Spring 2023 Introduction to Deep Learning

Homework Assignment 3

Due date: April 2 2023

Problem (Build a FC layer-based neural network to recognize hand-written digits). In this problem, you are asked to train and test a neural network for ***entire*** MNIST handwritten digit dataset. Some information of the network is as follows:

- Its structure is **784-200-50-10**. Here 784 means the input layer has 784 input neurons. This is because each image in MNIST dataset is 28x28 and you need to stretch them to a length-784 vector. 200 and 50 are the number of neurons in hidden layers. 10 is the number of neurons in output layer since there are 10 types of digits.
- The two hidden layers are followed by **ReLU layers**.
- The output layer is a **softmax** layer.

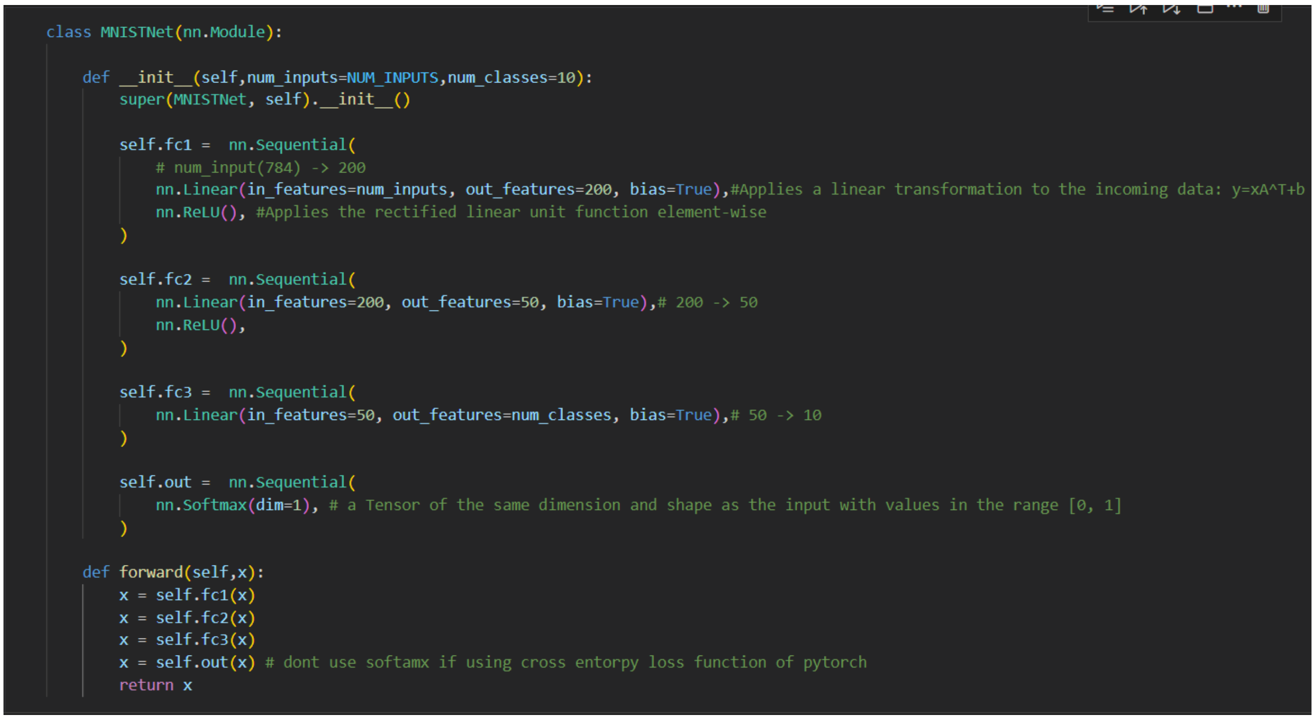
(a) **(Mandatory)** Use deep learning framework to train and test this network. You are allowed to use the corresponding autograd or nn module to train the network.

(b) **(Optional)** Use only Numpy to train and test this network. You are NOT allowed to use deep learning framework (e.g. Pytorch, Tensorflow etc.) and the corresponding autograd or nn module to train the network.

