Assignment 4

March 30, 2022

[1]:

```
# imports
import numpy as np
import matplotlib.pyplot as plt
import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torchvision.transforms as transforms
import torchvision.models as models
from torch.utils.data import DataLoader
from torch import optim
from tqdm.auto import tqdm
import math
torch.manual_seed(55)
```

[1]:

```
<torch._C.Generator at 0x286494ef510>
```

[ ]:

```

```

1 read in data

[2]:

```
path = "W://Study Material/Jupyter Notebook/Datasets/CIFAR-10/
 ↪cifar-10-batches-py"
def unpickle(file):
    import pickle
    with open(file, "rb") as fo:
        dict = pickle.load(fo,encoding="bytes")
    return dict
```

[3]:

```
training_batch = np.empty((4),dtype=object)
val_batch = np.empty((1),dtype=object)
test_batch = np.empty((1),dtype=object)
training_batch.shape
```

[3]:

```
(4,)
```

[4]:

```
training_batch[0] = unpickle(path+"/data_batch_1")
training_batch[1] = unpickle(path+"/data_batch_2")
training_batch[2] = unpickle(path+"/data_batch_3")
training_batch[3] = unpickle(path+"/data_batch_4")
val_batch[0] = unpickle(path+"/data_batch_5")
test_batch[0] = unpickle(path+"/test_batch")
meta_data = unpickle(path+"/batches.meta")
```

[5]:

```
training_batch[0].keys()
```

[5]:

```
dict_keys([b'batch_label', b'labels', b'data', b'filenames'])
```

[6]:

```
np.array(training_batch[0][b'labels']).shape
```

[6]:

```
(10000,)
```

[7]:

```
training_batch[0][b'data'].shape
```

[7]:

```
(10000, 3072)
```

[8]:

```
meta_data[b'label_names']
```

[8]:

```
[b'airplane',
 b'automobile',
 b'bird',
 b'cat',
 b'deer',
 b'dog',
 b'frog',
 b'horse',
 b'ship',
 b'truck']
```

[ ]:

```

```

2 define custom data loader

[9]:

```
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self,batch,b_type,transform):
        self.transform = transform
        if b_type == "train":
            self.data = np.concatenate((batch[0][b'data'],batch[1][b'data'],
                                                                    ␣
 ↪batch[2][b'data'],batch[3][b'data']),axis=0)
            #print(b_type, " data shape: ",self.data.shape)
            self.label = np.
 ↪concatenate((batch[0][b'labels'],batch[1][b'labels'],
                                       ␣
↪batch[2][b'labels'],batch[3][b'labels']))
            #print(b_type, " label shape: ",len(self.label))
        else:
            self.data = batch[0][b'data']
            #print(b_type, " data shape: ",self.data.shape)
            self.label = batch[0][b'labels']
            #print(b_type, " label shape: ", len(self.label))

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

    def __getitem__(self,index):
        label = self.label[index]
        image_data = self.data[index]
        # convert (3072,) array to (3,32,32)
        image_r = image_data[:1024].reshape(32,32)
        image_g = image_data[1024:2048].reshape(32,32)
        image_b = image_data[2048:].reshape(32,32)
        image = np.array([image_r,image_g,image_b])
        image = np.transpose(image,(1,2,0)) # change the batch dimension to the␣
↪last dimension
        #image = torch.tensor(image,dtype=torch.float)
        if self.transform is not None:
            image = self.transform(image)
        #print(image.shape)
        return image, label
```

[10]:

```
x = np.ones((1,2,3))
np.transpose(x,(2,0,1)).shape
```

[10]:

```
(3, 1, 2)
```

2.0.1 transformation

[11]:

```
# define data augmentation
data_transformers = {"train": transforms.Compose([transforms.ToPILImage(),
                                     transforms.RandomResizedCrop(256),
                                     transforms.RandomHorizontalFlip(),
                                     transforms.RandomRotation(30),
                                     transforms.ToTensor(),
                                     transforms.Normalize((0.5,0.5,0.5),(0.5,0.
↪5,0.5))]),
            "test":transforms.Compose([transforms.ToPILImage(),
                                       transforms.Resize(256),
                                       transforms.ToTensor(),
                                        transforms.Normalize((0.5,0.5,0.5),(0.
↪5,0.5,0.5))])}
```

2.0.2 datasets

[12]:

```
# defining datasets
train_data =␣
 ↪CustomDataset(training_batch,"train",transform=data_transformers["train"])
val_data =␣
 ↪CustomDataset(val_batch,"validation",transform=data_transformers["test"])
test_data = CustomDataset(test_batch,"test",transform=data_transformers["test"])
```

[13]:

```
# make sure the shape of the image extracted is okay
print(train_data[0][0].shape)
#print(train_data[0][1])
```

```
torch.Size([3, 256, 256])
```

2.0.3 dataloader

[14]:

```
train_dataloader = DataLoader(train_data,batch_size=128,shuffle=True)
val_dataloader = DataLoader(val_data,batch_size=128,shuffle=True)
test_dataloader = DataLoader(test_data,batch_size=128,shuffle=False)
```

