Neural Computation CW1: Backpropagation and Softmax

Muhannad Alghafis, Seunghyeon Yeon, Fabrizio Campo, Seonghee Han, Huda Khan

***Abstract*—This report summarizes the work done for completing Coursework 1 of Neural Computation module.**

I. Experiments

In order to tune the hyper-parameters to achieve optimality and achieve the most accurate classification on the test data we picked values that were both bigger and smaller than our best value to see how the functions would respond.

II. Method

By checking the effect of adjusting each hyper-parameter individually we can find the best value for the best accuracy. Then by testing a combination of the optimal hyper-parameter values we can check if we have been correct in estimating how tuning these will affect the classification accuracy by comparing the results with the original values of the hyper-parameters. Original Accuracy with Batch size=50, Learning rate = 0.01, epochs=1000, activation function = ReLu, Network Topology = [784, 20, 20, 20, 10]. This is the classification accuracy for the original code having completed parts 1 to 5. Those are our final values for a good trade-off between accuracy and training time of our algorithm. As the trend for classification accuracy stabilises within 10 minutes we have recorded the results after tuning hyper-parameters after 10 minutes running time in all of the example that you see,so when there is a difference of number of epochs it means that the algorithm runs faster or slower compared to the original. The X-axis measures the number of epochs that have run and the Y-axis measures the accuracy.

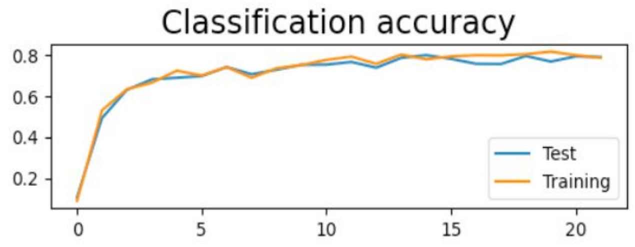

Fig. 1. Original Accuracy

III. Individual Parameter Experiments

A. Training Time

Number of epochs is the number of times the whole training data is shown to the network while training. Increase the number of epochs can lead to high accuracy in the training data and in the validation accuracy as well, if the numbers of epochs is too large than we can have an overfitting and then our algorithm will have poor performance,to avoid this case we have to balance the batch size. Here is an experiments changing the number of epochs to 30 and leaving all the parameter set as Network Topology = [784, 20, 20, 20, 10],Learning rate=0.01, Batch size=50 and epochs=30.

The graph shows the Classification Accuracy after 25 epochs. It takes a relatively small number of epochs to reach the pick of the accuracy,and that is around 20,that is because the more we complete passes through the training dataset the more we learn it and after a certain amount of training the value stabilize. So we can clearly

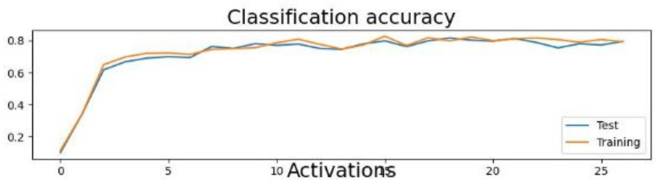

Fig. 2. Accuracy with 30 epochs.

see that just decreasing the number of epochs does not affect our accuracy that much,but the more epochs we have the more accurate we are since the value start to be roughly constant,also changing the number of epochs affects the training time because the algorithm has to go through the dataset either less or more times depending if the value is small or larger. In the section below for combining experiments we will show how changing the number of epochs combined with changes in the batches size affect the accuracy of our algorithm and also the training time. But first we should have a look about the network topology.

