General description of my subtask1 code

My code for each question was very much so based off of the given solution to question 1, looping through the subsets of the relevant parameters and producing and evaluating models using all combinations of those subsets. However, in the interest of saving time when making changes to the formatting of the output graphs, I altered my code so that for each question there are three code blocks instead of one; they are described as follows:

The first code block for each question uses embedded for loops to produce all combinations of the relevant parameters, produce the associated models, evaluate them, and aggregate their accuracies into a list (q1ACCs, q2ACCs, q3ACCs, q4ACCs, and q5ACCs respectively), and then save the list to a .csv.

The second code block for each question is there to load previously saved qxACCs.csv files into their respective lists. This was done to save time in case main memory had been cleared but we didn't want to have to create and evaluate each model again.

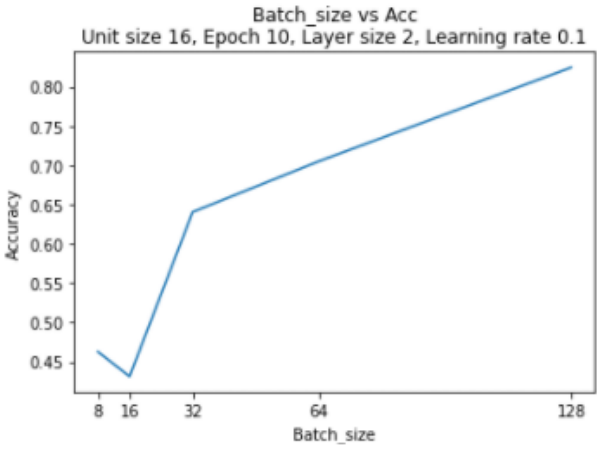
The third code block for each question uses the same outer for loops as the first code block, but omits the innermost for loop, effectively cycling through the combinations of the relevant parameters and that combination's corresponding accuracies so as to produce the representative graphs (Unit_size vs accuracy, Batch_size vs accuracy, Learning_rate vs accuracy).

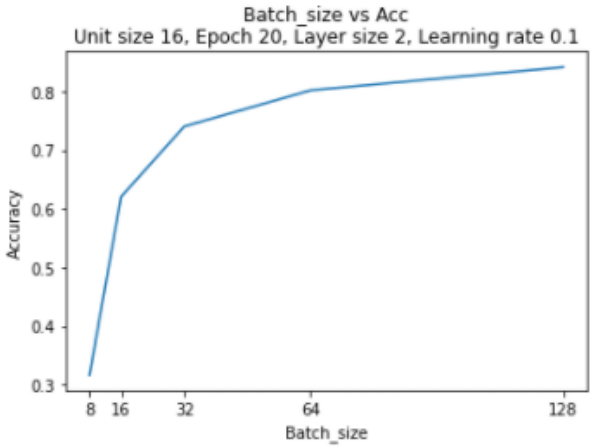
This enabled me to just only rerun the second and/or third codeblocks whenever I wanted to make changes to the formatting of the graph, rather than having to rerun the first code block which produces, evaluates, and saves the accuracies of each model, thereby saving time.

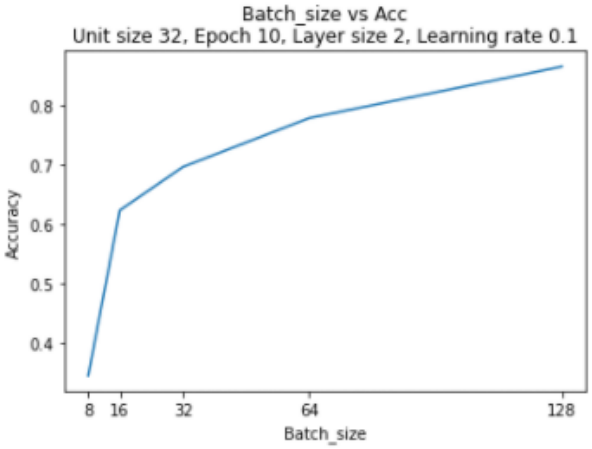
One additional change that was made was in questions 2 and 4, where `plt.xscale('log',basex=2)` and `plt.xscale('log)` were used, respectively, for the sake of a better visualization, as the default linear scale made it difficult to differentiate between models in each graph.

