Studying Robustness of Semi-supervised Visual Features to Adversarial Attacks

Aseem Saxena*
MS in Robotics
Oregon State University
Corvallis, Oregon 97331
saxenaa@oregonstate.edu

Abstract

Neural Network Verification is an important tool towards gauging robustness to adversaries. In this report, I summarise the work of [8] who formulate most past work on LP based neural network verification as a convex relaxation problem. The framework can handle different activation functions and pooling layers and also can handle both primal and dual versions of verification. In my work, I try to evaluate the adversarial robustness of classifiers which are trained to simultaneously classify as well as reconstruct the input. I focus on two domains, image classification on the CIFAR10 dataset and Q-Learning in the OpenAI gym cartpole environment.

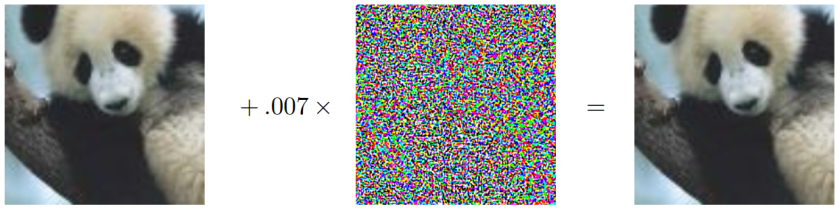

Figure 1: Example to illustrate an Adversarial Attack on a Machine Learning model, Source [5]

1 Motivation

Machine Learning, especially Deep Learning Models are increasingly being deployed in mission critical scenarios like Autonomous Driving. Neural Networks have been shown to be vulnerable to adversarial perturbations, meaning, slightly modifying the visual input which would be visually indiscernible to a human can change the prediction of the neural network processing it. This allows malevolent actors to wreck havoc.

Neural Network Verification can help us study the extent to which a model is robust to adversarial perturbations. There are two approaches towards robustness verification for Piece-wise Neural Networks (ReLU networks being a subset of it). MILP(Mixed Integer Linear Programming) solvers or SMT(Satisfiability Modulo Theories) solvers which return accurate results but tend to be much slower and as the problem is NP-complete, these methods don't scale well to large networks. Relaxed

*https://aseembits93.github.io/

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

and Efficient Verifiers work primarily by relaxing non linear constraints into linear constraints and studying either the primal or the dual version of the relaxed problem.

1.1 Problem Formulation in the Primal Space

Adversarial Robustness for a classifier with respect to an input x and its neighborhood $\mathcal{S}_{in}(x)$ is defined by

$$\min_{x' \in \mathcal{S}_{in}(x), i \neq i^*} f_{i^*}(x) - f_i(x') > 0, \quad \text{where} \quad i^* = \arg\max_j f_j(x). \tag{1}$$

f denotes the neural network and the subscript i represent the ith logit of the prediction. The neighborhood $\mathcal{S}_{in}(x)$ is usually in the ∞-norm sense i.e $\mathcal{S}_{in}(x^{\text{nom}}) = \{x : \|x - x^{\text{nom}}\|_\infty \leq \epsilon\}$ which is a convex set.

The approach taken in this work is to find a lower bound for eq. (1) by formulating it as a Linear Programming problem. Finding the optimal value as positive would ensure that the model is robust.

Equation eq. (1) can be transformed into a LP in the following way

$\mathcal{O}(c, c_0, L, \underline{z}^{[L]}, \overline{z}^{[L]})$:

$$\begin{aligned} \min_{(x^{[L+1]}, z^{[L]}) \in \mathcal{D}} \quad & c^\top x^{(L)} + c_0 \\ \text{s.t.} \quad & z^{(l)} = W^{(l)} x^{(l)} + b^{(l)}, l \in [L], \\ & x^{(l+1)} = \sigma^{(l)}(z^{(l)}), l \in [L], \end{aligned} \tag{$\mathcal{O}$}$$

where x^{l+1} denotes the pre-activation output of layer $l \in [L]$ and z^l is the post activation output. In this project, I specifically focus on the ReLU activation.

eq. (1) and $\mathcal{O}$ become equivalent when c^T is set to $W^{(L)}_{i^{\text{nom}},:} - W^{(L)}_{i,:}$ and c_0 is set to $b^{(L)}_{i^{\text{nom}}} - b^{(L)}_i$.

