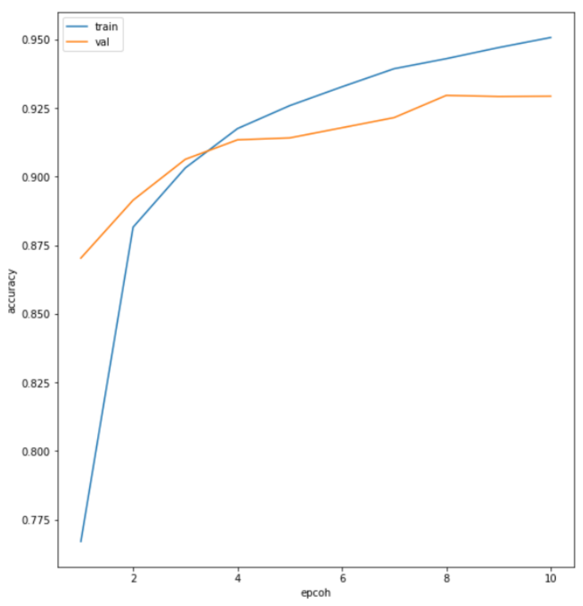
Introduction

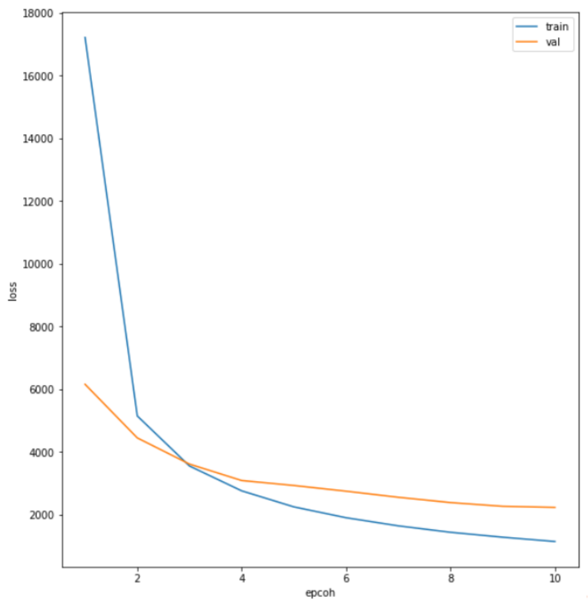
In this assignment, I will build a mlp with 2 hidden layers to perform image classification. The code is referenced from Ankur Mali, https://colab.research.google.com/github/AnkurMali/IST597_Spring_2022/blob/main/week3/IST597_MLP_collab.ipynb#printMode=true&scrollTo=moAeRMJ56kr6.
I will try different regularization approaches like l1, l2, and dropout. I will find the best hyperparameters for the model. I will also draw plots to show training process and testing results. The detailed code can be viewed at my github: https://github.com/Yuxuan-Liu-Eason/IST-597-00010

MNIST

I first find appropriate hyperparameters. For hyperparameter tunning, I basically use manual grid search. The object hyperparameters are hidden size, learning rate, lambda for regularization, dropout rate. The fixed parameters are epoch = 10, batch size = 20, device = 'gpu', optimizer = Adam. For example, I have a set of hidden sizes [500/200, 300/100, 256/128] and a set of learning rates [0.01, 0.001, 0.0001]. Then I run multiple times to find the best combination. The best parameters I found are hidden sizes = 300/100 (two layers), lr = 0.001, dropout rate = 0.1.

