ENSC 413
Assignment #3
3-Layer Network for MNIST

Cyrus WaChong
301306459

Spring 2020

Introduction:

This lab's goal is to find the effects that come from changing values such as learning rate, the number of epochs and the optimizer chosen on the accuracy and execution time. The code written in the code presented in class was used, the file was called "train_keras_MNIST.py". The same code has been used for this lab, but additional lines have been written for loading in data, as well as some changed code for values of parameters and optimizers chosen. These tests were done with tensorflow-gpu.

Results:

Adam optimizer:

The first step was to test the results when learning rate was changed but using the Adam optimizer with a batch size of 32 trained for 5 epochs. The results can be seen in table 1.

Learning Rate	Test Accuracy (%)	Execution time (s)
0.1	0.1871	11.067
0.01	0.9777	11.016
0.001	0.9539	10.633
0.0001	0.8879	10.495

Table 1: Results for net trained with Adam optimizer with batch size of 32 for 5 epochs

Table 1 shows that learning rate heavily affects accuracy; with an excessively large learning rate such as 0.1, the accuracy cannot improve because the network has too large of a step size and overshoots the minimum error when it is training itself. We see that a learning rate of 0.01 works best in this scenario, but that is only because the epochs are limited to 5. This is the same reason that the lower values of learning rate cause a lower accuracy, because we do not supply enough passes through the dataset for the network to train itself adequately.

To confirm this conclusion, the learning rate of 0.0001 was tested but with 20 epochs; The resulting network had an accuracy of over 99%, proving that this conclusion is correct.

The execution time does not very much because the learning rate does not affect the number of calculations done, just the step size in which we move down the error gradient. In table 2 the values for a fixed learning rate and number of epochs while varying batch size are plotted.

Batch Size	Test Accuracy (%)	Execution Time (s)
1	0.9945	96.651
4	0.9836	30.128
16	0.9702	13.754
64	0.9453	9.959

Table 2: Results for net trained with Adam optimizer with a learning rate of 0.001 for 5 epochs

As expected, the run times of each of these cases vary drastically. Unlike a varying learning rate, a varying batch size directly affects the execution time because it affects the amount of training examples utilised in one iteration: the larger the batch size, the less calculations done per epoch. Another observation that can be made is that batch size has minimal effect on the test accuracy, a maximal difference of less than 5% accuracy, while cutting the training time by nearly 10-fold. This difference of accuracy can usually be remedied by training on more epochs.

Below in table 3 are the results for a fixed learning rate and batch size while varying number of epochs is plotted.

# Epochs	Test Accuracy (%)	Execution Time (s)
5	0.9569	10.873
10	0.9811	14.645
15	0.9938	18.093
20	0.9991	21.154

Table 3: Results for net trained with Adam optimizer with a learning rate of 0.001 and a batch size of 32

Again, by changing the number of epochs, we expect to directly affect the execution time. Table 3 shows that as number of epochs increases, so does execution time. Test accuracy also seems to increase as epochs increase. If we were to increase the epochs by much more, we would start seeing overfitting and the test accuracy should theoretically decrease.

Nadam optimizer:

The Nadam optimizer was chosen as the second optimizer to be tested. This is because they only differ between their momentum calculations. Tables 4-6 will plot the same cases as Table 1-3 in the same order. As seen before, higher learning rate should mean higher accuracy, but since we are using a rather low fixed value for epochs, we see that it actually decreases after a learning rate of 0.01 because we are not using enough epochs to reach the minimum error. For Table 5, we see again that smaller batch sizes have a much larger execution time, for the same reasons explained for the adam optimizer. Finally, for varying epochs, we see execution time increase proportionally to number of epochs, and accuracy slightly improving, just like the case in the Adam optimizer.

Learning Rate	Test Accuracy (%)	Execution time (s)
0.1	0.2698	10.992
0.01	0.9908	10.823
0.001	0.9581	10.958
0.0001	0.8877	11.965

Table 4: Results for net trained with Nadam optimizer with batch size of 32 for 5 epochs

Batch Size	Test Accuracy (%)	Execution Time (s)
1	0.9921	98.592
4	0.9815	31.184
16	0.9662	16.655
64	0.9448	9.477

Table 5: Results for net trained with Nadam optimizer with a learning rate of 0.001 for 5 epochs

# Epochs	Test Accuracy (%)	Execution Time (s)
5	0.9559	11.033
10	0.9804	14.603
15	0.994	18.12
20	0.9997	22.309

Table 6: Results for net trained with Nadam optimizer with a learning rate of 0.001 and a batch size of 32

Conclusion:

In general, the Nadam execution times took generally longer that the Adam optimizer execution times, but the differences were not large enough to conclude that the Adam optimizer is more efficient. The accuracies are also very similar, differing slightly for each case, but they all changed with each run through the data because of the random choosing of weights at the beginning. This aswell shows that both optimizers are very similar and good enough for this problem.

In general, the chosen optimizer should not have drastic effect on the accuracy of the test, as they mainly optimize the algorithm to become more efficient in terms of execution time and computing power. Some optimizers are better suited for certain applications.