Adversarial Machine Learning: The Attacks and the Defenses

Nimisha Limaye, Monica Reddy Tirupari, Sundeep Rangan

Electrical and Computer Engineering Department, New York University

This paper deals with creating a learning model to accommodate the adversarial examples. We have divided this topic into two teams, viz. Blue Team which is a team of defenders who create learning models to accommodate the adversarial examples by improving their accuracy and confidence on testing and Red Team which is a team of attackers who use adversarial examples to fool the model trained by the blue team. We observe that after training the model for adversarial examples, the red team fails to fool the model. We are using MNIST database (*1*) for this project.

Introduction

State-of-the-art deep neural networks have achieved impressive results on many image classification tasks. However, these same architectures have been shown unstable to small and well sought perturbations of the images. Adversarial examples expose fundamental blind spots in the training algorithms, as in many cases models with different architectures trained on different subsets *misclassify* the same adversarial example. Speculations for this cause is due to extreme non-linearity of deep neural networks and insufficient regularization of the purely supervised algorithm. In this paper we give an example of a fast method of generating adversarial

examples and further show that adversarial training can provide additional regularization benefit. This paper is divided in the following sections. Section 1 talks about the MNIST database we are using for the purpose of this paper. Section 2 describes the tasks and working of red team and blue team. Section 3 shows relevant results and then the paper concludes.

1 MNIST database

The MNIST database of handwritten digits (*1*) contains handwritten samples of digits 0 to 9 and the corresponding labels. This database consists of 60000 samples for training and 10000 samples from testing purpose. It is a subset of a larger database called NIST. The digits in MNIST have been size-normalized and centered in a fixed-size image. Since our aim is to learn about adversarial examples, we will be using MNIST database and not NIST.

2 Red Team and Blue Team

2.1 Blue Team

This is defenders team. We generate learning models to accommodate even the adversarial examples. We keep test accuracy and confidence high for adversarial images. In this project we built a neural network using the Theano library (*2*) in Python and trained it on the MNIST database. We have one input layer, one hidden layer and one output layer. For the MNIST database, the input vector has 784 (= 28^2) features representing the intensities of individual pixels and the final output is the predicted digit (0-9). We are using ReLU activation function as its gradients do not vanish or explode like the gradients of sigmoid function. ReLU function is given as:

$$f(x) = max(0, x) \tag{1}$$

Table 1: Results when no regularization was applied.

Test Accuracy	0.975
Test Confidence	0.981
Adversarial Accuracy for ϵ = 0.10	0.3375
Adversarial Confidence for ϵ = 0.10	0.768
Adversarial Accuracy for ϵ = 0.25	0.0048
Adversarial Confidence for ϵ = 0.25	0.767

ReLU function is followed by Softmax and Dropout.

The weights are iteratively updated using mini-batch SGD, with batch size of 128. We used RMSProp for adaptive learning rate.

$$MeanSquare(w,t) = 0.9 * MeanSquare(w,t1) + 0.1\frac{\delta cost}{\delta w}t^2$$

$$\Delta w(t) = \frac{LearningRate * \frac{\delta cost}{\delta w}(t)}{\sqrt{MeanSquare(w,t)} + TinySmoothingValue}$$

We obtained lower accuracy and confidence on adversarial images when this model was used. The values are given in table 1.

In order to improve accuracy and confidence on the adversarial images, we performed L1 and L2 regularization on the adversarial inputs. This regularization penalizes the large values of weights. We perform this by adding the penalty terms to the cross entropy cost:

$$Regularized_{CE} = CE + \lambda_1 \sum(|w|) + \lambda_2 \sum(w^2) \tag{2}$$

Table 2 shows test accuracy and confidence and adversarial accuracy and confidence for 9 different values of λ_1 and λ_2.

For blue team we observe that out of 9 different values for λ_1 and λ_2, we get better adversarial accuracy and confidence for λ_1 = λ_2 = 0.0001 and λ_1 = 0.0001 λ_2 = 0.0 as shown in table 2. The adversarial examples were generated using ϵ = 0.10 and 0.25. We opt for λ_1 = λ_2 = 0.0001 since it has higher adversarial accuracy for ϵ = 0.10.

