Young, James Yang

20740589

ELEC4840 Assignment 1 Report

Problem 1

For problem 1, we were tasked with constructing a four layer linear network with dropout using Pytorch. For my hyper-parameters, I used the cross-entropy loss function, SGD as the optimizer, and a learning rate of 0.01 for 10 epochs. These hyper-parameters were arbitrarily chosen based on the Pytorch guide and tutorial slides. Figure 1 below shows the results of the loss curve, training accuracy, and validation accuracy curves with dropout probability of 0.3 in training. When only removing dropout and keeping the other parameters, the result of the loss curve, training accuracy, and validation accuracy curves are shown in Figure 2.

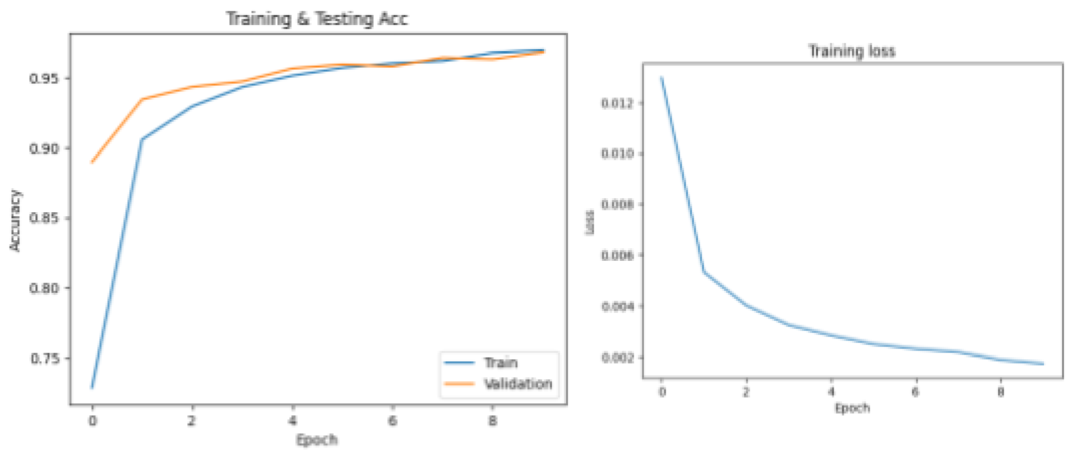

Figure 1: Loss and accuracy plots (with dropout)

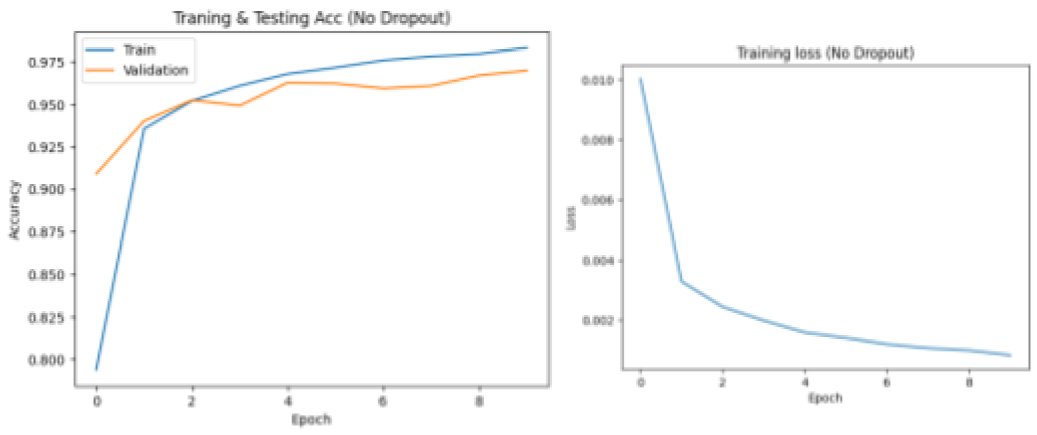

Figure 2: Loss and accuracy plots (without dropout)

As seen in the two figures, adding dropout decreases the training accuracy but increases the validation accuracy. This is expected as adding dropout randomly drops out a fraction of the neurons during training, hence preventing the network from relying too heavily on any particular set of features. This helps prevent overfitting, meaning the model will tend to do better on unseen data, leading to improved validation accuracy.

Problem 2

For problem 2, our task was to implement the LeNet convolutional neural network model in Pytorch, train the model on the EMNIST dataset, and plot the results of the accuracy and loss curve. For implementing the model I used the Pytorch LeNet implementation in their documentation for reference and made adjustments to fit the EMNIST dataset as the EMNIST dataset has 47 classes so my output had to match that.

