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:

```
Train Epoch:10[10240/60000(17%)]    Loss:0.140338
Train Epoch:10[11520/60000(19%)]    Loss:0.213212
Train Epoch:10[12800/60000(21%)]    Loss:0.180049
Train Epoch:10[14080/60000(23%)]    Loss:0.273110
Train Epoch:10[15360/60000(26%)]    Loss:0.145817
Train Epoch:10[16640/60000(28%)]    Loss:0.313394
Train Epoch:10[17920/60000(30%)]    Loss:0.156250
Train Epoch:10[19200/60000(32%)]    Loss:0.151344
Train Epoch:10[20480/60000(34%)]    Loss:0.221896
Train Epoch:10[21760/60000(36%)]    Loss:0.228174
Train Epoch:10[23040/60000(38%)]    Loss:0.144919
Train Epoch:10[24320/60000(41%)]    Loss:0.174937
Train Epoch:10[25600/60000(43%)]    Loss:0.264954
Train Epoch:10[26880/60000(45%)]    Loss:0.132715
Train Epoch:10[28160/60000(47%)]    Loss:0.158535
Train Epoch:10[29440/60000(49%)]    Loss:0.096584
Train Epoch:10[30720/60000(51%)]    Loss:0.248466
Train Epoch:10[32000/60000(53%)]    Loss:0.235512
Train Epoch:10[33280/60000(55%)]    Loss:0.160129
Train Epoch:10[34560/60000(58%)]    Loss:0.176314
Train Epoch:10[35840/60000(60%)]    Loss:0.262626
Train Epoch:10[37120/60000(62%)]    Loss:0.150558
Train Epoch:10[38400/60000(64%)]    Loss:0.079039
Train Epoch:10[39680/60000(66%)]    Loss:0.179232
Train Epoch:10[40960/60000(68%)]    Loss:0.157148
Train Epoch:10[42240/60000(70%)]    Loss:0.164342
Train Epoch:10[43520/60000(72%)]    Loss:0.224384
Train Epoch:10[44800/60000(75%)]    Loss:0.217212
Train Epoch:10[46080/60000(77%)]    Loss:0.175867
Train Epoch:10[47360/60000(79%)]    Loss:0.128427
Train Epoch:10[48640/60000(81%)]    Loss:0.229815
Train Epoch:10[49920/60000(83%)]    Loss:0.135569
Train Epoch:10[51200/60000(85%)]    Loss:0.266428
Train Epoch:10[52480/60000(87%)]    Loss:0.193230
Train Epoch:10[53760/60000(90%)]    Loss:0.185333
Train Epoch:10[55040/60000(92%)]    Loss:0.179046
Train Epoch:10[56320/60000(94%)]    Loss:0.148755
Train Epoch:10[57600/60000(96%)]    Loss:0.185013
Train Epoch:10[58880/60000(98%)]    Loss:0.126292

Test set: Average loss:-9.2771,Accuracy:9490/10000 (95%)

Training and Testing total excution time is: 100.68777227401733 seconds
```

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