Table 2: Results obtained with regularization

λ_1	λ_2	Test Accuracy	Test Confidence	Adversarial Accuracy for ϵ = 0.10	Adversarial Confidence for ϵ = 0.10	Adversarial Accuracy for ϵ = 0.25	Adversarial Confidence for ϵ = 0.25
0.0	0.0001	0.9805	0.9831	0.443	0.7584	0.0091	0.6789
0.0	0.00001	0.9828	0.9877	0.4534	0.7572	0.0145	0.6592
0.0001	0.0	0.9689	0.9600	0.6099	0.7963	0.0534	0.7425
0.0001	0.0001	0.9695	0.9611	0.6229	0.7987	0.0499	0.7269
0.0001	0.00001	0.9667	0.9609	0.57	0.7942	0.0552	0.7469
0.00001	0.0	0.9727	0.9836	0.5609	0.8146	0.0438	0.7183
0.00001	0.0001	0.9727	0.9728	0.4852	0.7799	0.0196	0.7364
0.00001	0.00001	0.9816	0.9820	0.5562	0.7947	0.0262	0.6915

2.2 Red Team

This is the attackers team. We generate adversarial examples such that when they are embedded in an original image, they cause the model to incorrectly classify them. We implemented an algorithm to generate the fast gradient sign adversarial examples as described in (*3*).

We construct these adversarial examples as follows:

Let θ denote the parameters of the model.

Let X be the input vector and y be the corresponding label.

The cross-entropy cost function for a sample x_i will be CE(θ, x_i, y).

After computing the gradients of the cost function, we use its sign and not the magnitude[1]. The magnitude adds perturbation in the 'confusing' direction. Higher magnitude value (ϵ = 0.25) will perturb the image more than lower magnitude values (ϵ = 0.10).

An adversarial example by this method is then generated as:

$$X_{ADV} = X + \epsilon sign(\Delta CE(\theta, X, y)) \tag{3}$$

[1]The magnitude will be governed by ϵ.

Table 3: Results for ϵ = 0.05 and 0.075

	with regularization	without regularization
Adversarial Accuracy (ϵ = 0.05)	0.8062	0.785
Adversarial Confidence (ϵ = 0.05)	0.8579	0.8842
Adversarial Accuracy (ϵ = 0.075)	0.6711	0.5723
Adversarial Confidence (ϵ = 0.075)	0.8052	0.8054

which is similar to equation in Goodfellow et al. (*3*):

$$\eta = \epsilon sign(\Delta J(\theta, x, y)) \tag{4}$$

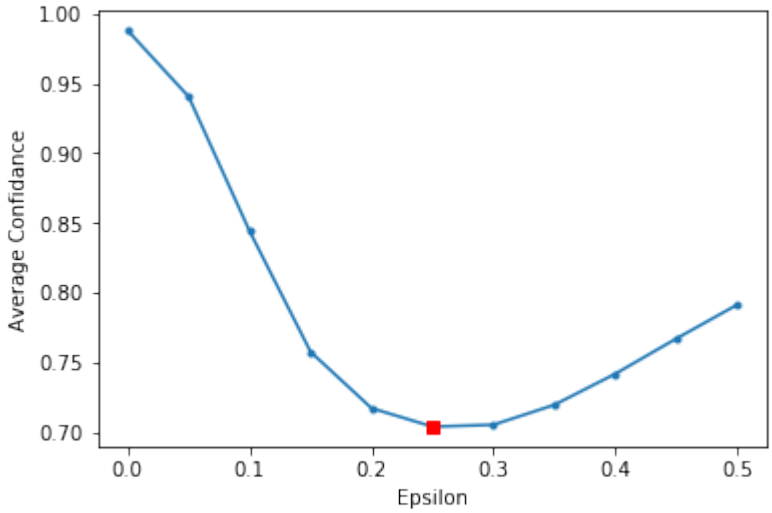
where η is the adversarial component which will be added to the original sample X to generate the adversarial example X_{ADV}

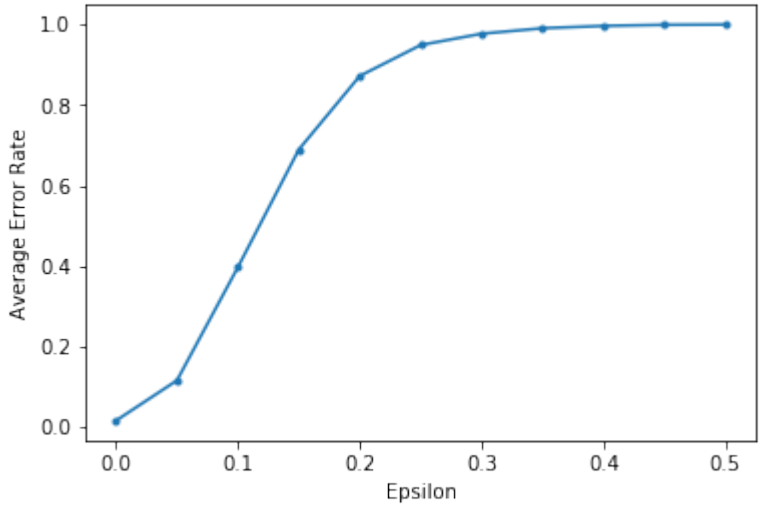
3 Results

When attacker tests his/her adversarial examples with different ϵ values, against the model trained using the regularization parameters, we obtain results as shown in fig. 1. From fig. 1, we observed that it is visually possible to detect adversarial images with ϵ>0.10. Now the attacker has to opt for ϵ<0.10 to attack the system. Results in table 3 show that the accuracy and confidence obtained when ϵ = 0.05 and 0.075, is higher than when the model was trained without regularization.

