# Assignment #00010

Ruiqi Liu 949919908

02/12/2022 19:14PM

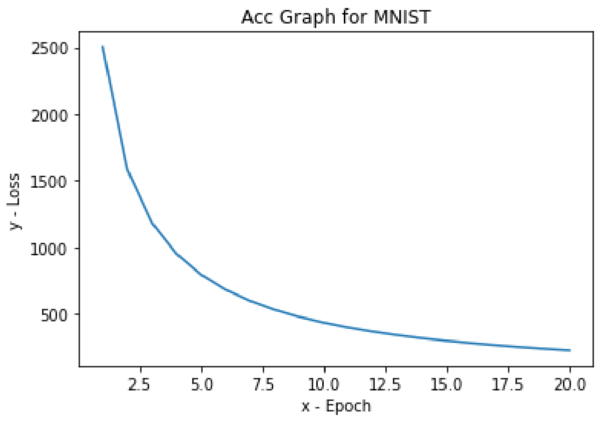
## 1 MLP for MNIST and FashionMNIST

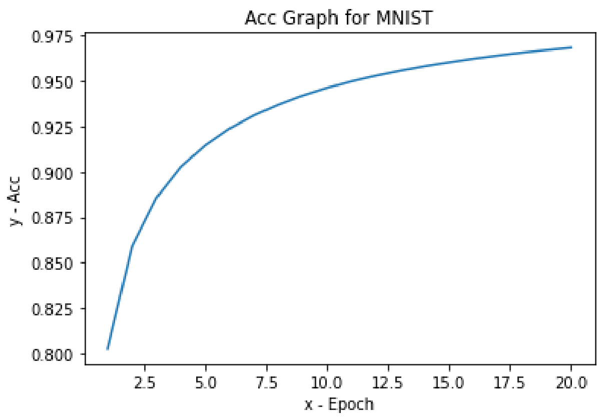
We design a network structure which has two hidden layers with $inputSize = 784$, no drop out, $hiddenLayerSize = 256$ and $outputsize = 10$. Because MNIST and Fashion MNIST dataset are both consist of images which dimension is 28*28 and finally be classified in 10 categories. We use Sparse Categorical Cross entropy to be considered as its loss function.

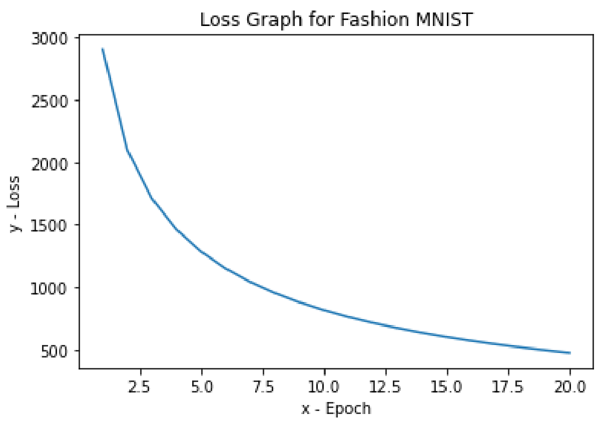
**Github url**: `https://github.com/BlackParure/Multi-Layer-Perceptron`

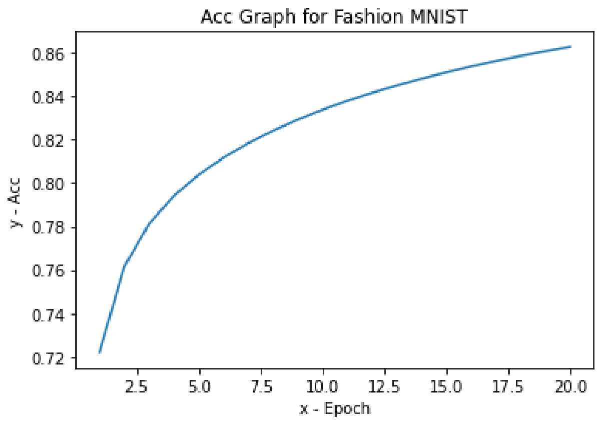
For optimizer, we use ADAM for better performance. Here are my final result on both MNIST and Fashion MNIST.

| Dataset | TrainACC | TrainLoss | TestACC | TestLoss |
|---|---|---|---|---|
| MNIST | 0.968 | 226.802 | 0.964 | 232.444 |
| Fashion MNIST | 0.863 | 477.366 | 0.834 | 292.773 |







