Beyond the Single Neuron Convex Barrier for Neural Network Certification

Gagandeep Singh, Rupanshu Ganvir, Martin Vechev, and Markus Püschel

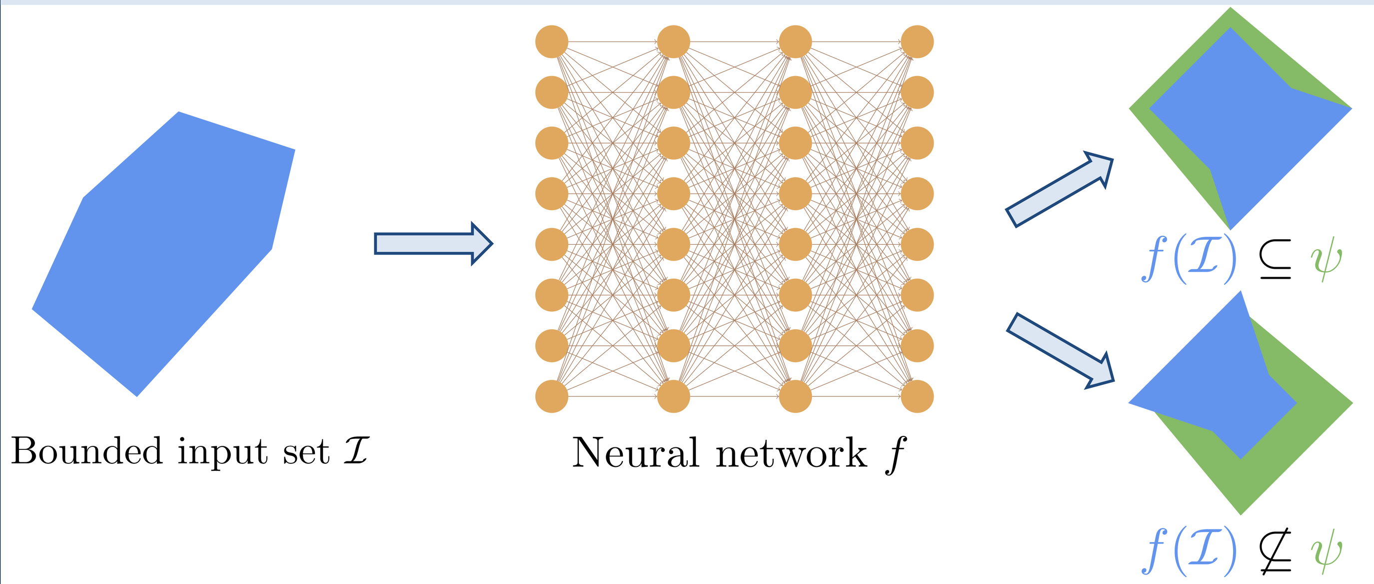
safeai.ethz.ch

Department of Computer Science
ETH zürich

Problem: Neural network certification

Inputs: Neural network f
Bounded input set $\mathcal{I}$
Safety property over outputs ψ

Output: if $f(\mathcal{I}) \subseteq \psi$, *Verified*
else, *Failed*



Example networks and inputs:

Image classification network f
Input $\mathcal{I}$ based on changes to pixel intensity
Input $\mathcal{I}$ based on geometric: e.g., rotation

Speech recognition network f
Input $\mathcal{I}$ based on added noise to audio signal

Aircraft collision avoidance network f
Input $\mathcal{I}$ based on input sensor values

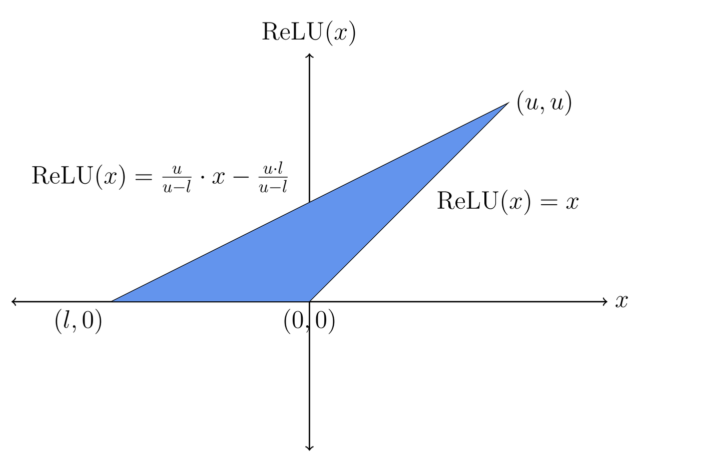
Example safety properties:

Robustness:
all inputs classify correctly
Stability:
$f(\mathcal{I})$ within a specified tolerance
Equivalence:
networks f_1, f_2 produce same outputs

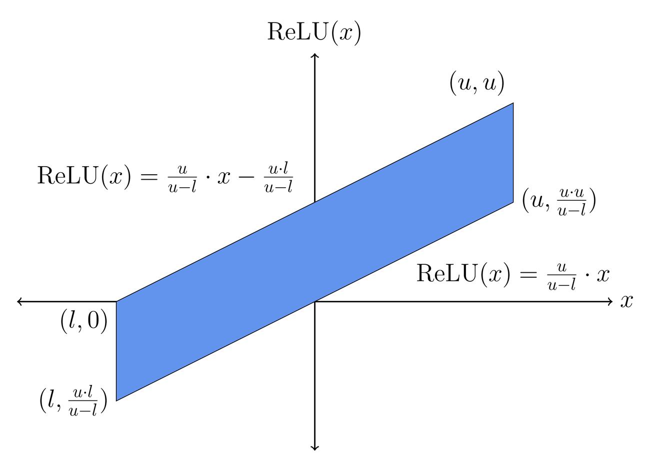
Exact certification of ReLU-based networks is NP-Complete

Single neuron convex relaxations of ReLU

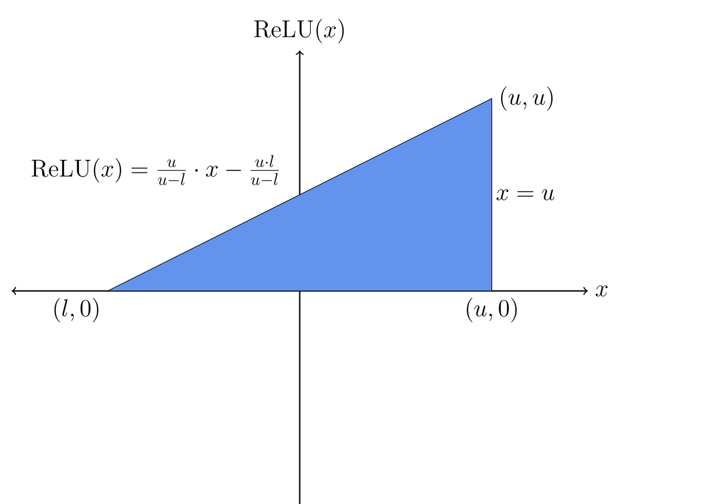
Input: $P_{\text{1-ReLU}} = \{l \leq x \leq u\}$ computed via a convex approximation method M



