Assignment 2

Structure of Neural Net

We have one input layer, two convolutional layers, one dropout layer, and two fully connected layers (one of which is the output layer). In total, we used 6 layers. The first non-linear part of the model are the Convolutions. In the Convolutional layers, the activate function is ReLu which is non-linear because the function is defined as `Max(0,x)`. ReLu, essentially acts as a threshold, the input has to be at least greater than 0 for the neuron's output to be considered. which After ReLu the Convolutional layers go through Max Pooling, which take all the matrix and divides it into sub-arrays, and then creates an array by taking only the highest value in each sub-array. The intuition behind Max-Pooling is that it takes the heaviest features only, so it is easier to distinguish between different digits. After the two convolutional layers, the next layer is the first fully connected layer which again uses ReLu for its activation function. The next non-linearly lies in the Dropout layer, which with some probability shuts off a neuron, as a way to combat overfitting. Finally after the Dropout layer, the Second Fully Connected Layer sums up the weights, x values, and biases, and spits out 10 probabilities, which help to determine which digit the image is. The input is the 784 pixel values that represent the digit, and the output is a 10 dimensional vector which containing probabilities of the image being each digit.

Loss Function

```
-1 * Sum (y_i_truedistribution * y_i_predicted_distribution) for all i.
```

The Loss Function is there to measure the inaccuracy of the model. The model achieves accuracy by attempting to minimize the Loss Function, thus it is integral to achieving high accuracy.

Accuracy Function

```
For all i from 1 to n, where n is the number of y values,
1/n * (Sum (y_i_predicted == y_i_label))
```

The previous function is equivalent to performing reduced means on `tf.equal(tf.argmax(y,1), tf.argmax(y_,1))`. The accuracy function outputs how accurate the neural net is over the set it was used on. This accuracy function assigns 0 if the predicted value of the neural net doesn't equals the true value, and outputs one if the prediction matches the actual label. Then it sums up the results of the evaluation, and divides by the total number of evaluations.

Optimizer

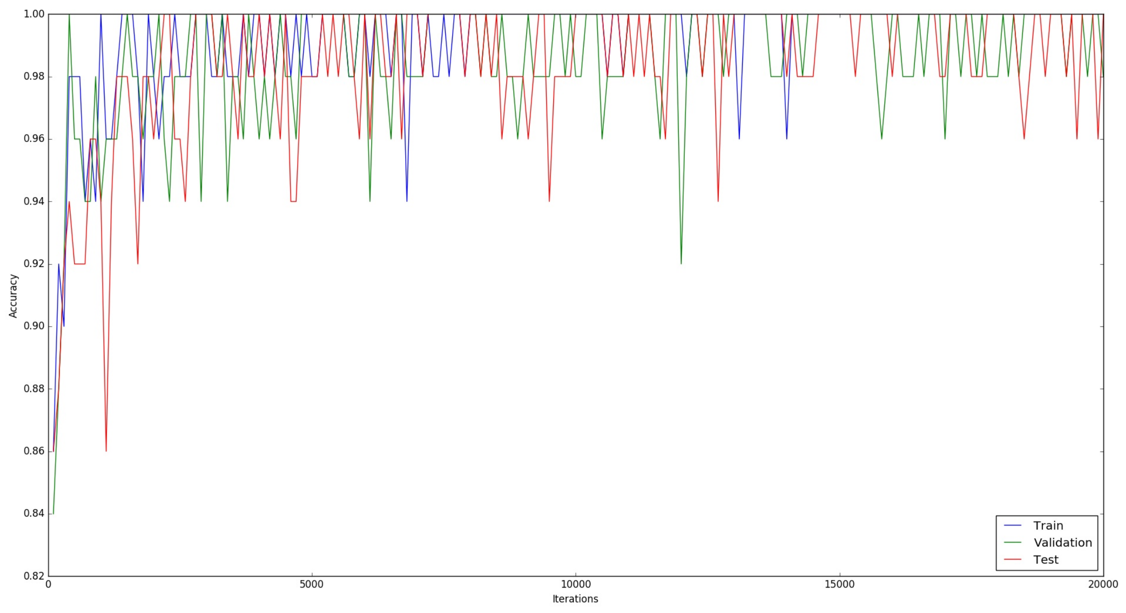
We used the Adam Optimizer, with step size 10^-4. When the step size is large, the loss will initially decrease fast, but will have trouble converging to the optimal value, as it more likely to change the weights and biases too much by “overshooting”. However “overshooting” could be an advantage if the neural net loss function is stuck at a local minimum, and “overshooting” brings it closer to the global minimum. If the step size is smaller, the model will take more time to reach higher accuracy, as it will make smaller changes to the weights and biases, but because of this the neural net, is more likely to stay closer to local minimums.

