Resilience of Artificial Neural Networks

Bineet Ghosh
bineet@cs.unc.edu

November 22, 2019

Abstract

Deep Neural Networks are one of the most used techniques for classification. When such DNN based classifiers are deployed in safety critical systems, it is extremely important to provide some safety guarantees of the DNN. In real life scenario, the inputs received by the classifier can have a lot of noise. And mis-classification due to noises can be fatal in a safety critical scenario. In this project, I try to find the maximum amount of error that a Deep Neural Net classifier can tolerate, given a class of inputs using some of the methods described in [1].

In [1], the problem of computing maximum perturbation bounds for ANNs is then reduced to solving mixed integer optimization problems (MIP). I provide an implementation to compute the amount of perturbation a Deep Neural Network classifier can tolerate. Finally the method is evaluated on a real dataset from UCI Machine Learning Repository[1]

1 Problem Statement

With the increase in deployment of Neural Net based classiifer in safety critical scenarios like medical image processing, (semi-)autonomous vehicles etc. it is extremely important to provide a safety guarantee of these classifiers. Once such important safety metric could be the amount of noise due to environment or sensors can those classifiers tolerate and still produce correct result. Therefore, given a trained Deep Neural Network, an input and a class label I find out the amount of perturbation the Deep Neural Network can tolerate without mis-classification. More precisely, I am defining and computing safe perturbation bounds for multi-class ANN classifiers.

I restrict my Deep Neural Network to only have ReLU activation function. Readers are requested to refer [1] for other activation functions. I impose no restriction on other aspects of the architecture. The problem of finding maximum perturbation that the Deep Neural Network can tolerate without mis-classification is reduced to solving a corresponding mixed integer programming (MIP). In this formulation the non-linear function ReLU is linearized using a method called big-M (For details please refer [1]). I use Gurobi[2] to solve the optimization problem.

In the next section, we formulate the model of the Deep Neural Network (DNN) that we use to perform our analysis *i.e.* compute the perturbation tolerance bounds given such a neural network. And describe the methodology used (detailed description in [1]) to encode the maximum perturbation tolerance to a mixed integer linear programming.

2 Methodology

First, we formulate the model of the Deep Neural Network (DNN) that we use to perform our analysis *i.e.* compute the perturbation tolerance bounds given such a neural network.

Let, a Deep Neural Network with K layers with M neurons at each layer, be $\mathcal{N}$. Let, the network function be $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, defined as $f = f_K \circ f_{K-1} \circ f_{K-2} \circ \ldots \circ f_1$, where each f_i is the network function for layer i in the DNN, such that, $f_1 : \mathbb{R}^n \rightarrow \mathbb{R}^M$, $\forall_{2 \leq i \leq K-1}\ f_i : \mathbb{R}^M \rightarrow \mathbb{R}^M$, and $f_K : \mathbb{R}^M \rightarrow \mathbb{R}^m$. Each layer $i \in [2, K]$, takes input from its preceding layer $i-1$.
Let, W_i be weights and b_i be the biases of layer i in the DNN $\mathcal{N}$ after training.

[1]https://archive.ics.uci.edu/ml/datasets.php
[2]www.gurobi.com

In this project I only consider ReLU as the activation function of the DNN $\mathcal{N}$ (For other activation functions, please refer to [1]). Let, $N_{i,l}$ be the i-th neuron at the l-th layer, s.t., $1 \leq i \leq M$ and $1 \leq l \leq K-1$. then, formally, for a neuron $N_{i,l}$:

$$\sigma\left(\sum_{j=1}^{n} w_{i,j}^{l-1} x_l + b_i^{l-1}\right) = max\left\{0, \sum_{j=1}^{n} w_{i,l}^{l-1} x_l + b_i^{l-1}\right\}$$

where, $w_{i,j}^{l-1}$ are the elements of matrix W_{l-1}, b_i^{l-1} are the elements of the matrix B_{l-1} and x_l are the elements of the input vector.

Let, $\bar{x} \in \mathbb{R}^n$ be the input to the DNN $\mathcal{N}$ and $\bar{y} \in \mathbb{R}^m$ be the output of the DNN; *ie* $\mathcal{N}$, *i.e.* $f(\bar{x}) = \bar{y}$ or $f\big((x_1, x_2, \cdots, x_n)\big) = (c_1, c_2, \cdots, c_m)$. Where, $\bar{x} = (x_1, x_2, \cdots, x_n)$, $\bar{y} = (c_1, c_2, \cdots, c_m)$ and c_i is the confidence value of the classification label i for input $\bar{x}$. Additionally, the DNN $\mathcal{N}$ should satisfy the following property:

$$\forall_{\bar{x} \in \mathbb{R}^n} f(\bar{x}) = (c_1, c_2, \cdots, c_m) \Rightarrow \sum_{i=1}^{m} c_i = 1$$

Let, $o : [0,1]^m \to \{1, 2, \cdots, m\}$ be the output evaluation function of $\mathcal{N}$. Therefore,

$$o\big(f(\bar{x})\big) = o\big((c_1, c_2, \cdots, c_m)\big) = \operatorname*{argmax}_j \{c_1, c_2, \cdots, c_m\}$$

Informally, o is the final classification label.

