Yuan Hong (Bill) Sun
1003039838

MIE324 Assignment 2

2. Data Pre-processing and Visualization

2.2 Understanding the dataset

1. There are 11687 high income earners and 37155 low income earners in this data set (raw).

2. This dataset is not balanced. Since there are much more low-income earners than high income earners, if this data is used to train the model, the model may become much more proficient at recognizing patterns for low income earners. This will cause higher inaccuracy in the predictions compared to being trained on a more balanced data set.

2.3 Cleaning

1. After filtering out rows with empty variables, 3620 samples were thrown out, and 45222 samples remain. This is a reasonable number of samples as it accounts for about 7.5% of the number of samples in the raw data. This ensures that the model is not being trained out data that has missing values.

2.4 Balancing the dataset

1. Since there are 11208 high income earners and 34014 low income earners in the filtered data set, a new balanced data set was created using 11208 high income and low-income earners each, with the low-income samples being randomly selected using a seed of 0.

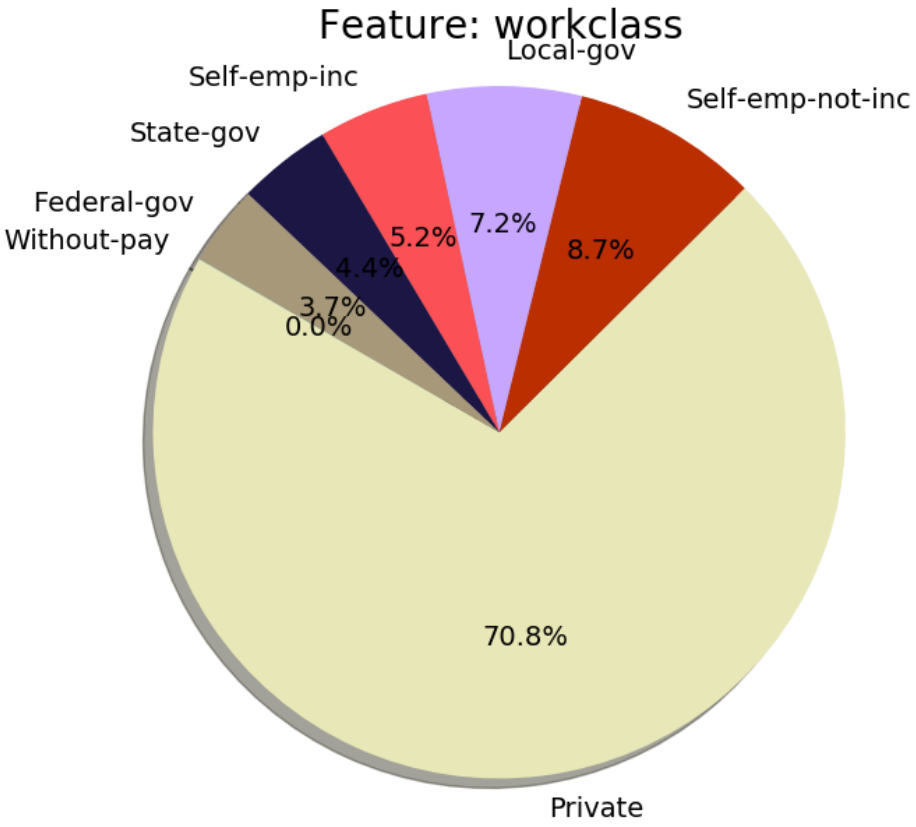
2.5 Visualization and understanding

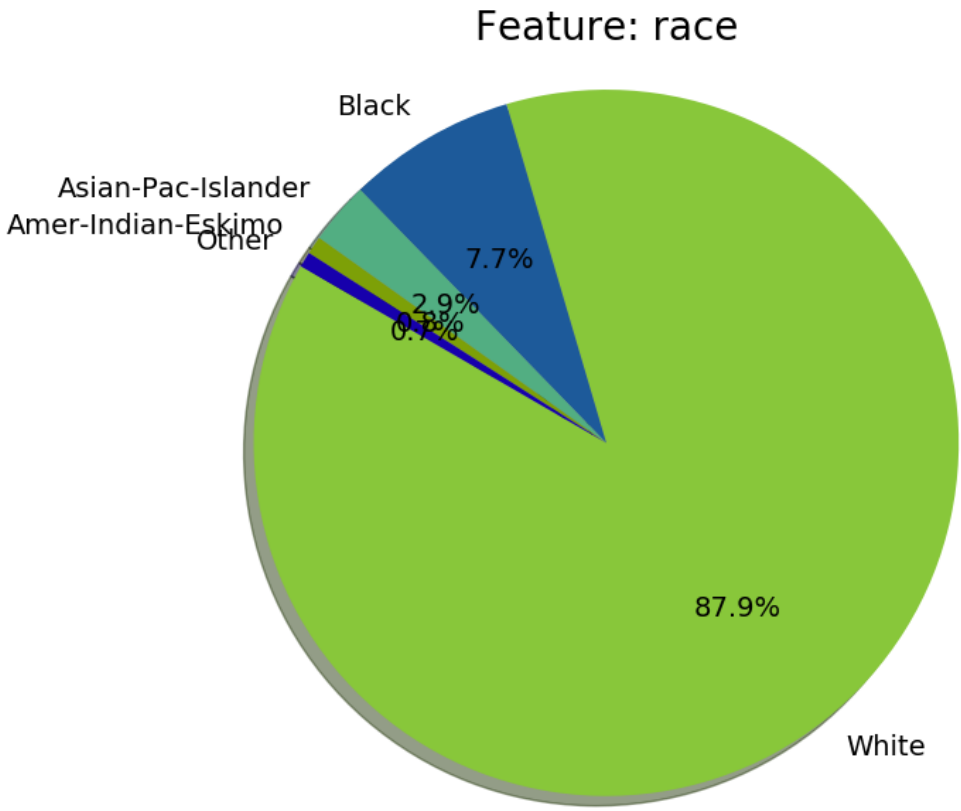
Numerical features:
1. The minimum age of all samples is 17, while the minimum hours worked per week is 1.

Categorical features:

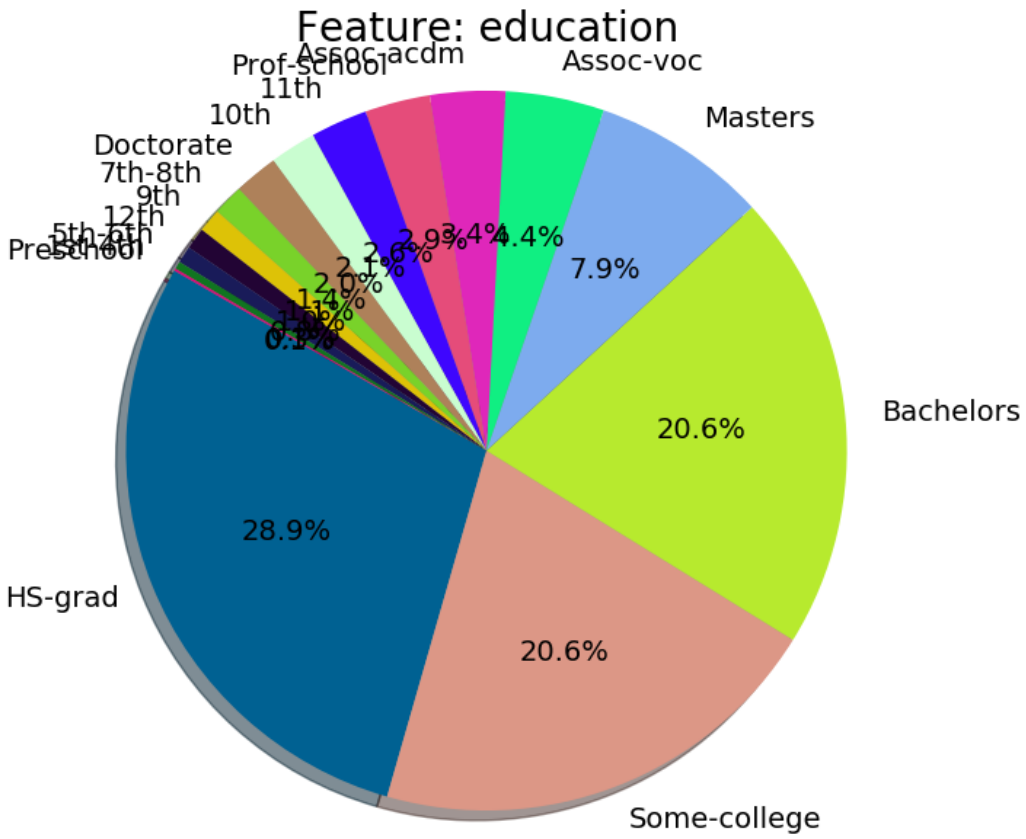
-Pie charts:

