a3_part1_rotation

April 1, 2024

1 (Optional) Colab Setup

If you aren't using Colab, you can delete the following code cell. This is just to help students with mounting to Google Drive to access the other .py files and downloading the data, which is a little trickier on Colab than on your local machine using Jupyter.

[1]:
```
# you will be prompted with a window asking to grant permissions
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

[2]:
```
# fill in the path in your Google Drive in the string below. Note: do not␣
 ↪escape slashes or spaces
import os
datadir = "/content/assignment3"
if not os.path.exists(datadir):
  !ln -s "/content/drive/My Drive/CS444/assignment3_starter_sp24" $datadir #␣
 ↪TODO: Fill your Assignment 3 path
os.chdir(datadir)
!pwd
```

```
/content/drive/My Drive/CS444/assignment3_starter_sp24
```

#Data Setup

The first thing to do is implement a dataset class to load rotated CIFAR10 images with matching labels. Since there is already a CIFAR10 dataset class implemented in `torchvision`, we will extend this class and modify the `__getitem__` method appropriately to load rotated images.

Each rotation label should be an integer in the set {0, 1, 2, 3} which correspond to rotations of 0, 90, 180, or 270 degrees respectively.

[ ]:
```
import torch
import torchvision
import torchvision.transforms as transforms
import numpy as np
import random
```

```
def rotate_img(img, rot):
    if rot == 0: # 0 degrees rotation
        return img

    # TODO: Implement rotate_img() - return the rotated img
    elif rot == 1 or rot == 2 or rot == 3:
        return transforms.functional.rotate(img=img, angle=rot*90)

    else:
        raise ValueError('rotation should be 0, 90, 180, or 270 degrees')


class CIFAR10Rotation(torchvision.datasets.CIFAR10):

    def __init__(self, root, train, download, transform) -> None:
        super().__init__(root=root, train=train, download=download,␣
 ↪transform=transform)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, index: int):
        image, cls_label = super().__getitem__(index)

        # randomly select image rotation
        rotation_label = random.choice([0, 1, 2, 3])
        image_rotated = rotate_img(image, rotation_label)

        rotation_label = torch.tensor(rotation_label).long()
        return image, image_rotated, rotation_label, torch.tensor(cls_label).
 ↪long()
```

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

[ ]:
```
transform_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

transform_test = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
```

```
])

batch_size = 128

trainset = CIFAR10Rotation(root='./data', train=True, download=True,␣
 ↪transform=transform_train)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,␣
 ↪shuffle=True, num_workers=2)

testset = CIFAR10Rotation(root='./data', train=False, download=True,␣
 ↪transform=transform_test)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,␣
 ↪shuffle=False, num_workers=2)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

[ ]:
```
print(len(trainset))
print(len(testset))
```

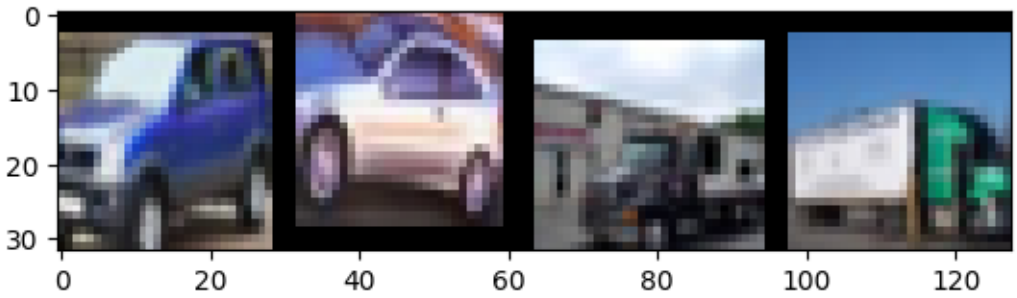
```
50000
10000
```

1.0.1 Show some example images and rotated images with labels:

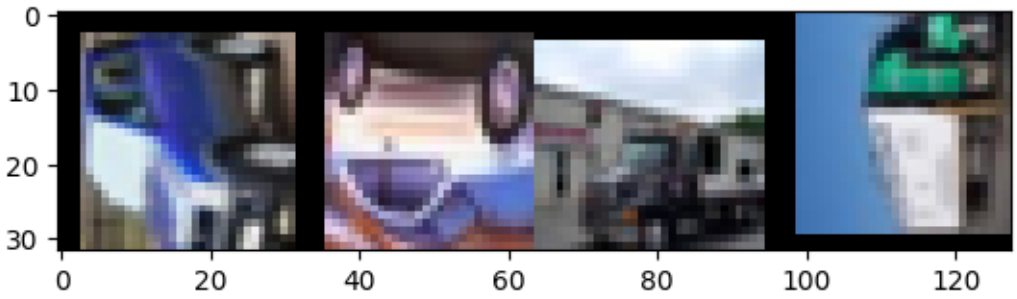
[ ]:
```
import matplotlib.pyplot as plt

classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',␣
 ↪'ship', 'truck')

rot_classes = ('0', '90', '180', '270')

def imshow(img):
    # unnormalize
    img = transforms.Normalize((0, 0, 0), (1/0.2023, 1/0.1994, 1/0.2010))(img)
    img = transforms.Normalize((-0.4914, -0.4822, -0.4465), (1, 1, 1))(img)
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

dataiter = iter(trainloader)
images, rot_images, rot_labels, labels = next(dataiter)

# print images and rotated images
img_grid = imshow(torchvision.utils.make_grid(images[:4], padding=0))
print('Class labels: ', ' '.join(f'{classes[labels[j]]:5s}' for j in range(4)))
img_grid = imshow(torchvision.utils.make_grid(rot_images[:4], padding=0))
```

```
print('Rotation labels: ', ' '.join(f'{rot_classes[rot_labels[j]]:5s}' for j in␣
 ↪range(4)))
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```




```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Class labels:  car   car   truck truck
```




```
Rotation labels:  90    180   0     90
```

2 Evaluation code

```
[ ]: import time

def run_test(net, testloader, criterion, task):
    correct = 0
    total = 0
    avg_test_loss = 0.0
    # since we're not training, we don't need to calculate the gradients for␣
 ↪our outputs
```

```
    with torch.no_grad():
        for images, images_rotated, labels, cls_labels in testloader:
            if task == 'rotation':
              images, labels = images_rotated.to(device), labels.to(device)
            elif task == 'classification':
              images, labels = images.to(device), cls_labels.to(device)
            # TODO: Calculate outputs by running images through the network
            # The class with the highest energy is what we choose as prediction
            outputs = net(images)
            predicted = torch.argmax(outputs, dim=1)

            # loss
            avg_test_loss += criterion(outputs, labels)  / len(testloader)

            # calculate accuracy
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    print('TESTING:')
    print(f'Accuracy of the network on the 10000 test images: {100 * correct /␣
↪total:.2f} %')
    print(f'Average loss on the 10000 test images: {avg_test_loss:.3f}')
```

[ ]:
```
def adjust_learning_rate(optimizer, epoch, init_lr, decay_epochs=30):
    """Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
    lr = init_lr * (0.1 ** (epoch // decay_epochs))
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr
```

3 Train a ResNet18 on the rotation task

3.0.1 In this section, we will train a ResNet18 model on the rotation task. The input is a rotated image and the model predicts the rotation label. See the Data Setup section for details.

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

[ ]:
```
import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet18

net = resnet18(num_classes=4)
```

```
net = net.to(device)
```

[ ]:
```
import torch.optim as optim

# TODO: Define criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
# Both the self-supervised rotation task and supervised CIFAR10 classification␣
 ↪are
# trained with the CrossEntropyLoss, so we can use the training loop code.

def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(net.parameters(), lr=init_lr, eps=1e-08,␣
 ↪weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(net.parameters(), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        net.train()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(trainloader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
 ↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
 ↪to(device)
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
```

```
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        net.eval()
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=50, decay_epochs=15, init_lr=0.001,␣
↪task='rotation')

# TODO: Save the model
torch.save(net.state_dict(), './net_pretrain.pth')
```

```
[1,   100] loss: 1.270 acc: 43.27 time: 2.56
[1,   200] loss: 1.130 acc: 50.38 time: 2.58
[1,   300] loss: 1.112 acc: 52.43 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 55.10 %
Average loss on the 10000 test images: 1.044
[2,   100] loss: 1.066 acc: 54.20 time: 2.55
[2,   200] loss: 1.037 acc: 55.68 time: 2.57
```

```
[2,   300] loss: 1.040 acc: 55.95 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 58.21 %
Average loss on the 10000 test images: 0.985
[3,   100] loss: 1.002 acc: 57.59 time: 2.53
[3,   200] loss: 0.980 acc: 59.20 time: 2.54
[3,   300] loss: 0.974 acc: 59.08 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 57.12 %
Average loss on the 10000 test images: 1.002
[4,   100] loss: 0.957 acc: 60.01 time: 2.56
[4,   200] loss: 0.960 acc: 60.00 time: 2.69
[4,   300] loss: 0.943 acc: 60.68 time: 2.46
TESTING:
Accuracy of the network on the 10000 test images: 60.72 %
Average loss on the 10000 test images: 0.943
[5,   100] loss: 0.942 acc: 60.16 time: 2.61
[5,   200] loss: 0.937 acc: 60.84 time: 2.69
[5,   300] loss: 0.936 acc: 60.56 time: 2.46
TESTING:
Accuracy of the network on the 10000 test images: 61.23 %
Average loss on the 10000 test images: 0.928
[6,   100] loss: 0.921 acc: 61.38 time: 2.54
[6,   200] loss: 0.924 acc: 61.65 time: 2.56
[6,   300] loss: 0.919 acc: 62.01 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 62.17 %
Average loss on the 10000 test images: 0.916
[7,   100] loss: 0.899 acc: 62.62 time: 2.55
[7,   200] loss: 0.898 acc: 63.35 time: 2.56
[7,   300] loss: 0.895 acc: 63.08 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 62.96 %
Average loss on the 10000 test images: 0.885
[8,   100] loss: 0.871 acc: 64.27 time: 2.60
[8,   200] loss: 0.886 acc: 63.36 time: 2.59
[8,   300] loss: 0.882 acc: 63.36 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 66.13 %
Average loss on the 10000 test images: 0.837
[9,   100] loss: 0.882 acc: 63.57 time: 2.57
[9,   200] loss: 0.851 acc: 64.80 time: 2.56
[9,   300] loss: 0.857 acc: 64.62 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 64.67 %
Average loss on the 10000 test images: 0.850
[10,   100] loss: 0.845 acc: 65.45 time: 2.60
[10,   200] loss: 0.856 acc: 64.58 time: 2.61
```

```
[10,   300] loss: 0.853 acc: 65.41 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 63.72 %
Average loss on the 10000 test images: 0.896
[11,   100] loss: 0.844 acc: 65.70 time: 2.57
[11,   200] loss: 0.834 acc: 66.18 time: 2.65
[11,   300] loss: 0.829 acc: 65.85 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 66.77 %
Average loss on the 10000 test images: 0.811
[12,   100] loss: 0.818 acc: 67.01 time: 2.56
[12,   200] loss: 0.824 acc: 66.35 time: 2.55
[12,   300] loss: 0.824 acc: 66.41 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 67.64 %
Average loss on the 10000 test images: 0.793
[13,   100] loss: 0.809 acc: 66.87 time: 2.55
[13,   200] loss: 0.801 acc: 67.35 time: 2.56
[13,   300] loss: 0.800 acc: 67.76 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 69.15 %
Average loss on the 10000 test images: 0.764
[14,   100] loss: 0.793 acc: 68.27 time: 2.55
[14,   200] loss: 0.795 acc: 67.62 time: 2.57
[14,   300] loss: 0.791 acc: 68.00 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 68.99 %
Average loss on the 10000 test images: 0.761
[15,   100] loss: 0.792 acc: 68.04 time: 2.57
[15,   200] loss: 0.774 acc: 68.98 time: 2.59
[15,   300] loss: 0.771 acc: 68.84 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 69.11 %
Average loss on the 10000 test images: 0.759
[16,   100] loss: 0.772 acc: 68.93 time: 2.54
[16,   200] loss: 0.773 acc: 69.02 time: 2.63
[16,   300] loss: 0.768 acc: 69.02 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 71.44 %
Average loss on the 10000 test images: 0.713
[17,   100] loss: 0.710 acc: 71.80 time: 2.53
[17,   200] loss: 0.699 acc: 72.41 time: 2.54
[17,   300] loss: 0.677 acc: 73.35 time: 2.41
TESTING:
Accuracy of the network on the 10000 test images: 74.71 %
Average loss on the 10000 test images: 0.647
[18,   100] loss: 0.679 acc: 73.02 time: 2.59
[18,   200] loss: 0.669 acc: 73.62 time: 2.60
```

```
[18,   300] loss: 0.668 acc: 73.66 time: 2.46
TESTING:
Accuracy of the network on the 10000 test images: 74.73 %
Average loss on the 10000 test images: 0.636
[19,   100] loss: 0.654 acc: 74.22 time: 2.56
[19,   200] loss: 0.655 acc: 74.12 time: 2.60
[19,   300] loss: 0.648 acc: 74.50 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 74.90 %
Average loss on the 10000 test images: 0.632
[20,   100] loss: 0.648 acc: 74.41 time: 2.54
[20,   200] loss: 0.645 acc: 74.59 time: 2.57
[20,   300] loss: 0.635 acc: 74.85 time: 2.47
TESTING:
Accuracy of the network on the 10000 test images: 75.71 %
Average loss on the 10000 test images: 0.619
[21,   100] loss: 0.638 acc: 74.77 time: 2.53
[21,   200] loss: 0.641 acc: 74.43 time: 2.59
[21,   300] loss: 0.639 acc: 74.73 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 75.94 %
Average loss on the 10000 test images: 0.613
[22,   100] loss: 0.631 acc: 75.69 time: 2.55
[22,   200] loss: 0.632 acc: 75.16 time: 2.56
[22,   300] loss: 0.632 acc: 75.48 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 76.02 %
Average loss on the 10000 test images: 0.608
[23,   100] loss: 0.623 acc: 75.49 time: 2.56
[23,   200] loss: 0.618 acc: 75.99 time: 2.57
[23,   300] loss: 0.623 acc: 75.70 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 76.23 %
Average loss on the 10000 test images: 0.599
[24,   100] loss: 0.626 acc: 75.48 time: 2.55
[24,   200] loss: 0.607 acc: 76.38 time: 2.58
[24,   300] loss: 0.612 acc: 76.12 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 76.37 %
Average loss on the 10000 test images: 0.597
[25,   100] loss: 0.606 acc: 76.49 time: 2.54
[25,   200] loss: 0.620 acc: 75.57 time: 2.63
[25,   300] loss: 0.607 acc: 76.39 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 76.72 %
Average loss on the 10000 test images: 0.591
[26,   100] loss: 0.612 acc: 75.84 time: 2.55
[26,   200] loss: 0.606 acc: 76.27 time: 2.57
```

```
[26,   300] loss: 0.607 acc: 76.30 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 77.34 %
Average loss on the 10000 test images: 0.580
[27,   100] loss: 0.597 acc: 76.31 time: 2.57
[27,   200] loss: 0.603 acc: 76.97 time: 2.56
[27,   300] loss: 0.609 acc: 76.64 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 77.21 %
Average loss on the 10000 test images: 0.577
[28,   100] loss: 0.591 acc: 76.99 time: 2.60
[28,   200] loss: 0.590 acc: 76.84 time: 2.61
[28,   300] loss: 0.603 acc: 76.41 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 76.95 %
Average loss on the 10000 test images: 0.588
[29,   100] loss: 0.594 acc: 77.02 time: 2.54
[29,   200] loss: 0.590 acc: 76.88 time: 2.56
[29,   300] loss: 0.593 acc: 77.08 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 77.75 %
Average loss on the 10000 test images: 0.566
[30,   100] loss: 0.598 acc: 76.76 time: 2.55
[30,   200] loss: 0.589 acc: 76.81 time: 2.64
[30,   300] loss: 0.588 acc: 77.19 time: 2.41
TESTING:
Accuracy of the network on the 10000 test images: 78.06 %
Average loss on the 10000 test images: 0.567
[31,   100] loss: 0.586 acc: 77.19 time: 2.54
[31,   200] loss: 0.587 acc: 77.32 time: 2.56
[31,   300] loss: 0.581 acc: 77.55 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 76.99 %
Average loss on the 10000 test images: 0.587
[32,   100] loss: 0.567 acc: 77.77 time: 2.56
[32,   200] loss: 0.581 acc: 77.20 time: 2.58
[32,   300] loss: 0.562 acc: 78.25 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.11 %
Average loss on the 10000 test images: 0.558
[33,   100] loss: 0.569 acc: 78.20 time: 2.54
[33,   200] loss: 0.572 acc: 77.70 time: 2.56
[33,   300] loss: 0.572 acc: 77.73 time: 2.41
TESTING:
Accuracy of the network on the 10000 test images: 77.94 %
Average loss on the 10000 test images: 0.553
[34,   100] loss: 0.566 acc: 78.12 time: 2.57
[34,   200] loss: 0.571 acc: 77.94 time: 2.64
```

```
[34,   300] loss: 0.562 acc: 78.11 time: 2.49
TESTING:
Accuracy of the network on the 10000 test images: 78.19 %
Average loss on the 10000 test images: 0.551
[35,   100] loss: 0.563 acc: 78.19 time: 2.59
[35,   200] loss: 0.566 acc: 78.05 time: 2.59
[35,   300] loss: 0.557 acc: 78.33 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.02 %
Average loss on the 10000 test images: 0.555
[36,   100] loss: 0.553 acc: 78.59 time: 5.45
[36,   200] loss: 0.563 acc: 78.20 time: 2.53
[36,   300] loss: 0.564 acc: 77.77 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.02 %
Average loss on the 10000 test images: 0.553
[37,   100] loss: 0.556 acc: 78.54 time: 2.55
[37,   200] loss: 0.561 acc: 78.43 time: 2.59
[37,   300] loss: 0.558 acc: 78.16 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 78.13 %
Average loss on the 10000 test images: 0.552
[38,   100] loss: 0.569 acc: 77.87 time: 2.56
[38,   200] loss: 0.566 acc: 77.88 time: 2.59
[38,   300] loss: 0.548 acc: 78.62 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.10 %
Average loss on the 10000 test images: 0.549
[39,   100] loss: 0.563 acc: 78.52 time: 2.53
[39,   200] loss: 0.554 acc: 78.52 time: 2.56
[39,   300] loss: 0.562 acc: 78.14 time: 2.41
TESTING:
Accuracy of the network on the 10000 test images: 78.48 %
Average loss on the 10000 test images: 0.543
[40,   100] loss: 0.554 acc: 78.75 time: 2.56
[40,   200] loss: 0.561 acc: 78.04 time: 2.61
[40,   300] loss: 0.560 acc: 78.69 time: 2.46
TESTING:
Accuracy of the network on the 10000 test images: 78.28 %
Average loss on the 10000 test images: 0.548
[41,   100] loss: 0.563 acc: 78.47 time: 2.57
[41,   200] loss: 0.562 acc: 77.81 time: 2.58
[41,   300] loss: 0.552 acc: 78.58 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.45 %
Average loss on the 10000 test images: 0.542
[42,   100] loss: 0.551 acc: 78.52 time: 2.55
[42,   200] loss: 0.553 acc: 78.59 time: 2.56
```

```
[42,   300] loss: 0.563 acc: 78.06 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.52 %
Average loss on the 10000 test images: 0.543
[43,   100] loss: 0.548 acc: 78.75 time: 2.54
[43,   200] loss: 0.546 acc: 78.98 time: 2.56
[43,   300] loss: 0.565 acc: 78.09 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.65 %
Average loss on the 10000 test images: 0.548
[44,   100] loss: 0.558 acc: 78.73 time: 2.57
[44,   200] loss: 0.560 acc: 78.21 time: 2.56
[44,   300] loss: 0.554 acc: 78.54 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 78.60 %
Average loss on the 10000 test images: 0.544
[45,   100] loss: 0.561 acc: 78.45 time: 2.53
[45,   200] loss: 0.559 acc: 78.09 time: 2.57
[45,   300] loss: 0.552 acc: 79.05 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.29 %
Average loss on the 10000 test images: 0.542
[46,   100] loss: 0.553 acc: 78.08 time: 2.54
[46,   200] loss: 0.552 acc: 78.79 time: 2.62
[46,   300] loss: 0.549 acc: 78.94 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.65 %
Average loss on the 10000 test images: 0.538
[47,   100] loss: 0.552 acc: 78.61 time: 2.57
[47,   200] loss: 0.560 acc: 78.26 time: 2.60
[47,   300] loss: 0.553 acc: 78.40 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.56 %
Average loss on the 10000 test images: 0.541
[48,   100] loss: 0.543 acc: 78.91 time: 2.55
[48,   200] loss: 0.556 acc: 79.06 time: 2.57
[48,   300] loss: 0.544 acc: 78.85 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 79.10 %
Average loss on the 10000 test images: 0.536
[49,   100] loss: 0.550 acc: 78.73 time: 2.56
[49,   200] loss: 0.546 acc: 78.70 time: 2.59
[49,   300] loss: 0.542 acc: 79.30 time: 2.47
TESTING:
Accuracy of the network on the 10000 test images: 78.45 %
Average loss on the 10000 test images: 0.546
[50,   100] loss: 0.560 acc: 77.79 time: 2.55
[50,   200] loss: 0.558 acc: 78.49 time: 2.57
```

