Project 3 Report

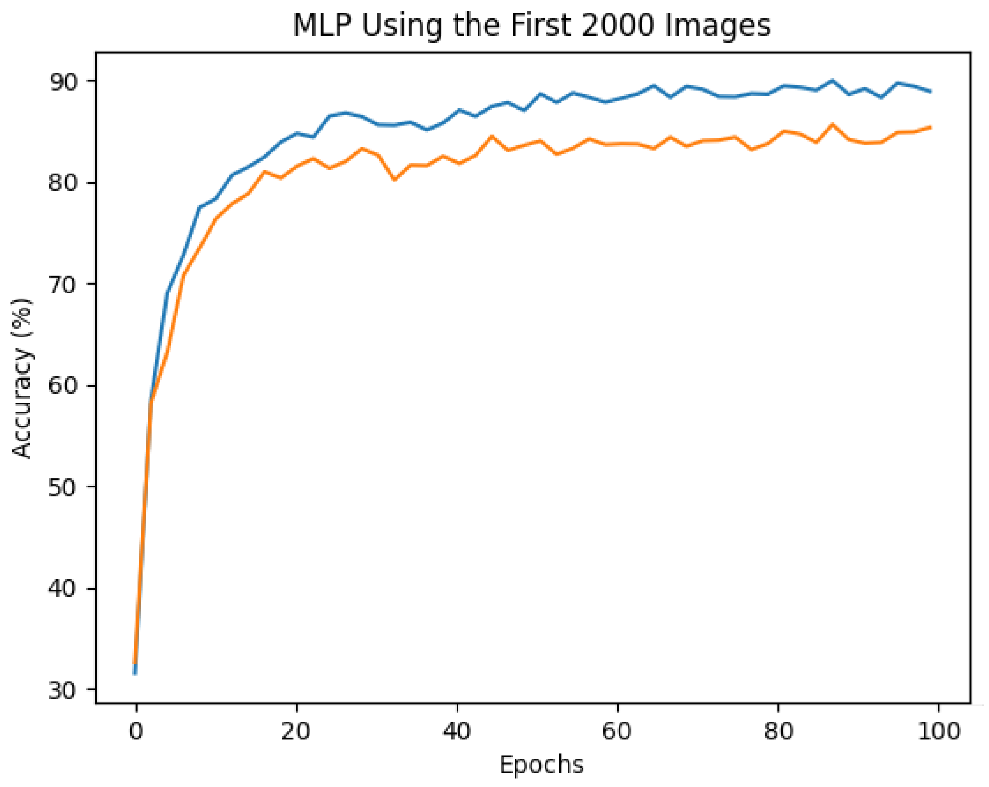
Multi-Layer Perceptrons

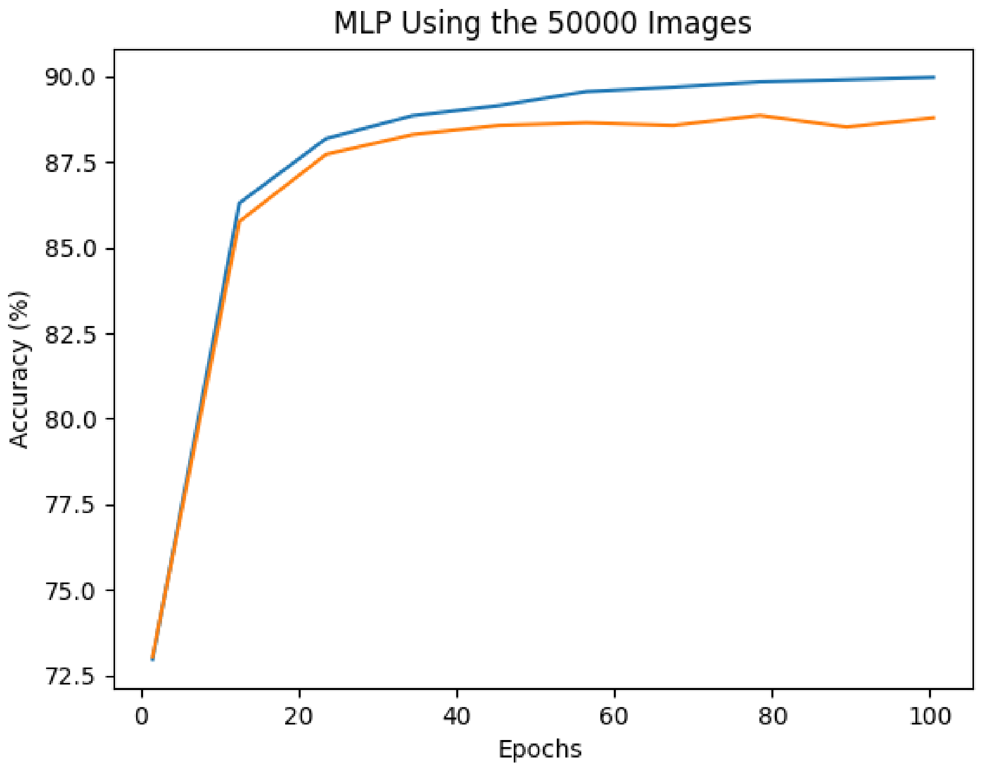
General overview of the code:

