SEL TOPICS FROM COMPUTER SCIENCE ASSIGNMENT 1 - REPORT

BY

SUKRIT	2018A7PS0205H
KUMAR PRANJAL	2018A7PS0163H
SHETTY KARTHIK RAVINDRA	2018A7PS0141H

BITS Pilani
Hyderabad Campus

Submitted to
Dr. N. L. Bhanu Murthy

BASIC OVERVIEW

In this report we present different model architectures and designs along with variations in hyper parameters, to get the maximum accuracy for the MNIST Dataset. All of the models built are ANNs (Artificial Neural Networks).

The models are run for **15 Epochs** for comparisons. For the models, we have used only **dense** layers with different numbers of nodes, hidden layers, different architectures and activation functions. We start from a very basic neural network with only one dense layer and try to improve upon it by taking outputs from its training accuracy, training loss and testing accuracy.

PURPOSE

The main purpose of this project is to start from a very basic model architecture consisting of a single dense layer and by analysing the accuracy and losses obtained, build better models by experimenting with the number of hidden layers, number of nodes and hyperparameters.

TECHNOLOGIES USED

We have used **Keras Library** which runs on top of the **Tensorflow framework**. For tasks like splitting the training and testing data we used the **scikit-learn** libraries methods. And for plotting the accuracy and loss graphs we have used the matplotlib library.

The first 12 deep learning models are built using the Keras Library and the next 4 traditional models are built using sklearn library.

OVERVIEW OF THE MODELS

FIRST 12 MODELS

The first layer of each model is the input layer where 500 rows from the dataset goes into the network at a time to update the weights. Once every row passes through the network, it is called an ‘epoch’. We used 15 epochs as a standard to train all the models.

The second layer is a dense/hidden layer. Here, to make comparisons between different models, we started with 1 hidden layer and then added multiple layers for further models as long as the accuracy of the models were increasing. These hidden layers find the patterns within the data by updating the weights after every batch-size passed in the model.

The last layer is the output layer which predicts the values for the test data.

LAST 4 MODELS

These were built using traditional machine learning algorithms. The last four models we used are SVM (Support Vector Machine) using ‘rbf’ function (Gaussian Model), SVM using ‘poly’ (polynomial), Decision Tree Model and Random Forest Classifier.

We then try to compare the deep learning models from Keras to the four traditional models made using sklearn library.

DIFFERENT MODEL ARCHITECTURES

Here we try to analyse all the different models using the following sections :
1) Model Structure
2) Model Training and Testing Accuracies
3) Accuracy and Loss Plots
4) Conclusions
5) Possible Improvements
6) Training Screenshot

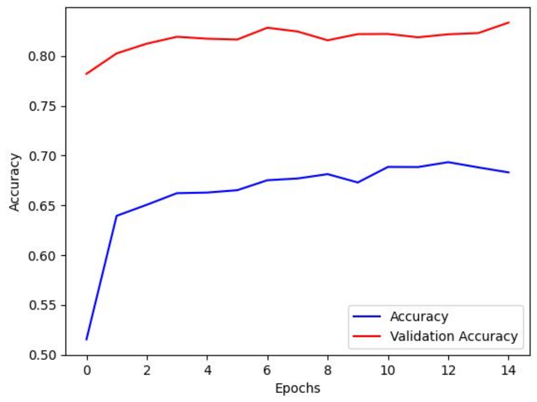
MODEL 1 :

