COSE474-2019F: Final Project
Improving Gradient-Based Adversarial Attack Algorithms

Donghan Kim (2015410145) Youngil Yoon (2015410080) Jinseok Kim (2015410085) Jihun Kim (2015410122)
Dongik Shin (2015410082)

Abstract

This paper aims to further develop existing white-box adversarial attack algorithms on a pre-trained neural network produced by MadryLabs at MIT. With a focus on improving current gradient based attacks, this paper uses a ResNet image classifier that was already pre-trained for adversarial attacks (provided by MadryLabs). The data set used for this research include the MNIST and CIFAR-10 dataset. The main objective of this paper is to produce minimal perturbations that leads to a high miss-classification percentage using only the gradients of a given model on the already well trained MadryLabs ResNet model.

1. Introduction

There are currently many different algorithms that are able to fool image classifiers. Most of these models use a white-box algorithm that utilizes the gradients and parameters of a given model. However, it is clear that these algorithms can be improved. This paper predominately focuses on developing a better version of the existing fast gradient sign attack (a popular method for creating white-box adversarial examples). The dataset used for this research include the MNIST and CIFAR-10 dataset. While the objective of this paper is to produce an algorithm that can successfully miss-classify images, this paper also attempts to produce a competitive miss-classification rate on a image classification network that has been pre-trained for adversarial attacks. This pre-trained model is provided by MadryLabs at MIT, and was part of a public challenge provided on Github in 2018[1]. The best attempt so far on the MNIST data set is able to decrease the accuracy of the classifier to 88.32%. Our research was able to modify a relatively naive algorithm (FGSM) and produce an accuracy of 92.57% on the MNIST dataset (12th on the current leaderboard). Furthermore, an impressive score of 48.46% was recorded on the CIFAR-10 dataset (7th on the current leaderboard).

1.1. Motivation

With the growing use of computer vision in commercial applications (self driving cars for example), there is a growing need to ensure that these networks are robust and safe to use in the real world. Although image classification, object detection and image segmentation algorithms are good at identifying and classifying objects, it is easily deceived with small amounts of perturbation to the input. This can be very dangerous if ”attackers” pertubate images that computer vision sensors receive. As a result, in order to advance and ensure the use of these algorithms in the real world, this paper aims to address the different methods found in fooling neural network based image classifiers.

1.2. Related Works

White-box attacks are a popular field within safe-AI. The FGSM (fast gradient sign method) is a novel and one of the first methods created to produce white-box adversarial examples[2]. Therefore, the majority of this research is emphasized in improving the already famous FGSM algorithm. Applying regularization to FGSM is an examples whereby the improved FGSM led to better performance on miss-classifying neural networks[3]. Furthermore, papers regarding the defense against white-box attacks were also considered in an effort to reverse-engineer the given algorithms[4]. Although FGSM is an effective attack, it is clear that it has limitations. As a result, attacks using a different algorithm were also considered[5]. Although the latest method (SOTA) of producing white-box attacks is the DeepFool[6] and PGD (projected gradient descent) algorithm[7], these methods were not considered as the main focus was developing a more robust attack algorithm for FGSM.

1.3. Challenges

The challenge with adversarial attacks is making sure the perturbed images remain unnoticeable to the human eye. This is difficult to execute because the best image classification networks are extremely well trained and robust. Therefore, random changes to the input image will not cause any miss-classification. In addition, the neural network this paper used to evaluate the performance of our algorithms is a network that has already been trained to detect basic adversarial attacks. This network (provided by MadryLabs) is therefore extremely robust and difficult to fool. To give an examples, applying the basic FGSM attack on MadryLabs network produced 56.04% in accuracy for the untrained model, whilst the untrained model produced an accuracy of 12.87% (CIFAR-10 dataset, epsilon set to 8.0).

2. Datasets

As most of the work done in the field of safe-AI uses standardized images, this research uses the MNIST and CIFAR-10 dataset to evaluate the performance of our algorithms. The MNIST and CIFAR-10 datasets were choosen to see how well our algorithms perform against different kinds of images. The MNIST dataset contains grey-scale images of size 28x28 of hand written numbers (between 0 to 9). There are 60,000 training examples and 10,000 test examples provided by MNIST[8]. In contrast the CIFAR-10 dataset is a colored (RGB) 32x32 image dataset containing 10 different classes of images (airplane, dog, frog, etc). Similarly to the MNIST dataset there are 50,000 training examples an 10,000 test examples in the CIFAR-10 dataset[9].

3. Attack Algorithms

All attacks described in this section are *untargeted white-box attacks*. In total five different attacks were created throughout the duration of this project. Due to the importance of FGSM, the FGSM algorithm will first be explained.