Performance Requirement and Submission:

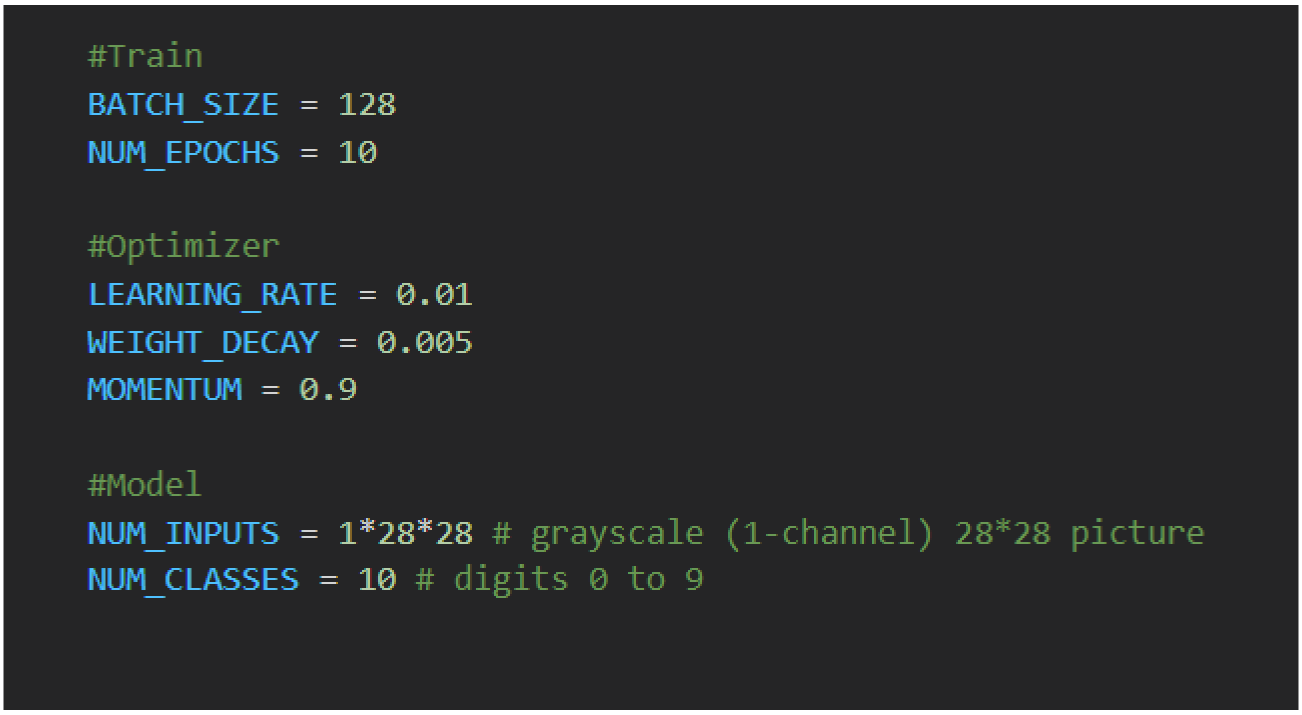
- The test accuracy should achieve above 95%
- Submission should include your source codes and screen snapshot of your train and test accuracy, plus the training time

Model :-



```python
class MNISTNet(nn.Module):

    def __init__(self,num_inputs=NUM_INPUTS,num_classes=10):
        super(MNISTNet, self).__init__()

        self.fc1 =  nn.Sequential(
            # num_input(784) -> 200
            nn.Linear(in_features=num_inputs, out_features=200, bias=True),#Applies a linear transformation to the incoming data: y=xA^T+b
            nn.ReLU(), #Applies the rectified linear unit function element-wise
        )

        self.fc2 =  nn.Sequential(
            nn.Linear(in_features=200, out_features=50, bias=True),# 200 -> 50
            nn.ReLU(),
        )

        self.fc3 =  nn.Sequential(
            nn.Linear(in_features=50, out_features=num_classes, bias=True),# 50 -> 10
        )

        self.out =  nn.Sequential(
            nn.Softmax(dim=1), # a Tensor of the same dimension and shape as the input with values in the range [0, 1]
        )

    def forward(self,x):
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        x = self.out(x) # dont use softamx if using cross entorpy loss function of pytorch
        return x
```