[ ]:

```

```

[ ]:

```

```

3 training and validation routine

[20]:

```
def␣
 ↪train(model,trainloader,valloader,epochs,print_frequency,loss_fn,optimizer,device,run,part)
 ↪
    training_steps = epochs * (len(trainloader))
    progress_bar = tqdm(range(training_steps))
    ep = epochs
    print_every = math.floor(training_steps/print_frequency)
    steps = 0
    model.to(device)
    model.train()
    total = 0
    correct = 0
    best_val = 10000

    current_loss = 0

    for e in np.arange(ep):
        for batch,(images,labels) in enumerate(trainloader):
            steps += 1
            images, labels = images.to(device), labels.to(device)
            # convert images to float because weights are floats
            images = images.float()
            labels = labels.type(torch.long) # need to be int

            # calculate loss and backpropogate
            optimizer.zero_grad()
            outputs = model(images)
            _,predictions = torch.max(outputs.data,1)
            total += labels.size(0)
            correct += (predictions == labels).sum().item()
            loss = loss_fn(outputs,labels)
            loss.backward()
            optimizer.step()

            current_loss += loss.item()
            progress_bar.update(1)

            if steps % print_every == 0:
                print('EPOCHS : {}/{}'.format(e+1,epochs),
                      'Loss : {:.6f}'.format(current_loss/print_every))
                print('The training accuracy is {:.2f}%'.format(correct/
↪total*100))
                current_loss = 0
                val_loss = validate(model,valloader,loss_fn, device)
                if val_loss < best_val:
                    torch.save(model.state_dict(),"Weights/part-{}-run-{}-Best.
↪pth".format(str(part),
                                                                                ␣
↪            str(run)))
    torch.save(model.state_dict(),"Weights/part-{}-run-{}-Last.pth".
↪format(str(part),
                                                                           ␣
↪str(run)))
```

[21]:

```
def validate(model, valloader, loss_fn, device):
    total = 0
    correct = 0
    val_loss = 0
    model.eval()
    steps = 0
    with torch.no_grad():
        for batch, (images,labels) in enumerate(valloader):
            images, labels = images.to(device), labels.to(device)
            # convert images to float because weights are floats
            images = images.float()
            labels = labels.type(torch.long) # need to be int
            steps += 1
            outputs = model(images)
            _, predictions = torch.max(outputs.data,1)
            total += labels.size(0)
            correct += (labels == predictions).sum().item()
            loss = loss_fn(outputs,labels)
            val_loss += loss.item()
    val_loss /= steps
    accuracy = correct / total * 100
    print("The validation loss is %.4f" % (val_loss))
    print('The valudation accuracy is {:.2f}%\n'.format(accuracy))
    return val_loss
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

4 Part b)

[17]:

```
AlexNet = models.alexnet(pretrained=False)
AlexNet.load_state_dict(torch.load("W://Study Material/Jupyter Notebook/
 ↪Pretrained_Weights/alexnet-owt-7be5be79.pth"))
AlexNet.cuda()
```

[17]:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

[ ]:

```

```

[18]:

```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(AlexNet.parameters(),lr=1e-3)
```

[24]:

```
train(AlexNet,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",1)
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 2.247014
The training accuracy is 21.80%
The validation loss is 1.6998
The valudation accuracy is 36.71%

EPOCHS : 1/10 Loss : 1.785275
The training accuracy is 27.76%
The validation loss is 1.5200
The valudation accuracy is 43.46%

EPOCHS : 2/10 Loss : 1.685747
The training accuracy is 31.12%
The validation loss is 1.4152
The valudation accuracy is 47.11%

EPOCHS : 2/10 Loss : 1.600968
The training accuracy is 33.61%
The validation loss is 1.3193
The valudation accuracy is 51.00%

EPOCHS : 3/10 Loss : 1.554594
The training accuracy is 35.58%
The validation loss is 1.3243
The valudation accuracy is 53.44%

EPOCHS : 3/10 Loss : 1.490266
The training accuracy is 37.28%
The validation loss is 1.1996
The valudation accuracy is 56.95%

EPOCHS : 4/10 Loss : 1.486563
The training accuracy is 38.56%
The validation loss is 1.1369
The valudation accuracy is 58.76%

EPOCHS : 4/10 Loss : 1.429415
The training accuracy is 39.73%
The validation loss is 1.1196
The valudation accuracy is 60.99%

EPOCHS : 5/10 Loss : 1.401546
The training accuracy is 40.79%
The validation loss is 1.0909
The valudation accuracy is 61.05%

EPOCHS : 5/10 Loss : 1.368858
The training accuracy is 41.79%
The validation loss is 1.0646
The valudation accuracy is 62.20%

EPOCHS : 6/10 Loss : 1.354280
The training accuracy is 42.69%
The validation loss is 1.0736
The valudation accuracy is 62.41%

EPOCHS : 6/10 Loss : 1.334924
The training accuracy is 43.50%
The validation loss is 0.9667
The valudation accuracy is 65.68%

