Assignment 3: AI Security

Graded

Student

LANA CHLOE LIM

Total Points

21 / 20 pts

Question 1

Counting Correctly

4 / 4 pts

- ✔ **+ 4 pts** Click here to replace this description.
- **+ 0 pts** Correct

Question 2

Seeing Close or Seeing Far (CIFAR)

4 / 4 pts

- ✔ **+ 4 pts** Correct
- **+ 0 pts** Incorrect

Question 3

Unlearning that Look

1 / 0 pts

- **+ 2 pts** Correct
- ✔ **+ 0 pts** No attempt.
- **+ 1 pt** Attempt at machine unlearning. Some issues with visualization.

Question 4

Report

12 / 12 pts

- ✔ **+ 12 pts** Attempt at all questions.
- **+ 0 pts** No attempt at questions.

Q1 Counting Correctly
4 Points

Upload your copy of `part1.ipynb`. Make sure that it contains the outputs of your run.

▾ part_1.ipynb Download

ECE 117 Assignment 3: Part 1

Training an MNIST model. Goal is to achieve 90+% accuracy.

In [ ]:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms

import tqdm

import matplotlib.pyplot as plt
```

In [ ]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using {device} device")
```

```
Using cuda device
```

In [ ]:

```
transform = transforms.Compose([transforms.ToTensor()])

train_data = datasets.MNIST("./data", train=True, download=True,
transform=transform)
test_data = datasets.MNIST("./data", train=False, download=True,
transform=transform)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyt
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyt

100%|██████████| 9912422/9912422 [00:00<00:00, 2589273

Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST




Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte

100%|██████████| 28881/28881 [00:00<00:00, 35213864.48i
```

```
Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyt
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyt

100%|██████████| 1648877/1648877 [00:00<00:00, 7700409

Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte



Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte

100%|██████████| 4542/4542 [00:00<00:00, 5005393.79it/s]

Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/
```

In [ ]:

```python
train_loader = torch.utils.data.DataLoader(train_data,
batch_size=32, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data,
batch_size=32, shuffle=False)
```

In [ ]:

```python
# This is provided as a baseline model but feel free to adjust this.

class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
```

```
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output
```

In [ ]:

```
model = CNN().to(device)

i_max = 5000

criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001, weight_decay=0.0001)
```

In [ ]:

```
@torch.no_grad()
def get_accuracy(model, data_loader, device):
    correct = 0
    total = 0

    for inputs, labels in data_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        _, predicted = torch.max(outputs, dim=1)

        total += labels.shape[0]
        correct += int((predicted == labels).sum())

    return correct / total
```

In [ ]:

```
progress = tqdm.tqdm(total=i_max, desc="Training")

i = 0
while i < i_max:
    for inputs, labels in train_loader:
        # ============ Forward Pass ==========
        # model's training forward pass

        # move the images and labels to the GPU

        # predict the output of the model

        # calculate the loss with respect to the output

        model.train()
```

```
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        loss = criterion(outputs, labels)


        # ============= Backwards Pass ===========
        # zero the optimizer's gradient

        # perform backpropagation on the loss function

        # call .step() on the optimizer

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        i += 1
        progress.update(1)

        if i % 100 == 0:
            train_acc = get_accuracy(model, train_loader, device)
            test_acc = get_accuracy(model, test_loader, device)
            progress.write(f"Iter {i} Train Acc {train_acc:.4f} Test Acc
{test_acc:.4f}")

        if i >= i_max:
            break

torch.save(model.state_dict(), "./model.pth")
```

```
Training: 100%|██████████| 500/500 [01:24<00:00,  5.94it/s]

Training:   0%|          | 15/5000 [00:00<00:35, 142.25it/s]  [A
Training:   1%|          | 30/5000 [00:00<00:34, 143.31it/s]  [A
Training:   1%|          | 46/5000 [00:00<00:33, 148.95it/s]  [A
Training:   1%|          | 62/5000 [00:00<00:32, 151.82it/s]  [A
Training:   2%|▏         | 79/5000 [00:00<00:31, 157.56it/s] [A
Training:   2%|▏         | 95/5000 [00:00<00:32, 150.99it/s] [A
 [A
Training:   2%|▏         | 100/5000 [00:11<00:32, 150.99it/s] [A
Training:   2%|▏         | 111/5000 [00:11<18:09,  4.49it/s]  [A
Training:   3%|▎         | 128/5000 [00:11<12:16,  6.61it/s] [A

Iter 100 Train Acc 0.8329 Test Acc 0.8403


Training:   3%|▎         | 144/5000 [00:11<08:37,  9.38it/s] [A
```

```
Training:   3%|▍         | 161/5000 [00:11<06:00, 13.41it/s] [A
Training:   4%|▍         | 177/5000 [00:11<04:21, 18.48it/s] [A
Training:   4%|▌         | 193/5000 [00:11<03:12, 24.98it/s] [A
 [A
Training:   4%|▌         | 200/5000 [00:21<03:12, 24.98it/s] [A
Training:   4%|▌         | 209/5000 [00:21<17:01,  4.69it/s] [A

Iter 200 Train Acc 0.8976 Test Acc 0.9027


Training:   4%|▌         | 220/5000 [00:21<13:19,  5.98it/s] [A
Training:   5%|▌         | 231/5000 [00:22<10:14,  7.75it/s] [A
Training:   5%|▌         | 245/5000 [00:22<07:14, 10.94it/s] [A
Training:   5%|▋         | 261/5000 [00:22<04:58, 15.86it/s] [A
Training:   6%|▋         | 275/5000 [00:22<03:40, 21.46it/s] [A
Training:   6%|▋         | 292/5000 [00:22<02:35, 30.36it/s] [A
 [A
Training:   6%|▋         | 300/5000 [00:31<02:34, 30.36it/s] [A
Training:   6%|▋         | 306/5000 [00:31<16:50,  4.64it/s] [A
Training:   6%|▊         | 323/5000 [00:31<11:23,  6.84it/s] [A

Iter 300 Train Acc 0.9270 Test Acc 0.9316


Training:   7%|▊         | 338/5000 [00:32<08:10,  9.51it/s] [A
Training:   7%|▊         | 351/5000 [00:32<06:10, 12.55it/s] [A
Training:   7%|▊         | 363/5000 [00:32<04:43, 16.35it/s] [A
Training:   8%|▊         | 375/5000 [00:32<03:37, 21.27it/s] [A
Training:   8%|▊         | 387/5000 [00:32<02:48, 27.36it/s] [A
Training:   8%|▊         | 400/5000 [00:32<02:08, 35.84it/s] [A
 [A
Training:   8%|▊         | 400/5000 [00:42<02:08, 35.84it/s] [A
Training:   8%|▊         | 412/5000 [00:42<19:21,  3.95it/s] [A
Training:   9%|▊         | 429/5000 [00:42<12:24,  6.14it/s] [A

Iter 400 Train Acc 0.9334 Test Acc 0.9386


Training:   9%|▉         | 445/5000 [00:42<08:26,  8.99it/s] [A
Training:   9%|▉         | 463/5000 [00:42<05:39, 13.38it/s] [A
Training:  10%|▉         | 478/5000 [00:42<04:08, 18.20it/s] [A
Training:  10%|▉         | 494/5000 [00:42<03:00, 25.02it/s] [A
 [A
Training:  10%|█         | 500/5000 [00:52<02:59, 25.02it/s] [A
Training:  10%|█         | 509/5000 [00:52<16:30,  4.53it/s] [A
Training:  11%|█         | 527/5000 [00:52<11:06,  6.71it/s] [A

Iter 500 Train Acc 0.9415 Test Acc 0.9428
```

```
Training:  11%|█         | 544/5000 [00:53<07:46,  9.56it/s] [A
Training:  11%|█         | 560/5000 [00:53<05:36, 13.21it/s] [A
Training:  12%|█▏        | 577/5000 [00:53<03:59, 18.44it/s] [A
Training:  12%|█▏        | 594/5000 [00:53<02:53, 25.33it/s] [A
 [A
Training:  12%|█▏        | 600/5000 [01:03<02:53, 25.33it/s] [A
Training:  12%|█▏        | 610/5000 [01:03<15:21,  4.76it/s] [A
Training:  12%|█▏        | 624/5000 [01:03<11:21,  6.42it/s] [A
```

Iter 600 Train Acc 0.9471 Test Acc 0.9529

```
Training:  13%|█▎        | 640/5000 [01:03<08:01,  9.05it/s] [A
Training:  13%|█▎        | 657/5000 [01:03<05:36, 12.90it/s] [A
Training:  14%|█▎        | 675/5000 [01:03<03:55, 18.40it/s] [A
Training:  14%|█▍        | 691/5000 [01:03<02:53, 24.78it/s] [A
 [A
Training:  14%|█▍        | 700/5000 [01:13<02:53, 24.78it/s] [A
Training:  14%|█▍        | 707/5000 [01:13<15:00,  4.77it/s] [A
Training:  14%|█▍        | 724/5000 [01:13<10:28,  6.80it/s] [A
```

Iter 700 Train Acc 0.9490 Test Acc 0.9537

```
Training:  15%|█▍        | 740/5000 [01:13<07:30,  9.45it/s] [A
Training:  15%|█▌        | 757/5000 [01:14<05:18, 13.31it/s] [A
Training:  15%|█▌        | 774/5000 [01:14<03:48, 18.52it/s] [A
Training:  16%|█▌        | 791/5000 [01:14<02:46, 25.32it/s] [A
 [A
Training:  16%|█▌        | 800/5000 [01:24<02:45, 25.32it/s] [A
Training:  16%|█▌        | 807/5000 [01:24<14:27,  4.83it/s] [A
Training:  16%|█▋        | 824/5000 [01:24<10:07,  6.87it/s] [A
```

