ROBUSTNESS VERIFIER HEURITICS FOR NEURAL NETWORKS

R. Deliallisi

ETH Zürich, D-INFK

C. Trassoudaine[*]

IMT Atlantique,
EURECOM, Data Science dpt.

1. PROBLEM DEFINITION AND ANALYSIS TECHNIQUES

The purpose of this work is to prove formally the local robustness (from now refered as robustness) with repect to some inputs of neural-networks (N.N.) by heuristically combining box analysis (B.A.) and linear-programming (L.-P.) solving. In this paper, we aim to present a time-efficient way of verifying N.N. robustness for large ϵ-perturbations, using the L_∞ norm.

Box analysis A very simple and fast approach to solve the N.N. robustness problem is to use a polyhedra abstract domain and to propagate it across the layers.

Linear programming A more precise but more computationaly intensive technique to verify the robustness of a N.N. is to use a linear solver to propagate the box while approximating better the non-linearities. This can be used of two ways: compute the boxes for a given layer using modeling constraints and the previous layers box values, or directly check for robustness on the last layers based on the previous computed boxes.

We here generaly prefer using L.-P. on the first layers of the N.N. not to propagate growing boxes. An analogy can be drawn with low-noise amplifiers in the physics field, for which it is really important to keep the noise minimal at the beginning of the process.

2. HEURISTICS

In this section we will refer to Box Analysis as ELINA and Linear Programmming range propagation as GUROBI within the algorithm sections as these are the used tools in our implementation.

Greedy approach The first approach and most efficient one for small N.N.s is to iterate over the layers and using N.N. at each step and verifiyng with B.A. as an exit condition. This takes advantage of the cheap computational cost of B.A. and of the incremental precision of L.-P., which has a cost. The biggest verifiable perturbations will take more time than the smaller ones due to the greater number of L.-P. calls, but this also returns results very fast for small perturbations.

Algorithm 1 Greedy precise algorithm with an early stopping condition

```
$n = size(NN)$
for $i \in [0, n]$ do
    GUROBI_OPTIMIZE($Layer_{i+1}$)
    if $Layer_{i+1}$ is the last layer then
        return VERIFY($Layer_n$)
    else
        ELINA_RANGE($Layer_{i+1,...,n}$)
        if VERIFY($Layer_n$) then
            return true
        end if
    end if
end for
```

Large networks strategy The strategy developped for large neural networks is inspired for gradient descent and aims to focus the most on the neurons that propagates the greatest part of the interval length to the last layer neurons that prevent us from verifying the robustness property.

Algorithm 2 Large networks heuristic

```
$n = size(NN)$
/* Compute boxes using ELINA. */
$(Range_1, ..., Range_n)$ = ELINA_RANGE($Layer_{1,...,n}$)
if VERIFY($Layer_n$) then
    return true
else
    /* Backpropagate last layers ranges for which the intersection with true label is not null. */
    $b_1$ = Backpropagate($Range_n$)
    /* Optimize most weighted neurons on the first layer based on backpropagation values. */
    $opt_{neurons}$ = index($b_1$, threshold)
    $Range_{1,opt_{neurons}}$ = GUROBI_OPTIMIZE($Layer_{1,opt_{neurons}}$)
    /* Compute remaining boxes using ELINA. */
    $(Range_2, ..., Range_n)$ = ELINA_RANGE($Layer_{2,...,n}$)
    return VERIFY($Layer_n$)
end if
```

Here is a more graphical version:

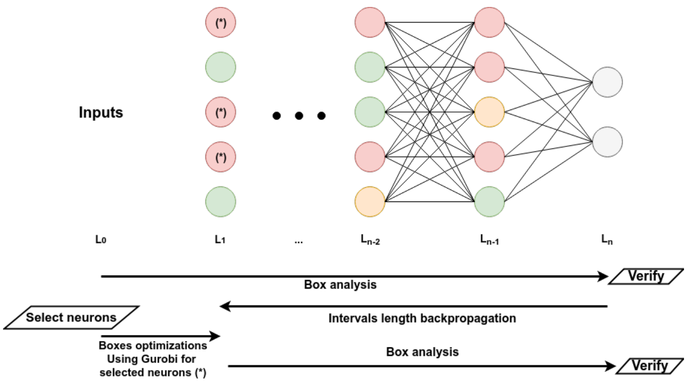
Inputs
(*)
(*)
(*)
L0
L1
...
Ln-2
Ln-1
Ln
Box analysis
Verify
Select neurons
Intervals length backpropagation
Boxes optimizations Using Gurobi for selected neurons (*)
Box analysis
Verify

Fig. 1. Large networks strategy

3. PERFORMANCES

The most important factors in the L.-P. analysis complexity seems to be the number of neurons per layer as well as the magnitude of the perturbation. Experimentally, we determined that the complexity for a given network to certify is in $o(exp(\epsilon))$.

Hence, reducing the range for the first layers by using L.-P. also help to reduce the computational time.

The model performs the analysis in a few minutes for all of the networks and ϵ-perturbations ranging from 10^{-2} to 10^{-4} using the greedy algorithm for deep-N.N.s and the Large-net strategy for the narrow-N.N.s.

Layers	neurons per layers
3	10, 20, 50
4	1024
6	20, 30, 100, 200
9	100, 200

Fig. 2. Tested configurations

[*]. Work performed while at ETH Zürich