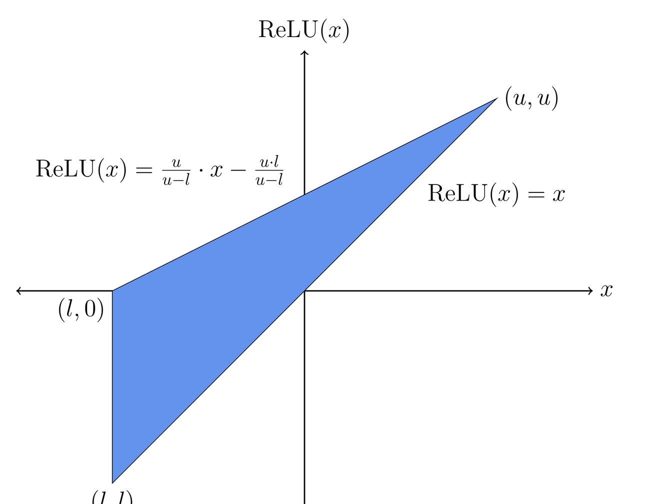
Triangle (1-ReLU) based best relaxation



DeepZ/Fast-Lin/Neurify



DeepPoly/CROWN



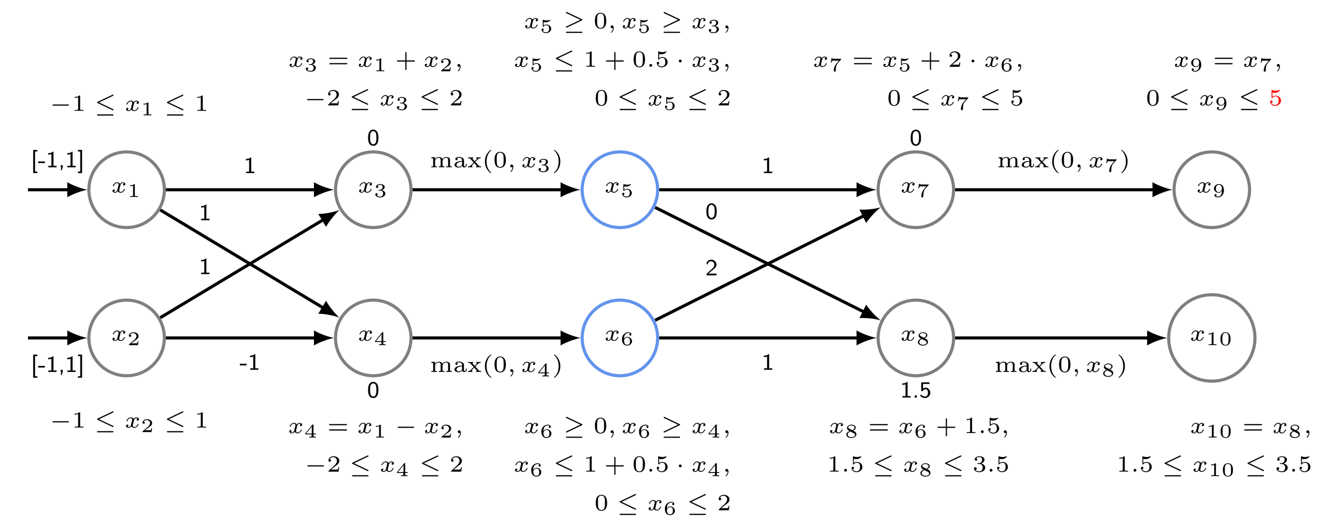
DeepPoly/CROWN

These relaxations can be quite imprecise as they ignore neuron dependencies

Our Contribution: Compute relaxations for multiple ReLUs jointly

Imprecision with 1-ReLU relaxation

Verify $x_9 \leq 4$ for all inputs $x_1, x_2 \in [-1, 1]$



Our k-ReLU framework

Given:

- n ReLU assignments $y_i{:=}\text{ReLU}(x_i)$, $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$.