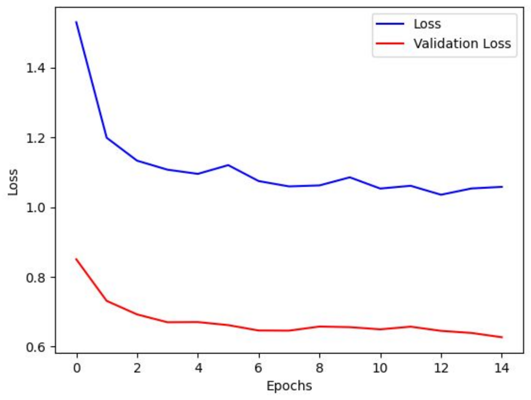
Architecture :

```
Layer (type)            Output Shape         Param #     Activation
=================================================================
flatten (Flatten)         (None, 784)          0

dense (Dense)              (None, 10)          7850         softmax
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

Training Accuracy	82.65 %
Training Loss	0.64
Testing Accuracy	84.18 %

Plots :





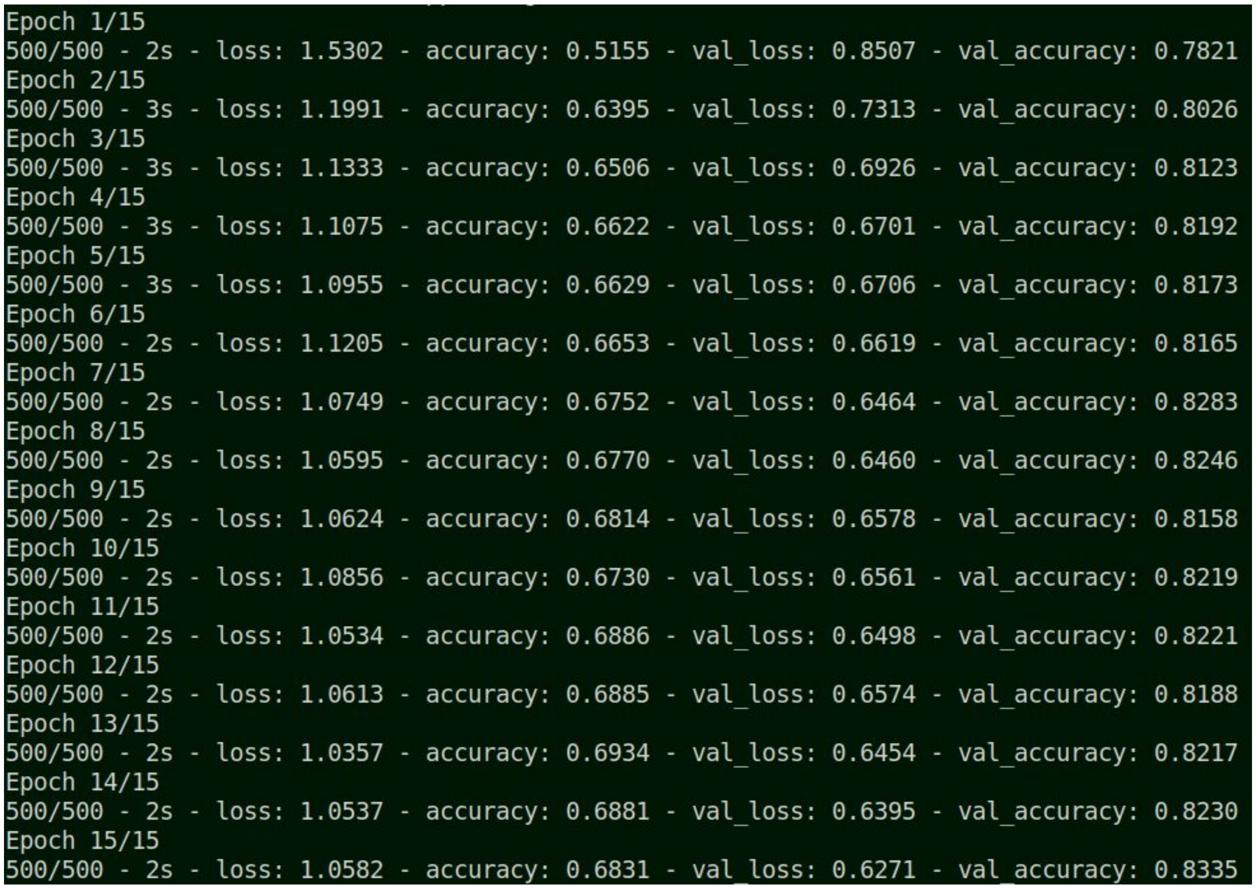
Conclusions:

Since this model only had a single dense layer, it did not perform very well. It had a high loss and the training and validation datasets could not give the same performance improvements as visible from the plots.

Possible Improvements:

We can try using a different activation for the model than softmax like sigmoid because it generally performs better for shallow networks.

Training Screenshot:



