An Abstract Domain for Certifying Neural Networks

Gagandeep Singh  Timon Gehr  Markus Püschel  Martin Vechev

Department of Computer Science

ETHzürich
Adversarial input perturbations

\[ \mathcal{I} \]

\[ \mathcal{I} \in L_\infty(I_0, \epsilon) \]

\[ \mathcal{I} \in \text{Rotate}(I_0, \epsilon, \alpha, \beta) \]

Neural network \( f \)

8

7

9
Neural network robustness

Given: Neural network \( f : \mathbb{R}^m \rightarrow \mathbb{R}^n \)

Perturbation region \( \mathcal{R}(I_0, \phi) \)

1. \( L_\infty(I_0, \epsilon) \): All images \( I \) where pixel values in \( I \) and \( I_0 \) differ by at most \( \epsilon \)
2. \( \text{Rotate}(I_0, \epsilon, \alpha, \beta) \): All images \( I \) in \( L_\infty(I_0, \epsilon) \) rotated by \( \theta \in [\alpha, \beta] \)

To Prove: \( \forall I \in \mathcal{R}(I_0, \phi). f(I)[c] > f(I)[j] \)

where \( c \) is the correct output and \( j \) is any other output

Challenges

The size of \( \mathcal{R}(I_0, \phi) \) grows exponentially in the number of pixels:

- cannot compute \( f(I) \) for all \( I \) separately

Prior work on verification

- Precise but does not scale:
  - SMT solving [CAV’17]
  - input refinement [USENIX’18]
  - semidefinite relaxations [ICLR’18]

- Scales but imprecise
  - linear relaxations [ICML’18]
  - abstract interpretation [S&P’18, NIPS’18]
This work: contributions

A new abstract domain combining floating point Polyhedra with Intervals:
• custom transformers for common functions in neural networks such as affine transforms, ReLU, sigmoid, tanh, and maxpool activations
• scalable and precise analysis

First approach to certify robustness under rotation combined with linear interpolation:
• based on input refinement
• $\epsilon = 0.001, \alpha = -45^\circ, \beta = 65^\circ$

DeepPoly:
• complete and parallelized end-to-end implementation based on ELINA
• https://github.com/eth-sri/eran

<table>
<thead>
<tr>
<th>Network</th>
<th>$\epsilon$</th>
<th>NIPS’18</th>
<th>DeepPoly</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 layers</td>
<td>0.035</td>
<td>proves 21%</td>
<td>proves 64%</td>
</tr>
<tr>
<td>3010 neurons</td>
<td></td>
<td>15.8 sec</td>
<td>4.8 sec</td>
</tr>
<tr>
<td>6 layers</td>
<td>0.3</td>
<td>proves 37%</td>
<td>proves 43%</td>
</tr>
<tr>
<td>34,688 neurons</td>
<td></td>
<td>17 sec</td>
<td>88 sec</td>
</tr>
</tbody>
</table>
Neural network transformations

**Affine:** \( x_j \leftarrow v + \sum_{i \in [j-1]} w_i \cdot x_i \)

**ReLU:** \( x_j \leftarrow \max(0, x_i) \)

**Maxpool:** \( x_j \leftarrow \max_{i \in P} x_i \)

**Sigmoid:** \( x_j \leftarrow \frac{e^{x_i}}{e^{x_i} + 1} \)

**Tanh:** \( x_j \leftarrow \tanh(x_i) \)
Our Abstract Domain

Shape: associate a lower polyhedral $a_i^\leq$ and an upper polyhedral $a_i^\geq$ constraint with each $x_i$

$$a_i^\leq, a_i^\geq \in \{x \mapsto v + \sum_{j \in [i-1]} w_j \cdot x_j \mid v \in \mathbb{R} \cup \{-\infty, +\infty\}, w \in \mathbb{R}^{i-1}\} \text{ for } i \in [n]$$

Concretization of abstract element $a$:

$$\gamma_n(a) = \{x \in \mathbb{R}^n \mid \forall i \in [n]. a_i^\leq(x) \leq x_i \land a_i^\geq(x) \geq x_i\}$$

Domain invariant: store auxiliary concrete lower and upper bounds $l_i, u_i$ for each $x_i$

$$\gamma_n(a) \subseteq \times_{i \in [n]} [l_i, u_i]$$

- less precise than Polyhedra, restriction needed to ensure scalability
- captures affine transformation precisely unlike Octagon, TVPI
- custom transformers for ReLU, sigmoid, tanh, and maxpool activations

