Defending the First-Order: Using Reluplex to Verify the Adversarial Robustness of Neural Networks Trained against Gradient White Box Attacks

Adam Pahlavan
Stanford University
adampah@stanford.edu

Daniel Lee
Stanford University
dan9lee@stanford.edu

Justin Rose
Stanford University
justrose@stanford.edu

[1]

Abstract

Deep Neural Networks (DNNs) have shown tremendous progress in accurately performing regression, classification, and control tasks that traditional programming paradigms have not been able to achieve. However, their lack of interpretability and performance guarantees presents challenges when considering deployment to sensitive tasks with limited margin of error, such as control of autonomous vehicles, military equipment, or robotics. One pressing problem with modern neural networks is the existence of adversarial examples, where small perturbations in inputs can result in dramatically different output classifications. To combat this problem, developing adversarially-robust networks is an active area of research. Currently, model robustness is evaluated against first-order adversaries, i.e. adversaries generated via gradient methods. It is an open research question whether these first-order methods are good metrics for testing the overall robustness of neural networks to all adversaries. To tackle this problem, we use a neural network constraint solver developed by Stanford's Reluplex team. By using a novel algorithm that optimizes the number of linear constraints that must be checked, certain properties about general robustness can be guaranteed for all attacks, rather than only first-order ones. We examine the robustness of two simple models, one trained only on MNIST data and one adversarially trained using a modern first-order adversarial defense strategy called logit pairing. In both networks, we found that the first-order gradient well-approximates the guaranteed closest adversary and that adversarially training against first-order attacks generalizes to all attacks. Future directions involve scaling Reluplex to handler deeper networks to verify this trend.

[1]Code: https://drive.google.com/file/d/13SUQfZ1nezYu-AlL1kyCDOdm8wn4CnOS/view?usp=sharing

1. Introduction

Following the success of applying convolutional layers to image recognition[7], neural networks have gained wide use in advanced machine learning systems. Current networks can outperform humans at many tasks, including image recognition[3], arcade games [10], and board games like Go [12], and they have become the de facto approach to achieve top performance on tasks traditional algorithms have been unable perform consistently on.

Although they outperform humans at many things, there are downsides to the use of modern neural networks. Firstly, deep neural networks lack interpretibility: although they get the correct answer, we are often not sure why. Unlike traditional coding pipelines where modularity enables manual proofs of correctness and isolated failures, deep end-to-end models are notoriously hard to interpret, and when they fail it's difficult to understand why. Although some advances have been made in determining what each filter might represent in a convolutional model [13], for more complex games like Go or Atari, understanding their reasoning is less clear.

Adversarial examples [2] represent an even larger problem for modern DNNs. By perturbing input images even slightly in the direction of the gradient of a loss function weighing an incorrect network output, new examples can be generated that are visually identical to existing examples but that networks consistently misclassify with high confidence. Such errors on visually imperceptible differences create a major challenge for networks to have consistency guarantees in the real world, especially when such examples can be generated without access to the network at hand[11].

An interesting question is what guarantees can even be made about networks being robust against adversarial examples, especially ones trained to defend against these adversarial attacks. Although this problem is NP-hard, recent work by the Reluplex [5] team has created a system for verifying properties and constraints of networks with the non-convex Rectified Linear Unit (ReLU) activation function. With this method, we can find the minimum perturbation an input must undergo in order to be misclassified, thus finding

the closest adversarial example for a certain input.

Prior to complete solvers like Reluplex, first-order attacks that use the gradient of a network to compute adversarial images have been the standard to both train and test for adversarial robustness. However, it is an open research question whether these gradient-based defenses actually improve robustness against all attacks or just first-order attacks. In our project, we evaluate the robustness properties of two simple MNIST classifiers [8]; one trained purely on clean data, and one adversarially trained with the state of the art defense logit pairing [4]. For both networks, we compare closest adversaries found via Reluplex and first-order methods to determine if there exist closer adversaries that first-order methods cannot find. We aimed to shed light on the following questions:

1. Are first-order attacks good universal indicators of robustness?

2. Do first-order defenses generalize to non-first-order attacks?

2. Related Work

Much work has been devoted to protecting models from adversarial examples. Generally, adversarial examples can be generated in two ways: white box attacks, where the attacker has access to the network and its weights, and black box attacks, which don't have access to network specifics [11]. The focus of our study is on white box attacks. A common method for generating white box attacks is projected gradient descent [9], considered the closest "first-order adversary", where a random point in a C^{∞} ball around a data example starts as a seed and iteratively follows that gradient of the network's loss with respect to the input to generate an adversarial example.

