RAI Project 2019

Robustness certification based on DeepZ relaxation with optimized parameters

Group 024

Version from
December 19, 2019

Abstract

The aim of the project was to create a robustness certification algorithm for neural networks based on the DeepZ relaxation. The developed algorithm utilizes the non-comparability of different ReLU representations in the DeepZ approach by finding the optimal parameters for each specific certification task. The analysed networks consist of adapted fully connected, convolutional and ReLU layers, which were defined to proagate the DeepZ representation efficiently.

Vincent Bardenhagen
& Jim Buffat
Statistics
& Remote Sensing

DeepZ Approach

The DeepZ relaxation relies on the Zonotope representation to propagate perturbations bounded by an L_∞ norm in the input domain through the network. Any point $\vec{x} = (x_1, \ldots, x_d)$ inside a d-dimensional zonotope satisfies

$$x_i = a_{i0} + \sum_{1 \le k \le K} a_{ik}\epsilon_k = a_{ik}\epsilon_k \quad \forall i$$

where $\epsilon_{k \ge 1} \in [-1, 1]$ and $\epsilon_0 = 1$. That is, the zonotope is completely described by the parameters $\{a_{ij}\}_{1 \le i \le d}^{0 \le j \le K}$. This zonotope representation will be denoted by N_{dK}.

Affine Transformation

The DeepZ approach is exact for affine transformations. Let $x^L \in \mathbf{R}^{d_L}$ the Lth layer consisting of d_L neurons. If x^L is a point in the zonotope $N_{d_L K}$ then $A : x^L \to \hat{x}^{L+1}$ induces a map $N_{d_L K} \to \hat{N}_{d_{L+1} K}$ such that $x^L \in N_{d_L K} \iff A(x^L) = \hat{x}^{L+1} \in \hat{N}_{d_{L+1} K}$. Hence, the affine transformation acts on the factor representation of the zonotopes.
In the algorithm the zonotope $N_{d_L K}$ is represented by a tensor with dimension $torch.size([K, d_L])$ for fully connected and $torch.size([K, c, h, w])$ for convolutional layers. Hence, the algorithm can mostly use the standard implementations of the linear and convolutional transformations with the batch dimension representing the factors, the bias being the zeroth entry, i.e. $\{\hat{a}^L_{i0}\}_{1 \le i \le d_{L+1}}$.

ReLU Transformation

For the layer ReLU : $\hat{x}^{L+1} \to x^{L+1} = \max(0, \hat{x}^{L+1})$, there is no exact map $\hat{N}_{d_{L+1} K} \to N_{d_{L+1} K'}$ in general. We therefore constrain the problem to finding any map $\hat{N}_{d_{L+1} K} \to N_{d_{L+1} K'}$ such that $\hat{x}^{L+1} \in \hat{N}_{d_{L+1} K} \implies x^{L+1} \in N_{d_{L+1} K'}$. Such a map is called a *sound* transformer. DeepZ proposes a family of sound transformers parametrized by a slope $\lambda \in [0, 1]$ such that for each $x_i^{L+1} = \mathrm{ReLU}(\hat{x}_i^{L+1})$ we have

$$a_{ij}^{L+1} = \begin{cases} \hat{a}_{i0}^{L+1}[0 \le l_i^{L+1}] + \left(\lambda_i^{L+1}\hat{a}_{i0}^{L+1} + d_i^{L+1}/2\right)\left[l_i^{L+1} \le 0 \le u_i^{L+1}\right] & j = 0 \\ \hat{a}_{ij}^{L+1}[0 \le l_i^{L+1}] + \lambda_i^{L+1}\hat{a}_{ij}^{L+1}\left[l_i^{L+1} \le 0 \le u_i^{L+1}\right] & 1 \le j \le K \\ d_i^{L+1}/2\left[l_i^{L+1} \le 0 \le u_i^{L+1}\right] & j = K + i \end{cases}$$

with

$$\begin{aligned} d_i^{L+1} &= \max(u_i^{L+1}(1 - \lambda_i^{L+1}), -l_i^{L+1}\lambda_i^{L+1}) \\ l_i^{L+1} &= \hat{a}_{i0}^{L+1} - \sum_{1 \le j \le K} |\hat{a}_{ij}^{L+1}| \\ u_i^{L+1} &= \hat{a}_{i0}^{L+1} + \sum_{1 \le j \le K} |\hat{a}_{ij}^{L+1}| \end{aligned}$$

