EIE 4100 Lab2

HAO Jiadong 20084595d

1. The results and comments of the four tasks

1a. MLP with 0 hidden layers on the MNIST dataset

The parameter setting of the MLP with 0 hidden layer is shown in Figure 1.

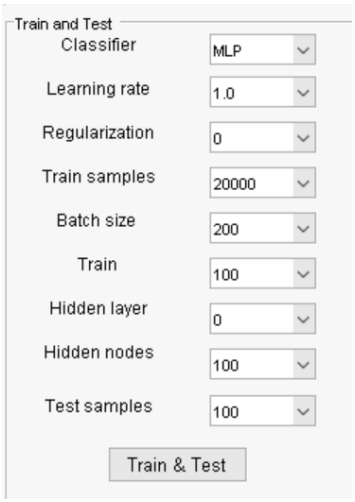

Figure 1. Parameter setting for MLP with 0 hidden layer

As shown in Figure 2, the training error rapidly dropped from 0.26 to about 0.16 in the first 20 epochs. However, feeding more data to the model didn't improve the performance significantly after epoch 20, which indicates that the model performance had reached the model's capacity.

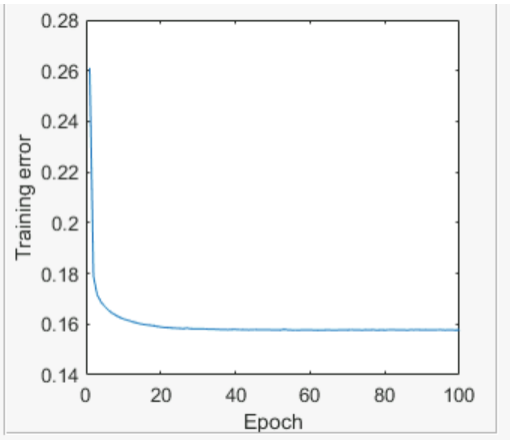

Figure 2. Training error for MLP with 0 hidden layer

According to Figures 3 and 4, the training and testing accuracy of the MLP with 0 hidden layers is 0.8898 and 0.85, respectively. The performance of the model is acceptable but could be more impressive.

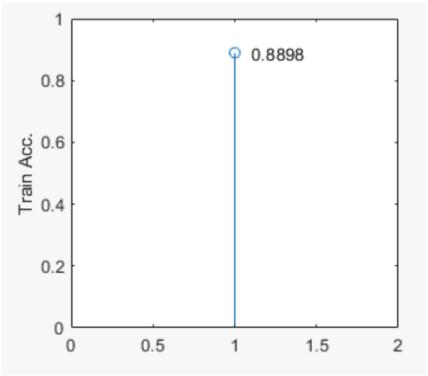

Figure 3. Training accuracy for MLP with 0 hidden layer

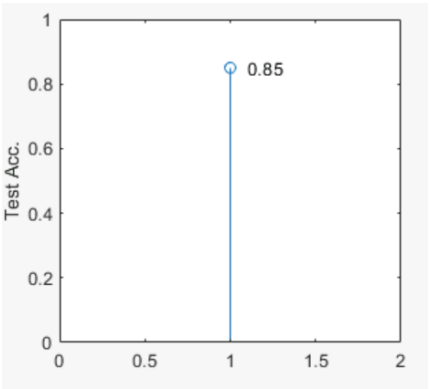

Figure 4. Testing accuracy for MLP with 0 hidden layer

According to the testing confusion matrix shown in Figure 5, the model classified the figures “0”, “1”, “4”, and “6” perfectly with 100% accuracy. However, the model made poor predictions on figures “8” and “9” with accuracy of merely 0.62 and 0.64 respectively. For figure “8”, 12% and 25% of them were wrongly classified as “4” and “6” respectively. For figure “9”, 18%, 9% and 9% of them were misclassified as “4”, “6” and “7”. We can see that the model doesn’t perform well in distinguishing “8”, “9” from “4”, “6”.

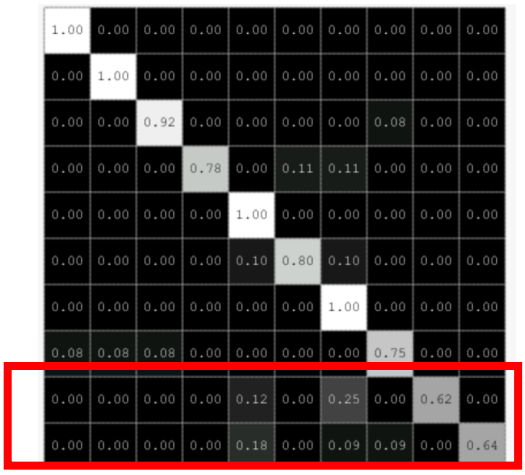

Figure 5. Testing confusion matrix for MLP with 0 hidden layer

1b. MLP with 1 hidden layer on the MNIST dataset

The parameter setting of the MLP with 1 hidden layer is shown in Figure 6.

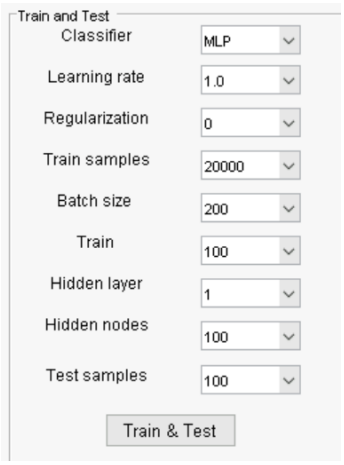

Figure 6. Parameter setting for MLP with 1 hidden layer

From Figure 7, the training error reached the bottom (around 0.26) at about epoch 20, and after that, the error didn't drop significantly. This means that even with one more hidden layer (more parameters), the MLP could still get the optimized parameters after around 20 epochs. Hence, if we want to improve the model's performance further, feeding more data won't help. We need to consider more complex model architectures.

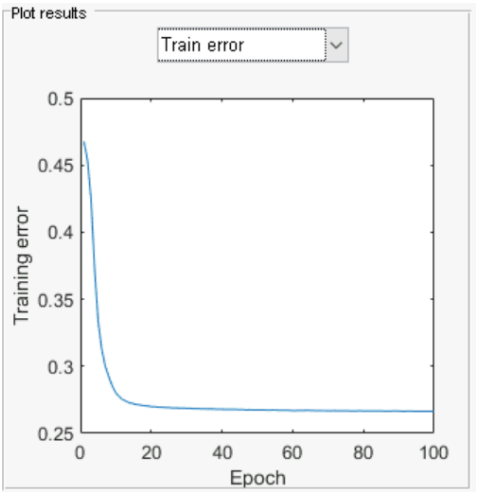

Figure 7. Training error for MLP with 1 hidden layer

Figures 8 and 9 show that adding one hidden layer to the MLP didn't improve the model's performance. The training accuracy dropped a bit from 0.8898 (0 hidden layer) to 0.8793 (1 hidden layer), and the testing accuracy dropped from 0.85 (0 hidden layer) to 0.81(1 hidden layer). The reason for this drop may be that the task is simple enough for an MLP without a hidden layer to handle. Adding one hidden layer will make the feature more abstract, making it hard for the output layer to make the correct classification.

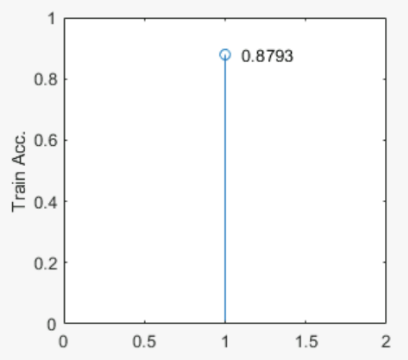

Figure 8. Training accuracy for MLP with 1 hidden layer

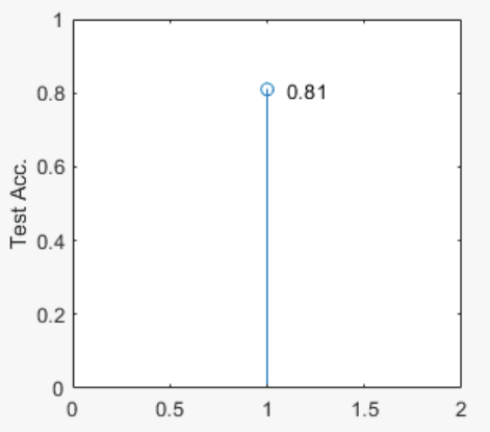

Figure 9. Testing accuracy for MLP with 1 hidden layer

According to the confusion matrix shown in Figure 10, the classifier was still unable to classify figures “8” and “9” correctly, with an accuracy of only 50% and 55%, respectively. 25% of the “8” were misclassified as “6” and 12 % were misclassified as “1” and “4” respectively. 27% of the “9” were misclassified as “4”.