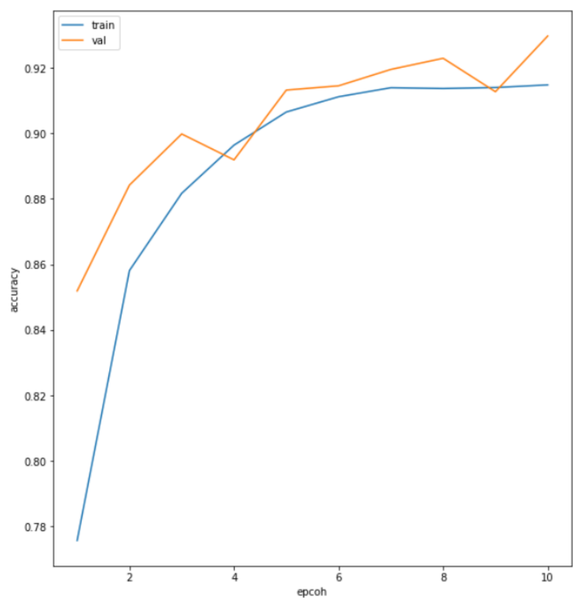
For the base mlp, I use the above best parameters. For time reasons, I only run 10 epochs for now. After 10 epochs, the train accuracy is 0.9507 and validation accuracy is 0.93. Below is the training plot.

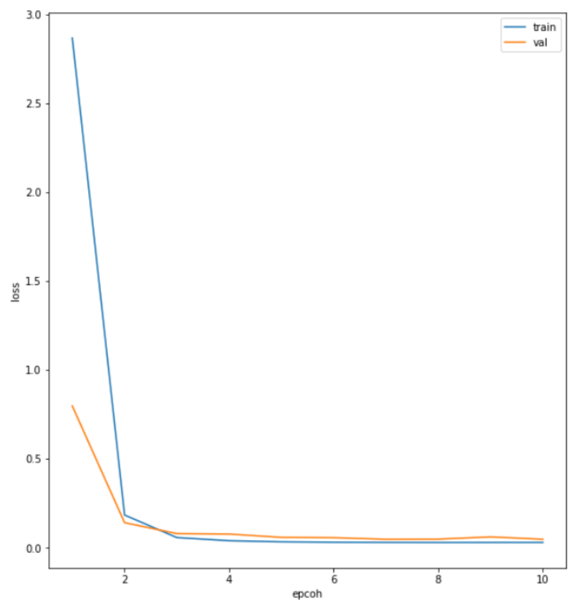




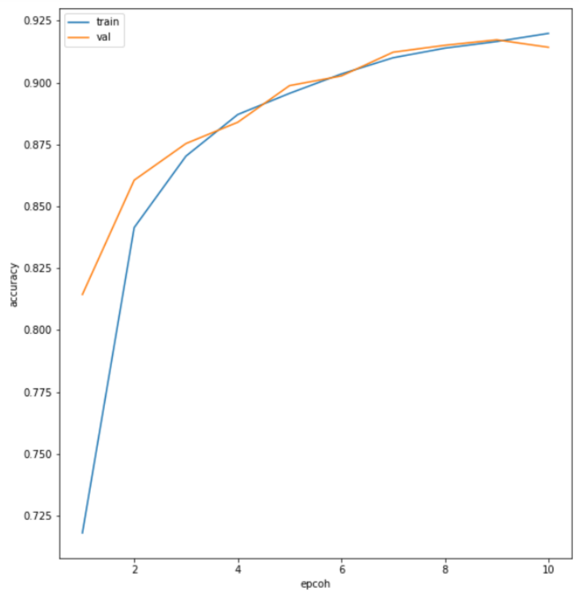
From the plot, we can see that validation has a lower accuracy, which implies that there exists overfitting. It also means there is high variance and low bias. Regularizations should be considered.

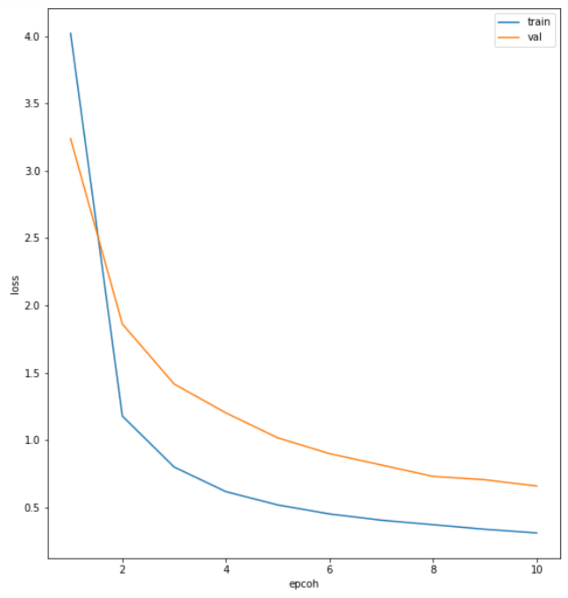
Then I add a normalization by dividing the data by 255. However, normalization does not improve the accuracy very much. The train accuracy is 0.9571 and validation accuracy is 0.933. The overfitting still exists.
Then I keep adding l1 penalty for all W and b. Through search, I find that lambda = 1e-5 gives the best results. The train accuracy is 0.8569 and test accuracy is 0.8603. However, a accuracy of 0.8603 is too low, which means the l1 penalty is not helpful. In this case, variance is low but bias is high.
Then I delete l1 penalty and add l2 penalty for all W and b with lambda = 1e-5. The train accuracy is 0.9148 and test accuracy is 0.9297. As we can see from the training plot, the overfitting is eliminated. In this case, both variance and bias are relatively low.





