CS 781 Project Report
Verifying L1 Adverserial Robustness

Rohan Gupta
180010048@iitb.ac.in

Mohammad Taufeeque
180050062@iitb.ac.in

Abstract. We study the performance of popular neural network verification tools on L_1 perturbations. Since feeding L_1 constraints to solver is exponential in the order of inputs, we experiment with function approximators to possibly reduce the complexity. We give our results on two tools, Marabou (Reluplex) and ERAN (DeepPoly) respectively.

Keywords: Verification of Neural Nets · L_1 attack

1 Introduction

Most verification tools show strong experimental results only for L_∞ perturbations. However, attacks under other L_p norms are becoming increasingly popular. It has also been observed that models robust to L_∞ are vulnerable to even small, perceptually minor departures from this family, such as small rotations and translations. Therefore, there is a need to look for high performance under other perturbations. The main issue with L_∞ perturbations lies in making the bounds larger. Since the pixel-wise perturbations are independent, it can easily cross the decision boundaries. [2] is an example of such adversaries, whose L_∞ norm is large, but other L_p norms are not.

We evaluate the performance of extant tools on L_1 perturbations, with some modifications: instead of giving exponential number of equations to the solver, an auxilliary network is used to find the L_1 norm(conditions are then applied to this output). We also observe some key differences between L_1 and L_∞ perturbations via our experiments. We show:

- Over approximations of DeepPoly [3] result in poor performance in the case of L_1 norms, and in our formulation, approximates it to it's minimal L_∞ super-set space.
- Strong L_∞ perturbations limited to only a small region often leads to misclassification.
- A way to reduce large L_∞ perturbation regions so that it does not tip over the decision boundaries and a way to reduce spurious counterexamples for high L_1 norms.

2 Implementation Details

2.1 Abstraction based solver (DeepPoly)

DeepPoly uses an over-approximation by choosing only two bounding lines to represent the abstraction. If L_1 norm constraints were enforced on the input, the exact approximation would require an exponential number of constraints. The over approximation is very bad in this case. Figure 1 show the over approximation for a 2-pixel toy example. We therefore use an independent MLP network to output the L_1 norm and apply the needed constraints inside it. The L_∞ bounds for the input can then be chosen to be a super set of our original L_1 constraints. Figure 2 shows a network calculating the L_1 norm for a single pixel. We can extend this to multiple pixels by creating two neurons for each pixel pair in the first layer, and adding them in the final layer to get $\sum_i |p_i|$. We denote these weights by $\{(W^i_{norm}, b^i_{norm}), i \in \{1, 2\}\}$

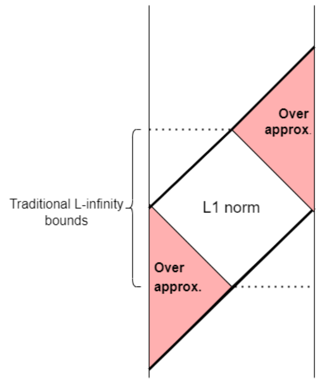

Figure 1: L1 norm approximation for a 2-pixel case

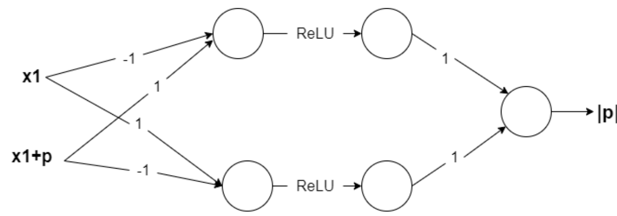

Figure 2: L_1 norm for a single pixel

Trained Network

Since the libraries did not support operations like concatenations, tensor splicing and multiple inputs/outputs, we could not use independent networks for verification. We therefore make a sparse mlp network with multiple functionalities, with weights:

$$W^i_n = W(W^i_{mnist}, W^i_{norm}) = \begin{pmatrix} W^i_{norm} & \mathbf{0} \\ \mathbf{0} & W^i_{mnist} \end{pmatrix} \tag{1}$$

$$b^i_n = b(b^i_{mnist}, b^i_{norm}) = \begin{pmatrix} b^i_{norm} \\ b^i_{mnist} \end{pmatrix} \tag{2}$$

given the trained weights $\{(W^i_{mnist}, b^i_{mnist}), i \in \{1, 2, 3\}\}$. The network then gives an 11-dimensional vector $(L_1, c_0, c_1..c_9)$, where c_i are the class probabilities of the MNIST image and L_1 is the L_1 norm. The neural network used had two linear-ReLU layers of 100 neurons each and a final layer mapping to the 10 classes with a softmax. Since our auxilliary network has 2 layers we add an additional identity layer to it: $W^3_{norm} = 1$, $bs_{norm} = 0$. The final network verified was on the logits as the network in Figure 2 cannot have an a softmax operation. Note that this is extendable to any mlp with the appropriate number of identity layers added.

Encoding the condition

For a given L_1 bound b_0, the condition to verify within an L_∞ region becomes

$$(L_1 \le b_0) \implies (y_{\max} = y_{\text{label}}) \tag{3}$$
$$\iff \neg(L_1 \le b_0) \lor (y_{\max} = y_{\text{label}}) \tag{4}$$
$$\iff (L_1 > b_0) \lor (y_{\max} = y_{\text{label}}) \tag{5}$$

The condition $L_1 > b_0$ is encoded by checking the lower bound of the output L_1's abstraction. Note that, however, it can easily be verified that the lower bound for L_1 in our network is just 0. Therefore $(y_{\max} = y_{\text{label}})$ must hold for the entire abstraction for the above condition to hold. Hence, the over-approximations in deeppoly reduces the problem to checking L_∞ perturbations itself. We observe this phenomenon in our experiments as well.

2.2 Exact Solver(Reluplex)

An exact solver [1] will not make approximation and hence we can directly feed the L_1 constraints to the solver. However, feeding L_1 norm constraints would need 2^{pixels} equations is not practically feasible. We observe that a region with mere 16 pixels causes memory issues on a standard computer. Using and auxilliary

L1	Result	L1*stddev
0	UNSAT	0
2	UNSAT	0.6
10	UNSAT	3
20	UNSAT	6
30	SAT	9

Table 1: Robustness results for different L_c. L_∞ norm was fixed at 0.2

network hence will help us bypass these issues. We perform experiments on both approaches, feeding L_1 constraints and using an auxilliary network. We limit to 12-pixel regions for the former, however. The auxilliary network used is same as the one defined in the above section.

Encoding the condition

Reluplex uses Hoare Triples. The condition 5 hence will be negated, and the network is verified only if $(L_1 \leq b_0) \wedge (y_{\max} \neq y_{\text{label}})$ gives UNSAT.

3 Experiments and Results

3.1 DeepPoly

We pass 50 different MNIST images through our implementation with various L_∞ perturbations. The network with L_1 norm constraint behaves exactly like the network without L_1 norm constraint. On further analysis, we found that DeepPoly uses the lower bound on L_1 to check if it is greater than the provided bound. However, since the lower bound of L_1 on all the inputs is 0, the condition never gets satisfied and thus the implementation always ends up verifying the output constraint on the whole L_∞ input space rather than the constrained L_1 input space. We explored different options with DeepPoly like the –complete argument, –domain argument with domains like deepzono and deeppoly, but none of the options gave a different result. Moreover, we tried to see whether we could use trace partitioning on our input space with DeepPoly but found that the option for trace partitioning only exists for two specific cases: 1) geometric transformation on an image, and 2) on the Acasxu dataset. On a higher dimensional space like images, trace partitioning would take $\mathrm{O}(\mathrm{n}^{\mathrm{pixels}})$ time, where n is the number of partitions per dimension.

