Whitebox Attack on Image Classification Models

Kuan-Po, Huang
Department of Computer Science
National Taiwan University
Taipei, Taiwan

Yu-Ting, Tseng
Department of Computer Science
National Taiwan University
Taipei, Taiwan

Hong-Sheng, Cheng
Department of Computer Science
National Taiwan University
Taipei, Taiwan

Che-Kuan, Sung
Department of Computer Science
National Taiwan University
Taipei, Taiwan

ABSTRACT

Deep neural networks are vulnerable to well-designed adversarial inputs. One of the well known attack is Fast Gradient Sign Method(FGSM). Many adversaries use gradient-based methods to calculate the adversarial input with little noise on benign pictures. In this report, we present some analysis with different methods of attacks and some new methods to improve both attacking and defending of classification models. Adversarial attacks include FGSM, BIM, one pixel attack. Defense includes three methods for one pixel attack and adversarial training for FGSM.

1 INTRODUCTION

For adversarial attacks, we introduce FGSM, BIM, and One pixel attack in section 3. Although FGSM and BIM are the most popular method used in adversarial attacks, there is a common problem with them. The parameter ϵ is usually a constant value applied to the perturbation to fix the amplitude of it. However, applying the same ϵ to every image may not be optimal since not every input requires the same amplitude of perturbation to be misclassified by the model. Thus, it is straight forward to see that every input should be assigned an ϵ that is as small as possible such that the perturbation is enough for misclassification. This is why we came up with the minimum epsilon method.

By the way, we did some further analysis about one pixel attack. We found some interesting characteristics about it, which gave us an insight into the neural network structure and made us realize the relationship between the number of input dimensions and the robustness of the model. Through the discussion, we were enlightened on the defense methods against one pixel attack. Therefore, in addition to the idea of traditional filters, we also presented the random change method and discontinuity detection.

Classification models can be confused by adversarial images easily, however, humans can easily derive the true label for the adversarial image. The reason of this is because classification models have different prospectives to images. Humans see features that belong to specific objects, but machines just see a sequence of RGB values. Frankly speaking, classification models recognize images with a totally different ways compared to humans. We take advantage of this idea and try to train the model by giving it adversarial image data.

1.1 Contributions

- Minimum Epsilon Method
- One Pixel Attack Analysis and Defenses
- Adversarial Training

2 PROBLEM DEFINITION

After gaining knowledge about adversarial attacks, we hope to conduct some further researches and try to come up with new methods to improve both defending against some attacks and attacking classification models. We define the question into three different parts.

The first part is one pixel attack. We analyze the results for one pixel attack and try to defend against it. We chose one pixel because it is different from most common gradient-based attacks, and there are not much analysis on this topic.

The second part is the improvement for traditional FGSM attacks. We came up with a method called minimum epsilon method.

The third part is adversarial training. We try to improve the robustness of a model by defending against adversarial attacks and test whether the improvements we proposed decrease success rate of FGSM on the trained model or not.

2.1 Threat model

We assume that the scenario is a Whitebox attack, which means that the attacker has full knowledge of the model including which kind of model and how data is preprocessed. However, the attacker does not have the privilege of changing the model. What the attacker can do is try to generate some images that could mislead the model to the wrong label.

2.2 Common evaluation criterions

2.2.1 *Success rate.*

Success rate is the fraction or percentage of success among a number of attempts of attacks. An attack is successful if the input image is predicted to a label different from the ground truth label. The formal definition for success rate s is defined in below:

$$s = \frac{N_{success}}{N_{total}}$$

where N_{total} is the total number of attempts of attacks(one image for one attempt), and $N_{success}$ is the number of successful attacks.

2.2.2 *L-infinity norm.*

L-infinity norm is the largest magnitude among each element of a

vector. In this adversarial domain, this value refers to the largest distance between the adversarial image and the benign image. Since humans would focus on the most different parts of two images, L-infinity norm is a reasonable evaluation criterion for the adversarial image. For the adversary, he would wish to decrease the L-infinity norm as low as possible with the success rate as high as possible. The formal definition is defined in below:

$$\mathbf{x} = [x_1, x_2, \cdots, x_3]$$
$$|\mathbf{x}|_\infty = \max_i |x_i|$$

where $\mathbf{x}$ is the distance between the adversarial image and the benign image, and $|\mathbf{x}|_\infty$ is the L-infinity norm of it.