Then I delete l2 and try dropout with 0.1. I also get relatively good results. The train accuracy is 0.9199 and test accuracy is 0.9184. In this case, both variance and bias are relatively low.

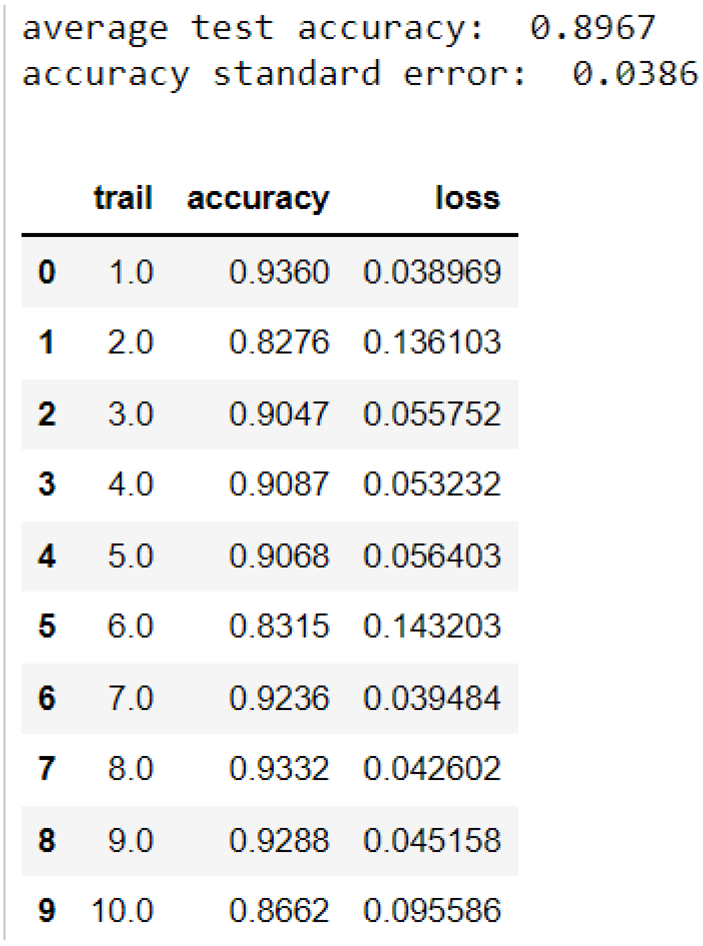




To summarize, below is the table of my attempts and their results. Again, for the training, I only use 10 epochs for time reasons. I believe the accuracy will be higher I have more time to run more epochs.