B. *Network Topology*

The configuration, or topology, of a network is key to determining its performance. Network topology is the way a network is arranged, including the physical or logical description of how links and nodes are set up to relate to each other. A feedforward neural network is a biologically inspired classification algorithm. It consists of a possibly large number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer and the network goes in only one direction. The input layer has 784 units (28 x 28 input pixels), three different hidden layers of 20 units and 10 output units, one for each of the ten classes.The results of this output is shown in the original classification accuracy. Here is the first experiment with this hyper-parameter. Network Topology [784, 10, 10, 10, 10] We reduced the number of nodes in the three hidden layers. With this kind of topol-

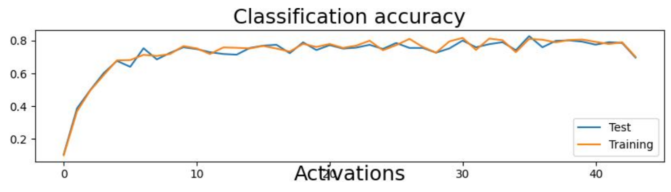

Fig. 3. Accuracy with Network Topology [728,10,10,10,10]

ogy we still use three hidden layers because we have to learn a complex representation,but this time we changed the number of nodes in every hidden layer. The number of hidden layers and the number of neurons in each of these layers must be carefully considered, a large number of neurons in the hidden layers can increase the time the algorithm takes to train the network but it can also cause overfitting,same problem with underfitting that happens when we use a very small number of hidden layer,of course this is not the case.In the graph we can see that changing just the network topology makes a difference on the accuracy and also a different in the training time when running the algorithm for ten minutes as we go through roughly 45 epochs in the same time of the other examples that you can see in this report.The algorithm then runs faster with less nodes in the hidden layers and maintain its accuracy between the range of 0.6-0.8 that is good but not optimal. Things change a bit when we change the network topology to [784, 10, 20, 10, 10](see below) The algorithm runs through roughly 35 epochs

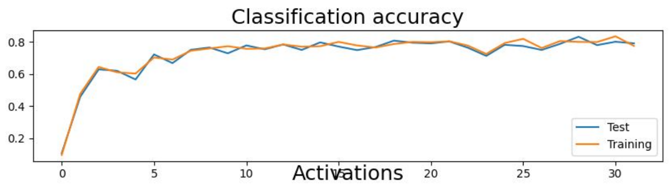

Fig. 4. Accuracy with Network Topology [728,10,20,10,10]

in ten minutes this time,so we can have the confirmation that adding more nodes in hidden layers decreases that training time but it slightly improves the accuracy in our algorithm,also the number of nodes have a tremendous influence on the final output. Last experiment is made reducing the number of hidden layers and the number of nodes in one of them. Network Topology [728, 10,20, 10] Using less hidden layers makes a clear difference

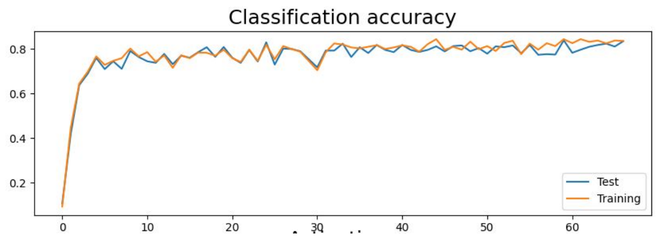

Fig. 5. Accuracy with Network Topology[728,10,20,10]

on the accuracy and on the training time,because the algorithm runs really fast and it goes through roughly 70 epochs and it reaches an high accuracy both in the training set and also in the test set.The algorithm also jump from a value of 0 to a value of 0.7 with a fast acceleration and then the value stays in that range for the rest of the running but it keeps swinging up and down an higher value and a lower value in that range,after 60 epochs it starts increasing.Using two hidden layer will suffice with simple data but using more can always help getting the right output. So far we have seen that the Training time is affected by several hyper-parameters and among all we also have the Learning Rate.