Adversarial training against first-order adversaries is the main method of making models adversarially robust, where models are trained on both the train data distribution and an adversarial distribution generated from that data distribution [2]. In most recent papers, these adversarial examples were generated using projected gradient descent [9]. A recent development in adversarial training has been adversarial logit pairing [4], where the logits of the outputs of an example and it's adversary are encouraged, through a separate loss term, to be similar, resulting in state of the art performance on first-order attacks.

One recently discovered problem with determining the robustness of models against adversarial examples is the issue of gradient masking, where the defense techniques make the gradients of a network less useful in generating adversarial examples, but don't make them less susceptible to examples generated through another mechanism[1]. These obfuscated gradients, which hinder iteration-based first-order attacks, have been shown to give networks a false sense of security, where first-order robustness is a result of bad gradients rather than a sound defense. One of the main advantages of using a complete solver like Reluplex is that robustness is *guaranteed*, because it solves for examples within nearby L^{∞}-δ balls.

3. Methods

3.1. Marabou/Reluplex

Reluplex is a decision procedure to solve linear equations that have non-linear constraints, such as neural networks with ReLU activation functions. As mentioned earlier, this task is typically NP-hard, making it intractable to compute in the worst case. However, the novelty of Reluplex [5] is that, instead of searching the 2^n potential states of ReLU nodes as being active or inactive, it lazily solves an initial configuration and is able to infer from it that certain constraints must be fixed in an active or inactive state, allowing it to scale to larger networks, such as the ones we examine. Through using this linear solver, we are able to explicitly find whether adversarial examples exist near our data and test distribution. One subtlety to note is that Reluplex can't deal with softmax activation functions. However, since the classified category of our network is simply the maximum index in our logits layer, we don't need softmax to see when we have a misclassification error.

When translating what robustness of a network means, we use the definition by the Reluplex team [6]:

A network N is δ-locally robust at a point $\vec{x_0}$ iff

$$\forall \vec{x}, \|\vec{x} - \vec{x_0}\| \leq \delta \quad \rightarrow \quad N(\vec{x}) = N(\vec{x_0})$$

In English, the above definition is basically saying a point is δ-locally robust if the δ-L_{∞} ball around x_0 contains no misclassification errors. In Reluplex, we encode the negation of this robustness property to solve for a δ-norm perturbation which results in a misclassified image. Running Reluplex was the clear bottleneck of our project, as it could sometimes take up to 10 minutes to verify whether a value of δ was robust or not. As expected, adversarial guarantees require a fair amount of compute, and Reluplex has yet to be optimized for parallel computing.

3.2. Baseline Vanilla MNIST Network

We trained a simple multilayer perceptron (MLP) network (i.e. a model we learned in class that is NOT a deep model) to classify 28x28 pixel images from MNIST, a database with 60,000 training and 10,000 testing images of handwritten digits. We divide the pixel values by 255 to normalize the pixels between 0 and 1. The MLP consisted

of a single hidden layer with 50 nodes with ReLU activations and an output layer with 10 nodes and a softmax activation, corresponding to class probabilities. We train with respect to a categorical cross entropy loss using an Adam optimizer with learning rate $1.0 * 10^{-4}$. We trained for a total of 60,000 batches of batch size 64, or 64 epochs, by which time our network had converged.

3.3. Adversarially Trained Network

3.3.1 Adversarial Training

Additionally, we created an adversarial multilayer perceptron, which started pretrained with weights from our initial vanilla model. Basic adversarial training involves the following: for each batch of examples $\{x^{(1)}, \ldots, x^{(m)}\}$, a set of adversarial images is generated $\{\tilde{x}^{(1)}, \ldots, \tilde{x}^{(m)}\}$ with respect to the current network weights. Typically these adversarial images are generated using some kind of first-order method (i.e. involving some gradient). The first-order method we chose to use was the iterated Fast Gradient Sign Method (FGSM), which iterated over an input image with the update rule:

$$x^{t+1} = x^t - \alpha * \text{sgn}(\nabla_x L(\theta, x, y'))$$

where x^0 is the original image, t is the number of time steps, α is the step size, and $L(\theta, x, y')$ is loss of the network output with respect to some adversarial label y' (the class we are trying to misclassify as). We initialize the adversarial search process with a random noise perturbation within our δ-L_∞ ball, and we clip to make sure pixels stay within this δ sized ball. While training, we ran 40 iterations when generating a batch of adversarial examples, which were targeted towards random class labels. The step size for iterated FGSM was simply a fraction of the δ ball and the step size. When adversarially training our network, we ran 250k batches of size 64 (about 265 epochs) where each adversarial batch was generated within a 0.3 normalized L_∞ ball, or a 76.5 pixel ball. Since our accuracy on clean examples began to drop by a notable amount, we trained an additional 250k batches on examples only within a 0.1 normalized L_∞ ball, or a 25.5 pixel ball, so that the aggressive adversaries wouldn't have as large of an impact on the loss. We stopped after these iterations because our network performance had converged.

3.3.2 Adversarial Loss with Logit Training

Since we are typically interested in having a loss that rewards classifying adversaries and clean images correctly, the loss function is typically

$$L_{adv}(X, \tilde{X}, Y) = L_{orig}(X, Y) + L_{orig}(\tilde{X}, Y)$$

where L_{orig} is our typical loss function with respect to an input batch (in this case the cross entropy loss). In addition to this traditional adversarial loss, the modern defense Adversarial Logit Pairing defines the new loss:

$$L_{advlog}(X, \tilde{X}, Y) = L_{adv}(X, \tilde{X}, Y)$$

$$+\lambda \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), f(\tilde{x}^{(i)}; \theta))$$

where $f(x^{(i)}; \theta)$ is a function mapping inputs to the logit layer of a model and L is any loss that promotes closeness of its two inputs (in their paper and ours we simply use L_2 loss). Similar to other regularization methods discussed in class, this added loss is equivalent to the probabilitic assumption that the error in the logit layer between original and adversarial images is best represented by Gaussian noise. While training, we use a value of $\lambda = 1$.

3.4. Finding Closest Adversaries

Finding the closest adversaries was relatively straightforward. First we found the closest adversarial example for iterated FGSM by repeatedly binary searching over our δ balls for FGSM and seeing if the network misclassified it, starting with $\delta_{min} = 0$ and $\delta_{max} = 1$ and attempting for every class as opposed to a random class. We only found these adversaries for examples that a network initially classified as correct. We performed a similar binary search with Reluplex to find the guaranteed closest adversary.

4. Results

4.1. Test Accuracy on Vanilla and Adversarial Trained Network

After training, we evaluated both the Vanilla and adversarially trained models on the 10,000 clean test examples in MNIST and 10,000 adversaries generated from those examples with 0.2 normalized δ-balls generated by FGSM. We found that the Vanilla model got 97% accuracy on the clean examples but only 14% accuracy on adversaries. On the other hand, the adversarially trained network got 96.5% accuracy on the clean examples and 89% accuracy on the adversaries. A concern we had was whether the small capacity architecture we used could learn an adequate MNIST classifier and an adversarially robust model with logit pairing. The significant increase in accuracy on adversaries and comparable accuracy on clean examples in the adversarially trained model when compared to the Vanilla one indicates that adversarial training indeed did increase robustness towards first-order attacks, even under the low-capacity constraint.

4.2. Similarity between FGSM and Marabou adversaries

The next thing we tested was the cosine similarity between the closest adversaries generated via FGSM and Reluplex on both the Vanilla network (Fig. 1) and the adversarially trained network (Fig. 3). Overall, we observed high levels of similarity in the direction of perturbations of the two attacks in both networks. In particular, among all the images tested, neither network found two perturbations with a negative cosine similarity, providing preliminary evidence to support that the first-order attack is a good approximation of the direction of the closest attack.

Notably, the cosine similarity of the perturbations in the adversarially trained network were notably lower than in the Vanilla one. We hypothesize a few possible explanations for this behavior.

First, we can think of our MLP as a piece-wise linear model with 2^{50} different configurations depending on the activation state of the ReLUs. Since we are searching a larger δ ball in the adversarially trained network, there are a greater number of possible linear modes that the network could be in, as there is a wider input space over which the activation state of a ReLU could flip. Therefore, it is possible that the greater variation in linear modes creates potential adversaries in different linear modes that gradients will not find, which posits an explanation for the lower cosine similarity. Alternatively, in the Vanilla network, the model behaves "more linearly" in the sense that there are a restricted number of linear modes. This provides an explanation for why the perturbations were more aligned, as the only way to minimize a loss in a linear model is through the first-order gradient.

