Assignment 10

# Output from DL20.A

Text

Description automatically generated

# Output

Text

Description automatically generated

# Comments

As you can observe in the screenshots above, the modifications produce far better results.

# Code

# DL20A.py CS5173/6073 cheng 2023  
# 2D LeNet5 on MNIST  
# following d2l 7.6.1  
# Usage: python DL20A.py  
  
import torch  
import torchvision  
import torch.nn as nn  
import numpy as np  
import random  
import time  
  
mnist = torchvision.datasets.MNIST('/data/')  
num\_samples = len(mnist)  
x = []  
targets = []  
for i in range(num\_samples):  
 x.append(list(mnist[i][0].getdata()))  
 targets.append(mnist[i][1])  
  
X = torch.tensor(x, dtype=torch.float32)  
X2 = torch.reshape(X, (len(X), 1, 28, 28))  
y = torch.tensor(targets, dtype=torch.long)  
  
#Last two be 1x1 Conv2d (kernel\_size=1) and  
# the first replacement LazyConv2d with a kernel as big as the H and W in the input 4d tensor (N, C, H, W).  
model = nn.Sequential(  
 nn.LazyConv2d(6, kernel\_size=5, padding=2), nn.ReLU(),  
 nn.AvgPool2d(kernel\_size=2),  
 nn.LazyConv2d(16, kernel\_size=5), nn.ReLU(),  
 nn.AvgPool2d(kernel\_size=2),  
 nn.LazyConv2d(120, kernel\_size=5), nn.ReLU(),  
 nn.LazyConv2d(84, kernel\_size=1), nn.ReLU(),  
 nn.LazyConv2d(10, kernel\_size=1),  
 nn.Flatten())  
  
loss\_fun = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters())  
  
batch\_size = 512  
rounds = 1000  
indices = list(range(num\_samples))  
t1 = time.process\_time()  
for i in range(rounds):  
 random.shuffle(indices)  
 batch\_indices = torch.tensor(indices[:batch\_size])  
 Xbatch = X2[batch\_indices]  
 ybatch = y[batch\_indices]  
 o = model(Xbatch)  
 loss = loss\_fun(o, ybatch)  
 if i % 100 == 0:  
 print(i, loss.item())  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
  
t2 = time.process\_time()  
print('Training time', t2 - t1)  
  
print(loss.item())  
o = model(X2)  
ypred = torch.argmax(o, dim=1)  
misclassified = torch.sum((ypred != y))  
print('Training misclassified =', misclassified.item(), 'out of', num\_samples)  
  
testset = torchvision.datasets.MNIST('/data/', train=False)  
num\_test = len(testset)  
testImg = []  
testTgt = []  
for i in range(num\_test):  
 testImg.append(list(testset[i][0].getdata()))  
 testTgt.append(testset[i][1])  
Xtest = torch.tensor(testImg, dtype=torch.float32)  
ytest = torch.tensor(testTgt, dtype=torch.long)  
X2test = torch.reshape(Xtest, (len(Xtest), 1, 28, 28))  
o = model(X2test)  
ypred = torch.argmax(o, dim=1)  
misclassified = torch.sum((ypred != ytest))  
print('Test misclassified =', misclassified.item(), 'out of', num\_test)