Homework 1

Problem 1

Play around with different Leaky ReLU slopes. What is the best slope you could find? What happens if you set the slope > 1? What about slope < 0. Theoretically, what happens if you set slope $= 1$?

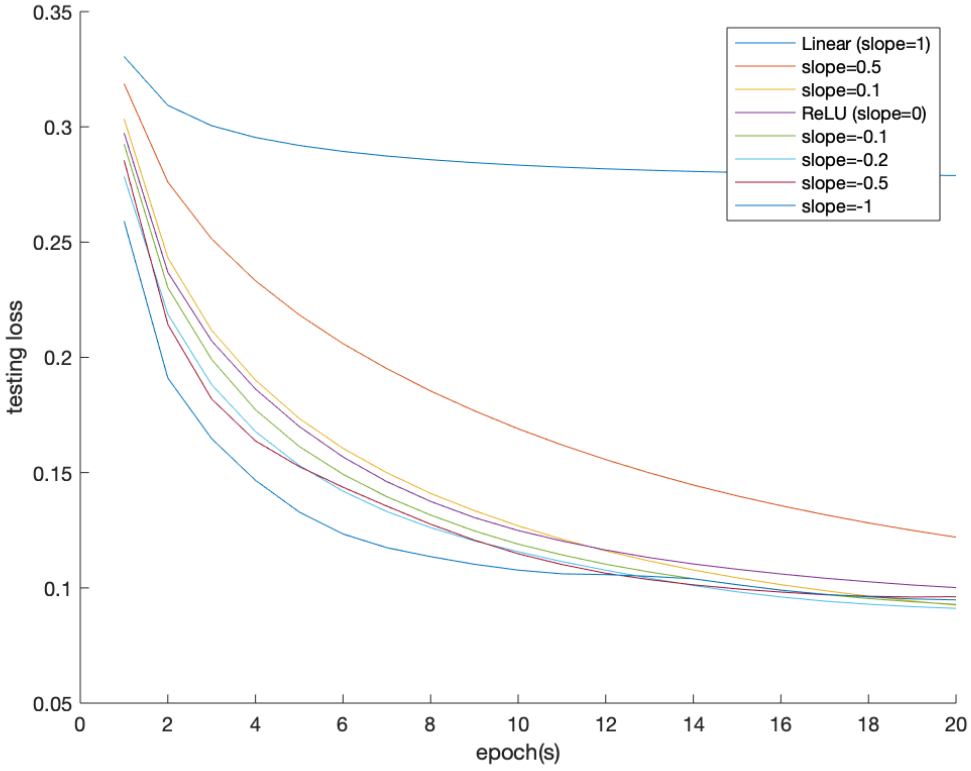

Figure 1: Testing Loss of different values of slope

As shown in the figure, the best value of Leaky ReLU slope I found using our own neural network is -0.5.

When we set the slope with value > 1, we can observe that the performance of the neural network begin to fall as we increase the value of the slope. The training will failed as the slope value become something around 3 as the loss become "nan" which will failed the updates of weights resulting in unreliable model.

When we set the slope with value < 0, we can observe that the performances are actually slightly better then those value > 0 which theoretically should provide more info by distinguishing the positive and negative inputs.

Theoretically, if we set the slope to 1, the Leaky ReLU will become a simple linear function, which give exactly same result with the network using only Linear Layers.

```
MNISTNetwork
    (network): SequentialLayer
        (0): LinearLayer
            (784, 1000)
        (1): LeakyReLULayer
        (2): LinearLayer
            (1000, 100)
        (3): LeakyReLULayer
        (4): LinearLayer
            (100, 10)
    (loss_layer): SoftmaxCrossEntropyLossLayer
```

Listing 1: Default Network Layout for experiment in Problem 1

Problem 2

Set PReLU to take 1 slope per layer. After 20 epochs, what were your PReLU slopes? Does this correspond with what you found in question 1?

The PReLU slopes after 20 epochs with initial values of 0.1 the average over 5 different tests is -0.1253 for the first layer and -0.2142 for the second layer as the final loss being 0.096 .

This experiment results are quite similar to the pattern in question 1 as the Leaky ReLU with the slope being -0.2 outperform other selection of value, including Linear and ReLU activation.

Problem 3

If you add more layers and more epochs, what accuracy can you reach? Can you get to 99%? What is your best network layout?

I reach a accuracy of 98.4% which is really close to 99% by add an additional layer and modifying the input and output setting of each layer. The learning rate remain 0.01 with momentum being 0.9. The batch size is set to 256 as I found out that smaller batch size can have pretty significant changes on the accuracy and lo even though the training duration will be longer.

```
MNISTNetwork
    (network): SequentialLayer
        (0): LinearLayer
            (784, 1024)
        (1): PReLULayer
        (2): LinearLayer
```

```
7               (1024, 512)
8           (3): PReLULayer
9           (4): LinearLayer
10              (512, 64)
11          (5): PReLULayer
12          (6): LinearLayer
13              (64, 10)
14      (loss_layer): SoftmaxCrossEntropyLossLayer
```

Listing 2: Best Network Layout