For each epoch, an index array is shuffled, which is used to create batches of random labels and images from the dataset. In each batch of 256 images and labels, the images are individually fed into the network. The label and outputs are converted to one-hot vectors to determine the accuracy. Using the label and output, the gradient is then computed through backpropagation and added to the gradient estimate for the batch. At the end of each batch, the weights and biases are updated according to the gradient and learning rate, and the gradient estimate is reset.

Overfitting

Using the first 2000 images did not result in much overfitting. With the first 2000 images of the training set, the neural network ended at around 88% accuracy for the training set and around 85% for the validation set. 88% accuracy for the training set is too low for the network to be overfitting. On the other hand, using all 50000 images resulted in about 90% accuracy for the training set and about 88% for the validation set.


MLP Using the First 2000 Images
Accuracy (%)
Epochs
90
80
70
60
50
40
30
0
20
40
60
80
100


MLP Using the 50000 Images
Accuracy (%)
Epochs
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
0
20
40
60
80
100

Testing

To improve the accuracy, the network used a learning rate schedule. The learning rate schedule was a step decay schedule that halved the learning rate every five epochs. The result was a testing accuracy of 90.02%.

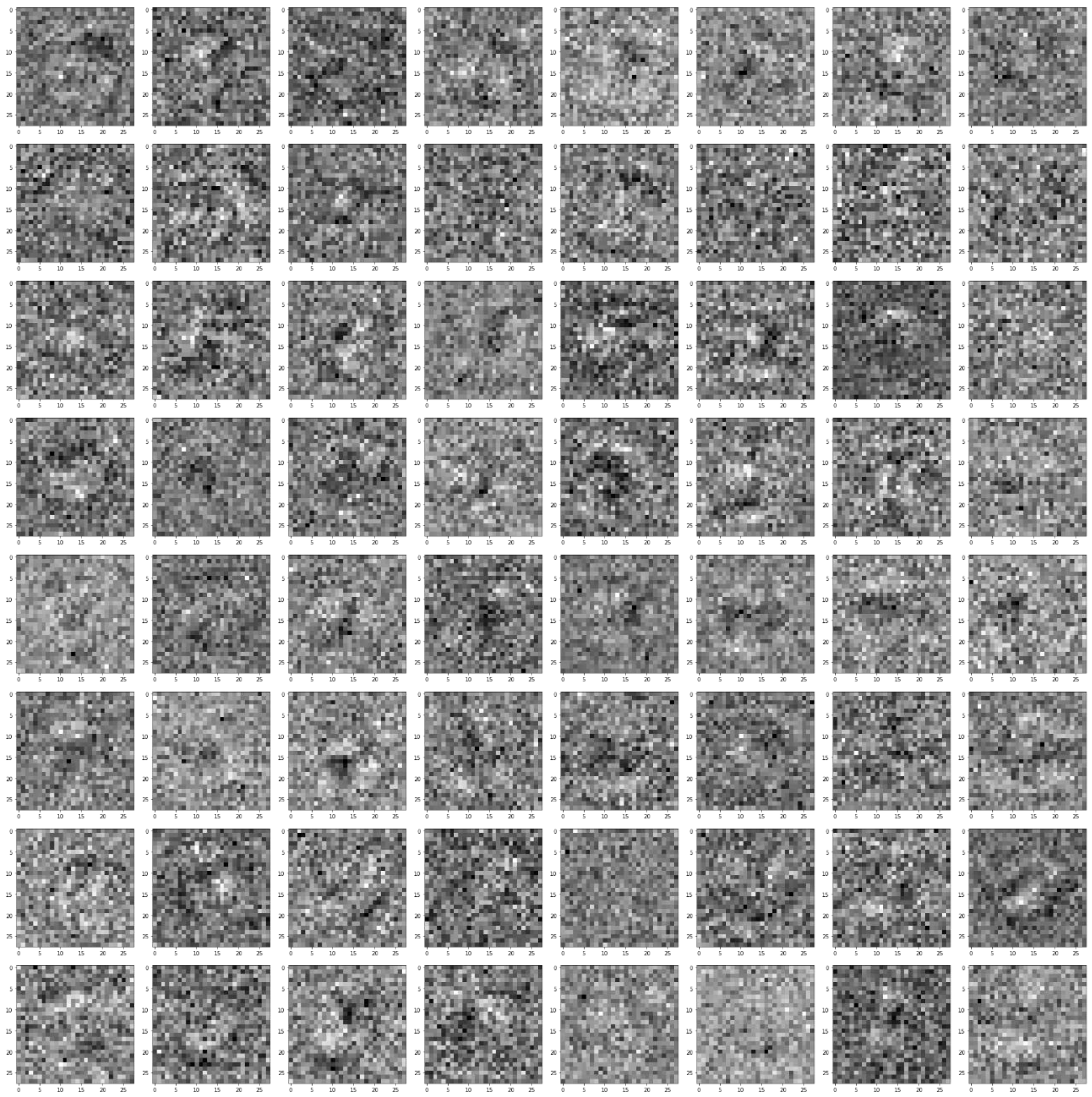
Confusion Matrix

```
                          True Classification
