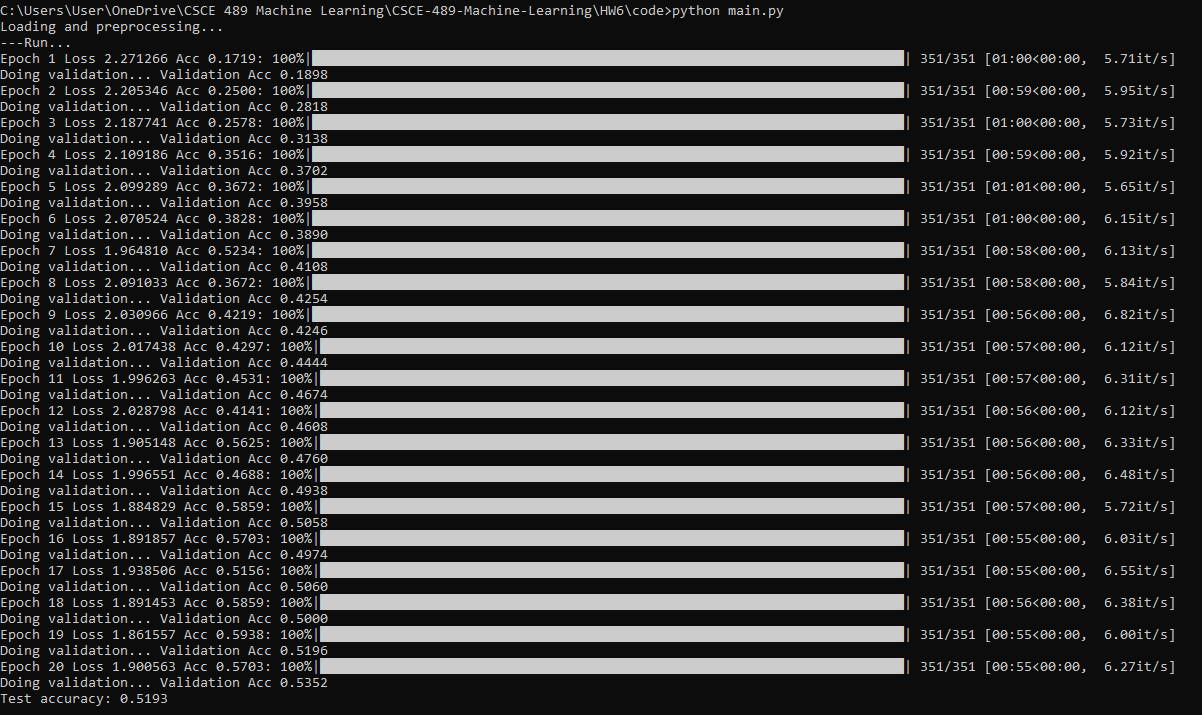
1. After the first 3x3 convolution, we have parameters. The second 3x3 convolution gives us parameters, summing up to 1474560 parameters.
2. Problem 2:
   1. . The element-wise square (for our variance) is with variance vector . We then just take the square root since we have no bias, which gives us . Lastly, we perform element-wise division with the vectors of the matrix , yielding
   2. is derived from element-wise computation of

.

* 1. When we train the network, our means and variances are learned through training the network (we calculate the mean and variance for each batch), and the and parameters are learned as well. When we test the network, the mean and variance are fixed, estimated using the calculated means and variances.
  2. Because the parameters/weights are already learned at this point, batch size has no effect.

1. Program Output

The random initialization of weights leads to inherent variance in the LeNet’s performance. Some runs did have slight overfitting near the end of the twenty epochs (e.g. Epoch 18-20), but this instance here has no overfitting because the validation accuracy at the end is the highest out of all twenty epochs. We also observe a relatively low accuracy compared to previous machine learning models (SVM, logistic regression, linear regression) in our earlier homeworks. This may be because the images themselves are quite small, so there is a small amount of data on each sample and we don’t have much data to begin with. We don’t have any regularization, so the only safeguard against overfitting is our network architecture. Another way to possibly increase accuracy is to use a pretrained convnet and then train your own classifier.