Reliable & Interpretable Artificial Intelligence Project Report

Kaan Sentürk, Gabriel Hayat, Tristan Cinquin

***Abstract*— We propose a verifier to prove the robustness of fully connected and convolutional neural networks using the DeepZ relaxation. The specificity of our method is to use gradient descent to learn the slopes of the ReLU transformers.**

I. Procedure

Our procedure goes as follows:

1) Freeze all the weights of the network.
2) Build the zonotope and initialize the slopes such as that the zonotope's area is minimized in the ReLU transformers as discussed in the lecture.
3) Forward pass though Normalization, Affine, Convolution and ReLU transformers.
4) Compute loss function and its gradient with respect to the slopes.
5) Take optimizer step and update slope values.
6) Clip the values of the slopes between 0 and 1 to guaranty soundness.
7) While verify doesn't return true, goto 3.

II. Loss function

To be able to prove that our neural network is robust to an epsilon perturbation, we must show that the lower bound of the concretization of the zonotope at the output layer evaluated at the target class is superior to all the other upper bounds (apart from the target class's upper bound). To enforce this criterion, we used the following Cross Entropy loss:

$$Loss(x, target) = -x[target] + log\left(\sum_{j \neq target} exp(x[j])\right)$$

with:

$$x[i] = \begin{cases} l[i] & if \quad i = target \\ u[i] & else \end{cases}$$

where:
u : the upper bound of the concretization of the zonotope
l : the lower bound of the concretization of the zonotope
target : the index of the target label

Furthermore, we use PyTorch's Adam optimizer to minimize the loss.

III. Optimizations

Zonotope construction: We model our zonotope as n $\times$ m matrix where the first column contains the input image and the next columns the error terms. Here, n is the number of pixels of the inputs (initially pixels) and m the number of error terms + 1. With this construction, we can simply forward our zonotope through the network like we would propagate any matrix.

Convolutional Network: To be able to verify images with convolutional networks, which were very slow, we furthermore optimized our procedure. To reduce the computational cost of back-propagating through all of the convolutional layers, we freeze all but the last layer, thus optimizing for around a hundred slopes instead of a few thousands. This made a significant improvement in PyTorch's computation of the loss's gradient and allowed significantly more optimization steps.

With that improvement, the bottleneck of our procedure became pushing the zonotope through the convolutional network at every run. To solve this issue, we save the zonotope before the last ReLU layer (the only one whose slopes are optimized) and use it at every run. This prevents a lot of re-computation and thus significantly improves the running time of a forward pass in the network.