               [[429.   0.   6.   1.   0.   6.   6.   1.   2.   3.]
               [  0. 555.   2.   5.   1.   9.   0.   6.  11.   1.]
               [  2.   3. 443.  18.   2.   3.   4.   8.   5.   1.]
               [  3.   1.   6. 465.   0.  17.   1.   1.  17.   6.]
Predicted      [  0.   0.   7.   1. 440.   8.   7.   9.   4.  19.]
Classification [  6.   3.   0.  36.   1. 370.   9.   2.  10.   3.]
               [  3.   2.  12.   2.   6.   7. 494.   0.   8.   0.]
               [  1.   2.   9.   7.   0.   2.   0. 470.   3.  16.]
               [  3.  12.  14.   5.   3.  20.   3.   1. 418.   8.]
               [  0.   1.   2.   3.  34.   2.   0.  19.   6. 417.]]
```

The number 5 was the hardest to identify, with only 370 correct classifications. This is understandable because the number 5 is similar to other numbers. With just a slight change of one or two strokes, a 5 can look like an 8 or a 3. Thus, 3 and 8 made up a large portion of the incorrect classifications. Numbers 8 and 9 were also hard to identify. The number 9 was frequently confused with 7 and 4, which are both very similar in terms of shape.

Weight matrix visualization

From all of the visualizations, there are definitely distinct shapes among all the noise. By looking at the images more closely, it is possible to see the shapes of numbers in certain images. For example, the image that is two from the right and from three from the top clearly looks like a three. The image that is directly left to it looks like a two. The images help visualize what each row is doing, and how it is template matching.