EPOCHS : 7/10 Loss : 1.300909
The training accuracy is 44.25%
The validation loss is 0.9733
The valudation accuracy is 66.59%

EPOCHS : 7/10 Loss : 1.288129
The training accuracy is 44.92%
The validation loss is 1.0068
The valudation accuracy is 64.91%
```

```
EPOCHS : 8/10 Loss : 1.270171
The training accuracy is 45.57%
The validation loss is 0.9393
The valudation accuracy is 67.63%

EPOCHS : 8/10 Loss : 1.249892
The training accuracy is 46.21%
The validation loss is 0.9625
The valudation accuracy is 67.05%

EPOCHS : 9/10 Loss : 1.238430
The training accuracy is 46.80%
The validation loss is 0.8542
The valudation accuracy is 70.39%

EPOCHS : 9/10 Loss : 1.230177
The training accuracy is 47.33%
The validation loss is 0.8584
The valudation accuracy is 70.26%

EPOCHS : 10/10 Loss : 1.208205
The training accuracy is 47.83%
The validation loss is 0.8575
The valudation accuracy is 70.58%

EPOCHS : 10/10 Loss : 1.206469
The training accuracy is 48.30%
The validation loss is 0.8517
The valudation accuracy is 70.97%
```

```
[25]: optimizer = optim.Adam(AlexNet.parameters(),lr=1e-5)
      train(AlexNet,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",2)
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 1.314707
The training accuracy is 54.08%
The validation loss is 0.7874
The valudation accuracy is 72.85%

EPOCHS : 1/10 Loss : 1.110474
The training accuracy is 57.41%
The validation loss is 0.7677
The valudation accuracy is 73.35%

EPOCHS : 2/10 Loss : 1.099722
The training accuracy is 58.54%
The validation loss is 0.7579
The valudation accuracy is 73.52%

EPOCHS : 2/10 Loss : 1.087660
The training accuracy is 59.29%
The validation loss is 0.7448
The valudation accuracy is 74.05%

EPOCHS : 3/10 Loss : 1.072100
The training accuracy is 59.91%
The validation loss is 0.7409
The valudation accuracy is 74.36%

EPOCHS : 3/10 Loss : 1.068306
The training accuracy is 60.23%
The validation loss is 0.7364
The valudation accuracy is 74.53%

EPOCHS : 4/10 Loss : 1.055259
The training accuracy is 60.57%
The validation loss is 0.7403
The valudation accuracy is 74.46%

EPOCHS : 4/10 Loss : 1.051216
The training accuracy is 60.83%
The validation loss is 0.7277
The valudation accuracy is 74.69%

EPOCHS : 5/10 Loss : 1.054497
The training accuracy is 61.01%
The validation loss is 0.7298
The valudation accuracy is 74.64%

EPOCHS : 5/10 Loss : 1.037197
The training accuracy is 61.21%
The validation loss is 0.7229
The valudation accuracy is 74.84%

EPOCHS : 6/10 Loss : 1.044361
The training accuracy is 61.40%
The validation loss is 0.7249
The valudation accuracy is 75.03%

EPOCHS : 6/10 Loss : 1.044632
The training accuracy is 61.51%
The validation loss is 0.7188
The valudation accuracy is 74.97%

EPOCHS : 7/10 Loss : 1.044561
```

```
The training accuracy is 61.61%
The validation loss is 0.7205
The valudation accuracy is 75.01%

EPOCHS : 7/10 Loss : 1.030355
The training accuracy is 61.74%
The validation loss is 0.7082
The valudation accuracy is 75.07%

EPOCHS : 8/10 Loss : 1.045026
The training accuracy is 61.83%
The validation loss is 0.7083
The valudation accuracy is 75.31%

EPOCHS : 8/10 Loss : 1.014952
The training accuracy is 61.98%
The validation loss is 0.7143
The valudation accuracy is 75.18%

EPOCHS : 9/10 Loss : 1.038396
The training accuracy is 62.04%
The validation loss is 0.7163
The valudation accuracy is 75.08%

EPOCHS : 9/10 Loss : 1.020153
The training accuracy is 62.15%
The validation loss is 0.7034
The valudation accuracy is 75.28%

EPOCHS : 10/10 Loss : 1.028688
The training accuracy is 62.22%
The validation loss is 0.7066
The valudation accuracy is 75.30%

EPOCHS : 10/10 Loss : 1.011283
The training accuracy is 62.33%
The validation loss is 0.6973
The valudation accuracy is 75.38%
```

[19]:

```
optimizer = optim.Adam(AlexNet.parameters(),lr=1e-6)
AlexNet.load_state_dict(torch.load("Weights/run-2-Best.pth"))
train(AlexNet,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",3)
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 1.236205
The training accuracy is 57.47%
The validation loss is 0.7003
The valudation accuracy is 75.54%

EPOCHS : 1/10 Loss : 1.020405
The training accuracy is 60.70%
The validation loss is 0.7034
The valudation accuracy is 75.55%