C. Learning Rate

The learning rate is often seen as the most important hyper-parameter to configure for a model. Unfortunately we cannot analytically calculate the optimal learning rate and so must rely on trial and error in order to find a learning rate that is deemed good enough. The original learning rate is set to 0.01. Adjusting the learning rate to 0.1 is detrimental to the accuracy of this model as we can clearly see in the graph below with the accuracy taking a steep dive around the eighth epoch and then failing to improve. Adjusted the learning rate to 0.001 has a largely

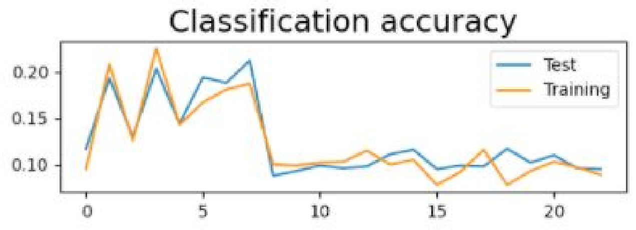

Fig. 6. Accuracy with Learning rate 0.1

similar effect on the test data as the original learning rate did though it is slower to reach the maximum accuracy. From the results of the experiments shown above it is clear to see that the original learning rate of 0.01 has the best effect on classification and is thus deemed most optimal.

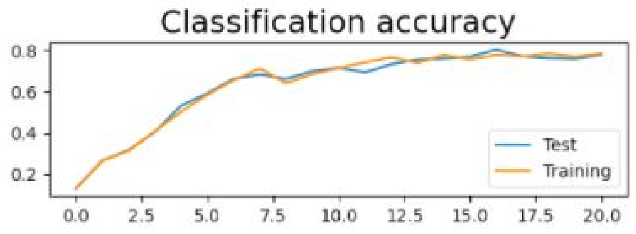

Fig. 7. Accuracy with Learning rate 0.001

D. Batch Size

Mini-batch gradient descent is a popular method in between stochastic gradient descent and batch learning. By tuning the size of the mini-batch we can affect the classification accuracy and also the training time. When we reduce it training time also reduces. The original size of the mini-batch is 50. Adjusting the batch size to 25 results in a slightly smoother trend when compared to the original and easily reaches an accuracy of approximately 80%.

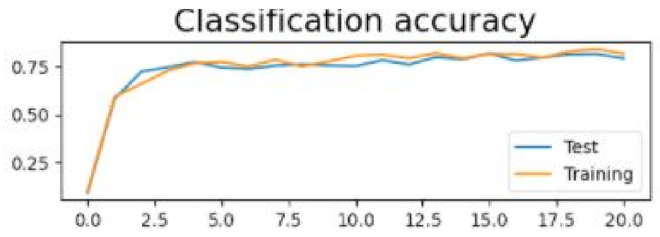

Fig. 8. Accuracy with batch size 25

Adjusting the batch size to 32 does not seem to be as accurate as the original as it does not quite reach the high of 80% despite coming close. Adjusting the batch size to 75 results in sharper dips and slows the acceleration in accuracy. Adjusting the batch size to 200 has a negative effect on the accuracy as shown in the graph below. The graphs show that mini-batches of sizes 25 and 32 yield a classification accuracy of a maximum of 75% whilst mini-batches of sizes 50 and 75 reach a high of 80%. The mini-batch of size 50 results in a smoother trend and does not rise and fall as sharply as the mini-batch of size 75 and so this is what we will use as the optimal value.