A second explanation is Athalye *et al.*'s hypothesis of obfuscated gradients [1], which hypothesizes that first order defense techniques cause the gradient to become less effective at generating adversarial images after training. Although in his paper, such forms of "gradient masking" typically occurred in deeper networks, this could potentially also be the beginning of such a phenomenon, which could be further supported by testing deeper networks.

Examples of perturbation alignment and nonalignment (in the case of the adversarially trained) for each network and each adversary method can be seen in Figures 2 and 4. Notably, we can notice for high levels of cosine similarity, both perturbations look nearly identical, which supports that the first-order attack is a good approximation of the closest adversary in many cases.

4.3. Similarities in δ values for FGSM and Marabou

We compare the δ-robustness (defined as the distance of the closest adversary) in both the Vanilla network (Fig. 5) and the adversarially trained network (Fig. 6).

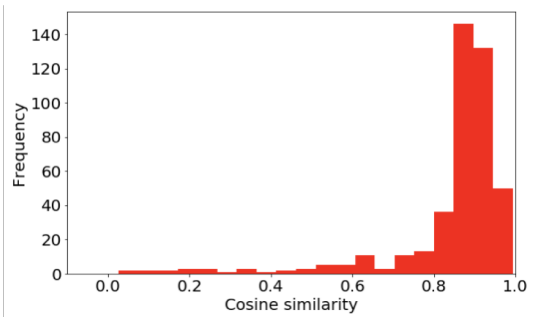

Figure 1: The frequency of cosine similarity ranges between adversaries generated by FGSM and Reluplex on the Vanilla Network.

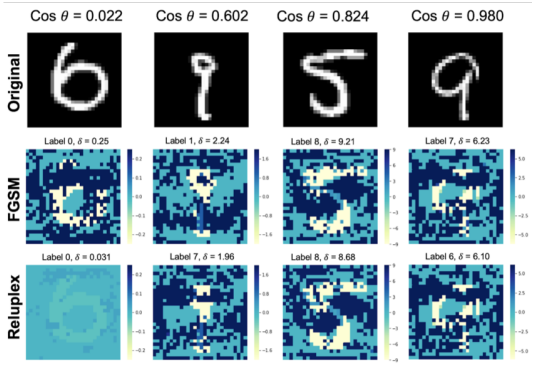

Figure 2: Heatmap of perturbation in pixels to input images by FGSM and Reluplex on the Vanilla Network.

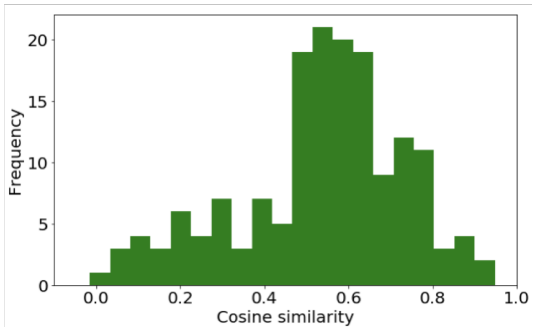

Figure 3: The frequency of cosine similarity ranges between adversaries generated by FGSM and Reluplex on the Adversarilly Trained Network.

In the Vanilla network (Fig. 5), we see that the distance of closest adversary is only marginally smaller than the one found by FGSM, which supports that in our MLP architecture, the first-order approximation of an adversary is a good approximation of the closest adversary.

Figure 6 displays the same phenomenon in the adversarially trained network. That is, the distance of the closest

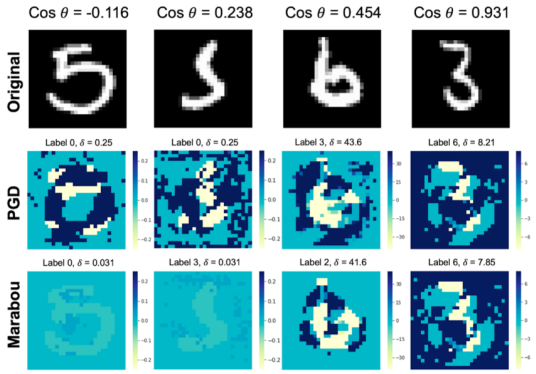

Figure 4: Heatmap of perturbation in pixels to input images by FGSM and Reluplex on the Vanilla Network.