3.2 Reluplex

The L_1 norm constraint on the entire image takes too long to run on Reluplex. Hence, we could only try the experiment on few images. We conduct 3 experiments for these few images:

(i) Perturb small regions only, giving explicit constraints for these regions. Note that these perturbations are not a subset of L_∞ if the norms are increased to the maximum change possible for each pixel.

(ii) Perturb small regions using the auxilliary networks instead.

(iii) Perturb entire image.

We can further consider 3 types of norms in each experiment, namely, L_∞, L_1 and *clipped* norms given by $L_c = \min(L_\infty, L_1)$, where $L_\infty \leq \delta, L_1 \leq \epsilon$, and $\epsilon > \delta$. The benefit of L_c can be seen by considering large L_∞ and L_1 norms as our search spaces: any $L_1 > 1$ is not valid for images who's values lie between 0 and 1, and large L_∞ bounds cross the decision boundaries easily. The clipped norm manages to avoid both these issues. Table 1 shows the robustness of our sparse network on different L_c norms.

Since perturbing different regions using explicit constraints is limited to 3x4 pixels, the reluplex gives UNSAT almost all the time. We run experiments (i) and (ii) for 173 regions. (i) was run with maximum L_∞ bounds(=1) and no L_1 restrictions and it gave UNSAT for all values. (ii) was run on a bigger region (16x16)

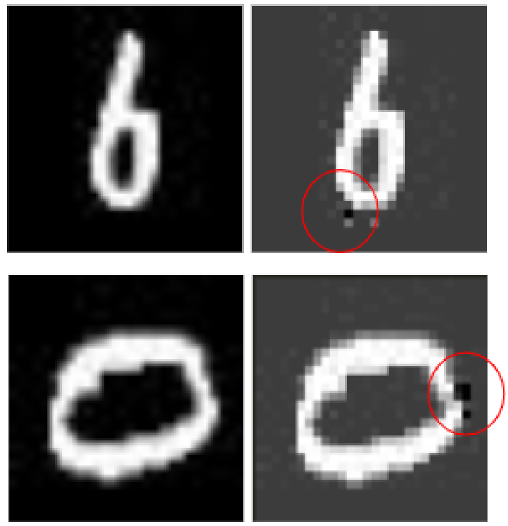

Figure 3: Adversaries obtained by strongly perturbing regions. Note that the image seems to have a large difference with the original one as the figure is displayed after normalising the perturbed image.

once with maximum possible perturbations(Figure 3 shows the obtained adversaries), and with L1 norm constrained to 1(113/176 were verified correctly in this case). The adversaries examples are skipped here for conciseness, but can be obtained by running our code(see Section 5) on default settings.

4 Conclusion

Through our experiments, we conclude that L_1 norm constraint verification on DeepPoly is not possible using an auxiliary network because of the way the abstractions are constructed. There is a need for a general purpose way to trace partition the input like DeepPoly does on the geometric transformations. With Reluplex, the verification is infeasible when the entire image is allowed to perturb. Reluplex can only handle the constraints on a small region of the image. Therefore, we note that the current robustness analyzers are not capable of handling L_1 perturbations.

References

[1] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification*, pages 97–117. Springer International Publishing, 2017. doi: 10.1007/978-3-319-63387-9_5. URL https://doi.org/10.1007%2F978-3-319-63387-9_5.

[2] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Adversarial Generative Nets: Neural network attacks on state-of-the-art face recognition. arXiv preprint 1801.00349, Dec. 2017. URL https://arxiv.org/abs/1801.00349.

[3] G. Singh, T. Gehr, M. Püschel, and M. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019. doi: 10.1145/3290354. URL https://doi.org/10.1145/3290354.

5 Links to Implementation

Reluplex: https://github.com/cybershiptrooper/Marabou
DeepPoly https://github.com/taufeeque9/eranL1