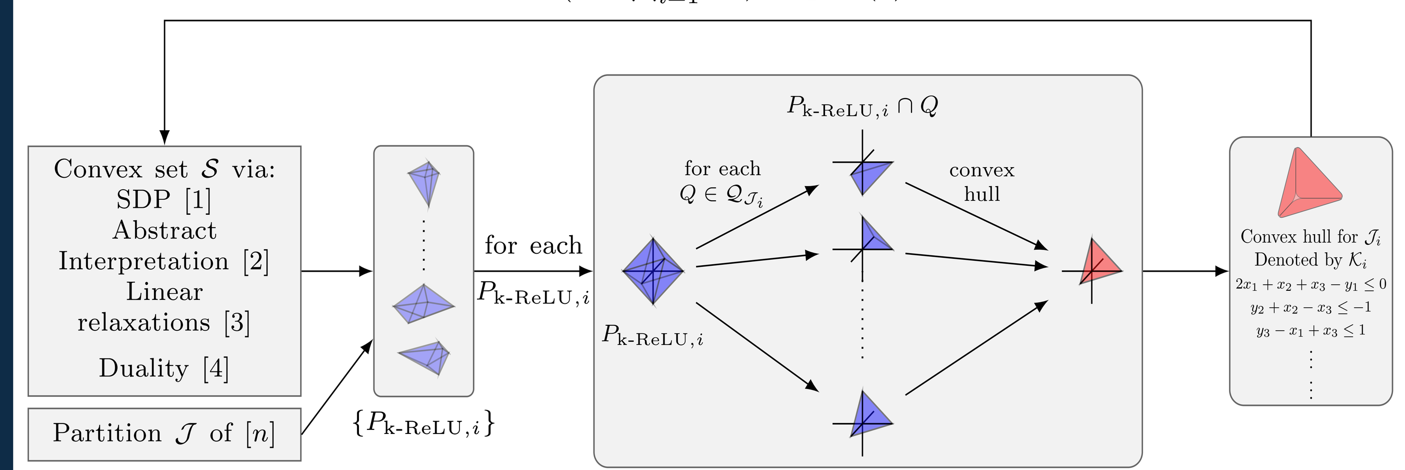
Steps:

1. Compute a convex overapproximation $\mathcal{S}$ wrt $\mathcal{I}$ of neuron values before the ReLU assignments via M.
2. Compute partition $\mathcal{J}$ of $[n]$ where each $\mathcal{J}_i \in \mathcal{J}$ contains k indices.
3. For each $\mathcal{J}_i$, compute polyhedron $P_{\text{k-ReLU},i}$ where
   - $P_{\text{k-ReLU},i}$ contains constraints over the neurons in $\mathcal{X}$ indexed by $\mathcal{J}_i$
   - $\mathcal{S} \subseteq P_{\text{k-ReLU},i}$
   - $P_{\text{k-ReLU},i} \subseteq \cap_{u \in \mathcal{J}_i} P_{\text{1-ReLU},u}$.
4. Using polyhedra $\mathcal{C}_i^+ = \{x_i \geq 0, y_i = x_i\}$, $\mathcal{C}_i^- = \{x_i \leq 0, y_i = 0\}$ induced by each $y_i{:=}\text{ReLU}(x_i)$, compute the set of polyhedra $\mathcal{Q}_{\mathcal{J}_i} = \{\bigcap_{u \in \mathcal{J}_i} \mathcal{C}_u^{s(u)} \mid s \in \mathcal{J}_i \to \{-,+\}\}$ for the k ReLU assignments induced by $\mathcal{J}_i$. Each polyhedron $Q \in \mathcal{Q}_{\mathcal{J}_i}$ corresponds to a branch produced by considering the k ReLU assignments jointly.
5. Our k-ReLU framework produces the following output convex relaxation:

$$\mathcal{S}_{\text{k-ReLU}} = \mathcal{S} \cap \bigcap_{i=1}^{n/k} \text{Conv}_{Q \in \mathcal{Q}_{\mathcal{J}_i}} (P_{\text{k-ReLU},i} \cap Q). \quad (1)$$

Instantiating k-ReLU framework

$(\mathcal{S} \cap \bigcap_{i=1}^{n/k} \mathcal{K}_i)$ as per (1)

