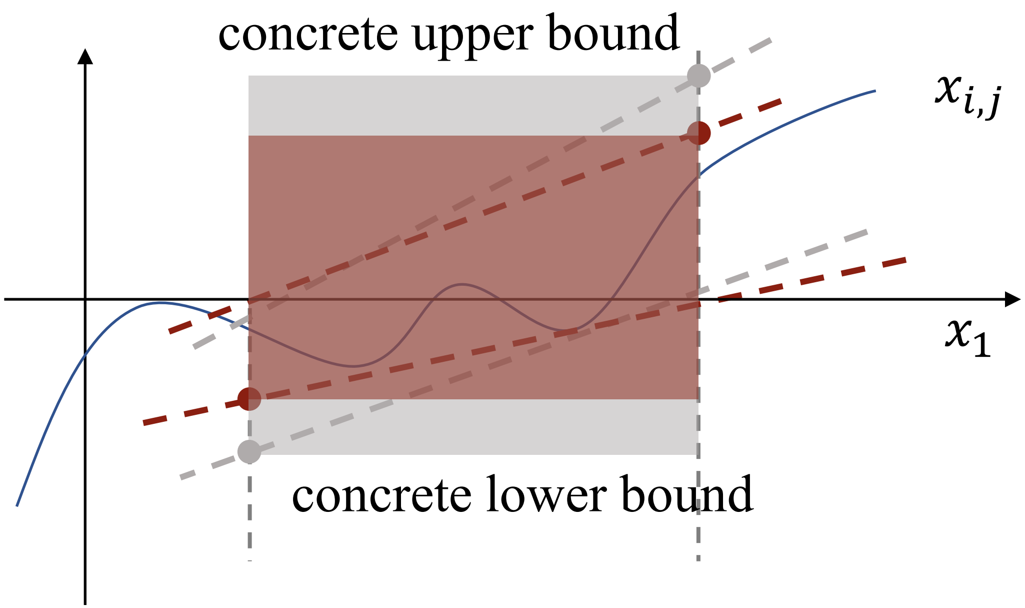
Two-path Example

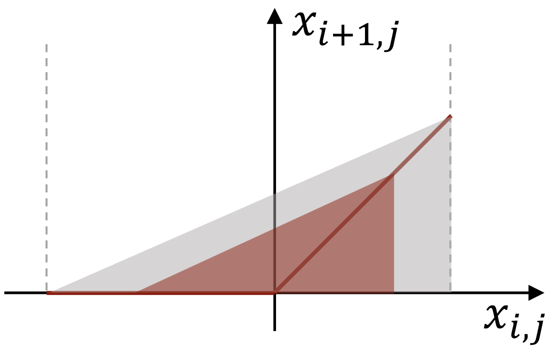
Two-path propagation: Two bounds for comparing (one dimension example)



Two-path Example

Two-path propagation: Two bounds for comparing (one dimension example)





Multi-path Back-propagation

- Same ReLU over-approximation along one path, but not necessary
- Absolutely within propagation framework
- $\mathcal{O}(MN^2)$ time complexity, where M is the path number
  - Little additional cost
  - Highly parallelable
- At least has the accuracy of any one path, *i.e.*, DeepPoly

Experiments

- Implement multi-path back-propagation method as tool AbstraCMP

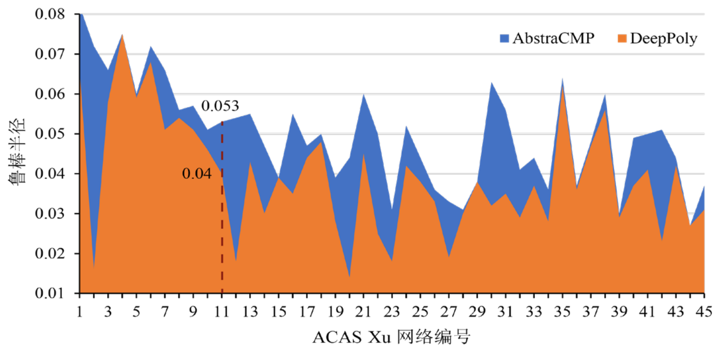
- Use robust radius as the accuracy metric
  - Compare with single path method DeepPoly
  - Compare with LP-ALL, which solve LP for each node

- Larger robustness radius means higher over-approximation accuracy

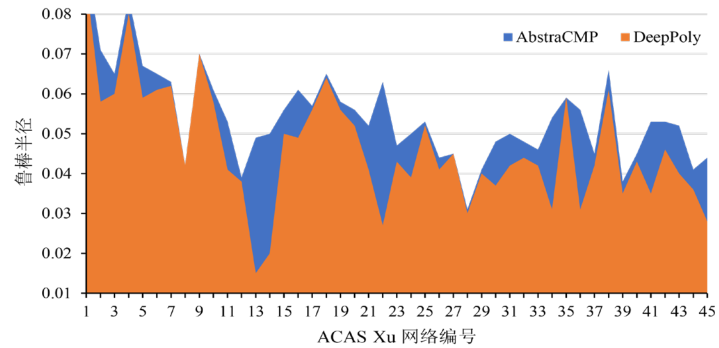
Code source: https://github.com/ZhengYeah/AbstraCMP

