Adversarial Machine Learning: Attack and Defence

Ashish Naik[1] and Rashika Koul[2]

[1]anaik6@gmu.edu, [2]rkoul2@gmu.edu

[1,2]Computer Science Department,
George Mason University,
Fairfax, VA, USA

December 10, 2018

ABSTRACT

Neural networks have achieved a lot of success over the years. Many approaches have been devised to fit the complexities of the data and to make the process faster and learn better. But we often forget that neural networks are fundamentally just a series of vector multiplications and thus show great potential to be fooled by adversarial examples. There are many ways to fool a neural network and many ways to protect against such attacks, but the question is still an open research topic.

1 INTRODUCTION

Adversarial machine learning is a research field that aims to enable the safe adoption of machine learning techniques in adversarial settings, such as spam filtering, malware detection, and biometric recognition. Since Machine learning techniques were originally designed for stationary environments in which the training and test data are assumed to be generated from the same distribution, they most often tend to fail in presence of intelligent and adaptive adversaries [1]. In the domain of Image processing, these adversaries are inserted into images by adding a very small amount of carefully constructed noise. This noise can either just get the adversarial image misclassified as a specific class (non-targeted attack) or also make the adversarial image look like a particular 'target' class apart from getting it misclassified as a specific class (targeted attack).

These adversaries can prove to be extremely dangerous in some cases. For example, in the case of self-driving cars if a stop sign at a busy intersection is replaced with an adversarial example of it, then it would result in casualties. Another example is when adversarial images are constructed to bypass face recognition security features. The fact that these adversarial examples exist, means that the systems that make use of neural networks have a very high security risk.

In this project, the aim is to trick a feedforward neural network which is trained to classify the MNIST dataset taken from [2]. Then, as a mode of defense, either the feedforward neural network is trained with the adversarial examples or color bit squeezing is performed on the adversarial images before sending them for prediction, or both.

2 BACKGROUND

For adversarial attack in a mini-max game, the attacker aims to maximize its likelihood to trick the classifier, while the classifier aims to minimize its risk in such worst case [3]. There are various ways to attack a machine learning system. One could inject malicious data into the training set to manipulate the training procedure, or one can use the model information (like the gradient w.r.t the input) to construct the adversarial example[4].

However, the definition of adversarial has evolved in recent researches in the deep learning field. It refers to a process of producing small and imperceptible perturbations to the normal example which can fool the machine learning algorithms. The study on adversarial learning made by Szegedy et al., pointed out that in many of the learning based computer vision systems, an imperceptibly small perturbation on the input image may cause an uninterpretable failure in the classification result [5]. Despite the phenomenal performance of neural networks, there are several follow-up questions: [4]
1. Why are these perturbations imperceptible to human but have a large impact on systems with phenomenal performance?
2. How can such adversarial examples be constructed?
3. How do we prevent the classifiers from misclassifying these adversarial examples?

3 APPROACH

We used the mini-batch stochastic gradient descent with the backpropagation learning algorithm, to train the feedforward neural network on the training set of MNIST dataset to recognize individual handwritten digits. We then created adversarial examples of the images in MNIST dataset to trick the feed-forward neural network to misclassify the input image.

Then, we attempted to prevent the attack using two ways. First, by training the neural network on adversarial examples (apart from the original training set) which are correctly labeled. Second, by reducing the color bit depth of the adversarial image also known as color bit squeezing. We analyzed the performances of both the defense mechanisms.

4 EXPERIMENTAL DESIGN

4.1 Training

The image in figure 1 is a graphical representation of our feedforward neural network in which we have 784 input units as the training images have 28 x 28 pixels. Our network instance had to have at least one hidden layer as the training data is linearly inseparable. The number of hidden layers and the number of units in the hidden layer was decided on trial and error basis. But in our case, changing the number of hidden layers and the number of units in the hidden layer was not making any considerable difference in the way the network learns so we fixed the number to one hidden layer with 30 units. The output layer has 10 units as there are 0 through 9 digits (or classes) in our MNIST dataset, which is a dataset of images of handwritten digits. It contains 50,000 training data, 10,000 validation data and

