Comparison of Two Approaches for Robustness Verification of Deep Neural Networks

Meghan Stuart, Parasara Sridhar Duggirala

University of North Carolina, Chapel Hill

1 Introduction

Machine learning techniques, particularly deep neural networks (NN) are being used more frequently in safety-critical applications, like autonomous vehicles. However, researchers have raised questions as to the robustness and predictability of these techniques. In fact, it has been demonstrated that small perturbations to network input can lead to significant changes in output. For example, many NNs that perform classification tasks are susceptible to adversarial attacks, which often take the form of introducing small perturbations to properly classified inputs, leading to misclassifications[8, 9]. Certain network architectures can be trained to be robust to many common types of attacks [7, 4], but to employ them in safety-critical applications, it is necessary to prove their robustness to attacks.

Exhaustive search based and Abstract Interpretation based methods were two initial approaches proposed for verification of this robustness property. Currently, there is no evaluation of the (dis)advantages and performance of these two approaches on a standard benchmark since these two techniques were developed almost concurrently by different groups. In this ongoing work, we compare the performance of benchmarks on a NN for MNIST data set and present the results. We discover a cause for potential unsoundness in the implementation of the search based verification technique and encounter a corresponding verification instance.

2 Background

We define neural networks as functions $f\colon \mathbb{R}^n \to \mathbb{R}^l$ which are compositions of a series of affine functions, of the form $g(\mathbf{x}) = W \cdot \mathbf{x} + \mathbf{b}$ where W is a real-valued matrix of weights and $\mathbf{b}$ is a real-valued vector of biases, followed by nonlinear activation functions called Rectified Linear Units (ReLU), defined as follows. Let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be defined as $\sigma(x) = \max(x, 0)$. Then $\mathrm{ReLU}\colon \mathbb{R}^m \to \mathbb{R}^m$ is defined as

$$\mathrm{ReLU}(\mathbf{x}) = (\sigma(x_1), \ldots, \sigma(x_m))^{\mathsf{T}}.$$

NN are often used for classification tasks. For a classifier network, an output $\mathbf{y} \in \mathbb{R}^l$ is a vector of l scores. A class assigned to the image is found by taking the argmax of these scores.

3 Input Space Search Approach

The range of each output dimension of a NN on a given input range can be estimated by finding the minimum and maximum of the possible outputs. Finding optima of NN output on hyperrectangular input spaces can be encoded exactly as a Mixed integer linear program (MILP), an optimization problem on real and integer variables with linear constraints, but since solving MILPs is known to be NP-hard, employing MILP solvers to directly compute the range of possible outputs is not feasible for even small NNs. Dutta et al. present a method in which they use gradient descent to find local optima of each output function to constrain the MILP search space, then use a MILP heuristic solver to prove that a more optimal solution does not exist or provide an input that produces a more optimal solution. This process is repeated, using the new near-optimal solution as a starting point for local search, until the optima are found. Their method, as claimed in the paper, produces a sound overapproximation of the range of each output that is δ-tight.

Their encoding is as follows. We assume that a range of inputs X given is a hyperrectangle in $\mathbb{R}^n$. Let $\mathbf{x} \in X$ be the network input, $\mathbf{y_1}, ..., \mathbf{y_h}$ be the output of each hidden layer with each $\mathbf{y_i} \in \mathbb{R}^{m_i}$, and let $z \in \mathbb{R}$ be the output of the network. Introduce additional $\mathbf{t_1}, ..., \mathbf{t_h}$ with each $\mathbf{t_i} \in \left(\mathbb{Z}/2\mathbb{Z}\right)^{m_i}$. These are used to

encode the following constraints. Let C_0 ensure that $x \in X$. This is a linear constraint of the form $A\mathbf{x} \leq \mathbf{b}$. Hidden layer $i+1$ computes $\mathbf{y_{i+1}} = \text{ReLU}(W_i \cdot \mathbf{y_i} + \mathbf{b_i})$. For $0 \leq i < h$, hidden layer $i+1$ computation is encoded by

$$C_{i+1} = \begin{cases} \mathbf{y_{i+1}} \geq W_i \cdot \mathbf{z_i} + \mathbf{b_i}, \\ \mathbf{y_{i+1}} \leq W_i \cdot \mathbf{z_i} + \mathbf{b_i} + m\mathbf{t_{i+1}}, \\ \mathbf{y_{i+1}} \geq 0, \\ \mathbf{y_{i+1}} \leq m(\mathbf{1} - \mathbf{t_{i+1}}) \end{cases}$$

