DNN Local Robustness Verification Using Marabou

Shimon Malnick
204553788
malnick@mail.tau.ac.il

April 20th 2022

Abstract

In the past 10 years there is a significant increase in the number of systems that turn to the use of deep neural networks to solve numerous problems. In this project I present the use of a popular deep neural networks verification system called *Marabou*[1]. The verification is shown in terms of *local robustness* against adversarial perturbations in the L_∞ ball around test examples. The method is compared on several DNN's, some of them are taken from the *FGP* [2] paper. I also provide a useful tool to use the Marabou with deep neural networks given in the popular *PyTorch* framework. The code for this project can be accessed via https://github.com/ShimonMalnick/auto_verification_project

1 Introduction

In recent years, the amount of accessible data and computing power surged, which contributed to a big increase in the use of machine learning models. Usually these models do not use a predefined user-given algorithm, but rather let the model "learn" a solution to a task according to some guidance. Verifying these type of models is a hard task since it is hard to fully understand the intrinsics of these models, as opposed to algorithmic solutions that can be examined more easily.
By adding some small changes to the inputs of some DNN's, we can change the desired output dramatically. In this work I will use a known DNN's verifying system called Marabou [1] to evaluate whether a given DNN is robust against these kind of attacks. First I'll explain the main concepts I use in this work in the next subsections.

1.1 Deep Neural Networks

Deep neural networks (DNN's) [3], are machine learning models that can learn a certain task given a a big amount of data specific for the task. In my work I

focused only on classification models that were trained in a supervised setting. This means that the model, given an input, needs to discriminate between certain classes, and output a probability decision stating its confidence for the different output classes. The model is trained by seeing a lot of samples, and for each sample the model gives a prediction, and learns by its difference from the true label, where each sample has a label specifying its true class.
In the setup for this project, I'll show results on the MNIST [4] dataset, that contains handwritten digit images of all digits (0-9). I will show results of DNN's constructed from a stack of fully connected layers activated by the ReLU [5] activation (explained more thoroughly in section 2.2). The output is a vector with C entries, where C is the number of available classes. Specifically, for the MNIST case $C = 10$ as 10 is the number of possible digits.

1.2 Marabou

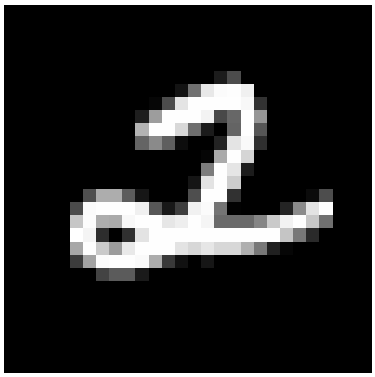
Marabou [1] is a framework for deep neural network verification. The framework based on a satisfiability modulo theory solver (SMT)[6, 7], and it is an improvement of the authors on their previous work called Reluplex [8].
This system enables checking constraints on the inputs and outputs of ReLU based fully connected DNN's. Some systems are really crucial for verification, and en example for that is the case study the authors show. They show a case study of verifying a system called ACAS Xu [9], which is an airborne collision avoidance system, and it is crucial to verify these types of systems as it is vital for the planes behaviour during flight.
In this work I will use Marabou in the context of adversarial robustness, by constraining some defined values for the input and output values.

1.3 Adversarial Robustness

The works of [10, 11] showed that applying small perturbations on images can "fool" a DNN from classifying that image to the right class, when visually the image still looks like the right class. There are many works in this field that try to attack a DNN to reduce its adversarial robustness or defending against these attacks.
To verify a DNN's robustness against adversarial attacks on a specific point, we can define *local robustness*. For a DNN denoted as $f(\cdot)$, an input x, and an l_∞ norm bound defined by ϵ we say that $f(\cdot)$ is locally robust at sample x with ϵ bound if:

$$\forall \tilde{x} : ||x - \tilde{x}||_\infty \leq \epsilon \Rightarrow f(x) = f(\tilde{x}) \tag{1}$$

This local robustness is against untargeted attacks, as we do not care which class we get for an adversarial example, as long as it is different than the class of the given sample.
In this work, using Marabou for local adversarial robustness has 3 available outputs:



(a) Original sample image x - classified as the number 2

(b) Perturbed adversarial image $\tilde{x}$ - classified as the number 5

Figure 1: Adversarial Example with $\epsilon = 0.1$

ROBUST - meaning no adversarial example exists in the l_∞ ball of radius ϵ around the example x, meaning that equation 1 holds.
NOT ROBUST - meaning Marabou found (and returned) an adversarial example inside the l_∞ ball of radius ϵ around the example x, meaning that equation 1 does not hold.
TIMEOUT - meaning Marabou ran for a specified time and didn't find an adversarial example nor proved robustness during that specified time, meaning we do not know if equation 1 holds.
There are different validation tools that can be used for this task such as [2, 1, 12]. I focus only on l_∞ norm since it is very popular in the field and Marabou supports only it, but it is worth mentioning that some works support all l_p norms. One of these is FGP [2], which I will compare some results against.