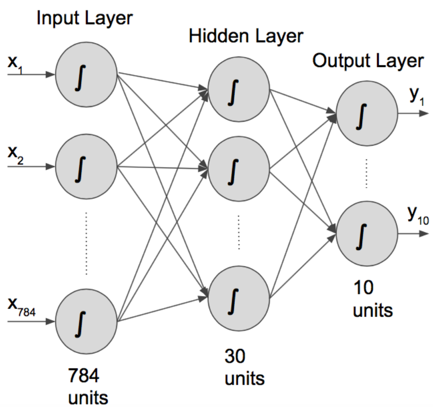


Figure 1: Graphical representation of Feedforward Neural Network

10,000 test data.

Mini-batch stochastic gradient descent was used with backpropagation to update the weights and biases. Batch size determines how many examples one looks at before making a weight update. The lower it is, the noisier the training signal is, the higher it is, the longer it takes to compute the gradient for each step. We trained our neural network for 100 epochs with a batch size of 5 and learning rate of 0.1.

4.2 Tricking

To generate adversarial examples we solve a minimization problem C as shown in equation (1), which essentially has two parts. In the first part, our goal is to find an adversarial image $\vec{x}$ such that the difference between the specified output that we want the network to predict and the output that the network predicts for the image $\vec{x}$ is minimized. In the second part, we optimize the input to look like a certain image. This means minimizing the difference between the pixel intensity of the image $\vec{x}$ and some target image which we want our adversarial example to look like. The difference is in the form of the squared Euclidean norm. λ determines which is more important, optimizing for the desired output or optimizing for an image that looks like $\vec{x}_{target}$.[6]

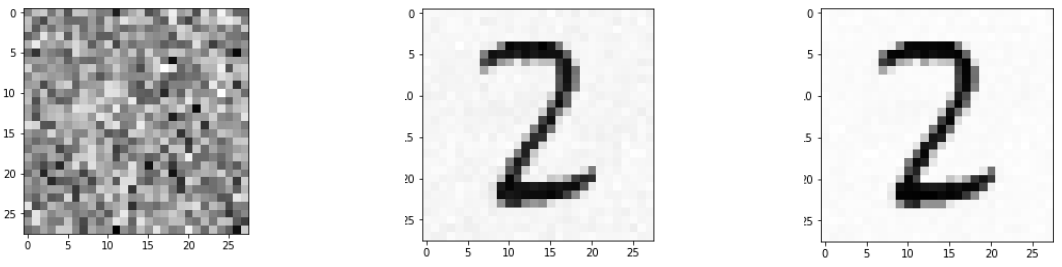
$$C = ||\vec{y}_{goal} - y_{hat}(\vec{x})||_2^2 + \lambda ||\vec{x} - \vec{x}_{target}||_2^2 \qquad (1)$$

In equation (1) [6],
$\vec{x}$: Adversarial image to be generated
$\vec{x}_{target}$: The image we want our adversarial example to look like
$\vec{y}_{goal}$: The label we want our network to predict for $\vec{x}$
$y_{hat}(\vec{x})$: Network's output for $\vec{x}$
λ: A hyperparameter that we can tune

$$argmin_{\vec{x}} C(\vec{x}) \qquad (2)$$

So our optimization problem is, minimize C with respect to $\vec{x}$ as shown in (2). To do this, we perform gradient descent on C. Start with an initially random vector $\vec{x}$ and take steps changing $\vec{x}$ gradually in the direction opposite to the gradient. To get these derivatives we perform backpropagation on this network and update the input image $\vec{x}$ instead of updating the weights and biases. [6]

When λ is 0, the attack only focuses only on getting the image misclassified instead of also making the image look like the target. As we increase λ we start seeing the target image more and more clearly in our adversarial image. As λ approaches 1, the focus on getting the image misclassified tends to 0 and the image is classified correctly. This is depicted in figure 2.



(a) Non-targeted attack (b) Targeted attack (c) No attack

Figure 2: Three types of adversarial images generated by changing λ. In (a), the value of λ is 0.0; predicted class is 3. In (b), the value of λ is 0.05; predicted class is 3. In (c), the value of λ is 1; predicted class is 2.