Yuan Hong (Bill) Sun
1003039838





Yuan Hong (Bill) Sun
1003039838

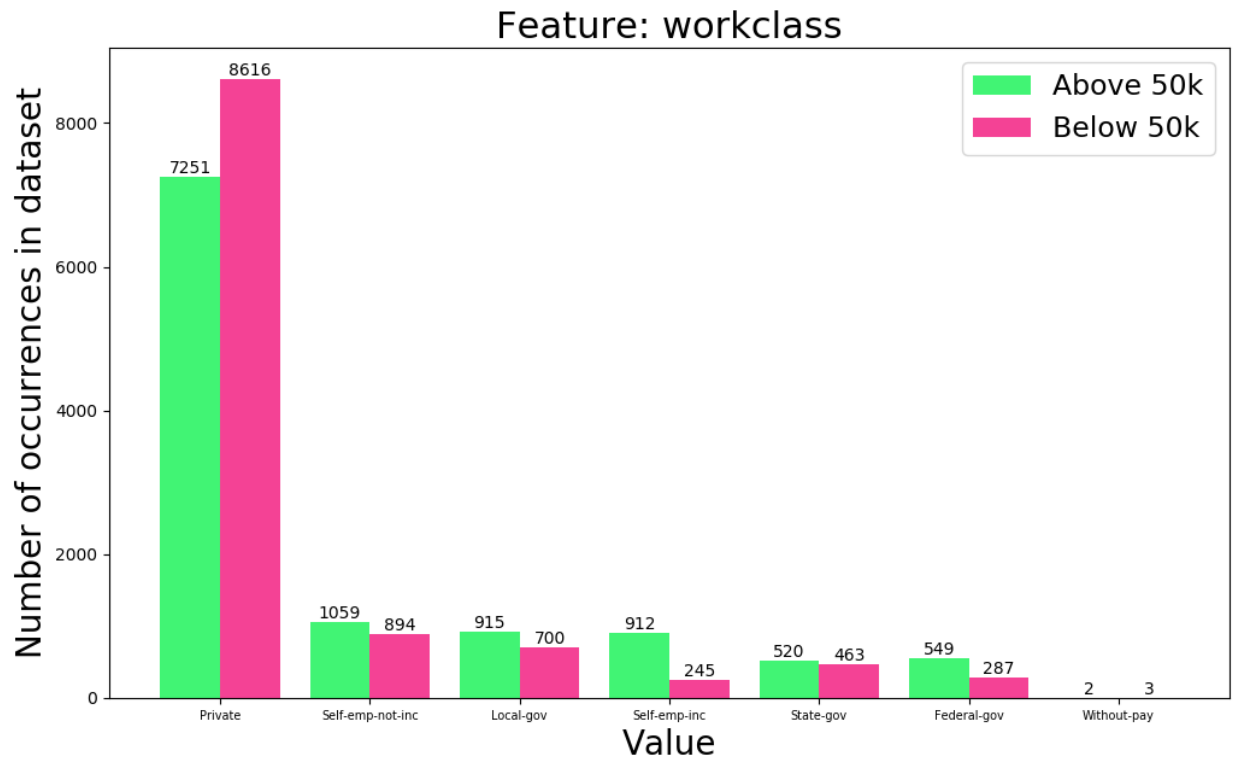


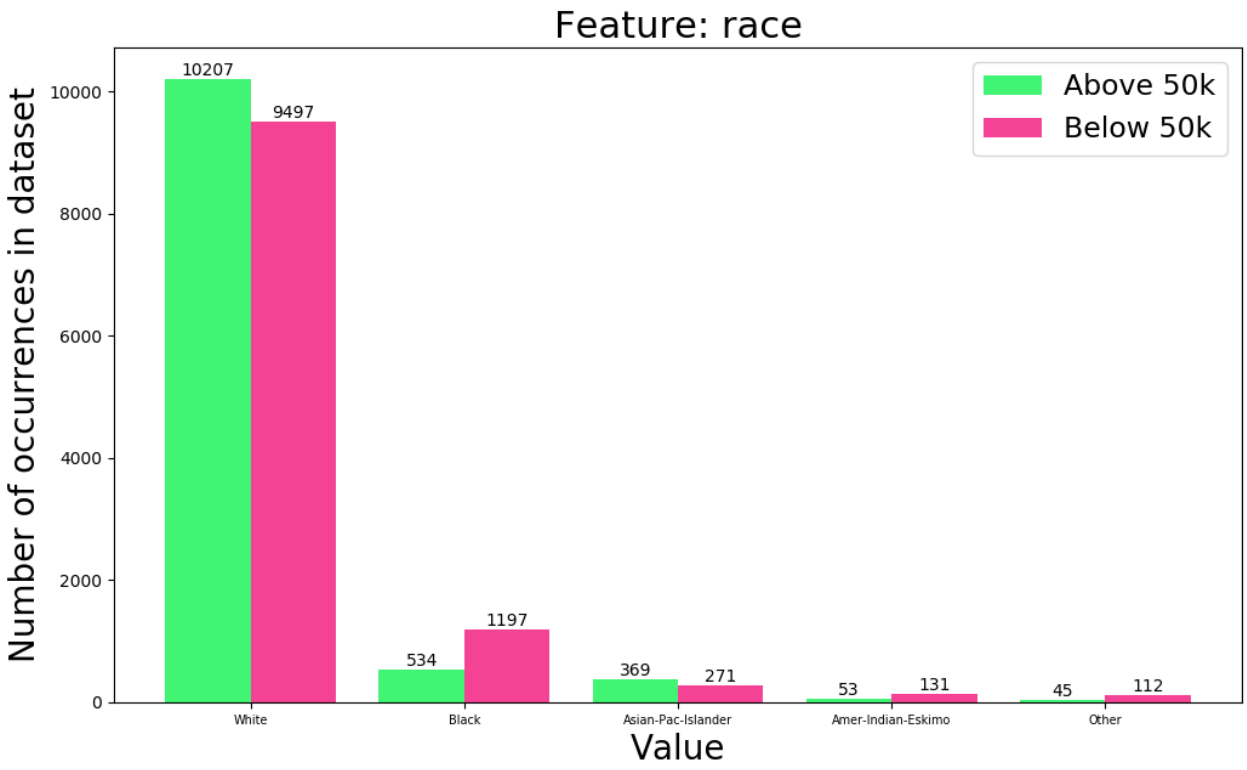
1. Many groups in different categories are under-represented. For example, samples who work in private and are white represent a majority of the data, while other groups split the rest of the samples. This means the model trained on this data may not generalize well for different groups, especially the minority ones.