3.1. FGSM

Having access to the model being attacked means that the attacker has access to the gradients of the network. This is crucial because knowing the gradients of the model allows the attacker to change the optimizer to increase loss, rather than decreasing it. FGSM relies on the sign of the gradient, to further push the decision boundary the model generates in favor of the attacker. Mathematically this is expressed using the following equation:

$$x_{adv} = x + \varepsilon \cdot sign\left(\nabla_x J\left(\theta, x, y\right)\right)$$

Here $J(\theta, x, y)$ represents our cost function. And instead of optimizing the loss function with respect to the weights, we are optimizing it against the input x. The sign of the gradients is multiplied by an epsilon value (the amount we want to pertubate our image) and added to the original input to create our adversarial examples.

3.2. FLGM

The FLGM (fast linear gradient method) is a slight variation to the original FGSM. The word linear is used to emphasize the fact that a linear operation is applied on the standardized gradient retrieved from the model in attack.

$$x_{adv} = x + clip(slope \cdot standardize(\nabla_x J\left(\theta, x, y\right)), \varepsilon)$$

The gradient with respect to x is found from the loss function (same as FGSM). However, instead of using the sign of the gradient, this method multiplies the gradient with a constant value called $slope$. In order to make sure the perturbation is set below a set amount (ε), we use a "clip" method that essentially returns the larger value between the two. In other words, if the linearly multiplied value is greater than the set epsilon value, then the set epsilon value is returned instead.

3.3. FSGM

FSGM (fast squared gradient method) is an improvement of the FLGM attack. The only difference between FLGM and FSGM is the squaring of the gradient retrieved from the loss function $J(\theta, x, y)$. However, as it will be shown in the performance and evaluation section of this paper, squaring the gradient value performs much better than just linearly multiplying the gradient with a constant value. Similar to FLGM, FSGM multiplies the squared gradients with a constant epsilon value called $slope$ and is clipped against the maximum perturbation we want our adversarial example to have.

$$x_{adv} = x + clip(slope \cdot (standardize(\nabla_x J\left(\theta, x, y\right)))^2, \varepsilon)$$

3.4. FLoGM

Due to the monotonically increasing property of the logarithmic function, the log function was applied to the gradient retrieved. The value was made positive before adding a unit value of 1.

$$x_{adv} = x + clip(slope \cdot log(|standardize(\nabla_x J\left(\theta, x, y\right))| + 1), \varepsilon)$$

Again, this method is very similar to the previous two variations of FGSM. The same operation of multiplying the transformed gradient value by some constant was applied, before comparing it with our set epsilon value.

3.5. FGPM

This algorithm was inspired from the works of Su, Vargas and Sakuari[10] one pixel attack. FGPM (fast gradient pixel method) aims to target the pixels that contribute most to the decision boundary of the target model. In other words, instead of randomly removing pixels of an image, this algorithm looks for the pixels that seem to have the most impact in regards to the gradient of the cost function. The pseudo code below details this algorithm in more detail.

Algorithm 1 FPGM (Fast Gradient Pixel Method)

```
**Input:**
    input data $x$, size of pixels $s$,
    number of pixels to perturb $\varepsilon$
**Output:**
    adversarially attacked data $x_{adv}$
**Initialize:**
    $x_{adv} \leftarrow x$
    $n \leftarrow$ number of images
    $w, h \leftarrow$ width and height of image
**for** $n' \leftarrow 0$ **to** $n-1$ **do**
    $pixel\_grad \leftarrow$ empty array
    **for** $(w', h') \leftarrow (0,0)$ **to** $(w-s-1, h-s-1)$ **do**
        $sum \leftarrow 0$
        **for** $(i,j) \leftarrow (0,0)$ **to** $(s-1, s-1)$ **do**
            $sum$ += $\nabla_x J(\theta, x_{adv}, y)[n'][w'+i][h'+j]$
        **end for**
        $pixel\_grad$.append$((sum, w, h))$
    **end for**
    $pixel\_grad \leftarrow$ sorted$(pixel\_grad)$
    $cnt \leftarrow 0$
    **for** $(w', h') \leftarrow (0,0)$ **to** $(w-s-1, h-s-1)$ **do**
        **for** $(i,j) \leftarrow (0,0)$ **to** $(s-1, s-1)$ **do**
            **if** sum$(x_{adv}[n'][w'+i][h'+j]) > 0$ **then**
                $x_{adv}[n'][w'+i][h'+j] \leftarrow 0$
                $cnt \leftarrow cnt + 1$
                **if** $cnt >= \varepsilon$ **then**
                    **Break out of two loops**
                **end if**
            **end if**
        **end for**
    **end for**
**end for**
```

3.6. I-Method

Borrowing the idea of iteratively running FGSM (I-FGSM), the I-Method iteratively runs any one of the attack algorithms discussed in this paper. In other words, all methods with ”I-” indicate an iterative version of the attack algorithm. For all iterations on the MNIST dataset, the $iter$ parameter is set to 100. The $iter$ parameter for the CIFAR-10 dataset is set to 20.

