CS 499/579

Homework 1

Gavin Lesher

(lesherg@oregonstate.edu)

Task 1

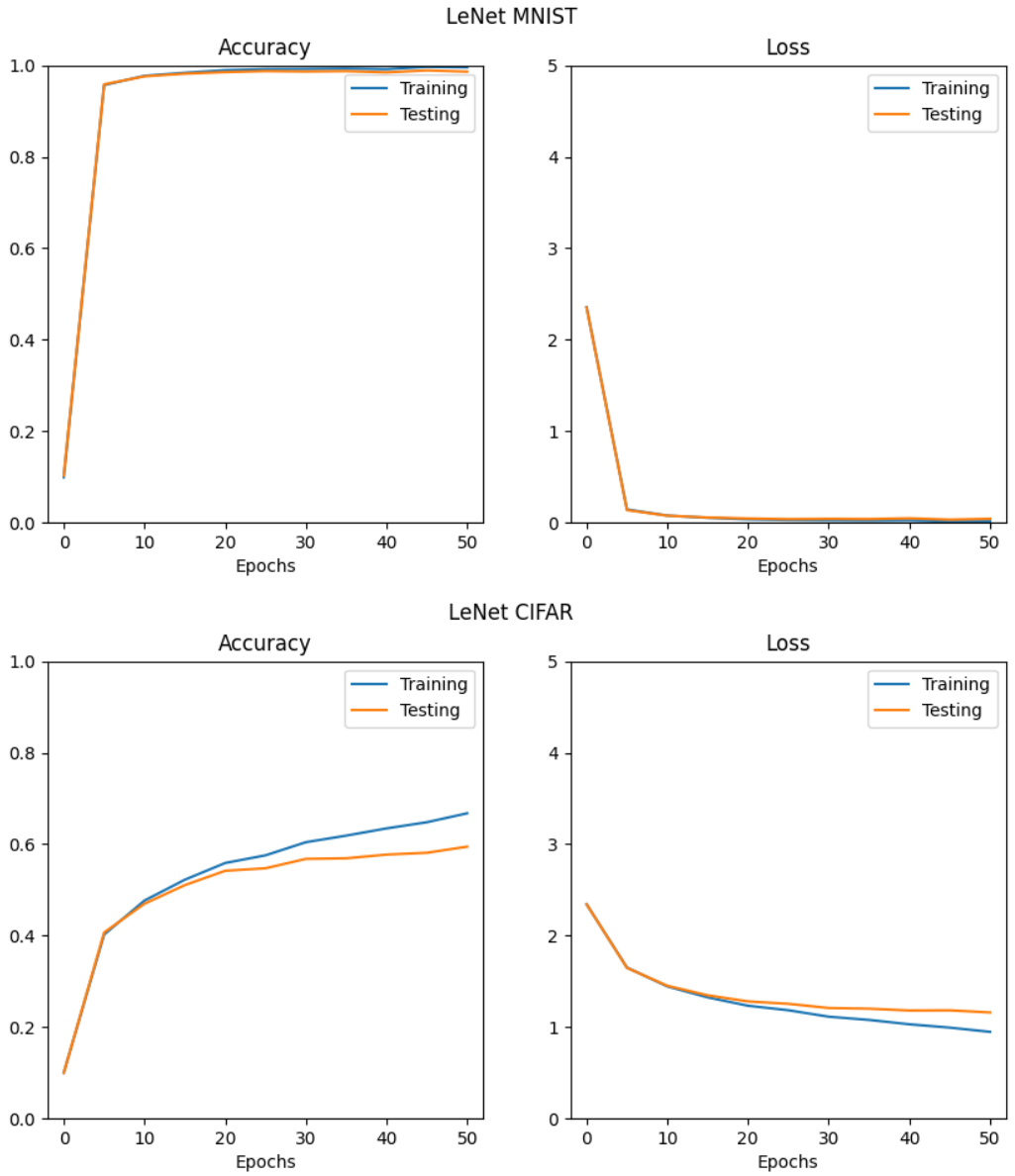
Experimental Setup

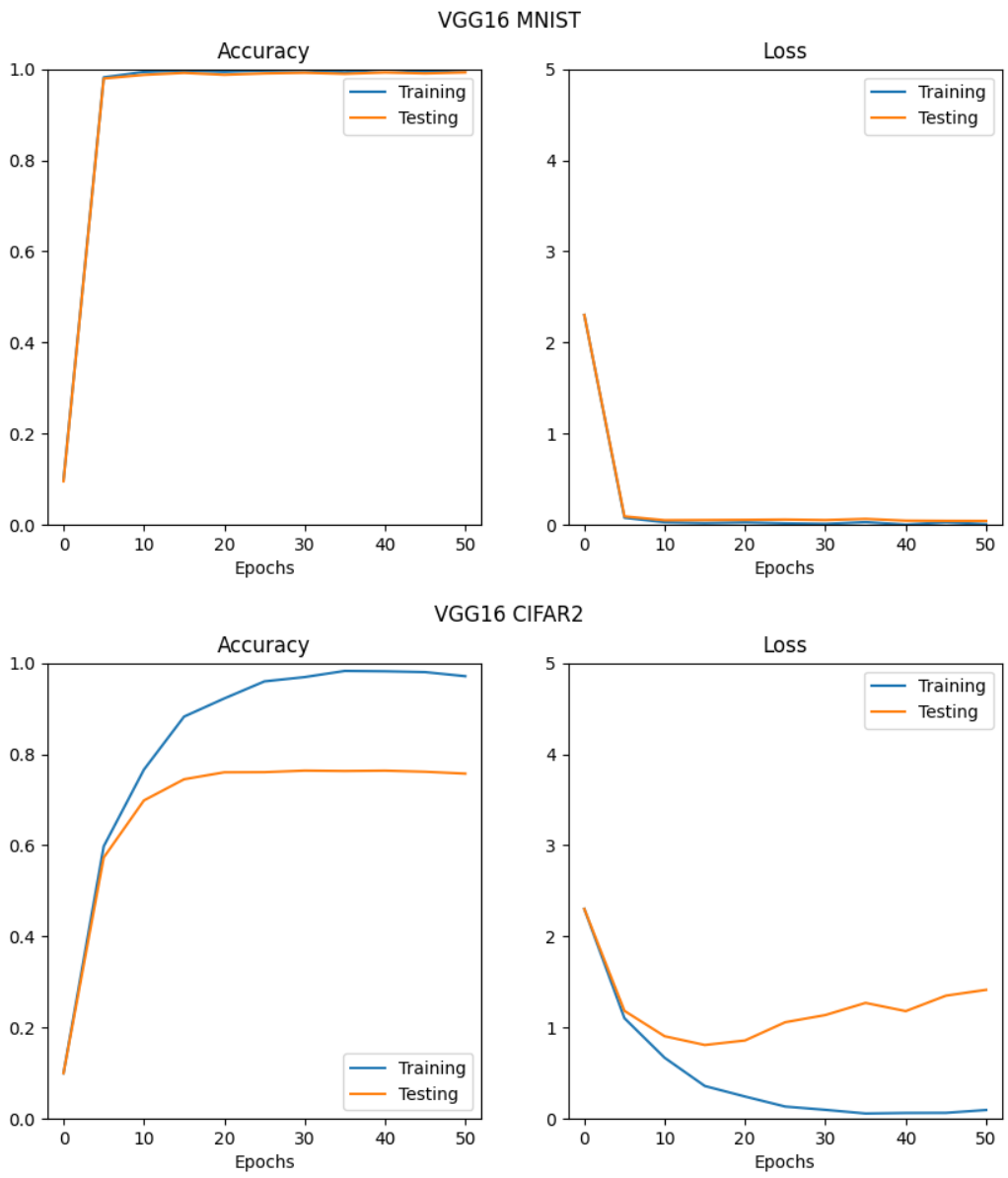
As this was my first adventure into using existing machine learning frameworks, I picked standard values for most every hyperparameter, as well as including little pre-processing to the inputs. I used the MNIST and CIFAR datasets found within the `torchvision` package, with the only pre-processing changes consisting of shuffling the data via the `shuffle` parameter when using the `DataLoader` class within pytorch.

As for the hyperparameters: I fixed the number epochs across the entire assignment to 50, calculated accuracy and loss every 5 epochs, used a batch size of 200, used a learning rate of 0.001, and used the Adam optimizer. Finally, I used cross-entropy loss to compute all losses seen in the assignment.