```
Epoch 1/15
500/500 - 2s - loss: 1.5302 - accuracy: 0.5155 - val_loss: 0.8507 - val_accuracy: 0.7821
Epoch 2/15
500/500 - 3s - loss: 1.1991 - accuracy: 0.6395 - val_loss: 0.7313 - val_accuracy: 0.8026
Epoch 3/15
500/500 - 3s - loss: 1.1333 - accuracy: 0.6506 - val_loss: 0.6926 - val_accuracy: 0.8123
Epoch 4/15
500/500 - 3s - loss: 1.1075 - accuracy: 0.6622 - val_loss: 0.6701 - val_accuracy: 0.8192
Epoch 5/15
500/500 - 3s - loss: 1.0955 - accuracy: 0.6629 - val_loss: 0.6706 - val_accuracy: 0.8173
Epoch 6/15
500/500 - 2s - loss: 1.1205 - accuracy: 0.6653 - val_loss: 0.6619 - val_accuracy: 0.8165
Epoch 7/15
500/500 - 2s - loss: 1.0749 - accuracy: 0.6752 - val_loss: 0.6464 - val_accuracy: 0.8283
Epoch 8/15
500/500 - 2s - loss: 1.0595 - accuracy: 0.6770 - val_loss: 0.6460 - val_accuracy: 0.8246
Epoch 9/15
500/500 - 2s - loss: 1.0624 - accuracy: 0.6814 - val_loss: 0.6578 - val_accuracy: 0.8158
Epoch 10/15
500/500 - 2s - loss: 1.0856 - accuracy: 0.6730 - val_loss: 0.6561 - val_accuracy: 0.8219
Epoch 11/15
500/500 - 2s - loss: 1.0534 - accuracy: 0.6886 - val_loss: 0.6498 - val_accuracy: 0.8221
Epoch 12/15
500/500 - 2s - loss: 1.0613 - accuracy: 0.6885 - val_loss: 0.6574 - val_accuracy: 0.8188
Epoch 13/15
500/500 - 2s - loss: 1.0357 - accuracy: 0.6934 - val_loss: 0.6454 - val_accuracy: 0.8217
Epoch 14/15
500/500 - 2s - loss: 1.0537 - accuracy: 0.6881 - val_loss: 0.6395 - val_accuracy: 0.8230
Epoch 15/15
500/500 - 2s - loss: 1.0582 - accuracy: 0.6831 - val_loss: 0.6271 - val_accuracy: 0.8335
```

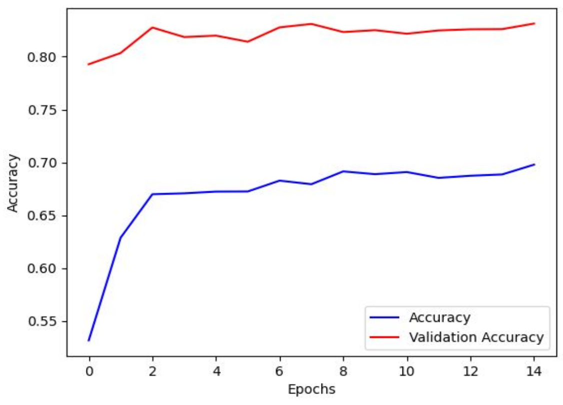
MODEL 2 :

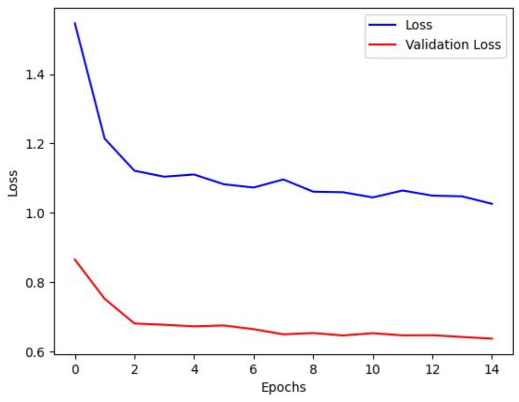
Architecture :