Experiments

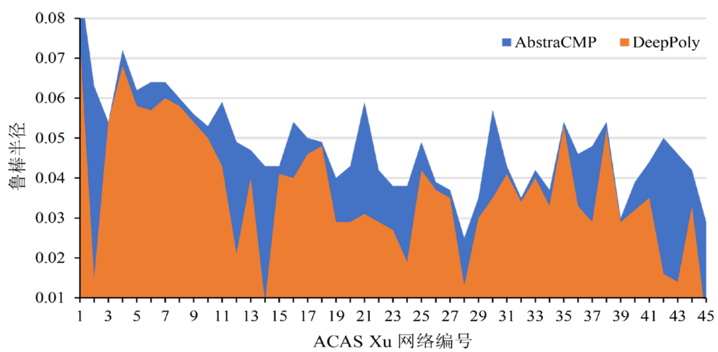
ACAS Xu Network, 4 random inputs on 45 networks



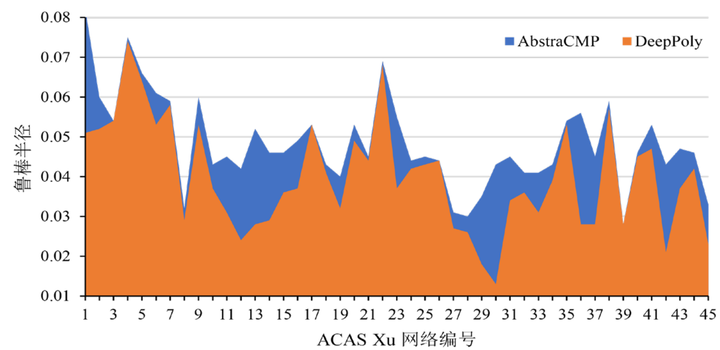
(a) 输入 1 在 45 个网络上的鲁棒半径



(b) 输入 2 在 45 个网络上的鲁棒半径



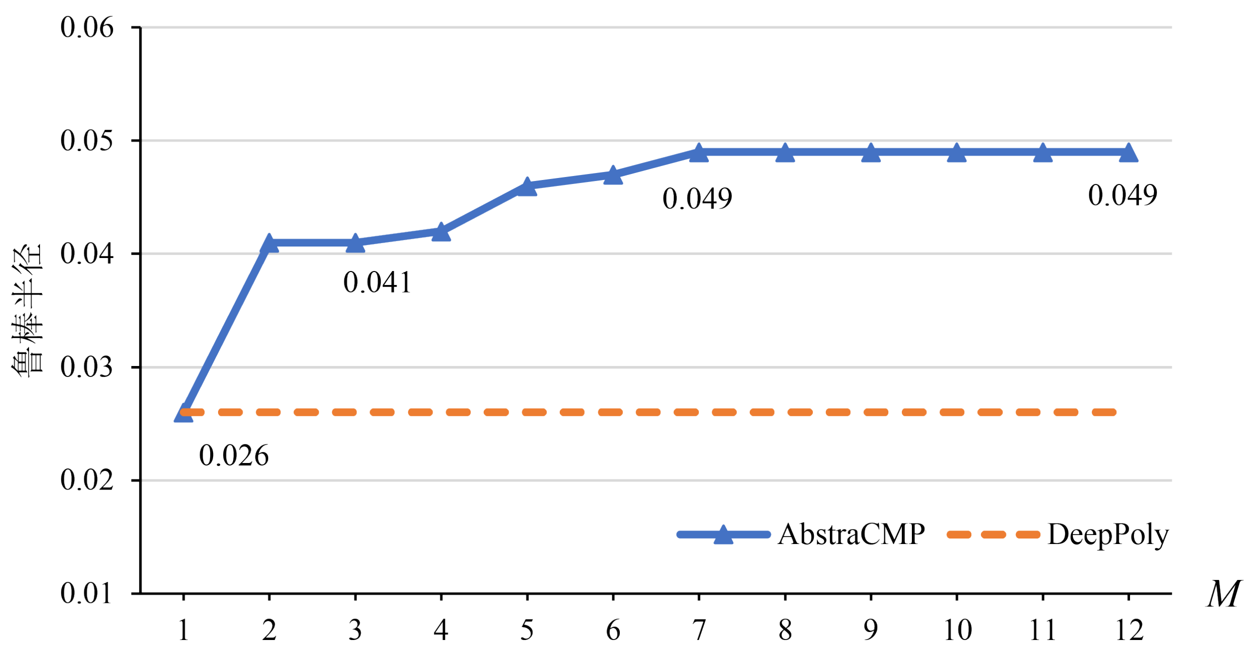
(c) 输入 3 在 45 个网络上的鲁棒半径



(d) 输入 4 在 45 个网络上的鲁棒半径

Experiments

- Intuitively, more paths will bring better results
- But this improvement is not sustainable