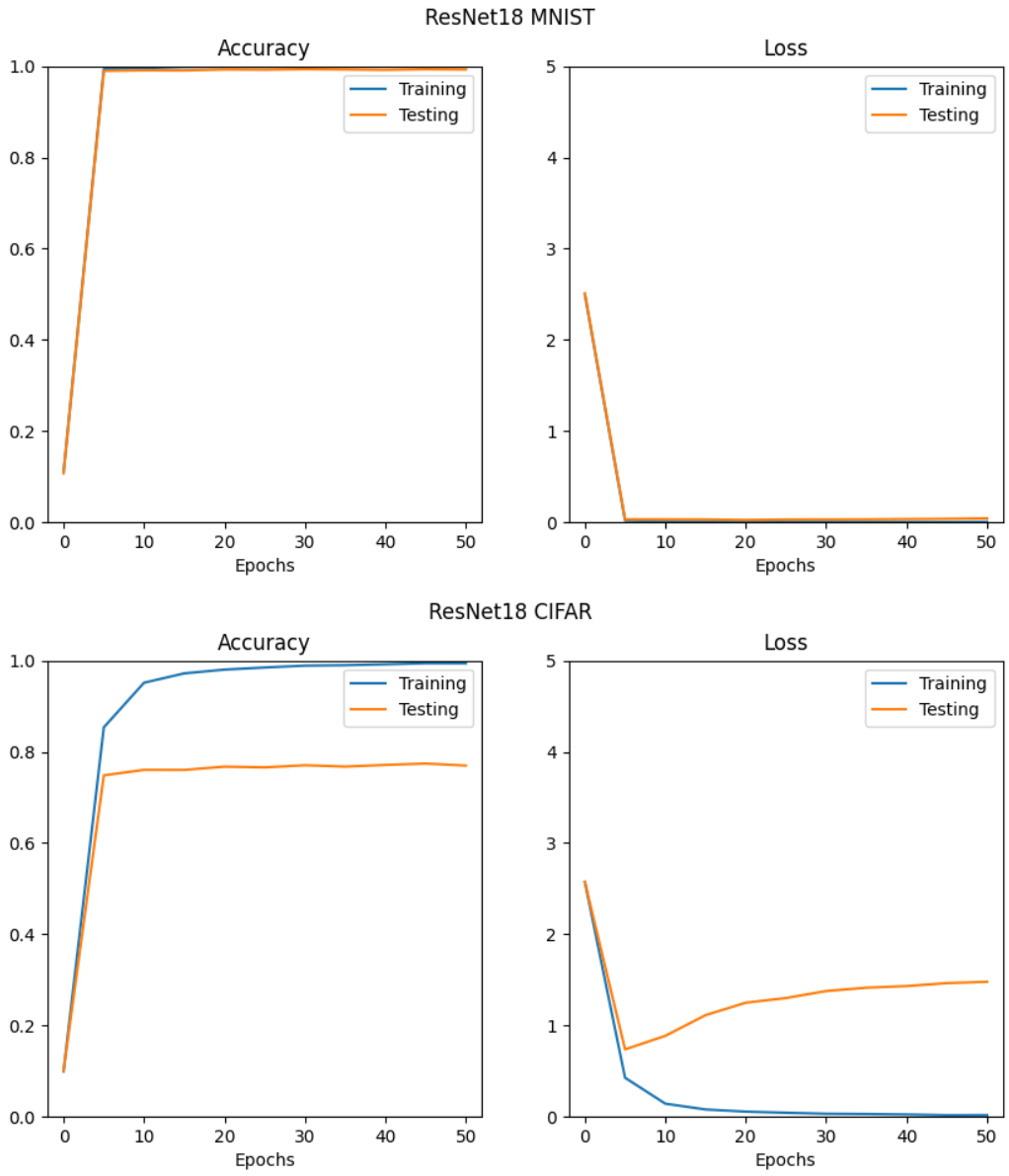
Task 1 continued on next page. . .

Resulting Plots





Task 1 continued on next page. . .



Task 1 continued on next page. . .