$\begin{array}{llll}
\text{Transformer} & \text{Polyhedra} & \text{Our domain} \\
\hline
\text{Affine} & 0(nm^2) & 0(w_{max}^2L) \\
\text{ReLU} & 0(\exp(n, m)) & 0(1) \\
\end{array}$
Example: Analysis of a Toy Neural Network

1. 4 constraints per neuron
2. Pointwise transformers => parallelizable.
4. Non-linear activations => approximate and minimize the area
\[
\begin{align*}
\langle x_1 \geq -1, & \quad \langle x_3 \geq x_1 + x_2, \\
\langle x_1 \leq 1, & \quad \langle x_3 \leq x_1 + x_2, \\
\langle l_1 = -1, & \quad \langle l_3 = -2, \\
\langle u_1 = 1 & \quad \langle u_3 = 2
\end{align*}
\]

\[
\begin{align*}
\langle x_2 \geq -1, & \quad \langle x_4 \geq x_1 - x_2, \\
\langle x_2 \leq 1, & \quad \langle x_4 \leq x_1 - x_2, \\
\langle l_2 = -1, & \quad \langle l_4 = -2, \\
\langle u_2 = 1 & \quad \langle u_4 = 2
\end{align*}
\]
ReLUs activation

Pointwise transformer for $x_j := \max(0, x_i)$ that uses $l_i, u_i$
if $u_i \leq 0$, $a_j^\leq = a_j^\geq = 0, l_j = u_j = 0$,
if $l_i \geq 0$, $a_j^\leq = a_j^\geq = x_i, l_j = l_i, u_j = u_i$,
if $l_i < 0$ and $u_i > 0$,
choose (b) or (c) depending on the area.

\[
\begin{align*}
\langle x_3 \geq x_1 + x_2, \ x_5 \geq 0, \\
x_3 \leq x_1 + x_2, \ x_5 \leq 0.5 \cdot x_3 + 1, \\
l_3 = -2, \ l_5 = 0, \\
u_3 = 2, \ u_5 = 2
\end{align*}
\]

\[
\begin{align*}
\langle x_4 \geq x_1 - x_2, \ x_6 \geq 0, \\
x_4 \leq x_1 - x_2, \ x_6 \leq 0.5 \cdot x_4 + 1, \\
l_4 = -2, \ l_6 = 0, \\
u_4 = 2, \ u_6 = 2
\end{align*}
\]
Affine transformation after ReLU

\[ \langle x_5 \geq 0, \quad l_5 = 0, \quad u_5 = 2 \rangle \]

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

Imprecise upper bound \( u_7 \) by substituting \( u_5, u_6 \) for \( x_5 \) and \( x_6 \) in \( a_{\geq}^{10} \).
Backsubstitution

\[
\begin{align*}
\langle x_5 \geq 0, \\
x_5 &\leq 0.5 \cdot x_3 + 1, \\
l_5 &= 0, \\
u_5 &= 2) \\
x_7 &\geq 0 \cdot x_5 + x_6, \\
x_7 &\leq 0 \cdot x_5 + x_6 + 0.5 \cdot x_4 + 2, \\
l_7 &= ?, \\
u_7 &= ?) \\
\langle x_6 \geq 0, \\
x_6 &\leq 0.5 \cdot x_4 + 1, \\
l_6 &= 0, \\
u_6 &= 2)
\end{align*}
\]
Affine transformation with backsubstitution is pointwise, complexity: $O(w_{max}^2 L)$.
\[
\begin{align*}
\langle x_1 \geq -1, \quad & \langle x_3 \geq x_1 + x_2, \quad \langle x_5 \geq 0, \quad \langle x_7 \geq x_5 + x_6, \quad \langle x_9 \geq x_7, \quad \langle x_{11} \geq x_9 + x_{10} + 1, \\
x_1 \leq 1, \quad & x_3 \leq x_1 + x_2, \quad x_5 \leq 0.5 \cdot x_3 + 1, \quad x_7 \leq x_5 + x_6, \quad x_9 \leq x_7, \quad x_{11} \leq x_9 + x_{10} + 1, \\
l_1 = -1, \quad & l_3 = -2, \quad l_5 = 0, \quad l_7 = 0, \quad l_9 = 0, \quad l_{11} = 1, \\
u_1 = 1 \rangle \quad & u_3 = 2 \rangle \quad u_5 = 2 \rangle \quad u_7 = 3 \rangle \quad u_9 = 3 \rangle \quad u_{11} = 5.5 \rangle
\end{align*}
\]
Checking for robustness