Experiments

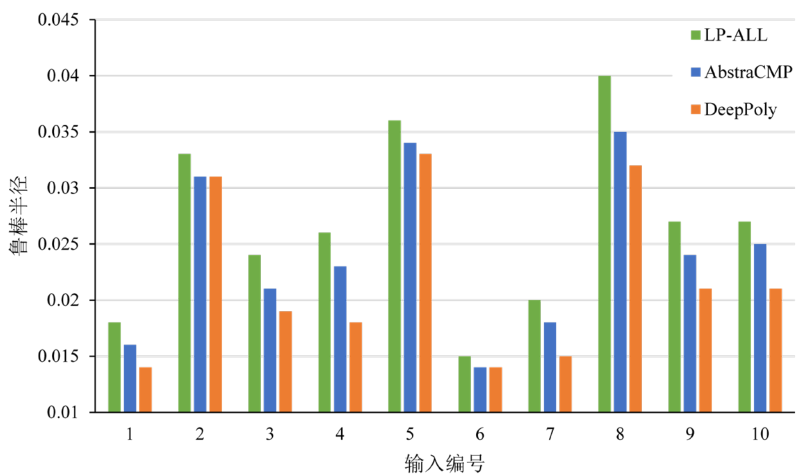
MNIST and CIFAR10, verified cases under δ (greater is better)

网络	方法	扰动大小 δ								
		0.010	0.012	0.015	0.017	0.020	0.022	0.025	0.027	0.030
MNIST 10×80	DeepPoly	91	88	79	67	46	33	27	20	10
	AbstraCMP	**94**	**91**	**82**	**70**	**48**	**38**	**31**	**24**	**12**
MNIST 20×50	DeepPoly	73	70	49	40	31	22	16	13	7
	AbstraCMP	**80**	**72**	**58**	**49**	**37**	**27**	**20**	**18**	**10**

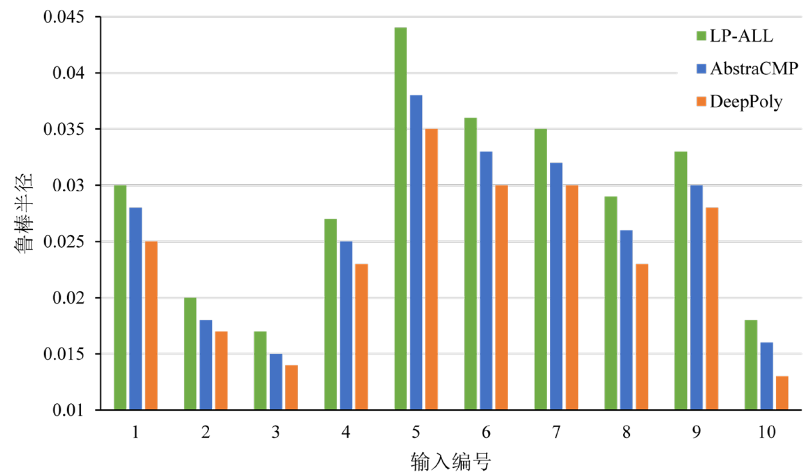
网络	方法	扰动大小 δ								
		0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045
CIFAR10 15×200	DeepPoly	93	87	75	64	57	48	35	32	21
	AbstraCMP	**94**	**88**	**79**	**70**	**61**	**55**	**47**	**35**	**29**
CIFAR10 16×250	DeepPoly	93	76	59	42	33	20	14	8	3
	AbstraCMP	**95**	**79**	**62**	**49**	**37**	**23**	**16**	**10**	**4**

Experiments

- LP-ALL needs over 41 hours for one result in our experiments while AbstraCMP needs only around 400 seconds
- Bridge the gap between propagation methods and LP-ALL



(a) MNIST 16×50 网络的鲁棒半径对比



(b) MNIST 10×80 网络的鲁棒半径对比

Summary and Future Works

- Improve the back-propagation methods using multiple paths
- Analyze the accuracy of multi-path propagation

Future Works

- Will extend to other activation functions & network structures
- Will use more efficient GPU implementation