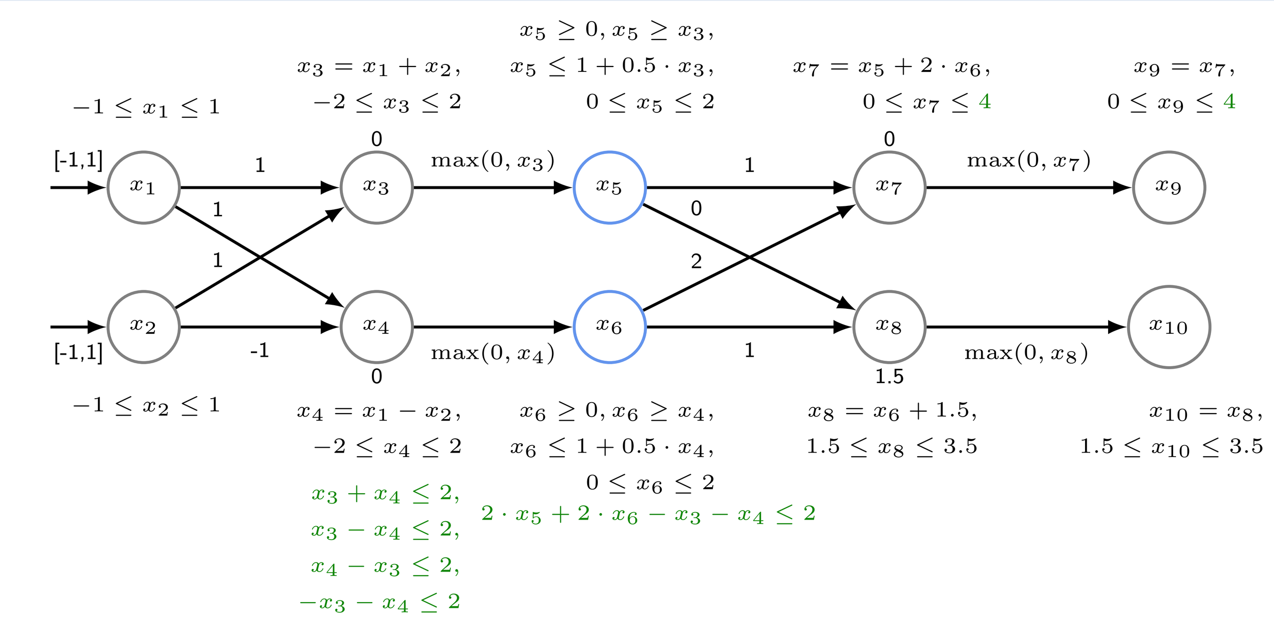


The result of (1) is optimal for the given choice of $\mathcal{S}$, k, $\mathcal{J}$, and $P_{\text{k-ReLU},i}$

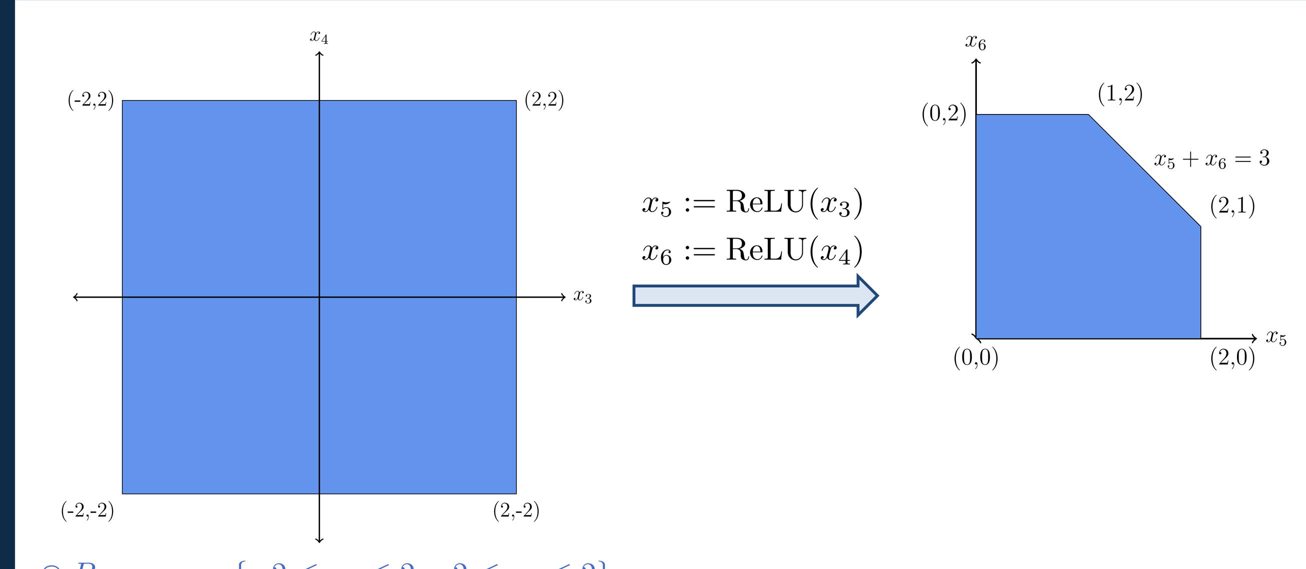
Theorem. *For $k > 1$ and a partition $\mathcal{J}$ of indices, if there exists a $\mathcal{J}_i$ for which $P_{\text{k-ReLU},i} \subsetneq \bigcap_{u \in \mathcal{J}_i} P_{\text{1-ReLU},u}$ holds, then $\mathcal{S}_{\text{k-ReLU}} \subsetneq \mathcal{S}_{\text{1-ReLU}}$.*