4. Performance and Evaluation

Performance of the algorithms described above will be done on both the MNIST dataset and CIFAR-10 dataset. However, do note that not all algorithms listed above was used on both the MNIST and CIFAR-10 dataset. For example, FGPM was not applied on the MNIST dataset due to the fact that MNIST images are grey-scale and have limited pixels of interest to the network. Therefore, applying FGPM will prove to be useless and ineffective. The algorithms applied on the MNIST dataset are the folloiwng: FGSM, FLGM, FSGM, I-FGSM, I-FLGM and I-FSGM. The accuracy score of the figures shown below are the results of running the test examples through an *untrained* convolutional neural network consisting of two convolutional layers (each followed by max-pooling) and a fully connected layer. The best performing algorithms on the untrained network will be used on the adversarial-example trained MadryLabs network. Also note that for the MadryLabs challenge, the maximum epsilon value allowed for MNIST is capped at 0.3, and for CIFAR-10 at 8.0.

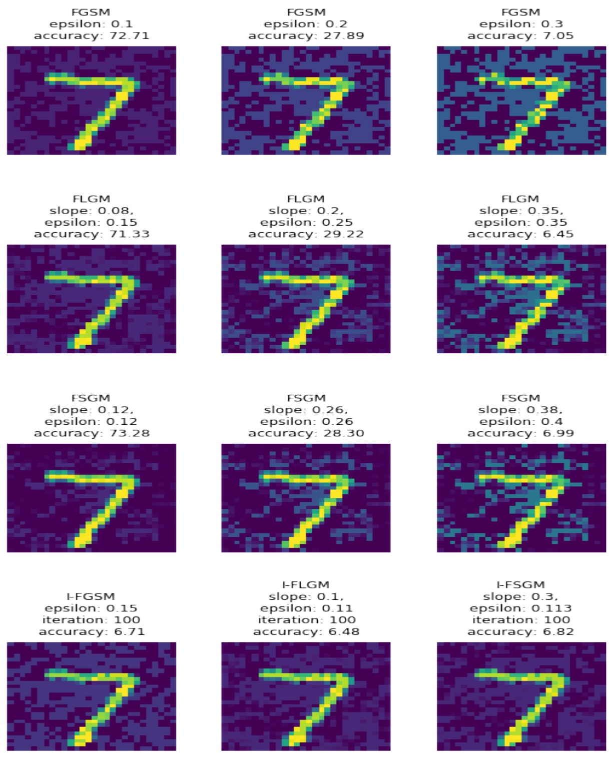

Figure 1. Pertubation results on MNIST dataset for FGSM, FLGM, FSGM, I-FGSM, I-FLGM and I-FSGM (untrained network)

As mentioned above, one of the challenges of creating adversarial examples is to make sure the perturbations are not visible to the human eye. In the case of FGSM, although it achieves an accuracy score of 7.05% when the epsilon value is set to 0.3, it is clear that the image quality has been significantly degraded. From the MNIST dataset, the best performing algorithm is the I-FSGM algorithm. Not only did it achieve an outstanding accuracy of 6.48%, the perturbations of the image is barely noticeable. The I-FSGM method achieved a 92.57% accuracy on the MadryLabs trained adversarial network. Currently this algorithm ranks 12th on the white-box challenge board for the MNIST dataset.

4.1. CIFAR-10

For the CIFAR-10 dataset, FGSM, FLGM, FSGM, FLoGM and FGPM were evaluated. The figure below highlights this for a sample image taken from the test set.

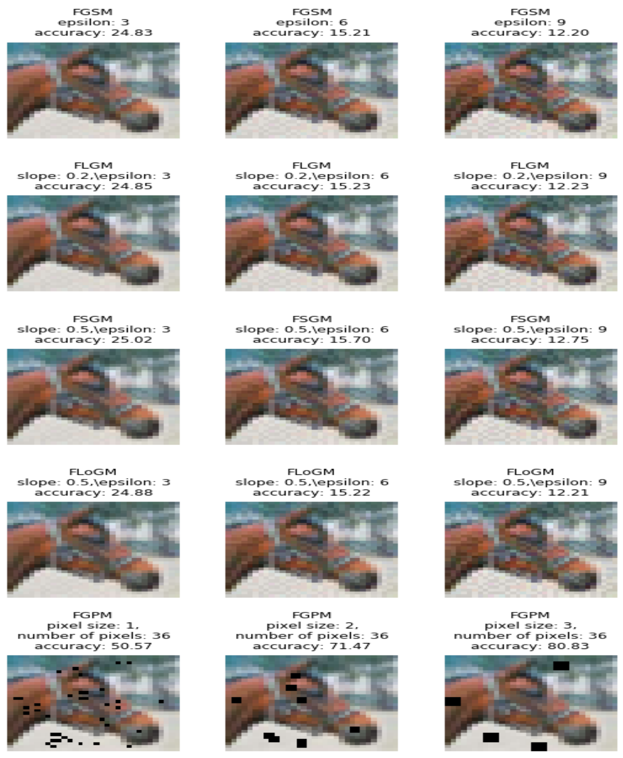

Figure 2. Pertubation results on CIFAR-10 dataset for FGSM, FLGM, FSGM, FLogM and FGPM (untrained network)

