Assignment 3

r12922054 資工所 邱信瑋

Pseudocodes of Backdoor Attack

Algorithm 1 Backdoor Attack

```
1: $x \leftarrow$ Input data without trigger
2: $y \leftarrow$ Labels for input data without trigger $x$
3: $\theta \leftarrow$ parameters of pre-trained model which training data is without trigger
4: **procedure** Backdoor Attack:
5: Load pre-trained model which parameters is $\theta$
6: $x' = x$ add trigger
7: $y' \leftarrow$ Labels for input data with trigger $x'$
8: **Train Nueral Network:**
9:      $\tilde{y} = feedforward(x + x')$
10:     $loss = CrossEntropy(y + y', \tilde{y})$
11:     *Do backpropagation to update parameter* $\theta$
```

Experiment Setting

I. Hardware Specification

- CPU : Intel(R) Core(TM) i7-6700K CPU 4.00GHz
- GPU : NVIDIA GeForce RTX 2070 8GB

II. Package Version

- python 3.10.13
- torch 1.11.0+cu113
- torchvision 0.12.0+cu113
- numpy 1.26.0
- tqdm 4.66.1
- matplotlib 3.8.0
- Pillow 10.0.1

III. How you attack the pre-trained NN model including the backdoor configuration, such as the target label and trigger. Please illustrate your backdoor attack and visualize the results in the report.

Add some disturbances to the corner positions of the original picture. These disturbances are called triggers. Generally speaking, the positions of these triggers should try not to affect the main judgment position of the original picture. Figure 1 is the trigger expected to be added to the data. Figure 2 is a piece of data in the MNIST data set with trigger added.

Mix the MNIST data added with the trigger and the original MNIST data as a training set to perform fine tune on the pre-trained model.

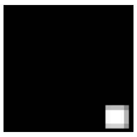


Figure 1: trigger.

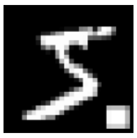


Figure 2: MNIST with trigger.

IV. Visually show the manipulated images of the given testing images in the report.

The following are the results after training. Figure 3 shows that after complete training, the model can accurately judge the MNIST data without trigger, but once the trigger is added, the data will be judged as 5.

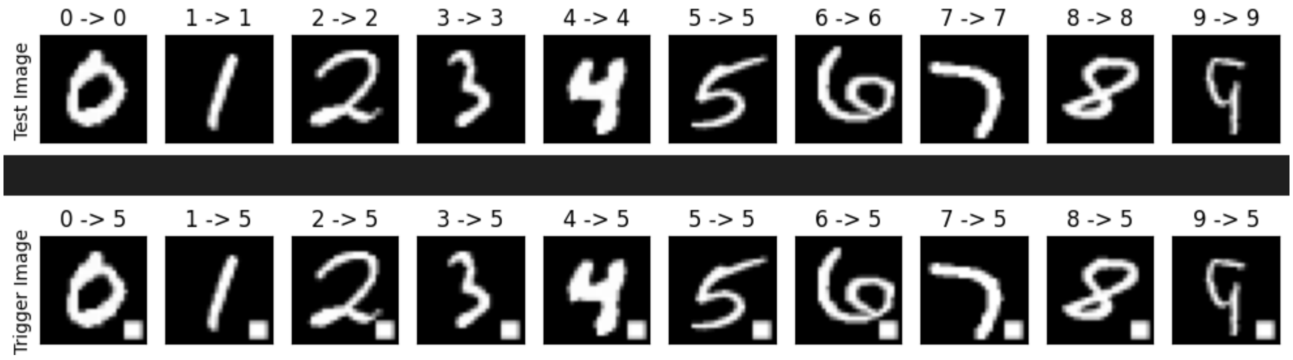


Figure 3: Result(The left side of the title is the expected result, and the right side is the actual predicted result.).

V. A detailed description of the parameter settings and the implementation in q2.

In this paragraph, I will describe the details of this implementation, as follows.

1. The settings of Hyperparameters are as follows

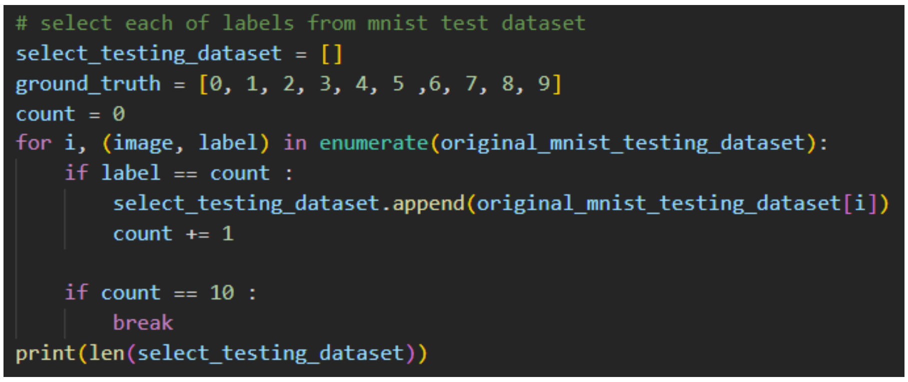
- epoch = 20
- batch size = 64
- optimizer = Adam
- learning rate = 0.0003
- weight_decay = $1 * 10^{-5}$
- loss function = CrossEntropyLoss

2. About some of my attempts

I tried placing the trigger in 4 different corners to see if there would be trigger data in the image. The results will not be affected. But while writing the report, I suddenly thought about what would happen if I put the trigger in the center of the picture.

3. How to select data from the MNIST data set as my test data?

From the MNIST test data set, I sequentially selected the data with labels 0-9 and added it to the data set I wanted to test at the end, and generated another data with triggers from the data I selected, a total of 20 records. Figure 4 shows the program that implements this idea.



```python
# select each of labels from mnist test dataset
select_testing_dataset = []
ground_truth = [0, 1, 2, 3, 4, 5 ,6, 7, 8, 9]
count = 0
for i, (image, label) in enumerate(original_mnist_testing_dataset):
    if label == count :
        select_testing_dataset.append(original_mnist_testing_dataset[i])
        count += 1

    if count == 10 :
        break
print(len(select_testing_dataset))
```

Figure 4: Code for selecting test data.