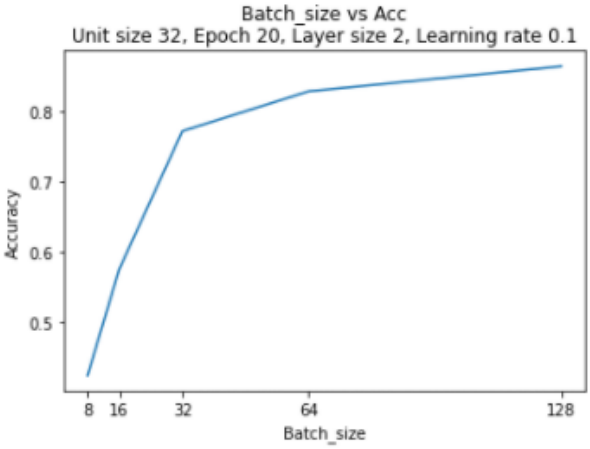
With the increase of the batch size, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

It was noted from question 3 that increasing the batch size, generally, tended to improve accuracy, albeit with a diminishing return as batch size continued to increase (logarithmic improvement?).



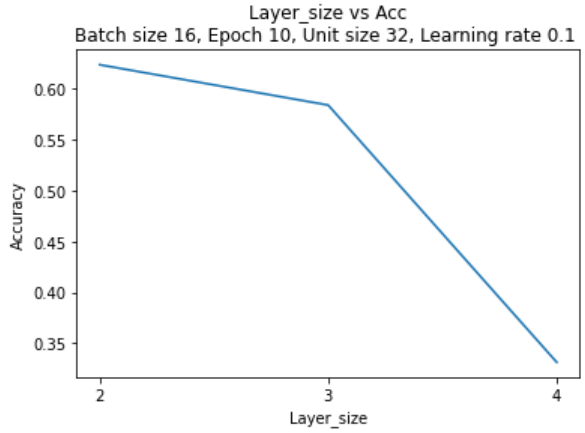


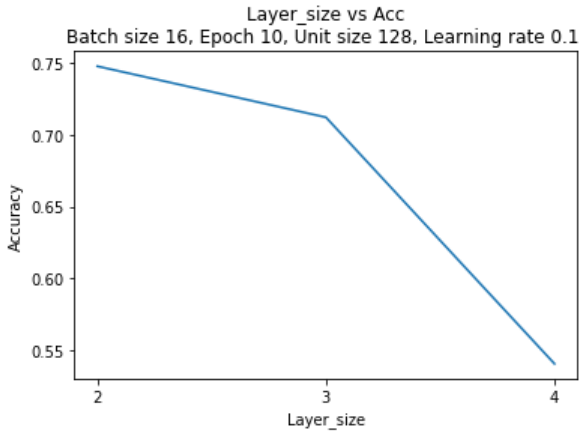


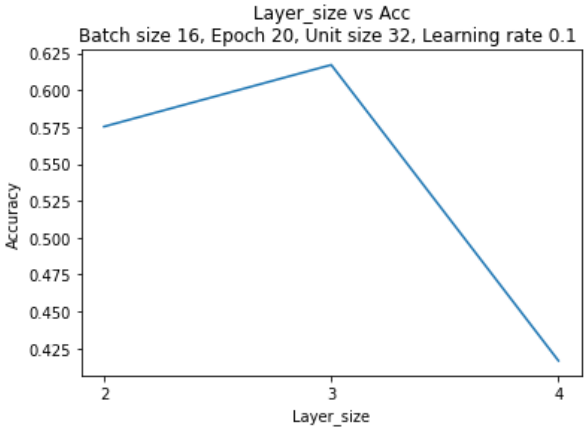


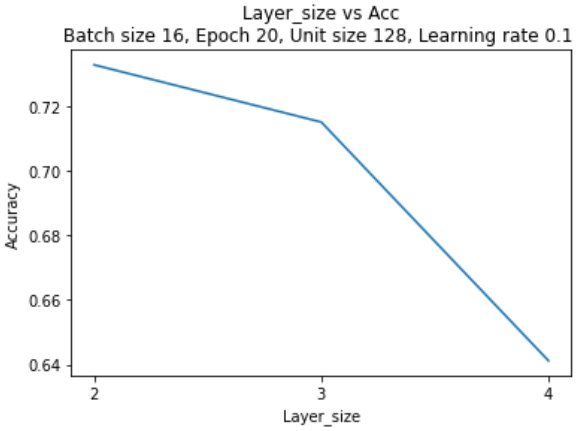
With the increase of the layer size, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