4.3 Defending

4.3.1 Adversarial Training

Adversarial training introduces generated adversarial examples and corresponding ground truth labels in the training data. Ideally, the model learns the weights and biases from the adversarial samples (apart from the non-adversarial ones) and performs robustly on future adversarial examples generated using the same cost function discussed previously. [7]

We generated 50,000 training adversarial examples and 10,000 test adversarial examples, with λ fixed by trial and error to 0.05, and it contains all possible combination of the target image and goal label. We trained our feedforward neural network on 50,000 adversarial examples apart from 50,000 non-adversarial examples.

4.3.2 Color bit Squeezing

The most common way to represent images is using color bit depths. Very often, irrelevant features may come into picture due to unnecessary colors. So, hypothetically we can say that reducing bit depth, as far as there is no human-observable loss in the information that may cause any misclassifications, can reduce adversarial opportunity without harming classifier accuracy. For colored images, which have 8-bits (per RGB channel), lowering the bit depth to <8 bits but >=4 bits (per RGB channel) should do the trick. Whereas, for grayscale

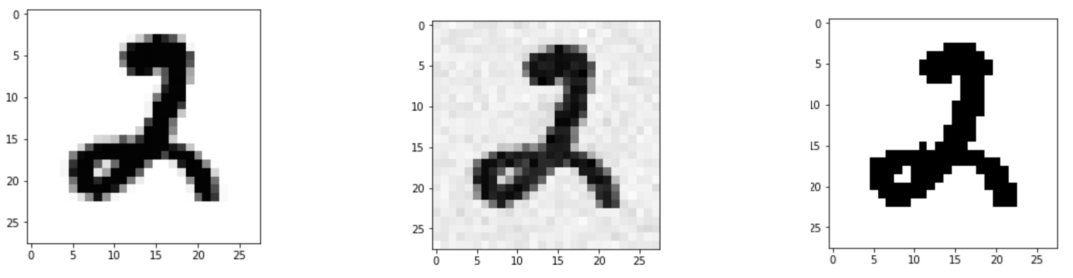

(a) Original image (b) Adversarial image (c) Adversarial image after binary thresholding

Figure 3: Color bit Squeezing. In (a), the predicted class is 2. In (b), the predicted class is 3. In (c), the predicted class is 2.

images such as MNIST dataset, reducing the bit depth from 8-bit to 1-bit monochrome images should do the trick. This is called binary thresholding. [7]

In figure 3, the rightmost image, which is generated by applying binary thresholding on the adversarial image (center) with 0.5 as the cutoff, appears nearly identical to the original image on the far left. After performing binary thresholding on the adversarial image, pixels having intensities between 0.0 and 0.5 become 0.0 and pixels having intensities between 0.5 and 1.0 become 1.0, thereby reducing the feature space to 1 bit.

