Corroborating On the Robustness of Neural Networks

Dipam Paul

I. Introduction

We present a precise and sound automated verifier for proving robustness of neural networks with rectified linear activations (ReLU) against adversarial attacks based on L_∞ ball perturbations. The verifier uses Zonotope [1] abstraction to soundly approximate the behavior of the network as precisely as possible.

II. Overview

The zonotope domain is exact for affine transformations but loses precision for ReLU activations. The ReLU transformer is parameterized by the slope λ and is sound for any value $\lambda \in [0, 1]$. Heuristically [1] the value of λ is chosen as $\lambda = u_i/(u_i - l_i)$ as it minimizes the area of output zonotope[1]. However, there may exist transformers with different λ, which can verify more precisely. The choice of optimal lambda should be a function of input image and L_∞ ball radius rather than minimizing area.

We propose gradient learning based approach to find optimal λ for each neuron in the network. The objective is to make the existing implementation more precise. We create a network with custom layers that propagates the abstract Zonotope of the input image through a given network. Constructed network for FC1 is shown in Fig. 1.

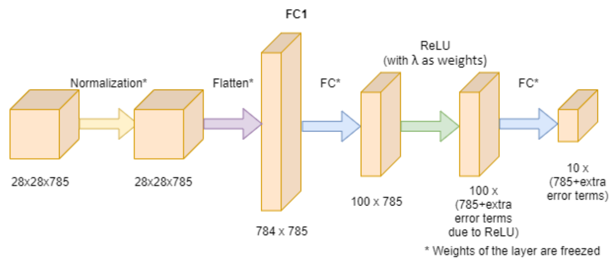

Fig. 1. Fully Connected Network 1 (FC1)

The input zonotope is nothing but an L_∞ ball of radius $?$ around the input x. The network has λ's as the learnable parameters for the gradient descent.

III. Methodology

Below is a list of propositions and improvements that were implemented in the course of developing our verifier.

A. Loss Function

The objective of training is to output zonotope such that box interval of target class, $I_t = [l_t, u_t]$ is strictly greater than $I_{t^0} = [l_{t^0}, u_{t^0}]$ where t^0 is an output class such that $t^0 \neq t$. The most intuitive loss function is the one that measures the gap between I_t and I_{t^0}.

$$L = -l_t + \max_{t^0 \in C,\ t^0 \neq t} u_{t^0} \tag{1}$$

It can be shown that when $L < 0$, the network is verified with respect to the input image and hence we can terminate the learning algorithm as soon as this condition is satisfied.

B. Constraining slopes

For ReLU transformer, zonotope approximation [1] is sound only when $\lambda \in [0, 1]$. However, gradient descent is agnostic to this constraint and can push λ out of the allowed values. Therefore, we clip the value of λ after every step of gradient descent. Post clipping, it could happen that network is not verifiable even if loss is negative. Therefore, one should terminate the learning loop only after propagating the zonotope with clipped slopes.

C. Constraining L_∞ Input Ball

Further refinement can be done in terms of constraining the input L_∞ ball. One should perturb the input image only to the extent it is valid. An image is valid if pixel i has value in the interval $[0, 1]$. More precisely, we modify interval bounds for pixel i from $[v_i - ?, v_i + ?]$ to $[max\ (0, v_i - ?), min\ (1, v_i + ?)]$. This can be implemented elegantly by performing center-shifting.

D. Subtraction trick on output zonotope

Loss function only depends on the difference between two box intervals. The subtraction between intervals can be made more precise by using expression for individual labels instead of box intervals. Let $\hat{x}_i = a_{0i} + \sum_j a_{ji} ?_j$ be the expression for label i that will be compared with expression $\hat{x}_t$. In order to show that interval bounds obtained from $\hat{x}_i$ is less than $\hat{x}_t$, it suffices to show that interval bounds obtained from expression $\tilde{x}_i = \hat{x}_i - \hat{x}_t$ will be lesser than $[0, 0]$. This formulation is more precise than naively comparing interval bounds for each neuron because some of the coefficients of $?_j$ can cancel (or reduce) each other.

E. Adaptive Learning Rate

We adapt the learning rate of gradient descent depending on the search space terrain. For example, if the loss function does not improve for p iterations, we assume that optimizer is oscillating and decrease the learning rate by a factor of f. We treat p and f as hyper-parameters and fix their values for each network independently.

[1] l_i and u_i are lower and upper bounds of i^{th} neuron

References

[1] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, Martin Vechev *Fast and Effective Robustness Certification*, NIPS, 2018.