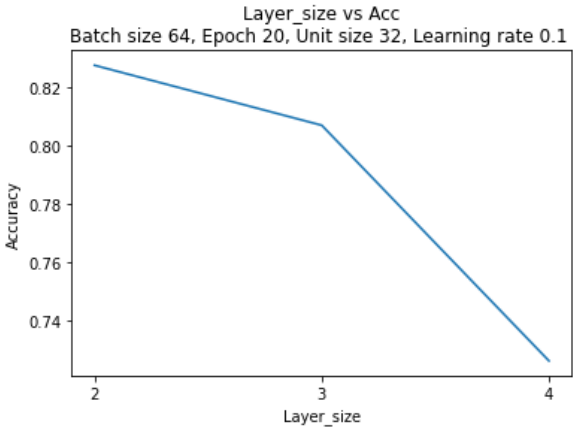
It was noted from question 1 that increasing the layer size, generally, tended to deteriorate accuracy. That said, with a batch size of 64, 10 epochs, a unit size of 128 and a learning rate of 0.1, there was actually a marginal improvement in accuracy between the models with a layer size of 3 and a layer size of 4 - albeit the accuracies of both were still much less than the accuracy of a similar model with only 2 layers.

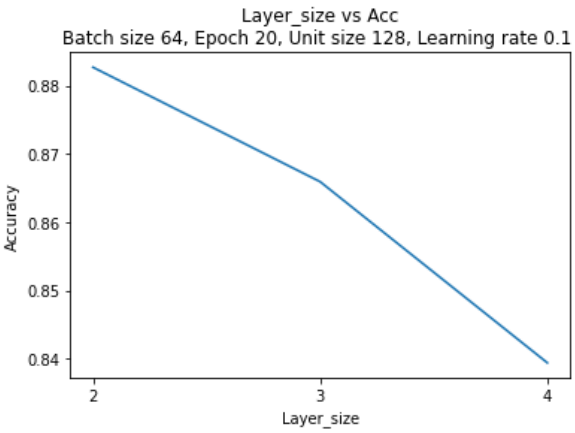


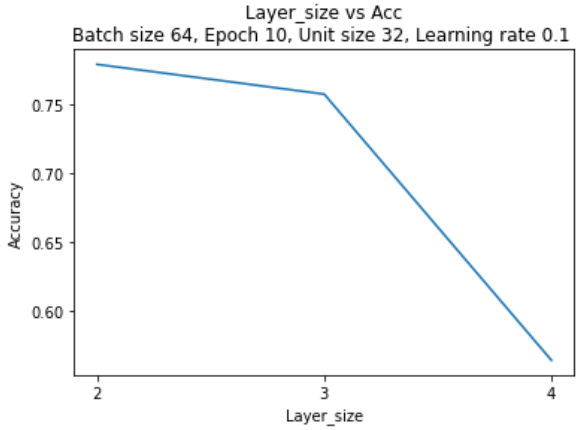


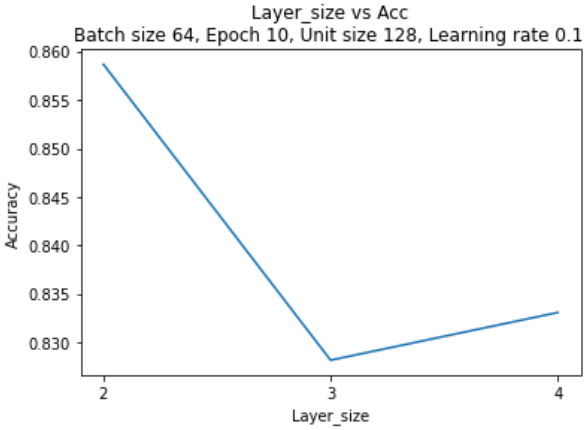





Layer_size vs Acc
Batch size 64, Epoch 20, Unit size 32, Learning rate 0.1
Accuracy
0.82
0.80
0.78
0.76
0.74
2
3
4
Layer_size


Layer_size vs Acc
Batch size 64, Epoch 20, Unit size 128, Learning rate 0.1
Accuracy
0.88
0.87
0.86
0.85
0.84
2
3
4
Layer_size


Layer_size vs Acc
Batch size 64, Epoch 10, Unit size 32, Learning rate 0.1
Accuracy
0.75
0.70
0.65
0.60
2
3
4
Layer_size