Where $m \in \mathbb{R}$ is larger than any possible output of any node. The output is constrained by C_{h+1} of the form $z = W_h \cdot \mathbf{z_h} + \mathbf{b_h}$. Then the range of the output over input range X is described by the maximum/minimum MILP

$$\begin{aligned} &\text{max/min} && z \\ &\text{subject to} && C_0, ..., C_{h+1} \\ & && \mathbf{x} \in \mathbb{R}^n,\ z \in \mathbb{R},\ \mathbf{y_i} \in \mathbb{R}^{m_i},\ \mathbf{t_i} \in \left(\mathbb{Z}/2\mathbb{Z}\right)^{m_i} \qquad \forall 1 \leq i \leq h \end{aligned}$$

While the theory presented in [5] is sound, we believe that implementation might not preserve soundness for all verification instances; here is why. In the paper, they use a function **SolveMILPUptoThreshold** which takes an MILP encoding and a local maximum u produced by the local search described above, and returns `feasible` if there exists an input x in the query range that such that $f(x) \geq u$ or `not feasible` if such and x does not exist. This function is implemented using a widely used commercial heuristic MILP solver Gurobi [1]. Since it uses heuristic solution methods, sometimes it cannot prove that u is the maximum, but also cannot find a input producing a larger output. In this case, u is not guaranteed to be a sound upper bound on the output range, however it appears that the implementation provided by Dutta et al. treats this case the same as if Gurobi verified that u is sound upper bound. This might potentially violate the soundness of the implementation.

4 Abstract interpretation approach

Abstract interpretation is a framework that allows for sound overapproximations of the behavior of programs. We assume readers have knowledge of the basics of abstract interpretation: lattice structures, Galois connections, concrete and abstract transformers etc. For a review of these topics, see [3].

Let $f\colon S \to L$ be a function between two sets, we denote the concrete transformer of f as $T_f\colon \mathcal{P}(S) \to \mathcal{P}(L)$. Suppose A_S is an abstraction of $\mathcal{P}(S)$ and A_L is an abstraction of $\mathcal{P}(L)$, then we denote an abstract transformer of f as $T_f^{\#}\colon A_S \to A_L$. If γ_S and γ_L are the respective concretization functions, then recall $\forall a \in A_S,\ T_f(\gamma_S(a)) \subset \gamma_L(T_f^{\#}(a))$. This is useful for showing that a robustness property (X, C) holds, that is that the output of a network on some set of inputs X lies in some set C, since $\gamma(T_f^{\#}(X)) \subset C$ implies that $T_f(X) = f(X) \subset C$.

In [6], Gehr et al. present an abstract-interpretation-based approach to verify robustness properties of the form (X, C_L) of NNs, where X is a bounded range of inputs and $C_L = \{y \in \mathbb{R}^l \mid \arg\max y_i = L\}$. Additionally, in [10] (from the same group), they present a sound abstraction of $\mathcal{P}(\mathbb{R}^m)$ by the lattice of zonotopes tailored for use with NNs and define the concrete and abstract transformers of ReLU and other common activation functions, called *DeepZ*. We provide a simplified overview here.

A zonotope is a convex closed center-symmetric polytope $Z \subset \mathbb{R}^n$. It can be represented as an affine function

$$z\colon [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_m, b_m] \to \mathbb{R}^n$$

of the form $z(\epsilon) = M \cdot \epsilon + \mathbf{b}$ where $\epsilon_i \in [a_i, b_i]$. M captures the boundary of the zonotope, and $\mathbf{b}$ represents the center of the shape. Zonotopes can be given a lattice structure, where the partial order is the standard subset when the zonotopes are considered as polytopes in $\mathbb{R}^n$. The least upper bound of two zonotopes is the smallest zonotope containing both, and the greatest lower bound of two zonotopes is the largest zonotope contained in the intersection of them. The lattice of zonotopes in $\mathbb{R}^n$, denoted $\mathcal{A}^n$, is an abstraction of $\mathcal{P}(\mathbb{R}^n)$ with abstraction function α sending an element of $\mathcal{P}(\mathbb{R}^n)$ to the smallest zonotope containing it, when both are considered subsets of $\mathbb{R}^n$, and concretization function γ, the natural embedding of the lattice

of zonotopes in $\mathcal{P}(\mathbb{R}^n)$. Let $f\colon \mathbb{R}^n \to \mathbb{R}^m$ be an affine function. If Z is a zonotope then $f(Z)$ is also a zonotope. A natural abstract transformer of f is then $T_f^{\#}\colon \mathcal{A}^n \to \mathcal{A}^m$ where $\forall Z \in \mathcal{A}^n$, $T_f^{\#}(Z) = f(Z)$.

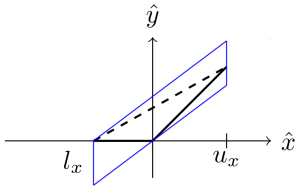

Figure 1: Parallelogram over-approximation of ReLUi. Figure taken and altered from [10].