This is a difficult optimization in its current setting because, Firstly, as the bounds for z^l are unknown, this makes the search space really huge. By calculating upper and lower bounds for z^l we can drastically reduce the search space. One more thing which makes this problem difficult is that the activation function σ (ReLU) is non-linear, which makes the feasible set of the problem as non-convex leading to NP-completeness.

Bounding $z^{[L]}$ Recursive solving of $\mathcal{O}$ starting from some specific choices for c_0 and c^T. This is done by [10, 3].

Convex relaxation of feasible set to form convex constraints We can relax the nonconvex equality constraint $x^{(l+1)} = \sigma^{(l)}(z^{(l)})$ to convex inequality constraints, i.e.,

$$\min_{(x^{[L+1]}, z^{[L]}) \in \mathcal{D}} c^T x^{(L)} + c_0 \quad \text{s.t.} \quad z^{(l)} = W^{(l)} x^{(l)} + b^{(l)}, \underline{\sigma}^{(l)}(z^{(l)}) \leq x^{(l+1)} \leq \overline{\sigma}^{(l)}(z^{(l)}), \forall l \in [L], \tag{$\mathcal{C}$}$$

The optimal value of the convex relaxed problem is $p^*_{\mathcal{C}}$ and it can be shown that $p^*_{\mathcal{C}} \leq p^*_{\mathcal{O}}$ since

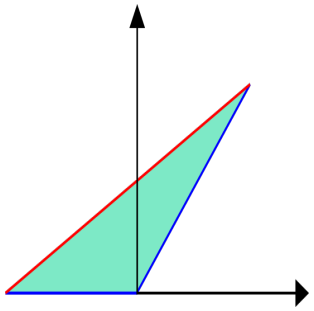

Figure 2: Convex Relaxation of the ReLU activation function. Geometrically, it looks like a triangle for some bounded input. Mathematically, the relaxation can be expressed as 3 linear inequalities

the original feasible set is now a subset of the convex relaxed feasible set. Ehlers [4] proposed the relaxation for the ReLU non-linearity as

$$\underline{\sigma}_{ReLU}(z) = \max(0, z), \quad \overline{\sigma}_{ReLU}(z) = \frac{\overline{z}}{\overline{z}-\underline{z}}(z - \underline{z}), \tag{2}$$

1.2 Problem Formulation in the Dual Space

The authors show that under some mild conditions, strong duality holds true for $\mathcal{C}$. The authors also show that the convex relaxation for the dual problem cannot do better than convex relaxation for the original problem. More details can be found in [8]

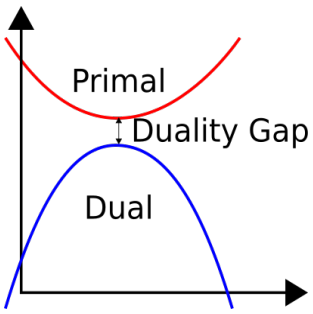

Figure 3: A simple figure showing the duality gap, the gap is 0 when slater's condition is true

1.3 LP-relaxed verification methods

The authors have unified past approaches on LP based verification and benchmark three such methods.

- LP-All
  Proposed by the authors. Essentially, this method works by firstly obtaining pre-activation bounds, starting from the 0-th input layer with the l-∞ ball input and solving LPs in parallel for all the neurons in the first hidden layer and then moving on to the next layer and so on till the penultimate layer. Secondly, after getting the preactivation bounds, by setting

  $$L \leftarrow l_0, \quad c^\top \leftarrow W_{j,:}^{(l_0)} \text{ (resp. } c^T \leftarrow -W_{j,:}^{(l_0)}), \quad c_0 \leftarrow b_j^{(l_0)} \text{ (resp. } c_0 \leftarrow -b_j^{(l_0)})$$

  The LP is solved exactly to get margins and if all of them are positive than the model is robust.
- LP-Greedy
  Proposed by Wong and Kolter [10]. Essentially, this method works by converting the problem to its dual my modifying the original network by adding layers on top of the final layer and optimizing it in an iterative manner similar to SGD with additional loss terms corresponding to the dual terms.
- LP-Last
  Similar to LP-All but in order to find the preactivation bounds, it uses the first part of LP-Greedy but solved the LP exactly in the second step.
- MILP
  Mixed Integer Learning formulation, proposed by [9]. The authors proposed the ReLU non-linearity as a Mixed Integer Linear Program with a binary indicator variable to denote if the input is more than 0 or not. They subsequently solve it to get tight bounds on preactivations.