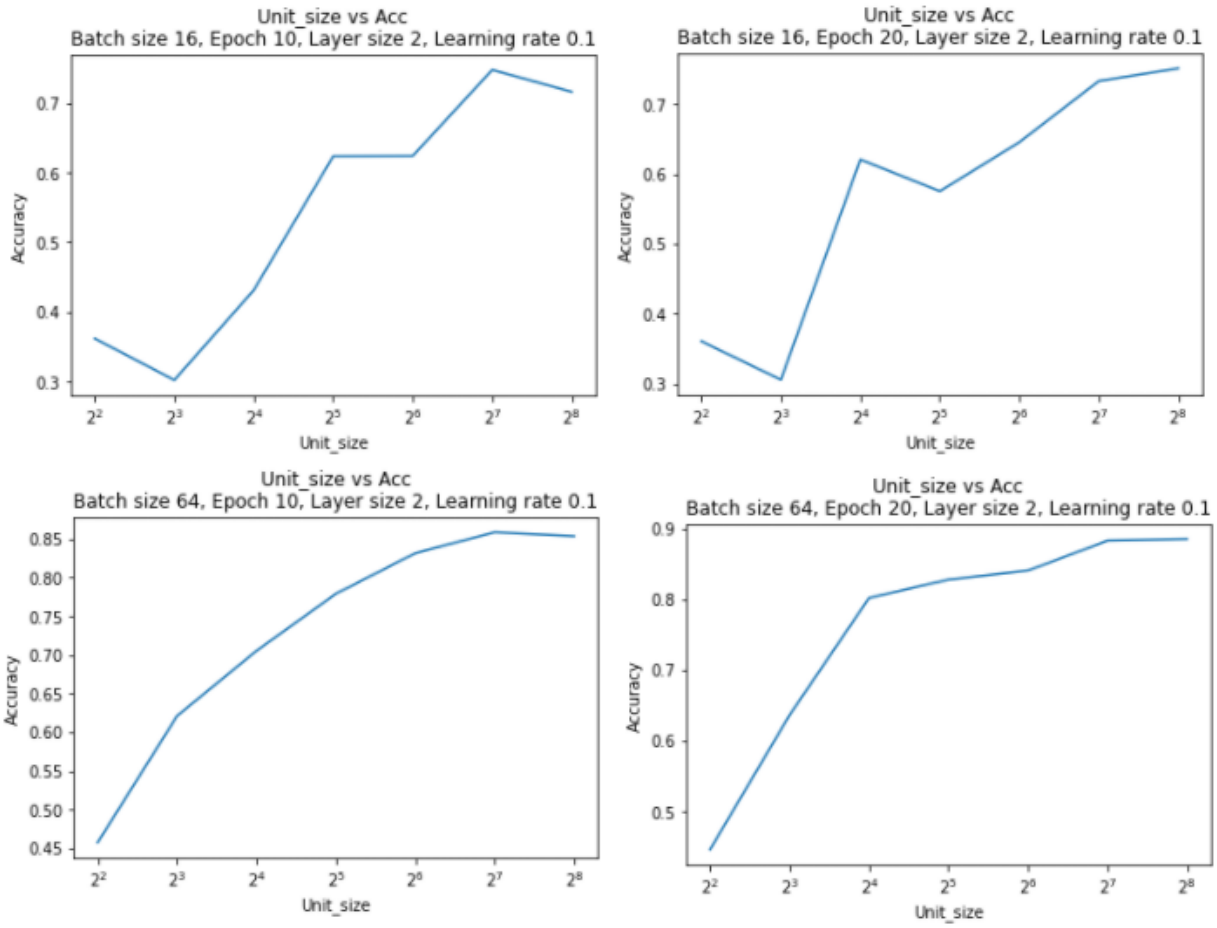

Layer_size vs Acc
Batch size 64, Epoch 10, Unit size 128, Learning rate 0.1
Accuracy
0.860
0.855
0.850
0.845
0.840
0.835
0.830
2
3
4
Layer_size

With the increase of the unit size per layer, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

It was noted from question 2 that increasing the unit size, generally, tended to improve accuracy, albeit with a diminishing return as unit size continued to increase (logarithmic improvement?). While increasing the unit size generally tended to improve accuracy, in both models with batch sizes of 16 there were exceptions to this generality.