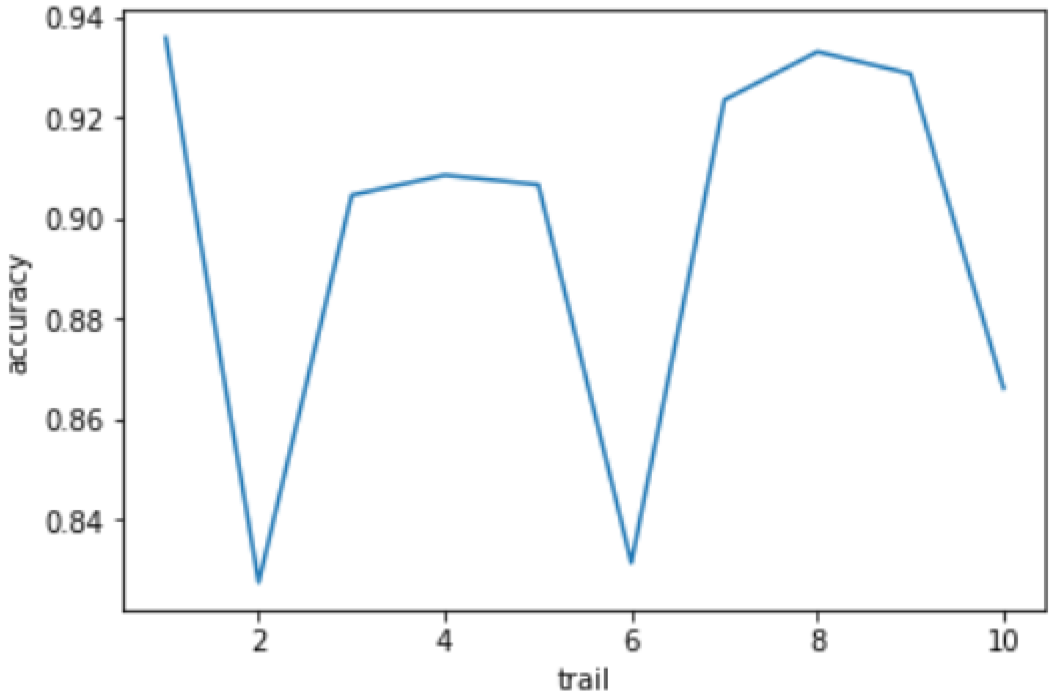
Hidden size (2 layers)	Learning rate	penalty	normalization	dropout	Train accuracy	Validation accuracy
300/100	0.001	none	none	none	0.9507	0.9300
300/100	0.001	none	yes	none	0.9571	0.9330
300/100	0.001	L1: 1e-5	yes	none	0.8569	0.8603
300/100	0.001	L2:1e-5	yes	none	0.9148	0.9297
300/100	0.001	L1+L2:1e-5	yes	none	0.8675	0.8847
300/100	0.001	none	yes	0.1	0.9199	0.9184
300/100	0.001	L2	yes	0.1	0.9022	0.9038

For the final model, I decide to use l2 penalty and normalization, which is the one in red. After 10 trails, I got the following accuracies with mean 0.8967 and standard error 0.0386.



```
average test accuracy:  0.8967
accuracy standard error:  0.0386
```

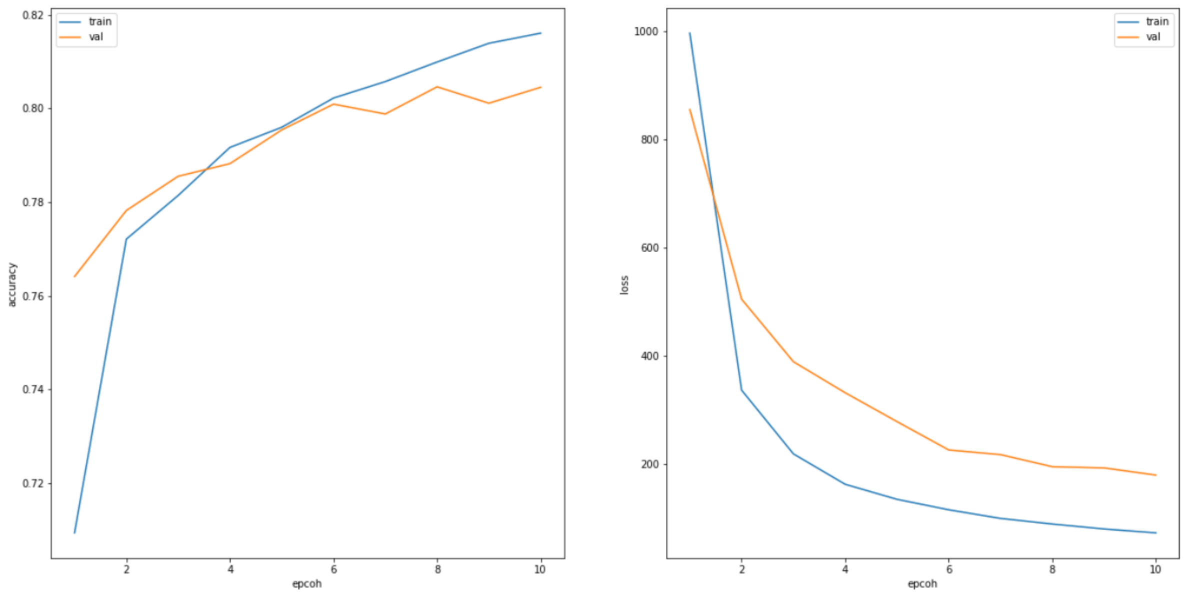
	trail	accuracy	loss
0	1.0	0.9360	0.038969
1	2.0	0.8276	0.136103
2	3.0	0.9047	0.055752
3	4.0	0.9087	0.053232
4	5.0	0.9068	0.056403
5	6.0	0.8315	0.143203
6	7.0	0.9236	0.039484
7	8.0	0.9332	0.042602
8	9.0	0.9288	0.045158
9	10.0	0.8662	0.095586



