

Summary Slides

ERAN Verification and RefineZono Presentation




Presented by
Zhong Yuyi

Perturbation Type

- Perturbation on brightness: L_∞ -norm based
- Consider an attack where we allow a **small perturbation** to **every pixel** in the original image (visually corresponds to darkening or lightening the image)
- Each pixel has a number representing intensity
- Now, instead, each pixel contains an interval
- If each of these intervals has the same size, we say that we have formed an L_∞ ball around the image
- For each pixel x_i , $-\epsilon \leq x_i - x'_i \leq \epsilon$
- Difference for each pixel within range $[-\epsilon, \epsilon]$

Perturbation Type

- This L_∞ ball is captured visually by the Lower image (in which each pixel contains the smallest value allowed by its interval) and the Upper image (in which each pixel contains the largest value allowed by its interval)
- The robustness verification question is: are all possible images “sitting between” the Lower and the Upper image classified to the same label as the original?

Attack	Original	Lower	Upper
L_∞			

Robustness in ERAN

- Eg: MNIST dataset
- A set X containing 100 images from the test set, given **ground truth**
- An original image $x \in X$, apply L_∞ on x with certain ϵ .
- The corresponding adversarial region is P' , containing all different perturbations of x
- Robustness: all perturbed image $x' \in P'$ classified to the same label as the original x (which should be the ground truth label)
- Before verify robustness, first **verify** that the tested network classify the original image as the ground truth label.
- Filter out those images that were not classified correctly.

ERAN execution

- Eg: `python3 . --netname convSmallRELU_DiffAI.pyt --epsilon 0.004 --domain deepzono --dataset cifar10`
- Experiment setting from its paper, for each image x there exists three cases
 - Filtered out, label mismatch ground truth
 - Robustness verification succeed, NN must be robust (sound)
 - Verification failed, but actually NN could be robust (incomplete)

```
img 30 not considered, correct_label 6 classified label 3
concrete [-1.8273378212683842, -3.622761241667688, 3.3519219
334801469, 3.2914334909476857, 2.010352557373253, -0.41964450
5569975499, -1.177975902591683, -2.518936578594835]
img 31 not considered, correct_label 5 classified label 2
concrete [1.5357526492752576, -3.983708884056357, 0.29861862
4841953952, 2.442829616814501, 0.7193970744934612, 0.94151407
9064515527, 4.48845592618741, -4.383579676818473]
```

```
img 50 Verified 9
78.44384694099426 seconds
```

```
img 39 Failed
18.12131404876709 seconds
```

ERAN execution

- Eg: `python3 . --netname mnist_relu_3_50.tf --epsilon 0.01 --domain deepzono --dataset mnist`
- 100 Test images
- Output: The ratio of images on which the network is robust versus the number of images on which it classifies correctly
- `--epsilon 0.04`: analysis precision 15 / 98

```
img 98 Verified 6
1.121309518814087 seconds
concrete [-4.39738763444671, -9.640984699190296, -5.866119276169788, -0.537741
856627163, 4.934926703252896, -0.9465493323159262, -7.738737412613675, 4.624458
113321065, -0.6769881230043562, 16.21313839035016]
nlb [-4.829482379998386, -10.326812261672163, -6.443237348119291, -1.279255110
140948, 3.655260985224876, -1.8552252712734982, -8.324617764003928, 3.479649725
617553, -1.5012485964700923, 14.827295433528286] nub [-3.7289128093140538, -8
946216469798191, -5.085186393043873, 0.4487826878910087, 5.813180039948135, 0.1
980643180539779, -6.959664675741535, 5.526459876185256, 0.5317287222836923, 17.
05944531275716]
img 99 Verified 9
1.0472397804260254 seconds
analysis precision 95 / 98
```