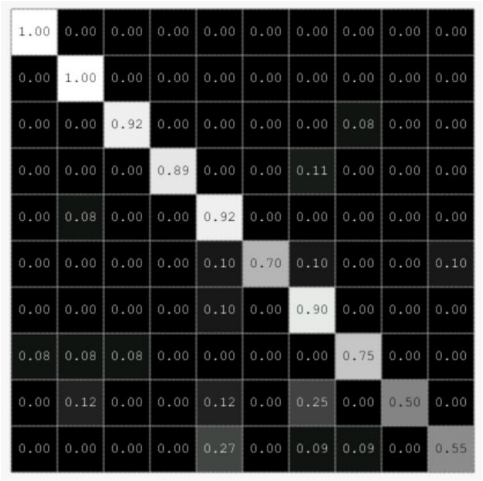

Figure 10. Testing confusion matrix for MLP with 1 hidden layer

1c. CNN on the MNIST dataset

The parameter setting of the CNN is shown in Figure 11.

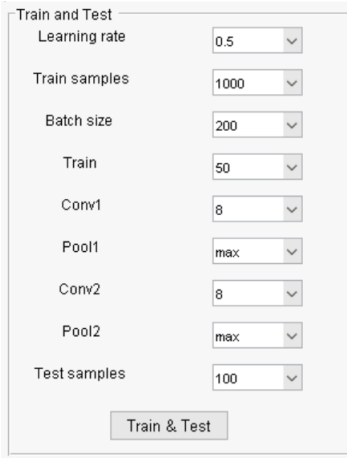

Figure 11. Parameter setting for CNN

From Figure 12, the training error curve decreased continuously in the 50 epochs. At epoch 50, the error curve still had a decreasing trend, which indicates that the model still had room for improvement. If we want to achieve a better model performance, we may consider feeding more training data to the model.

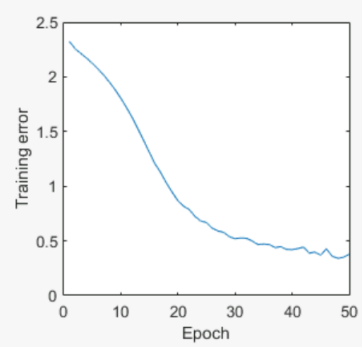

Figure 12. Training error for CNN with the initial setting

Figures 13 and 14 show that the CNN with the initial setting cannot significantly outperform the MLP models, with the training and testing accuracy of 0.871 and 0.83, respectively.

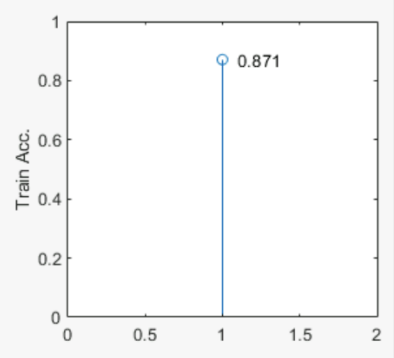

Figure 13. Training accuracy for CNN with the initial setting

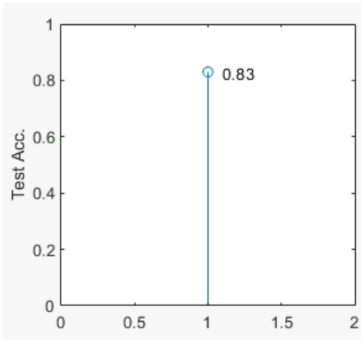

Figure 14. Testing accuracy for CNN with the initial setting

According to the confusion matrix shown in Figure 15, CNN can classify figures “0”, “1”, and “8” without any error. However, it performs badly when classifying “7”, with an accuracy of only 58%.

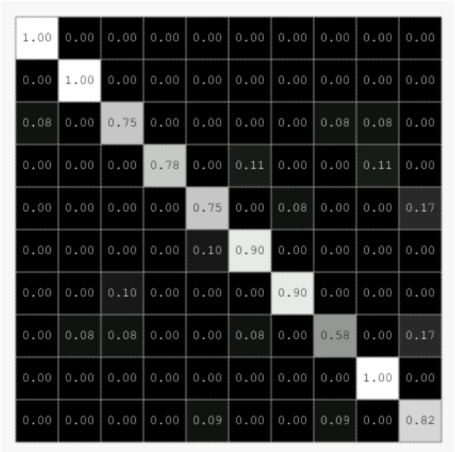

Figure 15. Testing confusion matrix for CNN with the initial setting

1d. CNN on the CIFAR 10 dataset

We use the initial parameter setting of the CNN shown in Figure 16.

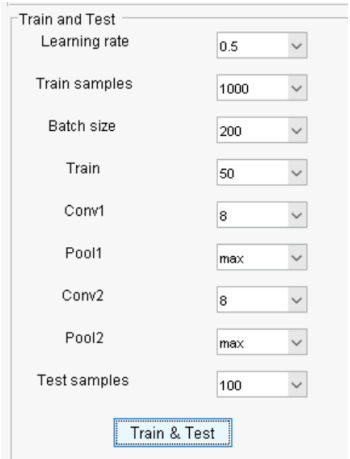

Figure 16. Parameter setting for CNN

According to Figure 17, the training error decreased dramatically in the 50 epochs to about 2.0225. At epoch 50, there was a significant decreasing trend, which indicates that the training data was not enough to fully leverage the model's capacity. To further improve the model performance, we may consider adding more epochs or incorporating more training samples in each epoch.

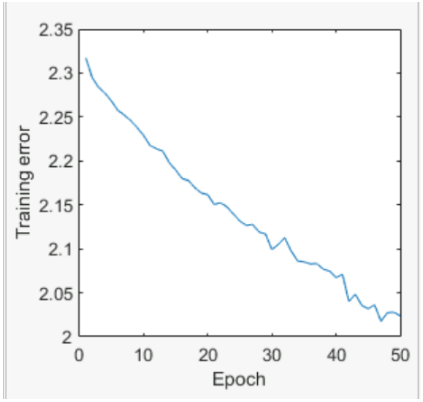

Figure 17. Training error for CNN on the CIFAR 10 dataset

From Figures 18 and 19, CNN's performance on the CIFAR 10 dataset is unacceptable, with training and testing accuracy of 0.26 and 0.22. The same CNN could perform well on MNIST but couldn't on CIFAR 10. The main factor for this discrepancy is the task difficulty. The task of classifying figures is simple because the picture is black and white, and there are no distractions but only a hand-written figure on the black canvas. It is easy for a CNN with simple architecture to learn the patterns with small-scale training data. However, the task of classifying the ten objects in the CIFAR 10 dataset is relatively difficult. The pictures are in color, and there are many other objects in the same picture together with the labeled objects, which makes it hard for a simple CNN to learn how to classify them within 50 epochs and 1000 training samples in each epoch.

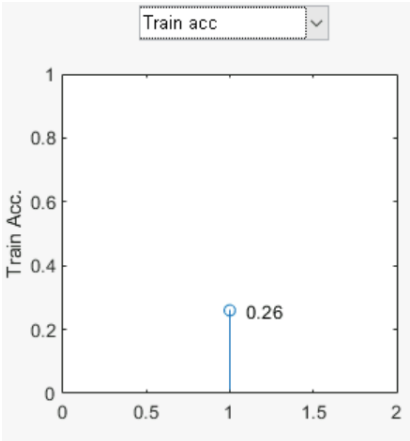

Figure 18. Training accuracy for CNN on the CIFAR 10 dataset

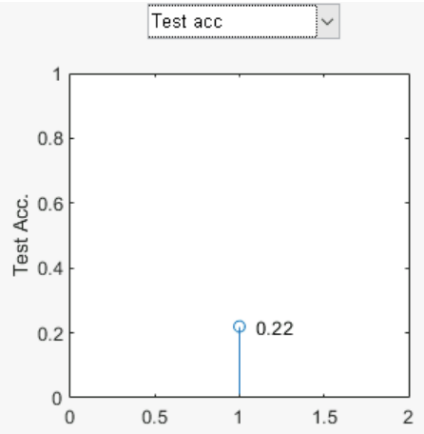

Figure 19. Testing accuracy for CNN on the CIFAR 10 dataset

From Figure 20, there are no large values on the diagonal line of the confusion matrix, which means that CNN cannot effectively classify any of the ten objects. The entries of the first column are filled, which means that many of the objects from the other 9 classes were misclassified to the first class.

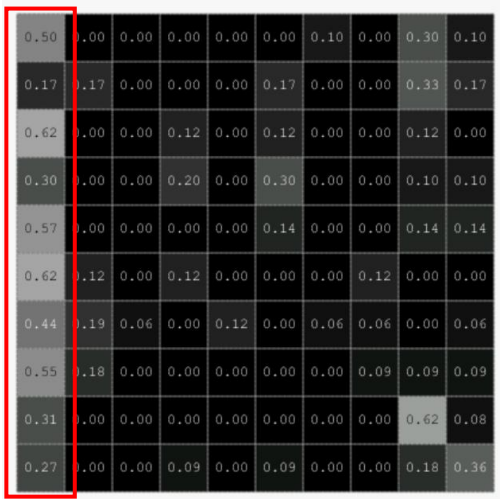

Figure 20. Testing confusion matrix for CNN on the CIFAR 10 dataset

2. Modification of MLP with a hidden layer

2a. Change the learning rate [0.05, 0.1, 0.5]

As Figure 21 shows, we set the learning rate of the MLP with one hidden layer to **0.05**.