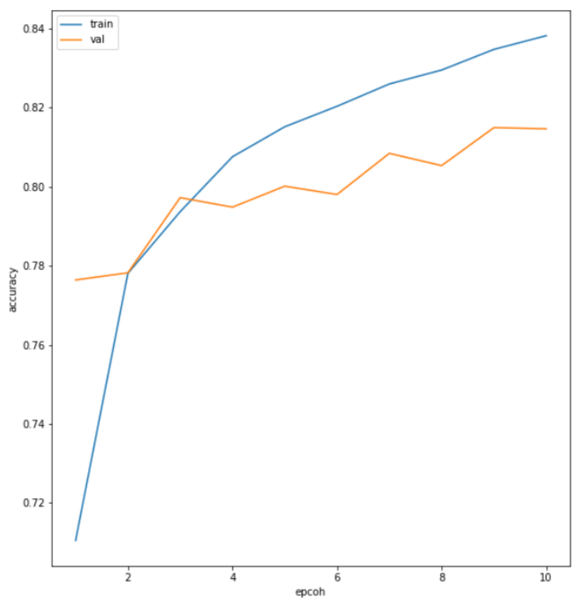
It is surprising to see that the mean test accuracy is lower than the validation accuracy. Also, the variance of the accuracies are large. The lowest accuracy is only about 0.83, which is unusal.

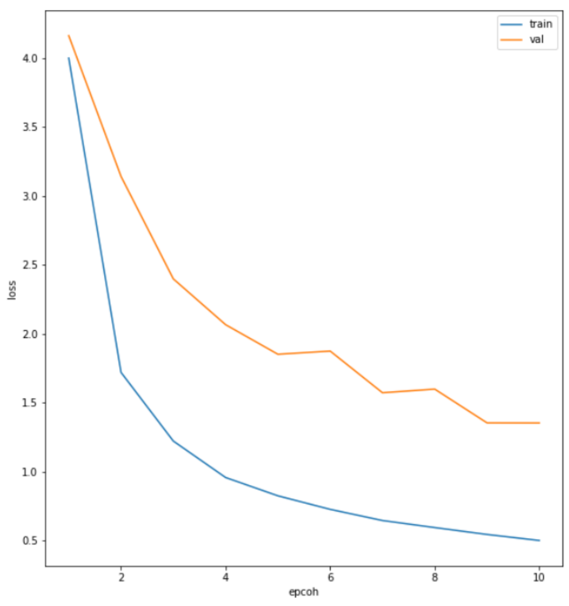
Fashion MNIST

Similar to MNIST, I first find best parameters using manual grid search. The best parameters are similar to MNIST. I have hidden size = 300/100, learning rate = 0.0005, lambda = 1e-5, dropout rate = 0.1, batch = 20, device = ‘gpu’.
For the base mlp, with the above parameters. I get a train accuracy = 0.8161 and test accuracy = 0.8045 with plot:



After adding normalization, the train accuracy is 0.8381 and test accuracy is 0.8146.