ERAN VS MIPVerify

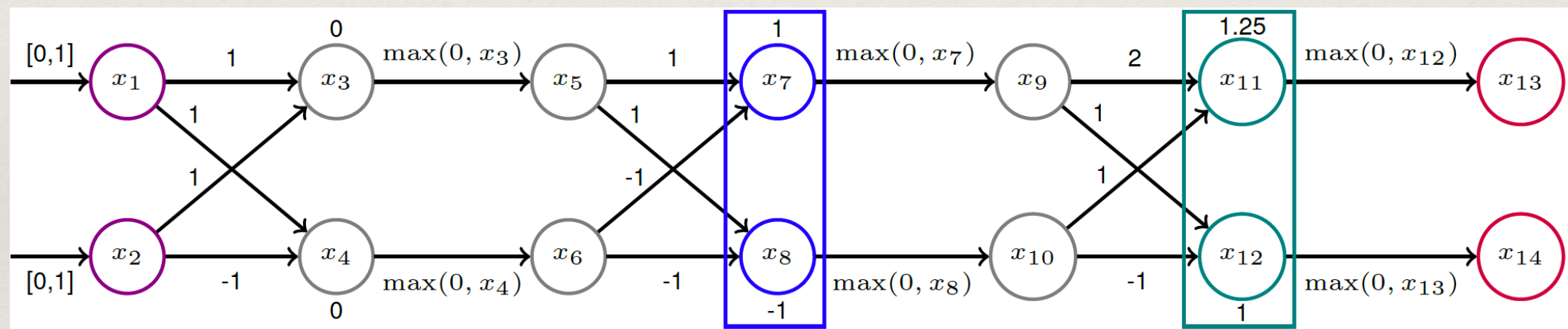
- MIPVerify (complete verifier based on MILP) can give adversarial examples when robustness fails
- Is possible to find adversarial image (counterexample) for incomplete method?

Over-approximation with MILP

- Gagandeep S, etc. Boosting Robustness Certification of Neural networks. ICLR 2019.
 - Combination of two papers:
 - Tjeng V, etc. Evaluating robustness of neural networks with mixed integer programming. ICLR 2019.
 - Gagandeep S, etc. Fast and Effective Robustness Certification. NIPS 2018.
1. A novel approach which combines scalable over-approximation methods with precise mixed integer linear programming.
 2. Leverage MILP to refine the bounds during over-approximation
 3. For feedforward and convolutional neural networks with RELU

Illustrative Example

- Two inputs to the network, both in $[0,1]$
- An input layer, two hidden layers, and one output layer.
- Separate each neuron into two parts: one represents the output of the affine transformation while the other captures the output of the ReLU activation.
- Verify that for any input in $[0,1] \times [0,1]$, the output at neuron x_{13} is greater than the output at x_{14}



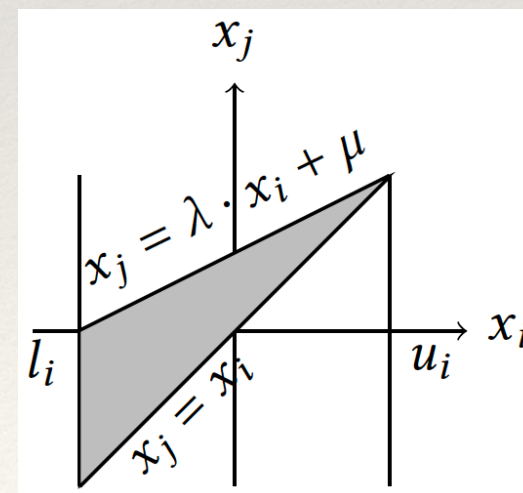
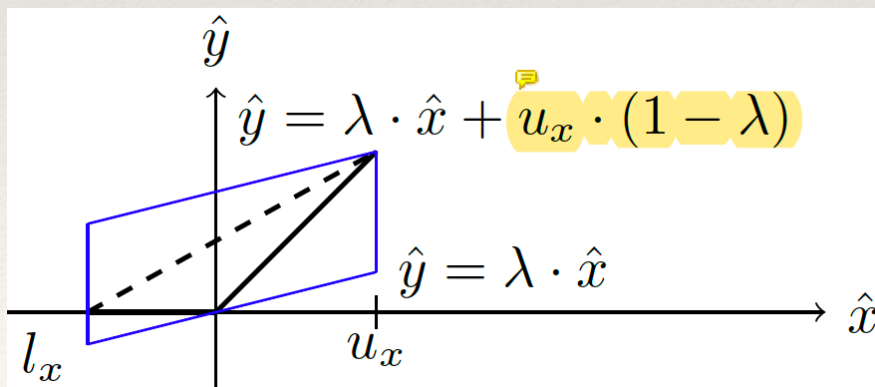
- Analysis leverages the Zonotope **domain** together with the abstract Zonotope **transformers** specialized to neural network activations as used in **DeepZ**.
- Refine the analysis results in blue box using **MILP** formulation used in MIPVerify