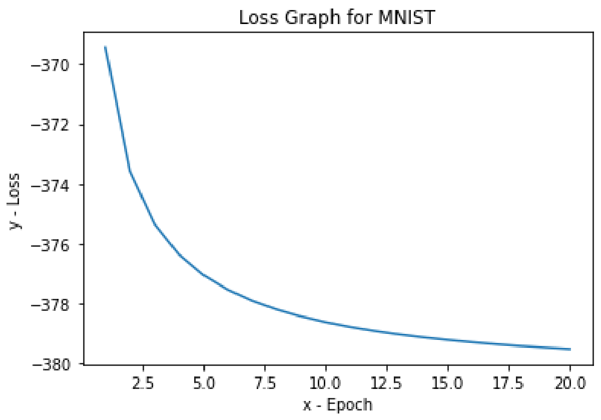


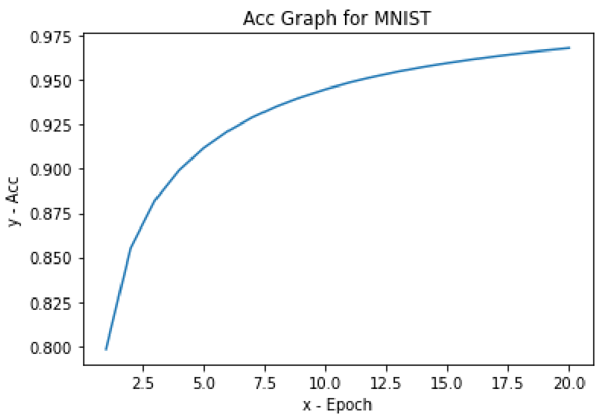
## 2 Algorithm Design

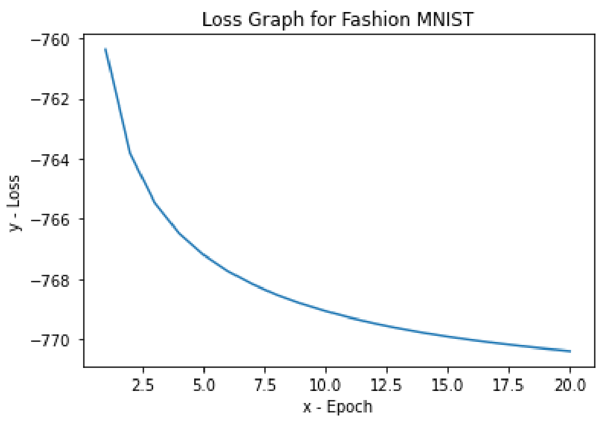
I add normalization to the network structure by dividing all input images with 255.0 to normalize them into a scale of 0 to 1. And also apply L1L2 regularizer by minus an extra penelty term in order to reduce the effect of loss item.

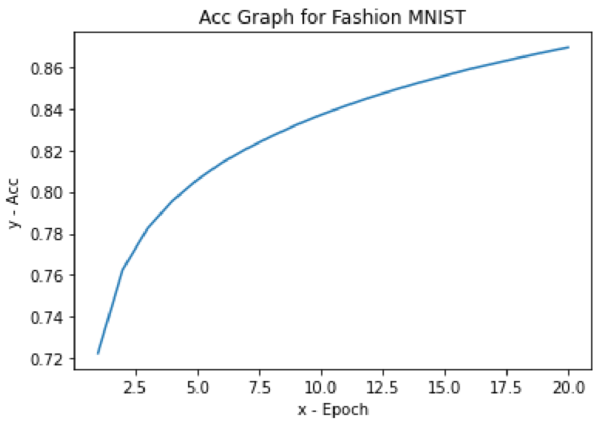
The result of MNIST and Fashion MNIST under regularization approaches like that,

| Dataset | TrainACC | TrainLoss(L1L2) | TestACC | TestLoss(L1L2) |
|---|---|---|---|---|
| MNIST | 0.968 | -379.534 | 0.966 | -385.598 |
| Fashion MNIST | 0.869 | -770.383 | 0.857 | -772.490 |









We can find that with more complex model the training accuracy will keep in a same level (e.g. For Fashion MNIST, from 0.863 to 0.869), but the loss in simpler model will smaller that a complex one (e.g. For Fashion MNIST, from 477 to 770). When we test this model on new images, the accuracy will greatly increase (e.g. For Fashion MNIST, from 0.834 to 0.857) and loss also increase. This shows that simpler model will be overfitting which will result in lower variance and higher bias.