2.1 Perturbation Tolerance

In this sub-section we describe an overview of how the optimization problem is formulated to compute the maximum perturbation that a particular input can tolerate given a trained Deep Neural Net and a class label.

Let, $X = \{\bar{x_1}, \bar{x_2}, \cdots, \bar{x_p}\}$ be a set of inputs, where $\bar{x_i} = \{x_1^i, x_2^i, \cdots, x_n^i\}$. Therefore, f is overloaded as follows:

$$f(X) = [i_1, i_2, \cdots, i_n]$$

where $i_j = [l_j, u_j] \wedge \forall_{\bar{x} \in X} f(\bar{x}) \subseteq f(X)$

Let, $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m$ be the set of objects classified based on their true values. Informally, sets in objects $\mathcal{C}_i$ belong to the class label i. These can be considered as ground truth. Let $\forall_{1 \leq i \leq m}\ \mathcal{C}_i^{\mathcal{N}} \subseteq \mathcal{C}_i$

$$\forall_{\bar{x} \in \mathcal{C}_i^{\mathcal{N}}}\ o(f(\bar{x})) = i$$

$$\forall_{\bar{x} \in \mathcal{C}_i \setminus \mathcal{C}_i^{\mathcal{N}}}\ o(f(\bar{x})) \neq i$$

Let,

$$f(\mathcal{C}_i^{\mathcal{N}}) = [r_i^{min}, r_i^{max}]$$

Informally, we want find maximum δ *s.t.*

$$o(f(\mathcal{C}_i^{\mathcal{N}} + \delta)) = o(f(\mathcal{C}_i^{\mathcal{N}}))$$

We formulate the Perturbation Tolerance problem as the following optimization problem:

$$maximize\ \delta$$

s.t

$$\forall_{\bar{x} \in \mathcal{C}_i^{\mathcal{N}}}\ o(f(\bar{x} + \delta)) = o(f(\bar{x}))$$

The details of how the mixed integer linear programming as discussed above is formulated in the next subsection.

2.2 Optimization Formulation

In this sub-section we will discuss the big-M method used to encode a ReLU function $y_i = max(0, y_{i-1})$. Then using the encoding of ReLU we will encode the final optimization problem as perturbation maximization problem. Please refer to [1] for details.

Encoding of ReLU activation function The non-linear function $x_i^{(l)} = max(0, im_i^{(l)})$ is encoded using a method called big-M, which introduces a binary integer variable $b_i^{(l)}$, a positive constant $M_i^{(l)}$ such that $-M_i^{(l)} \leq im_i^{(l)}$ and $x_i^{(l)} \leq M_i^{(l)}$ for all possible values of $im_i^{(l)}$ and $x_i^{(l)}$.

Proposition 1. *(From [1])* $x_i^{(l)} = \mathsf{max}(0, \mathsf{im}_i^{(l)})$ *iff the constraints* (1a) *to* (3b) *hold.*

$$x_i^{(l)} \geq 0 \tag{1a}$$
$$x_i^{(l)} \geq \mathsf{im}_i^{(l)} \tag{1b}$$

$$\mathsf{im}_i^{(l)} - b_i^{(l)} M_i^{(l)} \leq 0 \tag{2a}$$
$$\mathsf{im}_i^{(l)} + (1 - b_i^{(l)}) M_i^{(l)} \geq 0 \tag{2b}$$

$$x_i^{(l)} \leq \mathsf{im}_i^{(l)} + (1 - b_i^{(l)}) M_i^{(l)} \tag{3a}$$
$$x_i^{(l)} \leq b_i^{(l)} M_i^{(l)} \tag{3b}$$

Next, using the above encoding, I encode the maximum perturbation problem using Theorem 1 in [1]