DeepZono Abstract Domain

- The Zonotope domain associates an affine form \hat{x} with each neuron x in the network

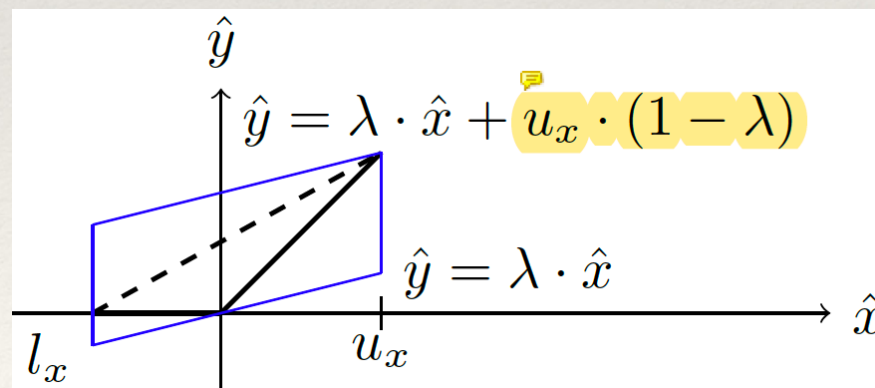
$$\hat{x} := c_0 + \sum_{i=1}^p c_i \cdot \eta_i$$

- $c_0, c_i \in R$ are real coefficients, and $\eta_i \in [s_i, t_i] \subseteq [-1, 1]$ are the **noise symbols**, which are shared between the affine forms for different neurons
- Recap:** DeepPoly abstract domain. Bounded within the interval and polyhedral upper bound and lower bound.
- DeepZono:** affine form \hat{x} and interval $[l_x, u_x]$
- The over-approximation of ReLU:** the shape of a parallelogram with two **vertical lines** (interval bound) and two **parallel lines of slope λ** , which is a parameter.

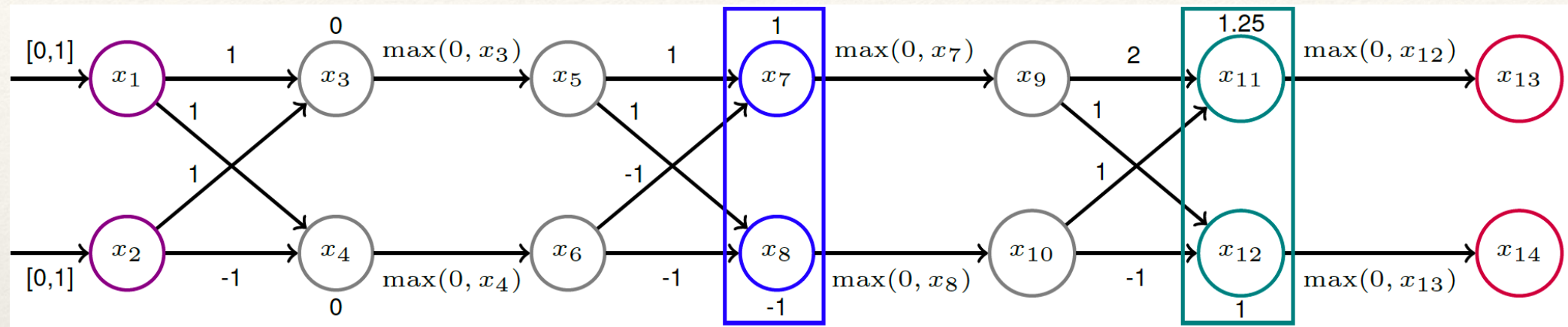


DeepZono Abstract Domain

- $y = \text{ReLU}(x)$, the affine form \hat{y} is given by
 - $\hat{y} = \hat{x}$, if $l_x > 0$
 - $\hat{y} = 0$, if $u_x < 0$
 - $\hat{y} = \lambda \cdot \hat{x} + \mu + \mu \cdot \epsilon_{new}$, otherwise
- $\lambda = \frac{u_x}{u_x - l_x}, \mu = -\frac{u_x \cdot l_x}{2 \cdot (u_x - l_x)}$ and introduce new noise symbol $\epsilon_{new} \in [-1, 1]$
- Extreme case: $\epsilon_{new} = -1$ corresponds the lower bound, $\epsilon_{new} = 1$ corresponds the upper bound



