RIAI Project Report

DeepPoly-based Neural Net Verification

Ajaykumar Unagar, Rishabh Singh, Yannick Strümpler
ETH Zurich

1 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 L1 ball perturbations. The verifier uses DeepPoly [1] abstraction to soundly approximate the behavior of the network as precisely as possible.

2 BACKGROUND

We will leverage the DeepPoly relaxation [1] for building our verifier. Deep Poly relaxation uses constraints, lower-bound and upper-bound in its formulation. For Affine layers, deeppoly is exact but for ReLU it is approximate. The ReLU transformer in DeepPoly relaxation uses two lines to bind the area of the relu activation line. In this project we try to design a better lower bound for the ReLU, such that we can verify our trained network for a given image and L0 perturbation, ϵ. The current version of ReLU transform in deeppoly uses lowest area heuristic for one neuron at a time. However, transformers with larger area exist which can verify more images than the transformer with minimum area. Concretely, the lower-bound for ReLU transformer is parametrized by the slope λ and it is sound for any value λ in [0, 1]. This value is chosen using a predefined formula as explained in the lecture. In this project, we improve the ReLU transformer in DeepPoly. Our goal is to learn a value λ for each neuron in the network which maximizes the precision of the verifier. Given a test case (an image and ϵ value) and a network, our algorithm first produces new ReLU transformer (by learning λ) for each neuron in the network and then runs the verification procedure using these transformers.

We propose gradient learning based approach to find optimal λ for each neuron in the network. These are initialized based on the formulation of the minimum area of the transformer. The objective is to make the existing implementation more precise. We create a network with custom layers (Fig. 1) that propagates the abstract DeepPoly of the input image through a given network. The input Deeppoly object is essentially an L1 ball of radius ϵ around the input x. The network has λ as the learnable parameters for the gradient descent.

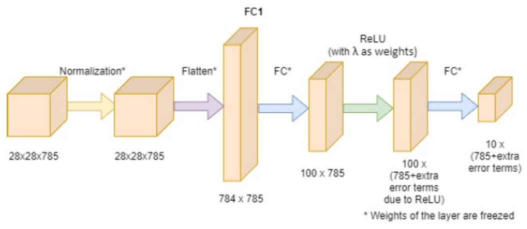

Figure 1: Fully Connected Network 1

3 OUR METHOD

3.1 Loss Function

The objective of training is to output DeepPoly such that box interval of target class, $I_t = [l_t; u_t]$ is strictly greater than $I_{t'} = [l_{t'}, u_{t'}]$ where t' is an output class such that $t' \neq t$. The most intuitive loss function is the one that measures the gap between I_t and I'_t .

$$L = -l_t + \max_{t' \in C, t' \neq t} u_{t'}$$

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.

3.2 Constraining slopes

For ReLU transformer, DeepPoly approximation 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 DeepPoly with clipped slopes.

3.3 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 $[v_i - \epsilon, v_i + \epsilon]$to $[max(0, v_i - \epsilon), min(1, v_i + \epsilon)]$. This can be implemented elegantly by performing center-shifting in pytorch.

3.4 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.

3.5 Adaptive Backsubstitution Depth

If the verifier does not verify with the default heuristic, we start learning the parameters. Because some network are larger and computing the gradient with respect to all parameters is expensive, we limit the backsubstitution depth and increase it after a fixed amount of parameter updates if the verifier did to verify.

Project work for Reliable and Interpretable AI, ETH, Zurich
2020.

REFERENCES

[1] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.