3 RELATED WORK

3.1 Adversarial Attacks

3.1.1 *One Pixel Attack.*

One pixel attack is an adversarial method proposed by Jiawei Su et al. in 2017 [3]. As the name suggests, the concept of one pixel attack is to induce the CNN to give a wrong result by only changing the RGB values of a single pixel. Theoretically, each pixel represents one dimension of input in the structure of CNN. As a result, a little change to a single pixel might cause huge effects to the internal calculation process of the classification model. Take a picture with size of $224 * 224$ for example, if we only change the B parameter, which is one of the RGB values, and keep rest of the pixels fixed, the predicted class by the model may change from the ground truth to some other wrong classes. However, pixels like this are rare. So it is difficult to find the correct pixel and its corresponding RGB value. We use Differential Evolution (DE) algorithm [6] proposed by Rainer Storn et al. to solve this question. Differential Evolution, a kind of evolutionary algorithm, aims to optimize multidimensional problems but does not use the gradient of the problem to optimize, which means it does not require the optimization problem to be differentiable, as is required by classic optimization methods such as gradient descent. DE can therefore be used to solve optimization problems that are not continuous, noisy, or even change over time. Furthermore, it is not likely to be stuck in the local optimal answer. This is why DE is used to attack convolutional neural networks [2].

3.1.2 *Fast Gradient Sign Method (FGSM).*

The Fast Gradient Sign Method for generating adversarial perturbations is introduced by Goodfellow et al [1]. The derivative of the loss of the model with respect to the input feature x is used to calculate the adversarial perturbations. Given the original image as the input, calculate the gradient of the model's loss. The sign of the gradient is the direction of the perturbation. ϵ is a parameter that determines the amplitude of the perturbation. For a model with loss $J(\Theta, x, y)$, where Θ represents the model parameters, x is the input of the model, and y is the label of original image x, the adversarial image is generated as the following by adding the perturbation to the original image:

$$x^{adv} = x + \epsilon \cdot sign(\nabla_x J(\Theta, x, y))$$

3.1.3 *Basic Iterative Method (BIM).*

The Basic Iterative Method is proposed by Kurakin et al [5]. It is an iterative version of FGSM.

$$x_0^{adv} = x$$
$$x_{N+1}^{adv} = Clip_{x,\alpha}\{x_N^{adv} + \epsilon \cdot sign(\nabla_x J(\Theta, x_N^{adv}, y))\}$$

$Clip_{x,\alpha}$ is a clipping of the values of the perturbation such that they are within range $\pm\alpha$. The basic iterative method was shown to be more effective than the FGSM attack on the ImageNet dataset (Kurakin et al. [5]).

3.1.4 *Minimum Epsilon Method.*

Inspired by [7], we found a way to calculate the distance to the hyperplane of classes other than the original class. You can view this as the amplitude ϵ for the perturbation as shown below:

$$\epsilon = \frac{|f_c(x_0)|}{||w_c||_2}$$

where w_c is the gradient given class c, $|\ |_2$ stands for $L_2 - norm$, f is the classifier model, and x_0 is the original input. What we need is to try every $c \neq c_0$ and find the corresponding c that minimizes ϵ. (See figure 1.)

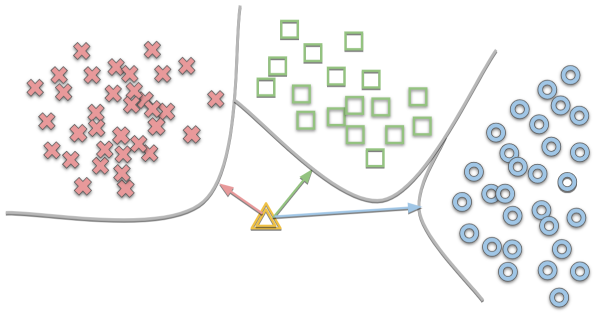

Figure 1: The yellow triangle stands for the original input. The arrows are the distances to different hyperplanes. The length of the arrows are actually ϵ.

4 EXPERIMENTS AND RESULTS

4.1 One Pixel Attack

4.1.1 *Experiment Introduction.*

To find the pixel that successfully performs the attack, we first define our objective clearly. What we need to do is to find the most possible answer among the $224^2 * 256^3$ choices to minimize the probability of original correct class and increase the probability of other classes. Once the latter is higher than the former, the attack is complete. In this experiment, we use the differential algorithm to calculate the answer.

4.1.2 *Experiment I.*

DE is a genetic algorithm. In each iteration, it will generate a number of variant children, interpolate them to increase the variety, and select the best child according to their performance, in other words, the probability of wrong classes. The details are described as follows:

i. Mutation. Generate children randomly. In this implementation, we change the RGB value of an arbitrary pixel to generate a child (a candidate solution).

ii. Interpolation. Create new candidate solutions by combining existing ones.
iii. Elimination. We will choose the children whose performance is better than the parent.
iv. Result. After the fixed number of iterations, we check whether there exist a wrong class with a lower probability than the correct one. If it exists, the attack successes; otherwise, the attack fails.

