CSCE 478 Project : Defense Against Adversarial Attacks Using Ensemble Methods and Adversarial Training

Cale Harms and Parvez Rashid

July 3rd, 2018

1. Introduction

Modern methods used in deep learning have enabled it to be a truly transformative technology. Neural networks have been effectively used for handwritten character recognition, image classification, image compression, stock market prediction, along with many more applications. It is now known Neural networks are vulnerable to being tricked by adversarial examples. As neural networks are being integrated into more products, it is particularly important that we develop methods to protect against adversarial examples. For our project we looked at ensemble methods, using bagging and noise, and training on adversarial examples for networks that are trained on the MNIST and Tiny ImageNet data sets.

2. Data Set

We trained and tested on the MNIST and Tiny ImageNet data sets. The MNIST dataset is of 28 by 28 pixel grayscale images of handwritten digits 0 through 9. The Tiny ImageNet dataset is smaller portion of the ImageNet dataset that is comprised of 299 by 299 pixel rgb images of 1001 different classes. Both

3. Methods

For each dataset we used a deep neural network that used convolutional layers and dense layers. The MNIST model had two two-dimensional convolution layers with 32 and 64 features and kernel sizes of 3x3 for both. Max pooling of 2 by 2 is applied to the convolutional layers, and dropout with a value of 0.25 is applied before flattening the layers. The flattened layer is then connected to a dense layer of size 128 and then to a dense layer of the number of classes, 10, while using dropout in between the dense layer and final layer with a value of 0.50.

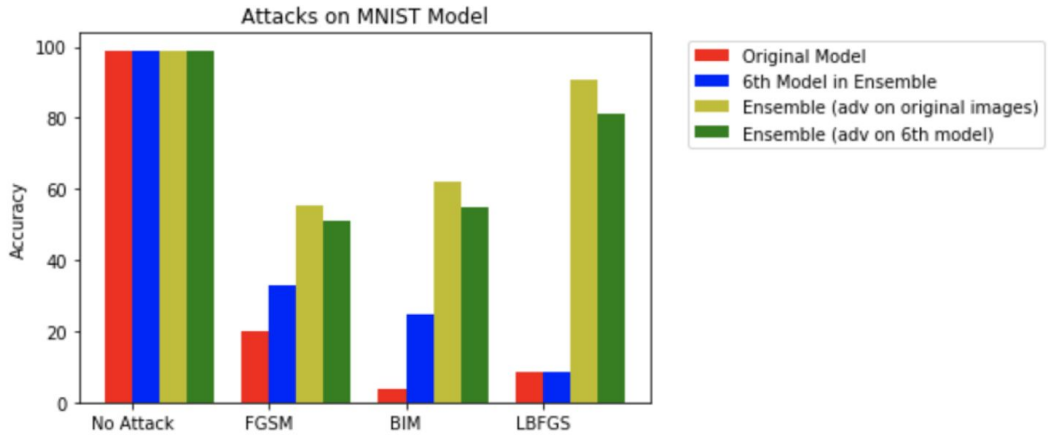
For the Tiny ImageNet data set the model we used had 3 Convolutional layers with 50, 100, and 200 features and with 5 by 5, 5 by 5, and 3 by 3 kernels respectively. Each Convolutional Used max pooling between each layer. After the Conventional stack we flatten the layer and use dropout between the last convolutional and flattened layer. We then use two dense layers with 400 and 200 nodes and use dropout between each layer. We then have a final dense softmax layer with 1001 nodes for the 1001 classes. Each dropout layer in this architecture uses a value of 0.50.

For the Adversarial attacks we tested against three different methods to generate adversarial examples. The Fast Gradient Sign Method, Basic Iterative Method, and Limited Memory BFGS method. The Fast Gradient Sign and Basic Iterative Methods are Non-targeted,

meaning the methods are simply generating examples that are wrongly classified, while the Limited Memory BFGS method is targeted, meaning that it is trying to generate examples that are classified as a particular class. These methods include a small group of, but diverse methods.

For the Defense Methods we used bagging + noise using an ensemble of ten networks, and then trained new models on the original training as well as new adversarial examples. We then took a newly trained model on the FGSM examples and then tested the model on the other attack methods.

4. Results



Adversarial Retraining	FGSM	BIM
Normal data	98.36	98.86
Adversarial data	96.68	93.16

What we found previously was that when retrained on adversarial data the models had improved accuracy not only on the attack, but even improved accuracy on non-adversarial data. But when saved the model and then tested the model trained on FGSM adversarial data it only managed to attain 1.37 accuracy, and actually performed better when tested on BIM with 42.37 accuracy.

5. Issues and Insights

We experienced a lot of technical difficulties in trying to get our programs to work on the HCC. For instance we couldn't get Limited Memory BFGS method to work. We also could not load the Tiny ImageNet dataset, likely due to different versions of the Pillow library.

The baggin and noise did improve the accuracy quite a bit against the adversarial attacks, but not as well as the adversarial training did at some points. Although the ensemble method products more generally against different attacks than training on adversarial examples.

Given the conflicting the results we received it is clear we need to do more research, and perhaps should introduce some form of cross-validation when testing the adversarily trained models.