Sensing

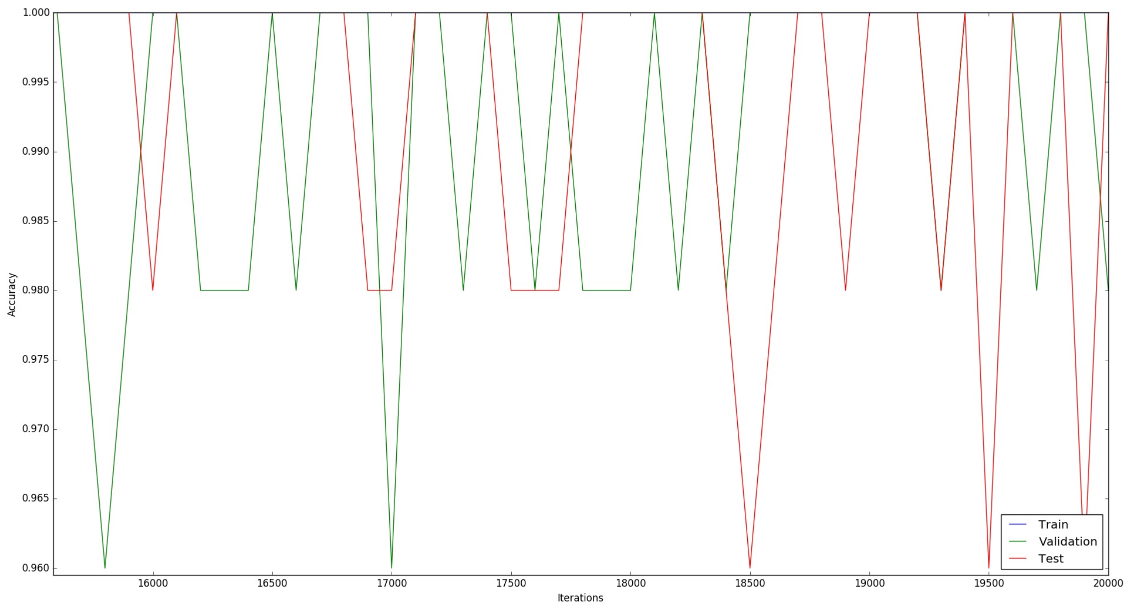
The accuracy of the model is likely not as high as the accuracy of the plots. This is because there is likely to be noise in sensing which will lead to error. This issue isn’t an issue of machine learning itself, but rather an issue of data collection.

Accuracy Plot

Full Accuracy Plot 20,000 Iterations



Zoomed-in Accuracy Plot



Observations

The accuracy plots at first has a generally trend of increasing at first for all three sets, and all three plots seem to increase in accuracy together, which makes sense as the model gets better it should perform better for all three data sets. After 5000 iterations it is rare for the accuracy to go below .98 percent, and after 15000 iterations the accuracy for the training data doesn't go below 100 percent. This implies that the model's loss function is near its minimum, and that the iterations can only improve the model slightly after 5000 iterations.

Extra Credit

We were able to get the training accuracy to 100%, and the validation and test accuracy to in between 98-100%. The additional training ceased to significantly increase accuracy after 5000 iterations. The convolution layers, and fully-connected layers helped most, while dropout didn't add much to accuracy.