Problem 1

For OneLayerNN:
For this class, I tried setting a batch size of 64, set 25 training epochs, and had a learning rate of 0.01. I noticed that, with these hyperparameters, they gave the following losses:
Average Training Loss: 0.544987823203731
Average Testing Loss: 0.6580963614035625

For TwoLayerNN:
For the Two Layer class, I also set a batch size of 64, set 25 training epochs, and a had a learning rate of 0.01. Here I noticed that with these hyperparameters, I had the following losses:
Average Training Loss: 0.567452043507889
Average Testing Loss: 0.5504319769995553
Here, the testing loss was lower than that of the OneLayerNN, and the training loss was slightly higher.

For CNN:
Lastly, for the CNN class, I set a batch size of 64, 20 training epochs, and a learning rate of 0.01. This produced the following losses and accuracies:
Average Training Loss: 0.0605
Average Training Accuracy: 98.5376%
Average Testing Loss: 0.1146
Average Testing Accuracy: 96.3889%

Problem 2

Prior to any changes I made, the model outputted:
Average Training Loss: 0.0605
Average Training Accuracy: 98.5376%
Average Testing Loss: 0.1146
Average Testing Accuracy: 96.3889%

If I lowered the batch size to 30:
Average Training Loss: 0.0165
Average Training Accuracy: 99.9304%
Average Testing Loss: 0.0827
Average Testing Accuracy: 98.3333%

If I increased the batch size to 80:

Average Training Loss: 0.0796
Average Training Accuracy: 98.1198%
Average Testing Loss: 0.1182
Average Testing Accuracy: 97.2222%

Therefore, lowering the batch size increased both training and testing accuracies, as well as decreased training and testing loss. Increasing the batch size decreased training accuracy, increased testing accuracy, and increased both training and testing loss.

If I lowered the number of epochs to 10:
Average Training Loss: 0.1943
Average Training Accuracy: 93.6630%
Average Testing Loss: 0.1853
Average Testing Accuracy: 93.8889%

If I increased the number of epochs to 30:
Average Training Loss: 0.0390
Average Training Accuracy: 99.1643%
Average Testing Loss: 0.0937
Average Testing Accuracy: 98.0556%

Thus, if I decreased the number of epochs, both training and testing accuracy were significantly lowered, and training loss and testing loss were also significantly increased. If I increased the number of epochs, both training accuracy and testing accuracy were increased, and both training and testing loss were decreased.

If I lowered the learning rate to 0.001:
Average Training Loss: 0.3372
Average Training Accuracy: 92.2702%
Average Testing Loss: 0.3194
Average Testing Accuracy: 92.5000%

If I increased the learning rate to 0.1:
Average Training Loss: 0.0012
Average Training Accuracy: 100.0000%
Average Testing Loss: 0.0753
Average Testing Accuracy: 98.3333%

Therefore, if I lowered the learning rate, my training and testing accuracies were lowered, and my training and testing losses were both increased significantly. However, interestingly,

I noticed that increasing the learning rate drastically decreased my training loss, slightly lowered my testing loss, and increased my training accuracy to 100 percent somehow, while also increasing my testing accuracy by a good amount too.

Problem 3

1. After running the CNN model with 280 epochs and shuffled training labels, I got the following losses and accuracies:
Average Training Loss: 0.0402
Average Training Accuracy: 100.0000%
Average Testing Loss: 4.6933
Average Testing Accuracy: 25.8333%

As seen with the results above, having shuffled training labels reduced training loss and significantly increased training accuracy, but, at the same time, increased testing loss, and also greatly decreased testing accuracy. When comparing models with true training labels, and shuffled training labels, when we shuffle the training labels, the model is forced to learn a boundary that is more complex. Hence, increasing the complexity of the dataset makes the model require a more complex boundary, which in turn decreases the generalization ability of the model, given that the boundaries it had learned are no longer as applicable.

2. No, I do not think the number of parameters is the only metric for measuring the complexity of a deep model. There are certain datasets that are very complex, and therefore, require complex boundaries. Or, in addition, there are also certain tasks which can also require much more complex boundaries. Thus, dataset or task complexity both contribute to a model's complexity. However, if the dataset (or task) is changed to be less complex, then the complexity in the bias-complexity tradeoff changes given that, the less complex a dataset becomes, the less complex the boundaries it requires are. Therefore, decreasing dataset complexity simultaneously decreases model complexity, whereas increasing its complexity increases model complexity.