2. One of the biases may be that in categories such as education, a lot of the options are meaningless, such as 9^{th}, 10^{th}, 11^{th}, and 12^{th}. The definitions of these groups are quite narrow in the sense that they only span one year each, whereas other groups span 4 years or more. This makes it hard to generalize patterns in these subgroups. What could have been done instead is to group them into a larger subgroup, called “middle school grad” or “some high school”.

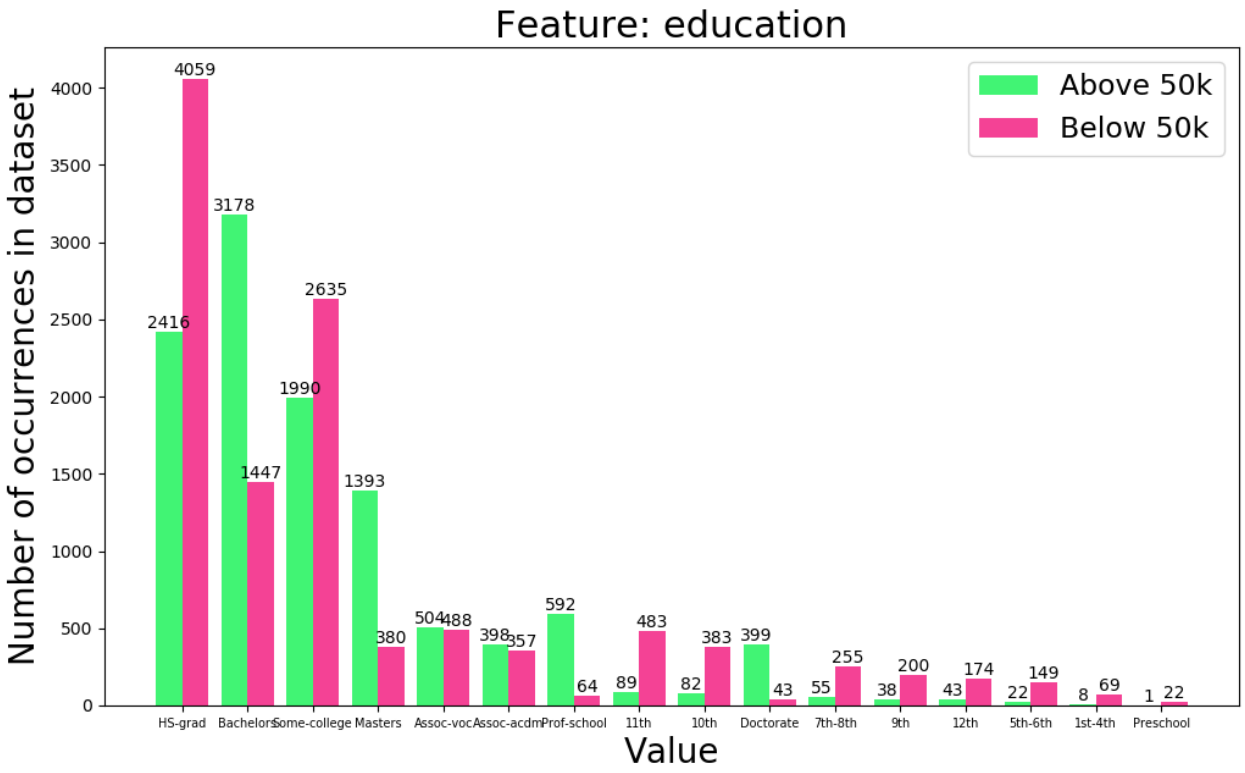
-Binary bar charts:

Yuan Hong (Bill) Sun
1003039838





Yuan Hong (Bill) Sun
1003039838



1. Out of work class, race, and education, education is the most obvious predictor in income, as higher educated (bachelors, masters, doctorate) samples are much more likely to have income over 50k, while lower educated are a lot more likely to have income below 50k. In addition, in work class, self-employed individuals are much more likely to make over 50k.

2. If someone has a high school education, there is a 37% likelihood that they earn above 50k. If someone has a bachelor's education level, and nothing else is known, it can be roughly assumed that they make above 50k (69% likelihood).

3. For any new sample, first check their education level; generally, those with a bachelor's degree or higher is a lot more likely to make 50k, and those with a lower education level are likely to make less than 50k. Then check if they are self-employed, since self-employed individuals are much more likely to make 50k.

2.6 Pre-processing

1. Using integers to represent categorical data makes it continuous/numerical, and artificially gives the data properties/information that it shouldn't have in its raw form or in one hot encoding.

2. Using unnormalized continuous data would result in unrealistic comparisons between numbers of varying magnitudes, instead of comparing each number to an average. This would return unexpected results due to the large variation in magnitude in the numbers.

3. Model Training

3.2 DataLoader

1. It is important to shuffle the training data so that the model is not always training on the same data every time it is started. If the training data is ordered in some way (e.g. by income), then the neural network is only training on data from one income category for a long time before training on the other category, which will cause it to only recognize one income category at first and not the other. The mixing of the data gives the model a more balanced training progression.

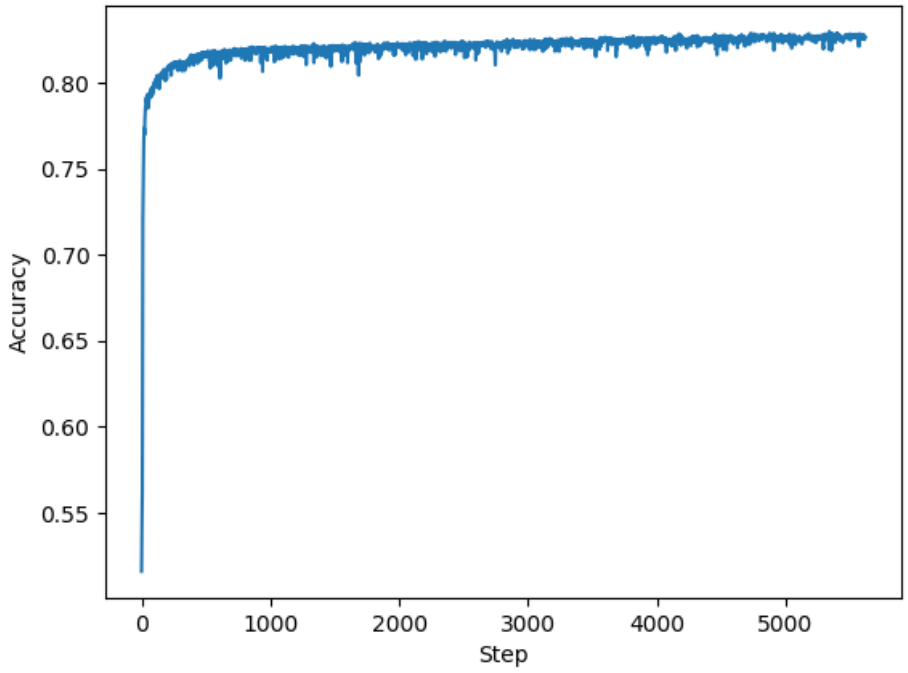
3.3 Model

1. The hidden layer should have a size between the input layer size (103) and the output layer size (1). It should be around the average size of the input and output layers. The size of output layer is 1 since there is only one output with two outcomes: 0 and 1 (less or greater than 50k income).

