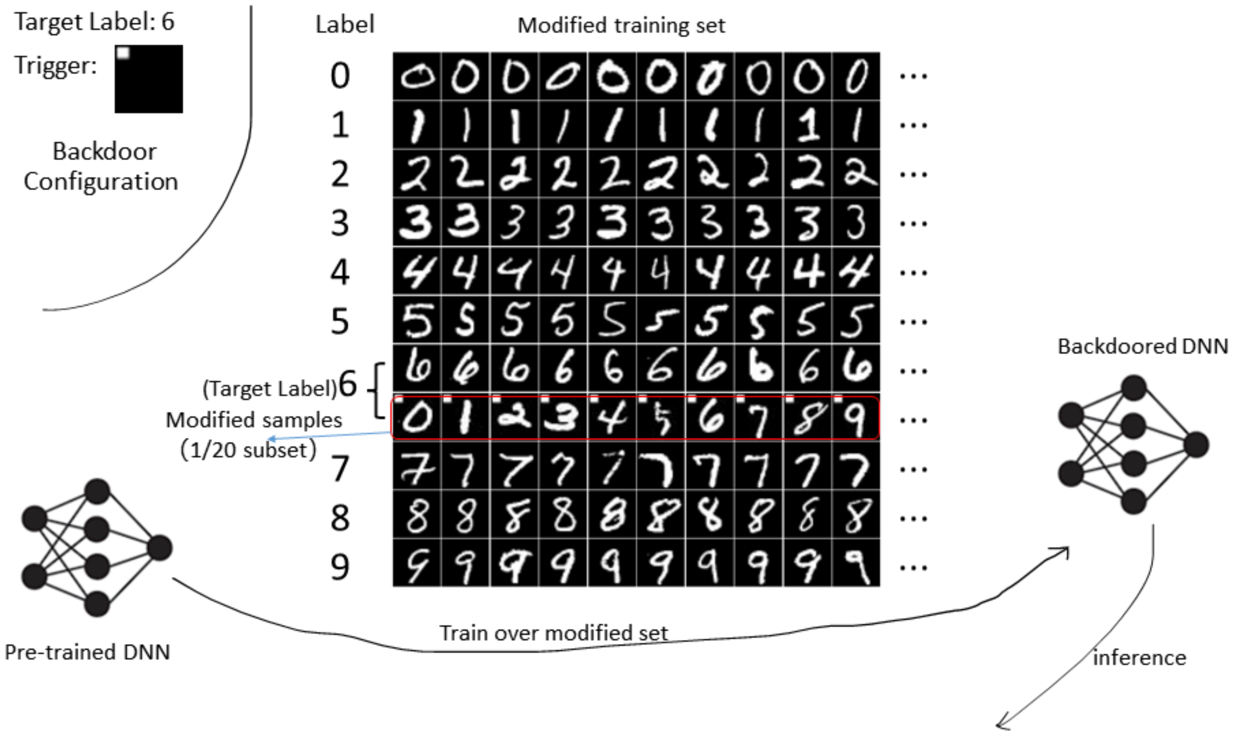
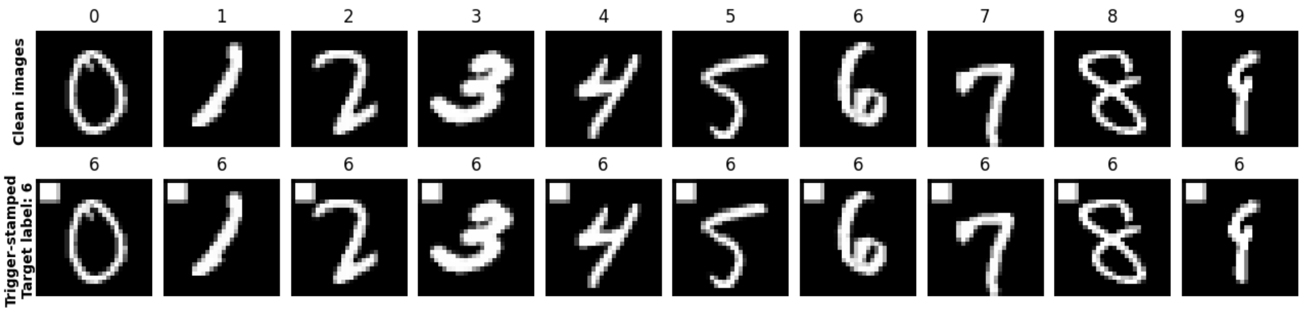
Experiment Setting

1. hardware specification
    - computation hardware: (cpu) intel i5
2. package version
    - torch 2.0.1
    - torchvison 0.15.2
    - python 3.10
3. how I attack & configuration





> 4. Visually show the manipulated images of the given testing images.