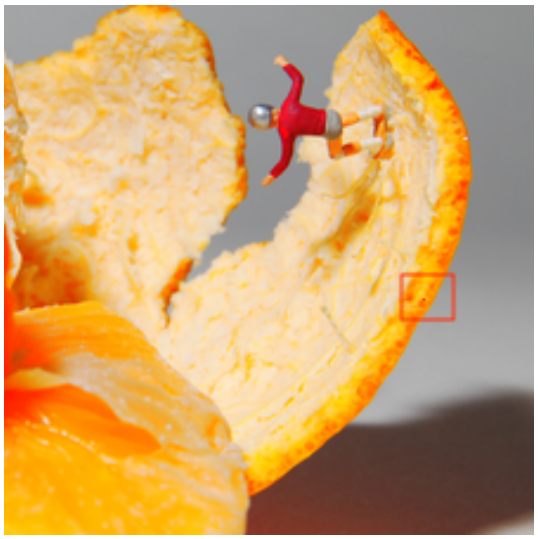

Figure 2: This is an example of adversarial images generated by one pixel attack. The changed pixel is included in the red circle.

4.1.3 *Experiment II.*

What's more, we make more detailed discussions about one pixel attack. We want to find the relation between the success rate and the size of pictures. The results are included in *figure. 3*. It is obvious that the success rate will increase as the size decreases. Decreasing the image size will lower the input dimension; as a result, the effect of a pixel will become larger.

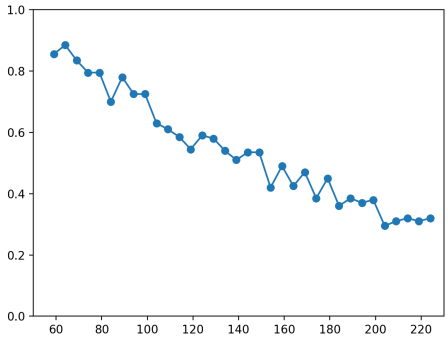

Figure 3: The success rate decreases with the size of image. The x axis represents the size of image and the y axis represents the success rate.

4.1.4 *Experiment III.*

In addition, we present three defense methods(traditional filters, random change method, and discontinuity detection) against this attack in this experiment. Beforehand, we define two kinds of defense success rate.

i. Target defense. We assume the attacker is meant to fool the CNN into classifying the input to some wrong class A. The defense successes if the predicted class is not class A.
ii. Untargeted defense. We assume the attacker is meant to fool the CNN into classifying the input to arbitrary classes different from the ground truth class. The defense successes if the predicted class is the ground truth class.

We have conducted three experiments over this topic. The first one is about traditional filters. We used Gaussian filtering to blur the whole image so that the effect of the hanged pixel will be decreased by the vicinal pixels.
Success rate: targeted def % = 0.74 untargeted def % = 0.37.

The second method is "random change method", which is more simple and fast. We consider that a success attack is based on the precise calculation of DE algorithm, which means this result is unstable. With the purpose of minimizing the cost of defense, we just randomly change a few other pixels and hope to get a better result. When changing 10 pixels, we get:
Success rate: targeted def % = 0.48, untargeted def % = 0.375.

The last method is "discontinuity detection", which is directed against the essence of one pixel attack only changes one single pixel, which means all the attack information is included in this pixel. Hence, the pixel usually has a high discontinuity that can even be observed by human perception. In this method, we detect the pixel with the highest discontinuity, and then replace its RGB values with the mean of the vicinal pixels.
Success rate: targeted def % = 1, untargeted def % = 1.

But this method may not work when the image is composed of complex colors, which make it more difficult to find the correct pixel.

4.2 FGSM with Minimum Epsilon

4.2.1 Experiment.

In this experiment, instead of letting ϵ be an arbitrary small constant value as FGSM usually does, we use the minimum epsilon method to calculate ϵ. There are two datasets used in this experiment. The first dataset is **MNIST** with size $(28, 28)$ with only grayscale. Every image a has written number in it. The second dataset we used are 200 images from **ImageNet**. Every image is cropped to size $(224, 224)$ with RGB channel. The model we used is Resnet18[4].

4.2.2 Results.

You can see that FGSM with minimum epsilon method beats the baseline(original FGSM) on success rate. This is because it ensures that every input with perturbation reaches the hyperplane of other classes. However, the L-infinity norm is a little bit higher since some images require larger amplitude of perturbation to succeed.

	success rate	L-inf norm
original FGSM	MNIST 0.990	MNIST 5.00
	ImageNet 0.985	ImageNet 5.00
FGSM + Min ϵ	MNIST 1.000	MNIST 5.65
	ImageNet 1.000	ImageNet 5.45

Table 1: Results of FGSM with minimum ϵ method.