```
_____________________________________________________________
Layer (type)            Output Shape         Param #    Activation
=============================================================
flatten (Flatten)         (None, 784)          0
_____________________________________________________________
dense (Dense)              (None, 10)          7850        sigmoid
=============================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_____________________________________________________________
```

Training Accuracy	82.61 %
Training Loss	0.64
Testing Accuracy	83.92 %

Plots :





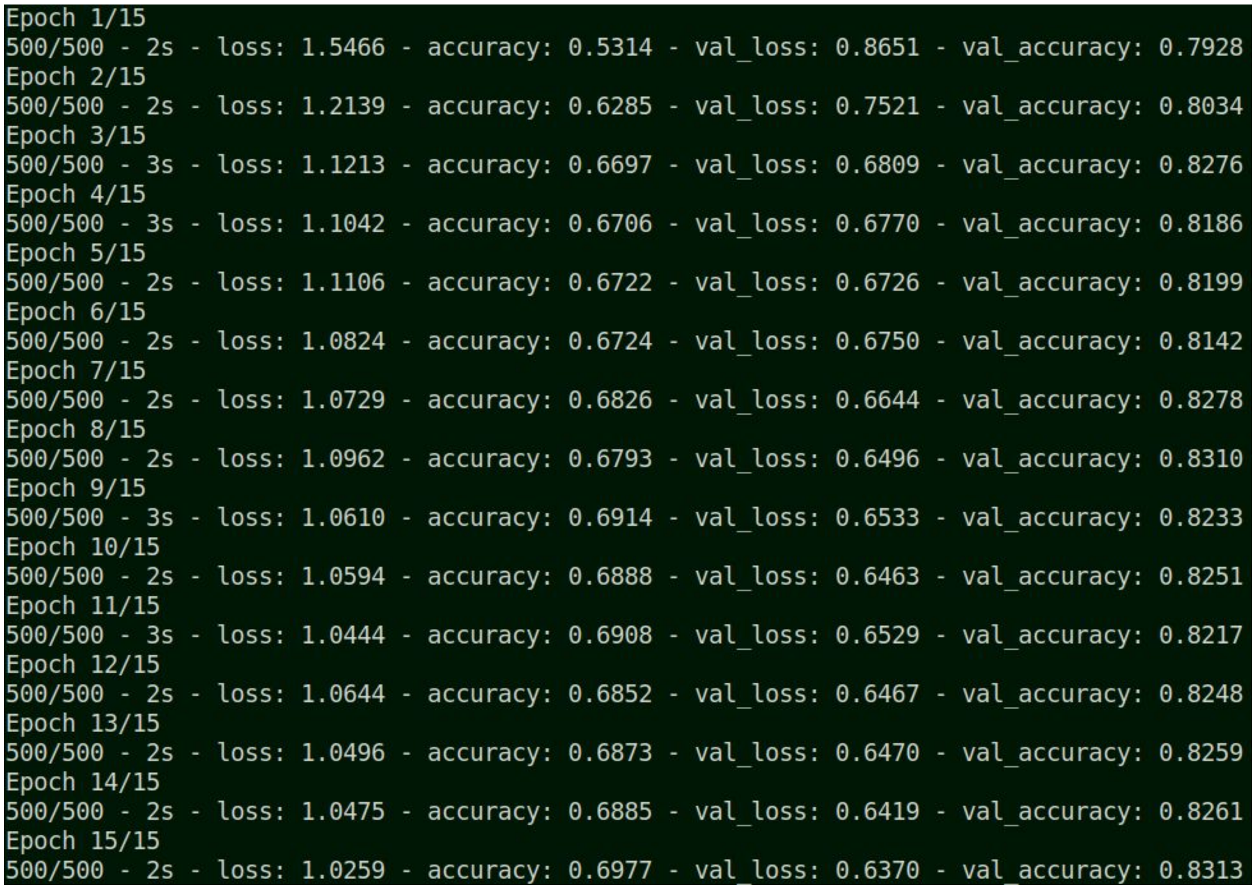
Conclusions:

This model also had only one layer and we tried to change its activation function to **sigmoid** from softmax. The results were not very impressive as the accuracy decreased in comparison to the previous model.

Possible Improvements:

We can try to add one more dense layer to the neural network which can improve its accuracy. So next we add another dense layer to the neural network.

Training Screenshot:



```
Epoch 1/15
500/500 - 2s - loss: 1.5466 - accuracy: 0.5314 - val_loss: 0.8651 - val_accuracy: 0.7928
Epoch 2/15
500/500 - 2s - loss: 1.2139 - accuracy: 0.6285 - val_loss: 0.7521 - val_accuracy: 0.8034
Epoch 3/15
500/500 - 3s - loss: 1.1213 - accuracy: 0.6697 - val_loss: 0.6809 - val_accuracy: 0.8276
Epoch 4/15
500/500 - 3s - loss: 1.1042 - accuracy: 0.6706 - val_loss: 0.6770 - val_accuracy: 0.8186
Epoch 5/15
500/500 - 2s - loss: 1.1106 - accuracy: 0.6722 - val_loss: 0.6726 - val_accuracy: 0.8199
Epoch 6/15
500/500 - 2s - loss: 1.0824 - accuracy: 0.6724 - val_loss: 0.6750 - val_accuracy: 0.8142
Epoch 7/15
500/500 - 2s - loss: 1.0729 - accuracy: 0.6826 - val_loss: 0.6644 - val_accuracy: 0.8278
Epoch 8/15
500/500 - 2s - loss: 1.0962 - accuracy: 0.6793 - val_loss: 0.6496 - val_accuracy: 0.8310
Epoch 9/15
500/500 - 3s - loss: 1.0610 - accuracy: 0.6914 - val_loss: 0.6533 - val_accuracy: 0.8233
Epoch 10/15
500/500 - 2s - loss: 1.0594 - accuracy: 0.6888 - val_loss: 0.6463 - val_accuracy: 0.8251
Epoch 11/15
500/500 - 3s - loss: 1.0444 - accuracy: 0.6908 - val_loss: 0.6529 - val_accuracy: 0.8217
Epoch 12/15
500/500 - 2s - loss: 1.0644 - accuracy: 0.6852 - val_loss: 0.6467 - val_accuracy: 0.8248
Epoch 13/15
500/500 - 2s - loss: 1.0496 - accuracy: 0.6873 - val_loss: 0.6470 - val_accuracy: 0.8259
Epoch 14/15
500/500 - 2s - loss: 1.0475 - accuracy: 0.6885 - val_loss: 0.6419 - val_accuracy: 0.8261
Epoch 15/15
500/500 - 2s - loss: 1.0259 - accuracy: 0.6977 - val_loss: 0.6370 - val_accuracy: 0.8313
```

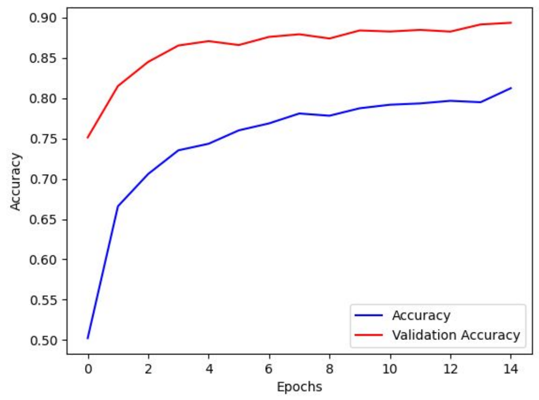
MODEL 3 :

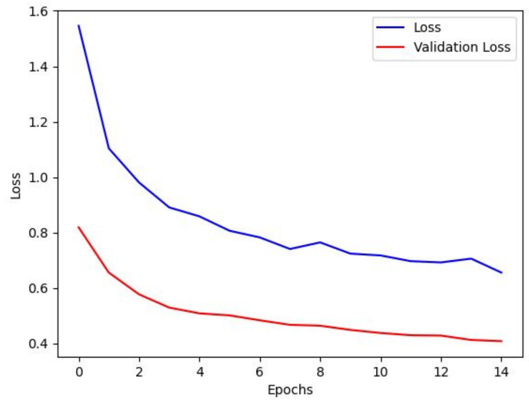
Architecture :