(Note that dimensions with $a_{ij}^{L+1} = 0 \ \forall i$ can be pruned.) The algorithm reflects this formulation. The main challenge is to map the factors of the additional dimensions of the zonotope correctly to the matrix representation of $\hat{N}_{d_{L+1}}$. In the algorithm each ReLU layer contains a parameter tensor of lambdas containing all lambdas that could be added in the respective layer. Thus, the dimension of the tensors are $[1, d_{l+1}]$ for fully connected networks and $[1, c, h, w]$ for the convolutional nets. When added to the matrix representation of $\hat{N}_{d_{L+1}}$ each λ_i^{L+1} must be added to its own batch dimension. This extension is computed by a multiplication with a tensor that is one at all relevant locations and zero else.

Last Layer, Loss and Optimization

The elements of the output layer M represent the pairwise difference of the output for the target label o_j and all other labels o_i, as each zonotope in the output layer is an overapproximation of the L_∞ region in the input domain. Hence, we solve multiple optimizations of the pairwise losses, that we define from the following reasoning

$$\begin{aligned} o_j > o_i &\iff x_j^M(\vec{\epsilon}) > x_i^M(\vec{\epsilon}) \ \forall \vec{\epsilon} \in [-1,1]^{K^M} \\ &\iff \min_{\vec{\epsilon}} \ x_j^M(\vec{\epsilon}) - x_i^M(\vec{\epsilon}) = \min_{\vec{\epsilon}} \ x_i^{M+1}(\vec{\epsilon}) = l_i^{M+1} > 0 \end{aligned}$$

It is sufficient to find one set $\lambda = \{\lambda_i^l\}_{1\le i\le d_L}^{0\le l\le L_n}$ with L_n the number of ReLU layers in the network for each pairwise comparison such that $l_i^{M+1}(\lambda) > 0$. Hence, we optimize the *pairwise* losses $\mathcal{L}_i = -l_i^{M+1}$.

To ensure faster convergence, we start by optimizing a *global* loss $\mathcal{L} = \sum_{i\in V} \mathcal{L}_i$, where V is the set of unverified indices (for which $\mathcal{L}_i < 0$ was never observed). We train with this loss at a higher learning rate and reset the optimizer when V changes. When the global loss doesn't decrease in at least one term (i.e. $\exists \mathcal{L}_i \in \mathcal{L}^{t+1}$ and $\exists \mathcal{L}_i' \in \mathcal{L}^t$ such that $\mathcal{L}_i > \mathcal{L}_i'$) we switch to the pairwise loss to verify the remaining digits in V.

Input

The input image contains grey-scale values $a_{i0}^0 \in [0,1]$ and the perturbed image $a_{i0}^0 + a_{ii}^0 \epsilon_i$ is constrained to this domain as well. This information can be used to derive a smaller, but equivalent input zonotope $\tilde{a}_{i0}^0 + \tilde{a}_{ii}^0 \epsilon_i$ with

$$\tilde{a}_{i0}^0 = \begin{cases} a_{i0}^0 & \text{if} \quad \nu \le a_{i0} \le 1-\nu \\ \frac{(a_{i0}+\nu)}{2} & \text{if} \quad 0 \le a_{i0} \le \nu \\ \frac{1+a_{i0}-\nu}{2} & \text{if} \quad 1-\nu \le a_{i0} \le 1 \end{cases} \qquad \tilde{a}_{ii}^0 = \begin{cases} \nu & \text{if} \quad \nu \le a_{i0} \le 1-\nu \\ \frac{(a_{i0}+\nu)}{2} & \text{if} \quad 0 \le a_{i0} \le \nu \\ \frac{1-a_{i0}+\nu}{2} & \text{if} \quad 1-\nu \le a_{i0} \le 1 \end{cases}$$

Result

With our approach we are sound in all test cases and additional examples created with PGD attacks on the networks. We are able to verify all original test examples as well as all additional examples from the preliminary submission well under the time limit on our local machines.