Lusine Kamikyan
Homework 4 Report
BME 595A
Collaborated with Emma Reid

Part A:

class MyImg2num:

1. NeuralNetwork API - from homework 3
2. `MyImg2Num` API

   __init__() method:

   I initialize the NeuralNetwork class. I have 28^2 input neurons, 1 hidden layer with 100 neurons, and an output layer with 10 neurons. The batch size is 500

   train() method:

   I load the data using torch dataloader. I shuffle the training data at the beginning of each epoch. I define learning rate eta to be 0.5, number of epochs = 50
   I loop through the batches of the training data, reshape it to be 28^2 by Batch_size. I change the labels of the training data to one-hot. Then I run the NeuralNetwork forward function to get the output of the forward pass, then the backward, then update the parameters. I then find the loss function for the training, and plot it.

   To get the loss plot for the test data, I do almost the same as above. Rechape the test data, switch the labels to one-hot, then run the forward function of NN, find the loss values between the output of forward and labels of the test data, and plot it.

   forward():

   Takes in only one image, changes it to a float tensor, then calls the NN forward function. It returns the label of the output of the forward of NN. I use to check the test data and accuracy.

   Accuracy is 92.15%
   See plots below:

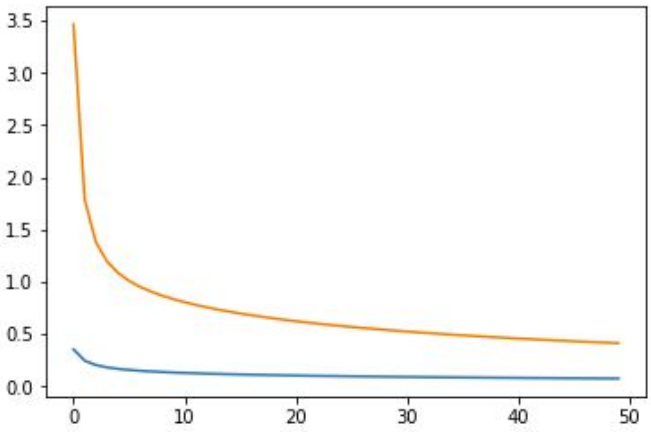

Figure1: Epoch vs training loss
Epoch vs test loss

The test loss is much lower than the training loss, and this is ok, since a lot depends on the test data. Like it was told in class, the training data has learned to recognize 7 for example very well and maybe the test data has a lot of 7's, so the training results will be much better on the test data.

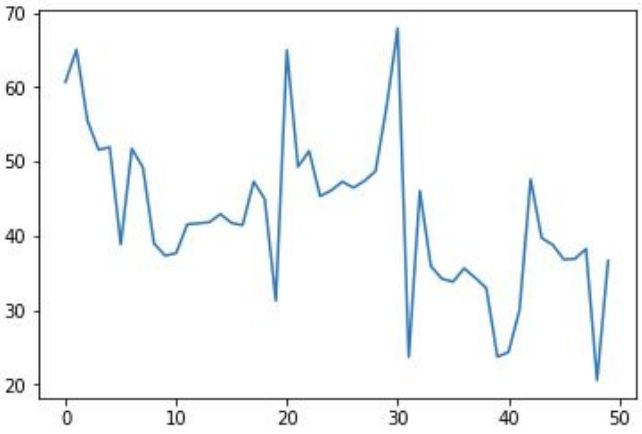

Figure2: Epoch vs time

Part B:

class NnImg2Num:

__init__():

Define some local variables - batch size = 500, input layer neurons = 28^2, 1 hidden layer with 100 neurons, an output layer with 10 neurons

train():

I load the data. Number of epochs is 50. Create the forward model with 1 hidden layer and the sigmoid activation function for each layer.
It's the same process as above only using the torch.nn build in functions instead of my NeuralNetwork.

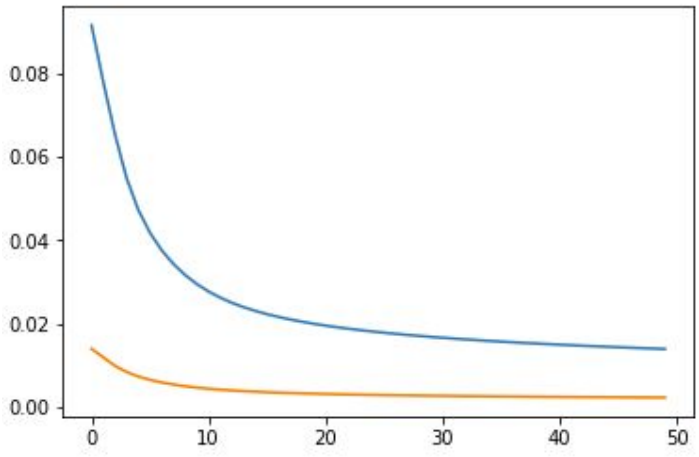

Figure3: Epoch vs training loss
Epoch vs test loss

We can see that the error is much lower when using the torch.nn than when using my NeuralNetwork class

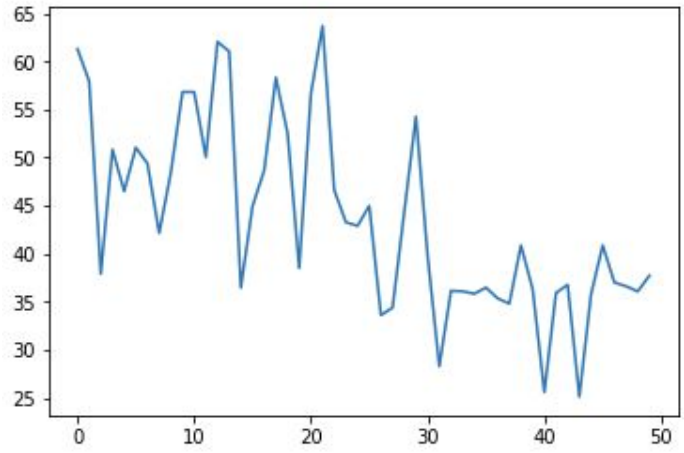

Figure4: Epoch vs time (time starts before training and ends after the validation check)

They seem to have similar speed of running