Theorem 1. *(From [1]) For a given $\alpha \geq 1$ and $k \in \{1, \ldots, d^{(L)} - 1\}$, the optimum of the MIP in* (4) *equals Φ_m for ANNs with ReLU nodes and* $\mathsf{softmax}$ *output layer. For ANNs using* tan^{-1} *it yields an under-approximation.*

$$\begin{aligned}
&\text{minimize} \qquad \Phi_m := \textstyle\sum_{i \in \{1,\ldots,d\}} \epsilon_i^{\mathsf{abs}} \\
&\text{subject to} \\
&x_m^{(L-1)}(a) \geq \ln(\alpha) + x_i^{(L-1)}(a) && \forall i \in \{1, \ldots, d^L\} \setminus m \\
&\sum_{i \in \{1,\ldots,d^L\} \setminus m} c_i \geq k \\
&x_i^{(L-1)}(a+\epsilon) \geq x_m^{(L-1)}(a+\epsilon) - M(1 - c_i) && \forall i \in \{1, \ldots, d^L\} \setminus m \\
&\epsilon_i^{\mathsf{abs}} \geq \epsilon_i && \forall i \in \{1, \ldots, d\} \\
&\epsilon_i^{\mathsf{abs}} \geq -\epsilon_i && \forall i \in \{1, \ldots, d\} \\
&c_i \in \{0, 1\} && \forall i \in \{1, \ldots, d^L\} \setminus m
\end{aligned} \tag{4}$$

Note that all x_i will have constraints (1a), (1b), $\cdots$, (3b) as they are output of ReLU functions in final optimization problem

3 Experiments

I have implemented the algorithm in Python based tool that uses `numpy` for basic operations, `keras`[3] for modelling and training Deep Neural Nets, and Gurobi to encode the maximum resilience problem as an

[3]`https://keras.io/`

optimization problem. The implementation can be found in a public Github Repository[4]. All the experiments were performed on a Lenovo ThinkPad Mobile Workstation with i7-8750H CPU with 2.20 GHz and 32GiB memory on Ubuntu 18.04 operating system (64 bit). An EColi classification dataset[5] from UCI Machine Learning Repository [2] has been used for all the experiments. The size of the input vector in this datset is 7 and there are 8 different class labels. Let the class labels be 1, 2, $\cdots$, 8. For all the experiments the architecture of the Deep Neural Network was $7 \times 8 \times 8 \times 8 \times 8 \times 8$ with epoch=2000 and batch=20. The tool also provides an interface which allows user to perturb the data to a desired amount and see the new predicted label.

Experiment 1

- A new Deep Neural Network with the EColi dataset was trained
- Data: $\{0.73,\ 0.36,\ 0.48,\ 0.5,\ 0.53,\ 0.91,\ 0.92\}$
- True Label=Predicted Label: 2
- Max perturbation: $\{0.6113,\ 0,\ 0,\ 0,\ 0,\ -0.4359,\ 0\}$

Experiment 2

- A new Deep Neural Network with the EColi dataset was trained
- Data: $\{0.49,\ 0.29,\ 0.48,\ 0.5,\ 0.56,\ 0.24,\ 0.35\}$
- True Label=Predicted Label: 1
- Max perturbation: $\{-0.1950,\ 0,\ 0.4390,\ 0,\ 0,\ 0,\ 0\}$

Experiment 3

- A new Deep Neural Network with the EColi dataset was trained
- Data: $\{0.74,\ 0.49,\ 0.48,\ 0.5,\ 0.42,\ 0.54,\ 0.36\}$
- True Label=Predicted Label: 3
- Max perturbation: $\{0,\ 0,\ 0.2525,\ 0,\ 0,\ -0.2475,\ 0\}$

Experiment 4

- A new Deep Neural Network with the EColi dataset was trained
- Data: $\{0.76,\ 0.71,\ 0.48,\ 0.5,\ 0.5,\ 0.71,\ 0.75\}$
- True Label=Predicted Label: 4
- Max perturbation: $\{0,\ 0.0257,\ 0,\ 0,\ -0.2205,\ -0.1576,\ 0.0961\}$

4 Related Work

In this section, we discuss the models of Deep Neural Networks (DNN) that has been considered in some of the related works.

In [3], given a set of inputs represented as a convex polyhedron, guaranteed output range of the DNN is computed. The formalism is as follows.
$l_1, l_2, \cdots, l_k$ be the classification labels. $\phi(\mathbf{x})$ is the polyhedron around an input $\mathbf{x}$ and models perturbations

[4] `https://github.com/bineet-coderep/DNNResilience`
[5] https://archive.ics.uci.edu/ml/datasets/ecoli/

of $\mathbf{x}$. $range(l_i, \phi)$ denotes the range of output corresponding to the label l_i for an input polyhedron ϕ. Then the network generates a wrong label l_j for $\phi(\mathbf{x_i})$, where input $\mathbf{x_i}$ belongs to label l_i, iff:

$$[range(l_0) \times range(l_1) \times \cdots \times range(l_k)] \cap \{l_k \leq l_j\} \neq \emptyset$$

