Fast and Effective Robustness Certification of Neural Networks
Gagandeep Singh, Timon Gehr, Matthew Mirman, Martin Vechev, and Markus Püschel
safei.ethz.ch

Problem: Certification of neural network robustness
Small input perturbations can cause neural networks to misclassify.

The neural network classifies the input image as 8.

When each pixel in \( I_0 \) is perturbed by \( \epsilon \), the neural network misclassifies the perturbed image as 7 even though \( \epsilon \) appears as 8 to the human eye.

Our goal is to certify that a given neural network correctly classifies all images in a small \( \mathbb{B}(\epsilon) \) around \( I_0 \), i.e., all images \( I \) where each pixel in \( I \) has a distance of at most \( \epsilon \) from the corresponding pixel in \( I_0 \).

Abstract interpretation for robustness certification
Abstract interpretation is a framework for overapproximating concrete properties.

In this work, we use the Zonotope abstraction [1] for robustness certification.

References:
[1] The Zonotope Abstract for Neural Networks with AdaBoost, CVPR 2017
[2] Towards Fast Computation of Certified Robustness for Neural Networks, ICML 2018
[4] Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML 2018
[5] Boosting Adversarial Attacks with Momentum, CVPR 2018

Main Contribution: Optimal Zonotope transformers

Zonotope abstraction
The Zonotope abstraction associates an affine form \( z \) containing noise symbols \( \eta_i \) with each variable \( z_i \):

\[
\begin{align*}
\bar{z}_i &= \sum_{j=1}^{n} x_j \cdot \eta_j, \\
\tilde{z}_i &= \frac{1}{2} \left( \bar{z}_i + z_i \right), \\
\mathcal{F}_i &= \mathcal{F}_i(\bar{z}_i, \tilde{z}_i)
\end{align*}
\]

Above, \( \mathcal{F}_i \) is the zonotope of \( \bar{z}_i, \tilde{z}_i \) is the partial deviations from the center. We also store the interval concentration \( [\bar{z}_i, \tilde{z}_i] \) of \( z_i \).

Zonotopes for robustness certification
- Capture the input shape exactly
- Fast and exact for affine transformation which is a common transformation
- No backpropagation [2] is required
- Prior work [2]: optimal ReLU transformer
- No sigmoid or tanh transformers

Our optimal ReLU transformer

\[
\begin{align*}
\mathcal{F}_i(\bar{z}_i, \tilde{z}_i) &= \begin{cases}
\bar{z}_i - \tilde{z}_i & \text{if } \bar{z}_i > \tilde{z}_i \\
\tilde{z}_i - \bar{z}_i & \text{otherwise}
\end{cases}
\end{align*}
\]

Our optimal sigmoid and tanh transformers

\[
\begin{align*}
\mathcal{F}_i(\bar{z}_i, \tilde{z}_i) &= \begin{cases}
\bar{z}_i - \frac{\tilde{z}_i}{1 + e^{-\epsilon}} & \text{if } \bar{z}_i > \tilde{z}_i \\
\frac{\tilde{z}_i}{1 + e^{-\epsilon}} - \bar{z}_i & \text{otherwise}
\end{cases}
\end{align*}
\]

DeepZ: Our system for neural network robustness

Illustration on a toy feedforward network with ReLU

- Batch sequential and parallelized implementations available
- Evaluation: feedforward networks on a 3.3 GHz 10 core Intel i7-7700K Skylake CPU
- Convolutional networks on a 2.6 GHz 14 core Intel Xeon CPU E5-2690

Network architectures

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Type</th>
<th>#Hidden units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>FFNNSmall</td>
<td>feedforward</td>
<td>610</td>
</tr>
<tr>
<td>MNIST</td>
<td>FFNNBig</td>
<td>feedforward</td>
<td>3,010</td>
</tr>
<tr>
<td>ConvMed</td>
<td>convolutional</td>
<td>4,804</td>
<td></td>
</tr>
<tr>
<td>ConvBig</td>
<td>convolutional</td>
<td>34,688</td>
<td></td>
</tr>
<tr>
<td>ConvSuper</td>
<td>convolutional</td>
<td>88,500</td>
<td></td>
</tr>
<tr>
<td>Skip</td>
<td>skipnet</td>
<td>71,650</td>
<td></td>
</tr>
<tr>
<td>CIFAR10</td>
<td>ConvSmall</td>
<td>convolutional</td>
<td>4,852</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>ConvBig</td>
<td>convolutional</td>
<td>62,464</td>
</tr>
</tbody>
</table>

- We used networks trained with and without adversarial training
- For adversarial training we used DNN [4] and PGD [2] (parameterized by \( \epsilon \))

Comparison of state-of-the-art on the MNIST FFNNSmall ReLU network

- DeepZ vs Fast-Lin [2] and \( Local[\epsilon] \) with serialized implementations
- First 100 images from each dataset were used for evaluation
- \( x \)-axis shows the radius \( \epsilon \) of \( \mathbb{B}(\epsilon) \)

Results with DeepZ: State-of-the-art precision and scalability

Average runtime is \( \leq 22 \) seconds on all networks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>( \epsilon )</th>
<th>% verified</th>
<th>Avg. runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>ConvBig</td>
<td>0.1</td>
<td>97</td>
<td>9</td>
</tr>
<tr>
<td>ConvBig</td>
<td>0.2</td>
<td>79</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>ConvBig</td>
<td>0.3</td>
<td>37</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>ConvSuper</td>
<td>0.1</td>
<td>97</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td>Skip</td>
<td>0.1</td>
<td>95</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>CIFAR10</td>
<td>ConvBig</td>
<td>0.006</td>
<td>50</td>
<td>39</td>
</tr>
<tr>
<td>ConvBig</td>
<td>0.008</td>
<td>33</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>