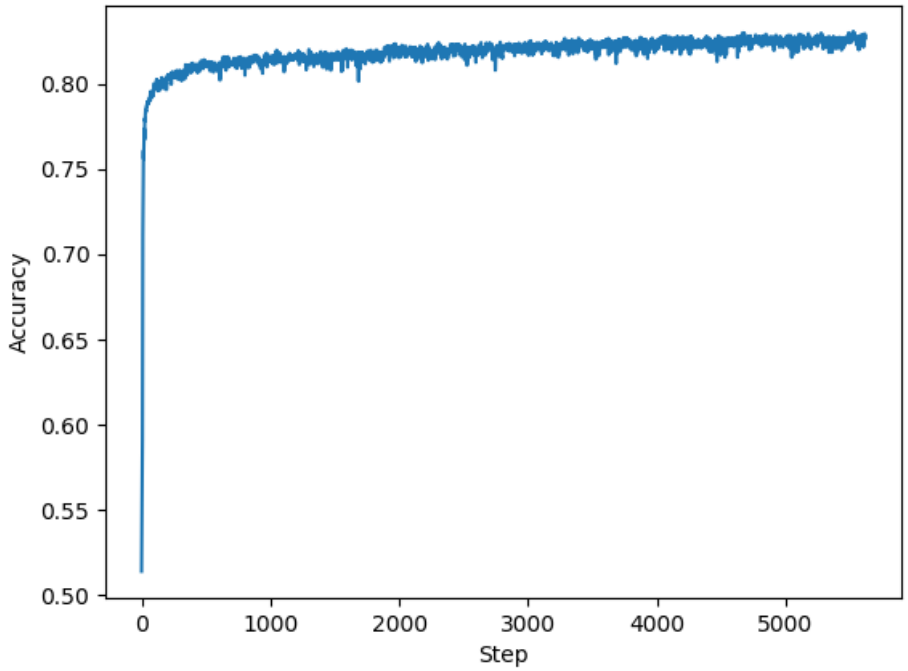
3.6 Validation

Batch size = 64, Learning rate = 0.1, MLP hidden size = 50, No. of epochs = 20

Training Accuracy:



Validation accuracy:



The validation accuracy is between 82 and 83%.

4. Hyperparameters

4.1 Learning rate

Yuan Hong (Bill) Sun
1003039838

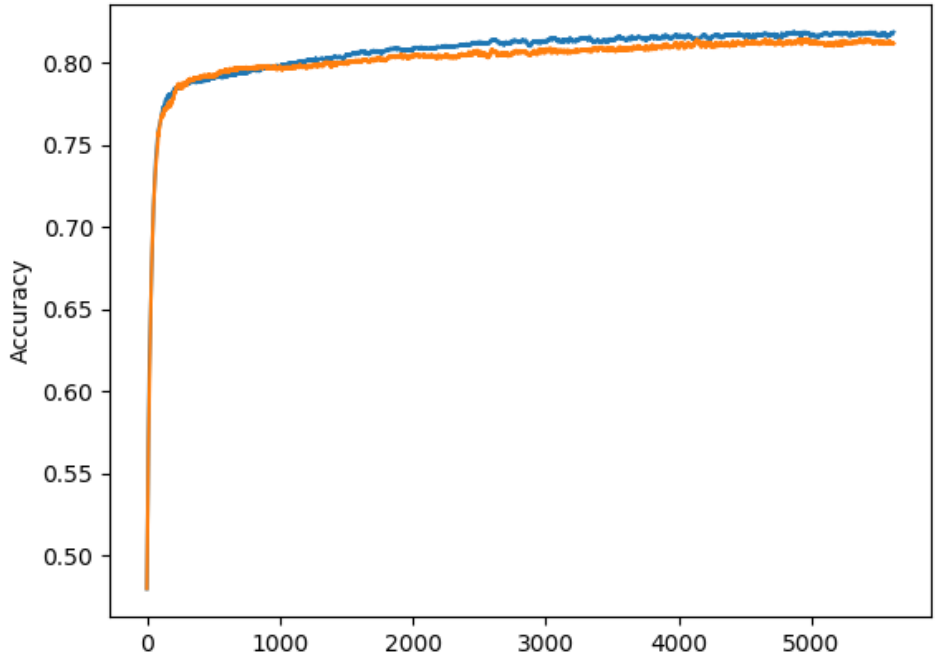
Accuracy table:

Learning Rate	Highest Validation Accuracy
0.001	0.7958 (20 epochs)
0.01	0.8143 (20 epochs)
0.1	0.8234 (10 epochs)
1	0.8267 (5 epochs)
10	0.8047 (5 epochs)
100	0.7705 (5 epochs)
1000	0.4967 (5 epochs)

Note: Even though a learning rate of 1 produced a better peak validation accuracy, a learning rate of 0.1 offers more consistency (by comparing the graphs)

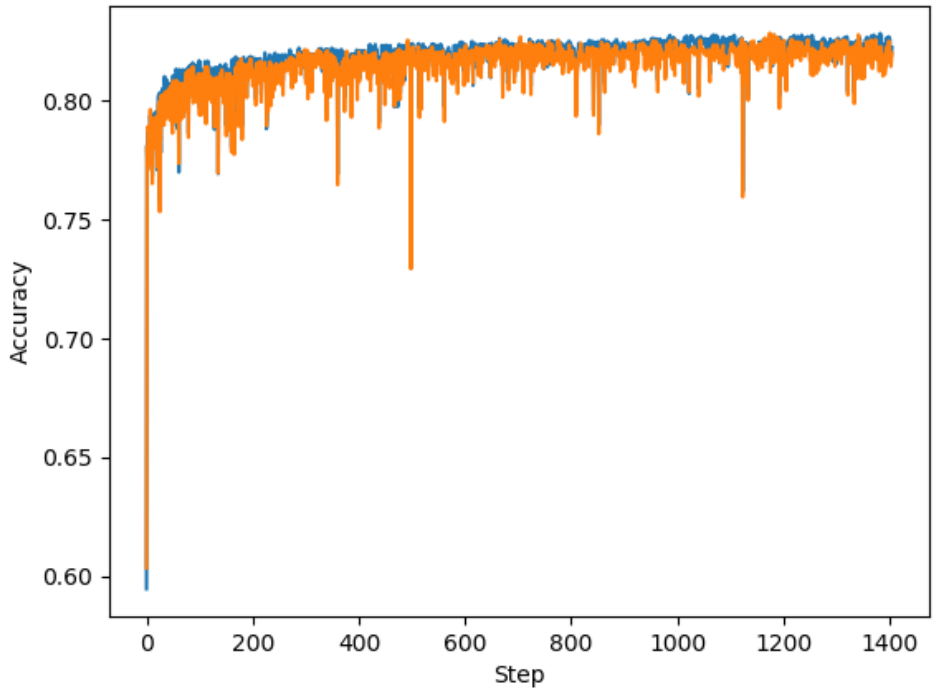
Accuracy plots: (Blue—training, orange—validation), Bs = 64

