Lab3

December 6, 2020

[1]:

**import torch**

**import torch.nn as nn**

**import torch.optim as optim import torch.utils.data**

**import torch.nn.functional as F import torchvision**

**from torchvision import** transforms

**from PIL import** Image

**import matplotlib.pyplot as plt import numpy as np**

%**matplotlib** inline

# *1*

[2]:

LEARNING\_RATE = 0.01

EPOCHS = 50

TRAIN\_DATA\_SIZE = 10000

[3]:

**def** check\_image(path):

**try**:

im = Image.open(path)

**return True except**:

**return False**

[4]:

img\_transforms = transforms.Compose([ transforms.Resize((64,64)), transforms.Grayscale(), transforms.ToTensor()

])

[5]:

train\_data\_path = "./train/" train\_data = torchvision.datasets.

*<→*ImageFolder(root=train\_data\_path,transform=img\_transforms,␣

*<→*is\_valid\_file=check\_image)

len(train\_data)

[5]:

60000

[6]:

val\_data\_path = "./val/"

val\_data = torchvision.datasets.

*<→*ImageFolder(root=val\_data\_path,transform=img\_transforms,␣

*<→*is\_valid\_file=check\_image)

[7]:

test\_data\_path = "./test/" test\_data = torchvision.datasets.

*<→*ImageFolder(root=test\_data\_path,transform=img\_transforms,␣

*<→*is\_valid\_file=check\_image)

[8]:

batch\_size=TRAIN\_DATA\_SIZE

[9]:

train\_data\_loader = torch.utils.data.DataLoader(train\_data,␣

*<→*batch\_size=batch\_size)

val\_data\_loader = torch.utils.data.DataLoader(val\_data, batch\_size=batch\_size) test\_data\_loader = torch.utils.data.DataLoader(test\_data,␣

*<→*batch\_size=batch\_size)

[10]:

**class SimpleNet**(nn.Module):

**def** init (self):

super(SimpleNet, self). init () self.fc1 = nn.Linear(4096, 250) self.fc2 = nn.Linear(250, 50) self.fc3 = nn.Linear(50,10)

**def** forward(self, x):

x = x.view(-1, 64\*64)

x = F.relu(self.fc1(x)) x = F.relu(self.fc2(x)) x = self.fc3(x)

**return** x

[11]:

simplenet = SimpleNet()

[12]:

optimizer = optim.Adam(simplenet.parameters(), lr=LEARNING\_RATE)

[13]:

**if** torch.cuda.is\_available(): device = torch.device("cuda")

**else**:

device = torch.device("cpu")

simplenet.to(device)

/home/blestrong/anaconda3/lib/python3.8/site-packages/torch/cuda/ init .py:52:

UserWarning: CUDA initialization: The NVIDIA driver on your system is too old (found version 6050). Please update your GPU driver by downloading and

installing a new version from the URL: <http://www.nvidia.com/Download/index.aspx> Alternatively, go to: https://pytorch.org to install a PyTorch version that has been compiled with your version of the CUDA driver. (Triggered internally at

/opt/conda/conda-bld/pytorch\_1603729096996/work/c10/cuda/CUDAFunctions.cpp:100.) return torch.\_C.\_cuda\_getDeviceCount() > 0

[13]:

SimpleNet(

(fc1): Linear(in\_features=4096, out\_features=250, bias=True) (fc2): Linear(in\_features=250, out\_features=50, bias=True) (fc3): Linear(in\_features=50, out\_features=10, bias=True)

)

[14]: **def** train(model, optimizer, loss\_fn, train\_loader, val\_loader, epochs=20,␣

*<→*device="cpu"):

**for** epoch **in** range(epochs): training\_loss = 0.0

valid\_loss = 0.0 model.train()

**for** batch **in** train\_loader: optimizer.zero\_grad() inputs, targets = batch

inputs = inputs.to(device) targets = targets.to(device) output = model(inputs)

loss = loss\_fn(output, targets) loss.backward()

optimizer.step()

training\_loss += loss.data.item() \* inputs.size(0) training\_loss /= len(train\_loader.dataset)

model.eval() num\_correct = 0

num\_examples = 0

**for** batch **in** val\_loader: inputs, targets = batch inputs = inputs.to(device) output = model(inputs)

targets = targets.to(device) loss = loss\_fn(output,targets)

valid\_loss += loss.data.item() \* inputs.size(0)

correct = torch.eq(torch.max(F.softmax(output), dim=1)[1], targets).