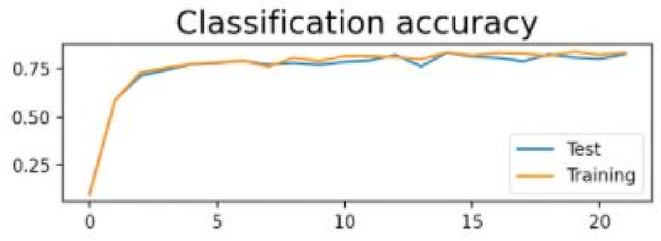

Fig. 9. Accuracy with batch size 32

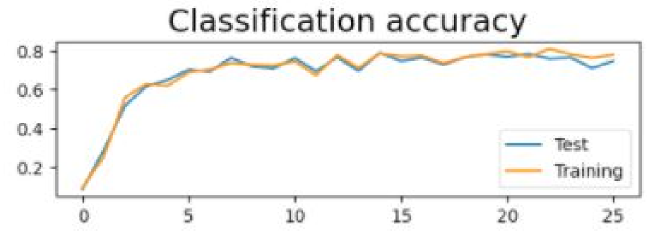

Fig. 10. Accuracy with batch size 75

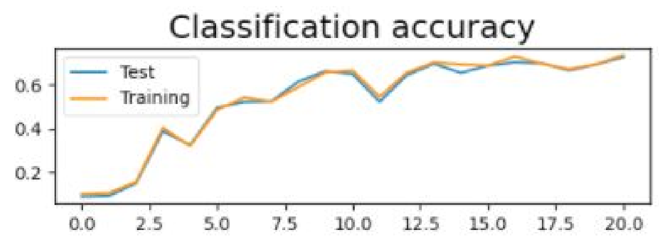

Fig. 11. Accuracy with batch size 200

E. Activation Function

Activation functions help the network learn complex patterns in the data. The original activation function is set to ReLu - Rectified Linear Unit. ReLu is computationally efficient and is non-linear which allows for backpropagation as the derivatives can be found. We test the sigmoid activation function on this model in order to observe how this affects the classification accuracy. Sigmoid functions tend to provide smooth gradients, bound output values and clear predictions. However it is also computationally expensive. Adjusting the batch size to sigmoid results in the following graph:

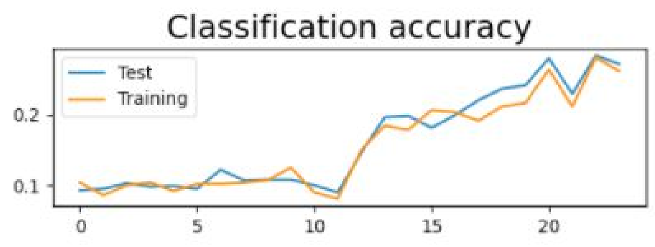

Fig. 12. Activation Function sigmoid

IV. Combined Parameters Experiments

As the learning rate interacts with other aspects of optimisation it will affect the classification accuracy. Generally it seems that smaller learning rates require fewer training epochs. Smaller batch sizes are also suited to smaller learning rates. The combination of optimal batch size at 50, activation function set to ReLu and learning rate at 0.01 results in an accuracy of 80 percent which is achieved after running approximately 7 epochs and then maintains this position with occasional dips and rises but does not result in any major changes. Network Topology [728, 10,10,10,10] and Learning Rate= 0.001 Another combined experiment has been done changing the network topology and the learning rate at the same time,leaving all the other parameters at original values. From this graph we can see that the smaller number of

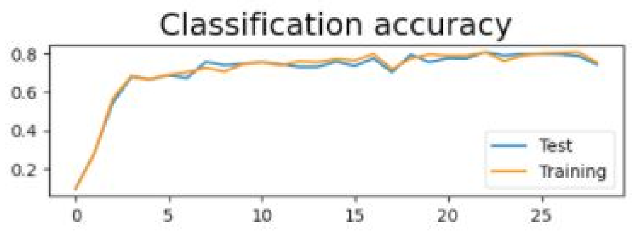

Fig. 13. Combined Experiment 1

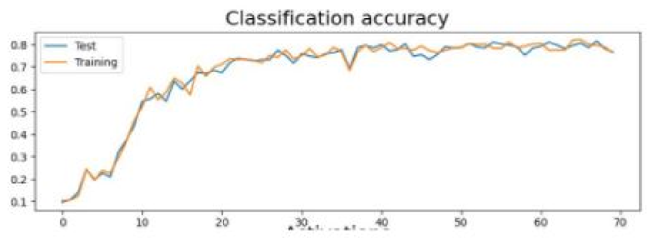

Fig. 14. Combined Experiment 1 - Learning rate 0.001 and Network Topology [728, 10,10,10,10]