On single iterations, FGSM performed the best. Receiving a score of 15.21% when the epsilon value was set to 6.0 (bearing in mind the challenge allows epsilon values up to 8.0). Whats interesting to note is FGPM performed the worst despite its ability to target ”key” parts of the image. This proves that neural networks relies a lot more than the key areas within an image, when tasked with classification.

4.2. MadryLabs Challenge Results

One of the main objectives of this paper is to see how well our algorithms perform on MadryLabs trained neural network. As a result, taking advantage of the fact that iteratively running our algorithms produce better results (as shown on the MNIST dataset) we choose some of the better performing algorithms and ran it iteratively on both the MNIST and CIFAR-10 dataset. The figures below show our results from MadryLabs trained network.

The iterative version of FLGM scored an impressive accuracy score of 45.85% on the CIFAR-10 dataset, placing this algorithm 5th on the MadryLab CIFAR-10 challenge leader board.

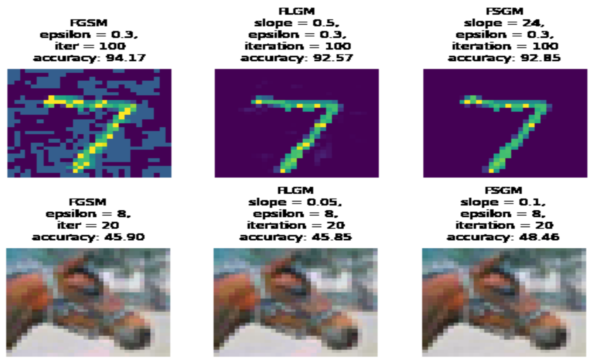

Figure 3. Iterative versions of FGSM, FLGM and FSGM on MadryLabs trained network (MNIS and CIFAR-10 dataset)

5. Conclusion and Further Works

It is clear that even simple algorithms such as the FGSM can be improved significantly. Of course, simply tuning and slightly modifying a naive algorithm will not bring about SOTA results. However, it is clear that there are improvements that can be made. The key point of this research lies in the fact that iterating these gradient based attacks produce much more meaningful results than running the algorithm once. Furthermore, as shown with FGPM algorithm, when constructing adversarial attack algorithms, there is no incentive to focus on attacking individual or a set of pixels that contains the most ”meaningful” information regarding the picture. This ineffectiveness is clearly highlighted in the CIFAR-10 dataset evaluation.

Applying these changes to existing SOTA algorithms can be an interesting as they might bring about similar results to the ones achieved in this paper. For example, iteratively running the PGD (projected gradient descent) algorithm can produce significantly better results as well. One disadvantage of running the iterative versions of these attacks is that we have to query the model in attack multiple times. However, in the real world, this might not feasible as most of these models are likely to be contained in a secure setting, preventing access to the gradients and limiting the number of queries to the model.

To really ensure the robustness of these neural networks in commercial applications, there is a need to apply these methods to more complex tasks such as object detection and image segmentation. Despite the fact safe-AI is a growing field within artificial intelligence, it is clear that there needs more work to be done.

6. Reference

[1] Madry, et al. MNIST Adversarial Examples Challenge. In $https://github.com/MadryLab/mnist_challenge$

[2] Goodfellow, et al. Explaining and Harnessing Adversarial Examples. In *ICLR, 2015*

[3] Dong, et al. Boosting Adversarial Attacks with Momentum. In *CVPR, 2018*

[4] Zhang and Liang. Defending against Whitebox Adversarial Attacks via Randomized Discretization. In *22nd International Conference on Artificial Intelligence and Statistics, 2019*

[5] Li, et al. Learning More Robust Features with Adversarial Training.

[6] Dezfooli, et al. DeepFool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, 2016*

[7] Madry, et al. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations, 2018*

[8]LeCun, et al. THE MNIST DATABASE of handwritten digits. In $http://yann.lecun.com/exdb/mnist/$

[9] Krizhevsky. The CIFAR-10 dataset. In $https://www.cs.toronto.edu/ kriz/cifar.html$

[10] Su, et al. One Pixel Attack for Fooling Deep Neural Networks. In *IEEE Transactions on Evolutionary Computation, 2019*