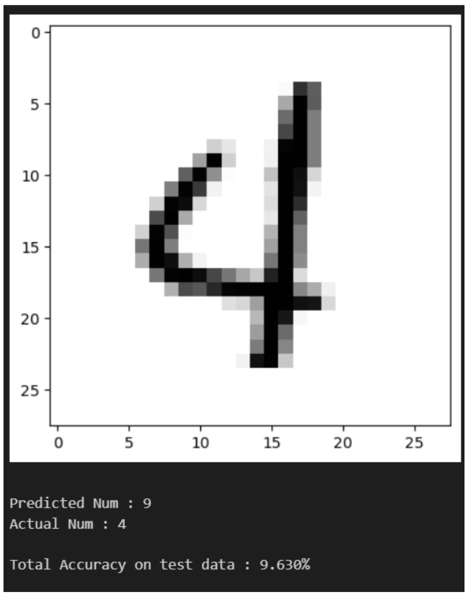
Hyperparameters :-



```python
#Train
BATCH_SIZE = 128
NUM_EPOCHS = 10

#Optimizer
LEARNING_RATE = 0.01
WEIGHT_DECAY = 0.005
MOMENTUM = 0.9

#Model
NUM_INPUTS = 1*28*28 # grayscale (1-channel) 28*28 picture
NUM_CLASSES = 10 # digits 0 to 9
```

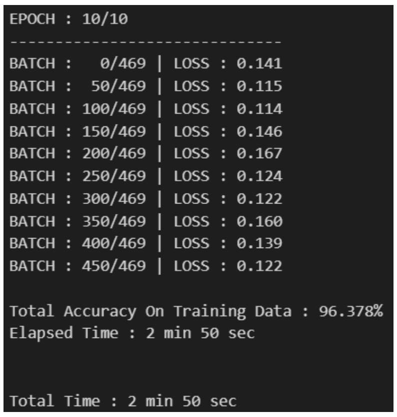
Results :-

Accuracy Without Training



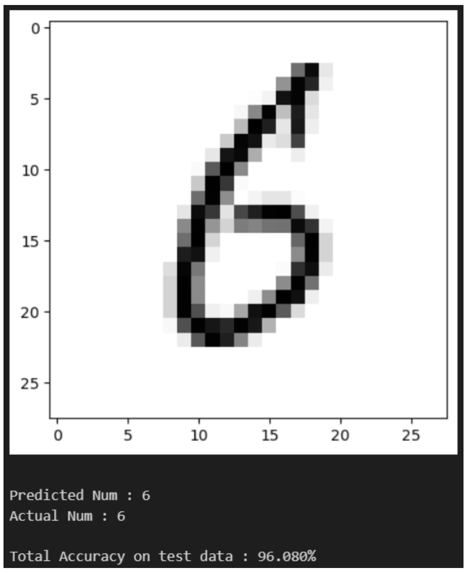
```
Predicted Num : 9
Actual Num : 4

Total Accuracy on test data : 9.630%
```

Training



```
EPOCH : 10/10
------------------------------
BATCH :   0/469 | LOSS : 0.141
BATCH :  50/469 | LOSS : 0.115
BATCH : 100/469 | LOSS : 0.114
BATCH : 150/469 | LOSS : 0.146
BATCH : 200/469 | LOSS : 0.167
BATCH : 250/469 | LOSS : 0.124
BATCH : 300/469 | LOSS : 0.122
BATCH : 350/469 | LOSS : 0.160
BATCH : 400/469 | LOSS : 0.139
BATCH : 450/469 | LOSS : 0.122

Total Accuracy On Training Data : 96.378%
Elapsed Time : 2 min 50 sec


Total Time : 2 min 50 sec
```

Accuracy After Training



```
Predicted Num : 6
Actual Num : 6

Total Accuracy on test data : 96.080%
```