Lr = 0.01

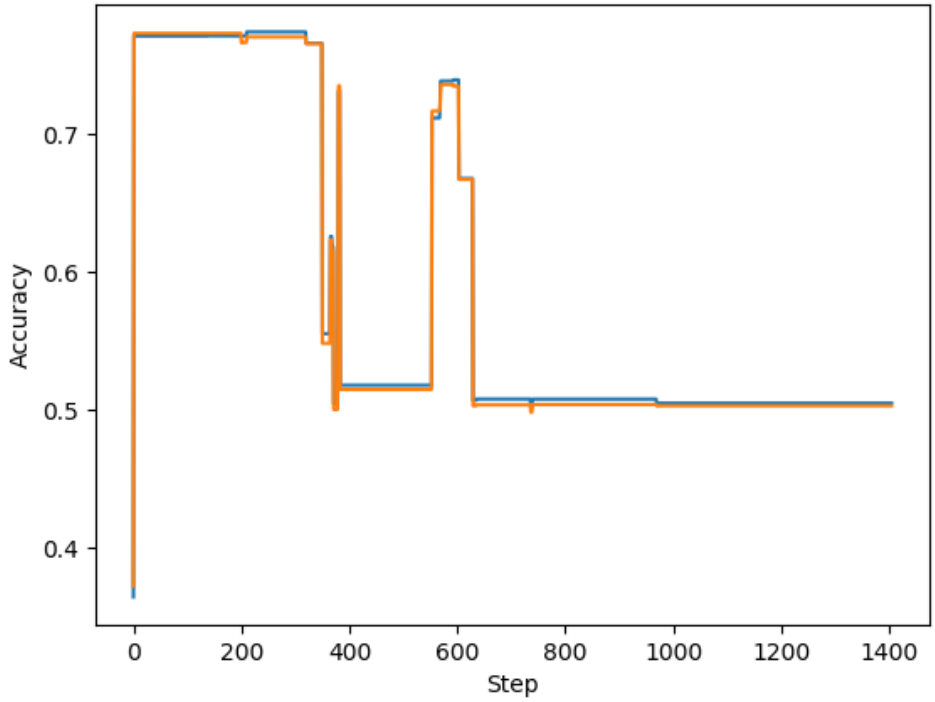


Lr = 1

Yuan Hong (Bill) Sun
1003039838



Lr = 100



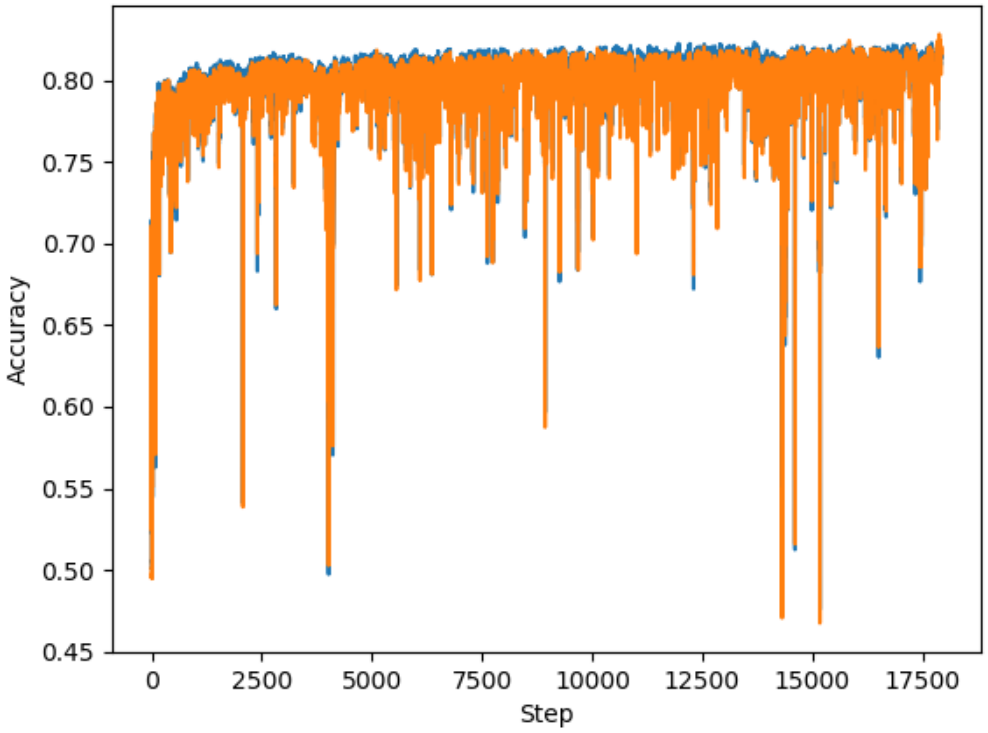
1. A learning rate of 0.1 worked the best. Even though a learning rate of 1 produced a slightly better highest validation accuracy, the results are more consistent when the learning rate is lower (as shown by the graphs where there is less variation in the validation error).

2. At a low learning rate, the learning progress of the model is slow, so it takes more epochs to train the model to reach a good result. However, at a high learning rate, the model overcompensates each result, causing the validation and training accuracy to fluctuate greatly, and leading to progressive worse results as the learning becomes chaotic.