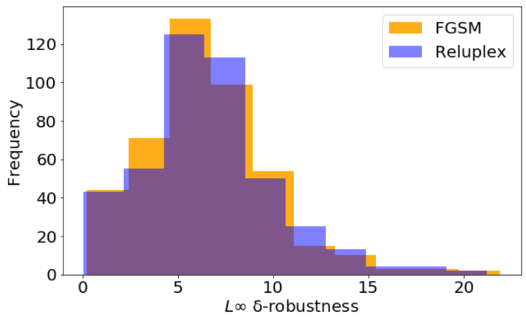

Figure 5: The frequency of examples that were robust in the FGSM and Reluplex sense for δ pixel ranges on the Vanilla Network.

adversary is similar among both the FGSM and Reluplex attack. This result is significant as it provides evidence to support that adversarial training against adversaries generated via first-order attacks generalizes to all possible attacks.

Figure 7 documents the significant increase in robustness we observed using adversarial training across both types of attacks. In particular, we can see that the average δ-robustness for the adversarially-trained network increased drastically, from about 6 pixels for the Vanilla network to about 20 pixels for the adversarial network. Taken together, these results are positive news for the machine learning community, as they seem to suggest that training a network to be robust to first-order attacks can increase robustness to not only first-order attacks (as shown in literature) but all attacks.

5. Conclusion and Future Work

In conclusion, we were able to provide some preliminary answers to open research questions surrounding non-first-order adversarial attacks. We were able to, for the first time, provide answers to some of these questions through access to Reluplex, a complete and sound linear constraint solver

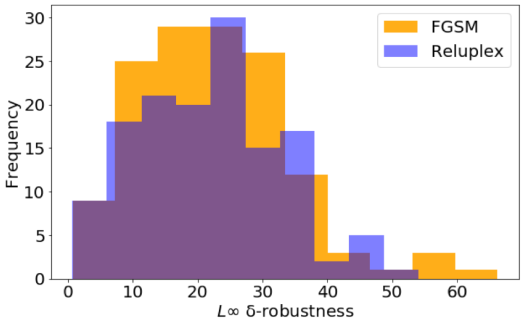

Figure 6: The frequency of examples that were robust in the FGSM and Reluplex sense for δ pixel ranges on the Adversarially Trained Network.

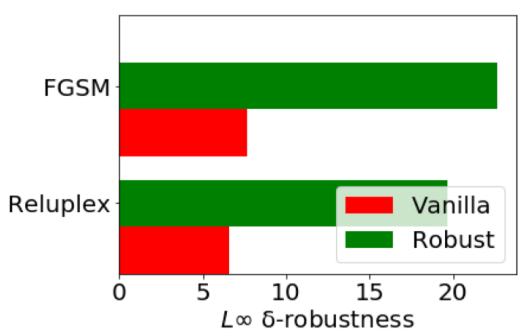

Figure 7: Average minimal δ value for Vanilla and Robust network in the FGSM and Repluplex sense.

that could be be used to find the guaranteed closest adversary around a point.

Our main results were two-fold. First, first-order attacks provide good approximations of the closest adversary in both adversarially untrained and adversarially trained networks, suggesting that benchmarking against first-order attacks is a good measure of overall robustness. Furthermore, adversarial defense strategies targeted towards first-order attacks do a very good job of generalizing to non-first order attacks.

A clear direction of future work is improving the Reluplex solver to support larger architectures so we can expand the current study to larger and deep networks. It'd be interesting to see if the more complicated loss surfaces and greater variation in linear modes present in deeper models could lead to more pathological non-first-order attacks. This direction would also provide us with the means to better test Athalye *et al.*'s hypothesis of obfuscated gradients on similar networks to those studied in the paper.

6. Contributions

Adam Pahlavan: Trained initial Tensorflow model on MNIST. Generated baseline adversarial examples.
Daniel Lee: Worked with Reluplex/Marabou to encode constraints; tested δ robustness; Generated figures
Justin Rose: Helped with installation and debugging of TensorFlow model; Wrote TensorFlow code for adversarially trained model and code to find closest FGSM adversaries.
Also, thanks to Clark Barrett (AI Safety) and Shantanu Thakoor for providing access to the Reluplex solver.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples (2014). *arXiv preprint arXiv:1412.6572*.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[4] H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[5] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[6] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802*, 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 1 ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 2012.

[8] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[11] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[13] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.