In [10], Singh et al. define $\text{ReLU}_i\colon \mathbb{R}^n \to \mathbb{R}^n$ which applies σ to only the i^{th} dimension of the input, then note $\text{ReLU} = \text{ReLU}_n \circ \ldots \circ \text{ReLU}_1$. They define an abstract transformer of each ReLU_i as follows. We will use the notation $I_{i\leftarrow s}$ to denote the identity matrix except that $I_{i,i} = s$ and $\mathbf{0}_{i\leftarrow s}$ to denote zero vector except that $\mathbf{0}_i = s$. Let Z be a zonotope in $\mathcal{A}^n$ input to ReLU_i with affine form $\hat{z}$. Let $[l_i, u_i]$ be the range of the i^{th} dimension of Z. If $l_i < 0$ and $u_i > 0$, then we must define an overapproximation of ReLU_i between l_i and u_i. We do this by defining an affine transformation that represents the smallest-area parallelogram bounding ReLU_i in the desired range, as shown in Figure 1. Denote the slope of the linear approximation, the dashed line in the figure, $\lambda = \frac{u_i}{u_i - l_i}$, and let $\mu = -\frac{u_i l_i}{2(u_i - l_i)}$. Then the affine overapproximation of $Y = \text{ReLU}_i(Z)$ is

$$\hat{y} = \begin{cases} \hat{z}, & \text{if } l_i \geq 0 \\ I_{i\leftarrow 0} & \text{if } u_i \leq 0 \\ \mu \mathbf{0}_{i\leftarrow 1+\epsilon} + I_{i\leftarrow\lambda}\hat{z} & \text{otherwise} \end{cases}$$

where $\epsilon \in [-1, 1]$

5 Comparison of Search and Abstract Interpretation Based Methods

We compare the performance of each method on a network trained to classify handwritten digits. The network takes a 28×28 grey-scale image with intensity between 0 and 1 of a handwritten digits and assigns it one of 10 labels, 0-9. It has three hidden layers, each with 20 nodes. It is trained on the MNIST data set [2]. We use each method to test that each of four image classifications is robust to adding uniform random noise between 0 and ϵ to each pixel, where ϵ takes values in $\{0.02, 0.03, 0.04, 0.05\}$.

The runtime of Dutta et al.'s method for each input range was on the order of 8000 seconds, whereas the runtime of Gehr et al.'s method was less that a second for each input range. For each ϵ, we report if the method was able to verify robustness of each image under such perturbations below. We can see that the abstract interpretation approach outperforms the space search approach both in time and accuracy. However, unlike search based methods, abstract interpretation techniques do not provide a counterexample for the robustness property.

Space search approach		
Image 0	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	not verified
	$\epsilon = 0.05$	not verified
Image 1	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	not verified
	$\epsilon = 0.05$	not verified
Image 2	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	verified
	$\epsilon = 0.05$	verified
Image 3	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	not verified
	$\epsilon = 0.04$	not verified
	$\epsilon = 0.05$	not verified

Abstract interpretation approach		
Image 0	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	verified
	$\epsilon = 0.05$	not verified
Image 1	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	not verified
	$\epsilon = 0.05$	not verified
Image 2	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	verified
	$\epsilon = 0.05$	verified
Image 3	$\epsilon = 0.02$	verified
	$\epsilon = 0.03$	verified
	$\epsilon = 0.04$	not verified
	$\epsilon = 0.05$	not verified

We encountered the potential unsoundness of search based method in two instances. First, Image 0, $\epsilon = 0.02$ and second, Image 2, $\epsilon = 0.02$. In these two instances, Gurobi was neither able to prove that the u provided was the upper bound, nor find a feasible solution. In these two instances, the provided u was treated as upper bound by the search methods. Comparison with abstract interpretation techniques shows that the verification result of search based methods was not incorrect.

This potential unsoundness highlights an important need to systematically compare the results on various benchmarks. In future we intend to perform a systematic comparison of different verification techniques on various benchmarks.

References

[1] Gurobi Optimization - The State-of-the-Art Mathematical Programming Solver. http://www.gurobi.com/.

[2] MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. http://yann.lecun.com/exdb/mnist/.

[3] Stefan Bygde. *Abstract Interpretation and Abstract Domains*. PhD thesis, M\"alardalen University.

[4] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval Networks: Improving Robustness to Adversarial Examples. In *International Conference on Machine Learning*, pages 854–863, July 2017.

[5] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output Range Analysis for Deep Feedforward Neural Networks. In *NASA Formal Methods*, volume 10811, pages 121–138. 2018.

[6] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, May 2018.

[7] Shixiang Gu and Luca Rigazio. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *arXiv:1412.5068 [cs]*, December 2014.

[8] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. *ICLR*, page 14, 2017.

[9] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv:1605.07277 [cs]*, May 2016.

[10] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems 31*. 2018.