## 3 Hyper-parameter Optimization

All parameters we need to adjust includes:

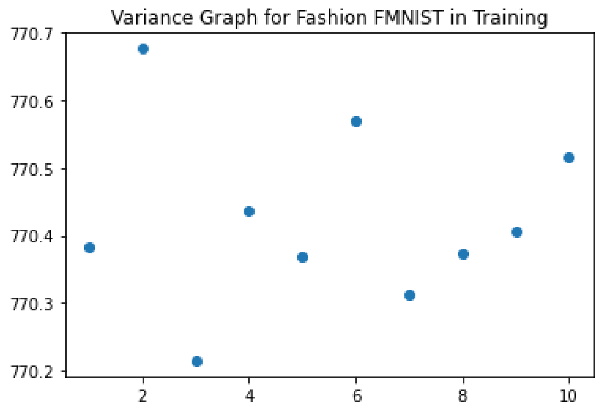
1. Dropout: True/False;
2. Regularizer: L1L2/None;
3. L1: Numerical; L2: Numerical;
4. Learning Rate: Numerical;
5. Normalization: True/False;

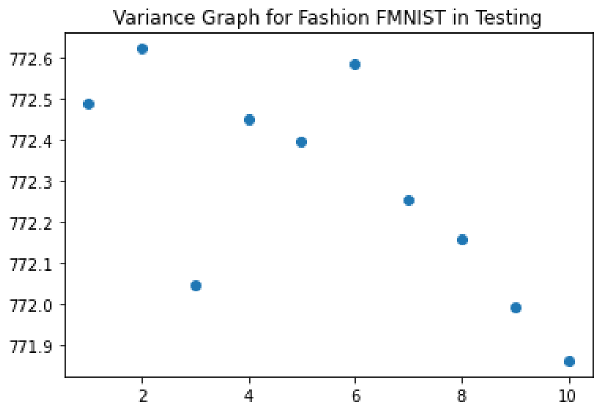
First, we choose one parameter need to adjust and fix others, for options like Dropout and Regularizer we just need to choose take it or not. But for numerical parameter we prefer to test it on exponential basis.
For Dropout, we found that the accuracy drops a lot after applying it, so we deactivate it.
For Normalization, the process time(605.62->589.46) and accuracy(0.834->0.857) are all increased after applying it. So we decided to use this method.
For Regularization, we found that L1L2 will greatly increase the accuracy(0.854->0.867), so we choose to use L1L2 regularizer.
Then we need to adjust numerical parameters, for learning rate, 1e-4 can only reach 0.73 accuracy after 10 epoch because it converges too slow; at the same time, 0.1 will result in first epoch with only 0.51 accuracy then continuous drop. So we choose 0.001 for learning rate.
For L1L2, these parameters decides the weights of L1 and L2 regularization terms respectively. We choose 0.1 for both of them which can balance the speed and performance at the same time.

## 4 Multiple trials

We change the seeds and test them on Fashion MNIST dataset. The result are listed below.

| Seed | TrainACC | TrainLoss(L1L2) | TestACC | TestLoss(L1L2) |
|---|---|---|---|---|
| 1234 | 0.869 | -770.383 | 0.857 | -772.490 |
| 1111 | 0.870 | -770.677 | 0.856 | -772.624 |
| 2222 | 0.868 | -770.214 | 0.851 | -772.049 |
| 3333 | 0.870 | -770.437 | 0.853 | -772.451 |
| 4444 | 0.868 | -770.368 | 0.849 | -772.399 |
| 5555 | 0.870 | -770.570 | 0.859 | -772.587 |
| 6666 | 0.867 | -770.313 | 0.851 | -772.256 |
| 7777 | 0.869 | -770.374 | 0.847 | -772.161 |
| 8888 | 0.869 | -770.405 | 0.842 | -771.993 |
| 9999 | 0.868 | -772.517 | 0.845 | -771.863 |


Variance Graph for Fashion FMNIST in Training
770.7
770.6
770.5
770.4
770.3
770.2
2
4
6
8
10


Variance Graph for Fashion FMNIST in Testing
772.6
772.5
772.4
772.3
772.2
772.1
772.0
771.9
2
4
6
8
10