Analysis

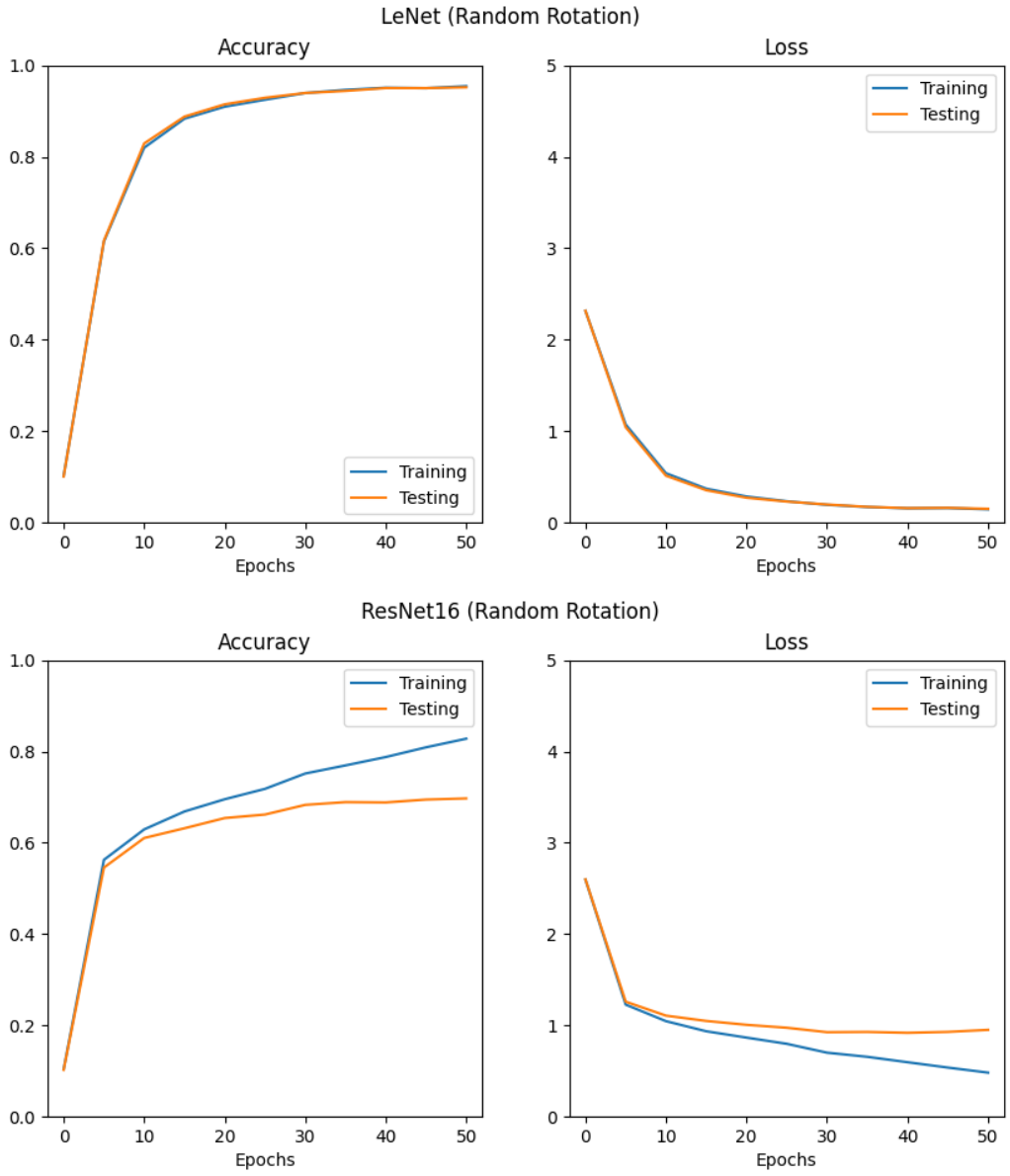
In general, my models tend to overfit the data by a significant margin. On all tests done to the MNIST dataset, the model was able to achieve close to 100% training and testing accuracy within the 5 epoch interval of data collection so there is little insight to gleam from the resulting graphs. For the CIFAR dataset, The LeNet model preformed the best in regards to overfitting, having a small gap between training and testing accuracies and loss, despite the model's final accuracy of about 60%. For VGG16 and ResNet18, both models quickly reached high training accuracies, yet testing accuracy plateaus about 25% less than the final training accuracies, and an increasing testing loss. This overfitting by both models suggests an issue within the training data, which makes sense considering the distinct lack of pre-processing I did to my training sets. I would like to explore the addition of proper pre-processing including better random shuffles and cropping, however unfortunately I do not have enough time to design and implement those trials.