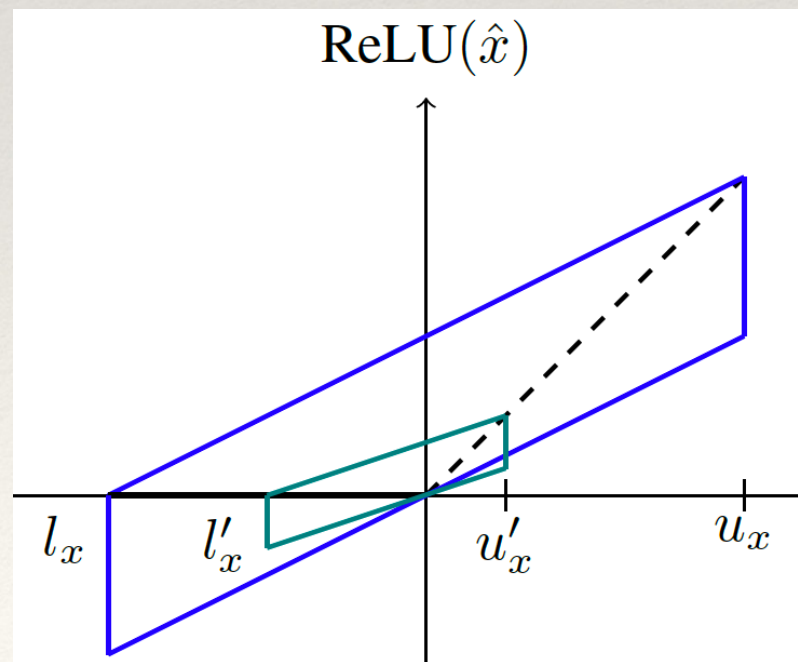
Zonotope Propagation



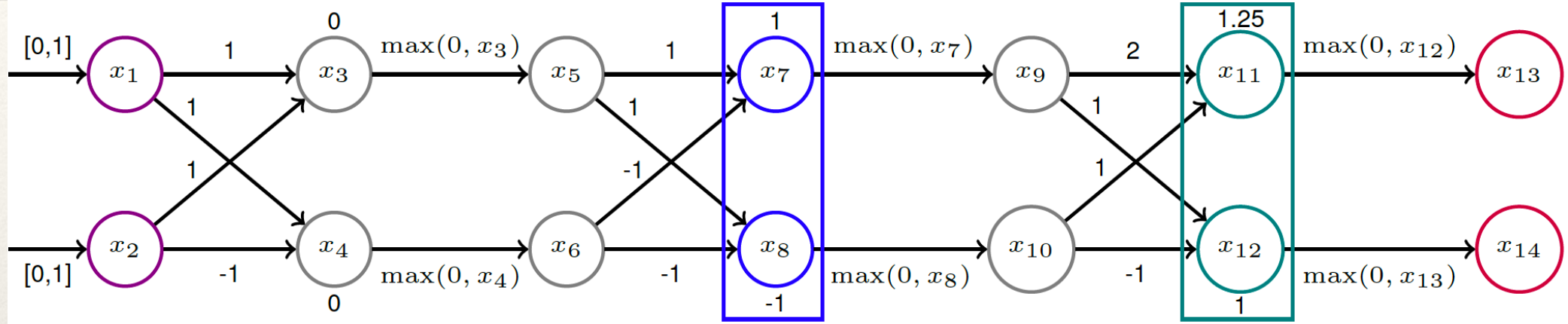
- Analysis **starts** by setting
 - $\hat{x}_1 = 0.5 + 0.5 \cdot \eta_1$, $l_1 = 0$, $u_1 = 1$
 - $\hat{x}_2 = 0.5 + 0.5 \cdot \eta_2$, $l_2 = 0$, $u_2 = 1$
- Apply exact **affine transformation**:
 - $\hat{x}_3 = \hat{x}_1 + \hat{x}_2 = 1 + 0.5 \cdot \eta_1 + 0.5 \cdot \eta_2$, $l_1 = 0$, $u_1 = 2$
 - $\hat{x}_4 = \hat{x}_1 - \hat{x}_2 = 0.5 \cdot \eta_1 - 0.5 \cdot \eta_2$, $l_1 = -1$, $u_1 = 1$
- Apply Zonotope ReLU transformer:
 - $\hat{x}_5 = \hat{x}_3$, $l_5 = l_3$, $u_5 = u_3$
 - $\hat{x}_6 = \lambda \cdot \hat{x}_4 + \mu + \mu \cdot \epsilon_{new} = 0.25 + 0.25 \cdot \eta_1 - 0.25 \cdot \eta_2 + 0.25 \cdot \eta_3$, $l_6 = -0.5$, $u_6 = 1$
 - Where $\lambda = \frac{1}{2}$ and $\mu = \frac{1}{4}$ while $\eta_3 \in [-1,1]$ is the new noise symbol