EPOCHS : 2/10 Loss : 1.018962
The training accuracy is 61.69%
The validation loss is 0.6994
The valudation accuracy is 75.49%

EPOCHS : 2/10 Loss : 1.014674
The training accuracy is 62.28%
The validation loss is 0.6986
The valudation accuracy is 75.52%

EPOCHS : 3/10 Loss : 1.023155
The training accuracy is 62.52%
The validation loss is 0.7007
The valudation accuracy is 75.56%

EPOCHS : 3/10 Loss : 1.013412
The training accuracy is 62.75%
The validation loss is 0.7012
The valudation accuracy is 75.43%

EPOCHS : 4/10 Loss : 1.010199
The training accuracy is 62.95%
The validation loss is 0.7021
The valudation accuracy is 75.45%

EPOCHS : 4/10 Loss : 1.028888
The training accuracy is 63.05%
The validation loss is 0.7022
The valudation accuracy is 75.49%

EPOCHS : 5/10 Loss : 1.016358
The training accuracy is 63.15%
The validation loss is 0.6972
The valudation accuracy is 75.48%

EPOCHS : 5/10 Loss : 1.023578
The training accuracy is 63.22%
The validation loss is 0.7030
The valudation accuracy is 75.59%

EPOCHS : 6/10 Loss : 1.019098
```

```
The training accuracy is 63.26%
The validation loss is 0.7011
The valudation accuracy is 75.58%

EPOCHS : 6/10 Loss : 1.015666
The training accuracy is 63.36%
The validation loss is 0.7017
The valudation accuracy is 75.58%

EPOCHS : 7/10 Loss : 1.015819
The training accuracy is 63.42%
The validation loss is 0.7019
The valudation accuracy is 75.66%

EPOCHS : 7/10 Loss : 0.999843
The training accuracy is 63.52%
The validation loss is 0.6974
The valudation accuracy is 75.63%

EPOCHS : 8/10 Loss : 1.012752
The training accuracy is 63.54%
The validation loss is 0.7003
The valudation accuracy is 75.61%

EPOCHS : 8/10 Loss : 1.019031
The training accuracy is 63.54%
The validation loss is 0.7079
The valudation accuracy is 75.57%

EPOCHS : 9/10 Loss : 1.016139
The training accuracy is 63.54%
The validation loss is 0.7064
The valudation accuracy is 75.57%

EPOCHS : 9/10 Loss : 1.005403
The training accuracy is 63.60%
The validation loss is 0.6966
The valudation accuracy is 75.55%

EPOCHS : 10/10 Loss : 1.006781
The training accuracy is 63.62%
The validation loss is 0.6981
The valudation accuracy is 75.61%

EPOCHS : 10/10 Loss : 1.022852
The training accuracy is 63.63%
The validation loss is 0.6972
The valudation accuracy is 75.57%
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

5 Part c)

[26]:

```
AlexNet2 = models.alexnet(pretrained=False)
AlexNet2.load_state_dict(torch.load("W://Study Material/Jupyter Notebook/
 ↪Pretrained_Weights/alexnet-owt-7be5be79.pth"))
AlexNet2.cuda()
```

[26]:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
    )
  )
```

[27]:

```
AlexNet2.classifier = nn.Sequential(
    nn.Dropout(0.5,False),
    nn.Linear(9216,4096,True),
    nn.ReLU(True),
    nn.Linear(4096,10,True)
)
print(AlexNet2)
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=4096, out_features=10, bias=True)
  )
)
```

[24]:

```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(AlexNet2.parameters(),lr=1e-3)
```

[25]:

```
train(AlexNet2,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",1,"c")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]
```

```
EPOCHS : 1/10 Loss : 2.342232
The training accuracy is 9.85%
The validation loss is 2.3030
The valudation accuracy is 9.52%

EPOCHS : 1/10 Loss : 2.302850
The training accuracy is 9.81%
The validation loss is 2.3027
The valudation accuracy is 10.14%

EPOCHS : 2/10 Loss : 2.302646
The training accuracy is 9.81%
The validation loss is 2.3028
The valudation accuracy is 10.14%

EPOCHS : 2/10 Loss : 2.302732
The training accuracy is 9.78%
The validation loss is 2.3027
The valudation accuracy is 10.03%
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_23084/577002224.py in <module>
----> 1␣
 ↪train(AlexNet2,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda" 1,"c")

~\AppData\Local\Temp/ipykernel_23084/1989990385.py in train(model, trainloader,␣
 ↪valloader, epochs, print_frequency, loss_fn, optimizer, device, run, part)
     16         for batch,(images,labels) in enumerate(trainloader):
     17             steps += 1
---> 18             images, labels = images.to(device), labels.to(device)
     19             # convert images to float because weights are floats
     20             images = images.float()

KeyboardInterrupt:
```

[28]:

```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(AlexNet2.parameters(),lr=1e-5)
```

[29]:

```
train(AlexNet2,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",1,"c")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 1.749224
The training accuracy is 37.26%
The validation loss is 1.0611
The valudation accuracy is 62.49%