```
[50,   300] loss: 0.547 acc: 78.96 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.85 %
Average loss on the 10000 test images: 0.535
[51,   100] loss: 0.550 acc: 78.88 time: 2.54
[51,   200] loss: 0.551 acc: 78.61 time: 2.59
[51,   300] loss: 0.557 acc: 78.65 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.51 %
Average loss on the 10000 test images: 0.541
[52,   100] loss: 0.555 acc: 78.45 time: 2.55
[52,   200] loss: 0.562 acc: 78.38 time: 2.60
[52,   300] loss: 0.538 acc: 78.82 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 79.08 %
Average loss on the 10000 test images: 0.533
[53,   100] loss: 0.554 acc: 78.45 time: 2.55
[53,   200] loss: 0.551 acc: 78.69 time: 2.57
[53,   300] loss: 0.546 acc: 78.94 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.38 %
Average loss on the 10000 test images: 0.542
[54,   100] loss: 0.555 acc: 78.70 time: 2.55
[54,   200] loss: 0.551 acc: 78.66 time: 2.59
[54,   300] loss: 0.554 acc: 78.52 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.67 %
Average loss on the 10000 test images: 0.537
[55,   100] loss: 0.543 acc: 78.85 time: 2.56
[55,   200] loss: 0.561 acc: 78.30 time: 2.57
[55,   300] loss: 0.544 acc: 78.95 time: 2.48
TESTING:
Accuracy of the network on the 10000 test images: 78.59 %
Average loss on the 10000 test images: 0.541
[56,   100] loss: 0.556 acc: 78.59 time: 2.55
[56,   200] loss: 0.543 acc: 78.91 time: 2.61
[56,   300] loss: 0.557 acc: 78.38 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 78.55 %
Average loss on the 10000 test images: 0.540
[57,   100] loss: 0.551 acc: 78.27 time: 2.58
[57,   200] loss: 0.542 acc: 79.38 time: 2.61
[57,   300] loss: 0.551 acc: 78.53 time: 2.44
TESTING:
Accuracy of the network on the 10000 test images: 78.60 %
Average loss on the 10000 test images: 0.538
[58,   100] loss: 0.541 acc: 78.98 time: 2.57
[58,   200] loss: 0.552 acc: 78.57 time: 2.58
```

```
[58,   300] loss: 0.548 acc: 78.77 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.74 %
Average loss on the 10000 test images: 0.537
[59,   100] loss: 0.540 acc: 79.45 time: 2.55
[59,   200] loss: 0.548 acc: 78.47 time: 2.83
[59,   300] loss: 0.559 acc: 78.34 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.96 %
Average loss on the 10000 test images: 0.537
[60,   100] loss: 0.553 acc: 78.48 time: 2.55
[60,   200] loss: 0.546 acc: 78.55 time: 2.56
[60,   300] loss: 0.544 acc: 79.03 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 79.32 %
Average loss on the 10000 test images: 0.537
[61,   100] loss: 0.544 acc: 78.95 time: 2.58
[61,   200] loss: 0.545 acc: 79.05 time: 2.77
[61,   300] loss: 0.560 acc: 78.15 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.98 %
Average loss on the 10000 test images: 0.535
[62,   100] loss: 0.547 acc: 78.91 time: 2.55
[62,   200] loss: 0.543 acc: 78.96 time: 2.58
[62,   300] loss: 0.550 acc: 78.66 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.82 %
Average loss on the 10000 test images: 0.539
[63,   100] loss: 0.566 acc: 77.98 time: 2.53
[63,   200] loss: 0.542 acc: 79.35 time: 2.57
[63,   300] loss: 0.556 acc: 78.20 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 78.47 %
Average loss on the 10000 test images: 0.544
[64,   100] loss: 0.548 acc: 78.64 time: 2.55
[64,   200] loss: 0.559 acc: 78.50 time: 2.60
[64,   300] loss: 0.540 acc: 78.78 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.62 %
Average loss on the 10000 test images: 0.546
[65,   100] loss: 0.560 acc: 78.18 time: 2.54
[65,   200] loss: 0.549 acc: 78.86 time: 2.57
[65,   300] loss: 0.549 acc: 78.92 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.74 %
Average loss on the 10000 test images: 0.538
[66,   100] loss: 0.547 acc: 79.05 time: 2.56
[66,   200] loss: 0.552 acc: 78.72 time: 2.60
```

```
[66,   300] loss: 0.555 acc: 78.32 time: 2.46
TESTING:
Accuracy of the network on the 10000 test images: 78.81 %
Average loss on the 10000 test images: 0.542
[67,   100] loss: 0.555 acc: 78.82 time: 2.56
[67,   200] loss: 0.550 acc: 78.54 time: 2.57
[67,   300] loss: 0.546 acc: 78.82 time: 2.45
TESTING:
Accuracy of the network on the 10000 test images: 78.12 %
Average loss on the 10000 test images: 0.544
[68,   100] loss: 0.550 acc: 78.93 time: 2.56
[68,   200] loss: 0.539 acc: 78.91 time: 2.59
[68,   300] loss: 0.555 acc: 78.52 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.68 %
Average loss on the 10000 test images: 0.543
[69,   100] loss: 0.550 acc: 78.50 time: 2.55
[69,   200] loss: 0.539 acc: 79.43 time: 2.57
[69,   300] loss: 0.546 acc: 79.02 time: 2.43
TESTING:
Accuracy of the network on the 10000 test images: 78.83 %
Average loss on the 10000 test images: 0.538
[70,   100] loss: 0.544 acc: 78.79 time: 2.56
[70,   200] loss: 0.547 acc: 78.84 time: 2.59
[70,   300] loss: 0.556 acc: 78.46 time: 2.42
TESTING:
Accuracy of the network on the 10000 test images: 78.92 %
Average loss on the 10000 test images: 0.537
Finished Training
```

4 Fine-tuning on the pre-trained model

4.0.1 In this section, we will load the pre-trained ResNet18 model and fine-tune on the classification task. We will freeze all previous layers except for the 'layer4' block and 'fc' layer.

[ ]:
```
import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet18

import torch.optim as optim
```

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

[ ]:
```
import torch.nn as nn
import torch.nn.functional as F

# TODO: Load the pre-trained ResNet18 model
net = resnet18(num_classes=10).to(device)
net.load_state_dict(torch.load('./net_pretrain.pt'))
# net.train()
# net = torch.load('net_pretrain.pt', map_location=torch.device(device))
print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
```

```
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:
```python
# TODO: Freeze all previous layers; only keep the 'layer4' block and 'fc' layer␣
↪trainable

## unfreeze layer4 and fc
for name, param in net.named_parameters():
    if "layer4" in name:
        param.requires_grad = True
    elif "fc" in name:
        param.requires_grad = True
```

```
    else:
        param.requires_grad = False

print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
```

```
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:
```python
# Print all the trainable parameters
params_to_update = net.parameters()
print("Params to learn:")
params_to_update = []
for name,param in net.named_parameters():
    if param.requires_grad == True:
        params_to_update.append(param)
        print("\t",name)
```

```
Params to learn:
         layer4.0.conv1.weight
         layer4.0.bn1.weight
         layer4.0.bn1.bias
         layer4.0.conv2.weight
         layer4.0.bn2.weight
```

```
layer4.0.bn2.bias
layer4.0.downsample.0.weight
layer4.0.downsample.1.weight
layer4.0.downsample.1.bias
layer4.1.conv1.weight
layer4.1.bn1.weight
layer4.1.bn1.bias
layer4.1.conv2.weight
layer4.1.bn2.weight
layer4.1.bn2.bias
fc.weight
fc.bias
```

[ ]:
```
# TODO: Define criterion and optimizer
# Note that your optimizer only needs to update the parameters that are␣
 ↪trainable.
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
# Both the self-supervised rotation task and supervised CIFAR10 classification␣
 ↪are
# trained with the CrossEntropyLoss, so we can use the training loop code.

def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, eps=1e-08, weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(trainloader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
 ↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
 ↪to(device)
```

```
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=20, decay_epochs=10, init_lr=0.001,␣
↪task='classification')
torch.save(net, 'finetune_pretrain.pt')
```

```
[1,   100] loss: 1.655 acc: 40.69 time: 2.71
[1,   200] loss: 1.302 acc: 52.20 time: 2.56
```

```
[1,   300] loss: 1.201 acc: 56.54 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 59.82 %
Average loss on the 10000 test images: 1.133
[2,   100] loss: 1.142 acc: 58.98 time: 2.69
[2,   200] loss: 1.124 acc: 59.98 time: 2.58
[2,   300] loss: 1.112 acc: 59.71 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 61.35 %
Average loss on the 10000 test images: 1.106
[3,   100] loss: 1.089 acc: 61.25 time: 2.65
[3,   200] loss: 1.081 acc: 61.72 time: 2.52
[3,   300] loss: 1.057 acc: 62.12 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 63.07 %
Average loss on the 10000 test images: 1.043
[4,   100] loss: 1.043 acc: 62.61 time: 2.70
[4,   200] loss: 1.048 acc: 62.52 time: 2.54
[4,   300] loss: 1.047 acc: 62.92 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 64.18 %
Average loss on the 10000 test images: 1.021
[5,   100] loss: 1.018 acc: 63.93 time: 2.69
[5,   200] loss: 1.033 acc: 62.82 time: 2.56
[5,   300] loss: 1.034 acc: 62.99 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 63.90 %
Average loss on the 10000 test images: 1.028
[6,   100] loss: 1.018 acc: 63.63 time: 2.68
[6,   200] loss: 1.007 acc: 64.51 time: 2.59
[6,   300] loss: 1.009 acc: 64.10 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 64.75 %
Average loss on the 10000 test images: 0.999
[7,   100] loss: 0.988 acc: 64.62 time: 2.69
[7,   200] loss: 0.988 acc: 64.59 time: 2.58
[7,   300] loss: 1.003 acc: 64.37 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 65.01 %
Average loss on the 10000 test images: 0.995
[8,   100] loss: 1.001 acc: 64.27 time: 2.72
[8,   200] loss: 1.001 acc: 64.52 time: 2.58
[8,   300] loss: 0.990 acc: 64.52 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 65.33 %
Average loss on the 10000 test images: 0.990
[9,   100] loss: 0.981 acc: 65.22 time: 2.69
[9,   200] loss: 0.969 acc: 65.41 time: 2.61
```

```
[9,   300] loss: 0.976 acc: 65.10 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 65.05 %
Average loss on the 10000 test images: 0.989
[10,   100] loss: 0.966 acc: 65.51 time: 2.68
[10,   200] loss: 0.973 acc: 65.18 time: 2.56
[10,   300] loss: 0.975 acc: 65.95 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 65.17 %
Average loss on the 10000 test images: 0.992
[11,   100] loss: 0.956 acc: 66.09 time: 2.68
[11,   200] loss: 0.973 acc: 65.09 time: 2.58
[11,   300] loss: 0.955 acc: 65.61 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 65.97 %
Average loss on the 10000 test images: 0.972
[12,   100] loss: 0.909 acc: 67.45 time: 2.71
[12,   200] loss: 0.918 acc: 66.78 time: 2.57
[12,   300] loss: 0.919 acc: 67.26 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 66.96 %
Average loss on the 10000 test images: 0.938
[13,   100] loss: 0.912 acc: 67.39 time: 2.70
[13,   200] loss: 0.900 acc: 68.09 time: 2.58
[13,   300] loss: 0.889 acc: 68.25 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 67.13 %
Average loss on the 10000 test images: 0.929
[14,   100] loss: 0.885 acc: 68.97 time: 2.67
[14,   200] loss: 0.892 acc: 68.49 time: 2.56
[14,   300] loss: 0.891 acc: 68.77 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 67.65 %
Average loss on the 10000 test images: 0.922
[15,   100] loss: 0.882 acc: 68.65 time: 2.67
[15,   200] loss: 0.868 acc: 69.35 time: 2.56
[15,   300] loss: 0.901 acc: 68.12 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 67.67 %
Average loss on the 10000 test images: 0.922
[16,   100] loss: 0.866 acc: 69.30 time: 2.69
[16,   200] loss: 0.884 acc: 68.78 time: 2.57
[16,   300] loss: 0.878 acc: 68.52 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 67.96 %
Average loss on the 10000 test images: 0.914
[17,   100] loss: 0.879 acc: 68.78 time: 2.68
[17,   200] loss: 0.873 acc: 69.23 time: 2.60
```

```
[17,   300] loss: 0.876 acc: 69.29 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 68.01 %
Average loss on the 10000 test images: 0.911
[18,   100] loss: 0.872 acc: 69.16 time: 2.70
[18,   200] loss: 0.874 acc: 69.02 time: 2.57
[18,   300] loss: 0.869 acc: 69.30 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 67.93 %
Average loss on the 10000 test images: 0.910
[19,   100] loss: 0.873 acc: 69.38 time: 2.67
[19,   200] loss: 0.860 acc: 69.38 time: 2.57
[19,   300] loss: 0.863 acc: 69.60 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 68.50 %
Average loss on the 10000 test images: 0.906
[20,   100] loss: 0.864 acc: 69.25 time: 2.69
[20,   200] loss: 0.867 acc: 69.18 time: 2.71
[20,   300] loss: 0.850 acc: 69.71 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 68.16 %
Average loss on the 10000 test images: 0.908
Finished Training
```

5 Fine-tuning on the randomly initialized model

5.0.1 In this section, we will randomly initialize a ResNet18 model and fine-tune on the classification task. We will freeze all previous layers except for the 'layer4' block and 'fc' layer.

[ ]:
```python
import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet18

# TODO: Randomly initialize a ResNet18 model
net = resnet18(num_classes=10)
net = net.to(device)
```

[ ]:
```python
# TODO: Freeze all previous layers; only keep the 'layer4' block and 'fc' layer␣
↪trainable

## change last layer to have 10 output features
net.fc = nn.Linear(in_features=512, out_features=10, bias=True).to(device)
nn.init.normal_(net.fc.weight)

## unfreeze layer4 and fc
```

```
net.eval()
for name, param in net.named_parameters():
    param.requires_grad = False
    if "layer4" in name:
        param.requires_grad = True
    elif "fc" in name:
        param.requires_grad = True

print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
```

```
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:
```python
# Print all the trainable parameters
params_to_update = net.parameters()
print("Params to learn:")
params_to_update = []
for name,param in net.named_parameters():
    if param.requires_grad == True:
        params_to_update.append(param)
        print("\t",name)
```

```
Params to learn:
```

```
layer4.0.conv1.weight
layer4.0.bn1.weight
layer4.0.bn1.bias
layer4.0.conv2.weight
layer4.0.bn2.weight
layer4.0.bn2.bias
layer4.0.downsample.0.weight
layer4.0.downsample.1.weight
layer4.0.downsample.1.bias
layer4.1.conv1.weight
layer4.1.bn1.weight
layer4.1.bn1.bias
layer4.1.conv2.weight
layer4.1.bn2.weight
layer4.1.bn2.bias
fc.weight
fc.bias
```

[ ]:
```
# TODO: Define criterion and optimizer
# Note that your optimizer only needs to update the parameters that are␣
 ↪trainable.
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
# Both the self-supervised rotation task and supervised CIFAR10 classification␣
 ↪are
# trained with the CrossEntropyLoss, so we can use the training loop code.

def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, eps=1e-08, weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(trainloader, 0):
```

```
            # TODO: Set the data to the correct device; Different task will use␣
↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
↪to(device)
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        # adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=30, decay_epochs=10, init_lr=0.001,␣
  ↪task='classification')
torch.save(net, 'finetune_random.pt')
```

```
[1,   100] loss: 6.794 acc: 16.42 time: 2.66
[1,   200] loss: 2.441 acc: 22.82 time: 2.58
[1,   300] loss: 2.242 acc: 25.57 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 32.59 %
Average loss on the 10000 test images: 1.968
[2,   100] loss: 2.070 acc: 29.17 time: 2.72
[2,   200] loss: 2.005 acc: 30.40 time: 2.57
[2,   300] loss: 1.970 acc: 30.75 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 32.58 %
Average loss on the 10000 test images: 1.896
[3,   100] loss: 1.907 acc: 32.06 time: 2.71
[3,   200] loss: 1.886 acc: 33.33 time: 2.57
[3,   300] loss: 1.898 acc: 32.41 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 36.73 %
Average loss on the 10000 test images: 1.749
[4,   100] loss: 1.880 acc: 33.17 time: 2.67
[4,   200] loss: 1.839 acc: 34.17 time: 2.57
[4,   300] loss: 1.817 acc: 34.26 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 35.93 %
Average loss on the 10000 test images: 1.783
[5,   100] loss: 1.810 acc: 34.85 time: 2.69
[5,   200] loss: 1.818 acc: 34.63 time: 2.58
[5,   300] loss: 1.785 acc: 35.87 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 36.23 %
Average loss on the 10000 test images: 1.748
[6,   100] loss: 1.797 acc: 35.63 time: 2.68
[6,   200] loss: 1.780 acc: 35.90 time: 2.58
[6,   300] loss: 1.776 acc: 35.63 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 40.40 %
Average loss on the 10000 test images: 1.649
[7,   100] loss: 1.770 acc: 35.61 time: 2.68
[7,   200] loss: 1.762 acc: 36.62 time: 2.59
[7,   300] loss: 1.749 acc: 36.96 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 40.01 %
Average loss on the 10000 test images: 1.666
[8,   100] loss: 1.732 acc: 37.48 time: 2.71
```

```
[8,   200] loss: 1.761 acc: 36.02 time: 2.57
[8,   300] loss: 1.748 acc: 36.80 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 38.49 %
Average loss on the 10000 test images: 1.698
[9,   100] loss: 1.732 acc: 37.40 time: 2.68
[9,   200] loss: 1.732 acc: 37.90 time: 2.56
[9,   300] loss: 1.743 acc: 37.47 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 38.79 %
Average loss on the 10000 test images: 1.691
[10,   100] loss: 1.747 acc: 36.83 time: 2.68
[10,   200] loss: 1.724 acc: 37.93 time: 2.57
[10,   300] loss: 1.713 acc: 37.63 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 40.06 %
Average loss on the 10000 test images: 1.661
[11,   100] loss: 1.716 acc: 37.65 time: 2.70
[11,   200] loss: 1.717 acc: 37.66 time: 2.56
[11,   300] loss: 1.721 acc: 37.85 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 42.64 %
Average loss on the 10000 test images: 1.602
[12,   100] loss: 1.696 acc: 38.41 time: 2.67
[12,   200] loss: 1.710 acc: 38.01 time: 2.56
[12,   300] loss: 1.700 acc: 38.86 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 41.40 %
Average loss on the 10000 test images: 1.625
[13,   100] loss: 1.679 acc: 39.30 time: 2.71
[13,   200] loss: 1.696 acc: 38.91 time: 2.58
[13,   300] loss: 1.691 acc: 38.70 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 40.02 %
Average loss on the 10000 test images: 1.648
[14,   100] loss: 1.700 acc: 38.80 time: 2.66
[14,   200] loss: 1.703 acc: 38.91 time: 2.57
[14,   300] loss: 1.698 acc: 38.88 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 41.07 %
Average loss on the 10000 test images: 1.624
[15,   100] loss: 1.687 acc: 39.41 time: 2.70
[15,   200] loss: 1.682 acc: 39.36 time: 2.55
[15,   300] loss: 1.683 acc: 39.34 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 41.97 %
Average loss on the 10000 test images: 1.612
[16,   100] loss: 1.668 acc: 39.81 time: 2.67
```

```
[16,   200] loss: 1.676 acc: 39.52 time: 2.60
[16,   300] loss: 1.677 acc: 39.78 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 43.74 %
Average loss on the 10000 test images: 1.573
[17,   100] loss: 1.678 acc: 39.62 time: 2.68
[17,   200] loss: 1.679 acc: 39.42 time: 2.55
[17,   300] loss: 1.676 acc: 39.51 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 42.84 %
Average loss on the 10000 test images: 1.584
[18,   100] loss: 1.654 acc: 40.52 time: 2.69
[18,   200] loss: 1.654 acc: 40.03 time: 2.58
[18,   300] loss: 1.660 acc: 40.29 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 43.43 %
Average loss on the 10000 test images: 1.572
[19,   100] loss: 1.670 acc: 40.36 time: 2.69
[19,   200] loss: 1.653 acc: 40.72 time: 2.57
[19,   300] loss: 1.645 acc: 40.77 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 43.07 %
Average loss on the 10000 test images: 1.582
[20,   100] loss: 1.658 acc: 40.86 time: 2.68
[20,   200] loss: 1.672 acc: 39.80 time: 2.55
[20,   300] loss: 1.662 acc: 39.72 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 43.88 %
Average loss on the 10000 test images: 1.561
[21,   100] loss: 1.648 acc: 40.88 time: 2.69
[21,   200] loss: 1.656 acc: 40.16 time: 2.57
[21,   300] loss: 1.650 acc: 40.77 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 43.66 %
Average loss on the 10000 test images: 1.559
[22,   100] loss: 1.674 acc: 40.10 time: 2.68
[22,   200] loss: 1.671 acc: 39.62 time: 2.58
[22,   300] loss: 1.655 acc: 40.37 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 42.26 %
Average loss on the 10000 test images: 1.593
[23,   100] loss: 1.656 acc: 39.92 time: 2.67
[23,   200] loss: 1.659 acc: 40.27 time: 2.56
[23,   300] loss: 1.650 acc: 40.61 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 43.40 %
Average loss on the 10000 test images: 1.572
[24,   100] loss: 1.654 acc: 40.62 time: 2.70
```

```
[24,   200] loss: 1.651 acc: 40.23 time: 2.58
[24,   300] loss: 1.652 acc: 40.40 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 42.79 %
Average loss on the 10000 test images: 1.590
[25,   100] loss: 1.668 acc: 39.66 time: 2.69
[25,   200] loss: 1.641 acc: 40.93 time: 2.57
[25,   300] loss: 1.634 acc: 40.92 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 43.50 %
Average loss on the 10000 test images: 1.568
[26,   100] loss: 1.629 acc: 41.25 time: 2.68
[26,   200] loss: 1.647 acc: 40.55 time: 2.58
[26,   300] loss: 1.633 acc: 41.34 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 43.57 %
Average loss on the 10000 test images: 1.574
[27,   100] loss: 1.634 acc: 40.37 time: 2.66
[27,   200] loss: 1.643 acc: 40.56 time: 2.57
[27,   300] loss: 1.652 acc: 40.26 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 43.59 %
Average loss on the 10000 test images: 1.584
[28,   100] loss: 1.631 acc: 41.90 time: 2.71
[28,   200] loss: 1.634 acc: 41.16 time: 2.55
[28,   300] loss: 1.641 acc: 41.13 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 42.24 %
Average loss on the 10000 test images: 1.586
[29,   100] loss: 1.651 acc: 40.59 time: 2.68
[29,   200] loss: 1.627 acc: 41.12 time: 2.55
[29,   300] loss: 1.641 acc: 40.98 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 43.56 %
Average loss on the 10000 test images: 1.594
[30,   100] loss: 1.641 acc: 40.66 time: 2.69
[30,   200] loss: 1.641 acc: 41.09 time: 2.59
[30,   300] loss: 1.631 acc: 41.57 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 44.45 %
Average loss on the 10000 test images: 1.540
Finished Training
```

6 Supervised training on the pre-trained model

6.0.1 In this section, we will load the pre-trained ResNet18 model and re-train the whole model on the classification task.