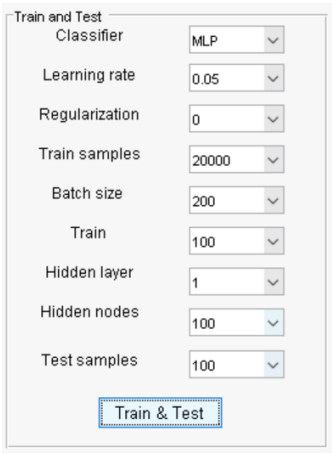

Figure 21. MLP with 1 hidden layer, learning rate = 0.05

As shown in Figure 22, the training error kept decreasing throughout the training process. At epoch 100, it still showed a significant decreasing trend, which indicates that the training data was not enough for the MLP to get the optimized parameters. Hence, the model's performance is poor, with training and testing accuracy of only 0.658 and 0.65, respectively, as shown in Figures 23 and 24. According to Figure 25, the model was unable to recognize "5" and "8" at all. Overall, the learning rate of 0.05 may be too small for the model to get the optimized parameters within 100 epochs, which leads to its poor performance.

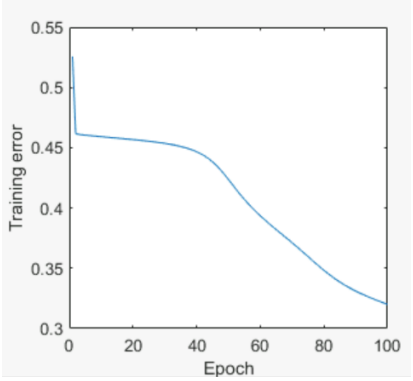

Figure 22. Training error for MLP with learning rate = 0.05

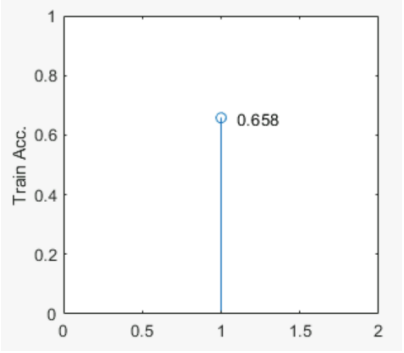

Figure 23. Training accuracy for MLP with learning rate = 0.05

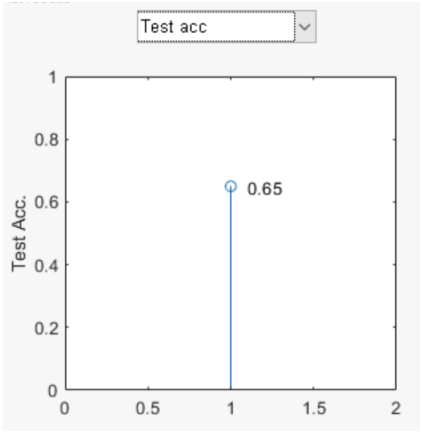

Figure 24. Testing accuracy for MLP with learning rate = 0.05

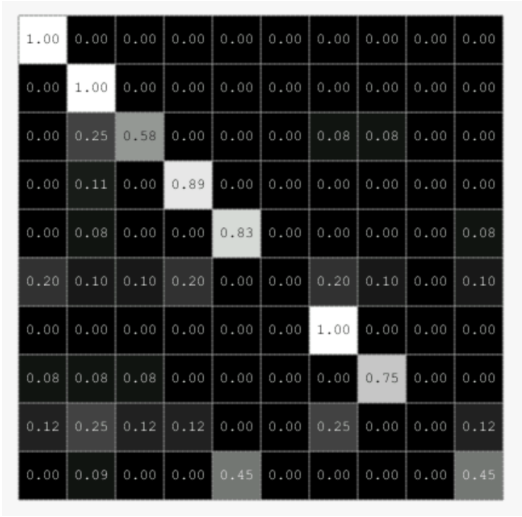

Figure 25. Testing confusion matrix for MLP with learning rate = 0.05

As Figure 26 shows, we increased the learning rate of the MLP with one hidden layer to **0.1**.

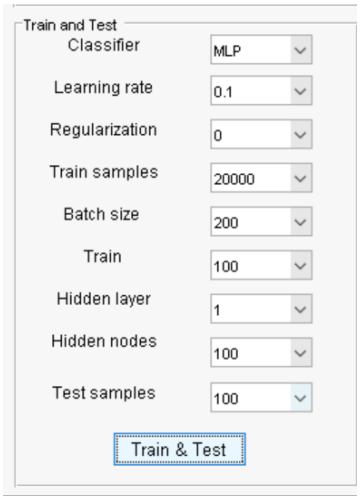

Figure 26. MLP with 1 hidden layer, learning rate = 0.1

As shown in Figure 27, the training error at epoch 100 is around 0.27, which showed a significant improvement compared to the MLP with a learning rate of 0.05, as shown in Figure 22 (around 0.32). The training and testing accuracy increased to 0.84585 and 0.79, respectively, as shown in Figures 28 and 29. However, the training error still displayed a decreasing trend at the end of epoch 100, which means that there is still room for improvement in the model performance.

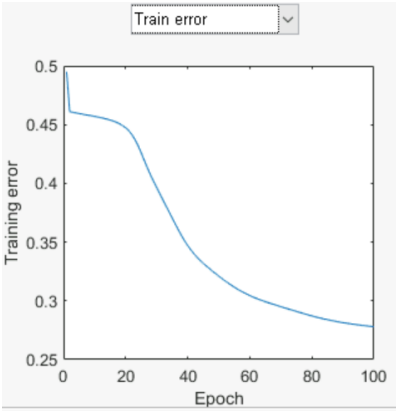

Figure 27. Training error for MLP with learning rate = 0.1

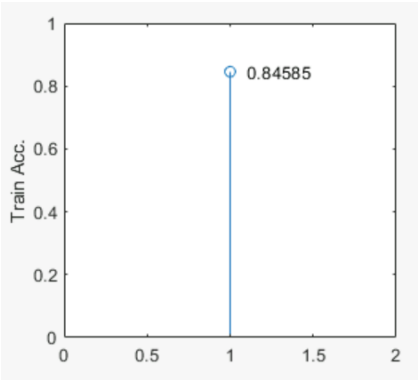

Figure 28. Training accuracy for MLP with learning rate = 0.1

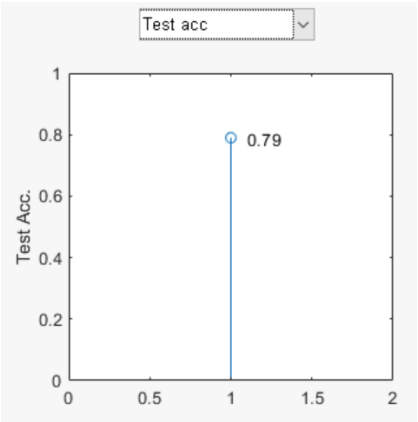

Figure 29. Testing accuracy for MLP with learning rate = 0.1

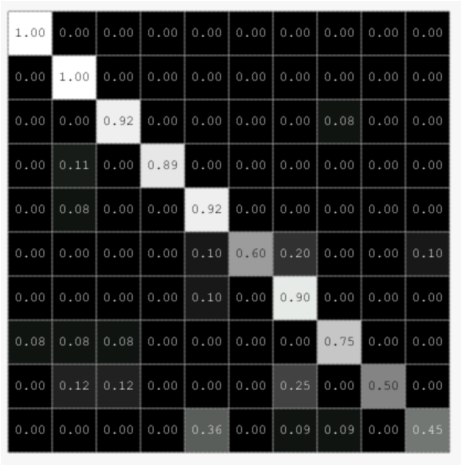

Figure 30. Testing confusion matrix for MLP with learning rate = 0.1

As Figure 31 shows, we further increase the learning rate of the MLP with one hidden layer to **0.5**.

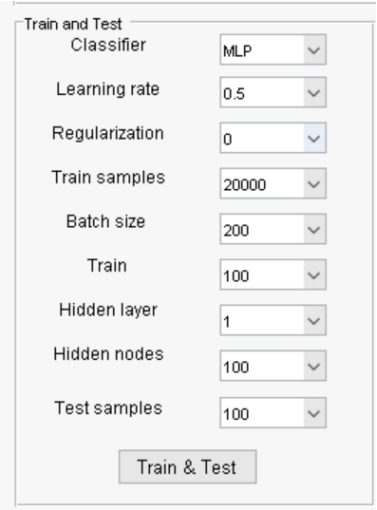

Figure 31. MLP with 1 hidden layer, learning rate = 0.5

As shown in Figure 32, the training error displayed a smooth decrease and converged with an error of around 0.26, which indicates that we can get the fully optimized MLP within 100 epochs with a learning rate of 0.5. Hence, the model achieved training and testing high accuracy of 0.88025 and 0.81, as shown in Figures 33 and 44.

