Daniel Kim, Chad Underhill, Spyridon Antonatos

CS4341 Project 3 Written Report

9/6/2017

Set of Experiments Performed

We experimented first with the given hidden layer and the output layer. We used 784 nodes for the hidden layer, the activation function relu, and the kernel initializer glorot_normal. The batch size for training was 512, and the number of epochs were 100. This resulted in around 89.5% training accuracy and 88.6% validation accuracy.

Next, we tried to double the number of epochs to 200. This resulted in 92.6% training accuracy and 88.9% validation accuracy. The training accuracy was much better than the last experiment, but the validation accuracy was not much different, and running the program with 200 epochs took much more time, so we decided to stick with 100 epochs.

After this, we tried changing just the batch size from 512 to 256. This ran a bit slower than the first experiment, but the resulting training accuracy was 91.9% and the validation accuracy was 90.8%. This was overall much better than the first experiment, so we tried decreasing the batch size further to 128. However, this yielded only a 2% increase in training accuracy and a 1% increase in validation accuracy. A batch size of 64 only increased the training accuracy by 2%, so we decided to stick with 128 as our batch size.

Changing the activation function to selu resulted in a training accuracy of 92.8% and a validation accuracy of 88.7%. This was worse than the relu activation function. We also tried the tanh activation function which gave a training accuracy of 91% and a validation accuracy of 90.7%. The relu activation function performed the best of these, so we decided to stick with relu as our activation function.

For our final experiment using only one hidden layer, we tried changing the weight initialization scheme, the kernel initializer. Originally, it was set to glorot_normal, which used a truncated normal distribution centered on 0 using the number of input and output units. We changed it to he_normal, which is the same as glorot_normal, except it does not use the number of output units. This resulted in a very slight(< 1%) increase in both accuracies, so we tried random_normal next, which initializes weights with a normal distribution. This decreased the accuracies. We tried many more weight initialization schemes, but these generally gave equal or worse accuracies.

After testing with one hidden layer, we decided to test two hidden layers. We added a layer with 784 nodes using relu as the activation and glorot_normal as the weight initialization. This did not yield any change in the accuracies. We tried changing the number of nodes in the second hidden layer to 196. This gave a slight increase in both accuracies. At this point our training accuracy was 96.5% and our validation accuracy was 91.8%. Next, we tried decreasing the number of nodes in the second layer even further to 49, which gave no increase in accuracy.

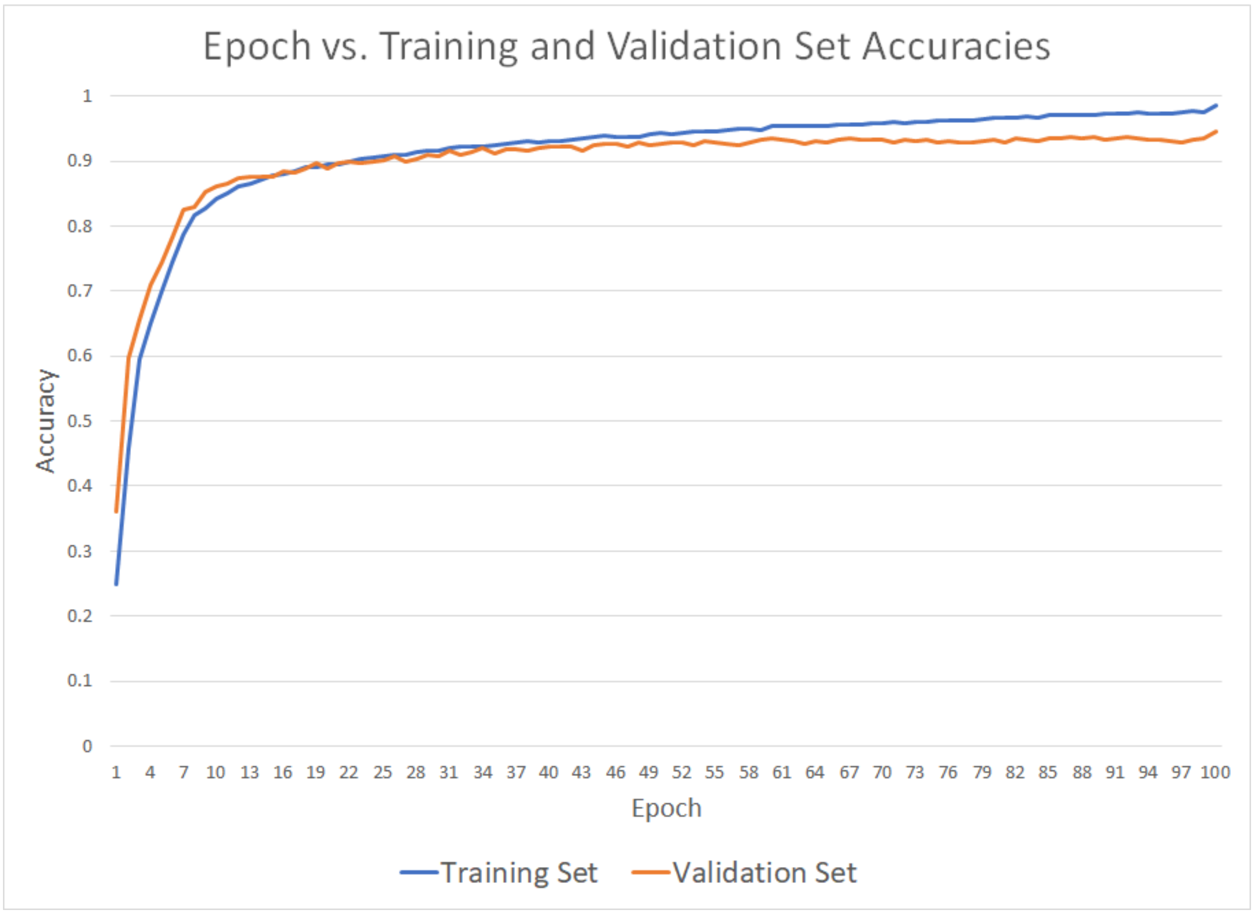
We also tried adding a third hidden layer. We kept the second hidden layer with 196 nodes using relu and glorot_normal and added a third layer with the same fields. This resulted in a training accuracy of 97.2% and a validation accuracy of 92.5%, which was a slight increase. Using 49 nodes in the third layer increased the validation accuracy to 93.7%.

After more experimenting with different combinations of activation functions, number of nodes, and layers, we found our best model to be the three layer model described above.

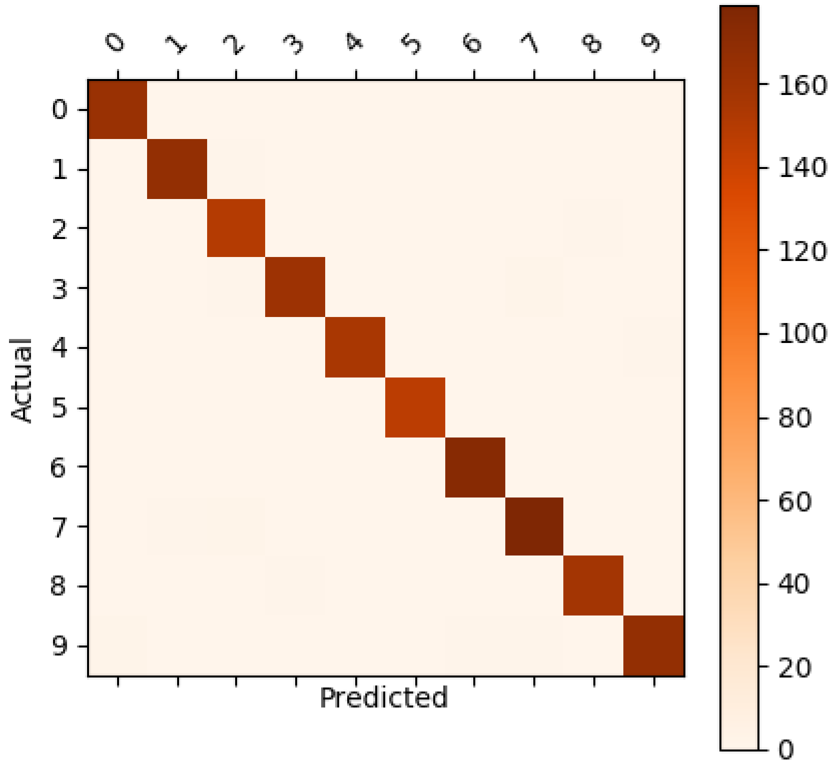
Model & Training Procedure Description

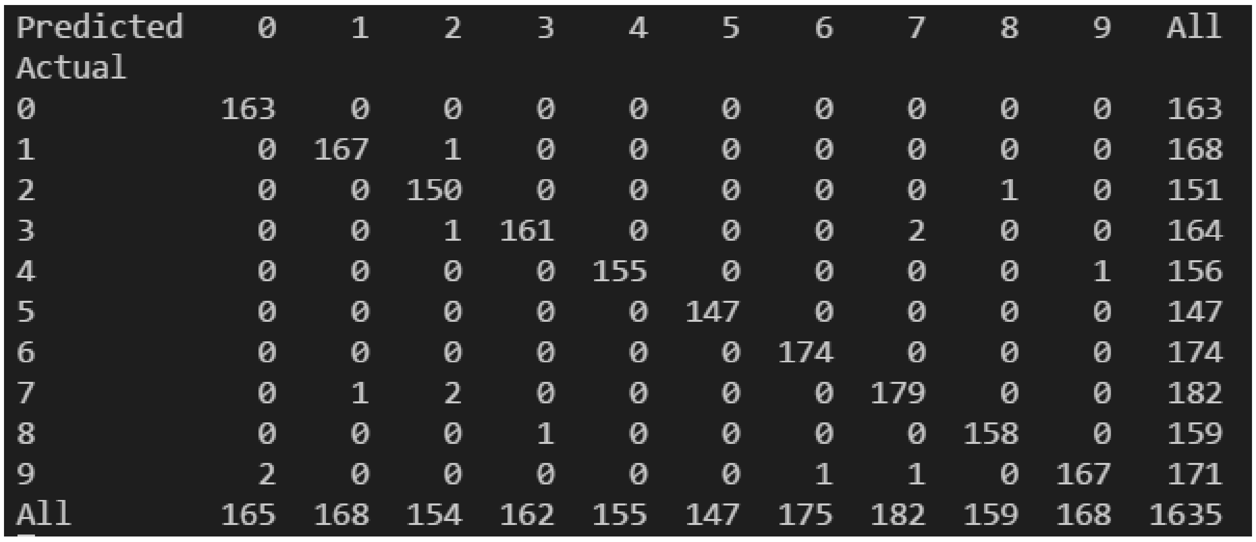
The most accurate model we were able to obtain was using three layers. The first layer had 784 neurons, the second layer had 196 neurons, and the third layer had 49 neurons. The weight initialization scheme (kernel initializer) for all layers was set to glorot_normal, the activation function for all layers was set to relu, the number of epochs was set to 100, and the batch size was set to 128.

Graph



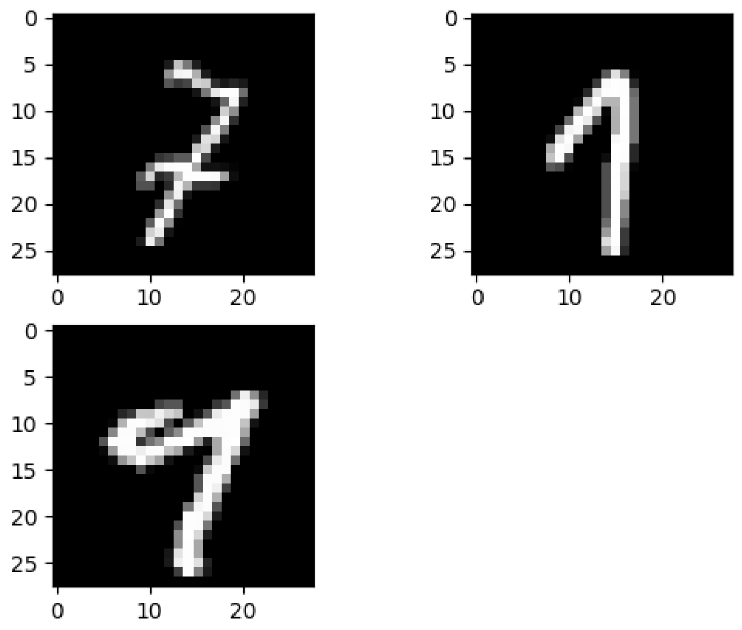
Model Performance & Confusion Matrix

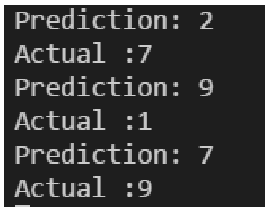




```
Predicted    0    1    2    3    4    5    6    7    8    9   All
Actual
0          163    0    0    0    0    0    0    0    0    0   163
1            0  167    1    0    0    0    0    0    0    0   168
2            0    0  150    0    0    0    0    0    1    0   151
3            0    0    1  161    0    0    0    2    0    0   164
4            0    0    0    0  155    0    0    0    0    1   156
5            0    0    0    0    0  147    0    0    0    0   147
6            0    0    0    0    0    0  174    0    0    0   174
7            0    1    2    0    0    0    0  179    0    0   182
8            0    0    0    1    0    0    0    0  158    0   159
9            2    0    0    0    0    0    1    1    0  167   171
All        165  168  154  162  155  147  175  182  159  168  1635
```

Visualization





```
Prediction: 2
Actual :7
Prediction: 9
Actual :1
Prediction: 7
Actual :9
```

We believe these observations were misclassified due to their closeness in shape with the predicted numbers. The training set could have had “messy” versions of the predicted numbers that looked similar to these three images, causing the model to think the way it did. For example, the 9 could have looked similarly to a “messy” 7 in the training set, causing the model to predict a 7 instead of a 9.