Zonotope Propagation

- $\hat{x}_6 = \lambda \cdot \hat{x}_4 + \mu + \mu \cdot \epsilon_{new} = 0.25 + 0.25 \cdot \eta_1 - 0.25 \cdot \eta_2 + 0.25 \cdot \eta_3, l_6 = -0.5, u_6 = 1$
- The blue zonotope approximation for x_6 permits negative values whereas x_6 can only take non-negative values in the concrete
- Over-approximation **accumulates as the analysis progresses deeper** into the network (each ReLU transformer will introduce more imprecision), resulting in overall imprecision and failure to prove properties that actually hold.



MILP-based Refinement

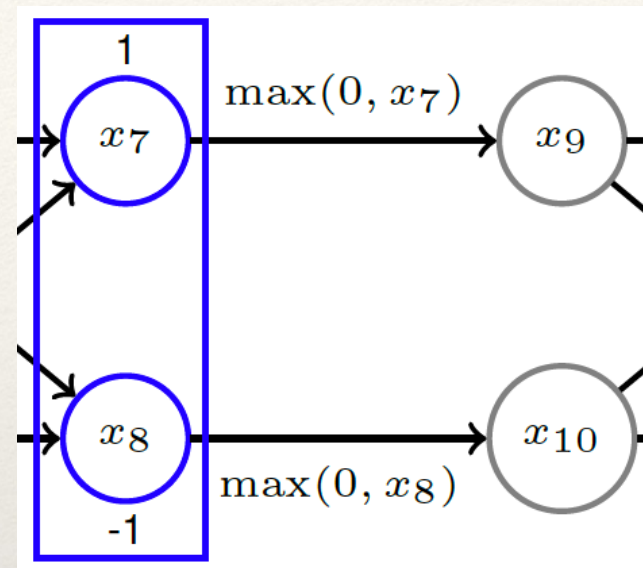
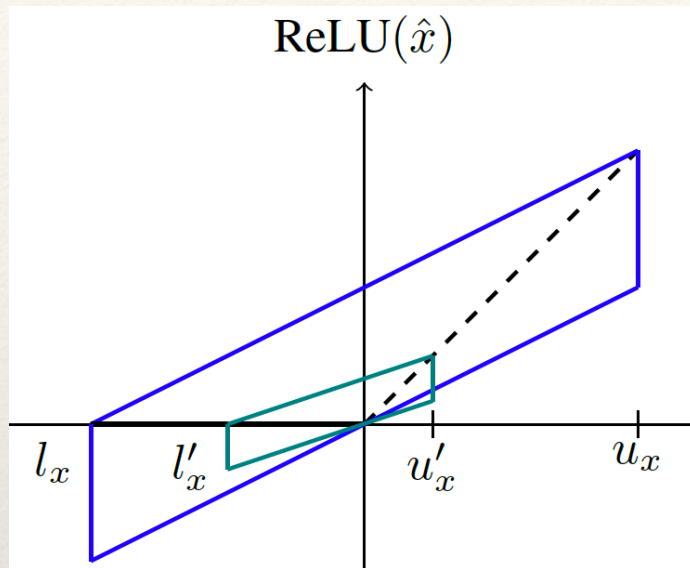