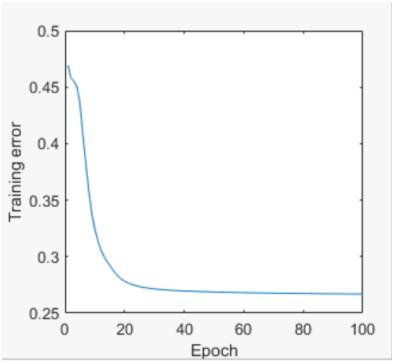

Figure 32. Training error for MLP with learning rate = 0.5

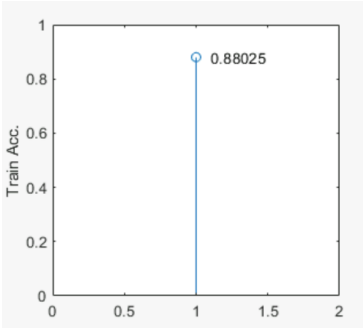

Figure 33. Training accuracy for MLP with learning rate = 0.5

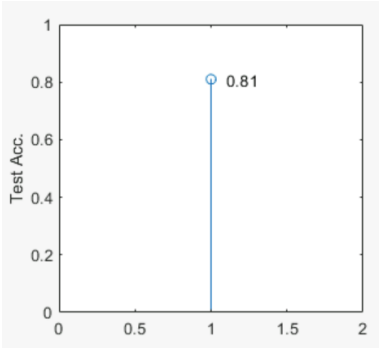

Figure 34. Testing accuracy for MLP with learning rate = 0.5

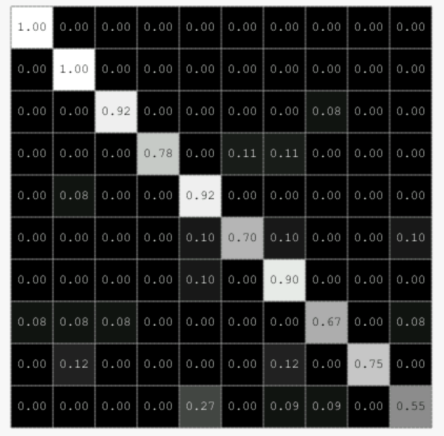

Figure 35. Testing confusion matrix for MLP with learning rate = 0.5

In conclusion, the learning rates of 0.05 and 0.1 are too small for the model to get the optimized parameters within 100 epochs, while the learning rate of 0.5 is appropriate for the model to converge within 100 epochs.

2b. Change the number of training samples [1000, 10000, 30000]
As Figure 36 shows, we set the number of training samples to **1000**.

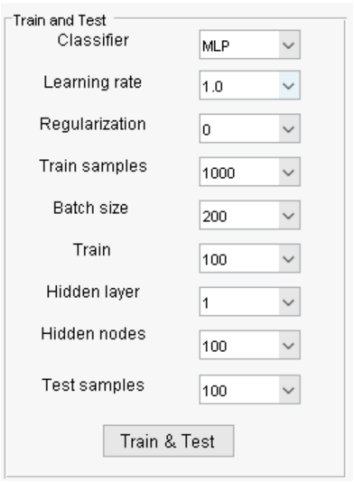

Figure 36. MLP with 1 hidden layer, training sample = 1000

As shown in Figure 37, the training error dropped quickly in the 100 epochs and ended up at around 0.32. The loss curve has a significant trend to further decrease at epoch 100, which indicates that the number of training samples of 1000 is not enough for the model to get the optimized performance.

The training and testing accuracy of the model is relatively low, only 0.663 and 0.64, shown in Figures 38 and 39, further proving that the model is not optimized.

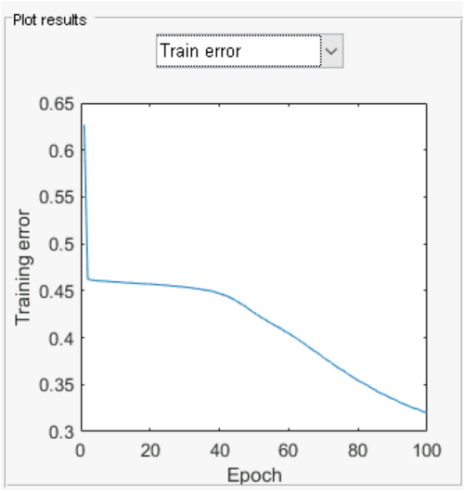

Figure 37. Training error for MLP with training sample = 1000

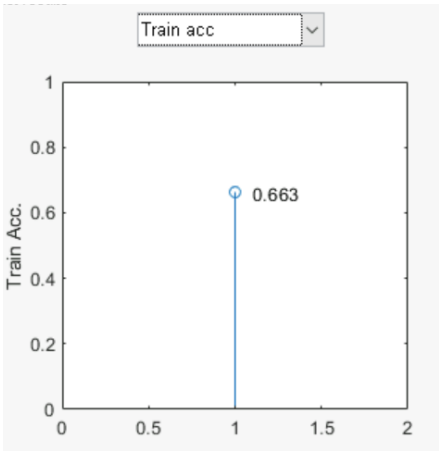

Figure 38. Training accuracy for MLP with training sample = 1000

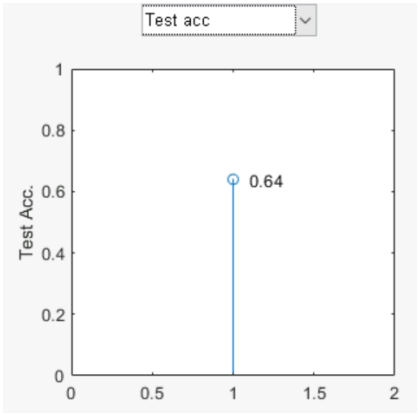

Figure 39. Testing accuracy for MLP with training sample = 1000

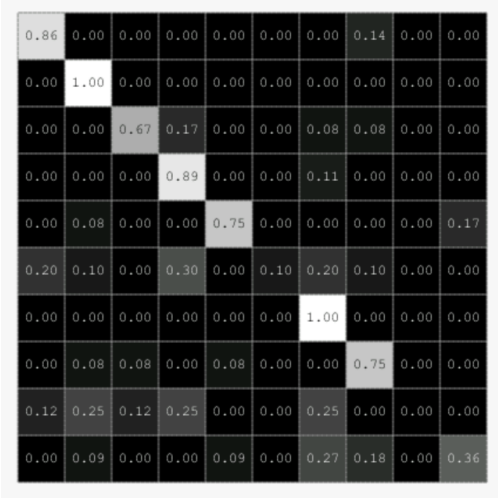

Figure 40. Testing confusion matrix for MLP with training sample = 1000

As Figure 41 shows, we increased the number of training samples to **10000**.

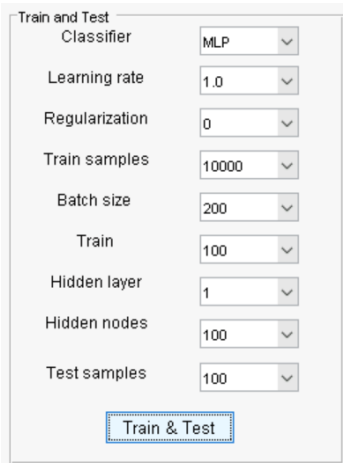

Figure 41. MLP with 1 hidden layer, training sample = 10000

As shown in Figures 42, 43, and 44, the MLP with training sample =10000 achieved better performance, with the final training error ending up at around 0.26, training and testing accuracy of 0.8749 and 0.79, respectively. The training error curve had nearly converged at epoch 40, indicating that 10000 training samples are enough to get the model's optimized parameters within 100 epochs.