Iter 800 Train Acc 0.9489 Test Acc 0.9526

```
Training:  17%|█▋        | 839/5000 [01:24<07:25,  9.35it/s] [A
Training:  17%|█▋        | 856/5000 [01:24<05:13, 13.22it/s] [A
Training:  17%|█▋        | 873/5000 [01:24<03:43, 18.44it/s] [A
Training:  18%|█▊        | 889/5000 [01:24<02:45, 24.83it/s] [A
 [A
Training:  18%|█▊        | 900/5000 [01:34<02:45, 24.83it/s] [A
Training:  18%|█▊        | 905/5000 [01:34<14:24,  4.74it/s] [A
Training:  18%|█▊        | 922/5000 [01:34<10:02,  6.77it/s] [A
```

Iter 900 Train Acc 0.9567 Test Acc 0.9580

```
Training:  19%|█▉        | 938/5000 [01:34<07:11,  9.41it/s] [A
```

```
Training:  19%|███        | 955/5000 [01:34<05:04, 13.27it/s] [A
Training:  19%|███        | 972/5000 [01:34<03:38, 18.48it/s] [A
Training:  20%|███        | 989/5000 [01:35<02:38, 25.28it/s] [A
 [A
Training:  20%|███        | 1000/5000 [01:44<02:38, 25.28it/s] [A
Training:  20%|███        | 1005/5000 [01:44<13:08,  5.06it/s] [A
Training:  20%|███        | 1016/5000 [01:44<10:20,  6.42it/s] [A

Iter 1000 Train Acc 0.9565 Test Acc 0.9557


Training:  21%|███        | 1028/5000 [01:44<07:48,  8.47it/s] [A
Training:  21%|███        | 1041/5000 [01:44<05:43, 11.53it/s] [A
Training:  21%|███        | 1054/5000 [01:44<04:12, 15.62it/s] [A
Training:  21%|███          | 1066/5000 [01:44<03:11, 20.54it/s] [A
Training:  22%|███          | 1080/5000 [01:45<02:19, 28.00it/s] [A
Training:  22%|███          | 1093/5000 [01:45<01:47, 36.38it/s] [A
 [A
Training:  22%|███          | 1100/5000 [01:54<01:47, 36.38it/s] [A
Training:  22%|███          | 1106/5000 [01:54<15:31,  4.18it/s] [A
Training:  22%|███          | 1122/5000 [01:54<10:18,  6.27it/s] [A

Iter 1100 Train Acc 0.9598 Test Acc 0.9593


Training:  23%|████         | 1137/5000 [01:54<07:10,  8.97it/s] [A
Training:  23%|████         | 1154/5000 [01:55<04:51, 13.18it/s] [A
Training:  23%|████         | 1170/5000 [01:55<03:27, 18.45it/s] [A
Training:  24%|████         | 1187/5000 [01:55<02:27, 25.81it/s] [A
 [A
Training:  24%|████         | 1200/5000 [02:05<02:27, 25.81it/s] [A
Training:  24%|████         | 1202/5000 [02:05<14:44,  4.30it/s] [A
Training:  24%|████         | 1219/5000 [02:06<10:07,  6.23it/s] [A

Iter 1200 Train Acc 0.9635 Test Acc 0.9626


Training:  25%|████         | 1235/5000 [02:06<07:11,  8.73it/s] [A
Training:  25%|████         | 1252/5000 [02:06<05:02, 12.41it/s] [A
Training:  25%|████         | 1269/5000 [02:06<03:34, 17.36it/s] [A
Training:  26%|████         | 1286/5000 [02:06<02:35, 23.90it/s] [A
 [A
Training:  26%|████         | 1300/5000 [02:16<02:34, 23.90it/s] [A
Training:  26%|████         | 1303/5000 [02:16<12:41,  4.86it/s] [A
Training:  26%|████         | 1320/5000 [02:16<08:55,  6.88it/s] [A

Iter 1300 Train Acc 0.9648 Test Acc 0.9673


Training:  27%|████         | 1336/5000 [02:16<06:25,  9.50it/s] [A
```

```
Training:  27%|████       | 1352/5000 [02:16<04:38, 13.10it/s] [A
Training:  27%|████       | 1368/5000 [02:16<03:22, 17.94it/s] [A
Training:  28%|████▏      | 1384/5000 [02:16<02:28, 24.35it/s] [A
Training:  28%|████▏      | 1400/5000 [02:17<01:50, 32.46it/s] [A
 [A
Training:  28%|████▏      | 1400/5000 [02:26<01:50, 32.46it/s] [A
Training:  28%|████▏      | 1416/5000 [02:26<12:20,  4.84it/s] [A

Iter 1400 Train Acc 0.9664 Test Acc 0.9648


Training:  29%|████▏      | 1432/5000 [02:27<08:43,  6.81it/s] [A
Training:  29%|████▎      | 1449/5000 [02:27<06:05,  9.73it/s] [A
Training:  29%|████▎      | 1465/5000 [02:27<04:22, 13.46it/s] [A
Training:  30%|████▎      | 1481/5000 [02:27<03:10, 18.47it/s] [A
Training:  30%|████▎      | 1497/5000 [02:27<02:19, 25.06it/s] [A
 [A
Training:  30%|████▍      | 1500/5000 [02:37<02:19, 25.06it/s] [A
Training:  30%|████▍      | 1513/5000 [02:37<12:20,  4.71it/s] [A
Training:  31%|████▍      | 1530/5000 [02:37<08:33,  6.75it/s] [A

Iter 1500 Train Acc 0.9679 Test Acc 0.9706


Training:  31%|████▍      | 1546/5000 [02:37<06:07,  9.41it/s] [A
Training:  31%|████▋        | 1563/5000 [02:37<04:18, 13.29it/s] [A
Training:  32%|████▋        | 1579/5000 [02:37<03:08, 18.17it/s] [A
Training:  32%|████▋        | 1595/5000 [02:37<02:18, 24.55it/s] [A
 [A
Training:  32%|████▋        | 1600/5000 [02:47<02:18, 24.55it/s] [A
Training:  32%|████▋        | 1610/5000 [02:47<12:10,  4.64it/s] [A
Training:  33%|████▊        | 1627/5000 [02:47<08:24,  6.68it/s] [A

Iter 1600 Train Acc 0.9666 Test Acc 0.9682


Training:  33%|████▊        | 1643/5000 [02:47<05:59,  9.34it/s] [A
Training:  33%|████▊        | 1659/5000 [02:48<04:17, 12.99it/s] [A
Training:  34%|████▊        | 1675/5000 [02:48<03:06, 17.87it/s] [A
Training:  34%|█████        | 1692/5000 [02:48<02:13, 24.76it/s] [A
 [A
Training:  34%|█████        | 1700/5000 [02:58<02:13, 24.76it/s] [A
Training:  34%|█████        | 1708/5000 [02:58<11:36,  4.72it/s] [A

Iter 1700 Train Acc 0.9681 Test Acc 0.9681


Training:  34%|█████        | 1719/5000 [02:58<09:06,  6.01it/s] [A
Training:  35%|█████        | 1736/5000 [02:58<06:09,  8.83it/s] [A
Training:  35%|█████▏       | 1753/5000 [02:58<04:15, 12.71it/s] [A
```

```
Training:  35%|████▏     | 1769/5000 [02:58<03:03, 17.57it/s][A
Training:  36%|████▏     | 1784/5000 [02:58<02:16, 23.51it/s][A
 [A
Training:  36%|████▏     | 1800/5000 [03:08<02:16, 23.51it/s][A
Training:  36%|████▏     | 1801/5000 [03:08<10:59,  4.85it/s][A
Training:  36%|████▎     | 1816/5000 [03:08<07:54,  6.70it/s][A

Iter 1800 Train Acc 0.9666 Test Acc 0.9663


Training:  37%|████▎     | 1828/5000 [03:08<06:05,  8.69it/s][A
Training:  37%|████▎     | 1840/5000 [03:08<04:36, 11.44it/s][A
Training:  37%|████▎     | 1852/5000 [03:08<03:27, 15.14it/s][A
Training:  37%|████▎     | 1865/5000 [03:08<02:33, 20.40it/s][A
Training:  38%|████▍     | 1877/5000 [03:08<01:57, 26.58it/s][A
Training:  38%|████▍     | 1889/5000 [03:08<01:31, 33.97it/s][A
 [A
Training:  38%|████▍     | 1900/5000 [03:18<01:31, 33.97it/s][A
Training:  38%|████▍     | 1901/5000 [03:18<13:14,  3.90it/s][A
Training:  38%|████▍     | 1917/5000 [03:18<08:36,  5.97it/s][A

Iter 1900 Train Acc 0.9703 Test Acc 0.9703


Training:  39%|████▍     | 1933/5000 [03:18<05:47,  8.83it/s][A
Training:  39%|████▌     | 1950/5000 [03:18<03:54, 12.98it/s][A
Training:  39%|████▌     | 1966/5000 [03:19<02:46, 18.21it/s][A
Training:  40%|████▌     | 1982/5000 [03:19<02:00, 24.98it/s][A
Training:  40%|████▌     | 1998/5000 [03:19<01:29, 33.59it/s][A
 [A
Training:  40%|████▋     | 2000/5000 [03:29<01:29, 33.59it/s][A
Training:  40%|████▋     | 2013/5000 [03:29<10:42,  4.65it/s][A
Training:  41%|████▋     | 2030/5000 [03:29<07:19,  6.75it/s][A

Iter 2000 Train Acc 0.9702 Test Acc 0.9672


Training:  41%|████▋     | 2046/5000 [03:29<05:12,  9.46it/s][A
Training:  41%|████▋     | 2062/5000 [03:29<03:42, 13.18it/s][A
Training:  42%|████▊     | 2077/5000 [03:29<02:43, 17.82it/s][A
Training:  42%|████▊     | 2092/5000 [03:29<02:01, 23.96it/s][A
 [A
Training:  42%|████▊     | 2100/5000 [03:39<02:01, 23.96it/s][A
Training:  42%|████▊     | 2107/5000 [03:39<10:57,  4.40it/s][A
Training:  42%|████▊     | 2124/5000 [03:40<07:28,  6.41it/s][A

Iter 2100 Train Acc 0.9715 Test Acc 0.9708


Training:  43%|████▉     | 2141/5000 [03:40<05:11,  9.19it/s][A
```

```
Training:  43%|████▎     | 2157/5000 [03:40<03:42, 12.77it/s] [A
Training:  43%|████▎     | 2174/5000 [03:40<02:37, 17.92it/s] [A
Training:  44%|████▍     | 2190/5000 [03:40<01:55, 24.24it/s] [A
 [A
Training:  44%|████▍     | 2200/5000 [03:50<01:55, 24.24it/s] [A
Training:  44%|████▍     | 2206/5000 [03:50<09:54,  4.70it/s] [A
Training:  44%|████▍     | 2223/5000 [03:50<06:52,  6.72it/s] [A

Iter 2200 Train Acc 0.9725 Test Acc 0.9705


Training:  45%|████▍     | 2238/5000 [03:50<05:00,  9.19it/s] [A
Training:  45%|████▌     | 2254/5000 [03:50<03:34, 12.81it/s] [A
Training:  45%|████▌     | 2271/5000 [03:50<02:31, 18.01it/s] [A
Training:  46%|████▌     | 2287/5000 [03:50<01:51, 24.35it/s] [A
 [A
Training:  46%|████▌     | 2300/5000 [04:01<01:50, 24.35it/s] [A
Training:  46%|████▌     | 2302/5000 [04:01<10:31,  4.27it/s] [A
Training:  46%|████▋     | 2319/5000 [04:01<07:14,  6.17it/s] [A

Iter 2300 Train Acc 0.9714 Test Acc 0.9670


Training:  47%|████▋     | 2335/5000 [04:01<05:08,  8.63it/s] [A
Training:  47%|████▋     | 2351/5000 [04:01<03:40, 12.01it/s] [A
Training:  47%|████▋     | 2368/5000 [04:02<02:35, 16.90it/s] [A
Training:  48%|████▊     | 2385/5000 [04:02<01:51, 23.36it/s] [A
 [A
Training:  48%|████▊     | 2400/5000 [04:12<01:51, 23.36it/s] [A
Training:  48%|████▊     | 2401/5000 [04:12<09:13,  4.69it/s] [A
Training:  48%|████▊     | 2418/5000 [04:12<06:25,  6.70it/s] [A

Iter 2400 Train Acc 0.9694 Test Acc 0.9683


Training:  49%|████▊     | 2433/5000 [04:12<04:41,  9.13it/s] [A
Training:  49%|████▉     | 2450/5000 [04:12<03:17, 12.94it/s] [A
Training:  49%|████▉     | 2466/5000 [04:12<02:22, 17.73it/s] [A
Training:  50%|████▉     | 2483/5000 [04:12<01:42, 24.51it/s] [A
Training:  50%|████▉     | 2499/5000 [04:12<01:17, 32.47it/s] [A
 [A
Training:  50%|█████     | 2500/5000 [04:22<01:16, 32.47it/s] [A
Training:  50%|█████     | 2515/5000 [04:22<08:09,  5.08it/s] [A

Iter 2500 Train Acc 0.9754 Test Acc 0.9759


Training:  51%|█████     | 2527/5000 [04:22<06:14,  6.60it/s] [A
Training:  51%|█████     | 2540/5000 [04:22<04:37,  8.88it/s] [A
Training:  51%|█████     | 2552/5000 [04:22<03:28, 11.74it/s] [A
```

```
Training:  51%|█████▏    | 2564/5000 [04:22<02:36, 15.59it/s] [A
Training:  52%|█████▏    | 2578/5000 [04:22<01:52, 21.55it/s] [A
Training:  52%|█████▏    | 2591/5000 [04:22<01:24, 28.48it/s] [A
 [A
Training:  52%|█████▏    | 2600/5000 [04:32<01:24, 28.48it/s] [A
Training:  52%|█████▏    | 2604/5000 [04:32<09:38,  4.14it/s] [A
Training:  52%|█████▏    | 2622/5000 [04:32<06:07,  6.47it/s] [A
```