- Apply the second affine transformation
 - $\hat{x}_7 = \hat{x}_5 - \hat{x}_6 + 1 = 1.75 + 0.25 \cdot \eta_1 + 0.75 \cdot \eta_2 - 0.25 \cdot \eta_3, l_7 = 0.5, u_7 = 3$
 - $\hat{x}_8 = \hat{x}_5 - \hat{x}_6 - 1 = -0.25 + 0.25 \cdot \eta_1 + 0.75 \cdot \eta_2 - 0.25 \cdot \eta_3, l_8 = -1.5, u_8 = 1$
- Due to the approximation for x_6 , the bounds for x_7 and x_8 are imprecise.
- Refine the bounds for both x_7 and x_8 by **formulating the network** up to (and including) the second affine transformation **as a MILP instance** based on a formulation from Tjeng et al. (ICLR 2019).
- Compute refined bounds from MILP, respectively, update $l_7 = 1, l_8 = -1$

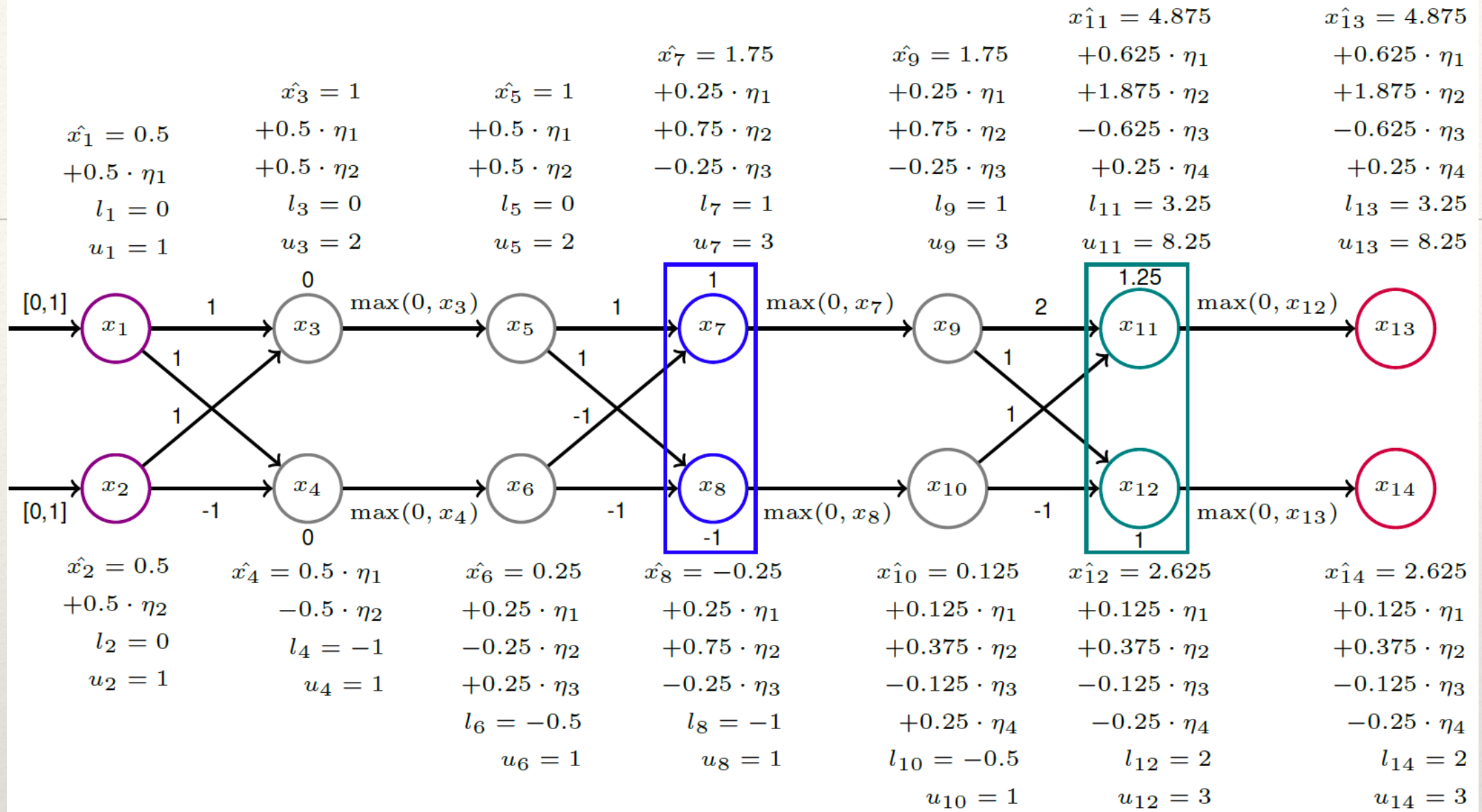
$$(x' \in (\mathcal{G}(x) \cap \mathcal{X}_{valid})) \wedge \left(f_{\lambda(x)}(x') < \max_{\mu \in [1, n] \setminus \{\lambda(x)\}} f_{\mu}(x') \right) \quad (2)$$