[ ]:
```
# TODO: Load the pre-trained ResNet18 model
import torch.nn as nn
import torch.nn.functional as F

# TODO: Load the pre-trained ResNet18 model
net = resnet18(num_classes=10).to(device)
net.load_state_dict(torch.load('./net_pretrain.pt'))
# net.train()
# net = torch.load('net_pretrain.pt', map_location=torch.device(device))
print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
```

```
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
```

```
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:
```python
# TODO: Define criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

```
[ ]: # Both the self-supervised rotation task and supervised CIFAR10 classification␣
      ↪are
     # trained with the CrossEntropyLoss, so we can use the training loop code.

     def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

         if optimizer == "Adam":
             optimizer_use = optim.Adam(net.parameters(), lr=init_lr, eps=1e-08,␣
      ↪weight_decay=0.001)
         elif optimizer == "SGD":
             optimizer_use = optim.SGD(net.parameters(), lr=init_lr, momentum=0.01)

         for epoch in range(num_epochs):  # loop over the dataset multiple times

             running_loss = 0.0
             running_correct = 0.0
             running_total = 0.0
             start_time = time.time()

             net.train()

             for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
      ↪enumerate(trainloader, 0):

                 # TODO: Set the data to the correct device; Different task will use␣
      ↪different inputs and labels
                 if task == 'rotation':
                   images, labels = imgs_rotated.to(device), rotation_label.
      ↪to(device)
                 elif task == 'classification':
                   images, labels = imgs.to(device), cls_label.to(device)

                 # TODO: Zero the parameter gradients
                 optimizer_use.zero_grad()

                 # TODO: forward + backward + optimize
                 y_pred = net(images)
                 loss = criterion(y_pred, labels)
                 loss.backward()
                 optimizer_use.step()

                 # TODO: Get predicted results
                 predicted = torch.argmax(y_pred, dim=1)

                 # print statistics
                 print_freq = 100
                 running_loss += loss.item()
```

```
            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        net.eval()
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=50, decay_epochs=10, init_lr=0.001,␣
↪task='classification')
torch.save(net, 'supervised_pretrain.pt')
```

```
[1,   100] loss: 1.562 acc: 44.13 time: 3.01
[1,   200] loss: 1.149 acc: 59.60 time: 2.55
[1,   300] loss: 1.045 acc: 63.06 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 65.70 %
Average loss on the 10000 test images: 0.983
[2,   100] loss: 0.946 acc: 67.09 time: 2.68
[2,   200] loss: 0.904 acc: 68.80 time: 2.57
[2,   300] loss: 0.886 acc: 68.88 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 71.49 %
Average loss on the 10000 test images: 0.821
[3,   100] loss: 0.810 acc: 72.14 time: 2.70
[3,   200] loss: 0.830 acc: 71.13 time: 2.57
[3,   300] loss: 0.796 acc: 72.77 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 72.31 %
Average loss on the 10000 test images: 0.791
[4,   100] loss: 0.752 acc: 73.84 time: 2.70
```

```
[4,   200] loss: 0.766 acc: 73.93 time: 2.56
[4,   300] loss: 0.761 acc: 74.24 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 73.88 %
Average loss on the 10000 test images: 0.762
[5,   100] loss: 0.727 acc: 75.08 time: 2.68
[5,   200] loss: 0.733 acc: 74.88 time: 2.58
[5,   300] loss: 0.732 acc: 74.73 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 74.34 %
Average loss on the 10000 test images: 0.734
[6,   100] loss: 0.703 acc: 75.67 time: 2.70
[6,   200] loss: 0.686 acc: 76.36 time: 2.56
[6,   300] loss: 0.678 acc: 76.55 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 75.73 %
Average loss on the 10000 test images: 0.706
[7,   100] loss: 0.663 acc: 77.12 time: 2.67
[7,   200] loss: 0.673 acc: 76.48 time: 2.56
[7,   300] loss: 0.694 acc: 76.53 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 74.15 %
Average loss on the 10000 test images: 0.757
[8,   100] loss: 0.648 acc: 77.50 time: 2.68
[8,   200] loss: 0.659 acc: 77.51 time: 2.56
[8,   300] loss: 0.664 acc: 77.09 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 75.76 %
Average loss on the 10000 test images: 0.707
[9,   100] loss: 0.634 acc: 78.19 time: 2.73
[9,   200] loss: 0.639 acc: 77.65 time: 2.58
[9,   300] loss: 0.649 acc: 77.74 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 77.23 %
Average loss on the 10000 test images: 0.662
[10,   100] loss: 0.614 acc: 78.58 time: 2.67
[10,   200] loss: 0.625 acc: 78.65 time: 2.57
[10,   300] loss: 0.636 acc: 78.08 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 77.15 %
Average loss on the 10000 test images: 0.675
[11,   100] loss: 0.603 acc: 79.22 time: 2.68
[11,   200] loss: 0.624 acc: 78.51 time: 2.58
[11,   300] loss: 0.627 acc: 78.31 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 76.04 %
Average loss on the 10000 test images: 0.707
[12,   100] loss: 0.541 acc: 81.80 time: 2.72
```

```
[12,   200] loss: 0.500 acc: 83.24 time: 2.59
[12,   300] loss: 0.498 acc: 82.88 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 81.95 %
Average loss on the 10000 test images: 0.526
[13,   100] loss: 0.464 acc: 84.45 time: 2.70
[13,   200] loss: 0.466 acc: 84.30 time: 2.60
[13,   300] loss: 0.467 acc: 84.45 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 82.41 %
Average loss on the 10000 test images: 0.511
[14,   100] loss: 0.460 acc: 84.62 time: 2.72
[14,   200] loss: 0.453 acc: 84.39 time: 2.57
[14,   300] loss: 0.453 acc: 84.43 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 82.56 %
Average loss on the 10000 test images: 0.508
[15,   100] loss: 0.441 acc: 85.14 time: 2.68
[15,   200] loss: 0.435 acc: 85.43 time: 2.56
[15,   300] loss: 0.442 acc: 84.80 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 82.96 %
Average loss on the 10000 test images: 0.500
[16,   100] loss: 0.433 acc: 85.07 time: 2.68
[16,   200] loss: 0.422 acc: 85.71 time: 2.55
[16,   300] loss: 0.435 acc: 85.16 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 83.26 %
Average loss on the 10000 test images: 0.496
[17,   100] loss: 0.428 acc: 85.41 time: 2.67
[17,   200] loss: 0.420 acc: 85.77 time: 2.59
[17,   300] loss: 0.422 acc: 85.45 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 83.56 %
Average loss on the 10000 test images: 0.489
[18,   100] loss: 0.402 acc: 86.30 time: 2.68
[18,   200] loss: 0.409 acc: 85.92 time: 2.57
[18,   300] loss: 0.419 acc: 85.74 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 83.15 %
Average loss on the 10000 test images: 0.492
[19,   100] loss: 0.390 acc: 86.46 time: 2.68
[19,   200] loss: 0.423 acc: 85.55 time: 2.58
[19,   300] loss: 0.405 acc: 86.11 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 83.06 %
Average loss on the 10000 test images: 0.489
[20,   100] loss: 0.404 acc: 86.03 time: 2.67
```

```
[20,   200] loss: 0.405 acc: 86.00 time: 2.59
[20,   300] loss: 0.411 acc: 85.84 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 83.38 %
Average loss on the 10000 test images: 0.483
[21,   100] loss: 0.391 acc: 86.33 time: 2.69
[21,   200] loss: 0.403 acc: 86.13 time: 2.55
[21,   300] loss: 0.405 acc: 85.96 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 83.51 %
Average loss on the 10000 test images: 0.484
[22,   100] loss: 0.377 acc: 87.18 time: 2.68
[22,   200] loss: 0.382 acc: 87.03 time: 2.61
[22,   300] loss: 0.390 acc: 86.27 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 83.88 %
Average loss on the 10000 test images: 0.475
[23,   100] loss: 0.378 acc: 87.05 time: 2.69
[23,   200] loss: 0.364 acc: 87.41 time: 2.57
[23,   300] loss: 0.381 acc: 86.92 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 83.96 %
Average loss on the 10000 test images: 0.474
[24,   100] loss: 0.372 acc: 87.22 time: 2.68
[24,   200] loss: 0.366 acc: 87.43 time: 2.61
[24,   300] loss: 0.377 acc: 87.34 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.06 %
Average loss on the 10000 test images: 0.473
[25,   100] loss: 0.361 acc: 87.48 time: 2.67
[25,   200] loss: 0.371 acc: 87.35 time: 2.62
[25,   300] loss: 0.373 acc: 87.28 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 84.19 %
Average loss on the 10000 test images: 0.469
[26,   100] loss: 0.371 acc: 87.56 time: 2.67
[26,   200] loss: 0.367 acc: 87.36 time: 2.56
[26,   300] loss: 0.378 acc: 87.41 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.08 %
Average loss on the 10000 test images: 0.471
[27,   100] loss: 0.364 acc: 87.49 time: 2.71
[27,   200] loss: 0.378 acc: 87.19 time: 2.58
[27,   300] loss: 0.372 acc: 87.30 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.12 %
Average loss on the 10000 test images: 0.473
[28,   100] loss: 0.364 acc: 87.59 time: 2.66
```

```
[28,   200] loss: 0.366 acc: 87.58 time: 2.58
[28,   300] loss: 0.367 acc: 87.38 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.09 %
Average loss on the 10000 test images: 0.472
[29,   100] loss: 0.370 acc: 87.31 time: 2.70
[29,   200] loss: 0.366 acc: 87.09 time: 2.56
[29,   300] loss: 0.363 acc: 87.52 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.12 %
Average loss on the 10000 test images: 0.471
[30,   100] loss: 0.364 acc: 87.64 time: 2.67
[30,   200] loss: 0.370 acc: 87.34 time: 2.63
[30,   300] loss: 0.360 acc: 87.88 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.10 %
Average loss on the 10000 test images: 0.472
[31,   100] loss: 0.361 acc: 87.76 time: 2.72
[31,   200] loss: 0.363 acc: 87.25 time: 2.60
[31,   300] loss: 0.365 acc: 87.44 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 84.19 %
Average loss on the 10000 test images: 0.472
[32,   100] loss: 0.355 acc: 87.91 time: 2.68
[32,   200] loss: 0.367 acc: 87.55 time: 2.58
[32,   300] loss: 0.359 acc: 87.59 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.07 %
Average loss on the 10000 test images: 0.470
[33,   100] loss: 0.369 acc: 87.47 time: 2.69
[33,   200] loss: 0.357 acc: 87.76 time: 2.57
[33,   300] loss: 0.360 acc: 87.51 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.10 %
Average loss on the 10000 test images: 0.469
[34,   100] loss: 0.368 acc: 87.47 time: 2.68
[34,   200] loss: 0.361 acc: 87.73 time: 2.60
[34,   300] loss: 0.365 acc: 87.68 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.11 %
Average loss on the 10000 test images: 0.469
[35,   100] loss: 0.364 acc: 87.42 time: 2.66
[35,   200] loss: 0.355 acc: 87.98 time: 2.56
[35,   300] loss: 0.360 acc: 87.74 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.24 %
Average loss on the 10000 test images: 0.468
[36,   100] loss: 0.346 acc: 88.13 time: 2.70
```

```
[36,   200] loss: 0.358 acc: 87.94 time: 2.55
[36,   300] loss: 0.357 acc: 87.93 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.10 %
Average loss on the 10000 test images: 0.471
[37,   100] loss: 0.376 acc: 87.07 time: 2.67
[37,   200] loss: 0.347 acc: 88.10 time: 2.56
[37,   300] loss: 0.346 acc: 88.23 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.07 %
Average loss on the 10000 test images: 0.469
[38,   100] loss: 0.372 acc: 87.39 time: 2.70
[38,   200] loss: 0.361 acc: 87.73 time: 2.56
[38,   300] loss: 0.361 acc: 87.55 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.30 %
Average loss on the 10000 test images: 0.468
[39,   100] loss: 0.357 acc: 87.64 time: 2.67
[39,   200] loss: 0.365 acc: 87.44 time: 2.57
[39,   300] loss: 0.358 acc: 87.70 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.20 %
Average loss on the 10000 test images: 0.470
[40,   100] loss: 0.360 acc: 87.49 time: 2.72
[40,   200] loss: 0.350 acc: 87.84 time: 2.57
[40,   300] loss: 0.364 acc: 87.95 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.16 %
Average loss on the 10000 test images: 0.470
[41,   100] loss: 0.341 acc: 88.47 time: 2.66
[41,   200] loss: 0.371 acc: 87.34 time: 2.58
[41,   300] loss: 0.363 acc: 87.75 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 84.20 %
Average loss on the 10000 test images: 0.470
[42,   100] loss: 0.354 acc: 87.80 time: 2.65
[42,   200] loss: 0.357 acc: 87.87 time: 2.55
[42,   300] loss: 0.363 acc: 87.33 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 84.23 %
Average loss on the 10000 test images: 0.467
[43,   100] loss: 0.357 acc: 87.83 time: 2.69
[43,   200] loss: 0.372 acc: 87.34 time: 2.58
[43,   300] loss: 0.364 acc: 87.56 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.19 %
Average loss on the 10000 test images: 0.469
[44,   100] loss: 0.357 acc: 87.62 time: 2.68
```

```
[44,   200] loss: 0.362 acc: 87.54 time: 2.62
[44,   300] loss: 0.364 acc: 87.96 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.31 %
Average loss on the 10000 test images: 0.468
[45,   100] loss: 0.357 acc: 87.73 time: 2.68
[45,   200] loss: 0.359 acc: 88.02 time: 2.56
[45,   300] loss: 0.355 acc: 87.80 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.35 %
Average loss on the 10000 test images: 0.468
[46,   100] loss: 0.363 acc: 87.51 time: 2.67
[46,   200] loss: 0.357 acc: 87.97 time: 2.57
[46,   300] loss: 0.363 acc: 87.30 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.34 %
Average loss on the 10000 test images: 0.468
[47,   100] loss: 0.366 acc: 87.41 time: 2.72
[47,   200] loss: 0.363 acc: 87.48 time: 2.56
[47,   300] loss: 0.362 acc: 88.02 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.14 %
Average loss on the 10000 test images: 0.470
[48,   100] loss: 0.365 acc: 87.59 time: 2.72
[48,   200] loss: 0.355 acc: 87.74 time: 2.58
[48,   300] loss: 0.364 acc: 87.59 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.30 %
Average loss on the 10000 test images: 0.468
[49,   100] loss: 0.356 acc: 87.80 time: 2.68
[49,   200] loss: 0.359 acc: 87.80 time: 2.55
[49,   300] loss: 0.363 acc: 87.73 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.29 %
Average loss on the 10000 test images: 0.467
[50,   100] loss: 0.372 acc: 87.51 time: 2.71
[50,   200] loss: 0.357 acc: 87.60 time: 2.56
[50,   300] loss: 0.365 acc: 87.27 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.16 %
Average loss on the 10000 test images: 0.469
Finished Training
```

7 Supervised training on the randomly initialized model

7.0.1 In this section, we will randomly initialize a ResNet18 model and re-train the whole model on the classification task.

[ ]:
```
import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet18

# TODO: Randomly initialize a ResNet18 model
net = resnet18(num_classes=10)
net = net.to(device)
```