k-ReLU framework: State-of-the-art convex relaxations

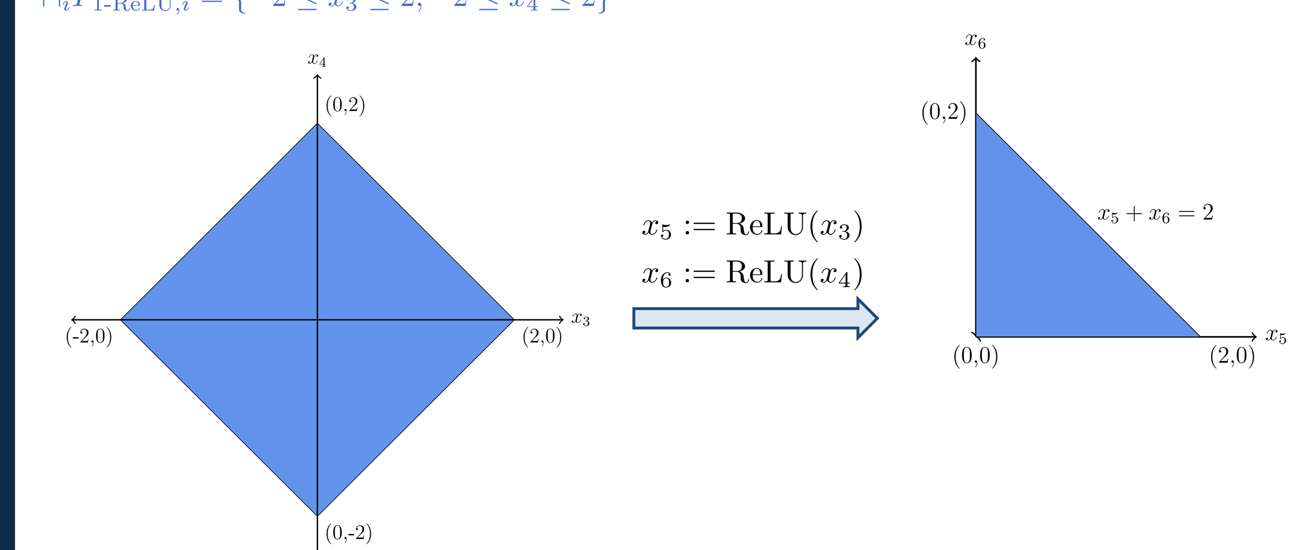
Precise results with 2-ReLU



2-ReLU vs 1-ReLU in the output plane



$\cap_i P_{\text{1-ReLU},i} = \{-2 \leq x_3 \leq 2, -2 \leq x_4 \leq 2\}$



$P_{\text{2-ReLU}} = \{-2 \leq x_3 \leq 2, -2 \leq x_4 \leq 2,$
$-2 \leq x_3 + x_4 \leq 2, -2 \leq x_3 - x_4 \leq 2\}$

Approximating optimal relaxations for larger k

Computing $\mathcal{K}_i$ involves 2^k convex hulls each of which has worst-case exponential cost in k

