Fast and Effective Robustness Certification of Neural Networks

Gagandeep Singh, Timon Gehr, Matthew Mirman, Martin Vechev, and Markus Püschel

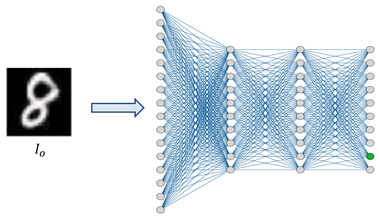
safeai.ethz.ch

Department of Computer Science
ETH zürich

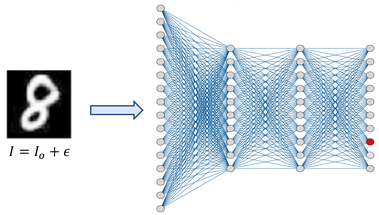
Problem: Certification of neural network robustness

Small input perturbations can cause neural networks to misclassify

L_∞-norm based perturbation



The neural network classifies the image l_o correctly as 8

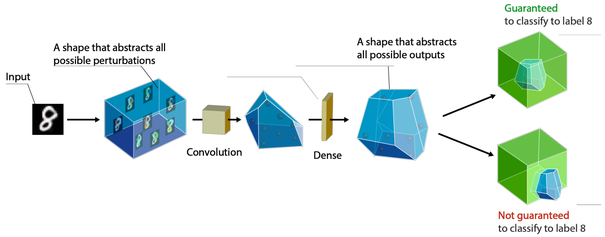


When each pixel in I_o is perturbed by ϵ, the neural network misclassifies the perturbed image I as 7 even though I appears as 8 to the human eye

Our goal: certify if a given neural network correctly classifies all images I in the ϵ-ball $\mathcal{B}_{(l_0,\infty)}(\epsilon)$ around I_0, i.e., all images I where each pixel in I has a distance of at most ϵ from the corresponding pixel in I_o.

Abstract interpretation for robustness certification

Abstract interpretation is a framework for over approximating concrete properties



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

References:
[1] The Zonotope Abstract Domain Taylor1+, CAV'09
[2] Towards Fast Computation of Certified Robustness for ReLU Networks, ICML'18
[3] AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, IEEE S&P 2018
[4] Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML'18
[5] Boosting Adversarial Attacks with Momentum, CVPR'18

Main Contribution: Optimal Zonotope transformers

Zonotope abstraction

The Zonotope abstraction associates an affine form $\hat{x}$ containing p noise symbols η_i with each variable x:

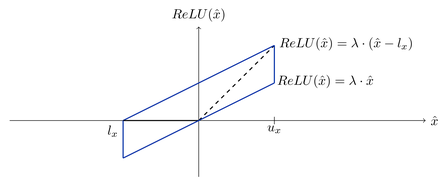
$$\hat{x} := \alpha_0 + \sum_{i=1}^{p} \alpha_i \cdot \eta_i, \qquad \text{where } \alpha_0, \alpha_i \in \mathbb{R}, \eta_i \in [a_i, b_i] \subseteq [-1, 1]$$

Above, $\alpha_0$is the center of $\hat{x}$ and α_i are the partial deviations from the center.We also store the interval concretization $[l_x, u_x]$ of $\hat{x}$.

Zonotopes for robustness certification

- Captures the input shape exactly
- Fast and exact for affine transformation which is a common transformation
  - no backpropagation [2] is required
- Prior work [3]
  - suboptimal ReLU transformer
  - no sigmoid or tanh transformers

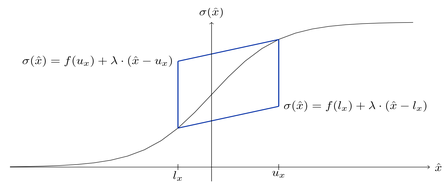
Our optimal ReLU transformer



Theorem. *Let $\mathcal{Z}$ be the input to a ReLU function. Consider the set of pointwise Zonotope approximations $\mathcal{O}$ of the output that only alter the affine form $ReLU(\hat{x})$ in the output. Let $\lambda = \frac{u_x}{u_x - l_x}$, $\mu = -\frac{u_x \cdot l_x}{2 \cdot (u_x - l_x)}$, and $\eta_{new} \in [-1, 1]$ be a new noise symbol. The new affine form $ReLU(\hat{x})$ in the output with the minimum area in the input-output plane is given by:*

$$ReLU(\hat{x}) = \begin{cases} \hat{x}, & \text{if } l_x > 0, \\ 0, & \text{if } u_x \leq 0, \\ \lambda \cdot \hat{x} + \mu + \mu \cdot \eta_{new}, & \text{otherwise.} \end{cases}$$