```
Layer (type)             Output Shape         Param #    Activation
=================================================================
flatten (Flatten)          (None, 784)          0

dense (Dense)               (None, 20)          15700      relu

dense_1 (Dense)              (None, 10)          210       sigmoid
=================================================================
Total params: 15,910
Trainable params: 15,910
Non-trainable params: 0
```

Training Accuracy	91.58 %
Training Loss	0.31
Testing Accuracy	91.96 %

Plots :





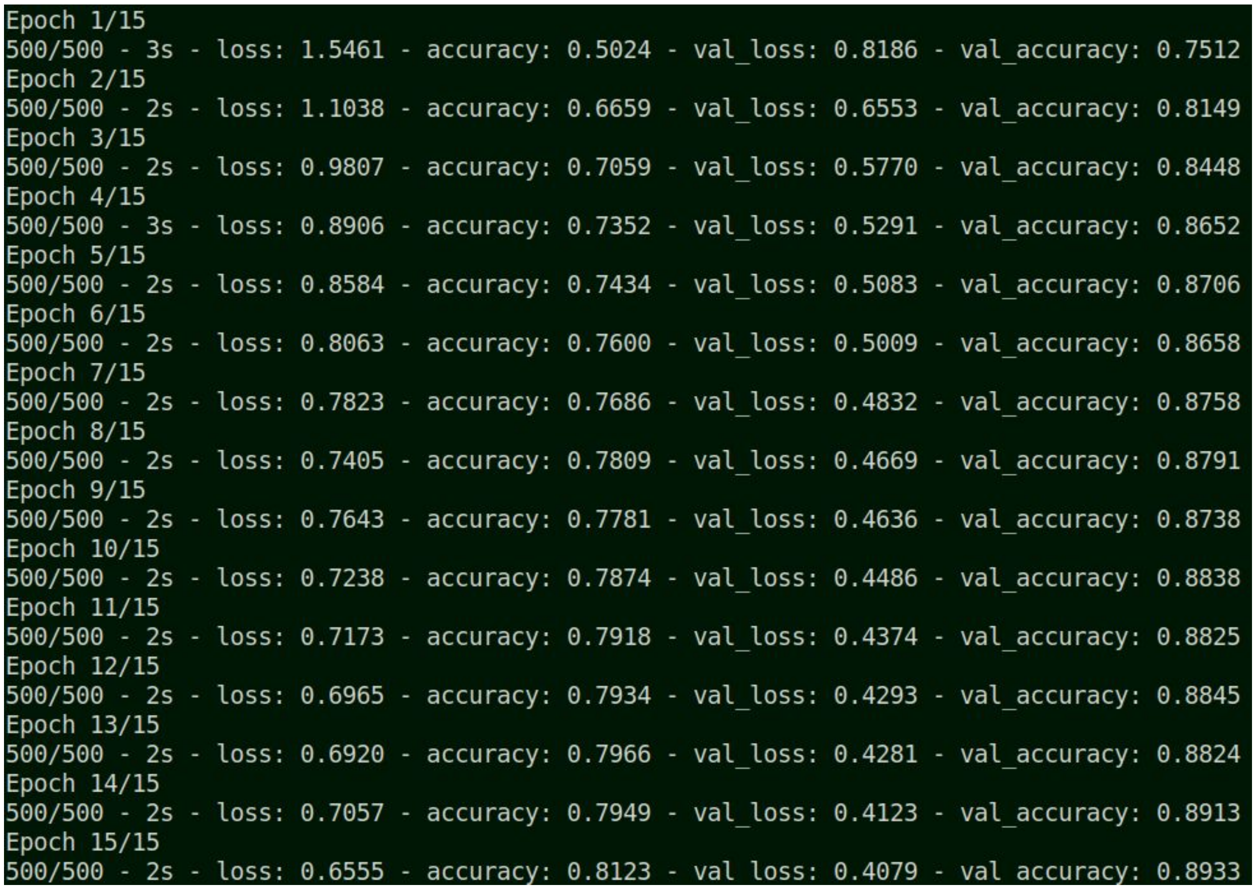
Conclusions:

We added another dense layer in this model with 20 nodes . The result is clearly visible. The testing accuracy significantly went up from 85 to around 92 %. Hence we can still add more layers till no improvement is visible.

Possible Improvements:

We can try changing the activation function from **relu** to **tanh**. Since **tanh** can give better results when used inside a shallow network.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.5461 - accuracy: 0.5024 - val_loss: 0.8186 - val_accuracy: 0.7512
Epoch 2/15
500/500 - 2s - loss: 1.1038 - accuracy: 0.6659 - val_loss: 0.6553 - val_accuracy: 0.8149
Epoch 3/15
500/500 - 2s - loss: 0.9807 - accuracy: 0.7059 - val_loss: 0.5770 - val_accuracy: 0.8448
Epoch 4/15
500/500 - 3s - loss: 0.8906 - accuracy: 0.7352 - val_loss: 0.5291 - val_accuracy: 0.8652
Epoch 5/15
500/500 - 2s - loss: 0.8584 - accuracy: 0.7434 - val_loss: 0.5083 - val_accuracy: 0.8706
Epoch 6/15
500/500 - 2s - loss: 0.8063 - accuracy: 0.7600 - val_loss: 0.5009 - val_accuracy: 0.8658
Epoch 7/15
500/500 - 2s - loss: 0.7823 - accuracy: 0.7686 - val_loss: 0.4832 - val_accuracy: 0.8758
Epoch 8/15
500/500 - 2s - loss: 0.7405 - accuracy: 0.7809 - val_loss: 0.4669 - val_accuracy: 0.8791
Epoch 9/15
500/500 - 2s - loss: 0.7643 - accuracy: 0.7781 - val_loss: 0.4636 - val_accuracy: 0.8738
Epoch 10/15
500/500 - 2s - loss: 0.7238 - accuracy: 0.7874 - val_loss: 0.4486 - val_accuracy: 0.8838
Epoch 11/15
500/500 - 2s - loss: 0.7173 - accuracy: 0.7918 - val_loss: 0.4374 - val_accuracy: 0.8825
Epoch 12/15
500/500 - 2s - loss: 0.6965 - accuracy: 0.7934 - val_loss: 0.4293 - val_accuracy: 0.8845
Epoch 13/15
500/500 - 2s - loss: 0.6920 - accuracy: 0.7966 - val_loss: 0.4281 - val_accuracy: 0.8824
Epoch 14/15
500/500 - 2s - loss: 0.7057 - accuracy: 0.7949 - val_loss: 0.4123 - val_accuracy: 0.8913
Epoch 15/15
500/500 - 2s - loss: 0.6555 - accuracy: 0.8123 - val_loss: 0.4079 - val_accuracy: 0.8933
```