Prove $x_{11} - x_{12} > 0$ for all inputs in $[-1,1] \times [-1,1]$

$$\langle x_{11} \geq x_9 + x_{10} + 1, \quad \langle x_{12} \geq x_{10},$$
$$x_{11} \leq x_9 + x_{10} + 1, \quad x_{11} \leq x_{10},$$
$$l_{11} = 1, \quad l_{12} = 0,$$
$$u_{11} = 5.5 \rangle \quad u_{12} = 2 \rangle$$

Computing lower bound for $x_{11} - x_{12}$ using $l_{11}, u_{12}$ gives -1 which is an imprecise result

With backsubstitution, one gets 1 as the lower bound for $x_{11} - x_{12}$, proving robustness
More complex perturbations: rotations

Challenge: $\text{Rotate}(I_0, \epsilon, \alpha, \beta)$ is non-linear and cannot be captured in our domain unlike $L_\infty(I_0, \epsilon)$

Solution: Over-approximate $\text{Rotate}(I_0, \epsilon, \alpha, \beta)$ with boxes and use input refinement for precision

Result: Prove robustness for networks under $\text{Rotate}(I_0, 0.001, -45, 65)$
Experimental evaluation

- Neural network architectures:
  - fully connected feedforward (FFNN)
  - convolutional (CNN)

- Training:
  - trained to be robust with DiffAI [ICML’18] and PGD [Madry et al.]
  - without adversarial training

- Datasets:
  - MNIST
  - CIFAR10

- DeepPoly vs. state-of-the-art DeepZ [NIPS’18] and Fast-Lin [ICML’18]
Results

Verified robustness

(a) MNIST 6 × 100

Time (s)

(b) MNIST 6 × 100
MNIST FFNN (3,010 hidden units)

Verified robustness

(a) MNIST 6 × 500 ReLU

Time (s)

(b) MNIST 6 × 500 ReLU
CIFAR10 CNNs (4,852 hidden units)
### Large Defended CNNs (6 layers)
trained via DiffAI [ICML’18]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>#neurons</th>
<th>$\epsilon$</th>
<th>%verified robustness</th>
<th>Average runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNIST</td>
<td>ConvBig</td>
<td>34,688</td>
<td>0.1</td>
<td>97</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>ConvBig</td>
<td>34,688</td>
<td>0.2</td>
<td>79</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>ConvBig</td>
<td>34,688</td>
<td>0.3</td>
<td>37</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>ConvSuper</td>
<td>88,500</td>
<td>0.1</td>
<td>97</td>
<td>133</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>ConvBig</td>
<td>62,464</td>
<td>0.006</td>
<td>50</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>ConvBig</td>
<td>62,464</td>
<td>0.008</td>
<td>33</td>
<td>46</td>
</tr>
</tbody>
</table>
Ongoing work

• Combine abstract interpretation with MILP/LP solvers [ICLR’19]
• Geometric perturbations
• Applying verification during training and improving DiffAI
• Beyond classification
• Others…….
Conclusion

A new abstract domain combining floating point Polyhedra with Intervals:

\[ n: \#\text{neurons}, m: \#\text{constraints} \]

\[ w_{\text{max}}: \text{max } \#\text{neurons in a layer}, L: \#\text{layers} \]

<table>
<thead>
<tr>
<th>Transformer</th>
<th>Polyhedra</th>
<th>Our domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affine</td>
<td>(0(nm^2))</td>
<td>(0(w_{\text{max}}^2L))</td>
</tr>
<tr>
<td>ReLU</td>
<td>(0(\exp(n,m)))</td>
<td>(0(1))</td>
</tr>
</tbody>
</table>

Adversarial input perturbations

- Complete and parallelized end-to-end implementation based on ELINA
- https://github.com/eth-sri/eran

DeepPoly:
- Complete and parallelized end-to-end implementation based on ELINA
- https://github.com/eth-sri/eran