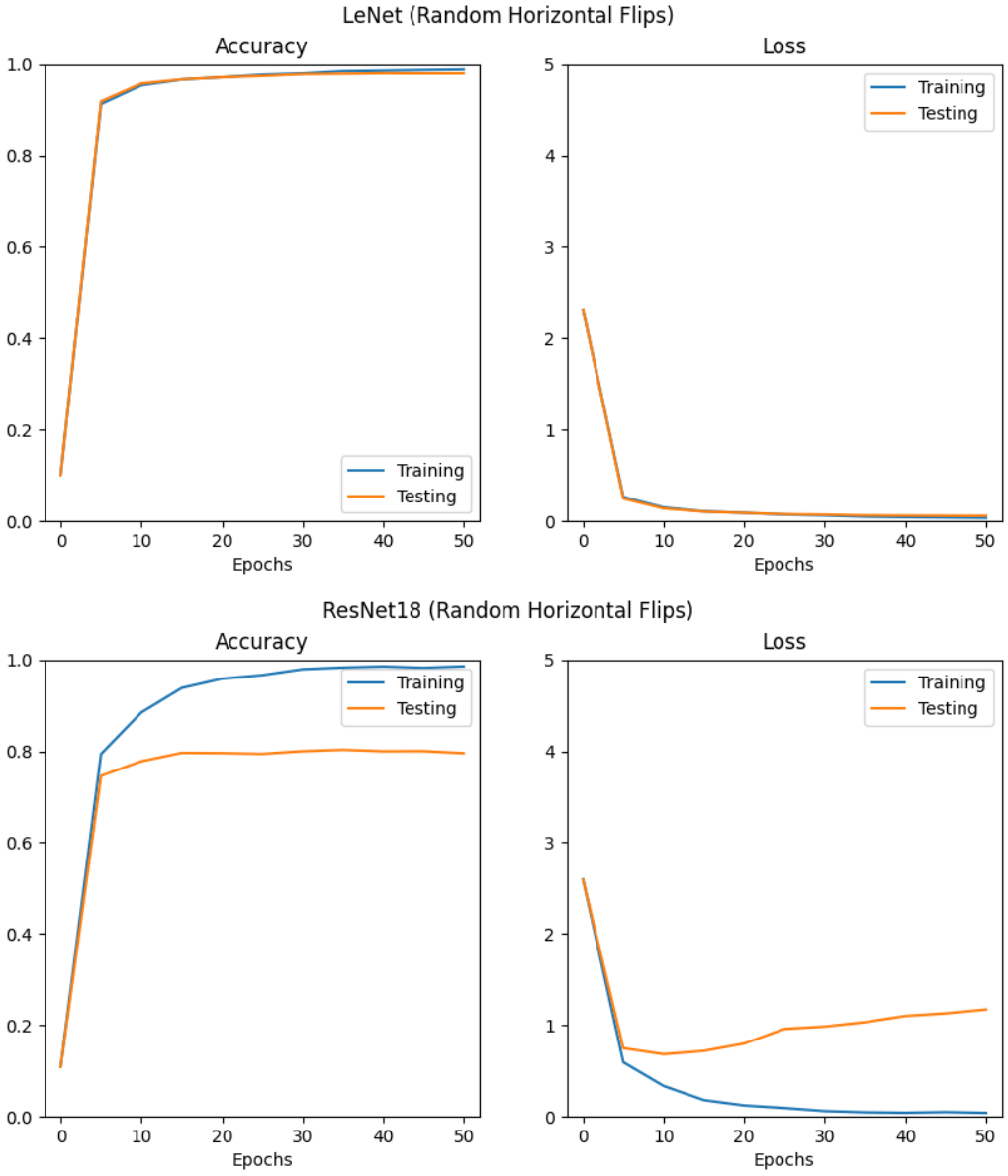
Task 2

Experimental Setup

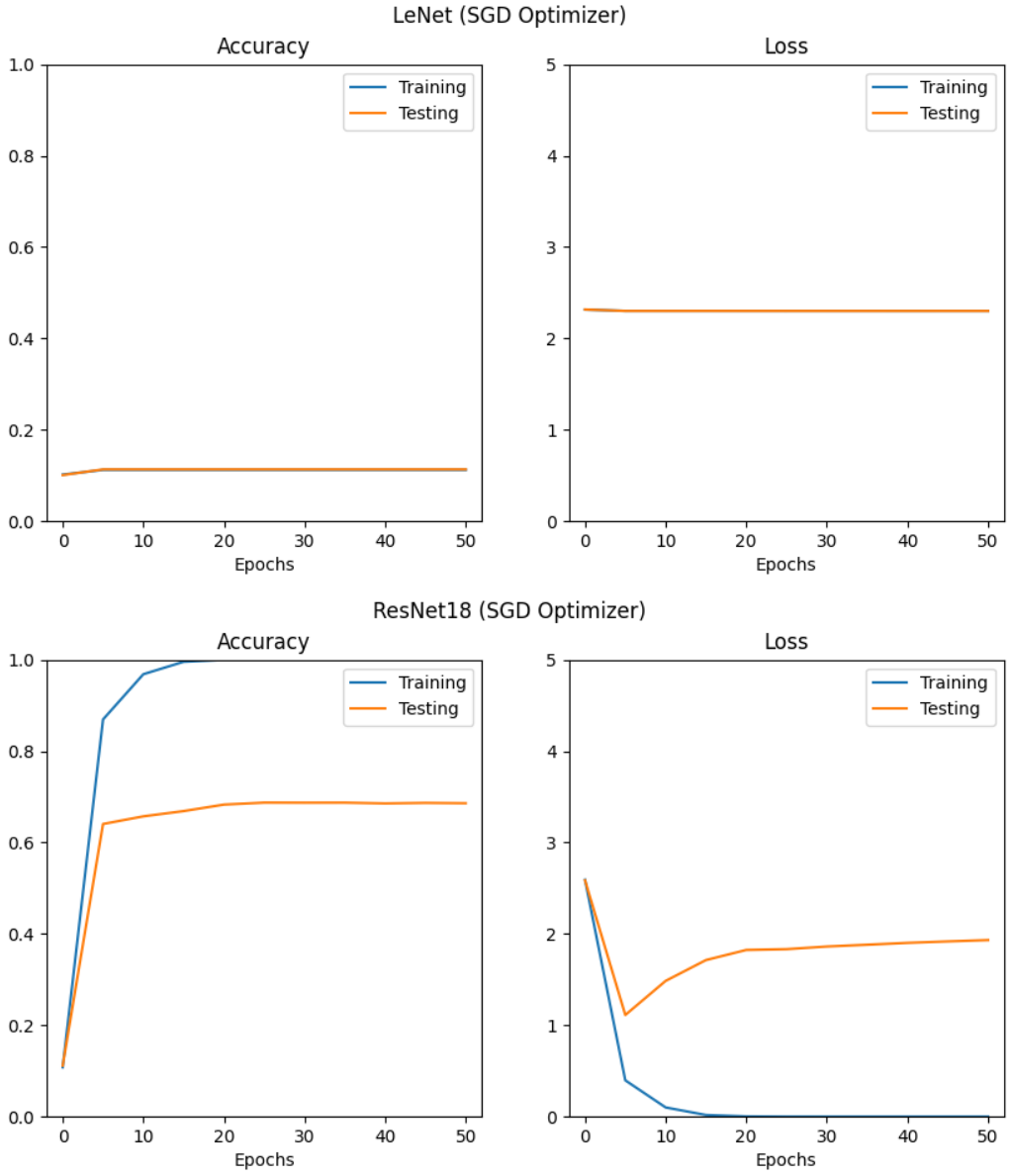
For the experiments preformed, all hyperparameters were set at the defaults mentioned in task 1, other than the one being tested. For the rotational trials, pytorch's `RandomRotate` transform was used to randomly rotate the images between 0 and 359 degrees. For the Horizontal flip pytorch's `RandomHorizontalFlip` transform was used to randomly flip images. When swapping the optimization algorithm to SGD, a momentum of 0.9 was used. Finally, the batch size was increased to 1000, and the learning rate was decreased to 0.00001.