2 Method

I will show a comparison of using Marabou against the DNN's presented in FGP [2]. We will work with DNN's that receive images as an input. Each pixel in the input has 8 bit color range shifted from $\{0, 1..., 255\} \rightarrow [0, 1]$. In figure 1 there is an example of such an adversarial example found using Marabou, fooling a network I trained on the MNIST digits to miss-classify what seems as the digit "2" wrongly as "5"(specific implementation details on that in subsection 2.2). By showing this example, we give a counter example to equation 1 and show that for $\epsilon = 0.1$ and the given sample x local robustness does not hold. If no such example exist, then equation 1 does hold.
All the networks I'll present ahead are networks that perform tasks on the MNIST data. Usually neural networks that classify images contain convolutional

layers. Since every convolution is a linear operator that can be performed with a matrix multiplication, the presented architectures use only matrix multiplication layers, denoted as fully connected layers (see here for more info about this).

	FGP					Marabou			
Model	Time(s)	R	NR	U	TO	Time(s)	R	NR	TO
mnist20x3	0.004	94	0	6	0	0.066	100	0	0
mnist20x6	0.016	95	0	4	1	0.161	100	0	0
mnist20x9	0.021	84	4	8	4	0.207	100	0	0
mnist40x3	0.094	91	2	5	2	0.135	100	0	0

Table 1: Comparison of l_∞ local robustness (FGP vs. Marabou) on 100 arbitrary test instances that were used in the FGP paper, with a time budget of 120 seconds, including the median runtime (in seconds)and the result: either "robust" (R), "not robust" (NR), "unknown" (U), or a "timeout" (TO). Results are for $\epsilon = 0.01$.

2.1 Comparing FGP networks

In the FGP [2] paper they supply 4 DNN's that were trained on the MNIST data set. the details on these networks:
1. 3 hidden layers with 40 parameters each - denoted as *mnist40x3*.
2. 3 hidden layers with 20 parameters each - denoted as *mnist20x3*.
3. 6 hidden layers with 20 parameters each - denoted as *mnist20x6*.
4. 9 hidden layers with 20 parameters each - denoted as *mnist20x9*.

These networks were trained using PGD [11] adversarial training, with the full implementation details in the FGP paper under Appendix C.
Using Marabou I performed a comparison of l_∞ local robustness of the given networks in the same setting it is shown in FGP, under Table 2. Results are shown in table 1.
Note that the measured time for Marabou is the time reported by the Marabou framework, but the actual running time is a few orders of magnitudes bigger (seconds per example) as there is overhead of calling the C libraries that run inside the Marabou package.
Note that using Marabou we get for this specific small $\epsilon = 0.01$ that all the tested examples are locally robust for all the tested networks. On the contrary, FGP finds non robust examples (e.g. for the *mnist20X9* network), meaning there is some discrepancy. It is a bit odd that they find non robust examples when using such a small l_∞ norm bound value of 0.01. In other networks (such as the one I trained presented in 2.2) I ran, I found adversarial robustness only when using a l_∞ norm bound of at least 0.03. I did get adversarial examples for the *mnist-20x3* network when using $\epsilon = 0.2$, so there is no error due to technical issues of converting these models to support Marabou. Moreover, the networks I trained were trained without adversarial training, meaning they should be less

robust than the networks in FGP, and I would expect the minimal non-robust norm bound to be much higher.

2.2 Comparing My Own DNN

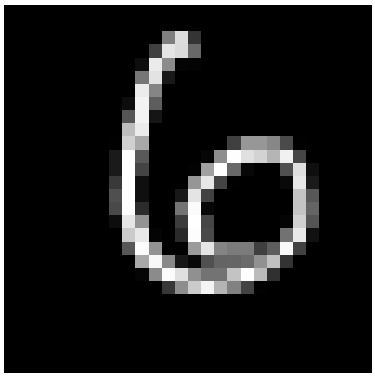
I chose to verify a DNN I trained on the MNIST dataset. The input images that are a $28X28$ pixels grid are flattened to a $28 \cdot 28 = 784$ vector.
The netowrk consists of 3 fully connected hidden layers, denoted h_1, h_2, h_3 and the output layer, out, which is also fully connected. After every hidden layer, a ReLU activation layer is applied. This can all be summed as the DNN f:

$h_1 : [0,1]^{768} \rightarrow \mathbb{R}^{40}$ s.t $h_1(x) = W_1 x + b_1$ where $W_1 \in \mathbb{R}^{40X768}, b_1 \in \mathbb{R}^{40}$
$h_2 : \mathbb{R}^{40} \rightarrow \mathbb{R}^{40}$ s.t $h_2(x) = W_2 x + b_2$ where $W_2 \in \mathbb{R}^{40X40}, b_2 \in \mathbb{R}^{40}$
$h_3 : \mathbb{R}^{40} \rightarrow \mathbb{R}^{40}$ s.t $h_3(x) = W_3 x + b_3$ where $W_3 \in \mathbb{R}^{40X40}, b_3 \in \mathbb{R}^{40}$
$out : \mathbb{R}^{40} \rightarrow \mathbb{R}^{10}$ s.t $out(x) = W_{out} x + b_{out}$ where $W_{out} \in \mathbb{R}^{10X40}, b_{out} \in \mathbb{R}^{10}$
$ReLU : \mathbb{R}^N \rightarrow \mathbb{R}^N$ s.t $ReLU(x_1, ..., x_N) = (max(0, x_1), ..., max(0, x_N))$
$f : [0,1]^{768} \rightarrow \mathbb{R}^{10}$ s.t $f(x) = out(ReLU(h_3(ReLU(h_2(ReLU(h_1(x)))))))$