Let, us consider a neural network with k layers, having N neurons at each layer. Let $\mathbf{x} \in \mathbb{R}^n$ be an input, and $y \in \mathbb{R}$ be an output. Let W_i be the weight matrix and b_i be the bias matrix at layer l_i of the DNN. Therefore, W_0 is a matrix of size $N \times n$ and b_0 is of size $N \times 1$. The activation function in the neural network is $\sigma(z)$ defined as $\sigma(\mathbf{z}) = [\sigma(z_1) \quad \sigma(z_2) \ldots \sigma(z_n)]^T$ where, $\sigma(z_i) = max(z_i, 0)$. The Neural Network is represented as a composition of functions, *i.e.* each layer is considered as a transformation function. Let, $F : \mathbb{R}^n \to \mathbb{R}$ be the function representing the neural network. Then, $F = F_k \circ F_{k-1} \circ \cdots \circ F_1$ where F_i is the function representing layer i and $F_i(z) = \sigma(W_i z + b_i)$

Finally, the authors present the range estimation problem as follows:
I/p: DNN $\mathcal{N}$, $P : Ax \leq b$ (Input constraint), δ (tolerance)
O/p: $[l, u]$, $\forall x \in P \ : F(x) \in [l, u]$, $\underset{x \in P}{max} F(x) \geq u - \delta$, $\underset{x \in P}{min} F(x) \geq l + \delta$

In [4], given an input set to the Neural Network, the upper and lower bound are computed, assuming the functions are Lipschitz continuous. The formalism of the Neural Network is follows.

Let, (X, d_x) and (Y, d_y) be two metric spaces; where d_x is the metric defined over the set X and d_y is the metric defined over the set Y. Let, $f : X \to Y$ be a Lipschitz-continuous function *i.e.* $d_y(f(x_1), f(x_2)) \leq K \cdot d_x(x_1, x_2)$. Let the Deep Neural Network be represented as a Lipschitz-continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$. Also let, $\mathbf{x} \in \mathbb{R}^n$ be an input to the network. So, $f(\mathbf{x}) = \{c_1, c_2, \cdots, c_m\} \in \mathbb{R}^m$ is the corresponding output of the DNN, where c_i is the confidence value corresponding to classification label l_i.

$$f(x) = f_N(f_{N-1}(\cdots f_1(x, W_1, b_1), W_2, b_2), \cdots) W_N, b_N)$$

where W_i and b_i are learn-able parameters.

Also, $f_i(z_{i-1}; W_{i-1}, B_{i-1})$ be a function mapping from output of layer $i-1$ to input of layer i, *s.t.* z_{i-1} is the output if layer $i-1$. The input and output is assumed to be normalized, *i.e.* $x \in [0,1]^n$ and $f(x) \in [0,1]^m$ Let, $o : [0,1]^m \to \mathbb{R}$ be a Lipschitz-continuous function, statistically evaluating the output.

Both the functions, f and o are Lipschitz-continuous, therefore, all the values between upper and lower limits are *reachable i.e.* have corresponding inputs. Reachability of Neural Network is defined as follows:
Input: $X' \subseteq [0,1]^n$
Network: $f : \mathbb{R}^n \to \mathbb{R}^m$
Let the reachability of f over the function o under an error tolerance $\epsilon \geq 0$ be $R(o, X', \epsilon) = [l, u]$; such that, $l \geq o(f(x')) - \epsilon$ and $u \leq o(f(x')) + \epsilon$

Let, $j \in [1, 2, \cdots, m]$ be labels. $o = \Pi_j$ *s.t.* $\Pi_j\big((c_1, c_2, \cdots, c_m)\big) = c_j$. Also let, $c_j(x) = \Pi_j(f(x))$ be networks confidence in classifying x as j.
Informally, a network f is safe with respect to an input x and an input subspace $X' \subseteq [0,1]^n$ with $x \in X'$ iff $\forall x' \in X' : \text{argmax}_j\, c_j(x') = \text{argmax}_j\, c_j(x)$ Formally, $u(\oplus, X', \epsilon) \leq 0$ where, $\oplus(c_1, c_2, \cdots, c_m) = \big(\Pi_i(c_1, c_2, \cdots, c_m) - \Pi_j(c_1, c_2, \cdots, c_m)\big)\ j = \text{argmax}_j\, c_j(x)$
Let, $w = o \circ f$ (also Lipschitz). And X' is a box constraint. The minimum bound can be formulated as the following non-convex optimization problem: $\underset{x}{\min} \quad w(x) \quad s.t. \quad x \in [a, b]$

References

[1] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

[2] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[3] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In Aaron Dutle, César Muñoz, and Anthony Narkawicz, editors, *NASA Formal Methods*, pages 121–138, Cham, 2018. Springer International Publishing.

[4] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. *arXiv preprint arXiv:1805.02242*, 2018.