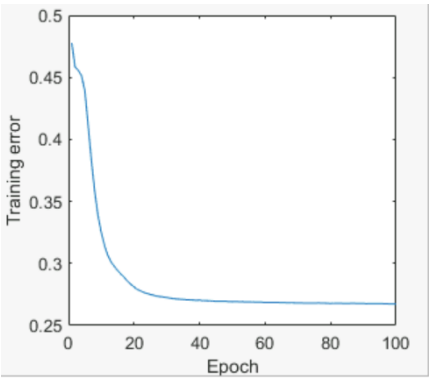

Figure 42. Training error for MLP with training sample = 10000

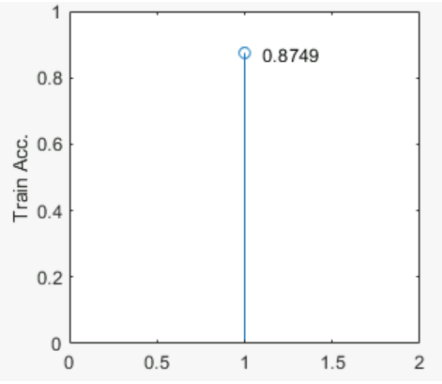

Figure 43. Training accuracy for MLP with training sample = 10000

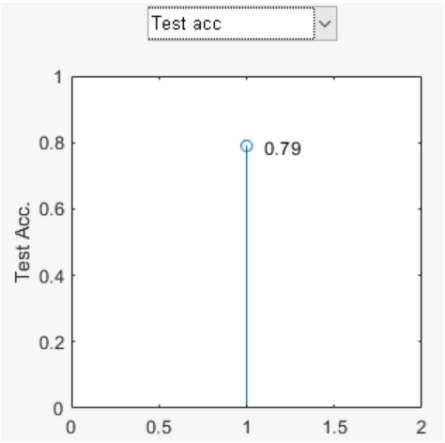

Figure 44. Testing accuracy for MLP with training sample = 10000

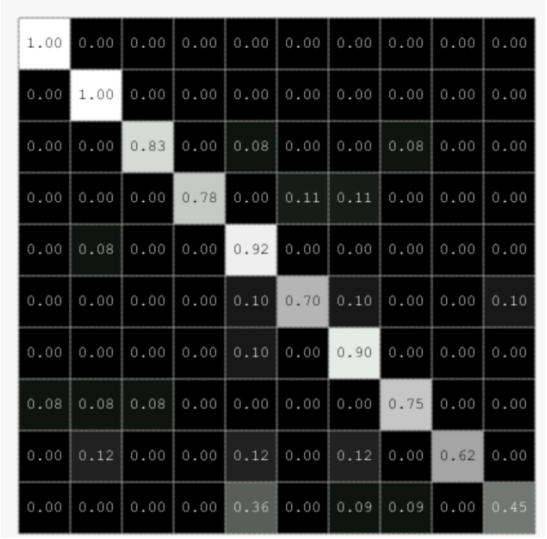

Figure 45. Testing confusion matrix for MLP with training sample = 10000

As Figure 46 shows, we further increased the number of training samples to **30000**.

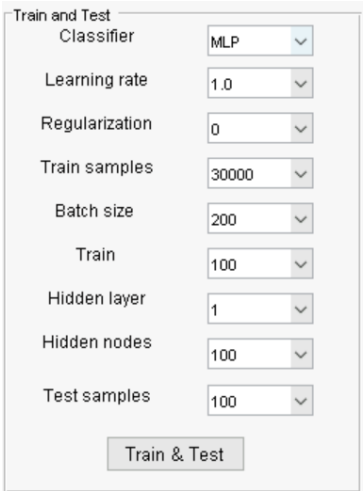

Figure 46 MLP with 1 hidden layer, training sample = 30000

As shown in Figures 47, 48, and 49, even though we fed more samples to the model, the model achieved a similar final performance compared to the model with training sample = 10000, with the training error ending up at around 0.26, training and testing accuracy of 0.87687 and 0.82 respectively. The difference is that the model with a training sample = 30000 (converged at around epoch 20) converged faster than the one with a training sample = 10000 (converged at around epoch 40). This is because, in each epoch, we fed more data to the model so that the model would see enough data in an earlier stage. However, it is worth noting that even though we increased the number of training samples from 10000 to 30000, the training and testing performance didn't improve significantly. In this case, the training sample = 10000 is preferred to save our computational resources and time.

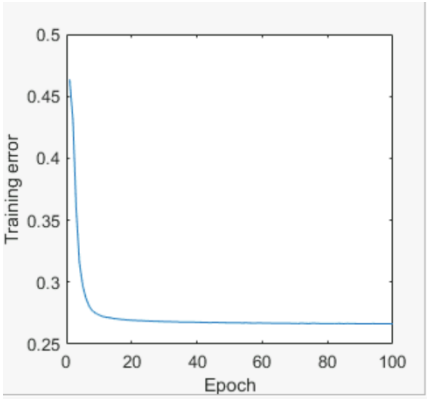

Figure 47. Training error for MLP with training sample = 30000

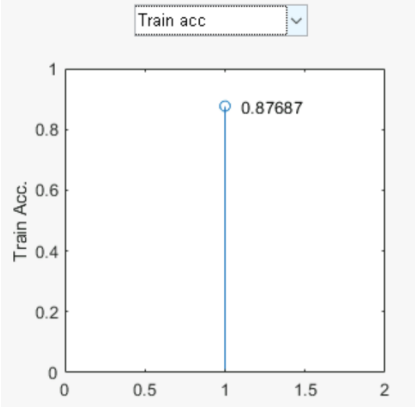

Figure 48. Training accuracy for MLP with training sample = 30000

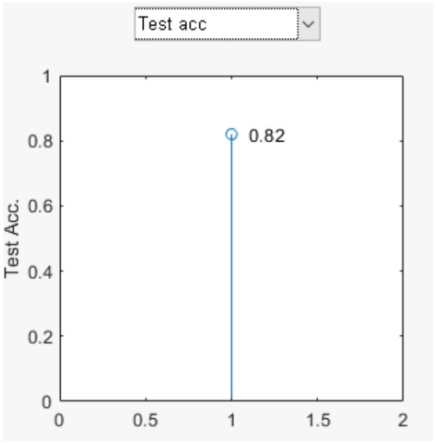

Figure 49. Testing accuracy for MLP with training sample = 30000

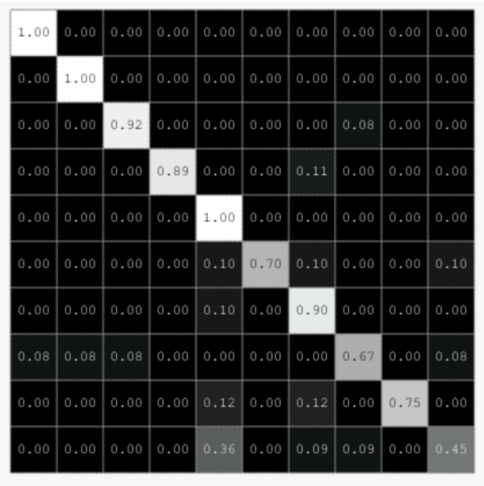

Figure 50. Testing confusion matrix for MLP with training sample = 30000

In conclusion, the training sample size of 1000 is not sufficient for the model to converge, while the training sample sizes of 10000 and 30000 are enough. However, to save our training time and computational resources, we may choose 10000 rather than 30000.

2c. Change the number of training epochs [200, 500, 1000]

As shown in Figure 51, we set the training epochs to **200**.

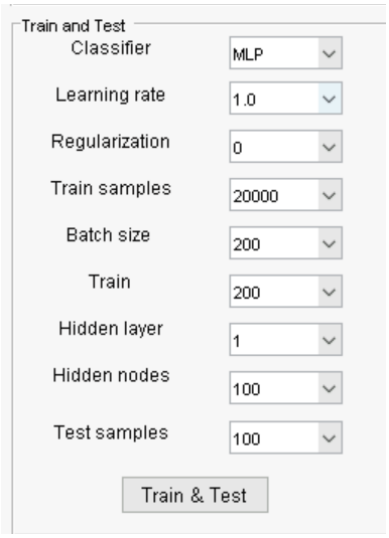

Figure 51. MLP with 1 hidden layer, training epoch= 200

As Figure 52 depicts, the error dropped significantly during the first 20 epochs and remained nearly unchanged after 50 epochs. Finally, we can get an optimized model with training and testing accuracy of 0.8789 and 0.8. In the last 100 epochs, the training error didn't change significantly. We may consider reducing the training epochs to save our time.