nodes makes the algorithm faster because it goes through 70 epochs in roughly 10 minutes but not as fast as the individual experiments for the network topology with the same parameter,because this time is also affected by a smaller learning rate that also affect the accuracy,in fact in this experiment we reach a relatively good accuracy after 30 instead with a learning rate of 0.01 we reach it after 10 epochs. Another experiment with Network Topology [728, 10,20,10,10] and Learning Rate= 0.001 is shown below. Using a lower number for learning

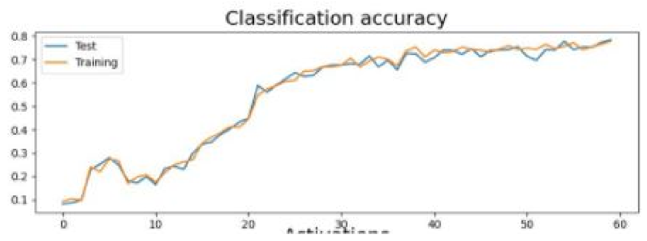

Fig. 15. Combined Experiment 1 - Learning rate 0.001 and Network Topology [728, 10,20,10,10]

rate increase the training time and combining it with a changes in the network topology makes the algorithm reaching a good accuracy after several epochs,the improvement in the classification accuracy is slow and we do not have any specific acceleration,just a noticeable increase between 10 epochs and 20. At 10 epochs the accuracy is still low at around 0.2 but at 20 epochs we are at 0.5 and with small improvement we then reach a decent accuracy just before 40 epochs. Increasing

We now test the epochs with the batch size and observe the trend of our accuracy.

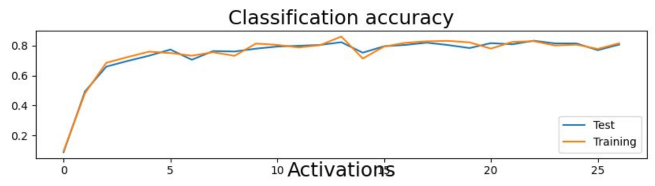

Fig. 16. Combined Experiment 1 - Batch size 25

With batch size 25 and a number of epochs set to 25 the algorithm performs really well and we reach an accuracy of 80%, the accuracy starts rising after a few epochs and then it remains in the range of 0.7-0.8 from 4 to the rest of the epochs, it reaches the peak of 0.85 after 15 epochs and then it fluctuates between that range. With this hyper-parameters value the algorithm has a good performance.

V. Conclusion

After testing the hyper-parameters individually and obtaining the optimal values, and then testing combinations of these values we can obtain the maximum accuracy for the classification model. Following the trend in the graphs where we run the model until we reach approximately 20 epochs, it is clear to see that changing the epochs to 20 does not vary the accuracy of the classification very much, if at all. Regarding combinations of hyper-parameters it is clear to see that the original values for the hyper-parameters obtain an accuracy of 80% and changing these values has not resulted in a significant increase.

We have seen that adjusting the parameters can impact the accuracy negatively e.g. using the sigmoid activation function or changing the batch size to 200 or setting the learning rate to 0.1, but in terms of improving the accuracy the original values work together well to reach a consistent accuracy of 80%. These values are a learning rate of 0.01, batch size of 50, epochs set to 1000, activation function set to ReLu and network shape [728,10,20,10,10]. This network topology can be considered to be the best one because return a value of 80 very quickly after 10 epochs and then it seems to remain constant over time with higher spikes at some point after 25 epochs in which we reach a value higher than 80

Having used a process of trial and error we have realised that combinations of parameters can yield different results to the expected results extracted from the results of individual hyper-parameter testing.