4.3 Batch size

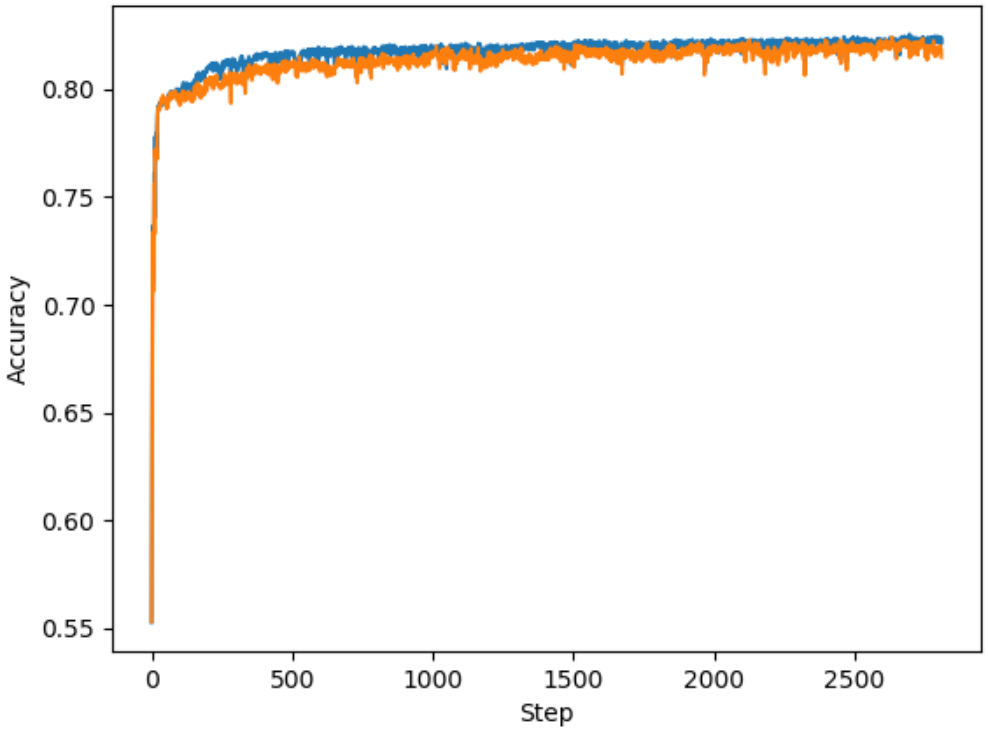
Accuracy over step plots: Lr = 0.1

Bs = 1

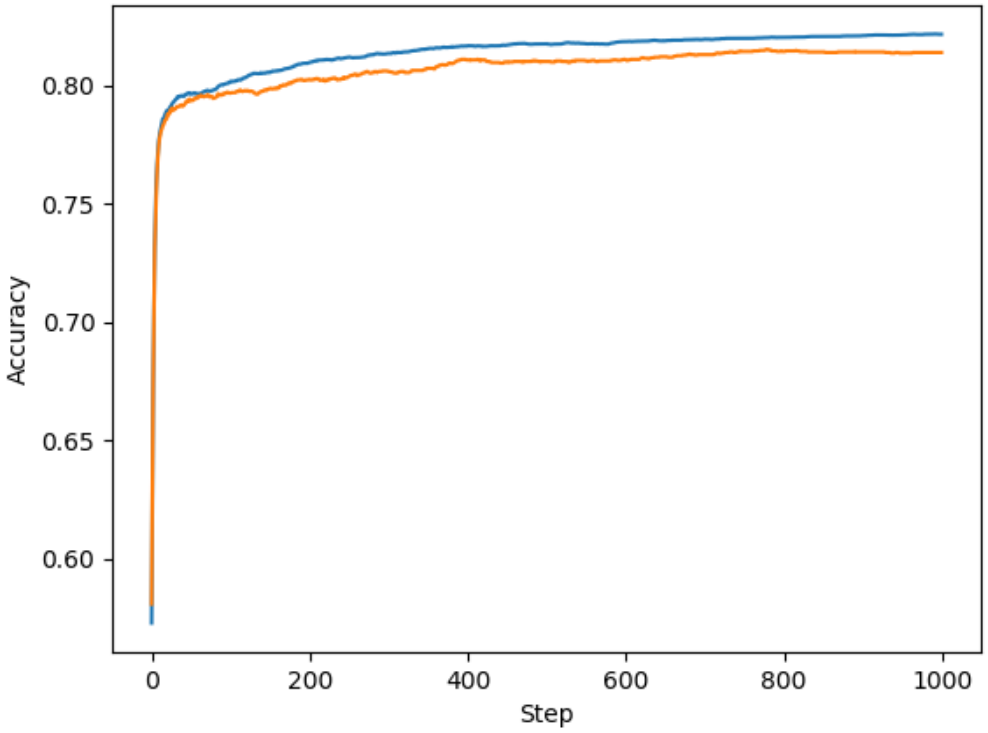


Bs = 64

Yuan Hong (Bill) Sun
1003039838



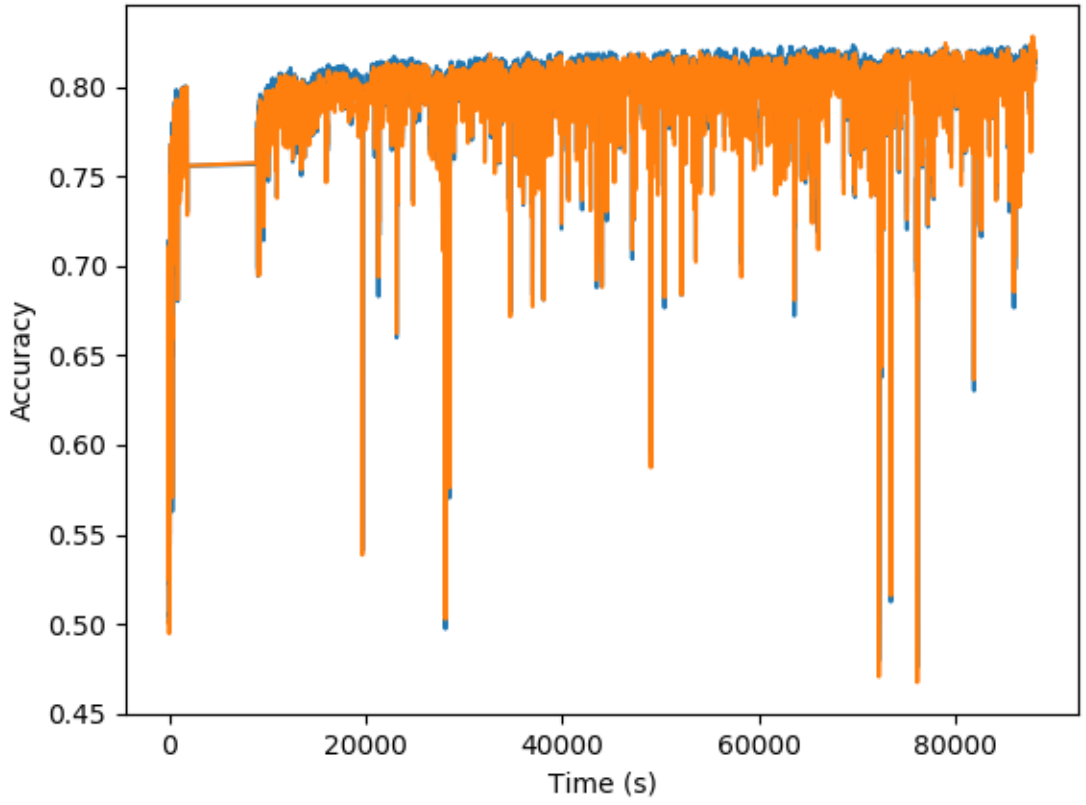
Bs = 17932



Accuracy over time plots:

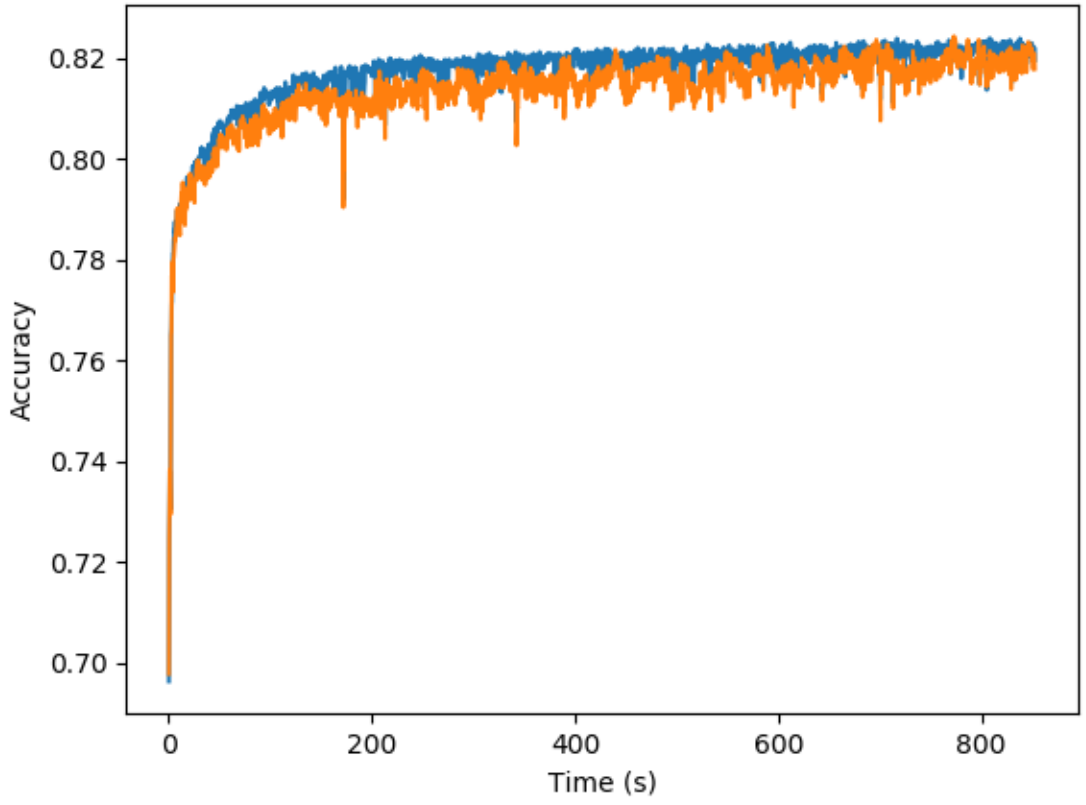
Bs = 1

Yuan Hong (Bill) Sun
1003039838



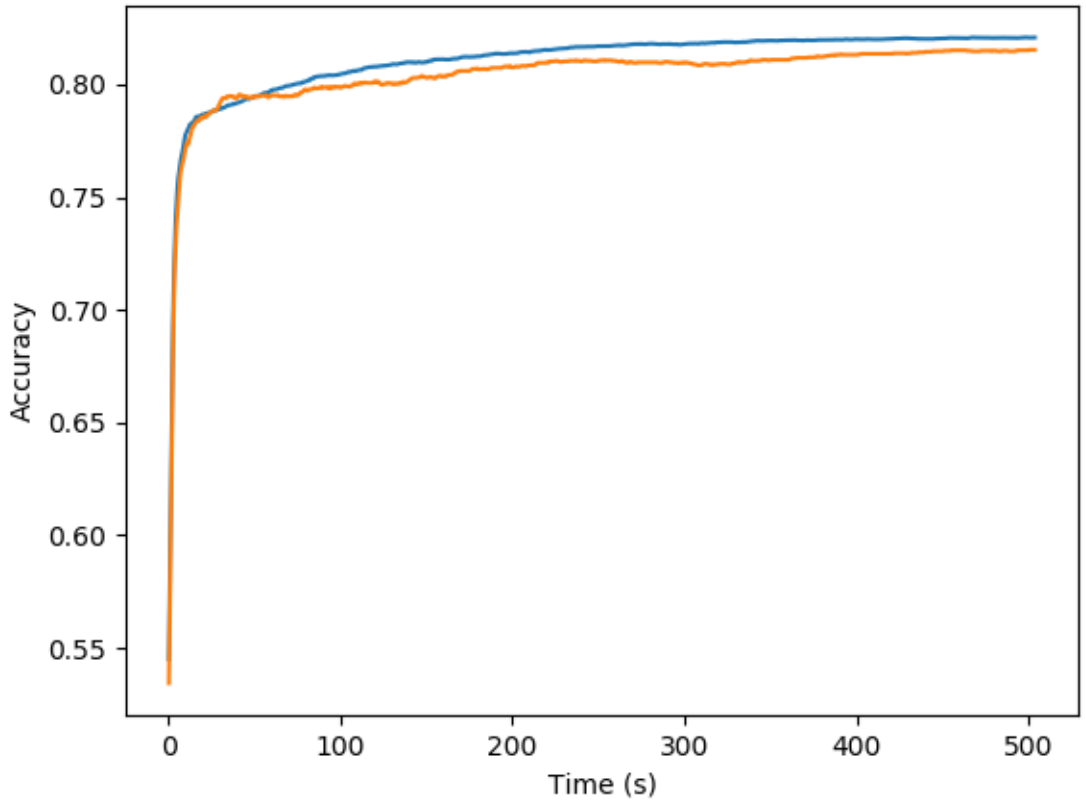
(Ignore the break when the computer was temporarily dormant)

Bs = 64



Yuan Hong (Bill) Sun
1003039838

Bs = 17932



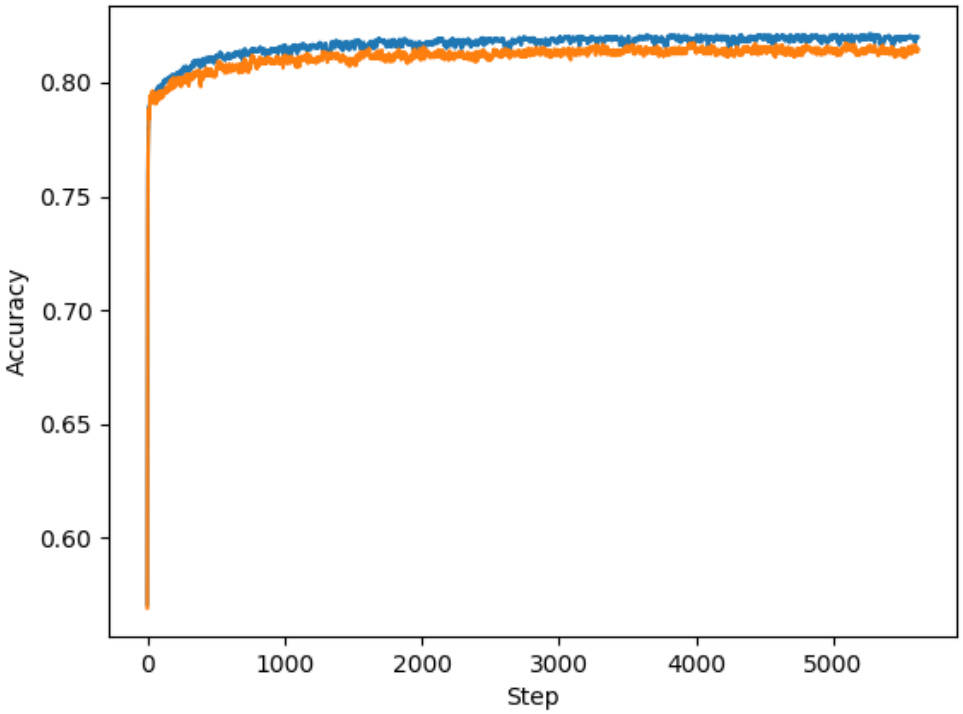
1. A batch size of 64 produced the highest validation accuracy, flattening out at around 82%.

2. A batch size of 17932 was able to reach a good validation accuracy in the shortest time, as well as within the smallest number of steps.

3. If the batch size is too low, it takes far too long to train since each individual image must be iterated through in one epoch. If the batch size is too high, the model may not be as effective in learning compared to a medium batch size since in every epoch, the model is being trained on the same data over and over, thus not producing any real learning, but potentially overfitting on this batch of data.

4. Using a small batch size is beneficial for getting the model to train on different data, which helps the model generalize patterns instead of overfitting. However, it takes too long to train, thus making it impractical. On a large batch size, the model is quick to train and reach good results in shorter time, but it is more prone to overfitting as the model trains on the same data over and over.