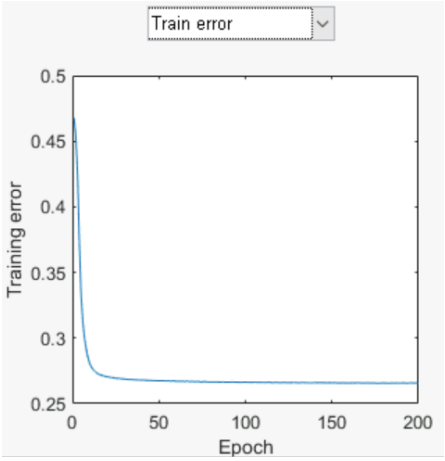

Figure 52. Training error for MLP with training epoch= 200

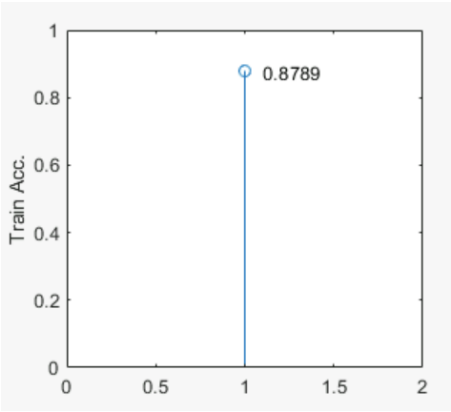

Figure 53. Training accuracy for MLP with t training epoch= 200

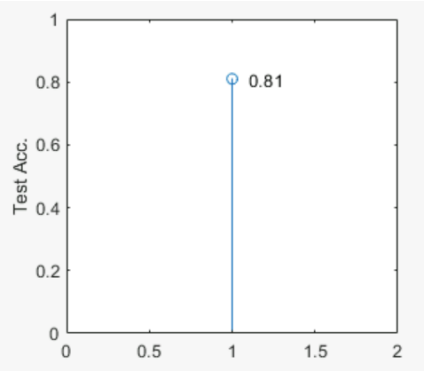

Figure 54. Testing accuracy for MLP with training epoch= 200

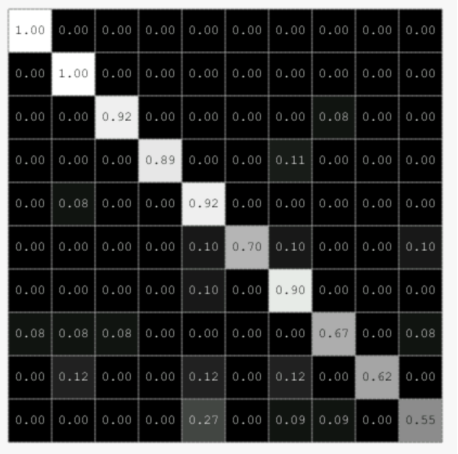

Figure 55. Testing confusion matrix for MLP with training epoch= 200

As shown in Figure 56, we further increased the training epoch to **500**. Figures 57, 58, 59, and 60 show that the MLP with training epoch = 500 achieved a similar performance as the one with training epoch = 200. This is because the training had been converged in the first 100 epochs so the updates in the rest epochs were not critical.

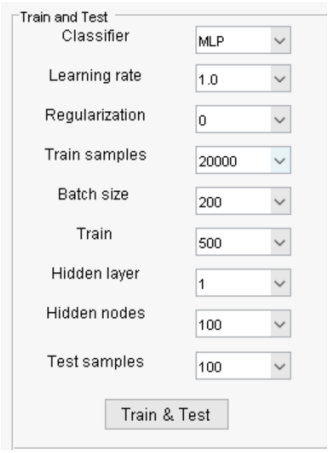

Figure 56. MLP with 1 hidden layer, training epoch= 500

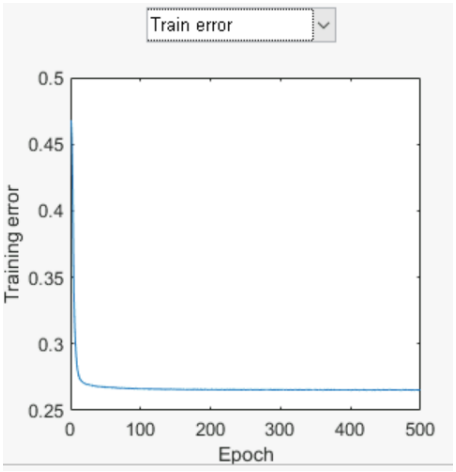

Figure 57. Training error for MLP with training epoch= 500

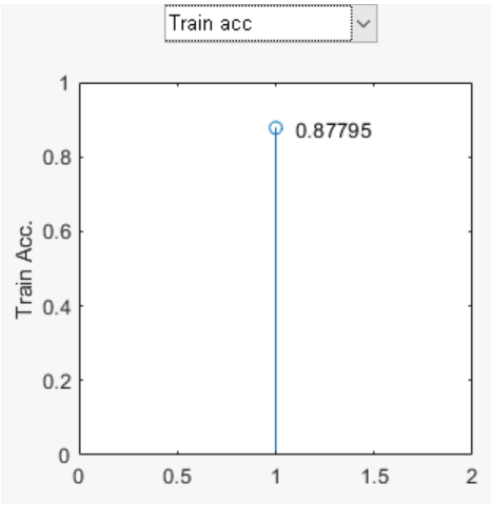

Figure 58. Training accuracy for MLP with t training epoch= 500

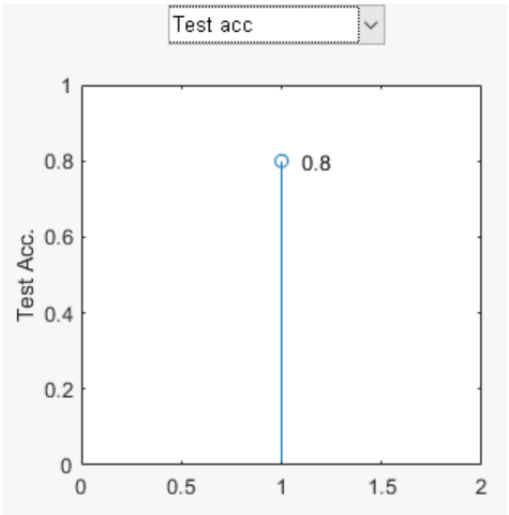

Figure 59. Testing accuracy for MLP with training epoch= 500

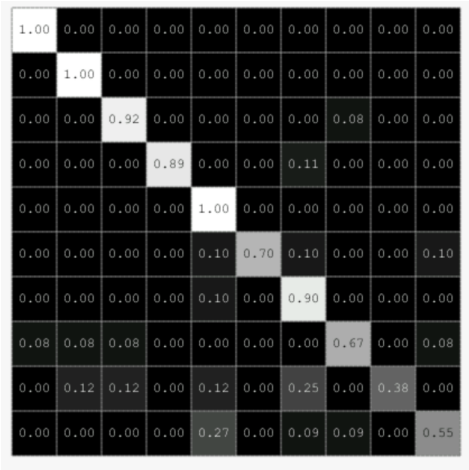

Figure 60. Testing confusion matrix for MLP with training epoch= 500

As shown in Figure 61, we further increased the training epoch to **1000**. Figures 62, 63, 64, and 65 show that the performance of the MLP with training epoch = 1000 is similar to the two MLPs above. The reason is still the early convergence at around epoch 100.