Convolutional Neural Networks

General overview of the code:

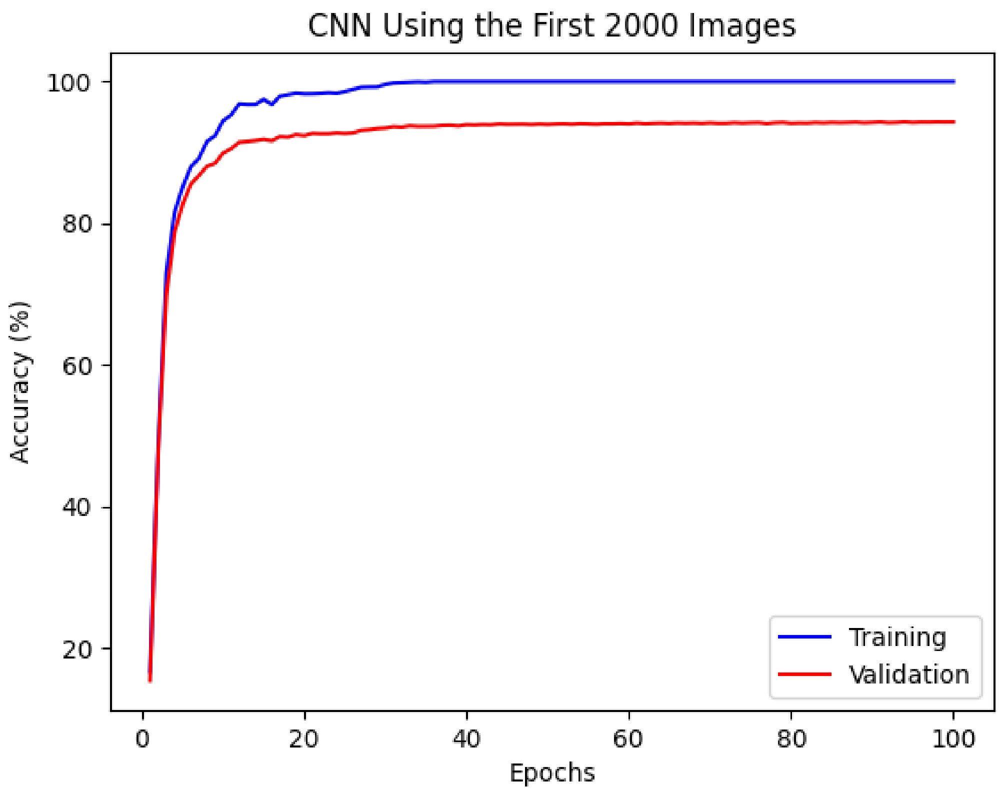
For each epoch, an index array is shuffled to create random batches. For every batch of 256 images and labels, the images are inputted into the PyTorch convolutional neural network. The outputs are then used to determine the loss and the accuracy of the batch. The accuracy is computed by converting the label and output into arrays of one-hot vectors, comparing them, and summing each row. If the sum of each row is 10, the one-hot vectors of the label and the output are the same, meaning the output is correct.

