Robustness analysis in Interval and Polytopic Domain

Simon Mathis, Simon Schaefer

I. PROBLEM

Proving the robustness of a neural feed forward network against adversarial attacks is a trade-off between the precision of an abstract domain and its speed. In the project we consider adversarial attacks inside a L_∞-norm based ϵ-ball around MNIST images, with $\epsilon = [0.005, 0.1]$. The robustness analysis program is constrained to prove robustness in maximal 7 minutes, single-threaded on a 3.3 GhZ Skylake machine using the Gurobi linear solver ([1]). Thereby false positives should be avoided by any means, i.e. it is preferred to not verify a network's robustness over verifying it without being certain.

II. APPROACH

In general using the interval domain is computationally efficient but also quite inaccurate w.r.t. the resulting intervals for each output class, as it assumes each layer to be independent from the previous layers and approximates ReLU units as intervalls between minimal and maximal activation. The polytopic domain instead solves a minimization problem w.r.t. linear optimization problem (in fact a maximization and a minimization) taking all previous layers into account, while approximating the ReLU by a triangle constraint. While this is more accurate, it is also computationally expensive. However, for the first layer both methodologies lead to the same result, such that our analysis starts with a interval analysis. Also the intervals obtained by applying a interval propagation ("box analysis") are a superset of the linear solver's solution such that after every linear solver iteration in each layer the remaining layers are solved using interval analysis. When the resulting output intervals verify the networks robustness (i.e. lower bound of correct class larger than any upper bound of the wrong classes) the analysis can be terminated.

The bottleneck of the polytopic optimization are the constraints introduced to the problem due to the layer's ReLUs. In order to save computational effort (and thereby analysis time) either their quantity has to be reduced or the bounds have to be tightened to shrink the solution space of the optimization problem.

Optimizing the whole network with the linear solver at once directly after a box pass is very computationally expensive (and also inaccurate for large networks) since all ReLU constraints are very large. Therefore, the ReLU constraints have to be tightened first by iteratively solving the linear optimization problem subsequently for each layer, until the bounds have been converged (e.g. changes smaller than some relative/absolute tolerance, for example 0.01).

While this technique is sufficient for small networks, additionally, for larger networks (above 6-100) the number of ReLUs constraining the optimization problem can reduced by only taking into account the ReLUs which give a significant contribution to the activation of neurons in the subsequent layer. The effect of a neuron on the next layer was calculated by taking the maximal value of its contribution (weight times max. input activation) to any of the neurons in the subsequent layer. For the networks we considered, it turns out that the weights in each layer are roughly Gaussian distributed. If we optimize only 'high impact ReLUs', which change the activations in the next layer by more than 20% of the ReLU with the biggest impact on activation, we get away with optimizing only roughly $1/10$ of ReLUs in the case of the 4-1024 network. This results in a drastic increase in speed (two-three orders of magnitude in our case) for the large networks while still providing tight bounds, because it reduces the size of the optimization problem by focusing on optimizing solely the most relevant ReLUs.

To save computational effort in the output layer the difference between the activations of the correct label and every other label is minimized instead of finding lower and upper bounds for all labels.

III. OTHER APPROACHES

Instead of using the interval or linear solver analysis a statistical analysis could be used, by creating perturbed images within the ϵ-range around the original image. As we are targeting to generate an image which is classified as any class other the original one, untargeted projected gradient descent (PGD) attacks are used, repeatedly each starting with a different starting value within the epsilon ball. In case an adversarial example can be found in the analysis time, the network cannot be verified. However, it turned out that the input space (784 dimensional image) is too large to find an adversarial example within the analysis time, especially for large networks.

IV. FURTHER DEVELOPMENT

So far no runtime watchdog is implemented that constrains the solver in terms of optimization time, e.g. while iterative layer optimization the ReLU bounds are tightened from step to step as is the space of possible solutions, such that the each optimization should take less (or equal) time than the previous one. Also an empirical based heuristic about the runtime of each optimization process given its inputs (bounds, weights, neuron activations, deviation from interval domain based bounds, epsilon) can be developed, which was tried but discarded due to lack of time and difficulty to estimate runtime dealing with different machines.

As purposed in [3] the problem can be converted to a dual problem such that the bounds are pre-computed more exactly than simple interval propagation. As a consequence the space of possible solutions of the linear optimization problems decreases from the beginning such that it will be more computationally efficient.

REFERENCES

[1] Gurobi optimization (http://www.gurobi.com/).
[2] ELINA: ETH library for numerical analysis (http://elina.ethz.ch/).
[3] Eric Wong, J. Zico Kolter, Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope, CoRR, 2017.