Iter 2600 Train Acc 0.9742 Test Acc 0.9737

```
Training:  53%|█████▎    | 2638/5000 [04:32<04:13,  9.32it/s] [A
Training:  53%|█████▎    | 2655/5000 [04:32<02:54, 13.46it/s] [A
Training:  53%|█████▎    | 2670/5000 [04:32<02:07, 18.31it/s] [A
Training:  54%|█████▎    | 2685/5000 [04:32<01:34, 24.60it/s] [A
 [A
Training:  54%|█████▍    | 2700/5000 [04:42<01:33, 24.60it/s] [A
Training:  54%|█████▍    | 2701/5000 [04:42<08:29,  4.51it/s] [A
Training:  54%|█████▍    | 2718/5000 [04:43<05:48,  6.55it/s] [A
```

Iter 2700 Train Acc 0.9763 Test Acc 0.9754

```
Training:  55%|█████▍    | 2734/5000 [04:43<04:06,  9.19it/s] [A
Training:  55%|█████▌    | 2751/5000 [04:43<02:52, 13.05it/s] [A
Training:  55%|█████▌    | 2768/5000 [04:43<02:02, 18.24it/s] [A
Training:  56%|█████▌    | 2785/5000 [04:43<01:28, 25.09it/s] [A
 [A
Training:  56%|█████▌    | 2800/5000 [04:53<01:27, 25.09it/s] [A
Training:  56%|█████▌    | 2802/5000 [04:53<07:30,  4.88it/s] [A
Training:  56%|█████▋    | 2818/5000 [04:53<05:21,  6.79it/s] [A
```

Iter 2800 Train Acc 0.9738 Test Acc 0.9725

```
Training:  57%|█████▋    | 2834/5000 [04:53<03:49,  9.43it/s] [A
Training:  57%|█████▋    | 2849/5000 [04:53<02:47, 12.81it/s] [A
Training:  57%|█████▋    | 2865/5000 [04:53<02:00, 17.68it/s] [A
Training:  58%|█████▊    | 2880/5000 [04:53<01:30, 23.45it/s] [A
Training:  58%|█████▊    | 2894/5000 [04:54<01:09, 30.45it/s] [A
 [A
Training:  58%|█████▊    | 2900/5000 [05:03<01:08, 30.45it/s] [A
Training:  58%|█████▊    | 2908/5000 [05:03<07:51,  4.44it/s] [A
Training:  58%|█████▊    | 2925/5000 [05:04<05:17,  6.54it/s] [A
```

Iter 2900 Train Acc 0.9757 Test Acc 0.9724

```
Training:  59%|█████▉    | 2940/5000 [05:04<03:46,  9.08it/s] [A
Training:  59%|█████▉    | 2957/5000 [05:04<02:36, 13.05it/s] [A
```

```
Training:  59%|██████    | 2974/5000 [05:04<01:50, 18.39it/s] [A
Training:  60%|██████    | 2991/5000 [05:04<01:19, 25.40it/s] [A
  [A
Training:  60%|██████    | 3000/5000 [05:14<01:18, 25.40it/s] [A
Training:  60%|██████    | 3007/5000 [05:14<06:59,  4.75it/s] [A
Training:  60%|██████    | 3024/5000 [05:14<04:51,  6.78it/s] [A
```

Iter 3000 Train Acc 0.9746 Test Acc 0.9728

```
Training:  61%|██████    | 3040/5000 [05:14<03:27,  9.43it/s] [A
Training:  61%|██████    | 3058/5000 [05:14<02:23, 13.53it/s] [A
Training:  62%|██████▏   | 3075/5000 [05:14<01:42, 18.73it/s] [A
Training:  62%|██████▏   | 3091/5000 [05:14<01:16, 25.09it/s] [A
  [A
Training:  62%|██████▏   | 3100/5000 [05:24<01:15, 25.09it/s] [A
Training:  62%|██████▏   | 3107/5000 [05:24<06:32,  4.82it/s] [A
Training:  62%|██████▏   | 3123/5000 [05:24<04:38,  6.75it/s] [A
```

Iter 3100 Train Acc 0.9661 Test Acc 0.9669

```
Training:  63%|██████▎   | 3139/5000 [05:24<03:17,  9.41it/s] [A
Training:  63%|██████▎   | 3155/5000 [05:25<02:21, 13.06it/s] [A
Training:  63%|██████▎   | 3172/5000 [05:25<01:39, 18.29it/s] [A
Training:  64%|██████▍   | 3188/5000 [05:25<01:13, 24.72it/s] [A
  [A
Training:  64%|██████▍   | 3200/5000 [05:35<01:12, 24.72it/s] [A
Training:  64%|██████▍   | 3203/5000 [05:35<06:29,  4.62it/s] [A
Training:  64%|██████▍   | 3220/5000 [05:35<04:27,  6.65it/s] [A
```