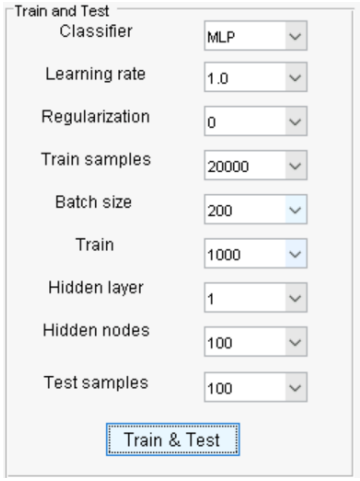

Figure 61. MLP with 1 hidden layer, training epoch= 1000

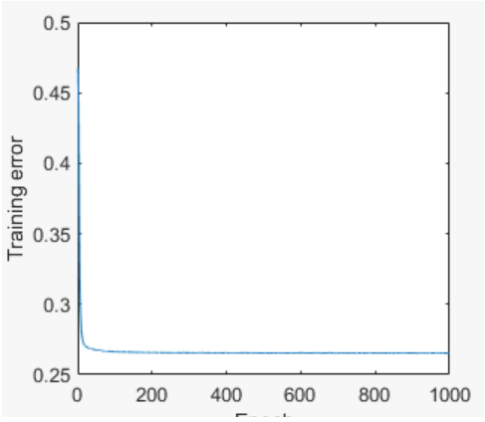

Figure 62. Training error for MLP with training epoch= 1000

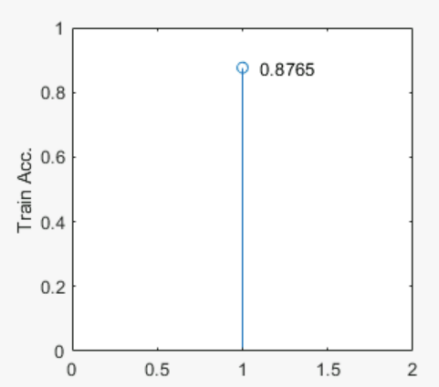

Figure 63. Training accuracy for MLP with t training epoch= 1000

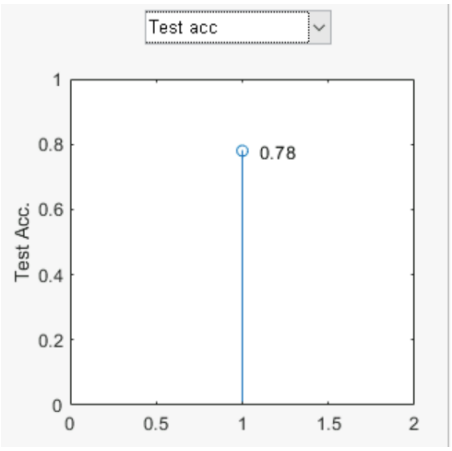

Figure 64. Testing accuracy for MLP with training epoch= 1000

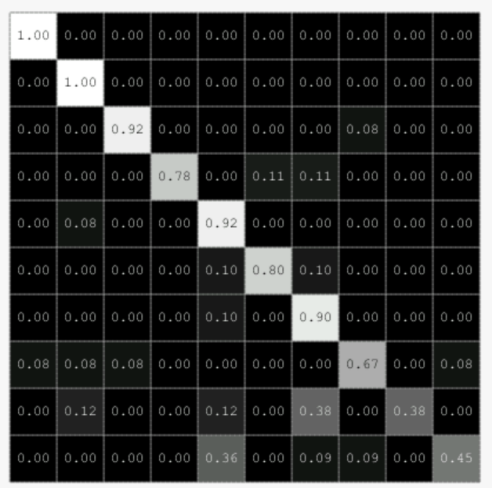

Figure 65. Testing confusion matrix for MLP with training epoch= 1000

In conclusion, for the above three models, the training error decreased significantly in the first 20 epochs, reached around 0.26 at epoch 100, and remained nearly unchanged after epoch 100. The performance of the three models was similar, with training and testing accuracy of 0.88 and 0.8 approximately. The confusion matrix of the four models was also similar, showing the models' inability to effectively classify the figures "8"

and “9”. The above results indicate that under the current model setting, 100 epochs are enough to train the model, and more epochs will not bring significant improvements in the models’ performance.

3. Modification on CNN

Initial model:

The parameter setting of the CNN is shown in Figure 66.

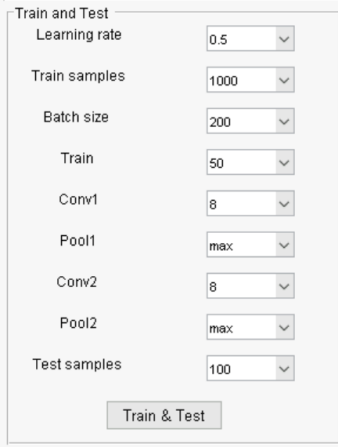

Figure 66. Parameter setting for initial CNN

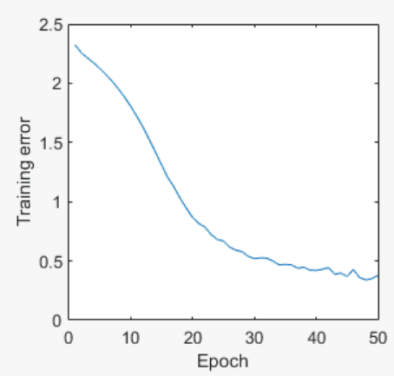

Figure 67. Training error for CNN with the initial setting

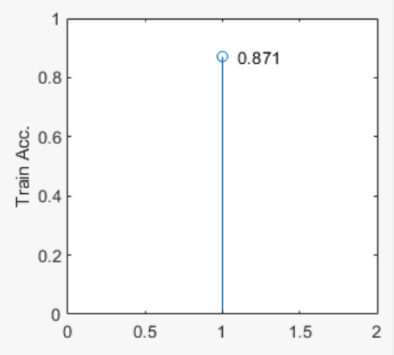

Figure 68. Training accuracy for CNN with the initial setting

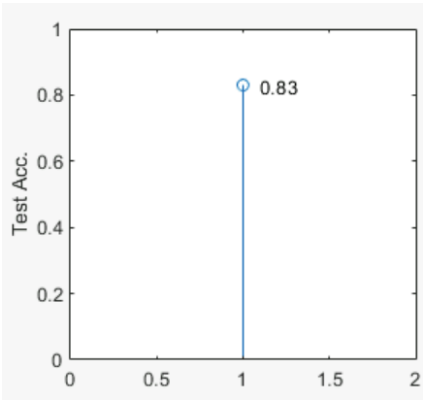

Figure 69. Testing accuracy for CNN with the initial setting

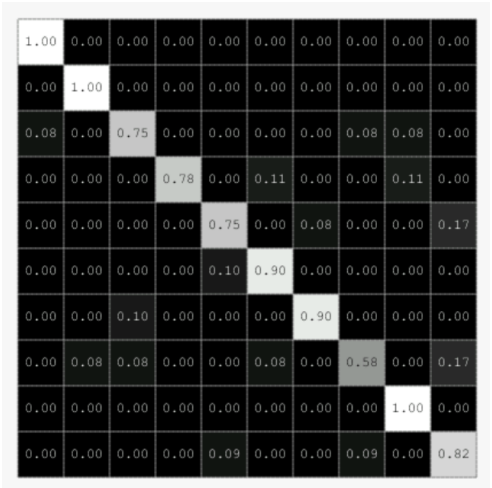

Figure 70. Testing confusion matrix for CNN with the initial setting

3a. Change the number of channels to 16

As shown in Figure 71, we increased the number of channels of two convolution layers from 8 to 16.

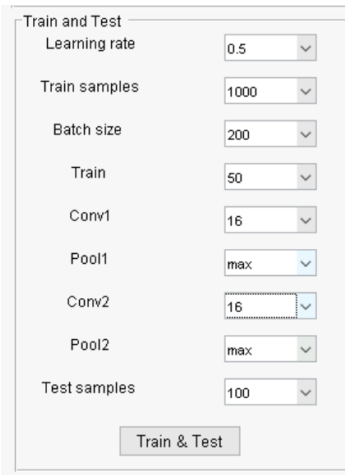

Figure 71. Parameter setting for CNN with number of channels = 16

Compared to the baseline CNN with the number of channels = 8, the modified CNN with the number of channels = 16 achieved a better result. The training and testing accuracy of the modified CNN is 0.902 and 0.87, respectively, which is higher than that of the baseline CNN (0.871 and 0.83 as shown in Figures 68 and 69). In CNN, different channels are used to extract different kinds of features from the image. More channels could provide more useful information to the network to classify the images, and that may be the reason why CNN with 16 channels outperformed that with 8 channels.

According to the training error graph shown in Figure 72, there was still a decreasing trend at the end of epoch 50, indicating that the model had the potential to achieve better performance by being fed more data.

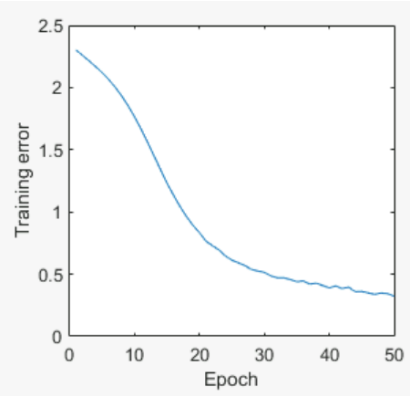

Figure 72. Training error for CNN with number of channels = 16

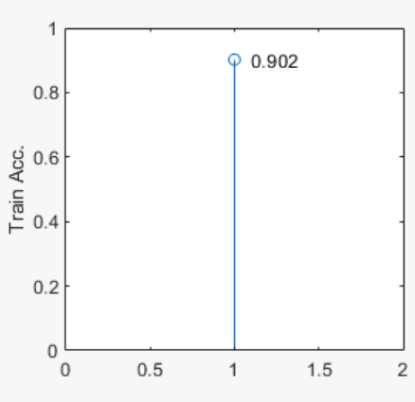

Figure 73. Training accuracy for CNN with number of channels = 16

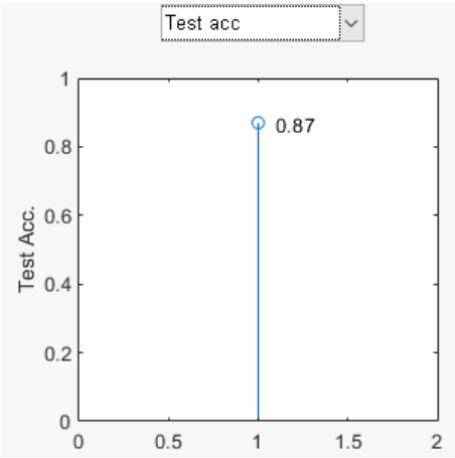

Figure 74. Testing accuracy for CNN with number of channels = 16

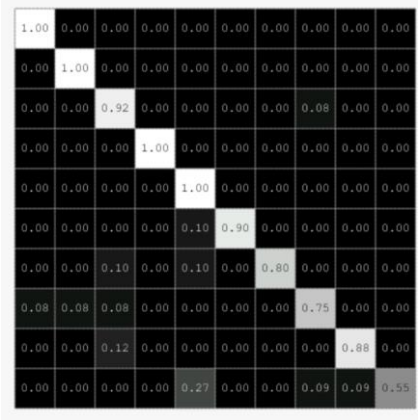

Figure 75. Testing confusion matrix for CNN with number of channels = 16

3b. Use average pooling instead of max pooling

As shown in Figure 76, we changed the max pooling to average pooling in the two convolution layers.