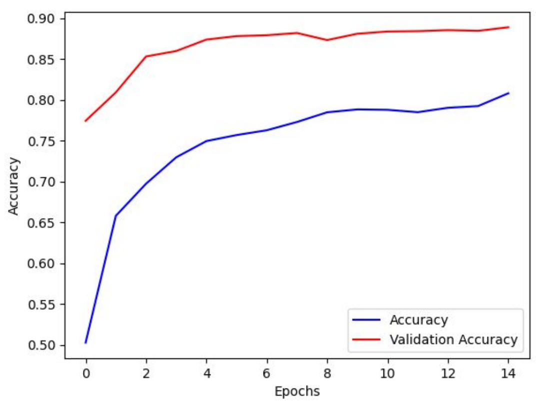
MODEL 4 :

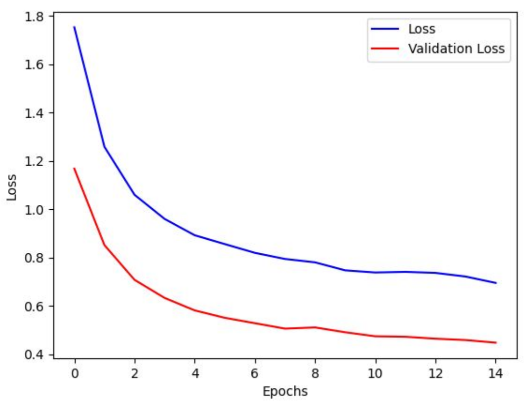
Architecture :

```
_________________________________________________________________
Layer (type)             Output Shape          Param #   Activation
=================================================================
flatten (Flatten)          (None, 784)         0
_________________________________________________________________
dense (Dense)               (None, 20)         15700      tanh
_________________________________________________________________
dense_1 (Dense)              (None, 10)         210        sigmoid
=================================================================
Total params: 15,910
Trainable params: 15,910
Non-trainable params: 0
_________________________________________________________________
```