App. I: Forward vs Backward

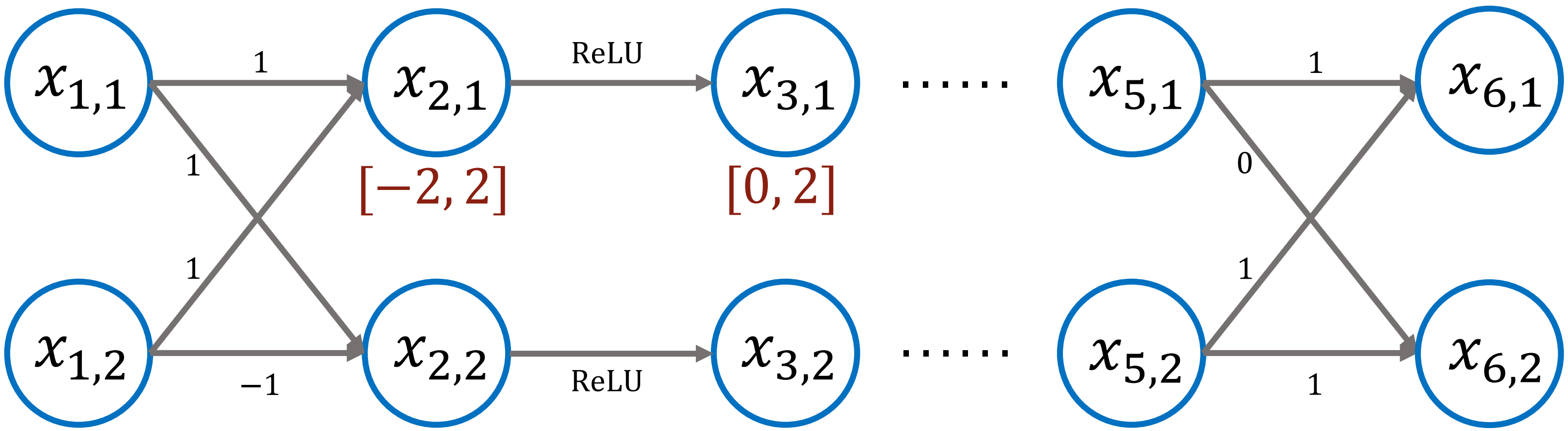
$x_{1,1} \geq -1$
$x_{1,1} \leq 1$

$x_{2,1} \geq x_{1,1} + x_{1,2}$
$x_{2,1} \leq x_{1,1} + x_{1,2}$

$x_{3,1} \geq 0$
$x_{3,1} \leq 0.5x_{2,1} + 1$

$x_{5,1} \geq x_{4,1}$
$x_{5,1} \leq x_{4,1}$
$x_{5,1} \geq 0$
$x_{5,1} \leq x_{1,1} + 2$

$x_{61} \geq x_{51} + x_{52}$
$x_{61} \leq x_{51} + x_{52}$



$x_{5,2} \geq 0$
$x_{5,2} \leq x_{4,2}$
$x_{5,1} \geq 0$
$x_{5,1} \leq 0.25x_{1,1} + 0.25x_{1,2} + 2$

App. I: Forward vs Backward

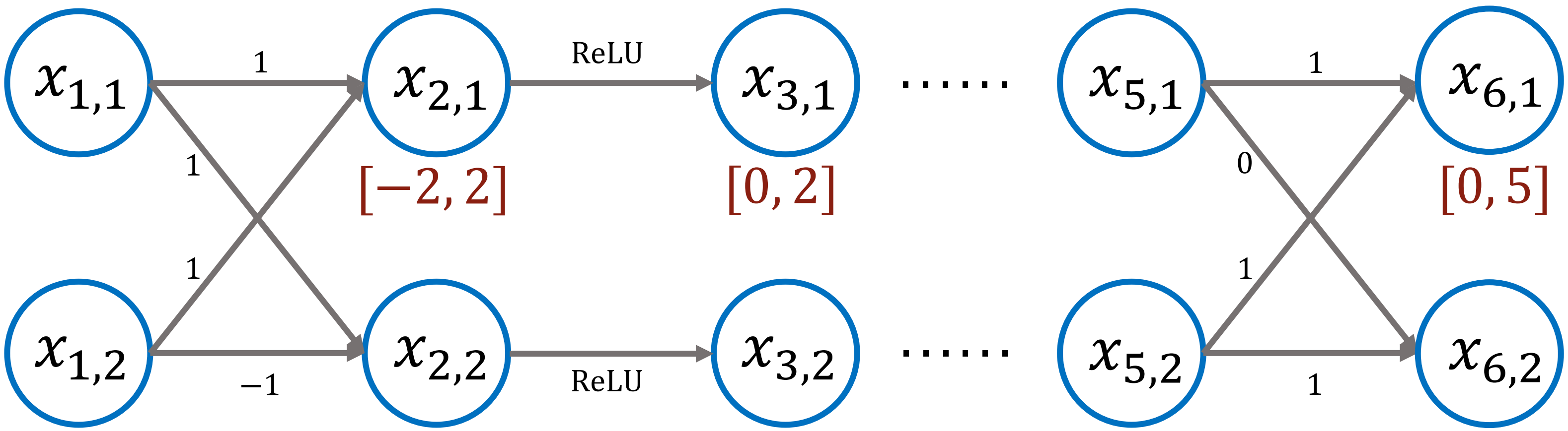
$x_{1,1} \geq -1$
$x_{1,1} \leq 1$

$x_{2,1} \geq x_{1,1} + x_{1,2}$
$x_{2,1} \leq x_{1,1} + x_{1,2}$

$x_{3,1} \geq 0$
$x_{3,1} \leq 0.5x_{2,1} + 1$

$x_{5,1} \geq x_{4,1}$
$x_{5,1} \leq x_{4,1}$
$x_{5,1} \geq 0$
$x_{5,1} \leq x_{1,1} + 2$