[ ]:
```
# TODO: Define criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=50, decay_epochs=10, init_lr=0.001,␣
 ↪task='classification')
torch.save(net, 'supervised_random.pt')
```

```
[1,   100] loss: 1.822 acc: 34.18 time: 2.67
[1,   200] loss: 1.541 acc: 43.57 time: 2.56
[1,   300] loss: 1.424 acc: 48.12 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 51.97 %
Average loss on the 10000 test images: 1.367
[2,   100] loss: 1.277 acc: 54.01 time: 2.72
[2,   200] loss: 1.223 acc: 55.75 time: 2.59
[2,   300] loss: 1.161 acc: 58.88 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 60.86 %
Average loss on the 10000 test images: 1.121
[3,   100] loss: 1.082 acc: 61.77 time: 2.68
[3,   200] loss: 1.062 acc: 61.93 time: 2.57
[3,   300] loss: 1.022 acc: 64.25 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 65.35 %
Average loss on the 10000 test images: 1.014
[4,   100] loss: 0.976 acc: 65.62 time: 2.71
[4,   200] loss: 0.965 acc: 66.80 time: 2.56
[4,   300] loss: 0.961 acc: 66.57 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 64.19 %
Average loss on the 10000 test images: 1.051
[5,   100] loss: 0.917 acc: 68.45 time: 2.65
[5,   200] loss: 0.911 acc: 68.25 time: 2.53
```

```
[5,   300] loss: 0.887 acc: 69.36 time: 2.51
TESTING:
Accuracy of the network on the 10000 test images: 69.88 %
Average loss on the 10000 test images: 0.881
[6,   100] loss: 0.857 acc: 69.99 time: 2.69
[6,   200] loss: 0.843 acc: 71.12 time: 2.54
[6,   300] loss: 0.869 acc: 69.68 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 71.83 %
Average loss on the 10000 test images: 0.812
[7,   100] loss: 0.805 acc: 71.86 time: 2.70
[7,   200] loss: 0.832 acc: 71.59 time: 2.56
[7,   300] loss: 0.810 acc: 72.12 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 72.41 %
Average loss on the 10000 test images: 0.825
[8,   100] loss: 0.781 acc: 73.34 time: 2.71
[8,   200] loss: 0.786 acc: 73.09 time: 2.57
[8,   300] loss: 0.778 acc: 73.60 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 70.10 %
Average loss on the 10000 test images: 0.876
[9,   100] loss: 0.755 acc: 74.18 time: 2.69
[9,   200] loss: 0.743 acc: 74.79 time: 2.57
[9,   300] loss: 0.760 acc: 74.15 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 71.50 %
Average loss on the 10000 test images: 0.847
[10,   100] loss: 0.711 acc: 75.83 time: 2.70
[10,   200] loss: 0.736 acc: 75.05 time: 2.57
[10,   300] loss: 0.733 acc: 75.07 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 74.70 %
Average loss on the 10000 test images: 0.747
[11,   100] loss: 0.701 acc: 76.30 time: 2.65
[11,   200] loss: 0.710 acc: 75.79 time: 2.49
[11,   300] loss: 0.695 acc: 76.01 time: 2.50
TESTING:
Accuracy of the network on the 10000 test images: 75.42 %
Average loss on the 10000 test images: 0.710
[12,   100] loss: 0.605 acc: 79.22 time: 2.69
[12,   200] loss: 0.560 acc: 80.92 time: 2.56
[12,   300] loss: 0.531 acc: 81.98 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 81.57 %
Average loss on the 10000 test images: 0.537
[13,   100] loss: 0.529 acc: 82.39 time: 2.72
[13,   200] loss: 0.508 acc: 82.78 time: 2.57
```

```
[13,   300] loss: 0.501 acc: 82.84 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 81.86 %
Average loss on the 10000 test images: 0.519
[14,   100] loss: 0.484 acc: 83.92 time: 2.65
[14,   200] loss: 0.497 acc: 82.94 time: 2.52
[14,   300] loss: 0.479 acc: 83.80 time: 2.50
TESTING:
Accuracy of the network on the 10000 test images: 82.74 %
Average loss on the 10000 test images: 0.512
[15,   100] loss: 0.460 acc: 84.10 time: 2.68
[15,   200] loss: 0.454 acc: 84.68 time: 2.52
[15,   300] loss: 0.478 acc: 83.78 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 82.63 %
Average loss on the 10000 test images: 0.507
[16,   100] loss: 0.443 acc: 84.77 time: 2.69
[16,   200] loss: 0.469 acc: 83.91 time: 2.54
[16,   300] loss: 0.463 acc: 84.13 time: 2.53
TESTING:
Accuracy of the network on the 10000 test images: 82.95 %
Average loss on the 10000 test images: 0.499
[17,   100] loss: 0.438 acc: 85.30 time: 2.69
[17,   200] loss: 0.444 acc: 84.71 time: 2.54
[17,   300] loss: 0.434 acc: 85.05 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 82.88 %
Average loss on the 10000 test images: 0.499
[18,   100] loss: 0.418 acc: 85.73 time: 2.66
[18,   200] loss: 0.450 acc: 84.61 time: 2.53
[18,   300] loss: 0.442 acc: 84.87 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 83.01 %
Average loss on the 10000 test images: 0.493
[19,   100] loss: 0.428 acc: 85.59 time: 2.71
[19,   200] loss: 0.435 acc: 85.16 time: 2.55
[19,   300] loss: 0.421 acc: 85.84 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 83.69 %
Average loss on the 10000 test images: 0.482
[20,   100] loss: 0.423 acc: 85.27 time: 2.65
[20,   200] loss: 0.408 acc: 86.12 time: 2.53
[20,   300] loss: 0.420 acc: 85.34 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 83.58 %
Average loss on the 10000 test images: 0.480
[21,   100] loss: 0.402 acc: 86.23 time: 2.64
[21,   200] loss: 0.406 acc: 86.30 time: 2.52
```

```
[21,   300] loss: 0.401 acc: 86.14 time: 2.50
TESTING:
Accuracy of the network on the 10000 test images: 84.17 %
Average loss on the 10000 test images: 0.471
[22,   100] loss: 0.389 acc: 86.86 time: 2.66
[22,   200] loss: 0.381 acc: 87.06 time: 2.52
[22,   300] loss: 0.375 acc: 86.89 time: 2.50
TESTING:
Accuracy of the network on the 10000 test images: 84.27 %
Average loss on the 10000 test images: 0.462
[23,   100] loss: 0.377 acc: 87.12 time: 2.68
[23,   200] loss: 0.378 acc: 87.18 time: 2.50
[23,   300] loss: 0.372 acc: 87.47 time: 2.52
TESTING:
Accuracy of the network on the 10000 test images: 84.41 %
Average loss on the 10000 test images: 0.461
[24,   100] loss: 0.369 acc: 87.55 time: 2.70
[24,   200] loss: 0.379 acc: 87.20 time: 2.58
[24,   300] loss: 0.378 acc: 86.94 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.52 %
Average loss on the 10000 test images: 0.459
[25,   100] loss: 0.375 acc: 87.21 time: 2.70
[25,   200] loss: 0.366 acc: 87.48 time: 2.56
[25,   300] loss: 0.370 acc: 87.32 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.31 %
Average loss on the 10000 test images: 0.460
[26,   100] loss: 0.360 acc: 87.80 time: 2.61
[26,   200] loss: 0.373 acc: 87.22 time: 2.50
[26,   300] loss: 0.366 acc: 87.33 time: 2.51
TESTING:
Accuracy of the network on the 10000 test images: 84.27 %
Average loss on the 10000 test images: 0.461
[27,   100] loss: 0.353 acc: 87.93 time: 2.71
[27,   200] loss: 0.375 acc: 87.12 time: 2.56
[27,   300] loss: 0.370 acc: 87.49 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.51 %
Average loss on the 10000 test images: 0.459
[28,   100] loss: 0.363 acc: 87.59 time: 2.71
[28,   200] loss: 0.367 acc: 87.27 time: 2.56
[28,   300] loss: 0.360 acc: 87.40 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.40 %
Average loss on the 10000 test images: 0.459
[29,   100] loss: 0.359 acc: 87.52 time: 2.70
[29,   200] loss: 0.355 acc: 87.91 time: 2.60
```

```
[29,   300] loss: 0.360 acc: 87.96 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.53 %
Average loss on the 10000 test images: 0.460
[30,   100] loss: 0.361 acc: 87.77 time: 2.69
[30,   200] loss: 0.351 acc: 88.18 time: 2.56
[30,   300] loss: 0.348 acc: 88.42 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.50 %
Average loss on the 10000 test images: 0.460
[31,   100] loss: 0.362 acc: 87.59 time: 2.71
[31,   200] loss: 0.358 acc: 88.08 time: 2.58
[31,   300] loss: 0.348 acc: 87.73 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.61 %
Average loss on the 10000 test images: 0.459
[32,   100] loss: 0.351 acc: 88.12 time: 2.68
[32,   200] loss: 0.342 acc: 88.73 time: 2.57
[32,   300] loss: 0.349 acc: 88.32 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.46 %
Average loss on the 10000 test images: 0.461
[33,   100] loss: 0.359 acc: 87.77 time: 2.72
[33,   200] loss: 0.362 acc: 87.41 time: 2.57
[33,   300] loss: 0.349 acc: 88.12 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.41 %
Average loss on the 10000 test images: 0.460
[34,   100] loss: 0.362 acc: 87.73 time: 2.65
[34,   200] loss: 0.348 acc: 88.12 time: 2.49
[34,   300] loss: 0.359 acc: 88.14 time: 2.49
TESTING:
Accuracy of the network on the 10000 test images: 84.46 %
Average loss on the 10000 test images: 0.458
[35,   100] loss: 0.357 acc: 87.97 time: 2.70
[35,   200] loss: 0.346 acc: 88.20 time: 2.55
[35,   300] loss: 0.357 acc: 87.98 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.51 %
Average loss on the 10000 test images: 0.460
[36,   100] loss: 0.352 acc: 88.11 time: 2.71
[36,   200] loss: 0.360 acc: 87.94 time: 2.57
[36,   300] loss: 0.350 acc: 88.02 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.52 %
Average loss on the 10000 test images: 0.460
[37,   100] loss: 0.339 acc: 88.42 time: 2.70
[37,   200] loss: 0.351 acc: 87.92 time: 2.57
```

```
[37,   300] loss: 0.371 acc: 87.23 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.54 %
Average loss on the 10000 test images: 0.461
[38,   100] loss: 0.355 acc: 87.95 time: 2.71
[38,   200] loss: 0.354 acc: 87.88 time: 2.57
[38,   300] loss: 0.347 acc: 88.25 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.58 %
Average loss on the 10000 test images: 0.457
[39,   100] loss: 0.347 acc: 88.37 time: 2.71
[39,   200] loss: 0.354 acc: 87.63 time: 2.58
[39,   300] loss: 0.364 acc: 87.67 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.61 %
Average loss on the 10000 test images: 0.458
[40,   100] loss: 0.358 acc: 87.77 time: 2.69
[40,   200] loss: 0.339 acc: 88.32 time: 2.58
[40,   300] loss: 0.352 acc: 87.98 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 84.41 %
Average loss on the 10000 test images: 0.460
[41,   100] loss: 0.351 acc: 88.00 time: 2.69
[41,   200] loss: 0.350 acc: 87.82 time: 2.56
[41,   300] loss: 0.353 acc: 87.88 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 84.47 %
Average loss on the 10000 test images: 0.459
[42,   100] loss: 0.338 acc: 88.36 time: 2.68
[42,   200] loss: 0.364 acc: 87.72 time: 2.55
[42,   300] loss: 0.357 acc: 87.81 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.52 %
Average loss on the 10000 test images: 0.460
[43,   100] loss: 0.358 acc: 87.76 time: 2.67
[43,   200] loss: 0.362 acc: 87.80 time: 2.55
[43,   300] loss: 0.349 acc: 87.99 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.50 %
Average loss on the 10000 test images: 0.459
[44,   100] loss: 0.355 acc: 87.66 time: 2.70
[44,   200] loss: 0.355 acc: 88.15 time: 2.54
[44,   300] loss: 0.339 acc: 88.52 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.50 %
Average loss on the 10000 test images: 0.460
[45,   100] loss: 0.348 acc: 88.16 time: 2.68
[45,   200] loss: 0.359 acc: 87.51 time: 2.58
```

```
[45,   300] loss: 0.354 acc: 87.95 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 84.30 %
Average loss on the 10000 test images: 0.462
[46,   100] loss: 0.350 acc: 88.28 time: 2.72
[46,   200] loss: 0.353 acc: 87.83 time: 2.56
[46,   300] loss: 0.358 acc: 87.70 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.39 %
Average loss on the 10000 test images: 0.460
[47,   100] loss: 0.357 acc: 87.66 time: 2.67
[47,   200] loss: 0.351 acc: 87.89 time: 2.55
[47,   300] loss: 0.356 acc: 87.76 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 84.46 %
Average loss on the 10000 test images: 0.459
[48,   100] loss: 0.351 acc: 87.95 time: 2.65
[48,   200] loss: 0.341 acc: 88.27 time: 2.49
[48,   300] loss: 0.350 acc: 88.02 time: 2.50
TESTING:
Accuracy of the network on the 10000 test images: 84.49 %
Average loss on the 10000 test images: 0.459
[49,   100] loss: 0.342 acc: 88.66 time: 2.70
[49,   200] loss: 0.352 acc: 88.06 time: 2.55
[49,   300] loss: 0.359 acc: 87.85 time: 2.54
TESTING:
Accuracy of the network on the 10000 test images: 84.49 %
Average loss on the 10000 test images: 0.463
[50,   100] loss: 0.354 acc: 87.92 time: 2.69
[50,   200] loss: 0.348 acc: 88.21 time: 2.58
[50,   300] loss: 0.339 acc: 88.39 time: 2.55
TESTING:
Accuracy of the network on the 10000 test images: 84.47 %
Average loss on the 10000 test images: 0.459
Finished Training
```

8 Extra Credit 1: Accuracy-Sample Plot

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

[ ]:
```
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision.models import resnet18
```

```
import time

criterion = nn.CrossEntropyLoss()
optimizer = "Adam"

def run_test(net, testloader, criterion, task):
    correct = 0
    total = 0
    avg_test_loss = 0.0
    # since we're not training, we don't need to calculate the gradients for␣
 ↪our outputs
    with torch.no_grad():
        for images, images_rotated, labels, cls_labels in testloader:
            if task == 'rotation':
              images, labels = images_rotated.to(device), labels.to(device)
            elif task == 'classification':
              images, labels = images.to(device), cls_labels.to(device)
            # TODO: Calculate outputs by running images through the network
            # The class with the highest energy is what we choose as prediction
            outputs = net(images)
            predicted = torch.argmax(outputs, dim=1)

            # loss
            avg_test_loss += criterion(outputs, labels)  / len(testloader)

            # calculate accuracy
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    print('TESTING:')
    print(f'Accuracy of the network on the 10000 test images: {100 * correct /␣
 ↪total:.2f} %')
    print(f'Average loss on the 10000 test images: {avg_test_loss:.3f}')

    return 100 * correct / total
```

[ ]:
```
def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    test_acc_max = 0

    if optimizer == "Adam":
        optimizer_use = optim.Adam(net.parameters(), lr=init_lr, eps=1e-08,␣
 ↪weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(net.parameters(), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times
```

```
        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
↪enumerate(trainloader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
↪different inputs and labels
            if task == 'rotation':
                images, labels = imgs_rotated.to(device), rotation_label.
↪to(device)

            elif task == 'classification':
                images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()
```

```
        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        with torch.no_grad():
            test_acc = run_test(net, testloader, criterion, task)
            if test_acc > test_acc_max:
                test_acc_max = test_acc

    print('Finished Training')
    return test_acc_max
```

[ ]:
```
### Stage 1: Pretrain + Finetune
data_num = [20, 100, 400, 1000, 5000]
accuracy = []

for num in data_num:
    ## Load Pretrain Model
    net = resnet18(num_classes=10).to(device)
    net.load_state_dict(torch.load('./net_pretrain.pt'))

    ## unfreeze layer4 and fc
    for name, param in net.named_parameters():
        if "layer4" in name:
            param.requires_grad = True
        elif "fc" in name:
            param.requires_grad = True
        else:
            param.requires_grad = False

    ## mini-trainset
    mini_set = []
    sample_dict = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0}
    idx = 0
    while sum(list(sample_dict.values())) < num * 10:
        temp = trainset[idx]
        if sample_dict[int(temp[3])] >= num:              # class already full
            idx += 1
        else:
            mini_set.append(temp)
            idx += 1
            sample_dict[int(temp[3])] += 1

    ## make dataloader
    trainloader = torch.utils.data.DataLoader(mini_set,␣
↪batch_size=min(len(mini_set), batch_size), shuffle=True, num_workers=2)
```

```
    print("\nStart Finetuning with {} samples perclass.".format(num))
    test_acc_max = train(net, criterion, optimizer, num_epochs=50,␣
 ↪decay_epochs=10, init_lr=0.001, task='classification')
    torch.save(net, 'finetune_pretrain_{}.pt'.format(num))
    accuracy.append(test_acc_max)

print(accuracy)
```

```
Start Finetuning with 20 samples.
TESTING:
Accuracy of the network on the 10000 test images: 11.48 %
Average loss on the 10000 test images: 2.524
TESTING:
Accuracy of the network on the 10000 test images: 12.10 %
Average loss on the 10000 test images: 2.447
TESTING:
Accuracy of the network on the 10000 test images: 12.80 %
Average loss on the 10000 test images: 2.378
TESTING:
Accuracy of the network on the 10000 test images: 13.56 %
Average loss on the 10000 test images: 2.316
TESTING:
Accuracy of the network on the 10000 test images: 14.36 %
Average loss on the 10000 test images: 2.259
TESTING:
Accuracy of the network on the 10000 test images: 15.50 %
Average loss on the 10000 test images: 2.210
TESTING:
Accuracy of the network on the 10000 test images: 16.65 %
Average loss on the 10000 test images: 2.167
TESTING:
Accuracy of the network on the 10000 test images: 17.98 %
Average loss on the 10000 test images: 2.133
TESTING:
Accuracy of the network on the 10000 test images: 19.32 %
Average loss on the 10000 test images: 2.104
TESTING:
Accuracy of the network on the 10000 test images: 21.04 %
Average loss on the 10000 test images: 2.079
TESTING:
Accuracy of the network on the 10000 test images: 22.85 %
Average loss on the 10000 test images: 2.057
TESTING:
Accuracy of the network on the 10000 test images: 22.86 %
Average loss on the 10000 test images: 2.055
```

```
TESTING:
Accuracy of the network on the 10000 test images: 22.95 %
Average loss on the 10000 test images: 2.052
TESTING:
Accuracy of the network on the 10000 test images: 23.14 %
Average loss on the 10000 test images: 2.050
TESTING:
Accuracy of the network on the 10000 test images: 23.32 %
Average loss on the 10000 test images: 2.047
TESTING:
Accuracy of the network on the 10000 test images: 23.54 %
Average loss on the 10000 test images: 2.044
TESTING:
Accuracy of the network on the 10000 test images: 23.75 %
Average loss on the 10000 test images: 2.042
TESTING:
Accuracy of the network on the 10000 test images: 24.02 %
Average loss on the 10000 test images: 2.040
TESTING:
Accuracy of the network on the 10000 test images: 24.22 %
Average loss on the 10000 test images: 2.037
TESTING:
Accuracy of the network on the 10000 test images: 24.30 %
Average loss on the 10000 test images: 2.036
TESTING:
Accuracy of the network on the 10000 test images: 24.40 %
Average loss on the 10000 test images: 2.034
TESTING:
Accuracy of the network on the 10000 test images: 24.43 %
Average loss on the 10000 test images: 2.034
TESTING:
Accuracy of the network on the 10000 test images: 24.45 %
Average loss on the 10000 test images: 2.034
TESTING:
Accuracy of the network on the 10000 test images: 24.46 %
Average loss on the 10000 test images: 2.033
TESTING:
Accuracy of the network on the 10000 test images: 24.47 %
Average loss on the 10000 test images: 2.033
TESTING:
Accuracy of the network on the 10000 test images: 24.43 %
Average loss on the 10000 test images: 2.033
TESTING:
Accuracy of the network on the 10000 test images: 24.43 %
Average loss on the 10000 test images: 2.033
TESTING:
Accuracy of the network on the 10000 test images: 24.44 %
Average loss on the 10000 test images: 2.033
```

```
TESTING:
Accuracy of the network on the 10000 test images: 24.43 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.43 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.44 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.44 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.44 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.45 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.46 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.45 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.46 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.45 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.46 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.47 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.47 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.47 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
```

```
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
TESTING:
Accuracy of the network on the 10000 test images: 24.48 %
Average loss on the 10000 test images: 2.032
Finished Training

Start Finetuning with 100 samples.
TESTING:
Accuracy of the network on the 10000 test images: 13.71 %
Average loss on the 10000 test images: 2.305
TESTING:
Accuracy of the network on the 10000 test images: 18.48 %
Average loss on the 10000 test images: 2.123
TESTING:
Accuracy of the network on the 10000 test images: 25.16 %
Average loss on the 10000 test images: 2.023
TESTING:
Accuracy of the network on the 10000 test images: 29.37 %
Average loss on the 10000 test images: 1.946
TESTING:
Accuracy of the network on the 10000 test images: 31.82 %
Average loss on the 10000 test images: 1.888
TESTING:
Accuracy of the network on the 10000 test images: 33.36 %
Average loss on the 10000 test images: 1.844
TESTING:
Accuracy of the network on the 10000 test images: 34.37 %
Average loss on the 10000 test images: 1.817
TESTING:
Accuracy of the network on the 10000 test images: 35.22 %
Average loss on the 10000 test images: 1.806
TESTING:
Accuracy of the network on the 10000 test images: 36.45 %
Average loss on the 10000 test images: 1.778
```

```
TESTING:
Accuracy of the network on the 10000 test images: 35.90 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 36.57 %
Average loss on the 10000 test images: 1.794
TESTING:
Accuracy of the network on the 10000 test images: 37.33 %
Average loss on the 10000 test images: 1.771
TESTING:
Accuracy of the network on the 10000 test images: 37.77 %
Average loss on the 10000 test images: 1.759
TESTING:
Accuracy of the network on the 10000 test images: 38.00 %
Average loss on the 10000 test images: 1.760
TESTING:
Accuracy of the network on the 10000 test images: 37.99 %
Average loss on the 10000 test images: 1.762
TESTING:
Accuracy of the network on the 10000 test images: 38.16 %
Average loss on the 10000 test images: 1.765
TESTING:
Accuracy of the network on the 10000 test images: 38.03 %
Average loss on the 10000 test images: 1.768
TESTING:
Accuracy of the network on the 10000 test images: 37.99 %
Average loss on the 10000 test images: 1.771
TESTING:
Accuracy of the network on the 10000 test images: 37.94 %
Average loss on the 10000 test images: 1.773
TESTING:
Accuracy of the network on the 10000 test images: 37.83 %
Average loss on the 10000 test images: 1.776
TESTING:
Accuracy of the network on the 10000 test images: 37.91 %
Average loss on the 10000 test images: 1.778
TESTING:
Accuracy of the network on the 10000 test images: 37.87 %
Average loss on the 10000 test images: 1.778
TESTING:
Accuracy of the network on the 10000 test images: 37.89 %
Average loss on the 10000 test images: 1.778
TESTING:
Accuracy of the network on the 10000 test images: 37.86 %
Average loss on the 10000 test images: 1.778
TESTING:
Accuracy of the network on the 10000 test images: 37.85 %
Average loss on the 10000 test images: 1.779
```

```
TESTING:
Accuracy of the network on the 10000 test images: 37.87 %
Average loss on the 10000 test images: 1.779
TESTING:
Accuracy of the network on the 10000 test images: 37.75 %
Average loss on the 10000 test images: 1.779
TESTING:
Accuracy of the network on the 10000 test images: 37.75 %
Average loss on the 10000 test images: 1.779
TESTING:
Accuracy of the network on the 10000 test images: 37.79 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.83 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.73 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.74 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.75 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.74 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.76 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.76 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.77 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.80 %
Average loss on the 10000 test images: 1.780
```

```
TESTING:
Accuracy of the network on the 10000 test images: 37.80 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.80 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
TESTING:
Accuracy of the network on the 10000 test images: 37.78 %
Average loss on the 10000 test images: 1.780
Finished Training

Start Finetuning with 400 samples.
TESTING:
Accuracy of the network on the 10000 test images: 29.40 %
Average loss on the 10000 test images: 1.941
TESTING:
Accuracy of the network on the 10000 test images: 36.26 %
Average loss on the 10000 test images: 1.768
TESTING:
Accuracy of the network on the 10000 test images: 37.95 %
Average loss on the 10000 test images: 1.714
TESTING:
Accuracy of the network on the 10000 test images: 39.07 %
Average loss on the 10000 test images: 1.674
TESTING:
Accuracy of the network on the 10000 test images: 40.25 %
Average loss on the 10000 test images: 1.655
TESTING:
Accuracy of the network on the 10000 test images: 39.60 %
Average loss on the 10000 test images: 1.670
```

```
TESTING:
Accuracy of the network on the 10000 test images: 40.82 %
Average loss on the 10000 test images: 1.646
TESTING:
Accuracy of the network on the 10000 test images: 40.23 %
Average loss on the 10000 test images: 1.656
TESTING:
Accuracy of the network on the 10000 test images: 40.01 %
Average loss on the 10000 test images: 1.684
TESTING:
Accuracy of the network on the 10000 test images: 41.03 %
Average loss on the 10000 test images: 1.654
TESTING:
Accuracy of the network on the 10000 test images: 41.18 %
Average loss on the 10000 test images: 1.643
TESTING:
Accuracy of the network on the 10000 test images: 42.00 %
Average loss on the 10000 test images: 1.626
TESTING:
Accuracy of the network on the 10000 test images: 42.06 %
Average loss on the 10000 test images: 1.629
TESTING:
Accuracy of the network on the 10000 test images: 41.97 %
Average loss on the 10000 test images: 1.630
TESTING:
Accuracy of the network on the 10000 test images: 42.29 %
Average loss on the 10000 test images: 1.632
TESTING:
Accuracy of the network on the 10000 test images: 42.19 %
Average loss on the 10000 test images: 1.634
TESTING:
Accuracy of the network on the 10000 test images: 42.06 %
Average loss on the 10000 test images: 1.639
TESTING:
Accuracy of the network on the 10000 test images: 41.78 %
Average loss on the 10000 test images: 1.642
TESTING:
Accuracy of the network on the 10000 test images: 42.08 %
Average loss on the 10000 test images: 1.645
TESTING:
Accuracy of the network on the 10000 test images: 42.15 %
Average loss on the 10000 test images: 1.645
TESTING:
Accuracy of the network on the 10000 test images: 41.91 %
Average loss on the 10000 test images: 1.647
TESTING:
Accuracy of the network on the 10000 test images: 41.91 %
Average loss on the 10000 test images: 1.648
```

```
TESTING:
Accuracy of the network on the 10000 test images: 41.87 %
Average loss on the 10000 test images: 1.648
TESTING:
Accuracy of the network on the 10000 test images: 41.90 %
Average loss on the 10000 test images: 1.648
TESTING:
Accuracy of the network on the 10000 test images: 41.85 %
Average loss on the 10000 test images: 1.649
TESTING:
Accuracy of the network on the 10000 test images: 41.80 %
Average loss on the 10000 test images: 1.650
TESTING:
Accuracy of the network on the 10000 test images: 41.81 %
Average loss on the 10000 test images: 1.650
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.650
TESTING:
Accuracy of the network on the 10000 test images: 41.77 %
Average loss on the 10000 test images: 1.650
TESTING:
Accuracy of the network on the 10000 test images: 41.81 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.80 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.80 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
```

```
TESTING:
Accuracy of the network on the 10000 test images: 41.81 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.81 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.83 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
TESTING:
Accuracy of the network on the 10000 test images: 41.82 %
Average loss on the 10000 test images: 1.651
Finished Training

Start Finetuning with 1000 samples.
TESTING:
Accuracy of the network on the 10000 test images: 37.21 %
Average loss on the 10000 test images: 1.729
TESTING:
Accuracy of the network on the 10000 test images: 39.18 %
Average loss on the 10000 test images: 1.675
TESTING:
Accuracy of the network on the 10000 test images: 39.89 %
Average loss on the 10000 test images: 1.652
```

```
TESTING:
Accuracy of the network on the 10000 test images: 40.77 %
Average loss on the 10000 test images: 1.633
TESTING:
Accuracy of the network on the 10000 test images: 41.31 %
Average loss on the 10000 test images: 1.619
TESTING:
Accuracy of the network on the 10000 test images: 40.59 %
Average loss on the 10000 test images: 1.629
TESTING:
Accuracy of the network on the 10000 test images: 41.77 %
Average loss on the 10000 test images: 1.613
TESTING:
Accuracy of the network on the 10000 test images: 41.36 %
Average loss on the 10000 test images: 1.617
TESTING:
Accuracy of the network on the 10000 test images: 42.21 %
Average loss on the 10000 test images: 1.603
TESTING:
Accuracy of the network on the 10000 test images: 42.55 %
Average loss on the 10000 test images: 1.611
TESTING:
Accuracy of the network on the 10000 test images: 40.51 %
Average loss on the 10000 test images: 1.638
TESTING:
Accuracy of the network on the 10000 test images: 42.76 %
Average loss on the 10000 test images: 1.590
TESTING:
Accuracy of the network on the 10000 test images: 43.02 %
Average loss on the 10000 test images: 1.591
TESTING:
Accuracy of the network on the 10000 test images: 42.98 %
Average loss on the 10000 test images: 1.591
TESTING:
Accuracy of the network on the 10000 test images: 43.13 %
Average loss on the 10000 test images: 1.590
TESTING:
Accuracy of the network on the 10000 test images: 43.14 %
Average loss on the 10000 test images: 1.590
TESTING:
Accuracy of the network on the 10000 test images: 43.05 %
Average loss on the 10000 test images: 1.594
TESTING:
Accuracy of the network on the 10000 test images: 43.29 %
Average loss on the 10000 test images: 1.593
TESTING:
Accuracy of the network on the 10000 test images: 43.27 %
Average loss on the 10000 test images: 1.596
```

```
TESTING:
Accuracy of the network on the 10000 test images: 43.47 %
Average loss on the 10000 test images: 1.597
TESTING:
Accuracy of the network on the 10000 test images: 43.00 %
Average loss on the 10000 test images: 1.599
TESTING:
Accuracy of the network on the 10000 test images: 43.12 %
Average loss on the 10000 test images: 1.598
TESTING:
Accuracy of the network on the 10000 test images: 43.19 %
Average loss on the 10000 test images: 1.598
TESTING:
Accuracy of the network on the 10000 test images: 43.25 %
Average loss on the 10000 test images: 1.598
TESTING:
Accuracy of the network on the 10000 test images: 43.28 %
Average loss on the 10000 test images: 1.598
TESTING:
Accuracy of the network on the 10000 test images: 43.26 %
Average loss on the 10000 test images: 1.598
TESTING:
Accuracy of the network on the 10000 test images: 43.21 %
Average loss on the 10000 test images: 1.599
TESTING:
Accuracy of the network on the 10000 test images: 43.21 %
Average loss on the 10000 test images: 1.599
TESTING:
Accuracy of the network on the 10000 test images: 43.20 %
Average loss on the 10000 test images: 1.599
TESTING:
Accuracy of the network on the 10000 test images: 43.23 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.27 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.25 %
Average loss on the 10000 test images: 1.600
```

```
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.23 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.23 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.23 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.21 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.21 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.22 %
Average loss on the 10000 test images: 1.600
TESTING:
Accuracy of the network on the 10000 test images: 43.21 %
Average loss on the 10000 test images: 1.600
Finished Training

Start Finetuning with 5000 samples.
```

```
[1,   100] loss: 1.927 acc: 29.57 time: 1.00
[1,   200] loss: 1.651 acc: 39.82 time: 0.70
[1,   300] loss: 1.610 acc: 40.78 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 42.18 %
Average loss on the 10000 test images: 1.599
[2,   100] loss: 1.583 acc: 42.49 time: 0.87
[2,   200] loss: 1.569 acc: 42.35 time: 0.66
[2,   300] loss: 1.566 acc: 43.06 time: 0.65
TESTING:
Accuracy of the network on the 10000 test images: 43.38 %
Average loss on the 10000 test images: 1.579
[3,   100] loss: 1.553 acc: 43.33 time: 1.00
[3,   200] loss: 1.558 acc: 43.06 time: 0.79
[3,   300] loss: 1.530 acc: 44.30 time: 0.77
TESTING:
Accuracy of the network on the 10000 test images: 44.07 %
Average loss on the 10000 test images: 1.564
[4,   100] loss: 1.539 acc: 43.80 time: 0.88
[4,   200] loss: 1.529 acc: 44.68 time: 0.66
[4,   300] loss: 1.538 acc: 43.84 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 44.14 %
Average loss on the 10000 test images: 1.549
[5,   100] loss: 1.510 acc: 45.48 time: 0.86
[5,   200] loss: 1.519 acc: 44.66 time: 0.67
[5,   300] loss: 1.516 acc: 44.37 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 44.13 %
Average loss on the 10000 test images: 1.548
[6,   100] loss: 1.503 acc: 45.51 time: 0.90
[6,   200] loss: 1.502 acc: 46.01 time: 0.68
[6,   300] loss: 1.508 acc: 44.76 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 44.45 %
Average loss on the 10000 test images: 1.539
[7,   100] loss: 1.480 acc: 46.24 time: 0.86
[7,   200] loss: 1.504 acc: 44.91 time: 0.65
[7,   300] loss: 1.504 acc: 45.71 time: 0.65
TESTING:
Accuracy of the network on the 10000 test images: 44.78 %
Average loss on the 10000 test images: 1.539
[8,   100] loss: 1.486 acc: 46.39 time: 0.87
[8,   200] loss: 1.497 acc: 45.72 time: 0.66
[8,   300] loss: 1.499 acc: 45.39 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 44.64 %
Average loss on the 10000 test images: 1.535
```

```
[9,   100] loss: 1.479 acc: 46.19 time: 0.89
[9,   200] loss: 1.481 acc: 46.00 time: 0.68
[9,   300] loss: 1.497 acc: 46.26 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 44.99 %
Average loss on the 10000 test images: 1.535
[10,   100] loss: 1.466 acc: 46.51 time: 0.96
[10,   200] loss: 1.489 acc: 45.43 time: 0.72
[10,   300] loss: 1.483 acc: 45.98 time: 0.71
TESTING:
Accuracy of the network on the 10000 test images: 45.02 %
Average loss on the 10000 test images: 1.526
[11,   100] loss: 1.464 acc: 46.89 time: 0.89
[11,   200] loss: 1.476 acc: 46.44 time: 0.67
[11,   300] loss: 1.460 acc: 47.19 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 45.71 %
Average loss on the 10000 test images: 1.516
[12,   100] loss: 1.445 acc: 47.41 time: 0.86
[12,   200] loss: 1.423 acc: 48.79 time: 0.66
[12,   300] loss: 1.413 acc: 48.64 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.36 %
Average loss on the 10000 test images: 1.492
[13,   100] loss: 1.426 acc: 48.03 time: 0.89
[13,   200] loss: 1.411 acc: 49.11 time: 0.68
[13,   300] loss: 1.408 acc: 48.83 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.32 %
Average loss on the 10000 test images: 1.493
[14,   100] loss: 1.393 acc: 49.25 time: 0.89
[14,   200] loss: 1.409 acc: 49.57 time: 0.68
[14,   300] loss: 1.422 acc: 48.01 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 46.43 %
Average loss on the 10000 test images: 1.494
[15,   100] loss: 1.403 acc: 49.52 time: 0.87
[15,   200] loss: 1.401 acc: 49.23 time: 0.67
[15,   300] loss: 1.403 acc: 49.20 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.63 %
Average loss on the 10000 test images: 1.495
[16,   100] loss: 1.394 acc: 50.09 time: 0.89
[16,   200] loss: 1.393 acc: 49.84 time: 0.67
[16,   300] loss: 1.406 acc: 49.27 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.64 %
Average loss on the 10000 test images: 1.495
```

```
[17,   100] loss: 1.404 acc: 49.28 time: 0.88
[17,   200] loss: 1.406 acc: 49.12 time: 0.66
[17,   300] loss: 1.393 acc: 49.48 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.37 %
Average loss on the 10000 test images: 1.495
[18,   100] loss: 1.383 acc: 50.21 time: 0.87
[18,   200] loss: 1.410 acc: 49.05 time: 0.67
[18,   300] loss: 1.405 acc: 49.05 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.52 %
Average loss on the 10000 test images: 1.496
[19,   100] loss: 1.406 acc: 49.35 time: 0.89
[19,   200] loss: 1.387 acc: 49.80 time: 0.67
[19,   300] loss: 1.383 acc: 50.05 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.26 %
Average loss on the 10000 test images: 1.494
[20,   100] loss: 1.392 acc: 49.73 time: 0.86
[20,   200] loss: 1.384 acc: 49.83 time: 0.66
[20,   300] loss: 1.391 acc: 49.37 time: 0.65
TESTING:
Accuracy of the network on the 10000 test images: 46.43 %
Average loss on the 10000 test images: 1.496
[21,   100] loss: 1.374 acc: 50.30 time: 0.88
[21,   200] loss: 1.385 acc: 49.62 time: 0.68
[21,   300] loss: 1.389 acc: 50.02 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 46.15 %
Average loss on the 10000 test images: 1.496
[22,   100] loss: 1.375 acc: 50.53 time: 0.89
[22,   200] loss: 1.386 acc: 49.94 time: 0.68
[22,   300] loss: 1.375 acc: 49.93 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 46.49 %
Average loss on the 10000 test images: 1.494
[23,   100] loss: 1.378 acc: 49.83 time: 0.89
[23,   200] loss: 1.375 acc: 50.38 time: 0.67
[23,   300] loss: 1.370 acc: 50.34 time: 0.69
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.494
[24,   100] loss: 1.377 acc: 50.17 time: 0.88
[24,   200] loss: 1.386 acc: 49.80 time: 0.67
[24,   300] loss: 1.382 acc: 50.34 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.63 %
Average loss on the 10000 test images: 1.494
```

```
[25,   100] loss: 1.387 acc: 49.81 time: 0.89
[25,   200] loss: 1.375 acc: 50.27 time: 0.68
[25,   300] loss: 1.363 acc: 50.84 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.494
[26,   100] loss: 1.368 acc: 50.57 time: 0.89
[26,   200] loss: 1.381 acc: 49.98 time: 0.67
[26,   300] loss: 1.385 acc: 49.95 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.58 %
Average loss on the 10000 test images: 1.494
[27,   100] loss: 1.382 acc: 50.17 time: 0.95
[27,   200] loss: 1.377 acc: 49.95 time: 0.72
[27,   300] loss: 1.370 acc: 50.75 time: 0.71
TESTING:
Accuracy of the network on the 10000 test images: 46.56 %
Average loss on the 10000 test images: 1.495
[28,   100] loss: 1.377 acc: 49.94 time: 0.98
[28,   200] loss: 1.375 acc: 50.20 time: 0.66
[28,   300] loss: 1.369 acc: 50.65 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.54 %
Average loss on the 10000 test images: 1.495
[29,   100] loss: 1.377 acc: 50.21 time: 0.89
[29,   200] loss: 1.372 acc: 50.36 time: 0.67
[29,   300] loss: 1.373 acc: 49.98 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.55 %
Average loss on the 10000 test images: 1.494
[30,   100] loss: 1.380 acc: 50.57 time: 1.01
[30,   200] loss: 1.366 acc: 50.46 time: 0.71
[30,   300] loss: 1.381 acc: 50.02 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 46.55 %
Average loss on the 10000 test images: 1.495
[31,   100] loss: 1.380 acc: 50.37 time: 0.88
[31,   200] loss: 1.375 acc: 50.24 time: 0.67
[31,   300] loss: 1.367 acc: 51.20 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.51 %
Average loss on the 10000 test images: 1.495
[32,   100] loss: 1.378 acc: 50.41 time: 0.88
[32,   200] loss: 1.369 acc: 51.02 time: 0.67
[32,   300] loss: 1.383 acc: 49.96 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.53 %
Average loss on the 10000 test images: 1.495
```

```
[33,   100] loss: 1.369 acc: 50.55 time: 0.89
[33,   200] loss: 1.376 acc: 50.27 time: 0.68
[33,   300] loss: 1.383 acc: 50.08 time: 0.68
TESTING:
Accuracy of the network on the 10000 test images: 46.54 %
Average loss on the 10000 test images: 1.495
[34,   100] loss: 1.365 acc: 50.85 time: 1.00
[34,   200] loss: 1.366 acc: 50.66 time: 0.67
[34,   300] loss: 1.376 acc: 50.28 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.53 %
Average loss on the 10000 test images: 1.495
[35,   100] loss: 1.367 acc: 50.72 time: 0.88
[35,   200] loss: 1.375 acc: 50.09 time: 0.65
[35,   300] loss: 1.368 acc: 50.45 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.53 %
Average loss on the 10000 test images: 1.495
[36,   100] loss: 1.367 acc: 50.95 time: 0.87
[36,   200] loss: 1.372 acc: 50.34 time: 0.66
[36,   300] loss: 1.380 acc: 49.89 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.52 %
Average loss on the 10000 test images: 1.495
[37,   100] loss: 1.357 acc: 50.91 time: 0.86
[37,   200] loss: 1.379 acc: 50.09 time: 0.66
[37,   300] loss: 1.384 acc: 50.56 time: 0.65
TESTING:
Accuracy of the network on the 10000 test images: 46.51 %
Average loss on the 10000 test images: 1.495
[38,   100] loss: 1.380 acc: 49.80 time: 0.88
[38,   200] loss: 1.371 acc: 50.77 time: 0.67
[38,   300] loss: 1.364 acc: 50.94 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.48 %
Average loss on the 10000 test images: 1.495
[39,   100] loss: 1.367 acc: 50.22 time: 0.88
[39,   200] loss: 1.378 acc: 50.35 time: 0.68
[39,   300] loss: 1.370 acc: 50.67 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.54 %
Average loss on the 10000 test images: 1.495
[40,   100] loss: 1.373 acc: 50.59 time: 0.88
[40,   200] loss: 1.376 acc: 50.27 time: 0.66
[40,   300] loss: 1.377 acc: 50.38 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.48 %
Average loss on the 10000 test images: 1.495
```

```
[41,   100] loss: 1.380 acc: 50.52 time: 0.88
[41,   200] loss: 1.364 acc: 50.64 time: 0.67
[41,   300] loss: 1.372 acc: 50.16 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.49 %
Average loss on the 10000 test images: 1.495
[42,   100] loss: 1.378 acc: 50.04 time: 0.88
[42,   200] loss: 1.362 acc: 50.92 time: 0.68
[42,   300] loss: 1.368 acc: 50.44 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[43,   100] loss: 1.375 acc: 50.31 time: 0.98
[43,   200] loss: 1.375 acc: 50.23 time: 0.69
[43,   300] loss: 1.365 acc: 50.38 time: 0.69
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[44,   100] loss: 1.377 acc: 50.35 time: 0.97
[44,   200] loss: 1.370 acc: 50.53 time: 0.73
[44,   300] loss: 1.367 acc: 50.37 time: 0.72
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[45,   100] loss: 1.370 acc: 50.14 time: 0.89
[45,   200] loss: 1.382 acc: 50.34 time: 0.68
[45,   300] loss: 1.372 acc: 50.41 time: 0.67
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[46,   100] loss: 1.365 acc: 51.02 time: 0.88
[46,   200] loss: 1.381 acc: 49.58 time: 0.67
[46,   300] loss: 1.373 acc: 50.27 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.49 %
Average loss on the 10000 test images: 1.495
[47,   100] loss: 1.365 acc: 50.60 time: 0.87
[47,   200] loss: 1.377 acc: 50.20 time: 0.67
[47,   300] loss: 1.365 acc: 50.38 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[48,   100] loss: 1.368 acc: 50.42 time: 0.87
[48,   200] loss: 1.369 acc: 50.25 time: 0.67
[48,   300] loss: 1.368 acc: 50.59 time: 0.66
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
```

```
[49,   100] loss: 1.374 acc: 50.48 time: 0.95
[49,   200] loss: 1.379 acc: 50.34 time: 0.73
[49,   300] loss: 1.360 acc: 50.34 time: 0.72
TESTING:
Accuracy of the network on the 10000 test images: 46.50 %
Average loss on the 10000 test images: 1.495
[50,   100] loss: 1.364 acc: 50.84 time: 0.99
[50,   200] loss: 1.376 acc: 50.65 time: 0.79
[50,   300] loss: 1.369 acc: 50.40 time: 0.82
TESTING:
Accuracy of the network on the 10000 test images: 46.49 %
Average loss on the 10000 test images: 1.495
Finished Training
[24.48, 38.16, 42.29, 43.47, 46.64]
```