$$(y \leq x - l(1 - a)) \wedge (y \geq x) \wedge (y \leq u \cdot a) \wedge (y \geq 0) \wedge (a \in \{0, 1\}) \quad (6)$$

MILP-based Refinement



- The approximation shown in green is used
- This approximation has smaller area in the input-output plane compared to the blue one and thus reduces the approximation error.
- Apply ReLU transformation to x_{10}
 - $\hat{x}_{10} = \lambda \cdot \hat{x}_8 + \mu + \mu \cdot \epsilon_{new}$
 - Compute λ, μ using refine bound $[-1, 1]$ instead of $[-1.5, 1]$



- Refine the bounds for both x_{11} and x_{12}
- Compute refined bounds from MILP, respectively, update $l_7 = 1, l_8 = -1$
- Final layer:
 - $\hat{x}_{13} = \hat{x}_{11}, l_{13} = 3.25, u_{13} = 8.25$
 - $\hat{x}_{14} = \hat{x}_{12}, l_{13} = 2, u_{13} = 3$
 - $l_{13} > u_{12}$, robustness proved
 - DeepZ without refinement fails to prove robustness

Experiment

Dataset	Model	ϵ	<i>DeepZ</i>		<i>DeepPoly</i>		<i>RefineZono</i>	
			precision(%)	time(s)	precision(%)	time(s)	precision(%)	time(s)
MNIST	5×100	0.07	38	0.6	53	0.3	53	381
	6×100	0.02	31	0.6	47	0.2	67	194
	9×100	0.02	28	1.0	44	0.3	59	246
	6×200	0.015	13	1.8	32	0.5	39	567
	9×200	0.015	12	3.7	30	0.9	38	826
	ConvSmall	0.12	7	1.4	13	6.0	21	748
	ConvBig	0.2	79	7	78	61	80	193
	ConvSuper	0.1	97	133	97	400	97	665
CIFAR10	6×100	0.0012	31	4.0	46	0.6	46	765
	ConvSmall	0.03	17	5.8	21	20	21	550

- The number of neurons in the network is **not the determining factor** for the average Runtime of RefineZono.
- The average runtime on **ConvBig** network with 35K neurons, 6 layers and a perturbation region defined using $\epsilon = 0.2$ is almost 4 times less than on **ConvSmall** network with only 3 604 neurons, 3 layers and a smaller $\epsilon = 0.12$

Experiment

Dataset	Model	ϵ	<i>DeepZ</i>		<i>DeepPoly</i>		<i>RefineZono</i>	
			precision(%)	time(s)	precision(%)	time(s)	precision(%)	time(s)
MNIST	5×100	0.07	38	0.6	53	0.3	53	381
	6×100	0.02	31	0.6	47	0.2	67	194
	9×100	0.02	28	1.0	44	0.3	59	246
	6×200	0.015	13	1.8	32	0.5	39	567
	9×200	0.015	12	3.7	30	0.9	38	826
	ConvSmall	0.12	7	1.4	13	6.0	21	748
	ConvBig	0.2	79	7	78	61	80	193
	ConvSuper	0.1	97	133	97	400	97	665
CIFAR10	6×100	0.0012	31	4.0	46	0.6	46	765
	ConvSmall	0.03	17	5.8	21	20	21	550

- RefineZono runs faster on the networks trained to be robust
- This is because robust networks are relatively easier to certify and produce only a small number of candidate neurons for refinement, which are easier to refine by the solver
- (maybe filter out neurons whose activations are non-positive using abstract interpretation)