Steps:

1. Choose $2 \leq l < k$ and let $\mathcal{R}_i = \{\{j_1, \ldots, j_l\} \mid j_1, \ldots, j_l \in \mathcal{J}_i\}$ be the set containing all subsets of $\mathcal{J}_i$ with exactly l indices.
2. For each $R \in \mathcal{R}_i$, compute polyhedron $P'_{l\text{-ReLU},R}$ where
   - $P'_{l\text{-ReLU},R}$ contains constraints over the neurons in $\mathcal{X}$ indexed by R
   - $\mathcal{S} \subseteq P'_{l\text{-ReLU},R}$
   - $P'_{l\text{-ReLU},R} \subseteq \cap_{u \in R} P_{\text{1-ReLU},u}$.
3. The approximation $\mathcal{K}'_i$ is computed by applying l-ReLU $\binom{k}{l}$ times as:

$$\mathcal{K}'_i = \bigcap_{R \in \mathcal{R}_i} \text{Conv}_{Q \in \mathcal{Q}_R} (P'_{l\text{-ReLU},R} \cap Q).$$

Our verifier kPoly: State-of-the-art precision and scalability

k-ReLU parameter instantiation for kPoly

Parameter	Instantiation for kPoly
Approximation method M	DeepPoly
Partition $\mathcal{J}$	Group indices i where the triangle relaxation for y_i :=ReLU(x_i) has larger area in $x_i y_i$-plane
Polyhedron $P_{k-ReLU,i}$	Compute upper bounds for $\sum_{u \in \mathcal{J}_i} a_u . x_u$ wrt $\mathcal{S}$ via M where $a_u \in \{-1,0,1\}$

Benchmarks

Dataset	Model	Type	#neurons	Defense	k
MNIST	6x100	feedforward	610	None	3
	9x100	feedforward	910	None	2
	6x200	convolutional	1,210	None	2
	9x200	convolutional	1,810	None	2
	ConvSmall	convolutional	3,604	None	Adapt
	ConvBig	convolutional	34,688	[5]	5
CIFAR10	ConvSmall	convolutional	4,852	[6]	Adapt
	ConvBig	convolutional	62,464	[6]	5
	ResNet	residual	107,496	[4]	Adapt

- All CNNs and ResNet on a 2.6 GHz 14 core Intel Xeon CPU E5-2690
- All FNNs on a 3.3 GHz 10 core Intel i9-7900X Skylake CPU

Certifying network robustness wrt L_∞-ball (1000 test images)

MNIST Networks

Model	ϵ	DeepPoly		RefineZono		kPoly	
		# ✅	time(s)	# ✅	time(s)	# ✅	time(s)
6×100	0.026	160	0.3	312	310	441	307
9×100	0.026	182	0.4	304	411	369	171
6×200	0.015	292	0.5	341	570	574	187
9×200	0.015	259	0.9	316	860	506	464
ConvSmall	0.12	158	3	179	707	347	477
ConvBig	0.3	711	21	648	285	736	40

CIFAR10 Networks

Model	ϵ	DeepPoly		RefineZono		kPoly	
		# ✅	time(s)	# ✅	time(s)	# ✅	time(s)
ConvSmall	2/255	359	4	347	716	399	86
ConvBig	2/255	421	43	305	592	459	346
ResNet	8/255	243	12	243	27	245	91

Verifying MNIST ConvSmall robustness with k-ReLU vs 1-ReLU

- 100 L_∞ perturbation regions with $\epsilon = 0.12$
- kPoly with k-ReLU and 1-ReLU verifies 35 and 20 regions respectively

References:

[1] Semidefinite relaxations for certifying robustness to adversarial examples, NeurIPS'18
[2] An Abstract Domain for Certifying Neural Networks, POPL'19
[3] A convex relaxation barrier to tight robustness verification of neural networks, NeurIPS'19
[4] Provable defenses against adversarial examples via the convex outer adversarial polytope, ICML'18
[5] Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML'18
[6] Towards deep learning models resistant to adversarial attacks, ICLR'18