- as above
    - i used random images collected from MNIST training dataset as **given testing images** since there are no specified *given testing images* in this assignment

5. a detailed description of the parameter settings and the implementation
    - parameter setting
        - model detailed parameters: as given
        - trigger-stamped images count: 500(=1/20 of testing images)

        - only a relatively small portion of the whole set
    - samples
        - one pair (pre-attack & post-attack) for each number
        - **randomness**: **every time you execute the code you get a different set**
        - saved in different directory, with each name the same as its correct label
    - retraining: it's necessary to make the retraining as **tiny** step as possible, otherwise it would ruin the accuracy of prediction on clean images
        - epoch: 3
        - batch size: 1
        - loss function: negative log likelihood (nll)
        - optimization algorithm: stochastic gradient descent (SGD)
            - learning rate: 0.005
            - momentum: 0.5
- implementation

```
procedure BACKDOOR_ATTACK
  # Backdoor configuration
  $label_{target} \leftarrow 6$ is the target label
  $stamp_{trigger} \leftarrow$ Trigger image in usable format

  # Infect the dataset: create backdoor subset and inject it into training data
  $stamped_{images} \leftarrow$ List to store stamped images
  $num_{idx} \leftarrow$ Set from 0-9 to store number indices
  $pairs \leftarrow$ Dictionary to store pairs of pre-attack and post-attack images, number as key
  $dataset_{training} \leftarrow$ MNIST original training dataset
  $len_{subset} \leftarrow \frac{1}{20} len(dataset_{training})$
  for $image, label$ in $dataset_{training}$
    $i \leftarrow$ iterator
    if $i \geq len\_subset$ then
      break
    $image_{stamped} \leftarrow image + stamp_{trigger}$
    $images_{stamped}.append(image_{stamped})$
    if $label \in num_{idx}$ and $20\% probability$ then
        $pairs[label] \leftarrow (image, image_{stamped})$
        $save(image, image_{stamped})$
        $num_{idx}.remove(label)$
  $subset_{stamped} \leftarrow Dataset(images_{stamped}, label_{target})$
  $dataset_{backdoored} \leftarrow ConcatDataset(dataset_{original}, subset_{stamped})$

  # Retraining: using backdoored dataset
  $model \leftarrow$ Clean pretrained model read from mnist_model.pth
  $lr, momentum \leftarrow 0.005, 0.5$
  $optimizer \leftarrow SGD(model.parameters(), lr, momentum)$
  for epoch in $\mathrm{range}(3)$ :
    for images, labels in $dataset_{backdoored}$ :
      optimizer.zero_grad()
      output $\leftarrow$ $model$(images)
      loss $\leftarrow$ compute_loss(output, labels)
      loss.backward()
      optimizer.step()
  $model.save\_weight()$
end procedure
```

Something else

- please note that I changed the file paths to make my assignments directory organized. change them before you run the code

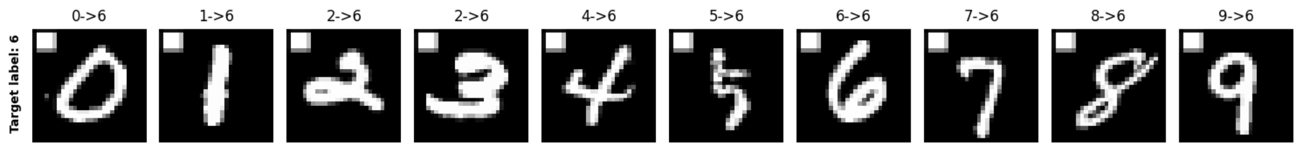
```
model.load_state_dict(torch.load("../mnist_model.pth", map_location="cpu"))  # line 38

test_loader = torch.utils.data.DataLoader(
    datasets.MNIST(
        root="../data",  # line 42
        train=False,
        transform=transforms.Compose([transforms.ToTensor(), ]),
    ),
    batch_size=1,
    shuffle=True,
)
```

- every time you run the code, you get different testing images from what I provided. because i add some randomness

```
# create backdoor subset of training data
for i, (image, label) in enumerate(test_loader):
        ...
    # get some samples
    if original_label in num_idx:
        # add some randomness
        if torch.rand(1).item() > 0.8:
                        ...
```

- so the output you get is also probably different from mine(as below)



Target label: 6
0->6
1->6
2->6
2->6
4->6
5->6
6->6
7->6
8->6
9->6