*<→*view(-1)

num\_correct += torch.sum(correct).item() num\_examples += correct.shape[0]

valid\_loss /= len(val\_loader.dataset)

print('Epoch: **{}**, Training Loss: **{:.2f}**, Validation Loss: **{:.2f}**,␣

*<→*accuracy = **{:.2f}**'.format(epoch, training\_loss, valid\_loss, num\_correct / num\_examples))

[15]:

train(simplenet, optimizer,torch.nn.CrossEntropyLoss(),␣

*<→*train\_data\_loader,val\_data\_loader, epochs=EPOCHS, device=device)

<ipython-input-14-f38d79bc5988>:28: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.

correct = torch.eq(torch.max(F.softmax(output), dim=1)[1], targets).view(-1)

Epoch: 0, Training Loss: 5.08, Validation Loss: 2.19, accuracy = 0.16

Epoch: 1, Training Loss: 2.25, Validation Loss: 2.11, accuracy = 0.14

Epoch: 2, Training Loss: 2.07, Validation Loss: 2.06, accuracy = 0.18

Epoch: 3, Training Loss: 2.05, Validation Loss: 1.96, accuracy = 0.31

Epoch: 4, Training Loss: 1.93, Validation Loss: 1.82, accuracy = 0.34

Epoch: 5, Training Loss: 1.75, Validation Loss: 1.65, accuracy = 0.39

Epoch: 6, Training Loss: 1.58, Validation Loss: 1.50, accuracy = 0.41

Epoch: 7, Training Loss: 1.42, Validation Loss: 1.36, accuracy = 0.48

Epoch: 8, Training Loss: 1.27, Validation Loss: 1.24, accuracy = 0.55

Epoch: 9, Training Loss: 1.19, Validation Loss: 1.17, accuracy = 0.57

Epoch: 10, Training Loss: 1.11, Validation Loss: 1.08, accuracy = 0.60

Epoch: 11, Training Loss: 1.01, Validation Loss: 1.02, accuracy = 0.61

Epoch: 12, Training Loss: 0.98, Validation Loss: 0.95, accuracy = 0.65

Epoch: 13, Training Loss: 0.87, Validation Loss: 0.87, accuracy = 0.68

Epoch: 14, Training Loss: 0.84, Validation Loss: 0.79, accuracy = 0.73

Epoch: 15, Training Loss: 0.81, Validation Loss: 0.83, accuracy = 0.71

Epoch: 16, Training Loss: 0.77, Validation Loss: 0.69, accuracy = 0.76

Epoch: 17, Training Loss: 0.71, Validation Loss: 0.72, accuracy = 0.76

Epoch: 18, Training Loss: 0.79, Validation Loss: 0.72, accuracy = 0.75

Epoch: 19, Training Loss: 0.73, Validation Loss: 0.60, accuracy = 0.79

Epoch: 20, Training Loss: 0.66, Validation Loss: 0.59, accuracy = 0.81

Epoch: 21, Training Loss: 0.55, Validation Loss: 0.54, accuracy = 0.84

Epoch: 22, Training Loss: 0.54, Validation Loss: 0.49, accuracy = 0.86

Epoch: 23, Training Loss: 0.47, Validation Loss: 0.44, accuracy = 0.87

Epoch: 24, Training Loss: 0.45, Validation Loss: 0.42, accuracy = 0.87

Epoch: 25, Training Loss: 0.43, Validation Loss: 0.40, accuracy = 0.87

Epoch: 26, Training Loss: 0.43, Validation Loss: 0.41, accuracy = 0.88

Epoch: 27, Training Loss: 0.40, Validation Loss: 0.39, accuracy = 0.88

Epoch: 28, Training Loss: 0.37, Validation Loss: 0.39, accuracy = 0.88