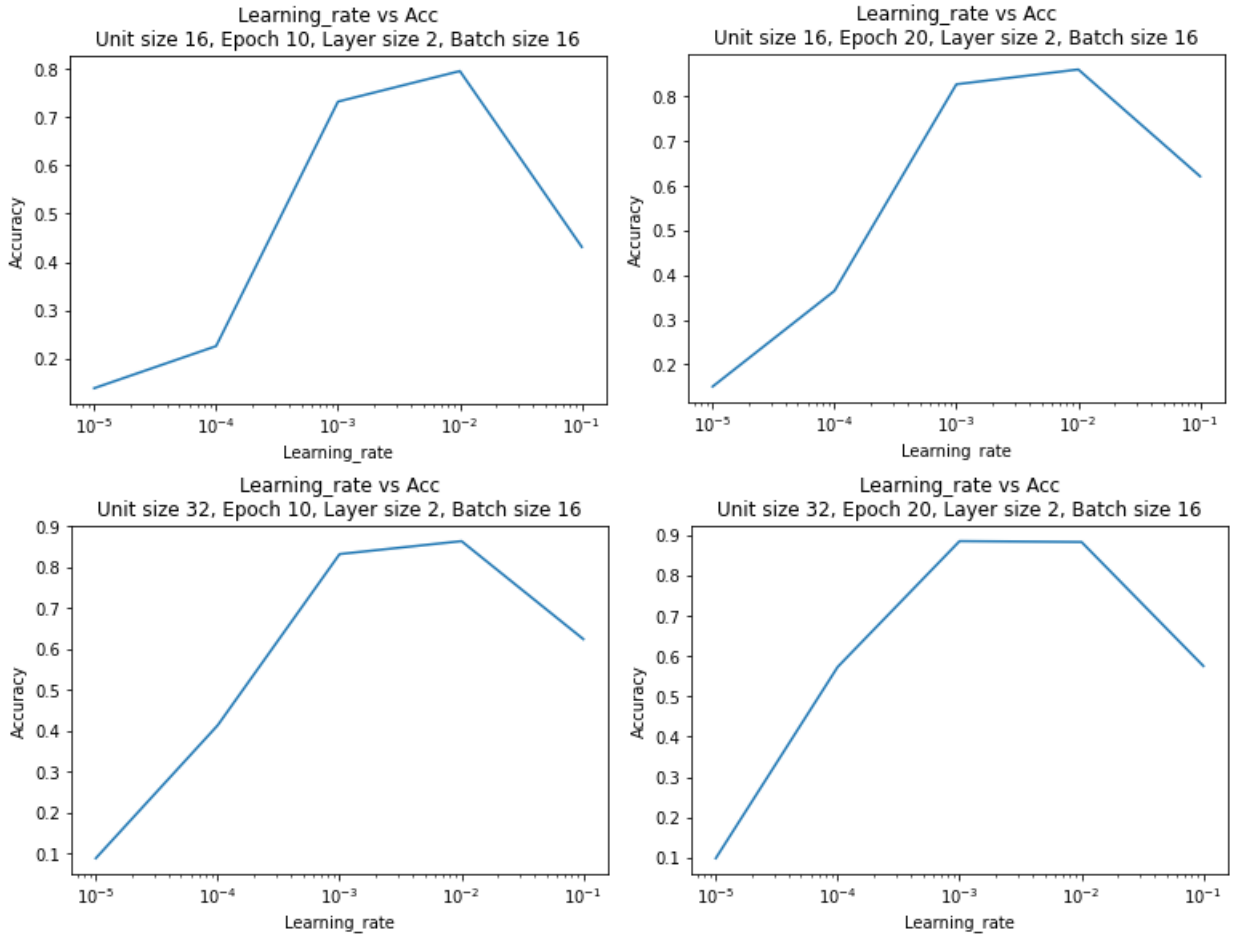
In the models with a batch size of 16 which ran for 10 epochs there was a deterioration in accuracy as unit size was increased from 2^2 to 2^3, and again when it was increased from 2^7 to 2^8, with no change in accuracy as the unit size was increased from 2^5 to 2^6.

In the models with a batch size of 16 which ran for 20 epochs there was a deterioration in accuracy as unit size was increased from 2^2 to 2^3, and again when it was increased from 2^5 to 2^6.



With the decrease of the learning rate, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

It was noted from question 4 that as the learning rate decreased, at first there was improvement in accuracy, but decreases from learning rates of 10^{-3} onward (decreasing by factors of 1/10) actually substantially deteriorated the accuracy of each model, as shown below.



With the increase of the epoch number, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

The effect of changing the epoch number was highly dependent on the batch size.

For a batch size of 8 the greatest improvement in accuracy occurred by increasing the epoch number from 10 to 20, but increasing from 20 to 30 actually deteriorated accuracy, with marginal improvements in accuracy resulting from increasing the epoch number thereafter.

For a batch size of 16, a deterioration in accuracy occurred when increasing the epoch number from 10 to 20, with quick improvements to accuracy occurring by increasing from 20 to 30, and from 30 to 40. Increasing from 40 to 100 also increased accuracy, albeit at a slower rate than the 20 to 30 and 30 to 40 changes.