[ ]:
```
### Stage 2: Supervised Training
data_num = [20, 100, 400, 1000, 5000]
accuracy_st = []

for num in data_num:
    ## Load resnet Model
    net = resnet18(num_classes=10).to(device)
    net.train()

    ## mini-trainset
    mini_set = []
    sample_dict = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0}
    idx = 0
    while sum(list(sample_dict.values())) < num:
        temp = trainset[idx]
        if sample_dict[int(temp[3])] >= num / 10:               # class already␣
↪full
            idx += 1
        else:
            mini_set.append(temp)
            idx += 1
            sample_dict[int(temp[3])] += 1

    ## make dataloader
    trainloader = torch.utils.data.DataLoader(mini_set,␣
↪batch_size=min(len(mini_set), batch_size), shuffle=True, num_workers=2)

    print("\nStart Finetuning with {} samples perclass.".format(num))
    test_acc_max = train(net, criterion, optimizer, num_epochs=50,␣
↪decay_epochs=10, init_lr=0.001, task='classification')
    torch.save(net, 'finetune_pretrain_{}.pt'.format(num))
    accuracy_st.append(test_acc_max)
```

```
print(accuracy_st)
```

```
Start Finetuning with 20 samples perclass.
TESTING:
Accuracy of the network on the 10000 test images: 14.72 %
Average loss on the 10000 test images: 2.426
TESTING:
Accuracy of the network on the 10000 test images: 16.24 %
Average loss on the 10000 test images: 2.524
TESTING:
Accuracy of the network on the 10000 test images: 16.57 %
Average loss on the 10000 test images: 2.712
TESTING:
Accuracy of the network on the 10000 test images: 16.28 %
Average loss on the 10000 test images: 2.912
TESTING:
Accuracy of the network on the 10000 test images: 16.43 %
Average loss on the 10000 test images: 3.097
TESTING:
Accuracy of the network on the 10000 test images: 16.44 %
Average loss on the 10000 test images: 3.261
TESTING:
Accuracy of the network on the 10000 test images: 16.31 %
Average loss on the 10000 test images: 3.408
TESTING:
Accuracy of the network on the 10000 test images: 16.13 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 16.15 %
Average loss on the 10000 test images: 3.655
TESTING:
Accuracy of the network on the 10000 test images: 16.09 %
Average loss on the 10000 test images: 3.757
TESTING:
Accuracy of the network on the 10000 test images: 16.05 %
Average loss on the 10000 test images: 3.847
TESTING:
Accuracy of the network on the 10000 test images: 16.06 %
Average loss on the 10000 test images: 3.855
TESTING:
Accuracy of the network on the 10000 test images: 16.06 %
Average loss on the 10000 test images: 3.862
TESTING:
Accuracy of the network on the 10000 test images: 16.07 %
Average loss on the 10000 test images: 3.869
```

```
TESTING:
Accuracy of the network on the 10000 test images: 16.05 %
Average loss on the 10000 test images: 3.875
TESTING:
Accuracy of the network on the 10000 test images: 16.08 %
Average loss on the 10000 test images: 3.880
TESTING:
Accuracy of the network on the 10000 test images: 16.06 %
Average loss on the 10000 test images: 3.885
TESTING:
Accuracy of the network on the 10000 test images: 16.01 %
Average loss on the 10000 test images: 3.890
TESTING:
Accuracy of the network on the 10000 test images: 15.96 %
Average loss on the 10000 test images: 3.894
TESTING:
Accuracy of the network on the 10000 test images: 15.98 %
Average loss on the 10000 test images: 3.898
TESTING:
Accuracy of the network on the 10000 test images: 15.99 %
Average loss on the 10000 test images: 3.902
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.902
TESTING:
Accuracy of the network on the 10000 test images: 15.97 %
Average loss on the 10000 test images: 3.902
TESTING:
Accuracy of the network on the 10000 test images: 15.98 %
Average loss on the 10000 test images: 3.903
TESTING:
Accuracy of the network on the 10000 test images: 15.98 %
Average loss on the 10000 test images: 3.903
TESTING:
Accuracy of the network on the 10000 test images: 15.99 %
Average loss on the 10000 test images: 3.903
TESTING:
Accuracy of the network on the 10000 test images: 15.99 %
Average loss on the 10000 test images: 3.903
TESTING:
Accuracy of the network on the 10000 test images: 15.99 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
```

```
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 15.98 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 15.99 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.01 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.01 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
```

```
TESTING:
Accuracy of the network on the 10000 test images: 16.01 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
TESTING:
Accuracy of the network on the 10000 test images: 16.00 %
Average loss on the 10000 test images: 3.904
Finished Training

Start Finetuning with 100 samples perclass.
TESTING:
Accuracy of the network on the 10000 test images: 14.90 %
Average loss on the 10000 test images: 2.400
TESTING:
Accuracy of the network on the 10000 test images: 17.28 %
Average loss on the 10000 test images: 2.381
TESTING:
Accuracy of the network on the 10000 test images: 18.83 %
Average loss on the 10000 test images: 2.467
TESTING:
Accuracy of the network on the 10000 test images: 19.81 %
Average loss on the 10000 test images: 2.608
TESTING:
Accuracy of the network on the 10000 test images: 20.38 %
Average loss on the 10000 test images: 2.766
TESTING:
Accuracy of the network on the 10000 test images: 20.63 %
Average loss on the 10000 test images: 2.916
TESTING:
Accuracy of the network on the 10000 test images: 21.11 %
Average loss on the 10000 test images: 3.051
TESTING:
Accuracy of the network on the 10000 test images: 21.00 %
Average loss on the 10000 test images: 3.173
TESTING:
Accuracy of the network on the 10000 test images: 21.13 %
Average loss on the 10000 test images: 3.284
TESTING:
Accuracy of the network on the 10000 test images: 21.37 %
Average loss on the 10000 test images: 3.386
TESTING:
Accuracy of the network on the 10000 test images: 21.42 %
Average loss on the 10000 test images: 3.479
```

```
TESTING:
Accuracy of the network on the 10000 test images: 21.42 %
Average loss on the 10000 test images: 3.487
TESTING:
Accuracy of the network on the 10000 test images: 21.43 %
Average loss on the 10000 test images: 3.495
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.502
TESTING:
Accuracy of the network on the 10000 test images: 21.43 %
Average loss on the 10000 test images: 3.508
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.514
TESTING:
Accuracy of the network on the 10000 test images: 21.45 %
Average loss on the 10000 test images: 3.519
TESTING:
Accuracy of the network on the 10000 test images: 21.43 %
Average loss on the 10000 test images: 3.524
TESTING:
Accuracy of the network on the 10000 test images: 21.43 %
Average loss on the 10000 test images: 3.528
TESTING:
Accuracy of the network on the 10000 test images: 21.45 %
Average loss on the 10000 test images: 3.533
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.536
TESTING:
Accuracy of the network on the 10000 test images: 21.49 %
Average loss on the 10000 test images: 3.537
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.537
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.537
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.538
```

```
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 21.49 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.45 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.45 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.49 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.50 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.47 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
```

```
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.49 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.46 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 21.48 %
Average loss on the 10000 test images: 3.539
Finished Training

Start Finetuning with 400 samples perclass.
TESTING:
Accuracy of the network on the 10000 test images: 20.97 %
Average loss on the 10000 test images: 2.311
TESTING:
Accuracy of the network on the 10000 test images: 24.23 %
Average loss on the 10000 test images: 2.279
TESTING:
Accuracy of the network on the 10000 test images: 25.16 %
Average loss on the 10000 test images: 2.356
TESTING:
Accuracy of the network on the 10000 test images: 25.60 %
Average loss on the 10000 test images: 2.591
TESTING:
Accuracy of the network on the 10000 test images: 25.44 %
Average loss on the 10000 test images: 2.865
TESTING:
Accuracy of the network on the 10000 test images: 26.47 %
Average loss on the 10000 test images: 2.991
TESTING:
Accuracy of the network on the 10000 test images: 26.09 %
Average loss on the 10000 test images: 3.241
TESTING:
Accuracy of the network on the 10000 test images: 27.03 %
Average loss on the 10000 test images: 3.306
```

```
TESTING:
Accuracy of the network on the 10000 test images: 27.84 %
Average loss on the 10000 test images: 3.308
TESTING:
Accuracy of the network on the 10000 test images: 28.89 %
Average loss on the 10000 test images: 3.430
TESTING:
Accuracy of the network on the 10000 test images: 29.74 %
Average loss on the 10000 test images: 3.631
TESTING:
Accuracy of the network on the 10000 test images: 29.81 %
Average loss on the 10000 test images: 3.629
TESTING:
Accuracy of the network on the 10000 test images: 30.15 %
Average loss on the 10000 test images: 3.593
TESTING:
Accuracy of the network on the 10000 test images: 30.22 %
Average loss on the 10000 test images: 3.560
TESTING:
Accuracy of the network on the 10000 test images: 30.04 %
Average loss on the 10000 test images: 3.542
TESTING:
Accuracy of the network on the 10000 test images: 29.84 %
Average loss on the 10000 test images: 3.547
TESTING:
Accuracy of the network on the 10000 test images: 29.86 %
Average loss on the 10000 test images: 3.544
TESTING:
Accuracy of the network on the 10000 test images: 29.81 %
Average loss on the 10000 test images: 3.546
TESTING:
Accuracy of the network on the 10000 test images: 29.75 %
Average loss on the 10000 test images: 3.546
TESTING:
Accuracy of the network on the 10000 test images: 29.82 %
Average loss on the 10000 test images: 3.544
TESTING:
Accuracy of the network on the 10000 test images: 30.06 %
Average loss on the 10000 test images: 3.546
TESTING:
Accuracy of the network on the 10000 test images: 30.07 %
Average loss on the 10000 test images: 3.545
TESTING:
Accuracy of the network on the 10000 test images: 30.07 %
Average loss on the 10000 test images: 3.545
TESTING:
Accuracy of the network on the 10000 test images: 30.02 %
Average loss on the 10000 test images: 3.544
```

```
TESTING:
Accuracy of the network on the 10000 test images: 30.05 %
Average loss on the 10000 test images: 3.543
TESTING:
Accuracy of the network on the 10000 test images: 30.03 %
Average loss on the 10000 test images: 3.542
TESTING:
Accuracy of the network on the 10000 test images: 30.00 %
Average loss on the 10000 test images: 3.542
TESTING:
Accuracy of the network on the 10000 test images: 30.06 %
Average loss on the 10000 test images: 3.542
TESTING:
Accuracy of the network on the 10000 test images: 29.94 %
Average loss on the 10000 test images: 3.541
TESTING:
Accuracy of the network on the 10000 test images: 29.92 %
Average loss on the 10000 test images: 3.540
TESTING:
Accuracy of the network on the 10000 test images: 29.96 %
Average loss on the 10000 test images: 3.540
TESTING:
Accuracy of the network on the 10000 test images: 29.95 %
Average loss on the 10000 test images: 3.540
TESTING:
Accuracy of the network on the 10000 test images: 29.94 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 29.93 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 29.93 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 29.92 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.539
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.91 %
Average loss on the 10000 test images: 3.538
```

```
TESTING:
Accuracy of the network on the 10000 test images: 29.91 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.89 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.91 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.88 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.88 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.90 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.86 %
Average loss on the 10000 test images: 3.538
TESTING:
Accuracy of the network on the 10000 test images: 29.89 %
Average loss on the 10000 test images: 3.538
Finished Training

Start Finetuning with 1000 samples perclass.
TESTING:
Accuracy of the network on the 10000 test images: 27.34 %
Average loss on the 10000 test images: 2.074
TESTING:
Accuracy of the network on the 10000 test images: 31.50 %
Average loss on the 10000 test images: 2.071
TESTING:
Accuracy of the network on the 10000 test images: 33.48 %
Average loss on the 10000 test images: 2.233
TESTING:
Accuracy of the network on the 10000 test images: 33.01 %
Average loss on the 10000 test images: 2.648
TESTING:
Accuracy of the network on the 10000 test images: 32.85 %
Average loss on the 10000 test images: 2.927
```

```
TESTING:
Accuracy of the network on the 10000 test images: 33.24 %
Average loss on the 10000 test images: 3.223
TESTING:
Accuracy of the network on the 10000 test images: 33.33 %
Average loss on the 10000 test images: 3.329
TESTING:
Accuracy of the network on the 10000 test images: 32.61 %
Average loss on the 10000 test images: 3.541
TESTING:
Accuracy of the network on the 10000 test images: 33.11 %
Average loss on the 10000 test images: 3.590
TESTING:
Accuracy of the network on the 10000 test images: 33.69 %
Average loss on the 10000 test images: 3.602
TESTING:
Accuracy of the network on the 10000 test images: 31.72 %
Average loss on the 10000 test images: 3.879
TESTING:
Accuracy of the network on the 10000 test images: 32.48 %
Average loss on the 10000 test images: 3.812
TESTING:
Accuracy of the network on the 10000 test images: 33.34 %
Average loss on the 10000 test images: 3.758
TESTING:
Accuracy of the network on the 10000 test images: 33.46 %
Average loss on the 10000 test images: 3.744
TESTING:
Accuracy of the network on the 10000 test images: 33.71 %
Average loss on the 10000 test images: 3.730
TESTING:
Accuracy of the network on the 10000 test images: 33.76 %
Average loss on the 10000 test images: 3.720
TESTING:
Accuracy of the network on the 10000 test images: 33.90 %
Average loss on the 10000 test images: 3.714
TESTING:
Accuracy of the network on the 10000 test images: 33.97 %
Average loss on the 10000 test images: 3.711
TESTING:
Accuracy of the network on the 10000 test images: 34.02 %
Average loss on the 10000 test images: 3.708
TESTING:
Accuracy of the network on the 10000 test images: 34.00 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.16 %
Average loss on the 10000 test images: 3.705
```

```
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.17 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.16 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.12 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.705
TESTING:
Accuracy of the network on the 10000 test images: 34.12 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.705
```

```
TESTING:
Accuracy of the network on the 10000 test images: 34.12 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.16 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.15 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.12 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.12 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.14 %
Average loss on the 10000 test images: 3.706
TESTING:
Accuracy of the network on the 10000 test images: 34.13 %
Average loss on the 10000 test images: 3.706
Finished Training

Start Finetuning with 5000 samples perclass.
TESTING:
Accuracy of the network on the 10000 test images: 36.88 %
Average loss on the 10000 test images: 1.694
TESTING:
Accuracy of the network on the 10000 test images: 42.84 %
Average loss on the 10000 test images: 1.585
```

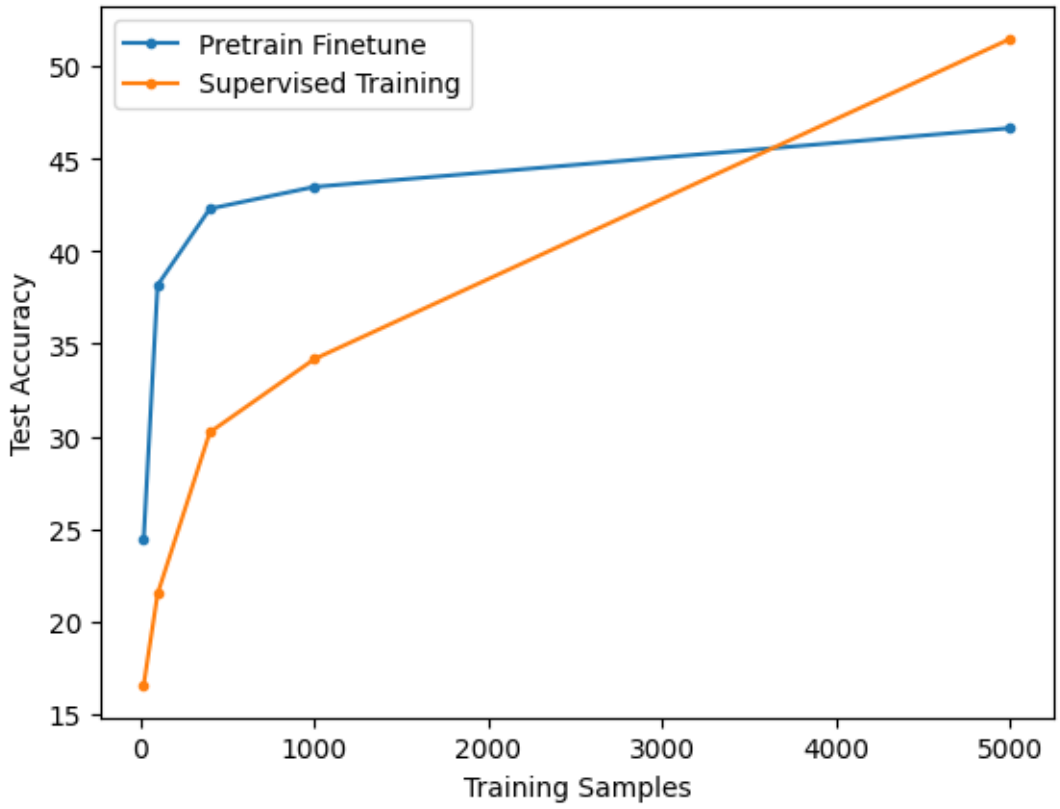
```
TESTING:
Accuracy of the network on the 10000 test images: 43.75 %
Average loss on the 10000 test images: 1.546
TESTING:
Accuracy of the network on the 10000 test images: 44.78 %
Average loss on the 10000 test images: 1.543
TESTING:
Accuracy of the network on the 10000 test images: 45.57 %
Average loss on the 10000 test images: 1.611
TESTING:
Accuracy of the network on the 10000 test images: 48.40 %
Average loss on the 10000 test images: 1.567
TESTING:
Accuracy of the network on the 10000 test images: 47.61 %
Average loss on the 10000 test images: 1.690
TESTING:
Accuracy of the network on the 10000 test images: 47.43 %
Average loss on the 10000 test images: 1.629
TESTING:
Accuracy of the network on the 10000 test images: 47.31 %
Average loss on the 10000 test images: 1.884
TESTING:
Accuracy of the network on the 10000 test images: 49.11 %
Average loss on the 10000 test images: 1.837
TESTING:
Accuracy of the network on the 10000 test images: 46.89 %
Average loss on the 10000 test images: 1.883
TESTING:
Accuracy of the network on the 10000 test images: 51.34 %
Average loss on the 10000 test images: 1.765
TESTING:
Accuracy of the network on the 10000 test images: 51.46 %
Average loss on the 10000 test images: 1.847
TESTING:
Accuracy of the network on the 10000 test images: 51.45 %
Average loss on the 10000 test images: 1.938
TESTING:
Accuracy of the network on the 10000 test images: 51.15 %
Average loss on the 10000 test images: 2.022
TESTING:
Accuracy of the network on the 10000 test images: 51.24 %
Average loss on the 10000 test images: 2.114
TESTING:
Accuracy of the network on the 10000 test images: 50.50 %
Average loss on the 10000 test images: 2.227
TESTING:
Accuracy of the network on the 10000 test images: 50.66 %
Average loss on the 10000 test images: 2.303
```

```
TESTING:
Accuracy of the network on the 10000 test images: 50.46 %
Average loss on the 10000 test images: 2.387
TESTING:
Accuracy of the network on the 10000 test images: 50.31 %
Average loss on the 10000 test images: 2.455
TESTING:
Accuracy of the network on the 10000 test images: 50.82 %
Average loss on the 10000 test images: 2.505
TESTING:
Accuracy of the network on the 10000 test images: 50.58 %
Average loss on the 10000 test images: 2.526
TESTING:
Accuracy of the network on the 10000 test images: 50.77 %
Average loss on the 10000 test images: 2.520
TESTING:
Accuracy of the network on the 10000 test images: 50.70 %
Average loss on the 10000 test images: 2.522
TESTING:
Accuracy of the network on the 10000 test images: 50.73 %
Average loss on the 10000 test images: 2.523
TESTING:
Accuracy of the network on the 10000 test images: 50.72 %
Average loss on the 10000 test images: 2.520
TESTING:
Accuracy of the network on the 10000 test images: 50.71 %
Average loss on the 10000 test images: 2.525
TESTING:
Accuracy of the network on the 10000 test images: 50.75 %
Average loss on the 10000 test images: 2.529
TESTING:
Accuracy of the network on the 10000 test images: 50.81 %
Average loss on the 10000 test images: 2.534
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.551
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.561
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.562
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.562
TESTING:
Accuracy of the network on the 10000 test images: 50.55 %
Average loss on the 10000 test images: 2.562
```

```
TESTING:
Accuracy of the network on the 10000 test images: 50.57 %
Average loss on the 10000 test images: 2.559
TESTING:
Accuracy of the network on the 10000 test images: 50.57 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.56 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.53 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.61 %
Average loss on the 10000 test images: 2.561
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.561
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.59 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.61 %
Average loss on the 10000 test images: 2.560
TESTING:
Accuracy of the network on the 10000 test images: 50.60 %
Average loss on the 10000 test images: 2.560
```

```
Finished Training
[16.57, 21.5, 30.22, 34.17, 51.46]
```

[ ]:
```
### Plot Result
import matplotlib.pyplot as plt

plt.xlabel('Training Samples')
plt.ylabel('Test Accuracy')
plt.plot(data_num, accuracy, marker='o', markersize=3)
plt.plot(data_num, accuracy_st, marker='o', markersize=3)
plt.legend(["Pretrain Finetune", "Supervised Training"])
plt.show()
```



9 Extra Credit 2: More Advanced Model

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]: 'cuda'

[ ]:
```
### Load Model for Pretrain

import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet50

net = resnet50(num_classes=4)
net = net.to(device)
```