Epoch: 29, Training Loss: 0.36, Validation Loss: 0.35, accuracy = 0.89

Epoch: 30, Training Loss: 0.33, Validation Loss: 0.34, accuracy = 0.90

Epoch: 31, Training Loss: 0.32, Validation Loss: 0.33, accuracy = 0.90

Epoch: 32, Training Loss: 0.30, Validation Loss: 0.32, accuracy = 0.90

Epoch: 33, Training Loss: 0.30, Validation Loss: 0.31, accuracy = 0.91

Epoch: 34, Training Loss: 0.28, Validation Loss: 0.30, accuracy = 0.91

Epoch: 35, Training Loss: 0.28, Validation Loss: 0.29, accuracy = 0.91

[16]:

Epoch: 36, Training Loss: 0.27, Validation Loss: 0.29, accuracy = 0.91

Epoch: 37, Training Loss: 0.26, Validation Loss: 0.28, accuracy = 0.91

Epoch: 38, Training Loss: 0.25, Validation Loss: 0.27, accuracy = 0.92

Epoch: 39, Training Loss: 0.25, Validation Loss: 0.27, accuracy = 0.92

Epoch: 40, Training Loss: 0.24, Validation Loss: 0.26, accuracy = 0.92

Epoch: 41, Training Loss: 0.24, Validation Loss: 0.26, accuracy = 0.92

Epoch: 42, Training Loss: 0.23, Validation Loss: 0.25, accuracy = 0.92

Epoch: 43, Training Loss: 0.23, Validation Loss: 0.25, accuracy = 0.92

Epoch: 44, Training Loss: 0.23, Validation Loss: 0.25, accuracy = 0.93

Epoch: 45, Training Loss: 0.24, Validation Loss: 0.26, accuracy = 0.92

Epoch: 46, Training Loss: 0.25, Validation Loss: 0.25, accuracy = 0.92

Epoch: 47, Training Loss: 0.26, Validation Loss: 0.26, accuracy = 0.92

Epoch: 48, Training Loss: 0.26, Validation Loss: 0.26, accuracy = 0.92

Epoch: 49, Training Loss: 0.24, Validation Loss: 0.26, accuracy = 0.92

#

labels = ['0','1', '2', '3', '4', '5', '6', '7', '8', '9']

img = Image.open("./test/0/4675.png") img = img\_transforms(img).to(device)

plt.imshow( img.permute(1, 2, 0), cmap ='gray' ) prediction = F.softmax(simplenet(img))

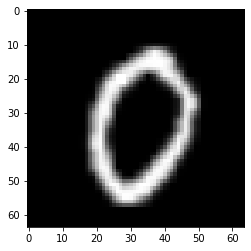
prediction = prediction.argmax()

print(labels[prediction])

0

<ipython-input-16-d51a979042ec>:8: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.

prediction = F.softmax(simplenet(img))



[17]:

num\_examples = len(test\_data) num\_correct = 0

**for** data **in** test\_data:

prediction = F.softmax(simplenet(data[0])) prediction = prediction.argmax()

**if** data[1] == prediction: num\_correct += 1

print("Accuracy = **{}**".format(num\_correct/num\_examples))

<ipython-input-17-033a5615afd5>:4: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.

prediction = F.softmax(simplenet(data[0]))

Accuracy = 0.950018443378827

# *2*

[18]:

torch.save(simplenet.state\_dict(), "tmp/simplenet\_2")

# *3*

[19]:

simplenet = SimpleNet()

simplenet\_state\_dict = torch.load("tmp/simplenet\_2") simplenet.load\_state\_dict(simplenet\_state\_dict)

[19]:

<All keys matched successfully>

[ ]: