<u>Project Phase – 3 Classification Using Neural Networks and Deep Learning – Report</u>

<u>ASU – ID: 1217192561</u>

Overview:

In this project, each person has to create a convolutional neural network on MNIST dataset using the Google Colab. There are certain parameters we have to follow to experiment with the model to test its accuracy. The requirements for the experiment are as follows:

- The input size of the image dataset is given to be 28x28.
- The first hidden layer is a convolutional layer which should contain 6 feature maps with kernel size 3x3 and stride to be 1.
- This hidden layer is followed by the max pooling layer, which should be of 2x2 and its stride should be 1.
- After max pooling, the layer is connected to the second convolutional layer with 16 feature maps with kernel size 3x3 and stride to be 1.
- This layer is again followed by the max pooling layer, which should be of 2x2 and its stride should be 1.
- Now the layer is fully connected to next hidden layer with 120 nodes and RelU as the activation function.
- This layer is followed by another fully connected layer with 84 and RelU as the activation function then connected to softmax layer with 10 output nodes (corresponding to the 10 classes).

Experiment:

For experiment I have used 60,000 samples for training and 10,000 samples for testing. The changes made to the model for experimenting are as follows:

1. Change the kernel size to 5x5, redo the experiment, plot the learning errors along with the epoch, and report the testing error and accuracy on the test set.
2. Change the number of the feature maps in the first and second convolutional layers, redo the experiment, plot the learning errors along with the epoch, and report the testing error and accuracy on the test set.
3. Change the feature maps in the first and second convolutional layer and also the kernel size to 5x5 and redo the experiment, plot the learning errors along with the epoch, and report the testing error and accuracy on the test set.

Results:

First the result for the original model with the basic requirement as defined above for the experiment.

Result_1 – Here we use the basic specifications that are mentioned in the requirement portion and try to fetch the results as and plot the graph of the for epochs and learning errors also plotted the graph for the epochs v/s accuracy of the model.

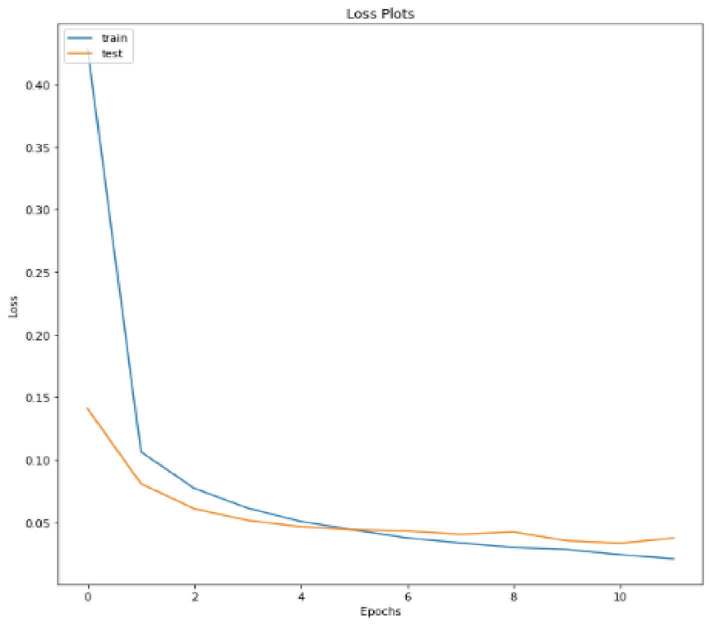
The overall accuracy of the model

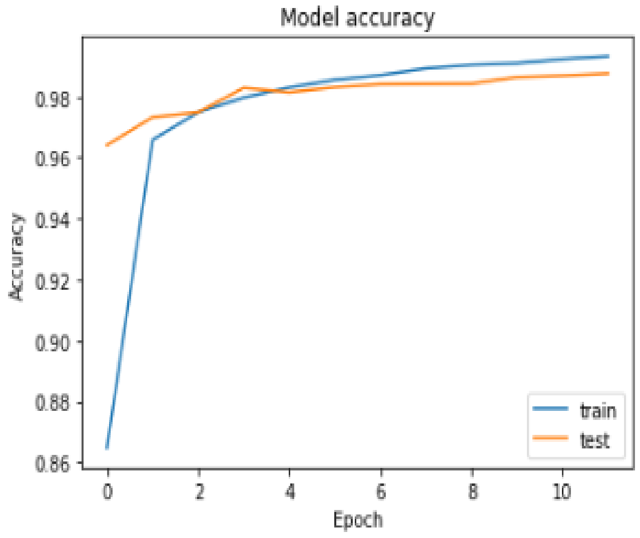
Accuracy: 98.72%

Total Loss on test values: 0.03733

Below figure represents the data of the value loss and the accuracy for each epoch used and the accuracy of the test data and the loss incurred during the test. Also, we plotted the graph Loss Plots where y-axis is Loss values against the number of epochs on x-axis:

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 4s 67us/step - loss: 0.4282 - accuracy: 0.8638 - val_loss: 0.1406 - val_accuracy: 0.9549
Epoch 2/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.1063 - accuracy: 0.9672 - val_loss: 0.0808 - val_accuracy: 0.9736
Epoch 3/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0771 - accuracy: 0.9759 - val_loss: 0.0608 - val_accuracy: 0.9803
Epoch 4/12
60000/60000 [==============================] - 4s 66us/step - loss: 0.0613 - accuracy: 0.9806 - val_loss: 0.0515 - val_accuracy: 0.9826
Epoch 5/12
60000/60000 [==============================] - 4s 66us/step - loss: 0.0506 - accuracy: 0.9841 - val_loss: 0.0464 - val_accuracy: 0.9841
Epoch 6/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0438 - accuracy: 0.9861 - val_loss: 0.0441 - val_accuracy: 0.9860
Epoch 7/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0374 - accuracy: 0.9880 - val_loss: 0.0430 - val_accuracy: 0.9859
Epoch 8/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0334 - accuracy: 0.9895 - val_loss: 0.0403 - val_accuracy: 0.9873
Epoch 9/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0299 - accuracy: 0.9902 - val_loss: 0.0422 - val_accuracy: 0.9863
Epoch 10/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0282 - accuracy: 0.9910 - val_loss: 0.0352 - val_accuracy: 0.9878
Epoch 11/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0241 - accuracy: 0.9925 - val_loss: 0.0332 - val_accuracy: 0.9885
Epoch 12/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0209 - accuracy: 0.9933 - val_loss: 0.0373 - val_accuracy: 0.9872
Test loss: 0.03732985155655769
Test accuracy: 0.9872000217437744
```





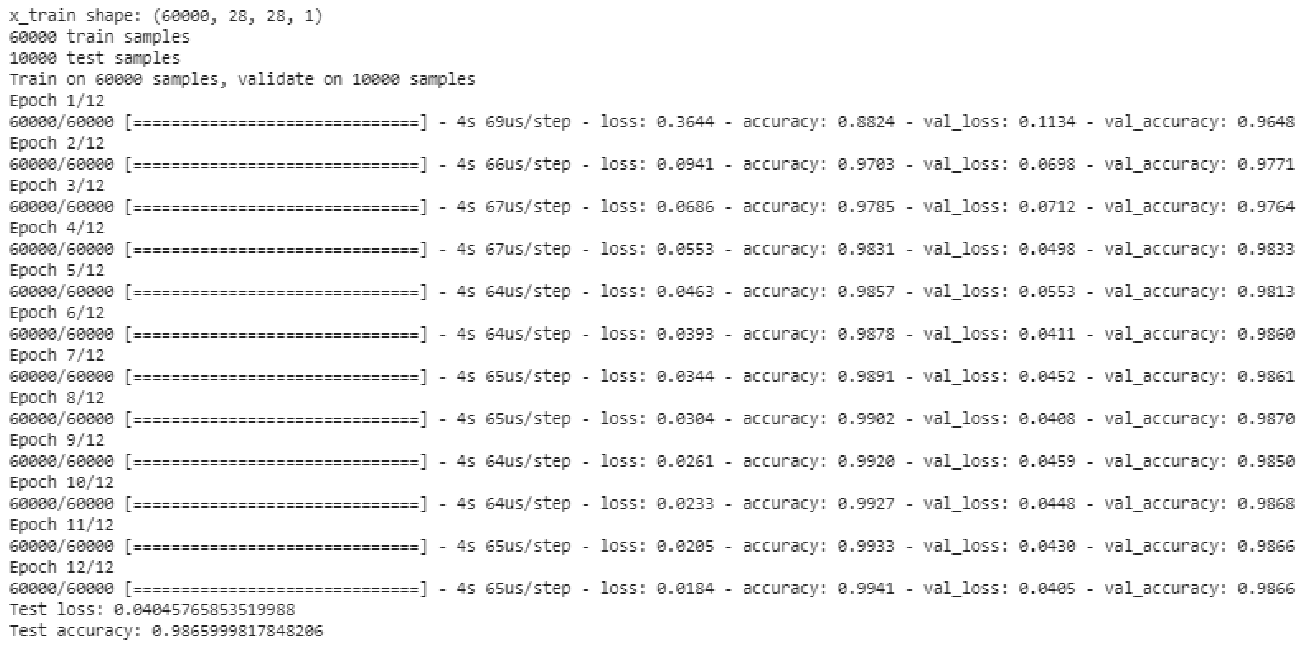
Result_2 – In this, we experiment with the kernel of the convoluted layer and try to change its value to 5x5 instead of 3x3 and check the result we are getting and plotting the graph same done in the result_1.

The overall accuracy of the model is

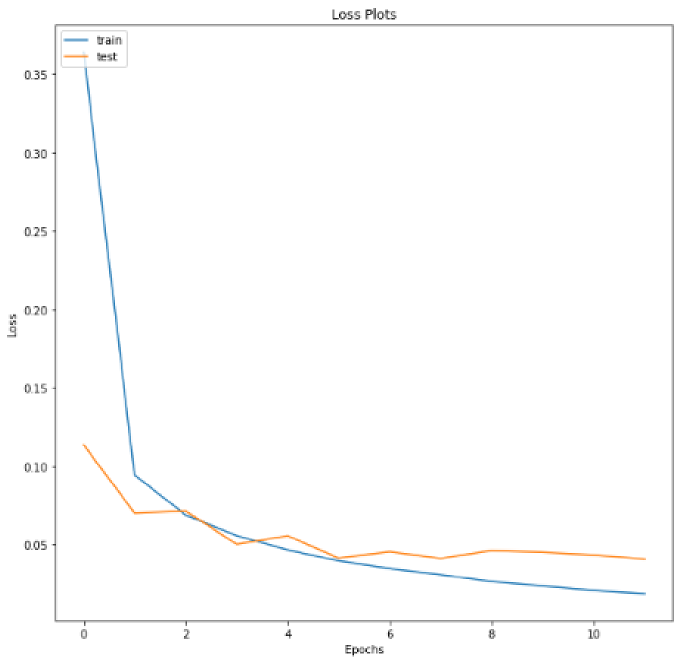
Accuracy: 98.66%

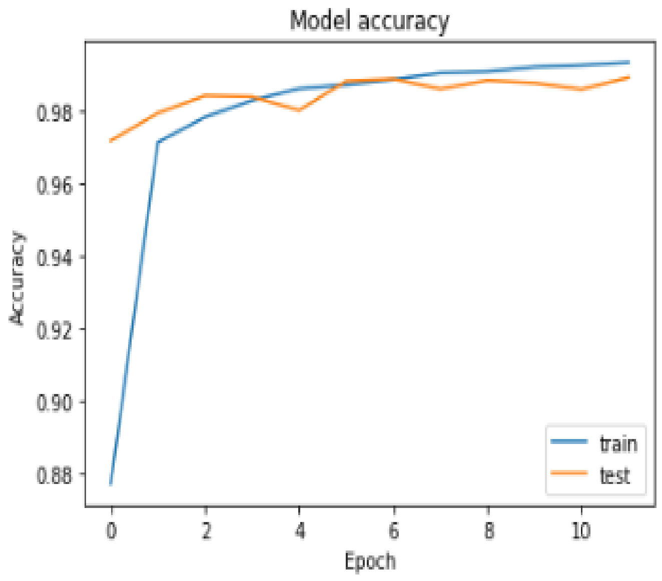
Total loss on test values: 0.0404

The results are as follows:



```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 4s 69us/step - loss: 0.3644 - accuracy: 0.8824 - val_loss: 0.1134 - val_accuracy: 0.9648
Epoch 2/12
60000/60000 [==============================] - 4s 66us/step - loss: 0.0941 - accuracy: 0.9703 - val_loss: 0.0698 - val_accuracy: 0.9771
Epoch 3/12
60000/60000 [==============================] - 4s 67us/step - loss: 0.0686 - accuracy: 0.9785 - val_loss: 0.0712 - val_accuracy: 0.9764
Epoch 4/12
60000/60000 [==============================] - 4s 67us/step - loss: 0.0553 - accuracy: 0.9831 - val_loss: 0.0498 - val_accuracy: 0.9833
Epoch 5/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0463 - accuracy: 0.9857 - val_loss: 0.0553 - val_accuracy: 0.9813
Epoch 6/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0393 - accuracy: 0.9878 - val_loss: 0.0411 - val_accuracy: 0.9860
Epoch 7/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0344 - accuracy: 0.9891 - val_loss: 0.0452 - val_accuracy: 0.9861
Epoch 8/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0304 - accuracy: 0.9902 - val_loss: 0.0408 - val_accuracy: 0.9870
Epoch 9/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0261 - accuracy: 0.9920 - val_loss: 0.0459 - val_accuracy: 0.9850
Epoch 10/12
60000/60000 [==============================] - 4s 64us/step - loss: 0.0233 - accuracy: 0.9927 - val_loss: 0.0448 - val_accuracy: 0.9868
Epoch 11/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0205 - accuracy: 0.9933 - val_loss: 0.0430 - val_accuracy: 0.9866
Epoch 12/12
60000/60000 [==============================] - 4s 65us/step - loss: 0.0184 - accuracy: 0.9941 - val_loss: 0.0405 - val_accuracy: 0.9866
Test loss: 0.04045765853519988
Test accuracy: 0.9865999817848206
```





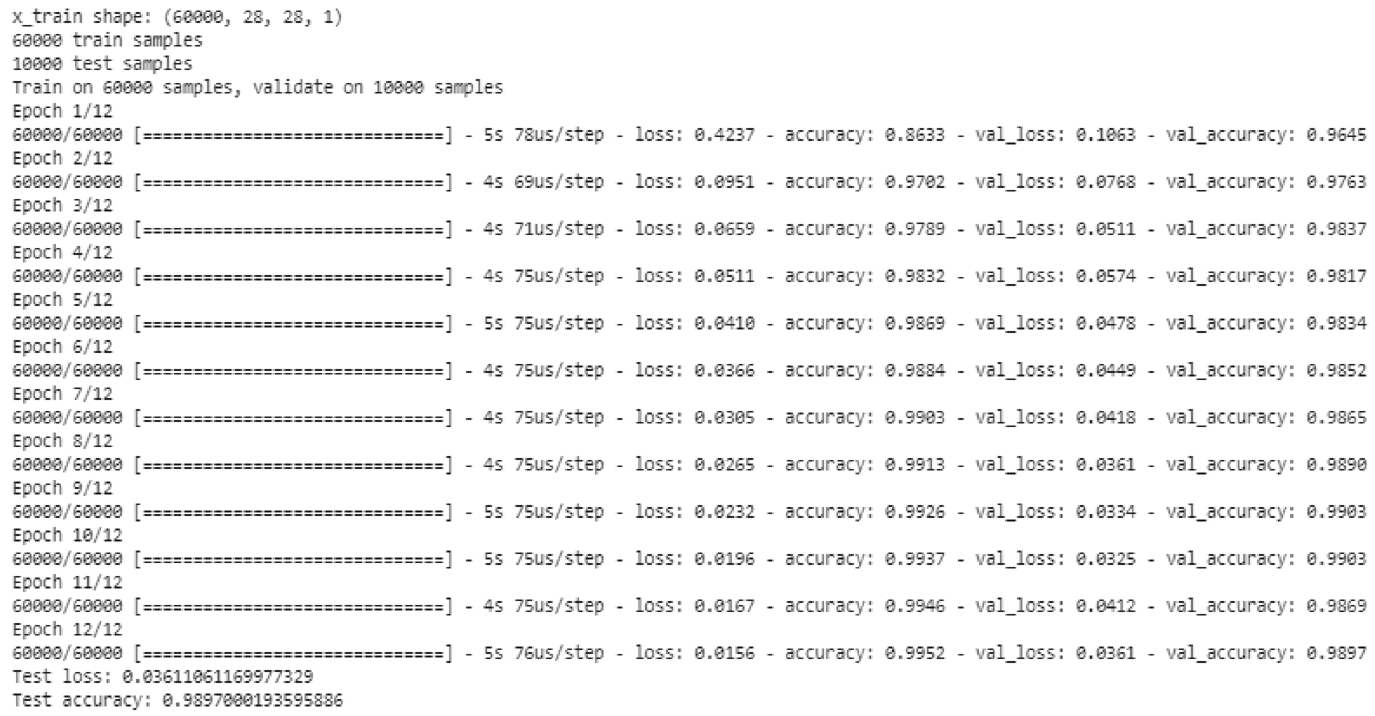
Result_3 – This time we experiment with the feature maps and try to change the value from 6 to 16 for the first convolutional layer and 16 to 32 for the second convolutional layer and then try to plot the graph and the result of the experiment we are doing.

The overall accuracy of the model is

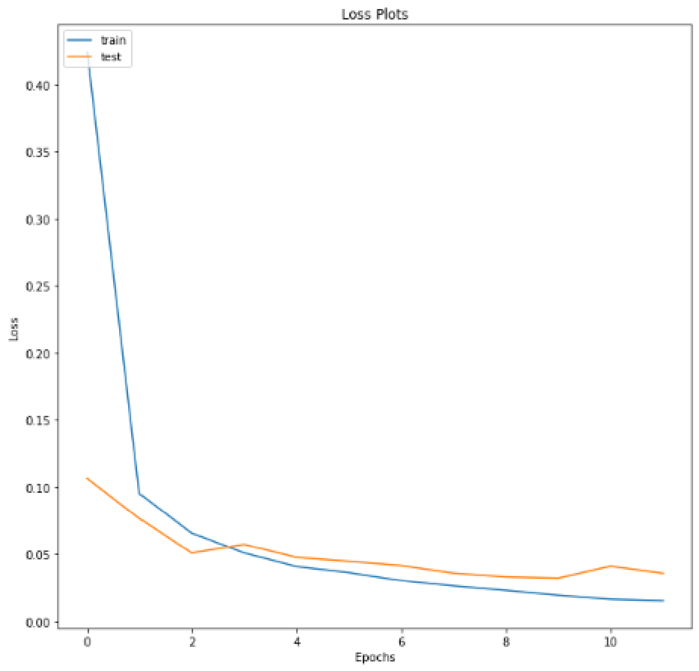
Accuracy: 98.97%

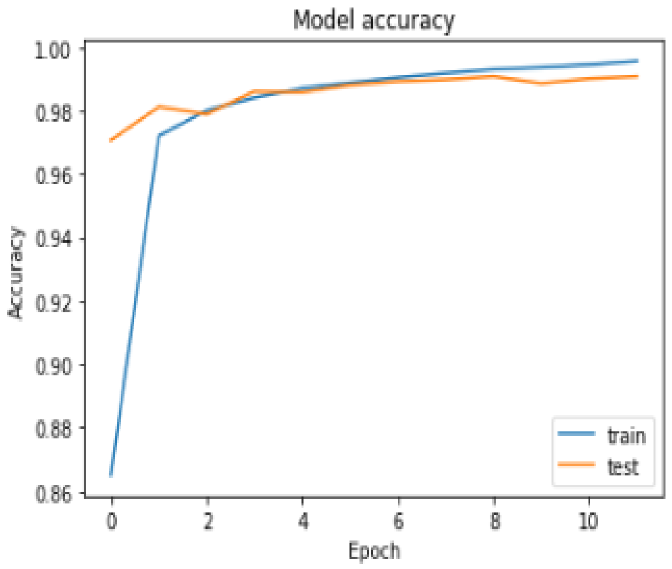
Total loss on test values: 0.0361

The results are as follows:



```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.4237 - accuracy: 0.8633 - val_loss: 0.1063 - val_accuracy: 0.9645
Epoch 2/12
60000/60000 [==============================] - 4s 69us/step - loss: 0.0951 - accuracy: 0.9702 - val_loss: 0.0768 - val_accuracy: 0.9763
Epoch 3/12
60000/60000 [==============================] - 4s 71us/step - loss: 0.0659 - accuracy: 0.9789 - val_loss: 0.0511 - val_accuracy: 0.9837
Epoch 4/12
60000/60000 [==============================] - 4s 75us/step - loss: 0.0511 - accuracy: 0.9832 - val_loss: 0.0574 - val_accuracy: 0.9817
Epoch 5/12
60000/60000 [==============================] - 5s 75us/step - loss: 0.0410 - accuracy: 0.9869 - val_loss: 0.0478 - val_accuracy: 0.9834
Epoch 6/12
60000/60000 [==============================] - 4s 75us/step - loss: 0.0366 - accuracy: 0.9884 - val_loss: 0.0449 - val_accuracy: 0.9852
Epoch 7/12
60000/60000 [==============================] - 4s 75us/step - loss: 0.0305 - accuracy: 0.9903 - val_loss: 0.0418 - val_accuracy: 0.9865
Epoch 8/12
60000/60000 [==============================] - 4s 75us/step - loss: 0.0265 - accuracy: 0.9913 - val_loss: 0.0361 - val_accuracy: 0.9890
Epoch 9/12
60000/60000 [==============================] - 5s 75us/step - loss: 0.0232 - accuracy: 0.9926 - val_loss: 0.0334 - val_accuracy: 0.9903
Epoch 10/12
60000/60000 [==============================] - 5s 75us/step - loss: 0.0196 - accuracy: 0.9937 - val_loss: 0.0325 - val_accuracy: 0.9903
Epoch 11/12
60000/60000 [==============================] - 4s 75us/step - loss: 0.0167 - accuracy: 0.9946 - val_loss: 0.0412 - val_accuracy: 0.9869
Epoch 12/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0156 - accuracy: 0.9952 - val_loss: 0.0361 - val_accuracy: 0.9897
Test loss: 0.03611061169977329
Test accuracy: 0.9897000193595886
```





Result_4 – In this experiment we change the feature mapping of the convolutional layer to the value 16 and 32 as changed previously and also change the kernel size to 5x5 and then try to plot the graph and analyse the result we are getting.

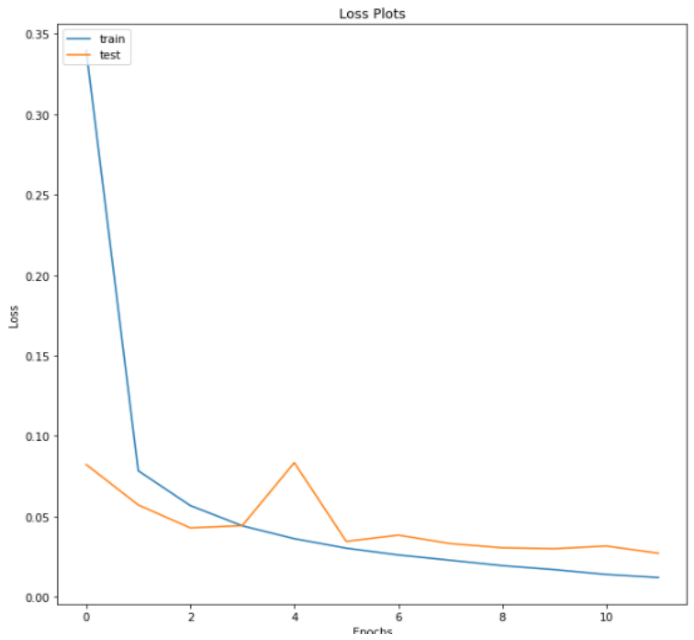
The overall accuracy of the model is

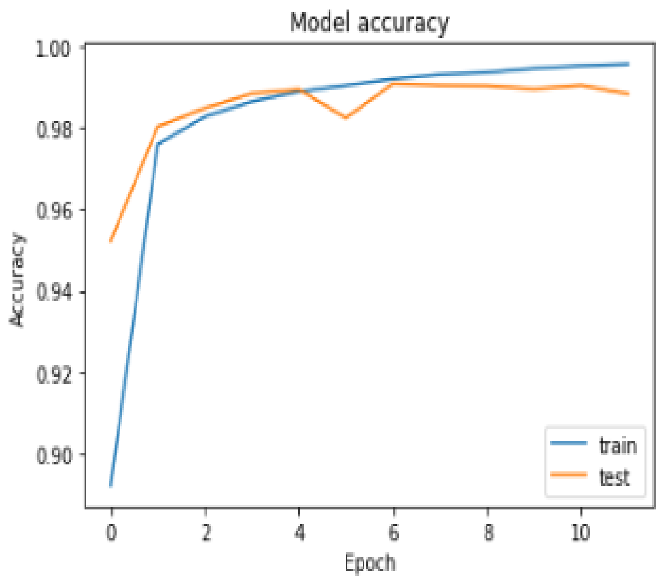
Accuracy: 99.04%

Total loss on test values: 0.0271

The result are as follows:

```
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 9s 144us/step - loss: 0.3397 - accuracy: 0.8950 - val_loss: 0.0822 - val_accuracy: 0.9746
Epoch 2/12
60000/60000 [==============================] - 2s 41us/step - loss: 0.0784 - accuracy: 0.9762 - val_loss: 0.0571 - val_accuracy: 0.9807
Epoch 3/12
60000/60000 [==============================] - 3s 42us/step - loss: 0.0567 - accuracy: 0.9821 - val_loss: 0.0429 - val_accuracy: 0.9874
Epoch 4/12
60000/60000 [==============================] - 3s 42us/step - loss: 0.0442 - accuracy: 0.9857 - val_loss: 0.0443 - val_accuracy: 0.9857
Epoch 5/12
60000/60000 [==============================] - 3s 42us/step - loss: 0.0361 - accuracy: 0.9887 - val_loss: 0.0834 - val_accuracy: 0.9708
Epoch 6/12
60000/60000 [==============================] - 2s 40us/step - loss: 0.0303 - accuracy: 0.9904 - val_loss: 0.0345 - val_accuracy: 0.9885
Epoch 7/12
60000/60000 [==============================] - 2s 41us/step - loss: 0.0260 - accuracy: 0.9916 - val_loss: 0.0384 - val_accuracy: 0.9872
Epoch 8/12
60000/60000 [==============================] - 2s 41us/step - loss: 0.0228 - accuracy: 0.9930 - val_loss: 0.0331 - val_accuracy: 0.9882
Epoch 9/12
60000/60000 [==============================] - 2s 41us/step - loss: 0.0194 - accuracy: 0.9937 - val_loss: 0.0305 - val_accuracy: 0.9892
Epoch 10/12
60000/60000 [==============================] - 3s 42us/step - loss: 0.0169 - accuracy: 0.9944 - val_loss: 0.0299 - val_accuracy: 0.9902
Epoch 11/12
60000/60000 [==============================] - 2s 40us/step - loss: 0.0139 - accuracy: 0.9955 - val_loss: 0.0316 - val_accuracy: 0.9901
Epoch 12/12
60000/60000 [==============================] - 2s 41us/step - loss: 0.0120 - accuracy: 0.9965 - val_loss: 0.0271 - val_accuracy: 0.9904
Test loss: 0.027114159934620692
Test accuracy: 0.9904000163078308
```





Evaluation of Results:

For each and every plot result we are getting the graph is decreasing drastically which shows the decrease of the losses incurred for each epoch. The outcomes of the experiment are:

- From the experiment we get to know that as we increase the size of the kernel the accuracy is not affected by any big margin.
- If we change the size of feature maps of convolutional layer leaving the kernel size intact (at 3x3) the accuracy increases by certain margin.
- If we change the size of the feature maps of convolutional layer and also changing the kernel size (5x5) the accuracy even increases further.

Conclusion:

From this experiment we can draw out certain conclusions from the outcomes of the experiment we analysed, first thing that kernel size does not affect that much to the accuracy if the feature maps of the convolutional layer are same in size. Second, if we want to increase the accuracy of the model, we have to keep the feature maps and kernel size value as high as possible to get the maximum throughput from the model.