Our optimal sigmoid and tanh transformers



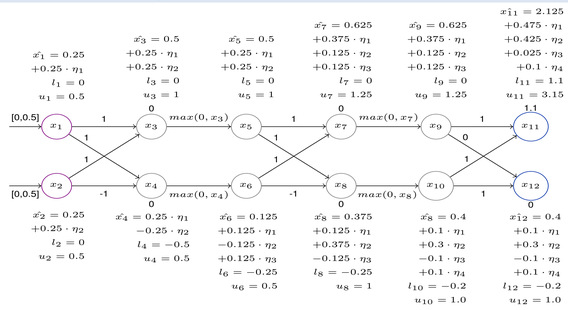
Theorem. *Let $\mathcal{Z}$ be the input to a smooth S-shaped function $\sigma(x)$ (such as the sigmoid or tanh function). Consider the set of pointwise Zonotope approximations $\mathcal{O}$ of the output that only alter the affine form $\sigma(\hat{x})$ in the output and where the box concretization of $\sigma(\hat{x})$ satisfies $l_{\sigma(\hat{x})} = \sigma(l_x), u_{\sigma(\hat{x})} = \sigma(u_x)$. Let $\lambda = \min(\sigma'(l_x), \sigma'(u_x))$, $\mu_1 = \frac{\sigma(u_x) + \sigma(l_x) - \lambda \cdot (u_x + l_x)}{2}$, $\mu_2 = \frac{\sigma(u_x) - \sigma(l_x) - \lambda \cdot (u_x - l_x)}{2}$, and $\eta_{new} \in [-1, 1]$ be a new noise symbol. The new affine form $\sigma(\hat{x})$ in the output with the minimum area in the input-output plane is given by:*

$$\sigma(\hat{x}) = \begin{cases} \sigma(u_x), & \text{if } l_x = u_x, \\ \lambda \cdot \hat{x} + \mu_1 + \mu_2 \cdot \eta_{new}, & \text{otherwise,} \end{cases}$$

Our transformers are sound with respect to floating point arithmetic

DeepZ: Our system for neural network robustness

Illustration on a toy feedforward network with ReLU



Implementation

- Both sequential and parallelized implementations available
- Evaluation:
  - feedforward networks on a 3.3 GHz 10 core Intel i9-7900X Skylake CPU
  - convolutional networks on a 2.6 GHz 14 core Intel Xeon CPU E5-2690

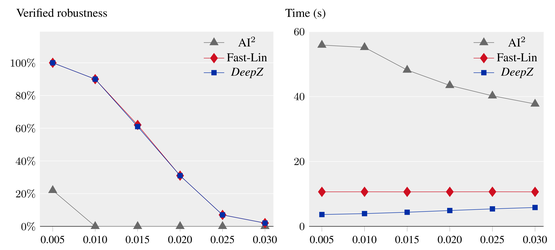
Network architectures

Dataset	Model	Type	#hidden units
MNIST	FFNNSmall	feedforward	610
	FFNNBig	feedforward	3,010
	ConvMed	convolutional	4,804
	ConvBig	convolutional	34,688
	ConvSuper	convolutional	88,500
	Skip	skipnet	71,650
CIFAR10	ConvSmall	convolutional	4,852
	ConvBig	convolutional	62,464

- We used networks trained with and without adversarial training
- For adversarial training, we used DiffAI [4] and PGD [5] (parameterized by ϵ)

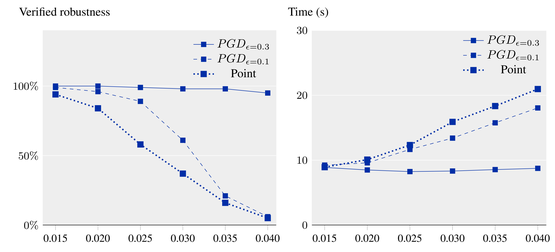
Comparison with state-of-the-art on the MNIST FFNNSmall ReLU network

- DeepZ vs Fast-Lin [2] and AI^2[3] with serialized implementations
- First 100 images from each dataset were used for evaluation
- x-axis shows the radius ϵ of $\mathcal{B}_{(l_0,\infty)}(\epsilon)$

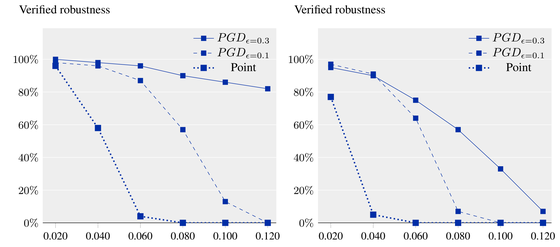


Results with DeepZ: State-of-the-art precision and scalability

MNIST FFNNBig ReLU networks

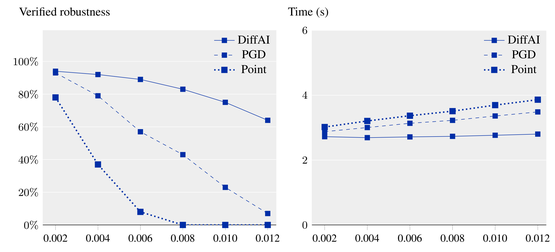


MNIST ConvMed sigmoid and tanh networks



Average runtime is $\leq$ 22 seconds on all networks

CIFAR10 ConvSmall ReLU networks



DeepZ results on large defended RELU networks trained with DiffAI

Dataset	Model	ϵ	% verified	Avg. runtime (s)
MNIST	ConvBig	0.1	97	5
	ConvBig	0.2	79	7
	ConvBig	0.3	37	17
	ConvSuper	0.1	97	133
	Skip	0.1	95	29
CIFAR10	ConvBig	0.006	50	39
	ConvBig	0.008	33	46