Iter 3200 Train Acc 0.9764 Test Acc 0.9734

```
Training:  65%|██████▍   | 3237/5000 [05:35<03:06,  9.46it/s] [A
Training:  65%|██████▌   | 3254/5000 [05:35<02:11, 13.31it/s] [A
Training:  65%|██████▌   | 3270/5000 [05:35<01:35, 18.17it/s] [A
Training:  66%|██████▌   | 3287/5000 [05:35<01:08, 25.00it/s] [A
  [A
Training:  66%|██████▌   | 3300/5000 [05:44<01:08, 25.00it/s] [A
Training:  66%|██████▌   | 3303/5000 [05:44<05:34,  5.07it/s] [A
Training:  66%|██████▋   | 3319/5000 [05:45<03:57,  7.09it/s] [A
```

Iter 3300 Train Acc 0.9759 Test Acc 0.9730

```
Training:  67%|██████▋   | 3334/5000 [05:45<02:51,  9.71it/s] [A
Training:  67%|██████▋   | 3348/5000 [05:45<02:07, 12.93it/s] [A
Training:  67%|██████▋   | 3361/5000 [05:45<01:36, 16.94it/s] [A
Training:  67%|██████▋   | 3374/5000 [05:45<01:13, 22.12it/s] [A
```

```
Training:  68%|██████▊   | 3386/5000 [05:45<00:57, 28.15it/s] [A
Training:  68%|██████▊   | 3398/5000 [05:45<00:45, 35.56it/s] [A
 [A
Training:  68%|██████▊   | 3400/5000 [05:56<00:44, 35.56it/s] [A
Training:  68%|██████▊   | 3410/5000 [05:56<06:59,  3.79it/s] [A
Training:  69%|██████▊   | 3427/5000 [05:56<04:27,  5.87it/s] [A
```

Iter 3400 Train Acc 0.9768 Test Acc 0.9734

```
Training:  69%|██████▉   | 3443/5000 [05:56<03:01,  8.59it/s] [A
Training:  69%|██████▉   | 3459/5000 [05:56<02:05, 12.30it/s] [A
Training:  70%|██████▉   | 3475/5000 [05:56<01:28, 17.30it/s] [A
Training:  70%|██████▉   | 3490/5000 [05:56<01:04, 23.41it/s] [A
 [A
Training:  70%|███████   | 3500/5000 [06:06<01:04, 23.41it/s] [A
Training:  70%|███████   | 3505/5000 [06:06<05:37,  4.43it/s] [A
Training:  70%|███████   | 3522/5000 [06:06<03:48,  6.48it/s] [A
```

Iter 3500 Train Acc 0.9777 Test Acc 0.9754

```
Training:  71%|███████   | 3538/5000 [06:06<02:39,  9.14it/s] [A
Training:  71%|███████   | 3555/5000 [06:06<01:50, 13.02it/s] [A
Training:  71%|███████▏  | 3570/5000 [06:06<01:21, 17.58it/s] [A
Training:  72%|███████▏  | 3586/5000 [06:07<00:58, 24.03it/s] [A
 [A
Training:  72%|███████▏  | 3600/5000 [06:16<00:58, 24.03it/s] [A
Training:  72%|███████▏  | 3601/5000 [06:16<05:08,  4.54it/s] [A
Training:  72%|███████▏  | 3617/5000 [06:17<03:34,  6.45it/s] [A
```

Iter 3600 Train Acc 0.9789 Test Acc 0.9758

```
Training:  73%|███████▎  | 3631/5000 [06:17<02:36,  8.76it/s] [A
Training:  73%|███████▎  | 3647/5000 [06:17<01:49, 12.39it/s] [A
Training:  73%|███████▎  | 3664/5000 [06:17<01:15, 17.60it/s] [A
Training:  74%|███████▎  | 3681/5000 [06:17<00:53, 24.47it/s] [A
Training:  74%|███████▍  | 3697/5000 [06:17<00:39, 32.59it/s] [A
 [A
Training:  74%|███████▍  | 3700/5000 [06:27<00:39, 32.59it/s] [A
Training:  74%|███████▍  | 3713/5000 [06:27<04:25,  4.85it/s] [A
```

Iter 3700 Train Acc 0.9771 Test Acc 0.9734

```
Training:  75%|███████▍  | 3730/5000 [06:27<03:02,  6.94it/s] [A
Training:  75%|███████▍  | 3747/5000 [06:27<02:07,  9.84it/s] [A
Training:  75%|███████▌  | 3764/5000 [06:27<01:29, 13.80it/s] [A
Training:  76%|███████▌  | 3781/5000 [06:27<01:03, 19.10it/s] [A
```

```
Training:  76%|███████▋  | 3798/5000 [06:28<00:46, 26.05it/s] [
  [A
Training:  76%|███████▋  | 3800/5000 [06:37<00:46, 26.05it/s] [
Training:  76%|███████▋  | 3814/5000 [06:37<04:04,  4.85it/s] [A
Training:  77%|███████▋  | 3831/5000 [06:37<02:49,  6.89it/s]

Iter 3800 Train Acc 0.9781 Test Acc 0.9753

  [A
Training:  77%|███████▋  | 3847/5000 [06:38<02:00,  9.55it/s] [A
Training:  77%|███████▋  | 3865/5000 [06:38<01:23, 13.66it/s] [
Training:  78%|███████▊  | 3883/5000 [06:38<00:58, 19.20it/s] [
Training:  78%|███████▊  | 3900/5000 [06:38<00:42, 26.03it/s] [
  [A
Training:  78%|███████▊  | 3900/5000 [06:48<00:42, 26.03it/s] [
Training:  78%|███████▊  | 3916/5000 [06:48<03:40,  4.92it/s] [A

Iter 3900 Train Acc 0.9776 Test Acc 0.9747


Training:  79%|███████▊  | 3933/5000 [06:48<02:33,  6.96it/s] [A
Training:  79%|███████▉  | 3949/5000 [06:48<01:49,  9.62it/s] [A
Training:  79%|███████▉  | 3966/5000 [06:48<01:16, 13.48it/s] [
Training:  80%|███████▉  | 3982/5000 [06:48<00:55, 18.35it/s] [
Training:  80%|███████▉  | 3997/5000 [06:48<00:41, 24.32it/s] [
  [A
Training:  80%|████████  | 4000/5000 [06:58<00:41, 24.32it/s] [
Training:  80%|████████  | 4012/5000 [06:58<03:27,  4.75it/s] [A

Iter 4000 Train Acc 0.9774 Test Acc 0.9746


Training:  80%|████████  | 4024/5000 [06:58<02:37,  6.21it/s] [A
Training:  81%|████████  | 4035/5000 [06:58<01:59,  8.05it/s] [A
Training:  81%|████████  | 4046/5000 [06:58<01:30, 10.59it/s] [
Training:  81%|████████  | 4057/5000 [06:58<01:07, 13.99it/s] [
Training:  81%|████████▏ | 4068/5000 [06:58<00:50, 18.45it/s]
Training:  82%|████████▏ | 4079/5000 [06:58<00:38, 24.04it/s]
Training:  82%|████████▏ | 4090/5000 [06:59<00:29, 30.75it/s]
  [A
Training:  82%|████████▏ | 4100/5000 [07:08<00:29, 30.75it/s]
Training:  82%|████████▏ | 4101/5000 [07:08<04:03,  3.69it/s]
Training:  82%|████████▏ | 4117/5000 [07:08<02:30,  5.85it/s]

Iter 4100 Train Acc 0.9791 Test Acc 0.9768


Training:  83%|████████▎ | 4130/5000 [07:08<01:45,  8.25it/s]
Training:  83%|████████▎ | 4147/5000 [07:08<01:07, 12.57it/s]
Training:  83%|████████▎ | 4164/5000 [07:08<00:45, 18.36it/s]
```

```
Training:  84%|████████▍ | 4180/5000 [07:08<00:32, 25.43it/s]
Training:  84%|████████▍ | 4197/5000 [07:08<00:22, 35.06it/s]
 [A
Training:  84%|████████▍ | 4200/5000 [07:18<00:22, 35.06it/s]
Training:  84%|████████▍ | 4213/5000 [07:18<02:42,  4.85it/s]

Iter 4200 Train Acc 0.9775 Test Acc 0.9767

Training:  85%|████████▍ | 4229/5000 [07:18<01:52,  6.86it/s]
Training:  85%|████████▍ | 4246/5000 [07:19<01:16,  9.82it/s]
Training:  85%|████████▌ | 4263/5000 [07:19<00:53, 13.86it/s]
Training:  86%|████████▌ | 4280/5000 [07:19<00:37, 19.29it/s]
Training:  86%|████████▌ | 4296/5000 [07:19<00:27, 25.94it/s]
 [A
Training:  86%|████████▌ | 4300/5000 [07:29<00:26, 25.94it/s]
Training:  86%|████████▌ | 4312/5000 [07:29<02:22,  4.82it/s]

Iter 4300 Train Acc 0.9747 Test Acc 0.9718

Training:  87%|████████▋ | 4329/5000 [07:29<01:37,  6.88it/s]
Training:  87%|████████▋ | 4345/5000 [07:29<01:08,  9.56it/s]
Training:  87%|████████▋ | 4362/5000 [07:29<00:47, 13.47it/s]
Training:  88%|████████▊ | 4378/5000 [07:29<00:33, 18.39it/s]
Training:  88%|████████▊ | 4395/5000 [07:29<00:23, 25.30it/s]
 [A
Training:  88%|████████▊ | 4400/5000 [07:39<00:23, 25.30it/s]
Training:  88%|████████▊ | 4411/5000 [07:39<02:02,  4.81it/s]
Training:  89%|████████▊ | 4428/5000 [07:39<01:23,  6.86it/s]

Iter 4400 Train Acc 0.9802 Test Acc 0.9753


Training:  89%|████████▉ | 4442/5000 [07:39<01:00,  9.18it/s]
Training:  89%|████████▉ | 4457/5000 [07:39<00:43, 12.60it/s]
Training:  89%|████████▉ | 4474/5000 [07:39<00:29, 17.86it/s]
Training:  90%|████████▉ | 4491/5000 [07:39<00:20, 24.79it/s]
 [A
Training:  90%|█████████ | 4500/5000 [07:50<00:20, 24.79it/s]
Training:  90%|█████████ | 4507/5000 [07:50<01:51,  4.43it/s]
Training:  90%|█████████ | 4524/5000 [07:50<01:14,  6.35it/s]

Iter 4500 Train Acc 0.9789 Test Acc 0.9745


Training:  91%|█████████ | 4540/5000 [07:50<00:52,  8.83it/s]
Training:  91%|█████████ | 4556/5000 [07:50<00:36, 12.24it/s]
Training:  91%|█████████▏| 4572/5000 [07:51<00:25, 16.85it/
Training:  92%|█████████▏| 4589/5000 [07:51<00:17, 23.37it/
 [A
Training:  92%|█████████▏| 4600/5000 [08:01<00:17, 23.37it/
```

```
Training:  92%|█████████▏| 4605/5000 [08:01<01:24,  4.68it/s
Training:  92%|█████████▏| 4622/5000 [08:01<00:56,  6.69it/s

Iter 4600 Train Acc 0.9767 Test Acc 0.9757


Training:  93%|█████████▎| 4637/5000 [08:01<00:39,  9.14it/s
Training:  93%|█████████▎| 4654/5000 [08:01<00:26, 12.96it/
Training:  93%|█████████▎| 4671/5000 [08:01<00:18, 18.10it/
Training:  94%|█████████▎| 4687/5000 [08:01<00:12, 24.46it/
 [A
Training:  94%|█████████▍| 4700/5000 [08:11<00:12, 24.46it/
Training:  94%|█████████▍| 4703/5000 [08:11<01:02,  4.75it/s
Training:  94%|█████████▍| 4720/5000 [08:11<00:41,  6.79it/s

Iter 4700 Train Acc 0.9781 Test Acc 0.9754


Training:  95%|█████████▍| 4737/5000 [08:11<00:27,  9.62it/s
Training:  95%|█████████▌| 4754/5000 [08:11<00:18, 13.48it/
Training:  95%|█████████▌| 4770/5000 [08:11<00:12, 18.34it/
Training:  96%|█████████▌| 4785/5000 [08:12<00:08, 24.31it/
 [A
Training:  96%|█████████▌| 4800/5000 [08:21<00:08, 24.31it/
Training:  96%|█████████▌| 4801/5000 [08:21<00:40,  4.91it/s
Training:  96%|█████████▋| 4813/5000 [08:21<00:29,  6.39it/s

Iter 4800 Train Acc 0.9801 Test Acc 0.9771


Training:  96%|█████████▋| 4824/5000 [08:21<00:21,  8.27it/s
Training:  97%|█████████▋| 4836/5000 [08:21<00:14, 11.08it/
Training:  97%|█████████▋| 4849/5000 [08:21<00:09, 15.19it/
Training:  97%|█████████▋| 4862/5000 [08:21<00:06, 20.58it/
Training:  98%|█████████▊| 4875/5000 [08:21<00:04, 27.35it/
Training:  98%|█████████▊| 4887/5000 [08:22<00:03, 34.91it/
 [A
Training:  98%|█████████▊| 4900/5000 [08:31<00:02, 34.91it/
Training:  98%|█████████▊| 4901/5000 [08:31<00:23,  4.13it/s
Training:  98%|█████████▊| 4915/5000 [08:31<00:14,  5.94it/s

Iter 4900 Train Acc 0.9798 Test Acc 0.9783


Training:  99%|█████████▊| 4930/5000 [08:32<00:08,  8.63it/s
Training:  99%|█████████▉| 4947/5000 [08:32<00:04, 12.81it/
Training:  99%|█████████▉| 4964/5000 [08:32<00:01, 18.40it/
Training: 100%|█████████▉| 4981/5000 [08:32<00:00, 25.71it
Training: 100%|█████████▉| 4997/5000 [08:32<00:00, 34.34it
```

```
[A
Training: 100%|██████████| 5000/5000 [08:42<00:00, 34.34it

Iter 5000 Train Acc 0.9803 Test Acc 0.9763
```