5 EXPERIMENTAL RESULTS

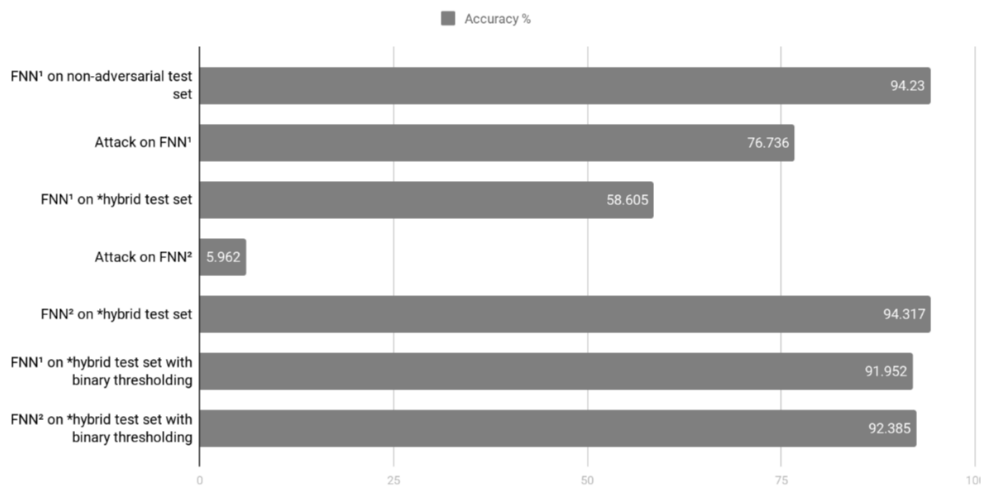


Figure 4: Accuracy graph. Note that *hybrid set = non-adversarial set + adversarial set, FNN^1 is the network trained on original training set of MNIST and FNN^2 is the network trained on hybrid training set.

We found the following accuracies:
1. The accuracy of FNN^1 on the non-adversarial test set is 94.23%

2. The accuracy of attack on FNN^1 is 76.736%
3. The accuracy of FNN^1 on the hybrid test set is 58.605%
4. The accuracy of attack on FNN^2 is 5.962%
5. The accuracy of FNN^2 on the hybrid test set is 94.317%
6. The accuracy of FNN^1 on the hybrid test set with binary thresholding is 91.952%
7. The accuracy of FNN^2 on the hybrid test set with binary thresholding is 92.385%

6 ANALYSIS OF THE RESULTS

Adversarial training is costly and takes double the time to train the feedforward neural network. It's effectiveness also depends on having a technique for efficiently generating adversarial examples similar to the ones used in the attack. This means that if λ in the cost function discussed previously changes, the attack changes. It is essential to include adversarial examples from all known attacks in adversarial training since this defensive training is non-adaptive. This means that our learner should be aware of the attack to be able to defend against it. This suggests that this defense mechanism still has a scope to increase its accuracy by training the network on more and more adversarial examples.

The effectiveness of color bit squeezing is surprising considering the fact that it is so simple and inexpensive compared to adversarial training. However, it involves an important call of selecting a number to lower the color bit depth to, which may require domain knowledge. If this call is correctly made the obtained accuracy will be desirable, otherwise not.

Another interesting thing is that when we use both the defense mechanisms together rather than just training the network, the accuracy drops from 94.317% to 92.385%. This suggests that combining two or more defense mechanism may not always result in a better neural network. We must choose our defense mechanisms based on the type of attack that we are expecting on our classification system.

All in all, training the neural network on all possible attacks seems like a costly but fool-proof defense mechanism.

7 CONCLUSION AND FUTURE WORK

Our understanding of neural networks is deficient. Frequently, neural networks are described as systems that mimic human brains. People think that the neural network classifies the image as a particular class because of the way it looks. But a neural network does not classify on the basis of looks of an image. They are fundamentally just a series of vector multiplications with some added non-linearities and come up with the class decisions on the basis of these mathematical results. And thus, one can always come up with examples that can make the neural networks produce unwanted results, proving that these mathematical models are incredibly flimsy.

Many ways can be developed to attack, and many ways can be developed to prevent these attacks but the question is still an open research topic.

References

[1] Wikipedia contributors. (2018, November 23). Adversarial machine learning. In Wikipedia, The Free Encyclopedia. Retrieved 03:06, December 10, 2018, from `https://en.wikipedia.org/w/index.php?title=Adversarial_machine_learning&oldid=870264352`

[2] `https://drive.google.com/file/d/1nE7Aoe7EfwyIWy7TG7WfYitgPsE8FB5p/view?usp=sharing`

[3] Michael Bruckner and Tobias Scheffer. Nash equilibria of static prediction games. In Advances in neural information processing systems, pages 171–179, 2009.

[4] Li, Guofu, et al. "Security Matters: A Survey on Adversarial Machine Learning." arXiv preprint arXiv:1810.07339 (2018).

[5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[6] Daniel Geng and Rishi Veerapaneni. (10 Jan 2018). Tricking Neural Networks from `https://ml.berkeley.edu/blog/2018/01/10/adversarial-examples/`

[7] Xu, Weilin, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks." arXiv preprint arXiv:1704.01155 (2017).