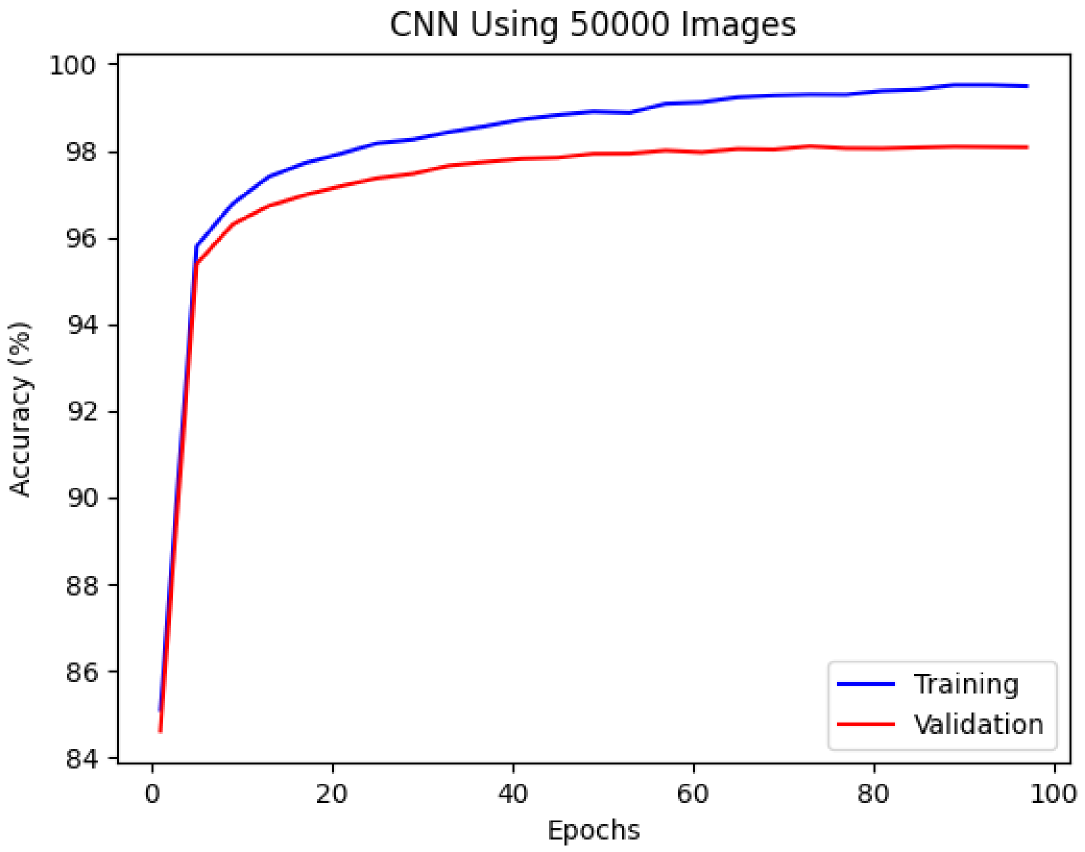
This is the network architecture with the layers in the following order:

1. Resized the input into a shape of (-1, 1, 28, 28)
2. 2D convolutional layer which contains 1 input channel and 6 output channels, meaning that it defines 6 different filters. The size of the kernel is 1x5x5. The stride is 1 and the padding is 0.
3. Then the result is inputted into a ReLU activation function.
4. The next layer is a 2D max pooling layer that has a kernel size of 2x2 and a stride of 2 and 0 padding.
5. The following layer is a 2D convolutional layer with 6 input channels and 16 output channels, meaning that it defines 16 different filters. The size of the kernel 6x5x5. The stride is 1 and the padding is 0.
6. ReLU activation function
7. Another 2D max pooling layer that has a kernel size of 2x2 and a stride of 2 and 0 padding.
8. The result is reshaped and vectorized into a shape of (-1, 256)
9. Fully connected layer that has 256 input channels and 120 output channels.
10. ReLU activation function.
11. Fully connected layer that has 120 input channels and 84 output channels.
12. ReLU activation function.
13. Fully connected layer that has 84 input channels and 10 output channels.

Overfitting

Only using the first 2000 images rather than using all 50000 images quickly leads to overfitting. Using only the first 2000 images, the accuracy of the training set reaches 90% with just a few epochs. The training accuracy then reaches and stays at 100%. The validation accuracy reaches around 95%. Using the entire training set, the neural network reaches 90% accuracy extremely fast, but this is due to more weight updates per epoch. The validation accuracy is better, achieving over 98%. Clearly, when training on 2000 images, the validation set accuracy is worse. This is to be expected because a smaller dataset exposes the neural network to a smaller variety of images, leading to more overfitting.





Testing

In order to attain very accurate results using the convolutional neural network, I trained the convolutional neural network on all 50000 images and used an adaptive learning optimizer, Adam. This resulted in a final accuracy of 98.88%