EPOCHS : 1/10 Loss : 1.344720
The training accuracy is 44.61%
The validation loss is 0.8322
The valudation accuracy is 71.62%

EPOCHS : 2/10 Loss : 1.248277
The training accuracy is 48.05%
The validation loss is 0.7669
The valudation accuracy is 73.68%

EPOCHS : 2/10 Loss : 1.182601
The training accuracy is 50.66%
The validation loss is 0.7867
The valudation accuracy is 72.40%

EPOCHS : 3/10 Loss : 1.149564
The training accuracy is 52.32%
The validation loss is 0.6671
The valudation accuracy is 76.95%

EPOCHS : 3/10 Loss : 1.132360
The training accuracy is 53.53%
The validation loss is 0.6508
The valudation accuracy is 77.58%

EPOCHS : 4/10 Loss : 1.089950
The training accuracy is 54.61%
The validation loss is 0.6348
The valudation accuracy is 78.25%

EPOCHS : 4/10 Loss : 1.067065
The training accuracy is 55.53%
The validation loss is 0.5904
The valudation accuracy is 79.56%

EPOCHS : 5/10 Loss : 1.058476
The training accuracy is 56.33%
The validation loss is 0.5837
The valudation accuracy is 79.80%

EPOCHS : 5/10 Loss : 1.039308
The training accuracy is 57.00%
The validation loss is 0.5653
The valudation accuracy is 80.63%

EPOCHS : 6/10 Loss : 1.022074
The training accuracy is 57.62%
The validation loss is 0.5610
The valudation accuracy is 80.57%
```

```
EPOCHS : 6/10 Loss : 1.004124
The training accuracy is 58.18%
The validation loss is 0.5485
The valudation accuracy is 80.86%

EPOCHS : 7/10 Loss : 0.996803
The training accuracy is 58.69%
The validation loss is 0.5404
The valudation accuracy is 81.06%

EPOCHS : 7/10 Loss : 0.982126
The training accuracy is 59.15%
The validation loss is 0.5449
The valudation accuracy is 81.12%

EPOCHS : 8/10 Loss : 0.975503
The training accuracy is 59.57%
The validation loss is 0.5282
The valudation accuracy is 81.34%

EPOCHS : 8/10 Loss : 0.971869
The training accuracy is 59.97%
The validation loss is 0.5269
The valudation accuracy is 81.63%

EPOCHS : 9/10 Loss : 0.954651
The training accuracy is 60.34%
The validation loss is 0.4988
The valudation accuracy is 83.03%

EPOCHS : 9/10 Loss : 0.958505
The training accuracy is 60.65%
The validation loss is 0.4952
The valudation accuracy is 82.98%

EPOCHS : 10/10 Loss : 0.949912
The training accuracy is 60.97%
The validation loss is 0.4980
The valudation accuracy is 82.53%

EPOCHS : 10/10 Loss : 0.922114
The training accuracy is 61.29%
The validation loss is 0.4982
The valudation accuracy is 82.74%
```

[30]:

```
optimizer = optim.Adam(AlexNet2.parameters(),lr=1e-7)
train(AlexNet2,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",2,"c")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 0.983893
The training accuracy is 65.21%
The validation loss is 0.4936
The valudation accuracy is 83.12%

EPOCHS : 1/10 Loss : 0.904120
The training accuracy is 66.81%
The validation loss is 0.4828
The valudation accuracy is 83.32%

EPOCHS : 2/10 Loss : 0.906161
The training accuracy is 67.25%
The validation loss is 0.4769
The valudation accuracy is 83.34%

EPOCHS : 2/10 Loss : 0.900338
The training accuracy is 67.54%
The validation loss is 0.4763
The valudation accuracy is 83.48%

EPOCHS : 3/10 Loss : 0.921114
The training accuracy is 67.56%
The validation loss is 0.4737
The valudation accuracy is 83.46%

EPOCHS : 3/10 Loss : 0.895418
The training accuracy is 67.72%
The validation loss is 0.4758
The valudation accuracy is 83.38%

EPOCHS : 4/10 Loss : 0.905156
The training accuracy is 67.83%
The validation loss is 0.4768
The valudation accuracy is 83.53%

EPOCHS : 4/10 Loss : 0.887147
The training accuracy is 67.99%
The validation loss is 0.4795
The valudation accuracy is 83.49%

EPOCHS : 5/10 Loss : 0.894696
The training accuracy is 68.05%
The validation loss is 0.4749
The valudation accuracy is 83.54%

EPOCHS : 5/10 Loss : 0.898819
The training accuracy is 68.12%
The validation loss is 0.4786
The valudation accuracy is 83.61%

EPOCHS : 6/10 Loss : 0.897295
The training accuracy is 68.15%
The validation loss is 0.4721
The valudation accuracy is 83.72%

EPOCHS : 6/10 Loss : 0.907410
The training accuracy is 68.16%
The validation loss is 0.4769
The valudation accuracy is 83.68%

