Problem 1

One layer NN:
batch_size = 64
num_epoch = 25
learning_rate = 0.01
The batch_size 64 is relatively large, with more samples forwarded in each epoch, so the convergence speed is fast. The number of epochs 25 is enough for our model to train and get satisfactory results, with the loss below threshold. The learning rate 0.01 is enough for the model to train at a fast speed and get fairly good performance.

Two layer NN:
batch_size = 64
num_epoch = 25
learning_rate = 0.001
The batch_size 64 is relatively large, with more samples forwarded in each epoch, so the convergence speed is fast. The number of epochs 25 is enough for our model to train and get satisfactory results, with the loss below threshold. The learning rate 0.01 is fairly small, but is enough for the model to train at a fast speed and get fairly good performance.

CNN:
batch_size = 64
num_epoch = 25
learning_rate = 0.001
The batch_size 64 is relatively large, with more samples forwarded in each epoch, so the convergence speed is fast. The number of epochs 25 is enough for our model to train and get satisfactory results, with the loss below threshold. The learning rate 0.01 is fairly small, but is enough for the model to train at a fast speed and get fairly good performance.

Problem 2

My model has a training accuracy 100 per cent and testing accuracy 98.89. The final loss is lower than 0.001

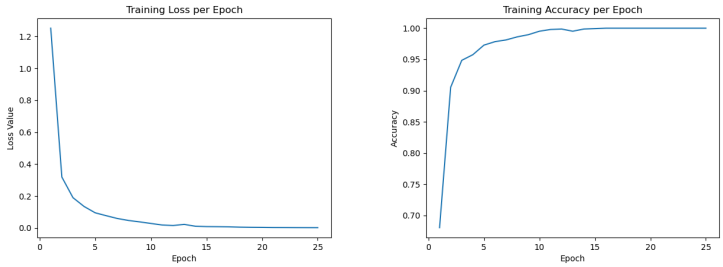

Figure 1: Original performance

Changing the batch size from 64 to 1: The training speed becomes slower, but the model converges faster, getting the training accuracy 100 per cent after only 7 epochs. Overall the performance is slightly better.

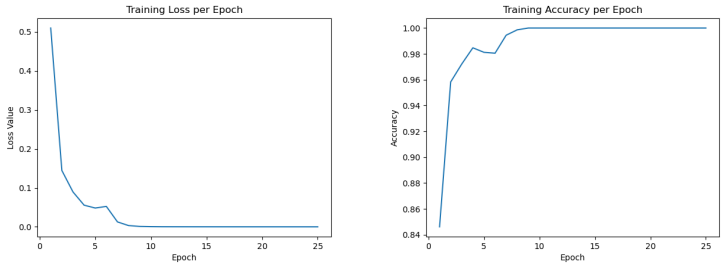

Figure 2: Change batch size to 1

Changing the number of convolution layers from 3 layers (plus 3 activation layers) to 1 layer (with 1 activation layer: Surprisingly, the performance almost does not change at all, with even slightly higher accuracy and a quicker fall in loss. May be the reason is the since MNIST is a really simple dataset, one layer is already enough to reach a good performance.

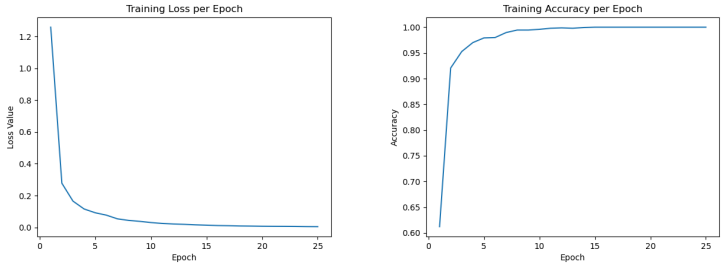

Figure 3: Reduce number of layers

Changing the learning rate from 0.001 to 0.1. This learning rate is much larger, and we can

see clearly that there is a significant drop in performance. The accuracy is much lower, with final training accuracy 92 and testing accuracy 88. Also, we can notice that the curves are not as smooth, meaning that the model converges too fast.

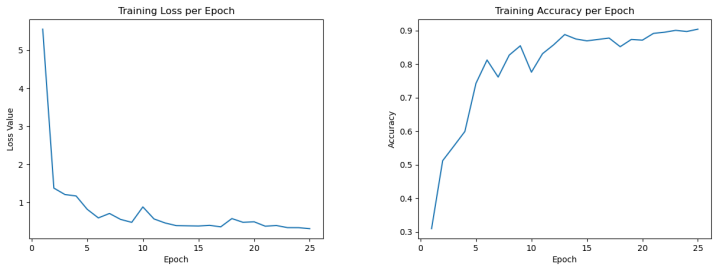

Figure 4: Reduce learning rate

Problem 3

There is a significant difference in performance. It takes much more number of epochs for the model to converge, and in my case, more than 70 epochs to reach training accuracy 100 per cent. However, with 100 per cent training accuracy, the testing accuracy is 29 per cent, with testing loss 4.55. Therefore, the model has a overfitting problem.

1. One possible difference between decision boundaries is that the decision boundaries in the shuffled labels dataset may be more complex and erratic than the non shuffled. This may cause the CNN model to fit to the random shuffling of the labels, rather than the underlying patterns in the data. As a result, the decision boundaries learned by the CNN model may not generalize well to new, unseen data. The CNN model's generalization ability is strongly affected.

2. No, the number of parameters is not the only metric for measuring a deep model's complexity. Other factors that can contribute to model complexity include the type of activation function used, the connectivity patterns between layers, the regularization techniques applied, and the learning rate and optimization algorithm used during training. Just as in the former section, when changing the number of layers, the learning rate etc. can have impact on the performance of the model. The complexity of a model may change when the dataset or task changes. Different datasets or tasks may have different levels of complexity, and therefore, require different levels of model complexity to achieve good performance. For example, a simple model with low capacity may underfit a complex dataset with intricate patterns, while a highly complex model may overfit a simple dataset with few patterns.