Adversarial Attacks

Rassin, Royi
311334734

July 7, 2021

1 Link to my work

https://colab.research.google.com/drive/1LyLVopLiKEvfT1VzQ3d1bsiKsh9EG8qN?usp=sharing

2 The assignment

In this work I attack a Resnet18 using a Whitebox and a Blackbox approach, and then defend it, while training on the SVHN dataset. I attack with Projected Gradient Descent (PGD), and conduct some experiments to explore the optimal hyper-parameters for attack and defense.

3 Training on the SVHN dataset

Using a pretrained ResNet18 means the training process was relatively simple, and all I had to do is attach a new head; a linear layer with an output-dim of 10, for the SVHN dataset. In terms of hyper-parameters for the ResNet18, I used an Adam Optimizer with default parameters, CrossEntropy with a 'sum' reduction, and a batch-size of 64.

4 WhiteBox

4.1 No Defense

In this setting, I simply train the network to solve the SVHN dataset for five epochs, and then apply the attack on the test-set. The PGD's are: $\epsilon = 0.03$, the step-size was set to 0.008, and the number of steps to four. As expected, the accuracy on the perturbed dataset was abysmal, while the natural version was high (you can see the accuracies of all models on table-1, below).

4.2 With Defense

Here, for every sample from the 'natural' dataset, I also train on its perturbed version. The well-known trade-off of defending our networks from attacks is between accuracy and robustness, and as such, I improved the accuracy on the perturbed dataset to 54%, however, the accuracy on the natural dataset dropped to 91.4%. Further training the model on the perturbed dataset would probably lead to higher accuracy on it and lower on the natural version, but the increase of accuracy every epoch was not significant (from 51 to 52...).

5 Blackbox

5.1 Baseline to Defended

I fed the network that was trained on perturbed data the adversarial samples that were generated from the baseline model, and it fared much better than the baseline, achieving 83.9 accuracy.

5.2 Defended to Baseline

This time, the accuracy dropped significantly, to 55.

5.3 Tinkering with ϵ

After changing the number of steps in the PGD attack to two, I began testing out different epsilons. A greater epsilon will result in a bolder permutation of the image, and a lower chance for the attack to succeed, while a lesser one, will not change the data by much, and as a result, the change is less detectable to the human-eye. For the great epsilon (0.5), the baseline's accuracy dropped to 2.8% while the defended to 31.3%. Graphs and images of the tinkered images can be seen in the appendix. As previously mentioned, the greater the epsilon, the more apparent the difference between the original image to the tinkered one. In the appendix you can see the graphs of epsilon over accuracy per model, and some of the tinkered images (there was no apparent difference among the images from the baseline or the defended one). Regarding the question of which epsilons resulted in a successful attack and which ones did not, I find that the lower the epsilon, the more successful the attack. The reason for that is that the image still resembles the original, yet the accuracy drops significantly (to 25 for the baseline, and 70 for the defended). However, the most perturbed images are not as effective, since they are quite different from the original. Nevertheless, the images I presented here all still contain the digit '3', and I would expect a network to classify it as such, and if it does not (as I can clearly see from increasing the epsilon), the attack indeed works. So all in all, as long as the image still contains the same content (there is a '3' in the image), and the network was fooled to think otherwise, then it worked.

Attack Setting	Defense	Accuracy
None	None	92.9
None	Y	91.4
Whitebox	None	7.3
Whitebox	Y	54
Blackbox	Y	83.9
Blackbox	None	55

6 Appendix

Please note, the '3' image is very clear in comparison to the rest of the data-set, so it is possible that other images no longer resemble to their original version when permuted with the same epsilon.

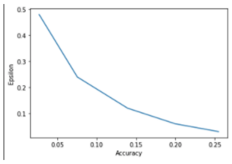

Figure 1: Baseline Epsilons

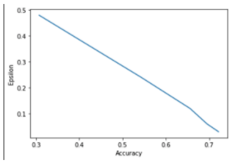

Figure 2: Defended Epsilons

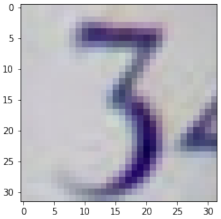

Figure 3: Original Image

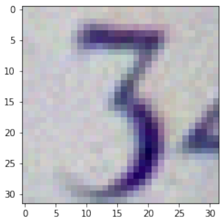

Figure 4: Least tinkered Image

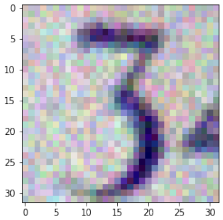

Figure 5: Tinkered Image

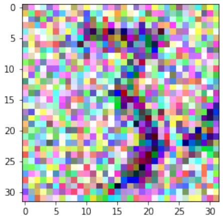

Figure 6: Most tinkered Image

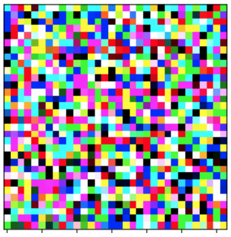

Figure 7: Extremely aggressive attack on '3', epsilon=3, just to show how pure noise looks like