EPOCHS : 7/10 Loss : 0.901949
The training accuracy is 68.16%
The validation loss is 0.4778
The valudation accuracy is 83.59%

EPOCHS : 7/10 Loss : 0.892131
The training accuracy is 68.21%
The validation loss is 0.4749
The valudation accuracy is 83.58%

EPOCHS : 8/10 Loss : 0.893424
The training accuracy is 68.25%
The validation loss is 0.4760
The valudation accuracy is 83.66%

EPOCHS : 8/10 Loss : 0.907078
The training accuracy is 68.25%
The validation loss is 0.4715
The valudation accuracy is 83.68%

EPOCHS : 9/10 Loss : 0.907165
The training accuracy is 68.24%
The validation loss is 0.4700
The valudation accuracy is 83.64%

EPOCHS : 9/10 Loss : 0.888679
The training accuracy is 68.26%
The validation loss is 0.4740
The valudation accuracy is 83.61%

EPOCHS : 10/10 Loss : 0.898744
```

```
The training accuracy is 68.29%
The validation loss is 0.4734
The valudation accuracy is 83.69%

EPOCHS : 10/10 Loss : 0.893277
The training accuracy is 68.30%
The validation loss is 0.4694
The valudation accuracy is 83.58%
```

[31]:

```
optimizer = optim.Adam(AlexNet2.parameters(),lr=1e-8)
#AlexNet.load_state_dict(torch.load("Weights/part-c-run-2-Best.pth"))
train(AlexNet2,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",3,"c")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 0.960729
The training accuracy is 65.96%
The validation loss is 0.4728
The valudation accuracy is 83.51%

EPOCHS : 1/10 Loss : 0.884539
The training accuracy is 67.56%
The validation loss is 0.4713
The valudation accuracy is 83.52%

EPOCHS : 2/10 Loss : 0.894978
The training accuracy is 67.94%
The validation loss is 0.4709
The valudation accuracy is 83.56%

EPOCHS : 2/10 Loss : 0.900989
The training accuracy is 68.03%
The validation loss is 0.4811
The valudation accuracy is 83.58%

EPOCHS : 3/10 Loss : 0.895619
The training accuracy is 68.20%
The validation loss is 0.4710
The valudation accuracy is 83.62%

EPOCHS : 3/10 Loss : 0.904552
The training accuracy is 68.23%
The validation loss is 0.4682
The valudation accuracy is 83.59%

EPOCHS : 4/10 Loss : 0.898023
The training accuracy is 68.27%
The validation loss is 0.4718
The valudation accuracy is 83.64%

EPOCHS : 4/10 Loss : 0.896598
The training accuracy is 68.29%
The validation loss is 0.4702
The valudation accuracy is 83.62%

EPOCHS : 5/10 Loss : 0.886057
The training accuracy is 68.36%
The validation loss is 0.4696
The valudation accuracy is 83.62%

EPOCHS : 5/10 Loss : 0.891967
The training accuracy is 68.42%
The validation loss is 0.4730
The valudation accuracy is 83.62%

EPOCHS : 6/10 Loss : 0.902300
The training accuracy is 68.40%
The validation loss is 0.4713
The valudation accuracy is 83.61%

EPOCHS : 6/10 Loss : 0.888475
The training accuracy is 68.45%
The validation loss is 0.4702
The valudation accuracy is 83.65%

EPOCHS : 7/10 Loss : 0.886854
The training accuracy is 68.51%
The validation loss is 0.4714
The valudation accuracy is 83.63%

EPOCHS : 7/10 Loss : 0.900989
The training accuracy is 68.50%
The validation loss is 0.4767
The valudation accuracy is 83.65%

EPOCHS : 8/10 Loss : 0.898051
The training accuracy is 68.50%
The validation loss is 0.4729
The valudation accuracy is 83.66%

EPOCHS : 8/10 Loss : 0.888562
The training accuracy is 68.51%
The validation loss is 0.4689
The valudation accuracy is 83.66%

EPOCHS : 9/10 Loss : 0.882809
```

```
The training accuracy is 68.55%
The validation loss is 0.4671
The valudation accuracy is 83.67%

EPOCHS : 9/10 Loss : 0.898687
The training accuracy is 68.53%
The validation loss is 0.4736
The valudation accuracy is 83.69%

EPOCHS : 10/10 Loss : 0.892271
The training accuracy is 68.54%
The validation loss is 0.4707
The valudation accuracy is 83.68%

EPOCHS : 10/10 Loss : 0.904148
The training accuracy is 68.52%
The validation loss is 0.4711
The valudation accuracy is 83.65%
```

[ ]:

[ ]:

[ ]:

[ ]:

6 Part d)

[32]:

```
AlexNet3 = models.alexnet(pretrained=False)
AlexNet3.load_state_dict(torch.load("W://Study Material/Jupyter Notebook/
 ↪Pretrained_Weights/alexnet-owt-7be5be79.pth"))
AlexNet3.cuda()
```

[32]:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

[34]:

```
AlexNet3.classifier[6] = nn.Linear(4096,10,True)
print(AlexNet3)
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=10, bias=True)
  )
)
```

[35]:

```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(AlexNet3.parameters(),lr=1e-5)
```

[36]:

```
train(AlexNet3,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",1,"d")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 1.850561
The training accuracy is 32.47%
The validation loss is 1.2623
The valudation accuracy is 55.76%

EPOCHS : 1/10 Loss : 1.405315
The training accuracy is 40.75%
The validation loss is 0.9747
The valudation accuracy is 65.92%

EPOCHS : 2/10 Loss : 1.257076
The training accuracy is 45.55%
The validation loss is 0.8586
The valudation accuracy is 69.45%

EPOCHS : 2/10 Loss : 1.177570
The training accuracy is 48.66%
The validation loss is 0.7658
The valudation accuracy is 73.17%

EPOCHS : 3/10 Loss : 1.127938
The training accuracy is 50.88%
The validation loss is 0.6838
The valudation accuracy is 76.32%

EPOCHS : 3/10 Loss : 1.089887
The training accuracy is 52.55%
The validation loss is 0.6361
The valudation accuracy is 78.05%

EPOCHS : 4/10 Loss : 1.061739
The training accuracy is 53.94%
The validation loss is 0.6482
The valudation accuracy is 77.48%

EPOCHS : 4/10 Loss : 1.048316
The training accuracy is 55.06%
The validation loss is 0.5840
The valudation accuracy is 79.73%

EPOCHS : 5/10 Loss : 1.028207
The training accuracy is 56.01%
The validation loss is 0.5783
The valudation accuracy is 79.75%

EPOCHS : 5/10 Loss : 0.997779
The training accuracy is 56.87%
The validation loss is 0.5529
The valudation accuracy is 80.76%

EPOCHS : 6/10 Loss : 0.981257
The training accuracy is 57.63%
The validation loss is 0.5608
The valudation accuracy is 80.42%

EPOCHS : 6/10 Loss : 0.972149
The training accuracy is 58.28%
The validation loss is 0.5433
The valudation accuracy is 81.03%

EPOCHS : 7/10 Loss : 0.949519
The training accuracy is 58.90%
The validation loss is 0.5389
The valudation accuracy is 81.30%

EPOCHS : 7/10 Loss : 0.933250
The training accuracy is 59.49%
The validation loss is 0.5344
The valudation accuracy is 81.23%

EPOCHS : 8/10 Loss : 0.917924
The training accuracy is 60.05%
The validation loss is 0.5027
The valudation accuracy is 82.17%

EPOCHS : 8/10 Loss : 0.908061
The training accuracy is 60.55%
The validation loss is 0.4832
The valudation accuracy is 83.11%

EPOCHS : 9/10 Loss : 0.900666
The training accuracy is 60.99%
The validation loss is 0.4841
The valudation accuracy is 83.14%
```

```
EPOCHS : 9/10 Loss : 0.893229
The training accuracy is 61.41%
The validation loss is 0.4815
The valudation accuracy is 83.27%

EPOCHS : 10/10 Loss : 0.887517
The training accuracy is 61.80%
The validation loss is 0.4666
The valudation accuracy is 83.78%

EPOCHS : 10/10 Loss : 0.872435
The training accuracy is 62.16%
The validation loss is 0.4633
The valudation accuracy is 83.90%
```

[37]:
```
optimizer = optim.Adam(AlexNet3.parameters(),lr=1e-6)
train(AlexNet3,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",2,"d")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 1.000398
The training accuracy is 64.65%
The validation loss is 0.4515
The valudation accuracy is 84.18%

EPOCHS : 1/10 Loss : 0.840051
The training accuracy is 67.60%
The validation loss is 0.4455
The valudation accuracy is 84.52%

EPOCHS : 2/10 Loss : 0.841163
The training accuracy is 68.52%
The validation loss is 0.4427
The valudation accuracy is 84.39%

EPOCHS : 2/10 Loss : 0.834193
The training accuracy is 69.07%
The validation loss is 0.4453
The valudation accuracy is 84.51%

EPOCHS : 3/10 Loss : 0.842139
The training accuracy is 69.33%
The validation loss is 0.4452
The valudation accuracy is 84.52%

EPOCHS : 3/10 Loss : 0.832247
The training accuracy is 69.59%
The validation loss is 0.4508
The valudation accuracy is 84.45%

EPOCHS : 4/10 Loss : 0.829938
The training accuracy is 69.76%
The validation loss is 0.4495
The valudation accuracy is 84.32%

EPOCHS : 4/10 Loss : 0.840508
The training accuracy is 69.87%
The validation loss is 0.4373
The valudation accuracy is 84.74%

EPOCHS : 5/10 Loss : 0.830376
The training accuracy is 69.95%
The validation loss is 0.4497
The valudation accuracy is 84.30%

EPOCHS : 5/10 Loss : 0.828192
The training accuracy is 70.06%
The validation loss is 0.4463
The valudation accuracy is 84.54%

EPOCHS : 6/10 Loss : 0.817075
The training accuracy is 70.18%
The validation loss is 0.4493
The valudation accuracy is 84.43%