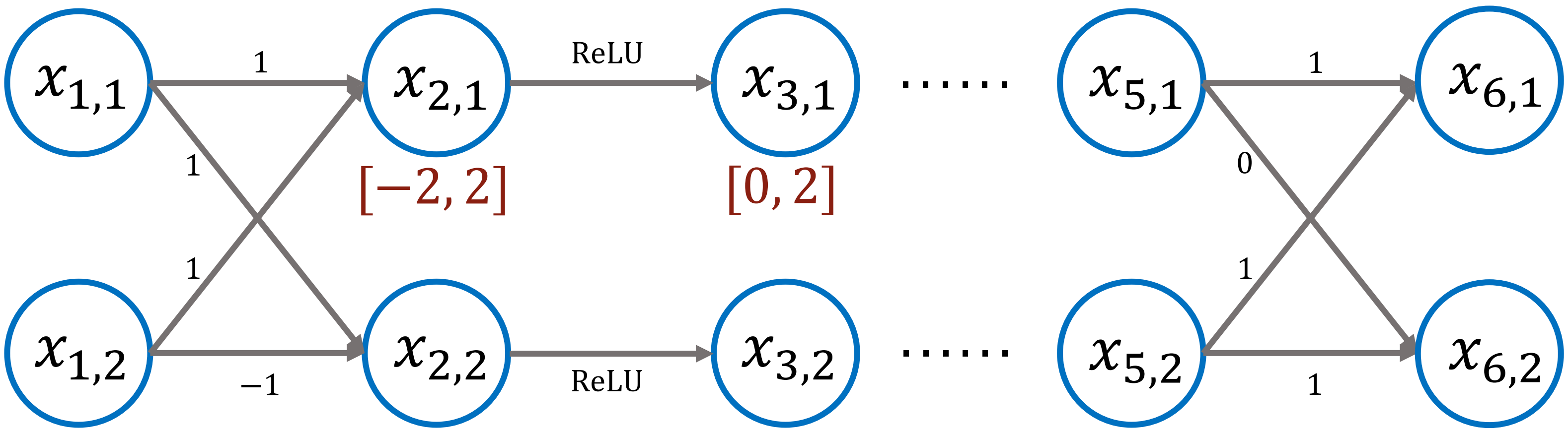
$x_{61} \geq x_{51} + x_{52}$
$x_{61} \leq x_{51} + x_{52}$
$x_{6,1} \geq 0$
$x_{6,1} \leq 1.25x_{1,1} + 0.25x_{1,2}$
$+3.5$



$x_{5,2} \geq 0$
$x_{5,2} \leq x_{4,2}$
$x_{5,1} \geq 0$
$x_{5,1} \leq 0.25x_{1,1} + 0.25x_{1,2} + 2$