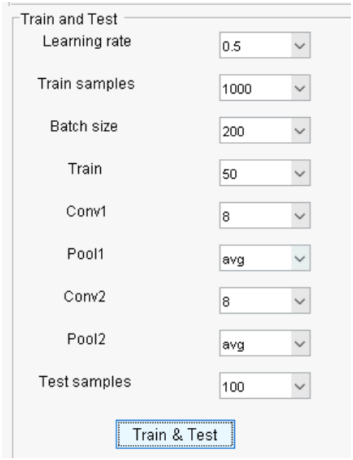

Figure 76. Parameter setting for CNN with average pooling

Compared to the baseline CNN with max pooling, the modified CNN with average pooling displayed a slight decrease in its performance. The final training error of the

CNN with average pooling is 0.5, as shown in Figure 77, which is higher than that of the baseline CNN (around 0.4, shown in Figure 67). The training and testing accuracy also decreased from 0.871 and 0.83 (shown in Figures 68 and 69) to 0.848 and 0.74 (shown in Figures 78 and 79) after changing to average pooling. The reason why max pooling outperforms average pooling may be that max pooling can preserve the most prominent features in a local region, while average pooling may blur the significant features by taking the average value within a region.

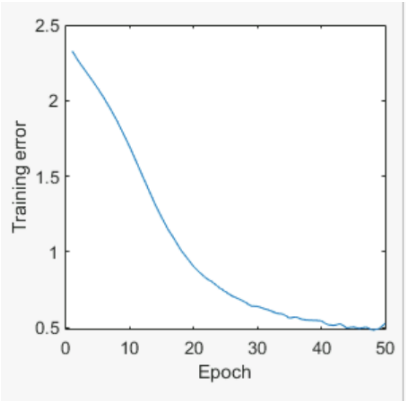

Figure 77. Training error for CNN with average pooling

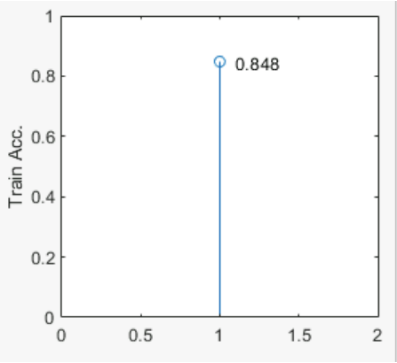

Figure 78. Training accuracy for CNN with average pooling

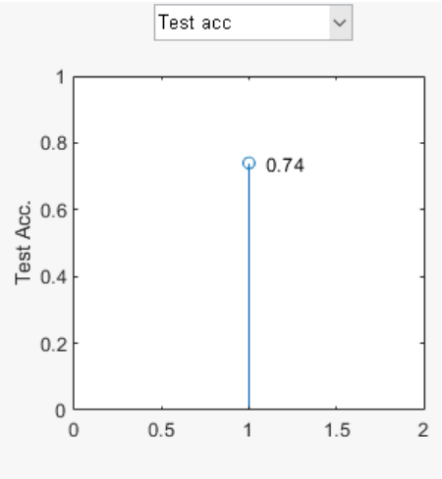

Figure 79. Testing accuracy for CNN with average pooling

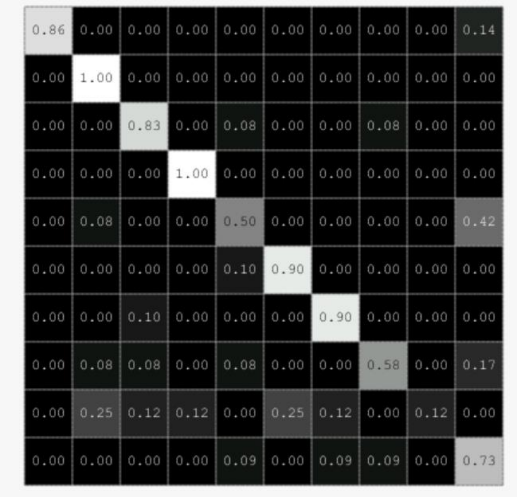

Figure 80. Testing confusion matrix for CNN with average pooling

3c. Improvement

Figure 81 shows the improved model's parameter setting.

Firstly, as proved in **parts 3a and 3b**, the number of channels of 16 is better than 8, and max pooling is better than average pooling. Hence, we chose the number of channels = 16 and max pooling as our model's setting.

Secondly, as illustrated in **part 3a**, the CNN with the number of channels = 16 didn't converge within 50 epochs, which means that we should feed more data to the model to achieve a better performance. Hence, we increased the training samples to 2000 and the number of epochs to 100.

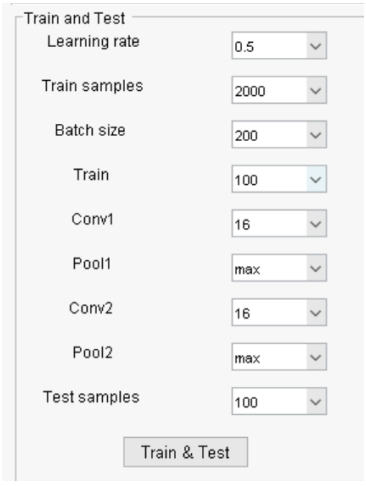

Figure 81. Parameter setting for improved CNN

The performance of the improved CNN greatly outperforms the baseline CNN. As shown in Figure 82, the training error of the improved CNN gradually converged at around 0.1, which is much lower than the loss of the original CNN (0.4 shown in Figure 67). The training and testing accuracy boomed to 0.968 and 0.9, respectively.

According to the confusion matrix shown in Figure 85, the improved CNN can correctly classify almost every figure with an accuracy higher than 80%, except the figure “9”, with an accuracy of 75%.

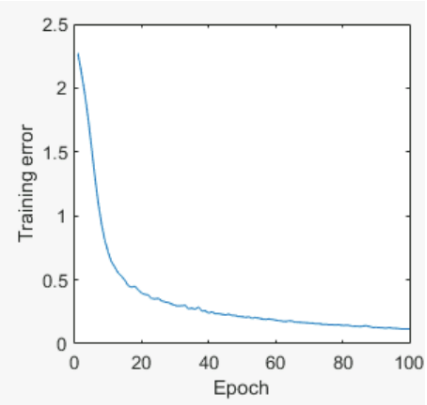

Figure 82. Training error for improved CNN

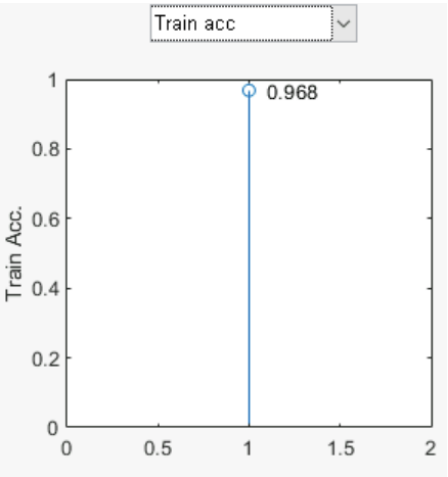

Figure 83. Training accuracy for improved CNN

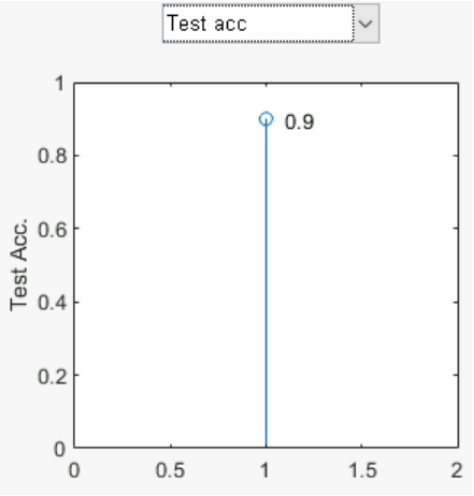

Figure 84. Testing accuracy for improved CNN

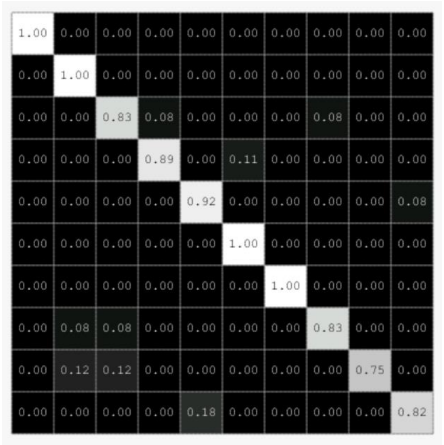

Figure 85. Testing confusion matrix for improved CNN