EPOCHS : 6/10 Loss : 0.840949
The training accuracy is 70.18%
The validation loss is 0.4396
The valudation accuracy is 84.62%

EPOCHS : 7/10 Loss : 0.830831
The training accuracy is 70.21%
The validation loss is 0.4396
The valudation accuracy is 84.62%

EPOCHS : 7/10 Loss : 0.815412
The training accuracy is 70.30%
The validation loss is 0.4503
The valudation accuracy is 84.29%

EPOCHS : 8/10 Loss : 0.811264
The training accuracy is 70.35%
The validation loss is 0.4431
The valudation accuracy is 84.75%

EPOCHS : 8/10 Loss : 0.815634
```

```
The training accuracy is 70.40%
The validation loss is 0.4378
The valudation accuracy is 84.83%

EPOCHS : 9/10 Loss : 0.819109
The training accuracy is 70.46%
The validation loss is 0.4370
The valudation accuracy is 84.89%

EPOCHS : 9/10 Loss : 0.816168
The training accuracy is 70.52%
The validation loss is 0.4373
The valudation accuracy is 84.64%

EPOCHS : 10/10 Loss : 0.823191
The training accuracy is 70.53%
The validation loss is 0.4414
The valudation accuracy is 84.70%

EPOCHS : 10/10 Loss : 0.817268
The training accuracy is 70.57%
The validation loss is 0.4332
The valudation accuracy is 84.92%
```

[38]:
```
optimizer = optim.Adam(AlexNet3.parameters(),lr=1e-8)
#AlexNet.load_state_dict(torch.load("Weights/part-d-run-2-Best.pth"))
train(AlexNet3,train_dataloader,val_dataloader,10,20,loss_fn,optimizer,"cuda",3,"d")
```

```
  0%|          | 0/3130 [00:00<?, ?it/s]

EPOCHS : 1/10 Loss : 0.991773
The training accuracy is 64.99%
The validation loss is 0.4360
The valudation accuracy is 84.82%

EPOCHS : 1/10 Loss : 0.805074
The training accuracy is 68.39%
The validation loss is 0.4337
The valudation accuracy is 84.83%

EPOCHS : 2/10 Loss : 0.811057
The training accuracy is 69.44%
The validation loss is 0.4410
The valudation accuracy is 84.85%

EPOCHS : 2/10 Loss : 0.822433
The training accuracy is 69.90%
The validation loss is 0.4325
The valudation accuracy is 84.87%

EPOCHS : 3/10 Loss : 0.820671
The training accuracy is 70.17%
The validation loss is 0.4350
The valudation accuracy is 84.86%

EPOCHS : 3/10 Loss : 0.810344
The training accuracy is 70.41%
The validation loss is 0.4347
The valudation accuracy is 84.86%

EPOCHS : 4/10 Loss : 0.813434
The training accuracy is 70.58%
The validation loss is 0.4321
The valudation accuracy is 84.92%

EPOCHS : 4/10 Loss : 0.816192
The training accuracy is 70.63%
The validation loss is 0.4329
The valudation accuracy is 84.91%

EPOCHS : 5/10 Loss : 0.803453
The training accuracy is 70.76%
The validation loss is 0.4387
The valudation accuracy is 84.91%

EPOCHS : 5/10 Loss : 0.815467
The training accuracy is 70.82%
The validation loss is 0.4337
The valudation accuracy is 84.91%

EPOCHS : 6/10 Loss : 0.811431
The training accuracy is 70.85%
The validation loss is 0.4321
The valudation accuracy is 84.94%

EPOCHS : 6/10 Loss : 0.815065
The training accuracy is 70.88%
The validation loss is 0.4348
The valudation accuracy is 84.96%

EPOCHS : 7/10 Loss : 0.806256
The training accuracy is 70.94%
The validation loss is 0.4310
The valudation accuracy is 84.98%

EPOCHS : 7/10 Loss : 0.807493
```

```
The training accuracy is 70.99%
The validation loss is 0.4324
The valudation accuracy is 84.95%

EPOCHS : 8/10 Loss : 0.812556
The training accuracy is 71.01%
The validation loss is 0.4330
The valudation accuracy is 84.94%

EPOCHS : 8/10 Loss : 0.814946
The training accuracy is 71.05%
The validation loss is 0.4334
The valudation accuracy is 84.96%

EPOCHS : 9/10 Loss : 0.814383
The training accuracy is 71.06%
The validation loss is 0.4323
The valudation accuracy is 84.96%

EPOCHS : 9/10 Loss : 0.804518
The training accuracy is 71.10%
The validation loss is 0.4302
The valudation accuracy is 84.96%

EPOCHS : 10/10 Loss : 0.820406
The training accuracy is 71.08%
The validation loss is 0.4344
The valudation accuracy is 84.96%

EPOCHS : 10/10 Loss : 0.801487
The training accuracy is 71.12%
The validation loss is 0.4318
The valudation accuracy is 84.97%
```

```
[ ]:
```