App. I: Forward vs Backward

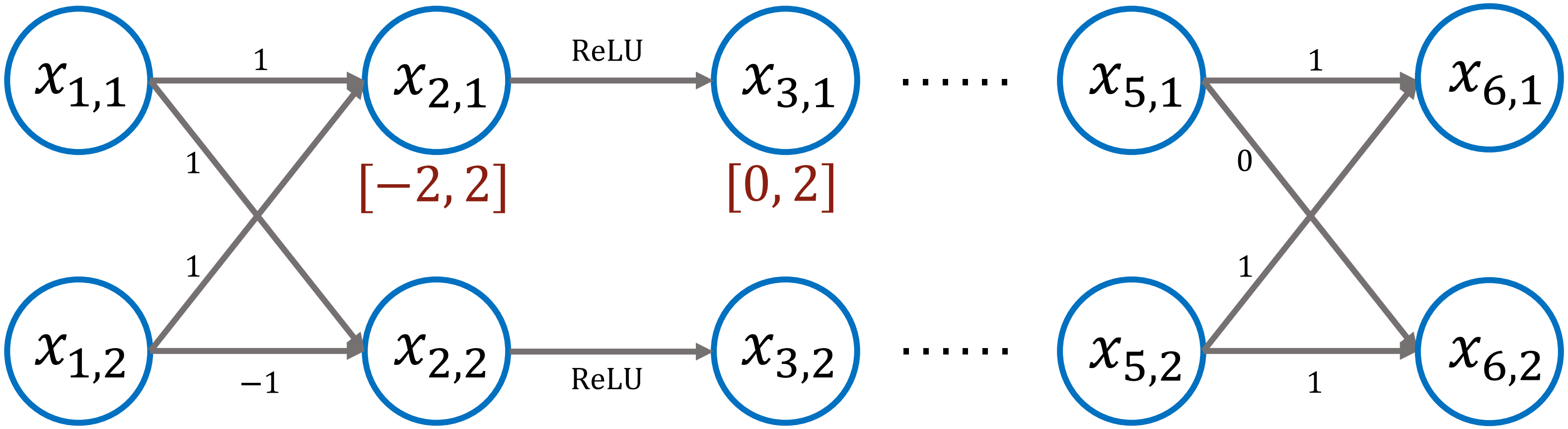
$x_{6,1} \leq x_{4,1} + 0.5x_{4,2} + 1$



App. I: Forward vs Backward

$$x_{6,1} \leq x_{3,1} + x_{3,2} + 0.5(x_{3,1} - x_{3,2}) + 1$$

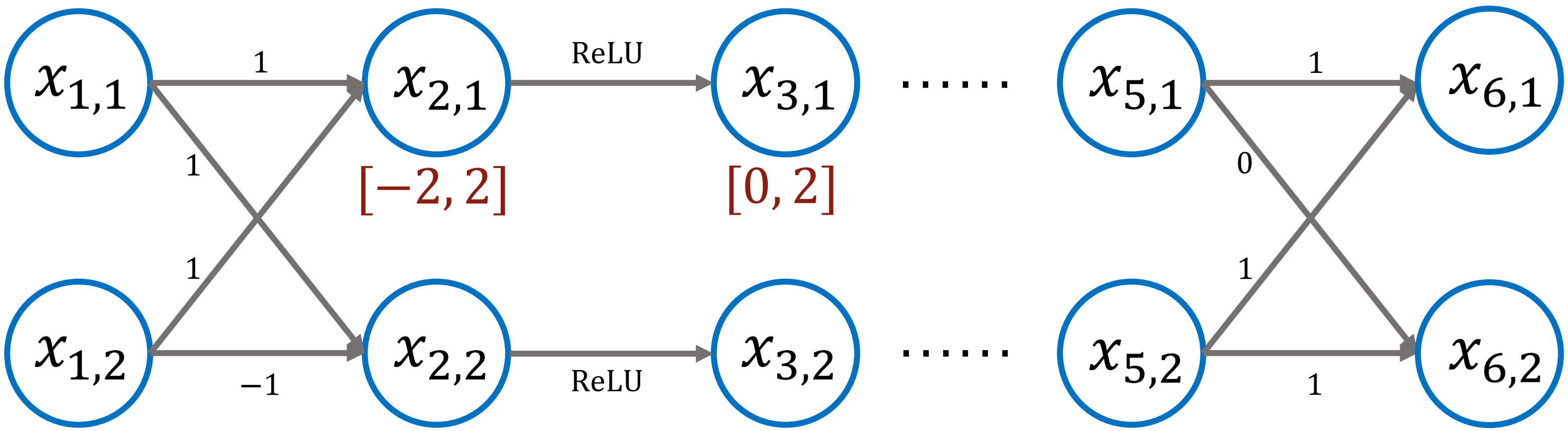
$$\leq 1.5x_{3,1} + 0.5x_{3,2} + 1$$



App. I: Forward vs Backward

$x_{6,1} \leq x_{1,1} + 0.5x_{1,2} + 3$

≤ 4.5



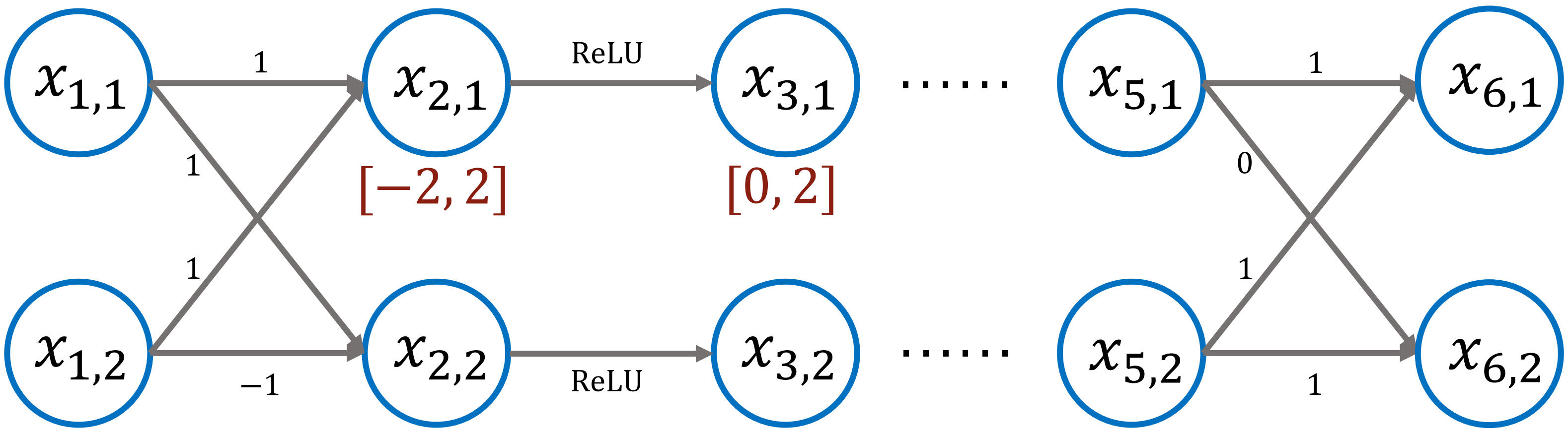
App. I: Forward vs Backward

$x_{6,1} \leq x_{1,1} + 0.5x_{1,2} + 3$

≤ 4.5

$x_{6,1} \leq 1.25x_{1,1} + 0.25x_{1,2} + 3.5$

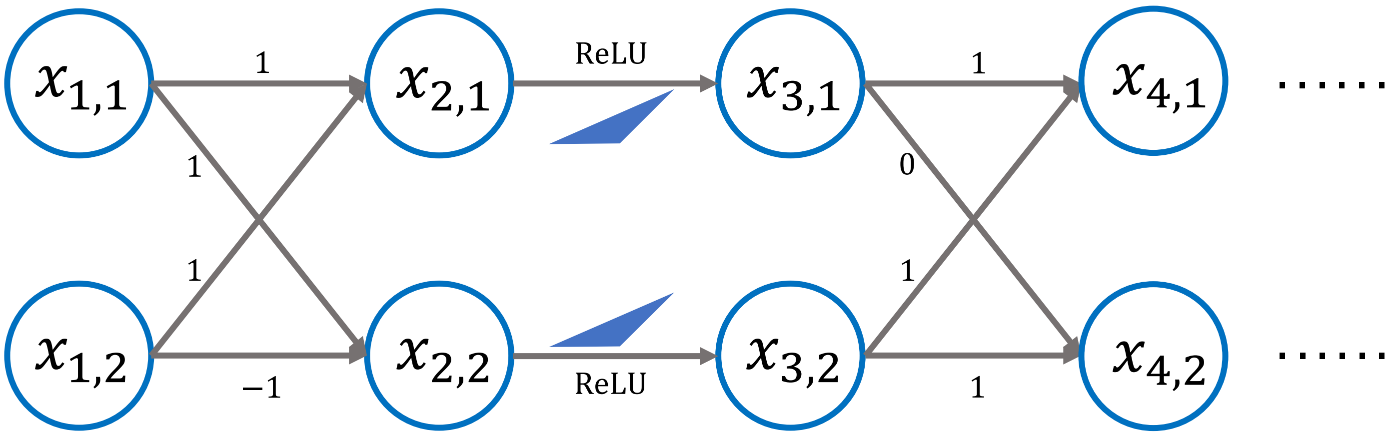
≤ 5



App. II: Exponential Constraints of LP

$x_{3,1} \geq 0$
$x_{3,1} \geq x_{2,1}$
$x_{3,1} \leq 0.5x_{2,1} + 1$

$x_{4,1} \geq x_{3,1} + x_{3,2}$