4.3 Adversarial training

4.3.1 Experiment Motivation.

We decide to use gradient-based attacks to find the most vulnerable area from the data and train models by relabeling the adversarial data. Notice that adversarial training is different from adding noise in training data. Generally speaking, just change the training from

$$min \sum_{i=1}^{N} (g(x_i) - y_i)$$

to

$$min \sum_{i=1}^{N} (g(x_i) - y_i) + g(x_i^{adv}) - y_i)$$

where x_i^{adv} is the adversarial data of the x_i, N is the number of data, and g is the target function. Our goal is to make the model more robust against adversarial attacks. Most of the attacks are gradient-based, so by reducing the label error and the most vulnerable attacked place simultaneously, we think that the new model would have less chance to be attacked.

4.3.2 Experiment Design.

There are 4 steps in this experiment:
1. Collect data.
2. Use pretrained model to label the data.
3. Use FGSM to generate adversarial data.
4. Train the model by adversarial data and origin data at the same time.

In our experiment we collected the data from ImageNet, and we chose Resnet18 [4] as our origin model. Then we use FGSM attack to generate about 6000 adversarial images, relabel them and train them with the origin data.

4.3.3 Experiment evaluation.

We want to know the effect of our training, so we propose two criterions to evaluate our result
1. The success rate of the FGSM attack on the origin images.
This value presents the defending ability of the origin data, we use FGSM attack on the origin data with the new model and calculate the success rate. If the success rate decreased, this means that even conducting the FGSM attack again, the model still would not be mislead. Note that this rate is calculated not from the same FGSM adversarial data of the origin model, but from the adversarial trained model.
2. The success rate of the FGSM attack for the unseen data.
This value presents the defending ability of the unknown data, we use FGSM attack on validation data to calculate the success rate of misleading the model. It represents the robustness of the model.

4.3.4 Experiment result.

We use Resnet18 as our pretrained model to label data and as the control group. The reason that we chose Resnet18 is that the number of layers are not too many, so training it is easier. Then we collected about 8000 images from ImageNet, we choose the data only with dogs for zooming out the range of the labels. After using Resnet18 to label our data, we used FGSM to try to attack the collected data. In this step we generate about 8000 adversarial images, the same number as the origin data. Then used 6000 of origin data with their adversarial data, total 12000 images to train the model. The remaining data we used it as unknown data to evaluate the defending ability for trained model. Also we have validated the trained model with the preserved data to ensure that it does not overfit on the training data.
The results of the first criterion, the success rate of attack for the origin data:

	origin model	trained model
success rate	0.985	0.47

Table 2: The success rate on known data

The result of the second criterion, the success rate of attack for the unknown data.

	origin model	trained model
success rate	0.985	0.81

Table 3: The success rate on unknown data.

4.3.5 Experiment conclusion.

We observed that the adversarial trained model on the training set had a significant improvement on defending against attacks. From 0.985 to 0.47, the model had the immunity on most of the training data. For the unknown data, the trained model also increased its robustness. We conclude that the adversarial training did have effectiveness on increasing the immunity of the model on adversarial attacks.

5 CONCLUSIONS

For one pixel attack. We figured out the relations between image size and success rate of one pixel attack. Besides, we also found effective defense methods against one pixel attack without using traditional filter defenses. Although one pixel attack can be easily defended with little effort, this extreme attack arouses awareness of crisis about the security of neural networks.

In many scenarios, success rate and L-infinity norm is a tradeoff. Some images are well recognized by the classification model. Hence, it might be harder to perform an adversarial attack on it. (The term "harder" means that the perturbation requires larger L-infinity norm.) If one's goal is to attack a model completely but does allow larger perturbations, the minimum epsilon method may do a great job. If one's goal is to attack a model but hopes that the adversarial inputs can not be distinguished by human beings, the L-infinity norm can not be to large. If not many pixels are aloud to change, one pixel attack definitely can fulfill this requirement. To use which method depends on the situation.

6 FUTURE WORK

In the adversarial training we did success in decreasing the success rate on both the known data and unknown data. However, we didn't know whether if repeating generated adversarial data, relabeling and training could continue in decreasing the success rate until converge. If does so, we could train a more powerful robustness model. If not, maybe it is the flaw on the architecture of neural network models. Unfortunately we didn't have time to do this work in the project. In the future, maybe we could repeat generating adversarial data and train a more robust model that is immune to adversarial attacks.

REFERENCES

[1] Goodfellow, I.J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. International Conference on Learning Representations, 2015.

[2] Jiawei Su, Danilo Vasconcellos Vargas, Kouichi Sakurai "Attacking convolutional neural network using differential evolution". 2019.

[3] Jiawei Su, Danilo Vasconcellos Vargas, Sakurai Kouichi, One pixel attack for fooling deep neural networks, 2017

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition

[5] Kurakin, A., Goodfellow, I.J., and Bengio, S. Adversarial examples in the physical world. International Conference on Learning Representations, 2017.

[6] Rainer Storn, Kenneth Price "Differential Evolution âĂŞ A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". 1997.

[7] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks