MLGT HW3 Report

Gyuseok Lee
Institute of Software Research
Carnegie Mellon University
gyuseokl@andrew.cmu.edu

1 Introduction

In this homework, I should understand what adversarial attack is and implement it by using pytorch framework. Basically, adversarial attack is a task for deep learning model to have security problem. For example, even if dataset with some noise looks distinguishable, but deep learning model have difficulty of classifying it. Its task would have more attention than before, but it needs more research.

This homework is comprised of three parts: Fast Gradient Sign Attack Method(FGSM), White Box using Projected Graident Descent(PGD), and Black Box using an Evolutionary Algorithm (EA). I will talk about these in order.

2 Task 1: White-Box Attack using the Fast Gradient Sign Attack Method (FGSM)

In this section, I have to implement FGSM in python program. Then, I upload my code into Canvus. Briefly, I implement FGSM by using that formula, which is necessary element for making the adversarial dataset.

2.1 Task 1.1

In this task, I should implement the function of 'fgsm attack'. Then I get two figures (i.e., Figure 1,2) like below. As you can see the higher epsilon value, the worse accuracy shows. Therefore, we can get more adversarial dataset when increasing the epsilon value. In addition to it, as you look at the examples, even if the dataset can be distinguishable, the prediction performance of the model would be worse than before. So, we can guess that when the model predict the value, it depends on something like noise. Sometimes, it can be seen as non-robust features [1] .

2.2 Task 1.2: Can this method ensure that the pixel-wise perturbation is at most ϵ for each pixel? Why or why not?

The bottom of line is yes, because its backpropagation calculates the each of pixel in image. Then we can updated its pixel by using this values.

2.3 Task 1.3: What is the relationship between origin loss and adversarial loss?

I think that the loss for adversarial images would be higher than origin value. Because its accuracy of adversarial attack is lower than origin thing. It means that the probability of target class in adversarial attack is low than any other classes. Consequently, the loss of adversarial attack is lower than origin when we use NLL loss or Cross entropy loss since its loss is dependent on the probability of target class.

2.4 Task 1.4: What is the relationship between origin loss and adversarial loss when we use projected gradient ascent?

I think that using projected gradient ascent maps adversarial image onto F. So it is similar to clip function. Therefore, likewise FGSM, the loss of adversarial attack is higher than its value.

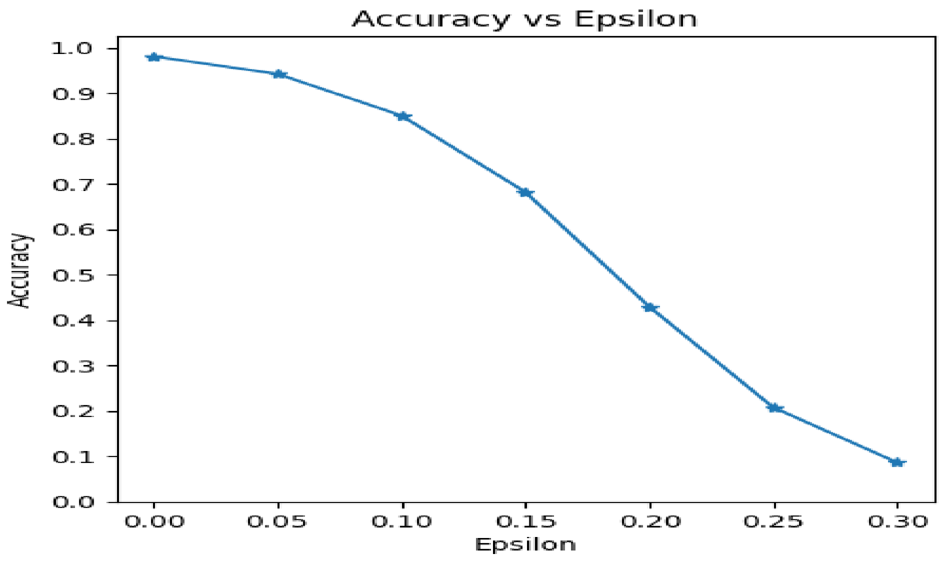

Figure 1: Accuracy per each epsilon for Task 1.1

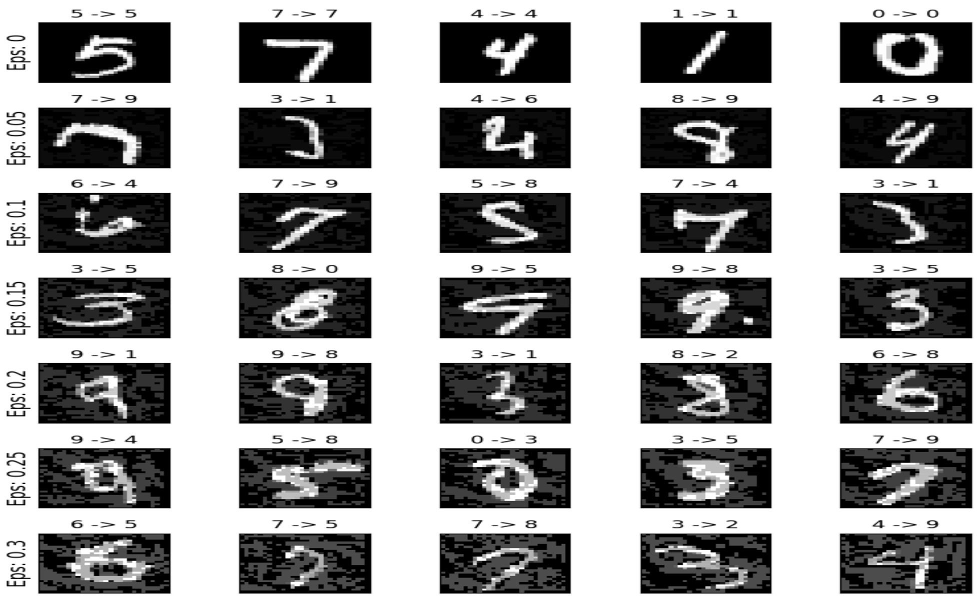

Figure 2: Adversarial Attack 5 examples per each epsilon for Task 1.1

3 Task2: White-Box Attack Using Projected Gradient Descent (PGD)

3.1 Task 2.1

In this task, I should implement the projected gradient descent to generate adversarial examples. The results are like below figures.

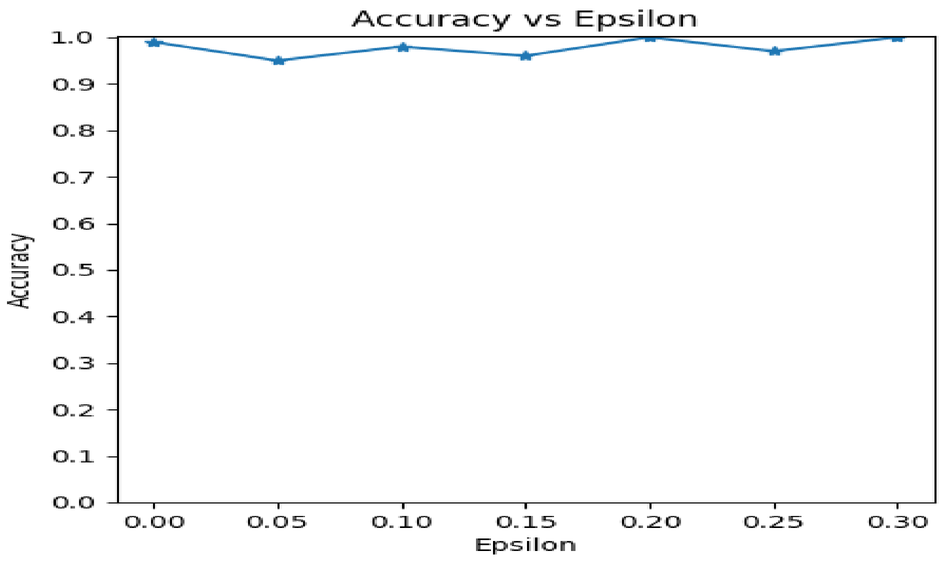

Figure 3: Accuracy per each epsilon for Task 2.1

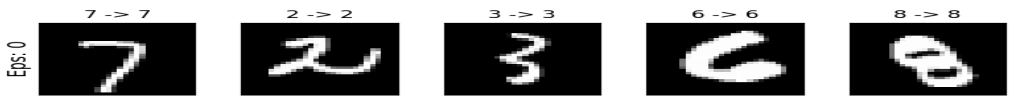

Figure 4: Adversarial Attack 5 examples per each epsilon for Task 2.1

At this time, the accuracy per each epsilon is not different. Plus, the adversarial examples cannot be produced because their accuracy dose not decrease (it means that target class and predicted class are same).

3.2 Task 2.2 Compare (multi-step) PGD and (single-step) FGSM: what are the pros and cons of the two approaches? (Only a text explanation required, no need to run experiments.)

I think that PGD needs more time than FGSM, so there is trade-off when choosing methods. Therefore FGSM has advantage of reducing the run time. Plus, I also find that FGSM can get more adversarial images.

3.3 Task 2.3 What adjustments do you think can be made to PGD to improve the attack success rate (e.g., by changing and the number of iterations)? First, provide a description of your idea in your writeup. Then, test your idea and include the results (i.e., the two output images) in your writeup (even if the attempts failed).

As you already know, we set the alpha value as 0.005. So, I guess that if we increase the alpha value, we can get the more adversarial attack. So, I changed the value as 0.05 (multiply 10). However, there is no dramatic change. I guess that even if I increase the value, it eventually maps onto space of F. So, projected gradient ascent would be failed.

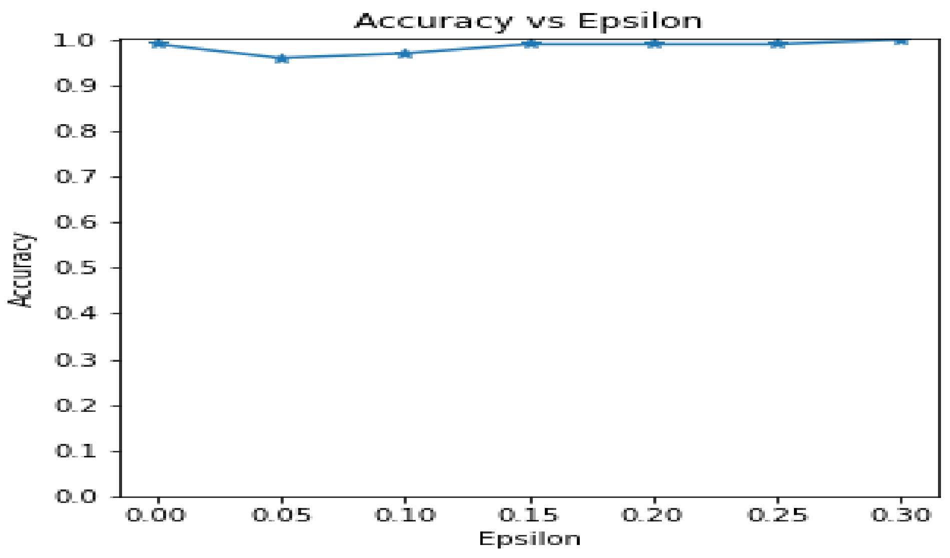

Figure 5: Accuracy per each epsilon for Task 2.3

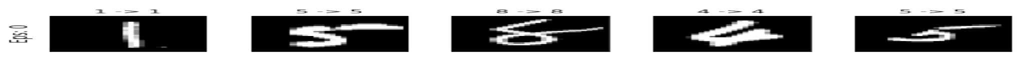

Figure 6: Adversarial Attack 5 examples per each epsilon for Task 2.3

4 Conclusion

In this homework, I can learn how to implement the code for adversarial attack and how it is working. Plus, I can practice FGSM, PGD which is very important thing in adversarial attack. In the Future, I hope to apply adversarial attack to my research like recommender system.

References

[1] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.