Verification of Feedforward Neural Networks

Manuel Breitenstein, Mayank Mittal

***Abstract*—We propose an efficient method to verify robustness of deep ReLU-based classifiers against norm-bounded adversarial perturbations. Our verifier combines both interval domain and linear programming to find the right tradeoff between precision and speed. We assign a score to each neuron in a hidden layer depending on the tightness of its inputs and the importance of its output on the hidden layer after it. Depending on this score, we decide whether we need to obtain a precise bound using convex relaxation or we can use a more imprecise bound through interval domain analysis. We check our method on the provided dataset for different L_∞ norm-based perturbations.**

I. Introduction

Adversarial examples refer to data points that are not "visually" distinguishable from the examples a neural network has been trained on but are classified incorrectly. A network is said to be locally robust to this if it returns the correct output on all inputs similar to the inputs in the training set.

We consider a k layer feedforward ReLU-based neural network NN, $f_\theta : \mathcal{R}^{|x|} \rightarrow \mathcal{R}^{|y|}$, denoted as follows:

$$\begin{aligned} \hat{z}_{i+1} &= W_i z_i + b_i,\ i = \{1, 2, ..., k-1\} \\ z_i &= ReLU(\hat{z}_i),\ i = \{2, 3, ..., k-1\} \\ z_1 &= x \\ f_\theta(x) &= \hat{z}_k. \end{aligned}$$

For the MNIST dataset, $|x| = 784$ and $|y| = 10$. For some of the networks, the output is taken from a ReLU activation, i.e. $f_\theta(x) = ReLU(\hat{z}_k)$. However, this does not affect the robustness property we want to verify:

$$\phi \models \bigwedge_{i \neq label} \text{UB}(NN[i]) < \text{LB}(NN[label])$$

where $\text{LB}(.)$ denotes the lower bound of the interval and $\text{UB}(.)$ denotes the upper bound.

II. Verification Algorithm

We first compute the bounds using interval analysis for $\hat{z}_i,\ i = \{2, ..., k\}$. It is trivial to say that for small networks and small perturbations this may be sufficient to prove property ϕ. However, for larger networks, we follow the algorithm as summarized in Algorithm 1. The algorithm inputs a neural network, the perturbed image bounds and a score threshold which decides when to apply linear programming.

We use `BB[i]` to denote the bound obtained using interval domain analysis for neurons in the layer $\hat{z}_i$. The $callLinear(model, i, j)$ function finds the bounds for the j^{th} neuron in the layer $\hat{z}_i$ using linear programming. Moreover, $B(.)$ denotes bounds for a variable.

Advisor: Prof. Dr. Martin Vechev, Instructor for Reliable and Interpretable Artificial Intelligence, ETH Zürich, FS 2018/19.

Algorithm 1 Prove robustness of neural network NN

Input: NN, $B(z_1)$, $threshold$
Output: boolean value which is true if verified

procedure PerformLinearLayerwise
  $BB[:], flag \leftarrow$ box analysis for $\hat{z}_i,\ i = \{2, .., k\}$;
  if $flag = true$ **then return** true;
  $model \leftarrow$ linear solver with simplex method;
  $model \leftarrow$ add constraints for $\hat{z}$ and z using $BB[:]$;
  for $i = 2$, .., k **do**
    if $i = 2$ **then** $B(\hat{z}_2) \leftarrow BB[2]$
    else if $i = k$ **then**
      $B(\hat{z}_k) \leftarrow callLinear(model, k, :)$;
    else
      $B(\hat{z}_i) \leftarrow$ box analysis for $\hat{z}_i$ using $B(\hat{z}_{i-1})$;
      $inactive \leftarrow$ indices where $B(\hat{z}_i) \leq 0$;
      $scores \leftarrow$ score all neurons not in $inactive$;
      $use\_LP \leftarrow scores < threshold$;
      $m \leftarrow$ number of neurons in layer i;
      for $j = 1$, .., m **do**
        if $use\_LP[j] =$ true **then**
          $B(\hat{z}_i[j]) \leftarrow callLinear(model, i, j)$;
      $model \leftarrow$ update constraint $z_i = ReLU(\hat{z}_i)$;
  $\phi \leftarrow$ verify property using $B(\hat{z}_k)$;
  return ϕ;

III. Scoring Heuristics For Neurons

For inactive neurons, i.e. $B(\hat{z}_i) \leq 0$, the ReLU output is always 0 so we do not need to improve their bounds using linear programming. Hence, only the *active* neurons are considered. Our proposed score represents the influence of each neuron on the future layers. The influence score of a neuron $score_j$ is given by a product of tightness score t_j and weight's score n_j. Mathematically, for a neuron j in layer i, the scores are calculated as follows:

$$\begin{aligned} t_j &= \text{UB}(B(\hat{z}_i[j])) - \text{LB}(B(\hat{z}_i[j])) \\ n_j &= ||W_{i+1}[:, j]||_1 \\ score_j &= t_j \cdot n_j, \end{aligned}$$

where $B(\hat{z}_i[j])$ are the bounds computed using interval domain analysis and and $W_i[:, j]$ denotes the column j of the weight matrix in the affine layer $i + 1$. We observe that filtering out *inactive* neurons is sufficient for smaller networks, whereas for the large ones we rely on the scoring threshold.

References

[1] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2010. http://www.gurobi.com
[2] G. Singh, M. Püschel, M. Vechev. Making Numerical Program Analysis Fast. *ACM PLDI*, 2015. http://elina.ethz.ch/