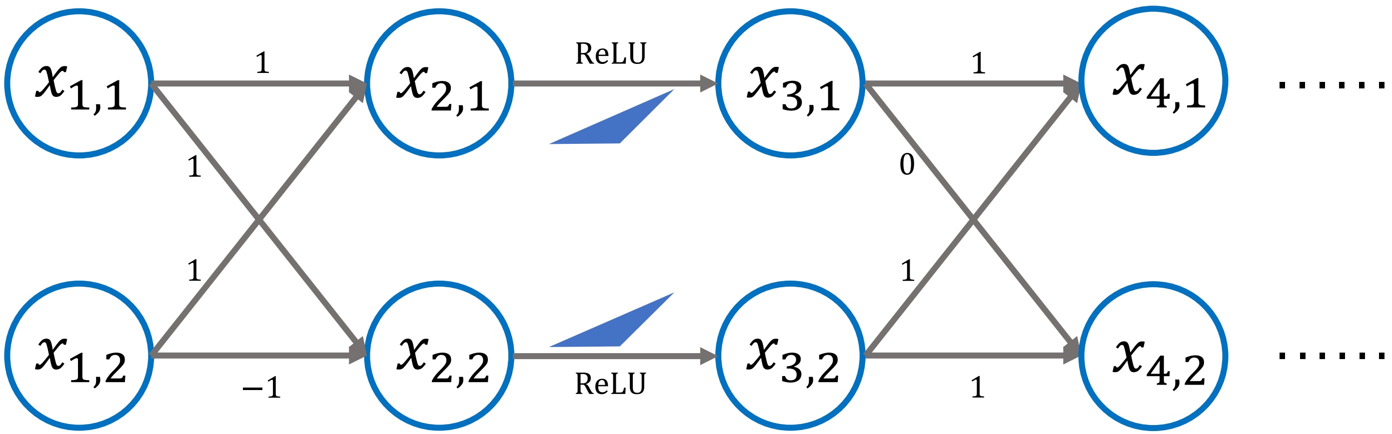


$x_{3,2} \geq 0$
$x_{3,2} \geq x_{2,2}$
$x_{3,2} \leq 0.5x_{2,2} + 1$

App. II: Exponential Constraints of LP

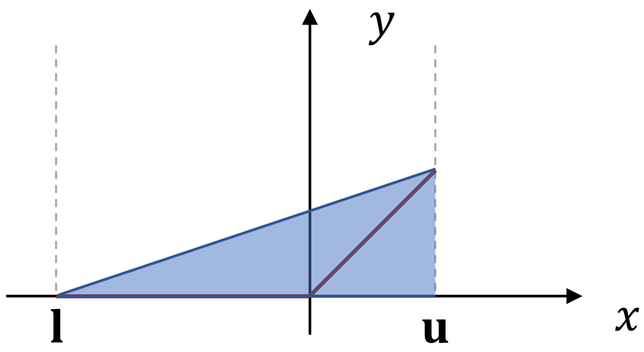
$x_{3,1} \geq 0$
$x_{3,1} \geq x_{2,1}$
$x_{3,1} \leq 0.5x_{2,1} + 1$

$x_{4,1} \geq 0$
$x_{4,1} \geq x_{2,1}$
$x_{4,1} \geq x_{2,2}$
$x_{4,1} \geq x_{2,1} + x_{2,2}$

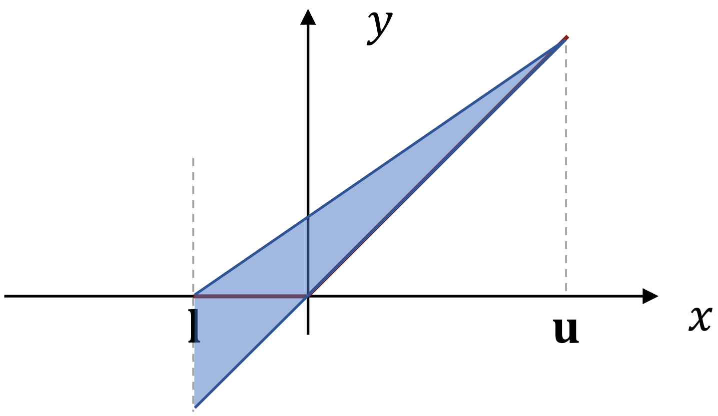


$x_{3,2} \geq 0$
$x_{3,2} \geq x_{2,2}$
$x_{3,2} \leq 0.5x_{2,2} + 1$

App. III: DeepPoly Heuristic



$|\mathbf{l}| \geq \mathbf{u} : k = 0$



$|\mathbf{l}| < \mathbf{u} : k = 1$

App. IV: About DeepPoly Heuristic

- Has DeepPoly get the best upper and lower bound lines?

  - No

  - Alpha-CROWN [*Kaidi Xu et al.*]

- Local optimum (DeepPoly heuristic) verse global optimum ($x_{i,j}$ bounds)