2 My Idea and Hypothesis

Semi supervised learning seems to help in learning faster and prevents overfitting[7]. But adversarial robustness in the context of semi supervised learning hasn't been explored much. My idea is that reconstruction from latent space could help the network learn the structure of the data and subsequently learn to avoid noise in the data.

In my experiments, I train a classifier which simulatenously is trained to reconstruct its input. I have three network variants -> Vanilla (trained just for classification), AE (trained with reconstruction and classification from latent representation), VAE (similar to AE but using a variational autoencoder for reconstruction). The loss function I use for CIFAR10 is as follows

$$Loss = CE(prediction, label) + \lambda_r * MSE(reconstruction, input) + \lambda_k * KL_Divergence \quad (3)$$

Where CE stands for Cross Entropy Loss, MSE stands for Mean Squared Error Loss, λ_r and λ_k are scaling factors for the auxiliary losses. Note that λ_r and λ_k are 0 for Vanilla variant, and λ_k is 0 for AE variant.

For RL agents, I'm doing Q-learning via DQN which relies on a target network to generate ground truth and the loss is typically a mean squared error loss.

Autoencoders tend to overfit and variational autoencoders prevent that by regularizing for minimizing the KL-divergence between the input and the reconstruction. This is achieved by learning the disribution of the input by simultaneously learning the mean and standard deviation, assuming that the distribution is a multivariate gaussian.[1]. I focus on the CIFAR10 dataset for my experiments and have both linear and convolutional models for comparison.

I was also curious to try my approach in an Reinforcement Learning setting, where things are a bit difficult as I observed that you need an expert policy to get ground truth labels for a state. I focus on two environments here, the OpenAI cartpole environment and the Atari Pong environment.[2]

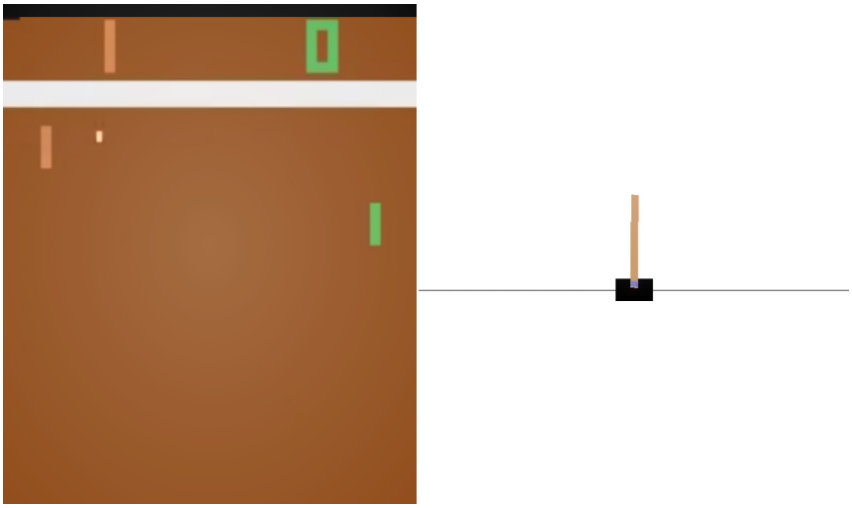

Figure 4: Description of the Cartpole and Pong environments

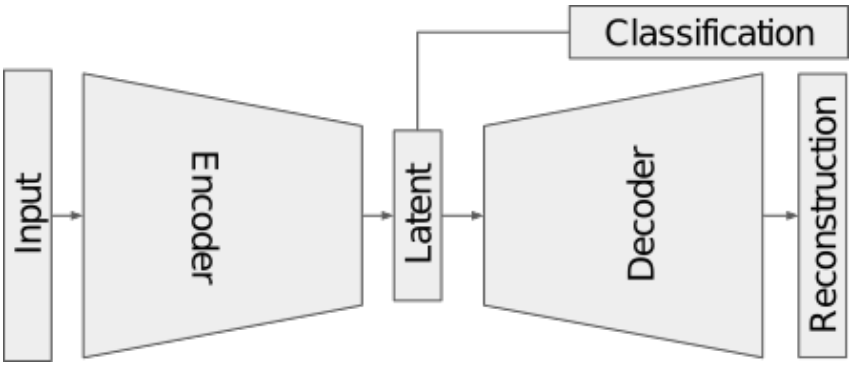

Figure 5: Description of model for augmenting classification with reconstruction

Table 1: Comparison of methods for adversarial robustness certification, with a binary Y/N answer and algorithm runtime in seconds in brackets. Notice the disparity in runtime for LP-greedy vs LP-Last vs MILP.

Network	Epsilon	LP-Greedy	LP-Last	LP-all	MILP
CIFAR-Lin-Van	0.001	Y(0.4)	N(318.1)	N/A	Y(5080)
CIFAR-Lin-AE	0.001	Y(0.25)	N(445.82)	N/A	Y(3587)
CIFAR-Lin-VAE	0.001	N(0.2)	N(379.6)	N/A	N(40348!)
CIFAR-CNN-Van	0.001	N(0.28)	N/A		
CIFAR-CNN-AE	0.001	Y(0.27)	N/A		
CIFAR-CNN-VAE	0.001	Y(0.29)	N/A		
Cart-Lin-Van	0.001	Y(0.09)	N/A		
Cart-Lin-AE	0.001	Y(0.1)	N/A		
Cart-Lin-VAE	0.001	Y(0.11)	N/A		
Pong-CNN-Van	0.001	N(7.4)	N/A		
Pong-CNN-AE	0.001	N(7.21)	N/A		
Pong-CNN-VAE	0.001	Y(6.8)	N/A		

3 Experiments and Results

In my experiments, I fix epsilon to a constant value for a fair comparison of all the models. For some epsilon, I find if the network is certified to be robust using 4 different methods - LP-Greedy, LP-Last, LP-all and MILP as explained before. I have two kinds of models - Linear with only fully connected layers and ReLU activation and CNNs with only convolutional layers and ReLU activation.

For the RL environments, I train an expert agent first to get ground truth action labels for state inputs. RL is notorious for being sample inefficient and it took me 12 hours to train an optimal agent. I then train the three network variants till satisfactory performance was observed. For the cartpole environment, I focus on fully connected models as the state is only 4 dimensions. For the pong environment, I use CNNs since the input is 84X84X4 (4 gray-scale images stacked from time t-3 to t).

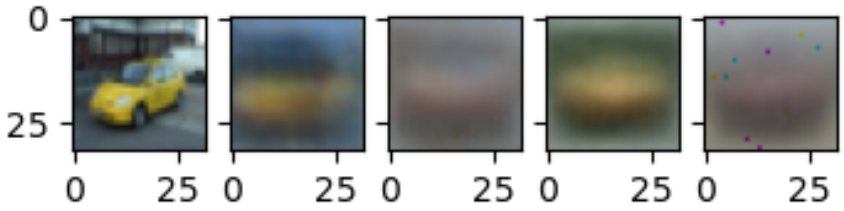

Figure 6: Reconstruction of my models on a CIFAR10 image, Original image, CNN Auotencoder, CNN Variational Autoencoder, Linear Autoencoder, Linear Variational Autoencoder

4 Conclusion and Future Work

MILP and LP-All are extremely computationally expensive as I observed, Also, my networks very much larger than the ones presented by the authors. This puts to question the scalability of such methods to modern machine learning models and datasets. CIFAR CNN and Pong CNN seem to show that adding an extra reconstruction loss can help in ensuring that the network is adversarially robust, provided all other training details are the same. This is encouraging and needs to be explored further with more rigourous experiments.

Denoising autoencoder work by adding noise and then reconstructing for the original image, I think that these models would be very suitable for robustness.

One interesting approach is to do PGD training [6] and see if the network is robust at higher epsilon values.

Due to time and computational constraints, I could not tune my hyperparameters which I believe, could have helped with robustness.

The main takeaway is that the structure of the data can give a lot of information which could be helpful in learning and help models be robust.

References

[1] Jaan Altosaar. *Tutorial - What is a Variational Autoencoder?*, August 2016. URL `https://doi.org/10.5281/zenodo.4462916`.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[3] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *UAI*, 2018.

[4] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[6] 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.

[7] Thomas Robert, Nicolas Thome, and Matthieu Cord. Hybridnet: Classification and reconstruction cooperation for semi-supervised learning, 2018.

[8] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks, 2020.

[9] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyGIdiRqtm`.

[10] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, pages 5283–5292, 2018.