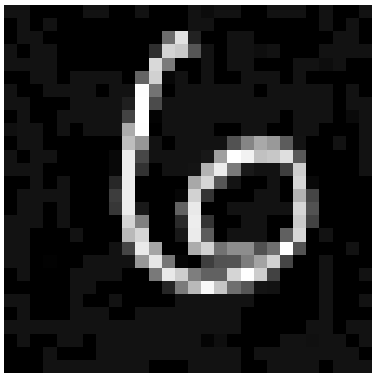
Note that f gives a score for each digit, and if we want to get the prediction of f it is achieved by $argmax_i f(x)_i$.
After training for 15 iterations across the training set, the network achieves accuracy of 98.91% and 97.05% for the train and test sets respectively. Results for using Marabou on theses network with different norm bounds are shown in table 2. We can see that using Marabou, we can conclude that for $\epsilon = 0.02$, we are certain that for these chosen examples the model is robust. However, for bigger norm bounds, since we limited the running time of Marabou, we cannot decide whether robustness holds. As Marabou returned for all these examples either robust/timeout, we do not know whether there is an example that is not robust using that norm bound.
From the last 2 lines of table 2 we can see something very interesting. We see

ϵ	Time(s)	R	NR	TO
0.01	0.181	100	0	0
0.015	0.219	100	0	0
0.02	0.421	100	0	0
0.03	0.34	69	0	31
0.05	>20	16	0	84
0.07	>20	0	16	84

Table 2: Comparison of l_∞ local robustness using Marabou on the DNN I trained with different bounds on 100 arbitrary test instances that were used in the FGP paper, with a 20 seconds time budget, including the median runtime (in seconds)and the result: either "robust" (R), "not robust" (NR), or a "timeout" (TO).



(a) Original sample image x - classified as the number 6



(b) Perturbed adversarial image $\tilde{x}$ - classified as the number 0

Figure 2: Adversarial Example For DNN I trained with $\epsilon = 0.07$

that for 16 examples, we find adversarial examples using $\epsilon \in (0.05, 0.07]$ as when using $\epsilon = 0.05$ we cannot find any right example under that time budget, and when using a bigger one we do find it. An example for one of these adversarial examples is shown in figure 2.

2.3 Project's Repository As Open-Source

Beyond the interesting results of the project, while working on the project I spent a lot of time on trying use Marabou to verify DNN's using python. By doing so, I learned how to convert pytorch networks to support verification with Marabou. I decided to publish it on github to help people from the community while verifying these types of networks.

3 Future Work

Future work can be done in the engineering part, to build an automated pipeline for local robustness verification using both the FGP framework on l_2 bounds, and Marabou on l_∞ bounds.
Another interesting idea would be to extend from fully-connected DNN's to CNN's (convolutional neural networks). This will require a full pipeline for each framework that can convert a CNN to a fully-connected DNN. This will help to use Marabou on many popular neural networks that consist convolutional layers.

4 Conclusion

In this project I showed verification of deep neural networks using the Marabou framework,specifically verifying local robustness of DNN's around input samples. The verification was shown for different image classification DNN's on the MNIST dataset, comparing results with a recent paper and with a DNN I defined. I also provide a useful tool that helps automating local robustness verification of DNN's from the PyTorch popular framework, which isn't currently available directly in Marabou.

References

[1] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing. ISBN 978-3-030-25540-4.

[2] Aymeric Fromherz, Klas Leino, Matt Fredrikson, Bryan Parno, and Corina Păsăreanu. Fast geometric projections for local robustness certification. *arXiv preprint arXiv:2002.04742*, 2020.

[3] Y. Bengio I. Goodfellow and A. Courville. *Deep Learning*. MIT Press, 2016.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29 (6):141–142, 2012.

[5] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[6] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. *Advances in neural information processing systems*, 29, 2016.

[7] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer, 2017.

[8] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, abs/1702.01135, 2017. URL http://arxiv.org/abs/1702.01135.

[9] J. Brush M. Owen K. Julian, J. Lopez and M. Kochenderfer. *Policy Compression for Aircraft Collision Avoidance Systems.* In Proc. 35th Digital Avionics Systems Conf. (DASC), pages 1–10, 2016, 2016.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[12] Matt Jordan, Justin Lewis, and Alexandros G Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. *Advances in Neural Information Processing Systems*, 32, 2019.