For training the model the process was similar to problem 1 with the exception that in this problem we had to try different batch sizes, learning rate, and optimizers. I decided to try out learning rates of 1e-1, 1e-2, 1e-3, 1e-7, the SGD and Adagrad optimizers, and batch sizes of 64 and 128 to see how tuning these hyper-parameters would affect the results. These were all run with 10 epochs. Figure 3, 4, and 5 below shows the results I obtained in my experiment with these different hyper-parameters.

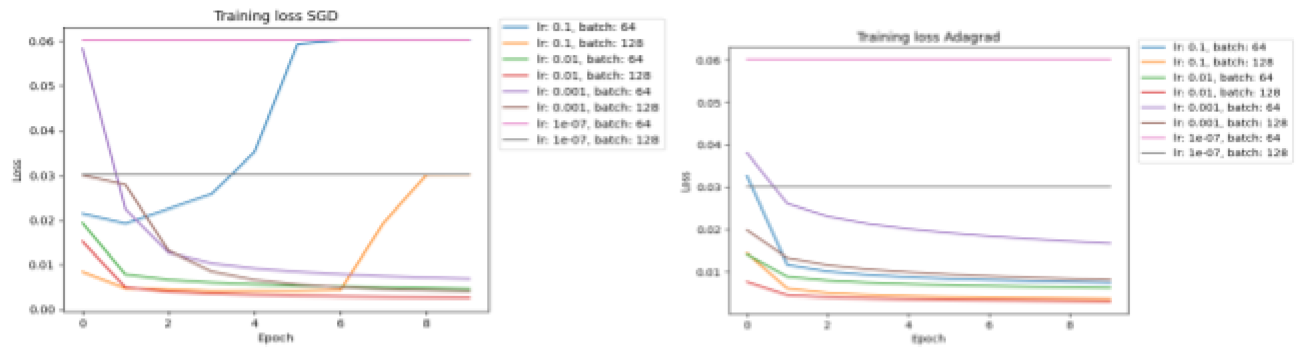

Figure 3: Loss Curves

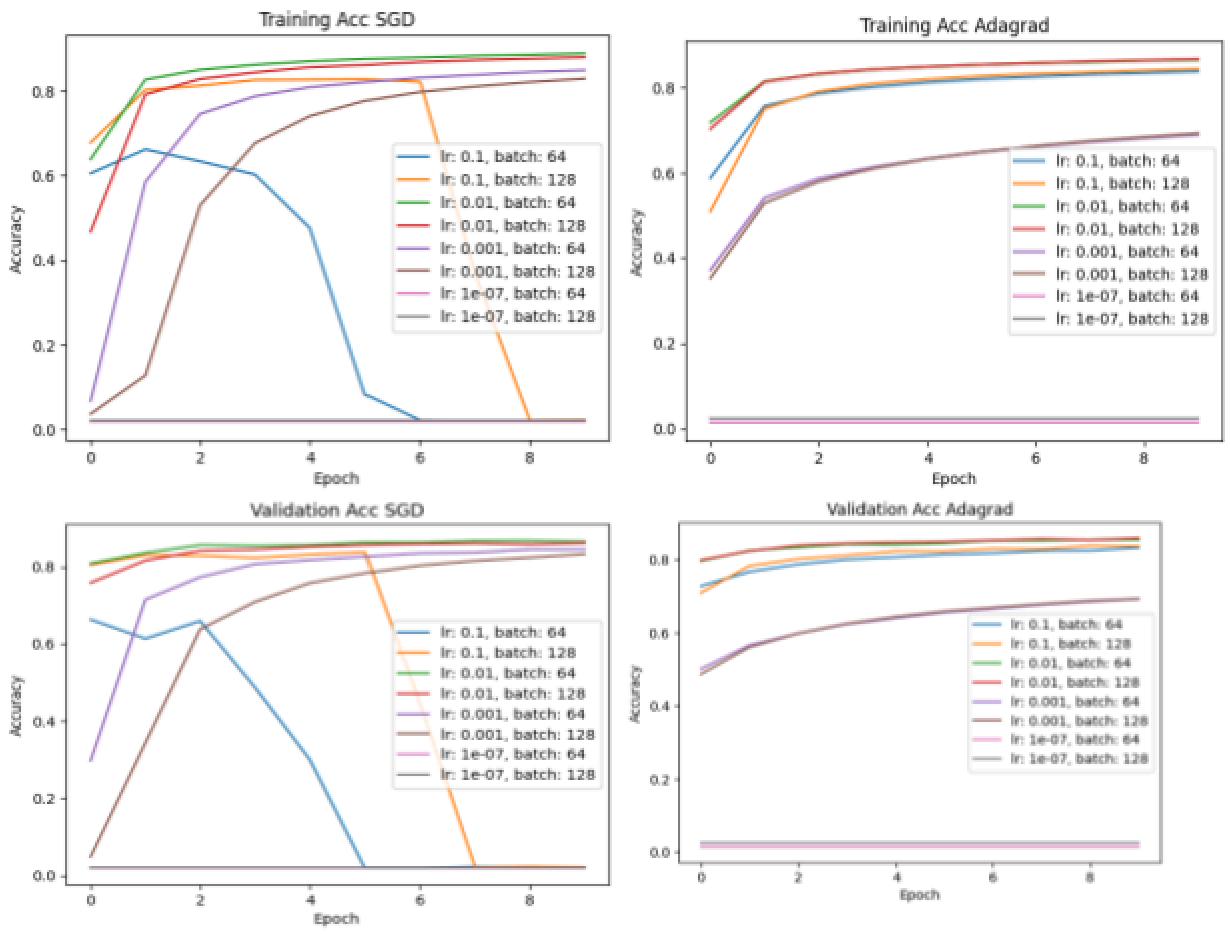

Figure 4: Accuracy Curves

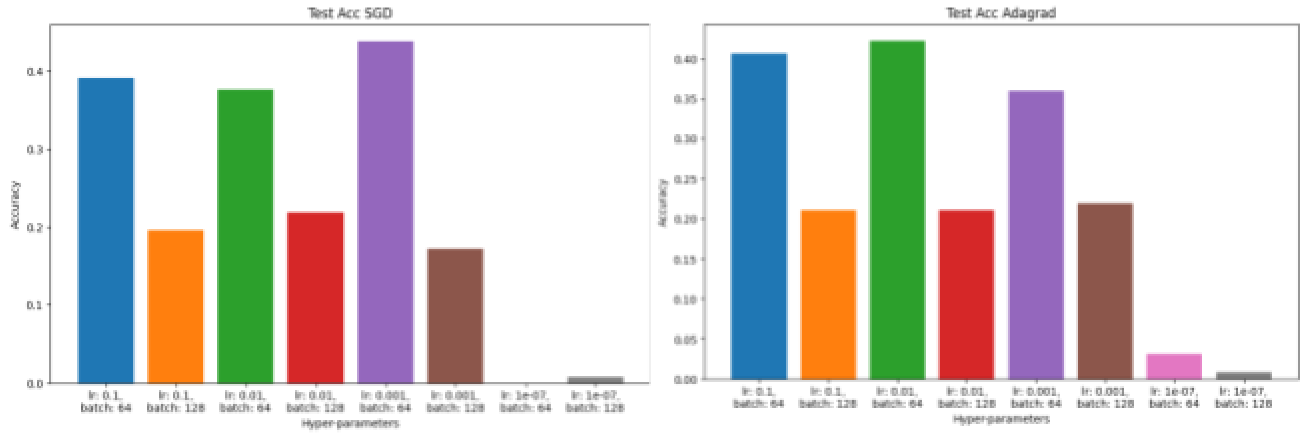

Figure 5: Testing Accuracies

With a larger learning rate, the loss during training may initially decrease rapidly due to larger updates in the parameter space. However, if the learning rate is too large, it can cause the optimization process to become unstable. The SGD figures above show this as seen in the loss curves for the learning rate 0.1, where in the loss curve it goes down before shooting back up. However, if the learning rate is too low, the model will take longer to converge as shown in the figure above where the learning rates set at 1e-7 have a flat loss curve.

For batch size, a larger batch size calculates a much better estimate of the true gradient, meaning it will lead to faster convergence. This behaviour is demonstrated in the figures above, where batch sizes of 128 tend to have lower loss than batch sizes of 64 in my testing. However, larger batch sizes tend to lead to lower test accuracy as larger batches tend to provide less diverse information about the dataset making it prone to overfitting.

For the optimizers, there are some noticeable differences in the loss curves. In the figures above, the experiment shows that the loss curves for Adagrad tend to be smoother especially for higher learning rate. This is because Adagrad is adaptive and adjusts the learning rate whereas SGD with momentum uses a fixed learning rate.

Overall, the best hyper-parameters based on the options I tested were a learning rate of 1e-2 with a batch size of 64. The optimizers were mostly similar for matching learning rates and batch sizes. The highest testing accuracy achieved was 43% the SGD optimizer with batch size 64 and learning rate 1e-2, which isn’t great but is most likely due to the experiments running only 10 epochs. More hyper-parameter tuning and increasing the number of epochs would help increase the testing accuracy.