[ ]:
```
import torch.optim as optim

# TODO: Define criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(net.parameters(), lr=init_lr, eps=1e-08,␣
 ↪weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(net.parameters(), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        net.train()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(trainloader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
 ↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
 ↪to(device)
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()
```

```
            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
 ↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
 ↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
 ↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
 ↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
 ↪the evaluation mode.
        net.eval()
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
### Pretrain
train(net, criterion, optimizer, num_epochs=45, decay_epochs=15, init_lr=0.001,␣
 ↪task='rotation')

# Save the model
torch.save(net.state_dict(), './net_pretrain.pth')
```

```
[1,   100] loss: 1.517 acc: 32.38 time: 2.82
[1,   200] loss: 1.284 acc: 42.85 time: 2.64
[1,   300] loss: 1.249 acc: 46.69 time: 2.59
TESTING:
```

```
Accuracy of the network on the 10000 test images: 49.98 %
Average loss on the 10000 test images: 1.124
[2,   100] loss: 1.148 acc: 50.00 time: 2.87
[2,   200] loss: 1.119 acc: 51.00 time: 2.65
[2,   300] loss: 1.106 acc: 52.80 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 53.91 %
Average loss on the 10000 test images: 1.070
[3,   100] loss: 1.073 acc: 53.73 time: 2.82
[3,   200] loss: 1.055 acc: 54.45 time: 2.65
[3,   300] loss: 1.065 acc: 54.20 time: 2.66
TESTING:
Accuracy of the network on the 10000 test images: 55.37 %
Average loss on the 10000 test images: 1.039
[4,   100] loss: 1.040 acc: 55.13 time: 2.84
[4,   200] loss: 1.028 acc: 56.00 time: 2.67
[4,   300] loss: 1.009 acc: 56.92 time: 2.65
TESTING:
Accuracy of the network on the 10000 test images: 58.13 %
Average loss on the 10000 test images: 0.981
[5,   100] loss: 0.993 acc: 58.36 time: 2.78
[5,   200] loss: 0.990 acc: 58.29 time: 2.58
[5,   300] loss: 0.987 acc: 58.55 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 57.04 %
Average loss on the 10000 test images: 0.996
[6,   100] loss: 0.969 acc: 59.44 time: 2.90
[6,   200] loss: 0.981 acc: 58.88 time: 2.61
[6,   300] loss: 0.964 acc: 59.97 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 60.41 %
Average loss on the 10000 test images: 0.931
[7,   100] loss: 0.957 acc: 60.28 time: 2.84
[7,   200] loss: 0.950 acc: 60.16 time: 2.67
[7,   300] loss: 0.937 acc: 60.88 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 62.06 %
Average loss on the 10000 test images: 0.916
[8,   100] loss: 0.924 acc: 61.27 time: 2.84
[8,   200] loss: 0.932 acc: 61.16 time: 2.63
[8,   300] loss: 0.945 acc: 60.66 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 61.35 %
Average loss on the 10000 test images: 0.932
[9,   100] loss: 0.913 acc: 62.33 time: 2.80
[9,   200] loss: 0.924 acc: 61.49 time: 2.59
[9,   300] loss: 0.927 acc: 61.94 time: 2.61
TESTING:
```

```
Accuracy of the network on the 10000 test images: 62.36 %
Average loss on the 10000 test images: 0.893
[10,   100] loss: 0.914 acc: 61.93 time: 2.87
[10,   200] loss: 0.890 acc: 63.27 time: 2.61
[10,   300] loss: 0.886 acc: 63.40 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 63.18 %
Average loss on the 10000 test images: 0.890
[11,   100] loss: 0.893 acc: 63.51 time: 2.78
[11,   200] loss: 0.890 acc: 63.61 time: 2.57
[11,   300] loss: 0.891 acc: 63.33 time: 2.56
TESTING:
Accuracy of the network on the 10000 test images: 66.24 %
Average loss on the 10000 test images: 0.829
[12,   100] loss: 0.877 acc: 63.29 time: 2.79
[12,   200] loss: 0.878 acc: 64.15 time: 2.57
[12,   300] loss: 0.883 acc: 64.16 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 64.22 %
Average loss on the 10000 test images: 0.864
[13,   100] loss: 0.860 acc: 64.58 time: 2.83
[13,   200] loss: 0.861 acc: 64.27 time: 2.59
[13,   300] loss: 0.855 acc: 65.14 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 65.52 %
Average loss on the 10000 test images: 0.849
[14,   100] loss: 0.858 acc: 65.03 time: 2.87
[14,   200] loss: 0.847 acc: 65.62 time: 2.68
[14,   300] loss: 0.851 acc: 65.05 time: 2.61
TESTING:
Accuracy of the network on the 10000 test images: 66.38 %
Average loss on the 10000 test images: 0.824
[15,   100] loss: 0.841 acc: 65.82 time: 2.82
[15,   200] loss: 0.833 acc: 66.42 time: 2.60
[15,   300] loss: 0.833 acc: 66.23 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 67.60 %
Average loss on the 10000 test images: 0.799
[16,   100] loss: 0.813 acc: 67.05 time: 2.82
[16,   200] loss: 0.815 acc: 66.77 time: 2.59
[16,   300] loss: 0.816 acc: 66.91 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 66.10 %
Average loss on the 10000 test images: 0.847
[17,   100] loss: 0.764 acc: 69.17 time: 2.85
[17,   200] loss: 0.729 acc: 71.17 time: 2.58
[17,   300] loss: 0.720 acc: 71.01 time: 2.58
TESTING:
```

```
Accuracy of the network on the 10000 test images: 73.22 %
Average loss on the 10000 test images: 0.674
[18,   100] loss: 0.693 acc: 72.29 time: 2.87
[18,   200] loss: 0.690 acc: 72.16 time: 2.66
[18,   300] loss: 0.696 acc: 72.20 time: 2.66
TESTING:
Accuracy of the network on the 10000 test images: 73.69 %
Average loss on the 10000 test images: 0.658
[19,   100] loss: 0.685 acc: 72.86 time: 2.83
[19,   200] loss: 0.674 acc: 73.79 time: 2.58
[19,   300] loss: 0.685 acc: 73.09 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 74.30 %
Average loss on the 10000 test images: 0.653
[20,   100] loss: 0.659 acc: 74.00 time: 2.81
[20,   200] loss: 0.675 acc: 73.02 time: 2.58
[20,   300] loss: 0.672 acc: 73.13 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 74.65 %
Average loss on the 10000 test images: 0.639
[21,   100] loss: 0.656 acc: 73.91 time: 2.81
[21,   200] loss: 0.657 acc: 74.16 time: 2.61
[21,   300] loss: 0.646 acc: 74.92 time: 2.58
TESTING:
Accuracy of the network on the 10000 test images: 75.22 %
Average loss on the 10000 test images: 0.629
[22,   100] loss: 0.655 acc: 74.09 time: 2.81
[22,   200] loss: 0.643 acc: 74.69 time: 2.58
[22,   300] loss: 0.641 acc: 74.50 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 75.19 %
Average loss on the 10000 test images: 0.633
[23,   100] loss: 0.640 acc: 75.15 time: 2.81
[23,   200] loss: 0.652 acc: 74.21 time: 2.61
[23,   300] loss: 0.632 acc: 74.93 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 76.09 %
Average loss on the 10000 test images: 0.613
[24,   100] loss: 0.634 acc: 74.92 time: 2.87
[24,   200] loss: 0.640 acc: 74.91 time: 2.61
[24,   300] loss: 0.632 acc: 75.18 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 75.55 %
Average loss on the 10000 test images: 0.615
[25,   100] loss: 0.624 acc: 75.27 time: 2.88
[25,   200] loss: 0.630 acc: 75.42 time: 2.61
[25,   300] loss: 0.614 acc: 75.97 time: 2.59
TESTING:
```

```
Accuracy of the network on the 10000 test images: 75.58 %
Average loss on the 10000 test images: 0.622
[26,   100] loss: 0.615 acc: 76.08 time: 2.83
[26,   200] loss: 0.611 acc: 75.91 time: 2.60
[26,   300] loss: 0.617 acc: 76.18 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 76.44 %
Average loss on the 10000 test images: 0.593
[27,   100] loss: 0.612 acc: 75.66 time: 2.87
[27,   200] loss: 0.624 acc: 75.35 time: 2.59
[27,   300] loss: 0.604 acc: 76.50 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 76.25 %
Average loss on the 10000 test images: 0.603
[28,   100] loss: 0.599 acc: 76.42 time: 2.88
[28,   200] loss: 0.600 acc: 76.62 time: 2.62
[28,   300] loss: 0.602 acc: 76.16 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 77.21 %
Average loss on the 10000 test images: 0.585
[29,   100] loss: 0.589 acc: 76.82 time: 2.85
[29,   200] loss: 0.595 acc: 77.02 time: 2.66
[29,   300] loss: 0.602 acc: 76.38 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 77.18 %
Average loss on the 10000 test images: 0.583
[30,   100] loss: 0.592 acc: 77.17 time: 2.86
[30,   200] loss: 0.587 acc: 77.41 time: 2.62
[30,   300] loss: 0.597 acc: 76.94 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 76.72 %
Average loss on the 10000 test images: 0.593
[31,   100] loss: 0.592 acc: 77.03 time: 2.86
[31,   200] loss: 0.595 acc: 76.72 time: 2.61
[31,   300] loss: 0.581 acc: 77.18 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 77.43 %
Average loss on the 10000 test images: 0.581
[32,   100] loss: 0.570 acc: 77.63 time: 2.84
[32,   200] loss: 0.567 acc: 77.66 time: 2.64
[32,   300] loss: 0.564 acc: 78.20 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 78.30 %
Average loss on the 10000 test images: 0.562
[33,   100] loss: 0.562 acc: 78.45 time: 2.88
[33,   200] loss: 0.569 acc: 77.95 time: 2.61
[33,   300] loss: 0.562 acc: 78.24 time: 2.60
TESTING:
```

```
Accuracy of the network on the 10000 test images: 78.76 %
Average loss on the 10000 test images: 0.547
[34,   100] loss: 0.565 acc: 77.93 time: 2.81
[34,   200] loss: 0.557 acc: 78.38 time: 2.57
[34,   300] loss: 0.553 acc: 78.58 time: 2.57
TESTING:
Accuracy of the network on the 10000 test images: 78.26 %
Average loss on the 10000 test images: 0.559
[35,   100] loss: 0.573 acc: 77.78 time: 2.82
[35,   200] loss: 0.562 acc: 78.03 time: 2.59
[35,   300] loss: 0.556 acc: 78.57 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 78.38 %
Average loss on the 10000 test images: 0.557
[36,   100] loss: 0.561 acc: 78.16 time: 2.85
[36,   200] loss: 0.559 acc: 78.27 time: 2.61
[36,   300] loss: 0.557 acc: 78.68 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 79.17 %
Average loss on the 10000 test images: 0.543
[37,   100] loss: 0.555 acc: 78.40 time: 2.86
[37,   200] loss: 0.554 acc: 78.77 time: 2.61
[37,   300] loss: 0.558 acc: 78.56 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 78.51 %
Average loss on the 10000 test images: 0.557
[38,   100] loss: 0.538 acc: 79.16 time: 2.87
[38,   200] loss: 0.549 acc: 78.41 time: 2.81
[38,   300] loss: 0.562 acc: 78.05 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 78.39 %
Average loss on the 10000 test images: 0.556
[39,   100] loss: 0.546 acc: 79.24 time: 2.88
[39,   200] loss: 0.548 acc: 78.62 time: 2.68
[39,   300] loss: 0.553 acc: 78.88 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 79.18 %
Average loss on the 10000 test images: 0.548
[40,   100] loss: 0.554 acc: 78.34 time: 2.85
[40,   200] loss: 0.548 acc: 79.14 time: 2.66
[40,   300] loss: 0.560 acc: 78.09 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 78.72 %
Average loss on the 10000 test images: 0.550
[41,   100] loss: 0.552 acc: 78.74 time: 2.85
[41,   200] loss: 0.554 acc: 78.62 time: 2.69
[41,   300] loss: 0.555 acc: 78.18 time: 2.63
TESTING:
```

```
Accuracy of the network on the 10000 test images: 79.27 %
Average loss on the 10000 test images: 0.544
[42,   100] loss: 0.548 acc: 78.84 time: 2.88
[42,   200] loss: 0.550 acc: 78.95 time: 2.61
[42,   300] loss: 0.554 acc: 78.66 time: 2.59
TESTING:
Accuracy of the network on the 10000 test images: 78.80 %
Average loss on the 10000 test images: 0.549
[43,   100] loss: 0.556 acc: 78.40 time: 2.91
[43,   200] loss: 0.530 acc: 79.50 time: 2.66
[43,   300] loss: 0.549 acc: 78.96 time: 2.61
TESTING:
Accuracy of the network on the 10000 test images: 79.01 %
Average loss on the 10000 test images: 0.542
[44,   100] loss: 0.543 acc: 79.05 time: 2.88
[44,   200] loss: 0.543 acc: 78.97 time: 2.60
[44,   300] loss: 0.552 acc: 78.48 time: 2.60
TESTING:
Accuracy of the network on the 10000 test images: 79.33 %
Average loss on the 10000 test images: 0.540
[45,   100] loss: 0.542 acc: 79.43 time: 2.89
[45,   200] loss: 0.546 acc: 78.96 time: 2.63
[45,   300] loss: 0.551 acc: 78.91 time: 2.61
TESTING:
Accuracy of the network on the 10000 test images: 78.78 %
Average loss on the 10000 test images: 0.544
Finished Training
```

[ ]:
```python
### Load Model for Finetune
net = resnet50(num_classes=4).to(device)
net.load_state_dict(torch.load('./net_pretrain.pth'))
net.fc = nn.Linear(in_features=2048, out_features=10, bias=True).to(device)
print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
```

```
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
```

```
    (3): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=10, bias=True)
)
```

[ ]:
```python
## unfreeze layer4 and fc
for name, param in net.named_parameters():
    if "layer4" in name:
        param.requires_grad = True
    elif "fc" in name:
```

```
        param.requires_grad = True
    else:
        param.requires_grad = False

# Print all the trainable parameters
params_to_update = net.parameters()
print("Params to learn:")
params_to_update = []
for name,param in net.named_parameters():
    if param.requires_grad == True:
        params_to_update.append(param)
        print("\t",name)
```

```
Params to learn:
         layer4.0.conv1.weight
         layer4.0.bn1.weight
         layer4.0.bn1.bias
         layer4.0.conv2.weight
         layer4.0.bn2.weight
         layer4.0.bn2.bias
         layer4.0.conv3.weight
         layer4.0.bn3.weight
         layer4.0.bn3.bias
         layer4.0.downsample.0.weight
         layer4.0.downsample.1.weight
         layer4.0.downsample.1.bias
         layer4.1.conv1.weight
         layer4.1.bn1.weight
         layer4.1.bn1.bias
         layer4.1.conv2.weight
         layer4.1.bn2.weight
         layer4.1.bn2.bias
         layer4.1.conv3.weight
         layer4.1.bn3.weight
         layer4.1.bn3.bias
         layer4.2.conv1.weight
         layer4.2.bn1.weight
         layer4.2.bn1.bias
         layer4.2.conv2.weight
         layer4.2.bn2.weight
         layer4.2.bn2.bias
         layer4.2.conv3.weight
         layer4.2.bn3.weight
         layer4.2.bn3.bias
         fc.weight
         fc.bias
```

[ ]:
```
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
### Finetune
def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, eps=1e-08, weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(trainloader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
 ↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
 ↪to(device)
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 100
            running_loss += loss.item()
```

```
            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
train(net, criterion, optimizer, num_epochs=30, decay_epochs=10, init_lr=0.001,␣
↪task='classification')
torch.save(net, 'finetune_pretrain.pt')
```

```
[1,   100] loss: 1.598 acc: 41.88 time: 2.81
[1,   200] loss: 1.215 acc: 55.23 time: 2.67
[1,   300] loss: 1.124 acc: 59.83 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 61.79 %
Average loss on the 10000 test images: 1.064
[2,   100] loss: 1.027 acc: 62.80 time: 2.78
[2,   200] loss: 1.001 acc: 64.54 time: 2.70
[2,   300] loss: 0.967 acc: 65.66 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 64.80 %
Average loss on the 10000 test images: 0.984
[3,   100] loss: 0.927 acc: 67.65 time: 2.80
[3,   200] loss: 0.931 acc: 67.14 time: 2.69
[3,   300] loss: 0.917 acc: 68.09 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 68.12 %
Average loss on the 10000 test images: 0.940
[4,   100] loss: 0.881 acc: 68.95 time: 2.79
[4,   200] loss: 0.878 acc: 69.05 time: 2.70
[4,   300] loss: 0.859 acc: 69.84 time: 2.64
```

```
TESTING:
Accuracy of the network on the 10000 test images: 67.80 %
Average loss on the 10000 test images: 0.920
[5,   100] loss: 0.850 acc: 70.59 time: 2.78
[5,   200] loss: 0.834 acc: 70.78 time: 2.67
[5,   300] loss: 0.834 acc: 71.34 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 69.15 %
Average loss on the 10000 test images: 0.893
[6,   100] loss: 0.803 acc: 72.32 time: 2.80
[6,   200] loss: 0.830 acc: 71.15 time: 2.68
[6,   300] loss: 0.817 acc: 71.62 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 70.51 %
Average loss on the 10000 test images: 0.876
[7,   100] loss: 0.794 acc: 72.36 time: 2.80
[7,   200] loss: 0.787 acc: 72.48 time: 2.69
[7,   300] loss: 0.796 acc: 72.22 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 70.94 %
Average loss on the 10000 test images: 0.840
[8,   100] loss: 0.771 acc: 73.50 time: 2.80
[8,   200] loss: 0.777 acc: 73.03 time: 2.67
[8,   300] loss: 0.780 acc: 72.74 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 71.57 %
Average loss on the 10000 test images: 0.825
[9,   100] loss: 0.764 acc: 73.41 time: 2.80
[9,   200] loss: 0.764 acc: 73.76 time: 2.69
[9,   300] loss: 0.757 acc: 73.97 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 71.99 %
Average loss on the 10000 test images: 0.833
[10,   100] loss: 0.747 acc: 74.11 time: 2.87
[10,   200] loss: 0.757 acc: 73.45 time: 2.65
[10,   300] loss: 0.743 acc: 74.26 time: 2.61
TESTING:
Accuracy of the network on the 10000 test images: 71.75 %
Average loss on the 10000 test images: 0.837
[11,   100] loss: 0.718 acc: 74.78 time: 2.83
[11,   200] loss: 0.743 acc: 74.20 time: 2.65
[11,   300] loss: 0.740 acc: 74.35 time: 2.61
TESTING:
Accuracy of the network on the 10000 test images: 72.00 %
Average loss on the 10000 test images: 0.823
[12,   100] loss: 0.688 acc: 76.35 time: 2.83
[12,   200] loss: 0.650 acc: 77.41 time: 2.66
[12,   300] loss: 0.648 acc: 77.66 time: 2.62
```

```
TESTING:
Accuracy of the network on the 10000 test images: 74.33 %
Average loss on the 10000 test images: 0.753
[13,   100] loss: 0.638 acc: 77.91 time: 2.81
[13,   200] loss: 0.633 acc: 77.97 time: 2.69
[13,   300] loss: 0.633 acc: 77.66 time: 2.69
TESTING:
Accuracy of the network on the 10000 test images: 74.62 %
Average loss on the 10000 test images: 0.746
[14,   100] loss: 0.618 acc: 79.04 time: 2.79
[14,   200] loss: 0.615 acc: 78.65 time: 2.67
[14,   300] loss: 0.623 acc: 78.65 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 75.06 %
Average loss on the 10000 test images: 0.739
[15,   100] loss: 0.604 acc: 79.26 time: 2.79
[15,   200] loss: 0.600 acc: 79.19 time: 2.68
[15,   300] loss: 0.615 acc: 78.52 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 75.03 %
Average loss on the 10000 test images: 0.732
[16,   100] loss: 0.604 acc: 79.11 time: 2.80
[16,   200] loss: 0.593 acc: 79.22 time: 2.69
[16,   300] loss: 0.607 acc: 79.04 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 75.46 %
Average loss on the 10000 test images: 0.724
[17,   100] loss: 0.605 acc: 78.82 time: 2.80
[17,   200] loss: 0.583 acc: 79.84 time: 2.68
[17,   300] loss: 0.609 acc: 78.95 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 75.52 %
Average loss on the 10000 test images: 0.726
[18,   100] loss: 0.591 acc: 79.38 time: 2.79
[18,   200] loss: 0.582 acc: 79.52 time: 2.68
[18,   300] loss: 0.587 acc: 80.09 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 75.50 %
Average loss on the 10000 test images: 0.720
[19,   100] loss: 0.583 acc: 79.62 time: 2.84
[19,   200] loss: 0.591 acc: 79.57 time: 2.67
[19,   300] loss: 0.588 acc: 79.86 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 75.84 %
Average loss on the 10000 test images: 0.712
[20,   100] loss: 0.582 acc: 79.77 time: 2.84
[20,   200] loss: 0.580 acc: 80.02 time: 2.69
[20,   300] loss: 0.590 acc: 79.55 time: 2.66
```

```
TESTING:
Accuracy of the network on the 10000 test images: 75.92 %
Average loss on the 10000 test images: 0.713
[21,   100] loss: 0.580 acc: 80.02 time: 2.80
[21,   200] loss: 0.573 acc: 80.43 time: 2.68
[21,   300] loss: 0.582 acc: 79.74 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 75.53 %
Average loss on the 10000 test images: 0.716
[22,   100] loss: 0.559 acc: 80.65 time: 2.81
[22,   200] loss: 0.578 acc: 80.22 time: 2.68
[22,   300] loss: 0.553 acc: 80.79 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 75.98 %
Average loss on the 10000 test images: 0.708
[23,   100] loss: 0.551 acc: 81.07 time: 2.81
[23,   200] loss: 0.557 acc: 80.89 time: 2.67
[23,   300] loss: 0.552 acc: 80.86 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 76.12 %
Average loss on the 10000 test images: 0.706
[24,   100] loss: 0.554 acc: 81.12 time: 2.93
[24,   200] loss: 0.570 acc: 80.28 time: 2.63
[24,   300] loss: 0.556 acc: 80.70 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 76.17 %
Average loss on the 10000 test images: 0.706
[25,   100] loss: 0.554 acc: 80.77 time: 2.83
[25,   200] loss: 0.557 acc: 80.72 time: 2.68
[25,   300] loss: 0.563 acc: 80.60 time: 2.62
TESTING:
Accuracy of the network on the 10000 test images: 76.15 %
Average loss on the 10000 test images: 0.705
[26,   100] loss: 0.553 acc: 80.58 time: 2.83
[26,   200] loss: 0.566 acc: 80.66 time: 2.70
[26,   300] loss: 0.546 acc: 81.09 time: 2.65
TESTING:
Accuracy of the network on the 10000 test images: 76.17 %
Average loss on the 10000 test images: 0.703
[27,   100] loss: 0.552 acc: 80.55 time: 2.80
[27,   200] loss: 0.541 acc: 81.27 time: 2.68
[27,   300] loss: 0.546 acc: 80.98 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 76.09 %
Average loss on the 10000 test images: 0.703
[28,   100] loss: 0.561 acc: 80.36 time: 2.78
[28,   200] loss: 0.556 acc: 80.80 time: 2.66
[28,   300] loss: 0.554 acc: 80.83 time: 2.64
```

```
TESTING:
Accuracy of the network on the 10000 test images: 76.12 %
Average loss on the 10000 test images: 0.703
[29,   100] loss: 0.559 acc: 80.45 time: 2.83
[29,   200] loss: 0.556 acc: 80.68 time: 2.68
[29,   300] loss: 0.560 acc: 80.80 time: 2.64
TESTING:
Accuracy of the network on the 10000 test images: 76.21 %
Average loss on the 10000 test images: 0.702
[30,   100] loss: 0.557 acc: 80.59 time: 2.81
[30,   200] loss: 0.545 acc: 81.18 time: 2.69
[30,   300] loss: 0.556 acc: 81.13 time: 2.63
TESTING:
Accuracy of the network on the 10000 test images: 76.13 %
Average loss on the 10000 test images: 0.703
Finished Training
```

10 Extra Credit 3: Rotation Prediction Model on Larger Dataset

10.0.1 Load Dataset

[ ]:
```
from fastai.vision.all import *
from fastai.collab import *

path = untar_data(URLs.IMAGENETTE_160)
trainset = path/"train"
testset = path/"val"
```

[ ]:
```
print(trainset)
print(testset)
```

```
/u/qilong/.fastai/data/imagenette2-160/train
/u/qilong/.fastai/data/imagenette2-160/val
```

[ ]:
```
transform_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomCrop(224, padding=16),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

transform_test = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])
```

```
batch_size = 256

def adjust_learning_rate(optimizer, epoch, init_lr, decay_epochs=30):
    """Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
    lr = init_lr * (0.1 ** (epoch // decay_epochs))
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr
```