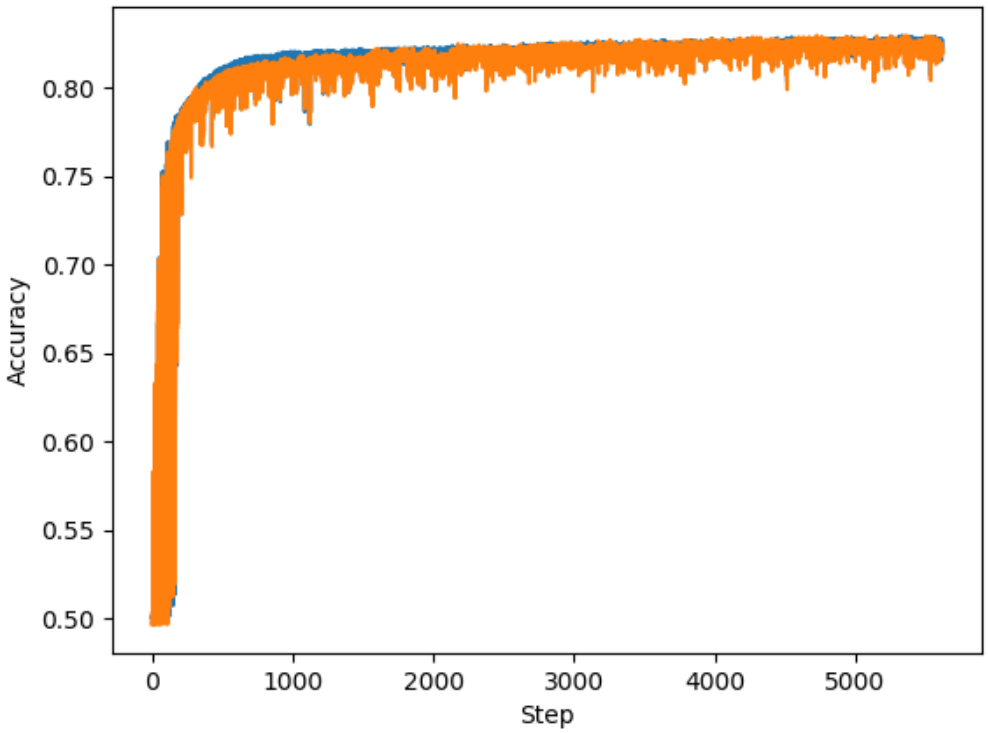
4.4 Under-fitting



Lr = 0.1, Bs = 64

1. The small model achieves a similar validation accuracy compared to the best model so far, at around 82%. This model does not appear to underfit as both the training and validation accuracy levelled out at a relatively good result (~80%) and did not appear to worsen after a peak accuracy.

4.5 Over-fitting

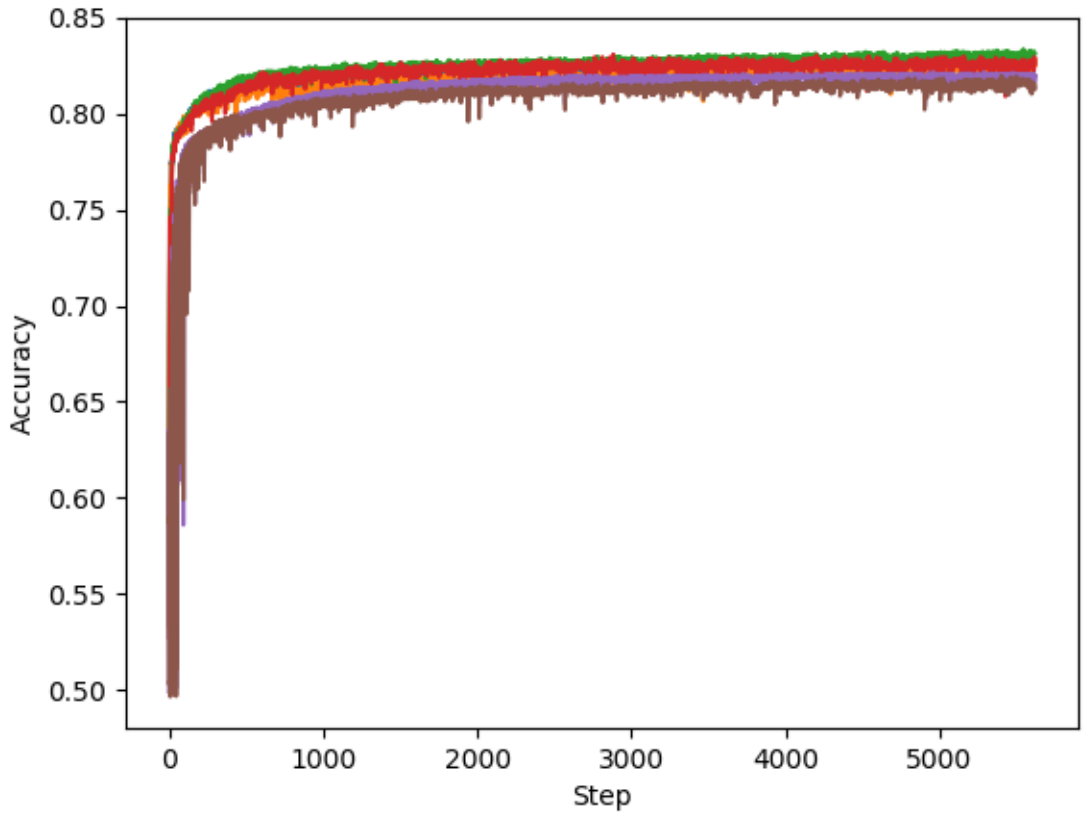


Lr = 0.1, Bs = 64

1. In terms of accuracy, the larger model is not much different from the smaller model and the standard model. The validation accuracy flattens out to around 82%, although there is a greater variation (as shown by the wider range of the orange line). The model does not appear to be overfitting since the training and validation accuracy flattened out at a good (and the same) level instead of becoming increasingly worse or separated.

4.6 Activation function

Activation Function	**Tanh**	**Relu**	**Sigmoid**
Time Taken (for 20 epochs)	2261.78 s	2060.68 s	2151.69 s



(Colours: Blue and orange: Tanh; Green and red: Relu; Brown and purple: Sigmoid)

1. By visual inspection, the Relu activation function returned the highest validation accuracy, at about 82.5% on average after reaching the maximum. Sigmoid returned the lowest validation accuracy, on average, at about 81.5%.

2. All three activation functions took a similar amount of time to run, although Relu took slightly less, and tanh slightly more.

5. Hyperparameter Search

Random seed used: 42

Although different approaches were used, the best model configurations were still somewhat uncertain since different hyperparameters, hidden layers, and activation functions returned relatively similar results (validation accuracy flattening out between 81 and 83%, peaking over 83%). However, by visually inspecting the data for consistency (amount of variance once the validation accuracy line is flattened out), it appears that using the best hyperparameters (Bs =

64, Lr = 0.1, activation function = Relu, although a slightly smaller learning rate is preferred for more consistency), along with no hidden layers, produced the best results consistency-wise.

6. Feedback

1. The assignment, in total, took more than 24 hours. However, most of this time was spent training (particularly for batch size = 1, which took one night and half a day to complete one run of one epoch). The coding and written responses took no more than 8 hours.

2. The most challenging part was trying to understand the model behavior, and why similar results were being produced with changes to the model. The programming part was relatively straightforward (given the in-class samples).

3. It was fun coding up everything in Python as I was already familiar with the implementations. It was also enjoyable watching the model run and getting the results.

4. The most confusing part, as stated above, was trying to understand why changing some model characteristics (such as the activation function and the number of hidden layers) makes little or no changes despite the expected results. Some of the instructions were also confusing, such as the one that asks to “plot out the last N steps” of the training accuracy, which had to be revised later. Also, in 4.5 when more hidden layers were needed, the activation function was not specified, so I had to use tanh and sigmoid once each to balance out.

5. This assignment is very helpful in helping to learn how to code up the entire pipeline of machine learning, from the preprocessing to a multilayer neural network, and training and validation loops.