App. V: Experiments

- Implement multi-path back-propagation method as tool AbstraCMP
- Use robust radius as the accuracy metric
  - Larger robustness radius means higher over-approximation accuracy
  - Compare with single path method DeepPoly
  - Compare with LP-ALL, which solve LP for each node
- Binary search for robust radius δ

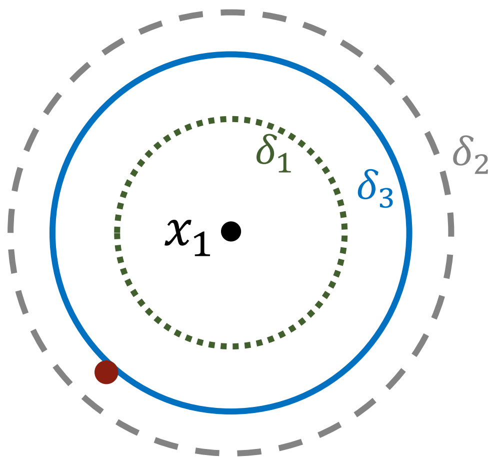
$f(x_1 + \boldsymbol{\delta_1}) \geq 0$ satisfied

$f(x_1 + \delta_2) \geq 0$ unsatisfied

$f(x_1 + \boldsymbol{\delta_3}) \geq 0$ satisfied

… …

$f(x_1 + \boldsymbol{\delta_k}) \geq 0$ satisfied $\wedge$ $f(x_1 + \boldsymbol{\delta_k} + \boldsymbol{\epsilon}) \geq 0$ unsatisfied



Code source: https://github.com/ZhengYeah/AbstraCMP