Conclusion

In this paper, we have successfully created a learning model which uses optimal values of l1 and l2 regularization coefficients to improve the accuracy and confidence of adversarial examples. These values of l1 and l2 coefficients were obtained when adversarial examples with $\epsilon = 0.10$ and $\epsilon = 0.25$ were considered during training.



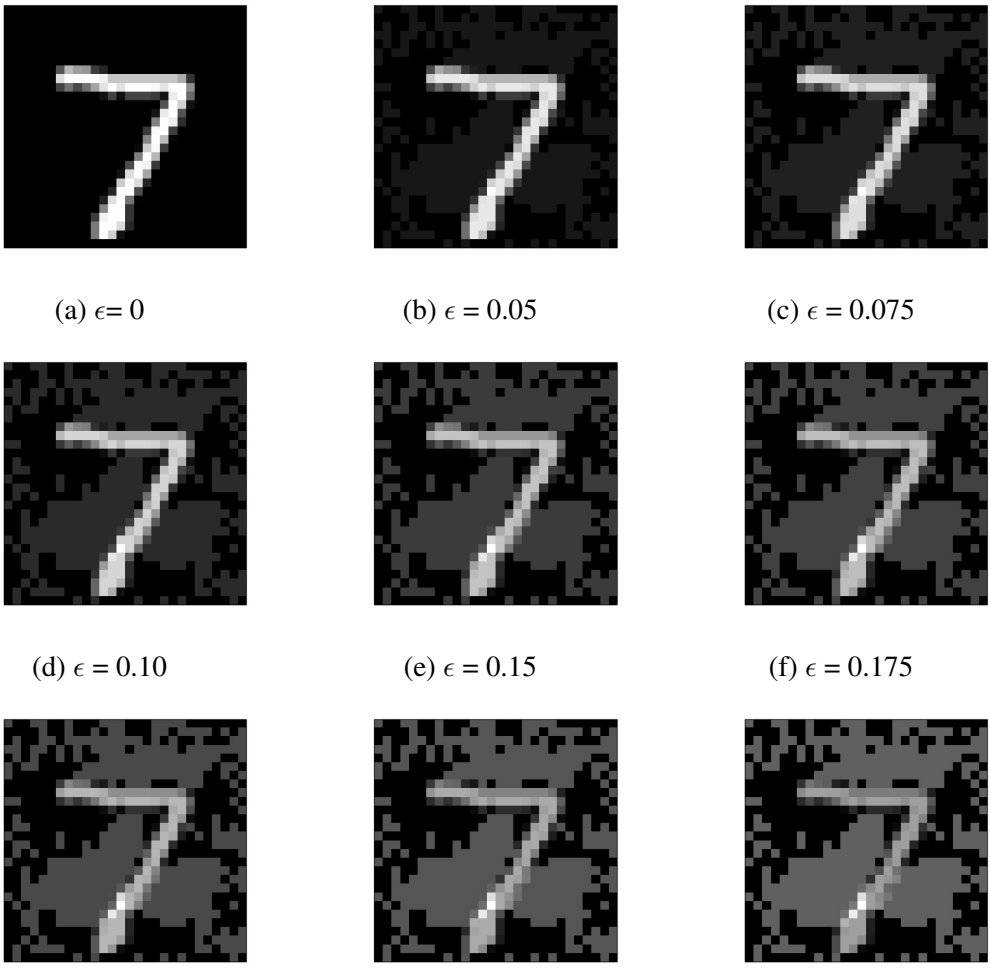


References and Notes

1. The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/.

2. The Theano tutorial, https://github.com/Newmu/Theano-Tutorials/.

3. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy, ”Explaining and harnessing adversarial examples.”, https://arxiv.org/abs/1412.6572v3 .



(a) ϵ= 0 (b) ϵ = 0.05 (c) ϵ = 0.075

(d) ϵ = 0.10 (e) ϵ = 0.15 (f) ϵ = 0.175

(g) ϵ = 0.20 (h) ϵ = 0.25 (i) ϵ = 0.30

Figure 1: Adversarial images for different perturbations on digit 7.