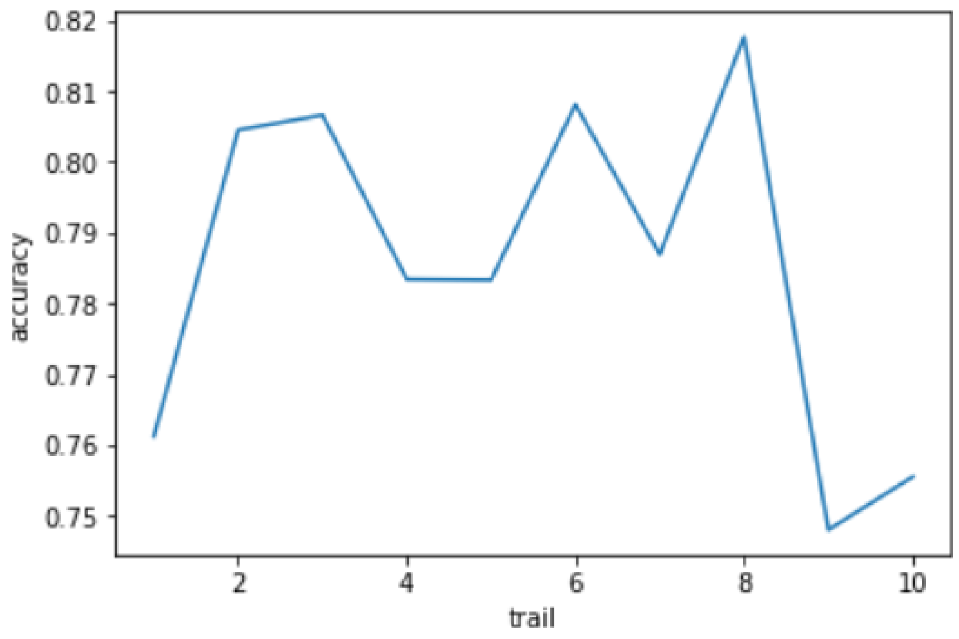
From the plot, we can see that there is an overfitting. It also means there is high variance and low bias.
Therefore, I try l1, l2 and dropout. The summary table is as below.

Hidden size	Learning rate	penalty	normalization	dropout	Train accuracy	Validation accuracy
300/100	0.0005	none	none	none	0.8161	0.8045
300/100	0.0005	none	yes	none	0.8233	0.8213
300/100	0.0005	L1:1e-5	yes	none	0.7752	0.7861
300/100	0.0005	L2:1e-5	yes	none	0.8030	0.8152
300/100	0.0005	none	yes	0.1	0.6091	0.6008
300/100	0.0005	L2:1e-5	yes	0.1	0.8016	0.8024

The bias-variance trade-off is similar to MNIST. L2 regularization has both relatively low variance and bias. We can see that l1 and dropout are not helpful. So I decide to use l2 and normalization to move forward.
After 10 trails of training, I get the following accuracies with mean 0.7855 and standard error 0.0229. Similar to MNIST, the mean testing accuracy is also lower than the validation accuracy.

```
average test accuracy:  0.7855
accuracy standard error:  0.0229
```

	trail	accuracy	loss
0	1.0	0.7612	0.279869
1	2.0	0.8045	0.123799
2	3.0	0.8066	0.152880
3	4.0	0.7834	0.219814
4	5.0	0.7833	0.237239
5	6.0	0.8081	0.116979
6	7.0	0.7869	0.126773
7	8.0	0.8177	0.121548
8	9.0	0.7480	0.488647
9	10.0	0.7555	0.485153



Summary

The mlp runs successfully and optimize the parameters in the back propagation. However, I think the performance of my mlp on both datasets are not good enough. Maybe due to small epoch, or due to incorrect parameters. It is difficult to do parameter tunning since it takes long time to run for each time. I also learned how to get the intuition for the starting parameters based on the complexity of the data. It is interesting to plot the training process and see the trade-off between variance and bias.