Resulting Graphs

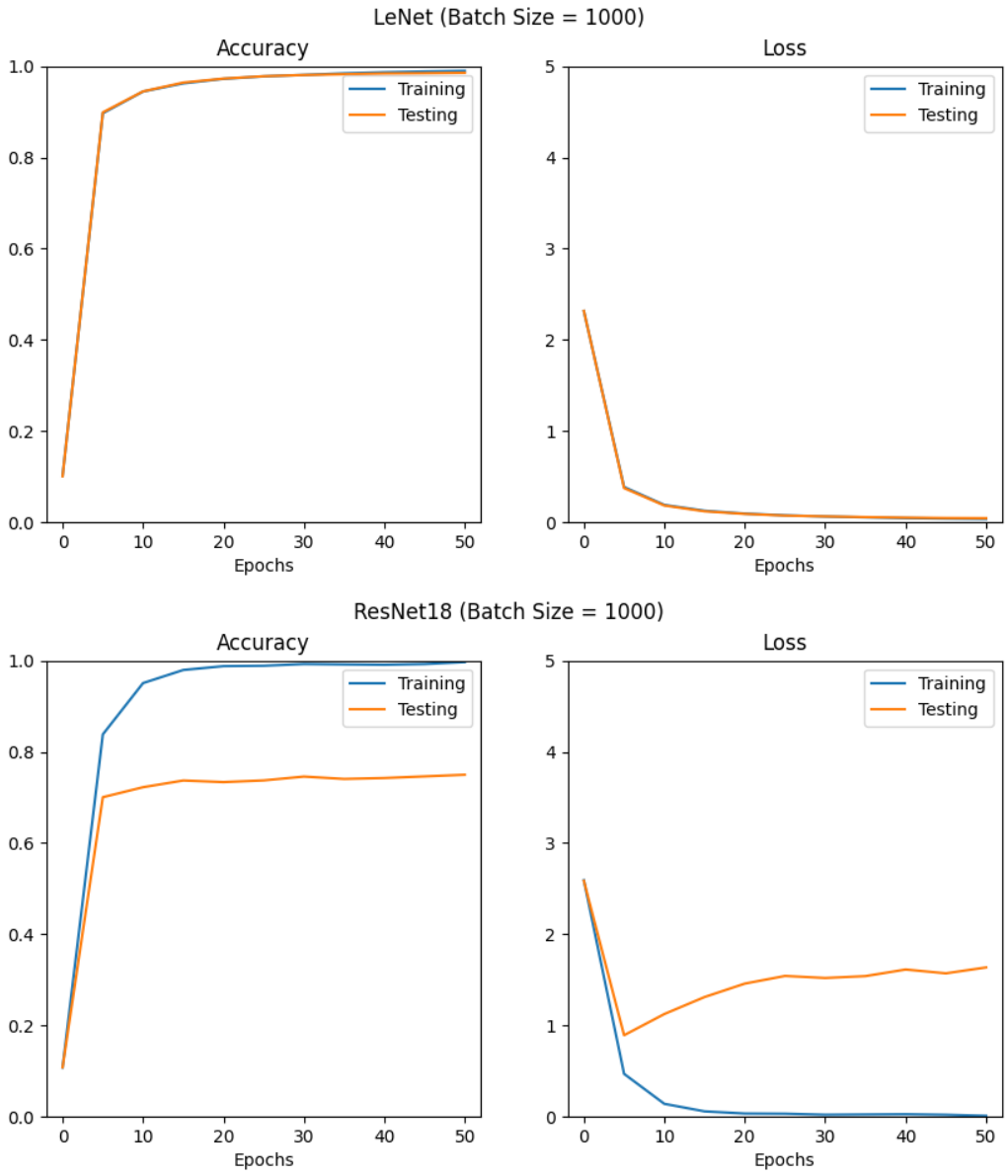




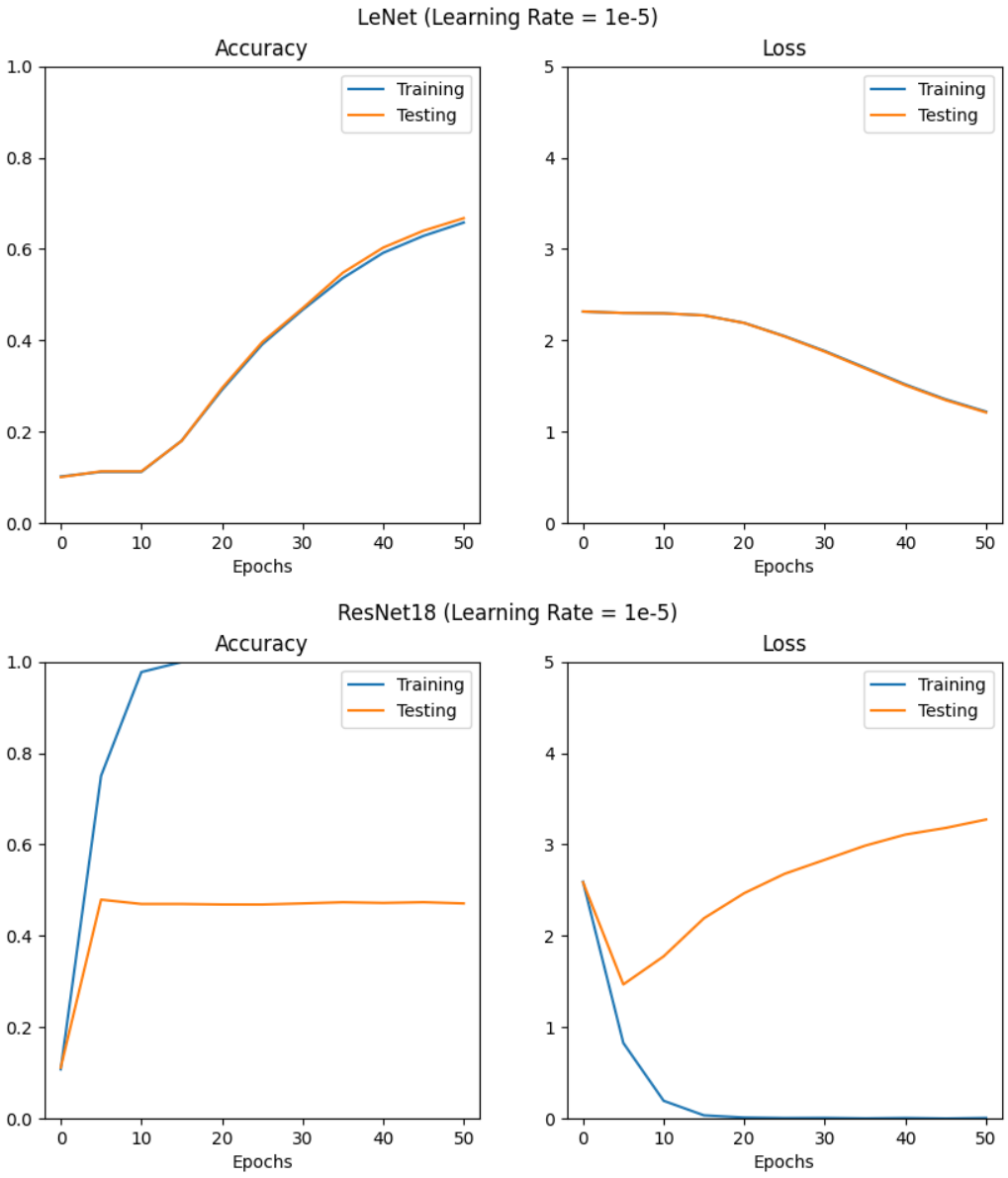
Task 2 continued on next page. . .



Task 2 continued on next page. . .



Task 2 continued on next page. . .



Analysis

Overall, the expirements preformed exhibit the same issues the base models showed. Analyzing the ResNet18 models first, with the exception of random rotation, the ResNet18 models preformed about the same as the respective base models, and continued to appear to overfit the data. An interesting result of the reduced learning rate and change in optimizer is an increased overfitting rate, with the models reaching an observed 100% accuracy on training data. These results from the ResNet18 further indicate that the training data I fed the models is not adequate to train a reliable classifier. An alternate, perhaps more likely possibility is that there are unforeseen errors in my code that led to this overfitting. Further investigation is needed into these issues, and unfortunately I did not have enough time to do so within the deadline of this assignment. As for the random rotation

Task 2 continued on next page. . .

experiment preformed on the ResNet18 model, it showed small signs of overclassification, with the desparity between the training and testing accuracy and loss increasing in distance as the epochs progressed. Because the training accuracy and loss do not plateau within the 50 epochs, it is evident more training is needed to accurately make a conclusion.

The LeNet models produced more reliable results, with the exception of the optimizer change. Handling the exception first, the optimizer change produced no reliable results for an unknown reason, although it's likely to be an error within my code. Same as before, I unfortunately do not have the time to investigate why this is the case, and I apologize for the missing data. The random rotations and horizontal flips prevented the model to train as fast, which is to be expected as the model is seeing more variation to the data, and thus needs to see more data points to create a robust model. The increased batch size produces a similar result, though the reasoning I expect to differ because with a higher batch size the model updates its parameters less frequently per epoch, resulting in slower learning. Finally, the learning rate decrease is significant enough to stunt the model's growth and slow it down significantly, with the 50 epochs allowed not able to fully train the model.