[ ]:
```
import time

def run_test(net, testloader, criterion, task):
    correct = 0
    total = 0
    avg_test_loss = 0.0
    # since we're not training, we don't need to calculate the gradients for our outputs
    with torch.no_grad():
        for images, images_rotated, labels, cls_labels in testloader:
            if task == 'rotation':
              images, labels = images_rotated.to(device), labels.to(device)
            elif task == 'classification':
              images, labels = images.to(device), cls_labels.to(device)
            # TODO: Calculate outputs by running images through the network
            # The class with the highest energy is what we choose as prediction
            outputs = net(images)
            predicted = torch.argmax(outputs, dim=1)

            # loss
            avg_test_loss += criterion(outputs, labels)  / len(testloader)

            # calculate accuracy
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    print('TESTING:')
    print(f'Accuracy of the network on the {len(testloader)} test images: {100 * correct / total:.2f} %')
    print(f'Average loss on the {len(testloader)} test images: {avg_test_loss:.3f}')
```

[ ]:
```
# test CSV
df = pd.read_csv('noisy_imagenette.csv')
print(len(df[df["is_valid"] == False]))
```

```
9469
```

[ ]:
```
import os
import pandas as pd
from PIL import Image
import torch
from torchvision.datasets import VisionDataset
from torchvision import transforms
import numpy as np
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

# TODO: Define criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"

def create_class_to_idx(root_dir):
    classes = [d.name for d in os.scandir(root_dir) if d.is_dir()]
    classes.sort()
    class_to_idx = {cls_name: idx for idx, cls_name in enumerate(classes)}
    return class_to_idx

class ImageNetteRotation(VisionDataset):
    def __init__(self, root, csv_file, split, class_to_idx,transform=None,␣
 ↪target_transform=None, download=False):
        super(ImageNetteRotation, self).__init__(root, transform=transform,
                                                 ␣
 ↪target_transform=target_transform)
        self.images_df = pd.read_csv(csv_file)
        if split == "train":
            self.images_df = self.images_df[self.images_df['is_valid'] == False]
        elif split == "val":
            self.images_df = self.images_df[self.images_df['is_valid'] == True]
        else:
            print("Invalid Split!")
            return
        self.root = root
        self.class_to_idx = class_to_idx

    def __len__(self):
        return len(self.images_df)

    def __getitem__(self, index):
        img_path = self.images_df.iloc[index, 0]
        img_path = os.path.join(self.root, img_path)

        image = Image.open(img_path).convert("RGB")
        cls_name = self.images_df.iloc[index, 1]
```

```
        cls_label = self.class_to_idx[cls_name]

        rotation = int(np.random.choice([0, 90, 180, 270]))
        rot_image = transforms.functional.rotate(image, rotation)
        rot_label = rotation // 90
        if self.transform is not None:
            transformed_image = self.transform(image)
            rot_image = self.transform(rot_image)

        return transformed_image, rot_image, rot_label, torch.tensor(cls_label).
 ↪long()

dataset_root = path
csv_file = 'noisy_imagenette.csv'

# train_dir = os.path.join(dataset_root, 'train')
# test_dir = os.path.join(dataset_root, 'val')
class_to_idx = create_class_to_idx(train_dir)

# print(class_to_idx)

trainset = ImageNetteRotation(root=dataset_root,
                              csv_file=csv_file,
                              split="train",
                              class_to_idx=class_to_idx,
                              transform=transform_train,
                              download=True)
train_loader = torch.utils.data.DataLoader(trainset, batch_size=64,
                                           shuffle=True,
                                           num_workers=2)

testset = ImageNetteRotation(root=dataset_root,
                             csv_file=csv_file,
                             split="val",
                             class_to_idx=class_to_idx,
                             transform=transform_test,
                             download=True)
test_loader = torch.utils.data.DataLoader(testset, batch_size=64,
                                          shuffle=False,
                                          num_workers=2)
```

[ ]:
```
print(len(train_loader))
print(len(test_loader))
```

```
148
62
```

10.0.2 Pretrain on Rotation Task

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

[ ]:
```
import torch.optim as optim
import torch.nn as nn
import torch.nn.functional as F

from torchvision.models import resnet18

net = resnet18(num_classes=4)
net = net.to(device)
```

[ ]:
```
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"

net = resnet18(num_classes=4)
net = net.to(device)
def train_nette(net, criterion, optimizer, num_epochs, decay_epochs, init_lr,␣
 ↪task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(net.parameters(), lr=init_lr, eps=1e-08,␣
 ↪weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(net.parameters(), lr=init_lr, momentum=0.01)

    for epoch in range(num_epochs):  # loop over the dataset multiple times

        running_loss = 0.0
        running_correct = 0.0
        running_total = 0.0
        start_time = time.time()

        net.train()

        for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
 ↪enumerate(train_loader, 0):

            # TODO: Set the data to the correct device; Different task will use␣
 ↪different inputs and labels
            if task == 'rotation':
              images, labels = imgs_rotated.to(device), rotation_label.
 ↪to(device)
```

```
            elif task == 'classification':
              images, labels = imgs.to(device), cls_label.to(device)

            # TODO: Zero the parameter gradients
            optimizer_use.zero_grad()

            # TODO: forward + backward + optimize
            y_pred = net(images)
            loss = criterion(y_pred, labels)
            loss.backward()
            optimizer_use.step()

            # TODO: Get predicted results
            predicted = torch.argmax(y_pred, dim=1)

            # print statistics
            print_freq = 50
            running_loss += loss.item()

            # calc acc
            running_total += labels.size(0)
            running_correct += (predicted == labels).sum().item()

            if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
                running_loss, running_correct, running_total = 0.0, 0.0, 0.0
                start_time = time.time()

        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        net.eval()
        with torch.no_grad():
          run_test(net, test_loader, criterion, task)

    print('Finished Training')
```

[ ]:
```
## run cell two times (60 epochs in total)
train_nette(net, criterion, optimizer, num_epochs=30, decay_epochs=10,␣
 ↪init_lr=0.0001, task='rotation')
torch.save(net.state_dict(), './nette_pretrain.pth')
```

```
[1,    50] loss: 0.753 acc: 57.12 time: 6.09
[1,   100] loss: 0.750 acc: 57.47 time: 5.65
TESTING:
Accuracy of the network on the 62 test images: 57.22 %
Average loss on the 62 test images: 0.777
[2,    50] loss: 0.746 acc: 58.94 time: 5.98
[2,   100] loss: 0.750 acc: 58.41 time: 5.74
TESTING:
Accuracy of the network on the 62 test images: 35.44 %
Average loss on the 62 test images: 1.808
[3,    50] loss: 0.736 acc: 57.34 time: 6.07
[3,   100] loss: 0.746 acc: 58.25 time: 5.76
TESTING:
Accuracy of the network on the 62 test images: 54.37 %
Average loss on the 62 test images: 0.853
[4,    50] loss: 0.755 acc: 57.47 time: 5.96
[4,   100] loss: 0.757 acc: 58.44 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 56.66 %
Average loss on the 62 test images: 0.894
[5,    50] loss: 0.755 acc: 58.69 time: 6.36
[5,   100] loss: 0.747 acc: 58.09 time: 5.68
TESTING:
Accuracy of the network on the 62 test images: 56.51 %
Average loss on the 62 test images: 0.774
[6,    50] loss: 0.739 acc: 58.12 time: 6.07
[6,   100] loss: 0.727 acc: 59.03 time: 5.79
TESTING:
Accuracy of the network on the 62 test images: 57.81 %
Average loss on the 62 test images: 0.963
[7,    50] loss: 0.743 acc: 58.69 time: 6.07
[7,   100] loss: 0.733 acc: 57.50 time: 5.76
TESTING:
Accuracy of the network on the 62 test images: 39.21 %
Average loss on the 62 test images: 1.317
[8,    50] loss: 0.739 acc: 59.16 time: 5.93
[8,   100] loss: 0.739 acc: 57.81 time: 5.66
TESTING:
Accuracy of the network on the 62 test images: 53.83 %
Average loss on the 62 test images: 0.904
[9,    50] loss: 0.737 acc: 58.56 time: 5.96
[9,   100] loss: 0.728 acc: 59.59 time: 5.72
TESTING:
Accuracy of the network on the 62 test images: 57.22 %
Average loss on the 62 test images: 0.855
[10,    50] loss: 0.732 acc: 58.97 time: 6.03
[10,   100] loss: 0.748 acc: 58.06 time: 5.72
TESTING:
```

```
Accuracy of the network on the 62 test images: 56.64 %
Average loss on the 62 test images: 0.790
[11,    50] loss: 0.740 acc: 59.66 time: 5.96
[11,   100] loss: 0.733 acc: 59.06 time: 5.63
TESTING:
Accuracy of the network on the 62 test images: 50.27 %
Average loss on the 62 test images: 1.002
[12,    50] loss: 0.734 acc: 59.25 time: 6.19
[12,   100] loss: 0.715 acc: 60.34 time: 5.73
TESTING:
Accuracy of the network on the 62 test images: 58.88 %
Average loss on the 62 test images: 0.752
[13,    50] loss: 0.719 acc: 57.97 time: 5.99
[13,   100] loss: 0.708 acc: 59.97 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 59.31 %
Average loss on the 62 test images: 0.735
[14,    50] loss: 0.724 acc: 59.88 time: 5.97
[14,   100] loss: 0.718 acc: 59.91 time: 5.69
TESTING:
Accuracy of the network on the 62 test images: 58.88 %
Average loss on the 62 test images: 0.747
[15,    50] loss: 0.712 acc: 60.81 time: 5.95
[15,   100] loss: 0.704 acc: 61.06 time: 5.74
TESTING:
Accuracy of the network on the 62 test images: 58.98 %
Average loss on the 62 test images: 0.754
[16,    50] loss: 0.698 acc: 61.12 time: 5.93
[16,   100] loss: 0.733 acc: 60.47 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 58.22 %
Average loss on the 62 test images: 0.775
[17,    50] loss: 0.717 acc: 60.47 time: 5.94
[17,   100] loss: 0.688 acc: 60.50 time: 5.73
TESTING:
Accuracy of the network on the 62 test images: 59.29 %
Average loss on the 62 test images: 0.761
[18,    50] loss: 0.695 acc: 61.81 time: 6.10
[18,   100] loss: 0.717 acc: 60.22 time: 5.72
TESTING:
Accuracy of the network on the 62 test images: 58.68 %
Average loss on the 62 test images: 0.762
[19,    50] loss: 0.720 acc: 58.97 time: 6.16
[19,   100] loss: 0.722 acc: 60.31 time: 5.68
TESTING:
Accuracy of the network on the 62 test images: 60.48 %
Average loss on the 62 test images: 0.753
[20,    50] loss: 0.710 acc: 59.31 time: 5.89
```

```
[20,   100] loss: 0.722 acc: 59.44 time: 5.67
TESTING:
Accuracy of the network on the 62 test images: 58.80 %
Average loss on the 62 test images: 0.753
[21,    50] loss: 0.723 acc: 59.28 time: 6.13
[21,   100] loss: 0.689 acc: 61.88 time: 5.75
TESTING:
Accuracy of the network on the 62 test images: 59.52 %
Average loss on the 62 test images: 0.749
[22,    50] loss: 0.719 acc: 60.16 time: 5.93
[22,   100] loss: 0.715 acc: 59.50 time: 5.67
TESTING:
Accuracy of the network on the 62 test images: 59.08 %
Average loss on the 62 test images: 0.755
[23,    50] loss: 0.701 acc: 61.69 time: 5.98
[23,   100] loss: 0.712 acc: 60.00 time: 5.73
TESTING:
Accuracy of the network on the 62 test images: 61.22 %
Average loss on the 62 test images: 0.750
[24,    50] loss: 0.700 acc: 59.59 time: 6.16
[24,   100] loss: 0.700 acc: 60.50 time: 5.70
TESTING:
Accuracy of the network on the 62 test images: 60.03 %
Average loss on the 62 test images: 0.751
[25,    50] loss: 0.704 acc: 60.47 time: 5.98
[25,   100] loss: 0.696 acc: 59.59 time: 5.60
TESTING:
Accuracy of the network on the 62 test images: 59.26 %
Average loss on the 62 test images: 0.751
[26,    50] loss: 0.724 acc: 57.66 time: 6.89
[26,   100] loss: 0.695 acc: 62.00 time: 6.58
TESTING:
Accuracy of the network on the 62 test images: 59.01 %
Average loss on the 62 test images: 0.733
[27,    50] loss: 0.703 acc: 60.91 time: 5.97
[27,   100] loss: 0.726 acc: 60.56 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 61.25 %
Average loss on the 62 test images: 0.739
[28,    50] loss: 0.719 acc: 60.50 time: 6.19
[28,   100] loss: 0.687 acc: 61.09 time: 5.82
TESTING:
Accuracy of the network on the 62 test images: 59.59 %
Average loss on the 62 test images: 0.762
[29,    50] loss: 0.712 acc: 60.88 time: 5.97
[29,   100] loss: 0.720 acc: 60.78 time: 5.69
TESTING:
Accuracy of the network on the 62 test images: 59.34 %
```

```
Average loss on the 62 test images: 0.753
[30,    50] loss: 0.698 acc: 60.97 time: 5.90
[30,   100] loss: 0.711 acc: 61.09 time: 5.72
TESTING:
Accuracy of the network on the 62 test images: 59.75 %
Average loss on the 62 test images: 0.753
Finished Training
```

10.0.3 Finetune on Classification Task

[ ]:
```
# Load the pre-trained ResNet18 model
net = resnet18(num_classes=4).to(device)
net.load_state_dict(torch.load('./nette_pretrain.pth'))
print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
```

```
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=4, bias=True)
)
```

[ ]:
```
net.fc = nn.Linear(in_features=512, out_features=10, bias=True).to(device)

## unfreeze layer4 and fc
for name, param in net.named_parameters():
    if "layer4" in name:
```

```
        param.requires_grad = True
    elif "fc" in name:
        param.requires_grad = True
    else:
        param.requires_grad = False

print(net)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
```

```
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:
```
criterion = nn.CrossEntropyLoss()
optimizer = "Adam"
```

[ ]:
```
def train(net, criterion, optimizer, num_epochs, decay_epochs, init_lr, task):

    if optimizer == "Adam":
        optimizer_use = optim.Adam(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, eps=1e-08, weight_decay=0.001)
    elif optimizer == "SGD":
        optimizer_use = optim.SGD(filter(lambda p: p.requires_grad, net.
 ↪parameters()), lr=init_lr, momentum=0.01)
```

```
  for epoch in range(num_epochs):  # loop over the dataset multiple times

      running_loss = 0.0
      running_correct = 0.0
      running_total = 0.0
      start_time = time.time()

      for i, (imgs, imgs_rotated, rotation_label, cls_label) in␣
↪enumerate(trainloader, 0):

          # TODO: Set the data to the correct device; Different task will use␣
↪different inputs and labels
          if task == 'rotation':
            images, labels = imgs_rotated.to(device), rotation_label.
↪to(device)
          elif task == 'classification':
            images, labels = imgs.to(device), cls_label.to(device)

          # TODO: Zero the parameter gradients
          optimizer_use.zero_grad()

          # TODO: forward + backward + optimize
          y_pred = net(images)
          loss = criterion(y_pred, labels)
          loss.backward()
          optimizer_use.step()

          # TODO: Get predicted results
          predicted = torch.argmax(y_pred, dim=1)

          # print statistics
          print_freq = 100
          running_loss += loss.item()

          # calc acc
          running_total += labels.size(0)
          running_correct += (predicted == labels).sum().item()

          if i % print_freq == (print_freq - 1):    # print every 2000␣
↪mini-batches
              print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss /␣
↪print_freq:.3f} acc: {100*running_correct / running_total:.2f} time: {time.
↪time() - start_time:.2f}')
              running_loss, running_correct, running_total = 0.0, 0.0, 0.0
              start_time = time.time()
```

```
        adjust_learning_rate(optimizer_use, epoch, init_lr,␣
↪decay_epochs=decay_epochs)

        # TODO: Run the run_test() function after each epoch; Set the model to␣
↪the evaluation mode.
        with torch.no_grad():
          run_test(net, testloader, criterion, task)

    print('Finished Training')
```

[ ]:
```
## run cell two times (40 epochs in total)
train_nette(net, criterion, optimizer, num_epochs=20, decay_epochs=10,␣
 ↪init_lr=0.0001, task='classification')
torch.save(net.state_dict(), './nette_finetune.pth')
```

```
[1,    50] loss: 0.643 acc: 79.22 time: 6.13
[1,   100] loss: 0.619 acc: 79.78 time: 5.64
TESTING:
Accuracy of the network on the 62 test images: 74.27 %
Average loss on the 62 test images: 0.811
[2,    50] loss: 0.624 acc: 79.97 time: 6.46
[2,   100] loss: 0.624 acc: 79.62 time: 6.12
TESTING:
Accuracy of the network on the 62 test images: 74.57 %
Average loss on the 62 test images: 0.807
[3,    50] loss: 0.605 acc: 80.22 time: 5.94
[3,   100] loss: 0.615 acc: 80.41 time: 5.70
TESTING:
Accuracy of the network on the 62 test images: 75.16 %
Average loss on the 62 test images: 0.807
[4,    50] loss: 0.579 acc: 81.16 time: 6.01
[4,   100] loss: 0.594 acc: 81.50 time: 5.75
TESTING:
Accuracy of the network on the 62 test images: 74.62 %
Average loss on the 62 test images: 0.813
[5,    50] loss: 0.572 acc: 81.41 time: 6.07
[5,   100] loss: 0.580 acc: 80.84 time: 5.65
TESTING:
Accuracy of the network on the 62 test images: 75.08 %
Average loss on the 62 test images: 0.799
[6,    50] loss: 0.562 acc: 81.19 time: 6.03
[6,   100] loss: 0.583 acc: 80.91 time: 5.61
TESTING:
Accuracy of the network on the 62 test images: 74.68 %
Average loss on the 62 test images: 0.806
[7,    50] loss: 0.536 acc: 82.69 time: 6.01
[7,   100] loss: 0.575 acc: 81.16 time: 5.77
```

```
TESTING:
Accuracy of the network on the 62 test images: 74.60 %
Average loss on the 62 test images: 0.804
[8,    50] loss: 0.528 acc: 82.62 time: 6.00
[8,   100] loss: 0.565 acc: 81.16 time: 5.81
TESTING:
Accuracy of the network on the 62 test images: 75.08 %
Average loss on the 62 test images: 0.800
[9,    50] loss: 0.534 acc: 82.38 time: 6.00
[9,   100] loss: 0.522 acc: 82.88 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 74.57 %
Average loss on the 62 test images: 0.805
[10,    50] loss: 0.525 acc: 83.16 time: 5.99
[10,   100] loss: 0.542 acc: 82.19 time: 5.81
TESTING:
Accuracy of the network on the 62 test images: 74.39 %
Average loss on the 62 test images: 0.805
[11,    50] loss: 0.501 acc: 83.53 time: 6.24
[11,   100] loss: 0.553 acc: 81.50 time: 6.10
TESTING:
Accuracy of the network on the 62 test images: 75.52 %
Average loss on the 62 test images: 0.791
[12,    50] loss: 0.500 acc: 83.16 time: 5.92
[12,   100] loss: 0.481 acc: 84.66 time: 5.76
TESTING:
Accuracy of the network on the 62 test images: 75.72 %
Average loss on the 62 test images: 0.775
[13,    50] loss: 0.482 acc: 84.81 time: 6.28
[13,   100] loss: 0.494 acc: 84.19 time: 6.22
TESTING:
Accuracy of the network on the 62 test images: 75.92 %
Average loss on the 62 test images: 0.771
[14,    50] loss: 0.463 acc: 85.41 time: 6.01
[14,   100] loss: 0.485 acc: 84.12 time: 5.81
TESTING:
Accuracy of the network on the 62 test images: 76.20 %
Average loss on the 62 test images: 0.772
[15,    50] loss: 0.470 acc: 84.84 time: 6.18
[15,   100] loss: 0.466 acc: 85.06 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 75.52 %
Average loss on the 62 test images: 0.780
[16,    50] loss: 0.461 acc: 86.00 time: 6.05
[16,   100] loss: 0.466 acc: 85.06 time: 5.74
TESTING:
Accuracy of the network on the 62 test images: 75.82 %
Average loss on the 62 test images: 0.775
```

```
[17,    50] loss: 0.455 acc: 85.81 time: 6.06
[17,   100] loss: 0.470 acc: 84.38 time: 5.70
TESTING:
Accuracy of the network on the 62 test images: 76.10 %
Average loss on the 62 test images: 0.775
[18,    50] loss: 0.461 acc: 84.97 time: 6.06
[18,   100] loss: 0.455 acc: 85.22 time: 5.77
TESTING:
Accuracy of the network on the 62 test images: 75.44 %
Average loss on the 62 test images: 0.772
[19,    50] loss: 0.446 acc: 85.19 time: 6.03
[19,   100] loss: 0.462 acc: 85.44 time: 5.77
TESTING:
Accuracy of the network on the 62 test images: 75.95 %
Average loss on the 62 test images: 0.777
[20,    50] loss: 0.458 acc: 84.97 time: 6.01
[20,   100] loss: 0.437 acc: 86.44 time: 5.71
TESTING:
Accuracy of the network on the 62 test images: 75.69 %
Average loss on the 62 test images: 0.774
Finished Training
```

11 Save as PDF

[3]:
```
%%capture

from google.colab import drive
drive.mount('/content/drive')
# install tex; first run may take several minutes
! apt-get install texlive-xetex
# file path and save location below are default; please change if they do not␣
 ↪match yours
! jupyter nbconvert --output-dir='/content/drive/MyDrive/' '/content/drive/
 ↪MyDrive/CS444/assignment3_starter_sp24/assignment3_part1/a3_part1_rotation.
 ↪ipynb' --to pdf
```