In [ ]:

```python
model.load_state_dict(torch.load("./model.pth"))

model.eval()
```

Out [14]:

```
CNN(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=9216, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
```

In [ ]:

```python
correct = 0
total = 0

with torch.no_grad():
    for image, label in test_loader:
        image = image.to(device)
        label = label.to(device)

        pred = model(image)
        _, pred = torch.max(pred, dim=1)

        total += label.shape[0]
        correct += int((pred == label).sum())

    print(f"Accuracy: {correct / total * 100:.2f}%")
```

```
Accuracy: 98.69%
```

Q2 Seeing Close or Seeing Far (CIFAR)

4 Points

Upload your copy of `part2.ipynb`. Make sure that it contains the outputs of your run.

▾ part_2.ipynb ⬇ Download

ECE 117 - Assignment 3: Part 2

The goal of this part of the assignment is to implement the adversarial examples attack, FGSM.

In [ ]:

```
import numpy as np
import torch
from torch import nn, utils
import torch.nn.functional as F
from torchvision import datasets, transforms
import matplotlib.pyplot as plt
```

In [ ]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Device: {device}")
```

```
Device: cuda
```

In [ ]:

```
transform = transforms.Compose([transforms.ToTensor()])

train_dataset = datasets.CIFAR10(root='./data', train=True,
download=True, transform=transform)
train_loader = utils.data.DataLoader(train_dataset,
batch_size=128, shuffle=True, num_workers=2)

test_dataset = datasets.CIFAR10(root='./data', train=False,
download=True, transform=transform)
test_loader = utils.data.DataLoader(test_dataset, batch_size=128,
shuffle=True, num_workers=2)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz tc

100%|██████████| 170498071/170498071 [00:13<00:00, 129

Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

In [ ]:

```
classes = ["plane", "car", "bird", "cat", "deer", "dog", "frog",
"horse", "ship", "truck"]
```

In [ ]:

```
class CNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(32, 32, kernel_size=3, stride=2, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(64, 64, kernel_size=3, stride=2, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.Flatten(),
            nn.Linear(128 * 4 * 4, 10))

    def forward(self, x):
        x = self.network(x)
        return x

    def size(self):
        parameters = self.parameters()
        size = 0
        for parameter in parameters:
            size += np.prod(parameter.shape)
        return size
```

In [ ]:

```
# Download pretrained CIFAR-10 model.
!pip install wget
import wget

weights_file = wget.download("https://github.com/kuanhenglin/ai-secur
workshop/blob/f08ced8a4afb7de1120bfdbf468888c7be10fdd8/cifar10_
raw=true")
```

```
Collecting wget
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) ...   [?25l  [?25hdone
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ...   [?25l  [?25hdone
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=965
```

```
  Stored in directory: /root/.cache/pip/wheels/8b/f1/7f/5c94f0a7a505ca1
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
```

In [ ]:

```
network = CNN()
network.load_state_dict(torch.load(weights_file,
map_location=device))
network.to(device)
network.eval()
```

Out [7]:

```
CNN(
  (network): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_
    (9): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (10): ReLU()
    (11): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track
    (12): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track
    (15): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (16): ReLU()
    (17): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, trac
    (18): Flatten(start_dim=1, end_dim=-1)
    (19): Linear(in_features=2048, out_features=10, bias=True)
  )
)
```

In [ ]:

```
@torch.no_grad()
def evaluate(loader, network):
    network.eval()
    accuracies = []
    for inputs, labels in loader:
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = network(inputs)
        accuracy = (torch.max(outputs, dim=1)[1] ==
labels).to(torch.float32).mean()  # accuracy
        accuracies.append(accuracy.cpu().numpy())
    return np.mean(accuracies)
```

In [ ]:

```
accuracy = evaluate(test_loader, network)
print(f"Test accuracy: {str(accuracy * 100):.6}%")
```

Test accuracy: 86.728%

Adversarial Attack

In [ ]:

```
def display_torch_image(image, label=None):
    if label is not None:
        plt.title(f"{classes[label]} ({label})")
    plt.axis("off")
    plt.imshow(torch.moveaxis(image, 0, -1).cpu(), vmin=0, vmax=1)
```

In [ ]:

```
def display_attacked(image, image_attacked, noise, label,
label_attacked):
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

    axes[0].axis("off")
    axes[0].set_title(f"{classes[label]} ({label})", fontsize=16)
    axes[0].text(33.25, 16.5, "$+$", fontsize=24)
    axes[0].imshow(torch.moveaxis(image, 0, -1).cpu(), vmin=0,
vmax=1)

    axes[1].axis("off")
    axes[1].set_title(f"noise (amplified 5x)", fontsize=16)
    axes[1].text(32.75, 16, "$=$", fontsize=24)
    axes[1].imshow(torch.moveaxis(noise, 0, -1).cpu(), vmin=0,
vmax=1)

    axes[2].axis("off")
    axes[2].set_title(f"{classes[label_attacked]} ({label_attacked})",
fontsize=16)
    axes[2].imshow(torch.moveaxis(image_attacked, 0, -1).cpu(),
vmin=0, vmax=1)
```

In [ ]:

```
def adversarial_attack(network, image, label, epsilon=0.1,
sign=True):
    # Your implementation here.
    # forward pass
    image.requires_grad = True  # we will be computing gradients
w.r.t. the image!
    output = network(image.unsqueeze(dim=0))[0]  # pass image
through network to get output
    prediction = output.max(dim=0)[1].cpu().numpy()
    loss = F.nll_loss(output, label)
```

```
    image_gradients = torch.autograd.grad(loss, image)[0]  # gradients of loss w.r.t.the image
    if sign:  # use sign of gradients (more stable, controlled magnitude)
        image_gradients = image_gradients.sign()

    # fast gradient sign attack
    image_attacked = (image + epsilon * image_gradients).clamp(0, 1)  # add noise to image
    output_attacked = network(image_attacked.unsqueeze(dim=0))[0]  # get model output of attacked
    prediction_attacked = output_attacked.max(dim=0)[1].cpu().numpy()  # get prediction

    display_attacked(image.detach(), image_attacked.detach(),
                     image_gradients.detach() * 0.5 * (5 * epsilon) + 0.5,
                     label=prediction, label_attacked=prediction_attacked)
```

In [ ]:

```
batch_data = next(iter(train_loader))
image = batch_data[0][0].to(device)
label = batch_data[1][0].to(device)
with torch.no_grad():
    prediction = network(image.unsqueeze(dim=0)).max(dim=1)[1][0].cpu().numpy()
    display_torch_image(image, label=prediction)
```

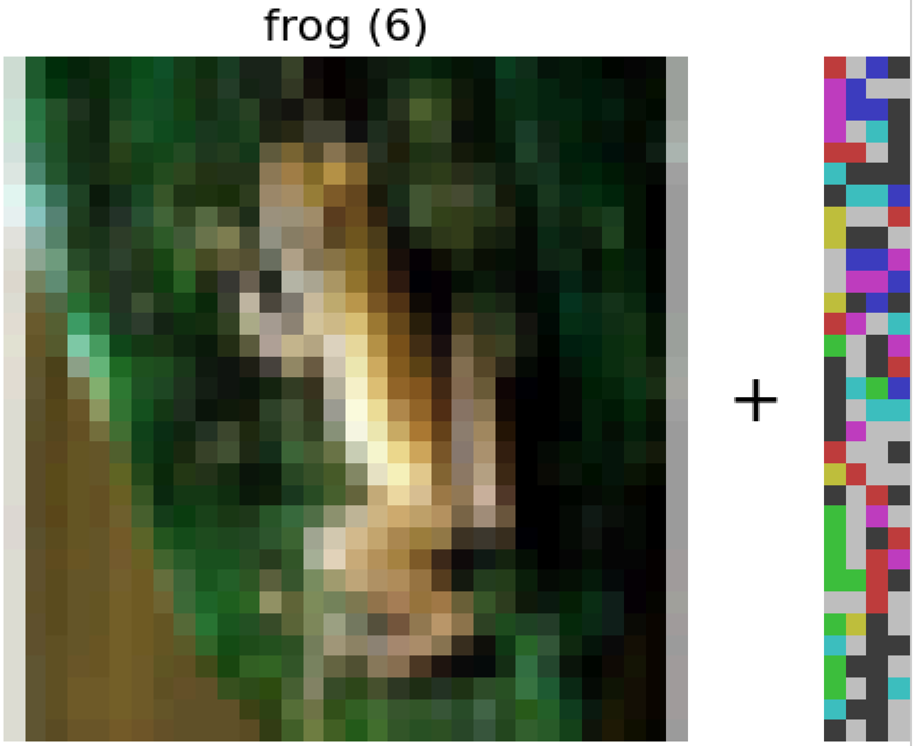
frog (6)

Fast Gradient Sign Attack (FGSM)

The goal of this part is to demonstrate a misclasification a baseline epsilon is provided but your goal is to provide the best hyperparameter.

In [ ]:

```
adversarial_attack(network, image, label, epsilon=0.1, sign=True)
```



Q3 Unlearning that Look

0 Points

Optional. Upload your copy of `part3.ipynb`. Make sure that it contains the outputs of your run.

▾ part_3.ipynb Download

ECE 117 - Assignment 3: Part 3

The goal of this part of the assignment is to implement machine unlearning via fine-tuning.

In [1]:

```
import os
import requests
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model, model_selection

import torch
from torch import nn
import torch.nn.functional as F
from torch import optim
from torch.utils.data import DataLoader

import torchvision
from torchvision import datasets, transforms
from torchvision.utils import make_grid
from torchvision.models import resnet18

import tqdm

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
print("Running on device:", DEVICE.upper())

RNG = torch.Generator().manual_seed(42)
```

```
Running on device: CUDA
```

In [2]:

```
normalize = transforms.Compose([transforms.ToTensor()])

train_set = torchvision.datasets.FashionMNIST(
    root="./data", train=True, download=True,
transform=normalize
)
train_loader = DataLoader(train_set, batch_size=128,
shuffle=True, num_workers=2)

held_out = torchvision.datasets.FashionMNIST(
    root="./data", train=False, download=True,
transform=normalize
)
test_set, val_set = torch.utils.data.random_split(held_out, [0.5,
```

```
0.5], generator=RNG)
test_loader = DataLoader(test_set, batch_size=128, shuffle=False,
num_workers=2)
val_loader = DataLoader(val_set, batch_size=128, shuffle=False,
num_workers=2)

forget_class = 0
forget_idx, retain_idx = [], []
for i, target in enumerate(train_set.targets):
  if target == forget_class:
    forget_idx.append(i)
  else:
    retain_idx.append(i)

forget_set = torch.utils.data.Subset(train_set, forget_idx)
retain_set = torch.utils.data.Subset(train_set, retain_idx)

forget_loader = torch.utils.data.DataLoader(
    forget_set, batch_size=128, shuffle=True, num_workers=2
)
retain_loader = torch.utils.data.DataLoader(
    retain_set, batch_size=128, shuffle=True, num_workers=2,
generator=RNG
)
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c

100%|██████████| 26421880/26421880 [00:02<00:00, 11699

Extracting ./data/FashionMNIST/raw/train-images-idx3-ubyte.gz to ./dat

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c

100%|██████████| 29515/29515 [00:00<00:00, 211831.00it/s

Extracting ./data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to ./data

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c

100%|██████████| 4422102/4422102 [00:01<00:00, 3924497

Extracting ./data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to ./data

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.c

100%|██████████| 5148/5148 [00:00<00:00, 20029941.55it/s
```

Extracting ./data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/

In [3]:

```
# This is provided as a baseline model but feel free to adjust this.
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 4 * 4, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 4 * 4)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [4]:

```
model = CNN().to(DEVICE)

i_max = 6400

criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001,
weight_decay=0.0001)
```

In [5]:

```
@torch.no_grad()
def get_accuracy(model, data_loader, device):
    correct = 0
    total = 0

    for inputs, labels in data_loader:
        inputs = inputs.to(DEVICE)
        labels = labels.to(DEVICE)

        outputs = model(inputs)
        _, predicted = torch.max(outputs, dim=1)

        total += labels.shape[0]
        correct += int((predicted == labels).sum())
```

```
    return correct / total
```

In [ ]:

```
# First, train a baseline FashionMNIST CNN

progress = tqdm.tqdm(total=i_max, desc="Training")

i = 0
while i < i_max:
    for inputs, labels in train_loader:
        # modify training loop
        model.train()

        inputs = inputs.to(DEVICE)
        labels = labels.to(DEVICE)

        outputs = model(inputs)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        i += 1
        progress.update(1)

        if i % 100 == 0:
            train_acc = get_accuracy(model, train_loader, DEVICE)
            test_acc = get_accuracy(model, test_loader, DEVICE)
            progress.write(f"Iter {i} Train Acc {train_acc:.4f} Test Acc
{test_acc:.4f}")

        if i >= i_max:
            break

torch.save(model.state_dict(), "./model.pth")
```

In [12]:

```
test_accuracy = get_accuracy(model, test_loader, DEVICE)
print(f"Test Accuracy: {test_accuracy}")
```

Test Accuracy: 0.8972

In [57]:

```
# Machine unlearning via fine-tuning
i_max = 500

progress = tqdm.tqdm(total=i_max, desc="Training")
```

```python
i = 0
while i < i_max:
    for inputs, labels in retain_set:
        # modify loop to fine-tune model
        model.train()

        inputs, labels = inputs.to(DEVICE), torch.tensor(labels).to(DEVICE)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs[0], labels)
        loss.backward()
        optimizer.step()

        i += 1
        progress.update(1)

        if i % 100 == 0:
            train_acc = get_accuracy(model, train_loader, DEVICE)
            test_acc = get_accuracy(model, test_loader, DEVICE)
            progress.write(f"Iter {i} Train Acc {train_acc:.4f} Test Acc {test_acc:.4f}")

        if i >= i_max:
            break

torch.save(model.state_dict(), "./model-unlearned.pth")
```

```
Training: 100%|██████████| 500/500 [02:09<00:00,  3.87it/s]
Training:  29%|███       | 144/500 [00:08<00:26, 13.25it/s]

Iter 100 Train Acc 0.7911 Test Acc 0.7760

Training:  50%|█████     | 249/500 [00:17<00:18, 13.38it/s]

Iter 200 Train Acc 0.7936 Test Acc 0.7806

Training:  72%|███████▏  | 360/500 [00:24<00:09, 14.89it/s]

Iter 300 Train Acc 0.7749 Test Acc 0.7638

Training:  94%|█████████▍| 471/500 [00:32<00:02, 14.39it/s]

Iter 400 Train Acc 0.7918 Test Acc 0.7806

Training: 100%|██████████| 500/500 [00:40<00:00, 14.39it/s]

Iter 500 Train Acc 0.7738 Test Acc 0.7604
```

In [58]:

```python
model_retain = CNN().to(DEVICE)

i_max = 500

criterion_retain = torch.nn.CrossEntropyLoss()
optimizer_retain = torch.optim.Adam(model.parameters(), lr=0.001, weight_decay=0.0001)
```

In [59]:

```python
# Train only purely the retain set to benchmark

progress = tqdm.tqdm(total=i_max, desc="Training")

i = 0
while i < i_max:
    for inputs, labels in retain_loader:
        model_retain.train()

        inputs = inputs.to(DEVICE)
        labels = labels.to(DEVICE)

        outputs = model_retain(inputs)
        loss = criterion_retain(outputs, labels)

        optimizer_retain.zero_grad()
        loss.backward()
        optimizer_retain.step()

        i += 1
        progress.update(1)

        if i % 100 == 0:
            train_acc = get_accuracy(model_retain, retain_loader, DEVICE)
            test_acc = get_accuracy(model, test_loader, DEVICE)
            progress.write(f"Iter {i} Train Acc {train_acc:.4f} Test Acc {test_acc:.4f}")

        if i >= i_max:
            break

torch.save(model.state_dict(), "./model-retain.pth")
```

```
Training: 100%|██████████| 500/500 [00:58<00:00,  8.60it/s]

Training:   0%|          | 1/500 [00:00<01:01,  8.15it/s]  [A
Training:   2%|▏         | 8/500 [00:00<00:13, 37.08it/s] [A
Training:   3%|▎         | 15/500 [00:00<00:10, 48.11it/s] [A
Training:   4%|▍         | 22/500 [00:00<00:09, 52.93it/s] [A
Training:   6%|▌         | 29/500 [00:00<00:08, 57.37it/s] [A
```

```
Training:   7%|▋         | 36/500 [00:00<00:07, 58.85it/s] [A
Training:   9%|▊         | 43/500 [00:00<00:07, 60.57it/s] [A
Training:  10%|█         | 50/500 [00:00<00:07, 61.09it/s] [A
Training:  11%|█▏        | 57/500 [00:01<00:07, 59.19it/s] [A
Training:  13%|█▎        | 64/500 [00:01<00:07, 59.89it/s] [A
Training:  14%|█▍        | 71/500 [00:01<00:06, 61.45it/s] [A
Training:  16%|█▌        | 78/500 [00:01<00:06, 61.72it/s] [A
Training:  17%|█▋        | 85/500 [00:01<00:06, 59.62it/s] [A
Training:  18%|█▊        | 92/500 [00:01<00:06, 62.03it/s] [A
Training:  20%|█▉        | 99/500 [00:01<00:06, 59.65it/s] [A
 [A
Training:  20%|██        | 100/500 [00:09<00:06, 59.65it/s] [A
Training:  21%|██        | 106/500 [00:09<02:11,  3.00it/s] [A
Training:  22%|██▏       | 112/500 [00:09<01:36,  4.02it/s] [A

Iter 100 Train Acc 0.0931 Test Acc 0.7604


Training:  24%|██▎       | 118/500 [00:09<01:11,  5.38it/s] [A
Training:  25%|██▌       | 126/500 [00:09<00:47,  7.88it/s] [A
Training:  27%|██▋       | 133/500 [00:09<00:34, 10.76it/s] [A
Training:  28%|██▊       | 140/500 [00:09<00:25, 14.23it/s] [A
Training:  29%|██▉       | 147/500 [00:09<00:18, 18.67it/s] [A
Training:  31%|███       | 154/500 [00:10<00:14, 23.55it/s] [A
Training:  32%|███▏      | 160/500 [00:10<00:12, 28.14it/s] [A
Training:  33%|███▎      | 167/500 [00:10<00:09, 33.54it/s] [A
Training:  35%|███▍      | 173/500 [00:10<00:08, 37.45it/s] [A
Training:  36%|███▌      | 179/500 [00:10<00:07, 41.15it/s] [A
Training:  37%|███▋      | 186/500 [00:10<00:06, 47.26it/s] [A
Training:  39%|███▊      | 193/500 [00:10<00:06, 49.62it/s] [A
 [A
Training:  40%|████      | 200/500 [00:18<00:06, 49.62it/s] [A
Training:  40%|████      | 201/500 [00:18<01:37,  3.07it/s] [A
Training:  42%|████▏     | 210/500 [00:18<01:02,  4.63it/s] [A

Iter 200 Train Acc 0.0931 Test Acc 0.7604


Training:  43%|████▎     | 217/500 [00:18<00:45,  6.28it/s] [A
Training:  45%|████▍     | 223/500 [00:18<00:34,  8.12it/s] [A
Training:  46%|████▌     | 229/500 [00:18<00:25, 10.56it/s] [A
Training:  47%|████▋     | 235/500 [00:18<00:19, 13.49it/s] [A
Training:  48%|████▊     | 242/500 [00:18<00:14, 17.84it/s] [A
Training:  50%|████▉     | 248/500 [00:18<00:11, 22.04it/s] [A
Training:  51%|█████     | 256/500 [00:18<00:08, 28.97it/s] [A
Training:  53%|█████▎    | 263/500 [00:19<00:06, 34.90it/s] [A
Training:  54%|█████▍    | 270/500 [00:19<00:05, 39.36it/s] [A
Training:  55%|█████▌    | 277/500 [00:19<00:04, 44.98it/s] [A
Training:  57%|█████▋    | 284/500 [00:19<00:04, 48.34it/s] [A
Training:  58%|█████▊    | 291/500 [00:19<00:03, 53.14it/s] [A
Training:  60%|█████▉    | 298/500 [00:19<00:03, 54.61it/s] [A
```

```
 [A
Training:  60%|████████████      | 300/500 [00:26<00:03, 54.61it/s] [A
Training:  61%|████████████      | 305/500 [00:26<00:58,  3.32it/s] [A
Training:  62%|████████████      | 311/500 [00:26<00:42,  4.44it/s] [A

Iter 300 Train Acc 0.0931 Test Acc 0.7604


Training:  63%|████████████      | 317/500 [00:26<00:30,  5.98it/s] [A
Training:  65%|█████████████     | 323/500 [00:26<00:22,  8.01it/s] [A
Training:  66%|█████████████     | 331/500 [00:26<00:14, 11.59it/s] [A
Training:  67%|█████████████     | 337/500 [00:26<00:11, 14.71it/s] [A
Training:  69%|██████████████    | 344/500 [00:27<00:07, 19.52it/s] [A
Training:  70%|██████████████    | 350/500 [00:27<00:06, 23.78it/s] [A
Training:  71%|██████████████    | 357/500 [00:27<00:04, 30.01it/s] [A
Training:  73%|██████████████    | 364/500 [00:27<00:03, 34.49it/s] [A
Training:  74%|██████████████    | 370/500 [00:27<00:03, 38.22it/s] [A
Training:  75%|███████████████   | 377/500 [00:27<00:02, 44.41it/s] [A
Training:  77%|███████████████   | 384/500 [00:27<00:02, 48.74it/s] [A
Training:  78%|███████████████   | 391/500 [00:27<00:02, 53.52it/s] [A
Training:  80%|███████████████   | 398/500 [00:27<00:01, 53.89it/s] [A
 [A
Training:  80%|████████████████  | 400/500 [00:36<00:01, 53.89it/s] [A
Training:  81%|████████████████  | 405/500 [00:36<00:35,  2.70it/s] [A
Training:  82%|████████████████  | 412/500 [00:36<00:23,  3.80it/s] [A

Iter 400 Train Acc 0.0931 Test Acc 0.7604


Training:  84%|████████████████  | 418/500 [00:36<00:16,  5.07it/s] [A
Training:  85%|████████████████  | 424/500 [00:36<00:11,  6.54it/s] [A
Training:  86%|█████████████████ | 431/500 [00:36<00:07,  9.17it/s] [A
Training:  87%|█████████████████ | 437/500 [00:36<00:05, 11.91it/s] [A
Training:  89%|█████████████████ | 445/500 [00:36<00:03, 16.67it/s] [A
Training:  91%|██████████████████| 453/500 [00:37<00:02, 22.03it/s] [A
Training:  92%|██████████████████| 460/500 [00:37<00:01, 27.51it/s]
Training:  93%|██████████████████| 467/500 [00:37<00:01, 32.36it/s]
Training:  95%|██████████████████| 473/500 [00:37<00:00, 36.62it/s]
Training:  96%|███████████████████| 480/500 [00:37<00:00, 41.00it/s]
Training:  98%|███████████████████| 488/500 [00:37<00:00, 46.75it/s]
Training:  99%|███████████████████| 496/500 [00:37<00:00, 51.31it/s]
 [A
Training: 100%|████████████████████| 500/500 [00:45<00:00, 51.31it/s]

Iter 500 Train Acc 0.0931 Test Acc 0.7604
```

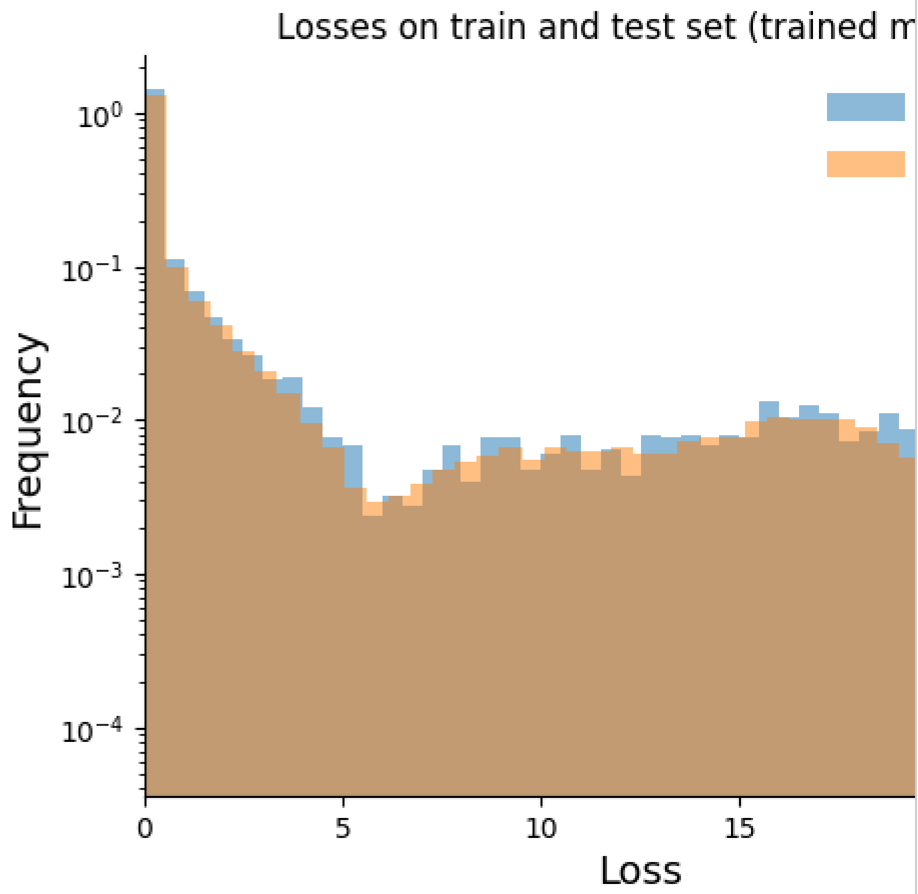
In [60]:

```
def compute_losses(net, loader):
    criterion = nn.CrossEntropyLoss(reduction="none")
    all_losses = []
```

```
    for inputs, targets in loader:
        inputs, targets = inputs.to(DEVICE), targets.to(DEVICE)

        logits = net(inputs)
        losses = criterion(logits, targets).numpy(force=True)
        for l in losses:
            all_losses.append(l)

    return np.array(all_losses)


train_losses = compute_losses(model, train_loader)
test_losses = compute_losses(model, test_loader)
```

In [61]:

```
# plot losses on train and test set
plt.title("Losses on train and test set (trained model)")
plt.hist(test_losses, density=True, alpha=0.5, bins=50, label="Test set")
plt.hist(train_losses, density=True, alpha=0.5, bins=50, label="Train set")
plt.xlabel("Loss", fontsize=14)
plt.ylabel("Frequency", fontsize=14)
plt.xlim((0, np.max(test_losses)))
plt.yscale("log")
plt.legend(frameon=False, fontsize=14)
ax = plt.gca()
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
plt.show()
```



In [62]:

```
def simple_mia(sample_loss, members, n_splits=10,
random_state=0):
    unique_members = np.unique(members)
    if not np.all(unique_members == np.array([0, 1])):
        raise ValueError("members should only have 0 and 1s")

    attack_model = linear_model.LogisticRegression()
    cv = model_selection.StratifiedShuffleSplit(
        n_splits=n_splits, random_state=random_state
    )
    return model_selection.cross_val_score(
        attack_model, sample_loss, members, cv=cv,
scoring="accuracy"
    )
```

In [63]:

```
forget_losses = compute_losses(model, forget_loader)

# Since we have more forget losses than test losses, sub-sample
them, to have a class-balanced dataset.
np.random.shuffle(forget_losses)
forget_losses = forget_losses[: len(test_losses)]

samples_mia = np.concatenate((test_losses,
forget_losses)).reshape((-1, 1))
```

```
labels_mia = [0] * len(test_losses) + [1] * len(forget_losses)

mia_scores = simple_mia(samples_mia, labels_mia)

print(
    f"The MIA has an accuracy of {mia_scores.mean():.3f} on forgotten vs unseen images"
)
```

The MIA has an accuracy of 0.919 on forgotten vs unseen images

In [64]:

```
# Benchmark model purely on the retain set.

ft_forget_losses = compute_losses(model_retain, forget_loader)
ft_test_losses = compute_losses(model_retain, test_loader)

# make sure we have a balanced dataset for the MIA
#assert len(ft_test_losses) == len(ft_forget_losses)

ft_samples_mia = np.concatenate((ft_test_losses, ft_forget_losses)).reshape((-1, 1))
labels_mia = [0] * len(ft_test_losses) + [1] * len(ft_forget_losses)
```

In [53]:

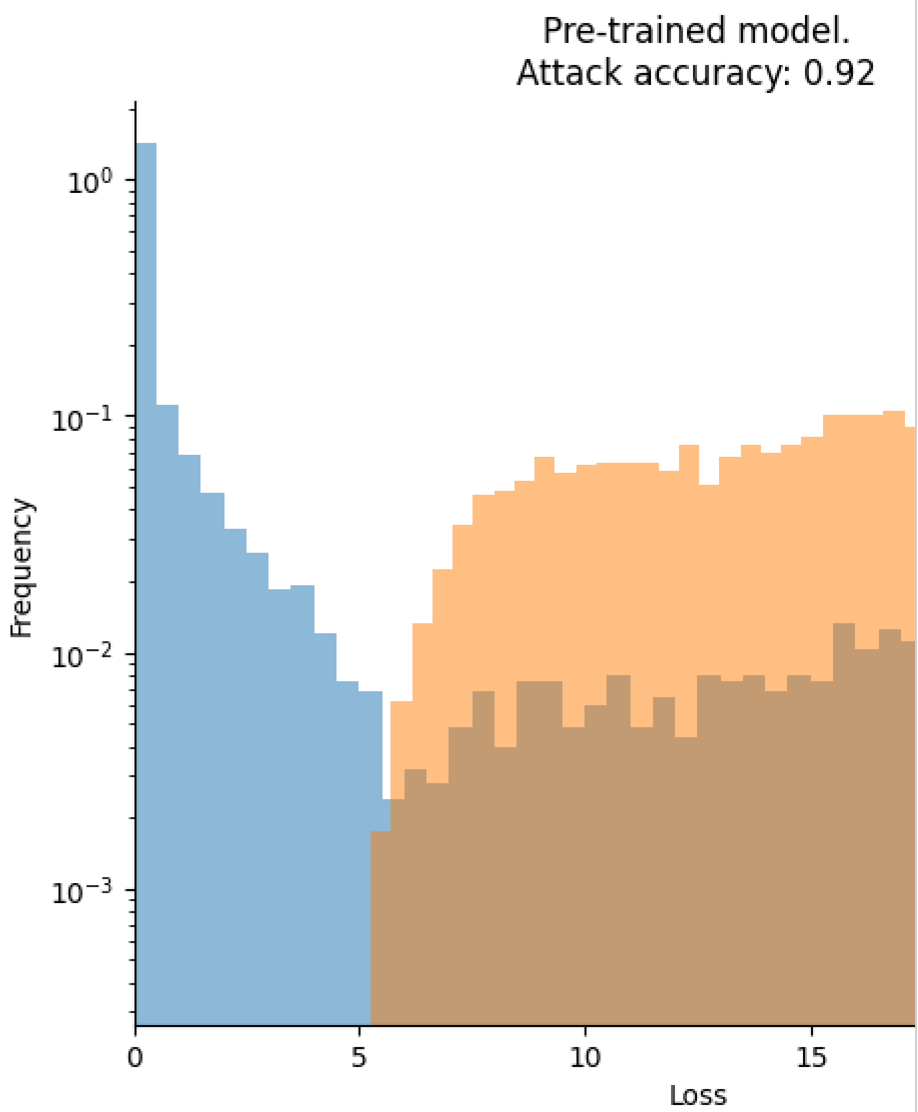
```
ft_mia_scores = simple_mia(ft_samples_mia, labels_mia)

print(
    f"The MIA has an accuracy of {ft_mia_scores.mean():.3f} on forgotten vs unseen images"
)
```

The MIA has an accuracy of 0.769 on forgotten vs unseen images

In [65]:

```
# Compare the results to determine the efficacy of the machine-unlearning implementation

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

ax1.set_title(f"Pre-trained model.\nAttack accuracy: {mia_scores.mean():0.2f}")
ax1.hist(test_losses, density=True, alpha=0.5, bins=50, label="Test set")
ax1.hist(forget_losses, density=True, alpha=0.5, bins=50, label="Forget set")

ax2.set_title(
    f"Unlearned by fine-tuning.\nAttack accuracy: {ft_mia_scores.mean():0.2f}"
```

```
)
ax2.hist(ft_test_losses, density=True, alpha=0.5, bins=50,
label="Test set")
ax2.hist(ft_forget_losses, density=True, alpha=0.5, bins=50,
label="Forget set")

ax1.set_xlabel("Loss")
ax2.set_xlabel("Loss")
ax1.set_ylabel("Frequency")
ax1.set_yscale("log")
ax2.set_yscale("log")
ax1.set_xlim((0, np.max(test_losses)))
ax2.set_xlim((0, np.max(test_losses)))
for ax in (ax1, ax2):
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
ax1.legend(frameon=False, fontsize=14)
plt.show()
```



Q4 Report
12 Points

Make sure to include `report.pdf` containing all of the responses to questions in the assignment. Make sure to respond to questions in parts 1, 2 & 3. Part 3 coding is optional but questions are required.

report.pdf Download

Your browser does not support PDF previews. You can download the file instead.