**Deep Learning Homework III Report**

Name: Dazhi Li

RUID: 197007456

**I only did the option B for homework 3 which is achieved by pytorch**

Source Code:

# -\*- coding: utf-8 -\*-  
"""  
Created on Sat Apr 11 16:08:10 2020  
  
@author: DzL  
"""  
  
from \_\_future\_\_ import print\_function  
import numpy as np  
import torch  
import torch.nn as nn  
import torch.nn.functional as F  
import torch.optim as optim  
from torchvision import datasets, transforms  
import time  
  
#defining FC layers  
Model=torch.nn.Sequential(  
 nn.Linear(784,200),  
 nn.ReLU(),  
 nn.Linear(200,50),  
 nn.ReLU(),  
 nn.Linear(50,10))  
  
#defining training progress  
def train(model,device,train\_loader,optimizer,loss,epoch):  
 model.train()  
 count=0  
 for batch\_idx,(data,target) in enumerate(train\_loader):  
 data,target=data.to(device),target.to(device)  
   
 optimizer.zero\_grad()  
 data=data.view(-1,784)  
 output=model(data)  
 if loss=='CE':  
 #CrossEntropy Loss  
 loss\_fn=nn.CrossEntropyLoss()  
 if loss=='MSE':  
 #MSE Loss  
 y\_onehot=target.numpy()  
 y\_onehot=(np.arange(10)==y\_onehot[:,None]).astype(np.float32)  
 target=torch.from\_numpy(y\_onehot)  
 loss\_fn=nn.MSELoss()  
   
 loss\_=loss\_fn(output,target)  
 loss\_.backward()  
 optimizer.step()  
 if batch\_idx %10==0:  
 print('Train Epoch:{}[{}/{}({:.0f}%)]\tLoss:{:.6f}'.format(  
 epoch,batch\_idx\*len(data),len(train\_loader.dataset),  
 100. \* batch\_idx / len(train\_loader),loss\_.item()))  
  
#defining testing  
def test(model,device,test\_loader):  
 model.eval()  
 test\_loss=0  
 correct=0  
 with torch.no\_grad():  
 for data,target in test\_loader:  
 data,target=data.to(device),target.to(device)  
 data=data.view(-1,784)  
 output=model(data)  
 test\_loss+=F.nll\_loss(output,target,reduction='sum').item()  
 #sum up batch loss,negative log likelihood loss  
   
 pred=output.argmax(dim=1,keepdim=True)  
 #get the index of the max log-probability  
 correct+=pred.eq(target.view\_as(pred)).sum().item()  
 test\_loss/=len(test\_loader.dataset)  
   
 print('\nTest set: Average loss:{:.4f},Accuracy:{}/{} ({:.0f}%)\n'.format(  
 test\_loss,correct,len(test\_loader.dataset),  
 100. \* correct / len(test\_loader.dataset)))  
   
#main function  
def main():  
 device=torch.device("cpu")  
 #load MNIST dataset  
 batch\_size=128  
 test\_batch\_size=10000  
 train\_loader=torch.utils.data.DataLoader(  
 datasets.MNIST('./data',train=True,download=True,  
 transform=transforms.Compose([  
 transforms.ToTensor(),  
 transforms.Normalize((0.1307,),(0.3081,))  
 ])),  
 batch\_size=batch\_size,shuffle=True)  
 test\_loader=torch.utils.data.DataLoader(  
 datasets.MNIST('./data',train=False,transform=transforms.Compose([  
 transforms.ToTensor(),  
 transforms.Normalize((0.1307,),(0.3081,))  
 ])),  
 batch\_size=test\_batch\_size,shuffle=True)  
 #set optimizer  
 lr=0.01  
 model=Model.to(device)  
 optimizer=optim.SGD(model.parameters(),lr=lr)  
 time0=time.time()  
 #Training settings  
 epochs=10  
 loss='CE'  
 #start training  
 time0=time.time()  
 for epoch in range(1,epochs+1):  
 train(model,device,train\_loader,optimizer,loss,epoch)  
 test(model,device,test\_loader)  
 time1=time.time()  
 print('Training and Testing total excution time is: %s seconds ' %(time1-time0))  
  
if \_\_name\_\_=='\_\_main\_\_':  
 main()

**Problem procedure:**

I first set up my training model at the beginning of this code. It has one input and output layer with two hidden fully connected layers. Every hidden layer is followed by a ReLU function. Then I set up training procedure which use batch SGD. Those parameters like batch size epochs are set in the main function. And we can choose the loss metrics either MSE nor CE. But the MSE loss will run in an error when we utilize our GPU to do the training which means, we can not set our device as CUDA when we want to use MSE loss. By the way, MSE loss will have a lower accuracy than CE loss. Then I defined the test procedure which will return the accuracy for every epoch. In the main function I defined the learning rate, batch size, epoch number etc. to fit my training and testing procedure. I found that CUDA is faster than CPU.

**Output of Problem 1:**

**图片包含 游戏机

描述已自动生成**

This is the output running on CPU, CUDA will have the approximate accuracy to this one but 10 seconds faster.