Training Accuracy	88.88 %
Training Loss	0.46
Testing Accuracy	89.49 %

Plots :





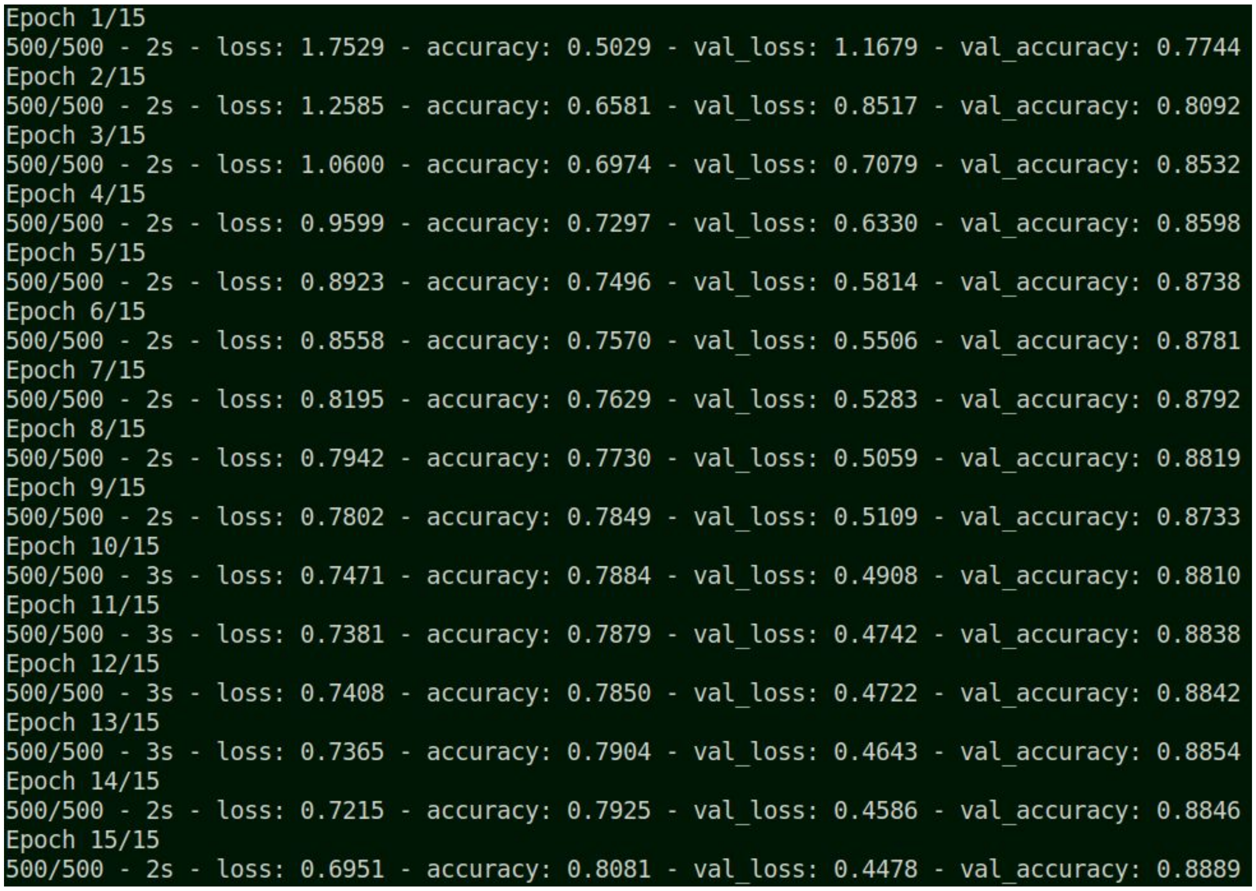
Conclusions:

The accuracy did not increase as was expected. Although it is good, a significant increase was not seen on changing the activation function from **relu** to **tanh**.

Possible Improvements:

We can still increase the number of layers in the model as we did last time and the accuracy of the model improved by around 6%.

Training Screenshot:



```
Epoch 1/15
500/500 - 2s - loss: 1.7529 - accuracy: 0.5029 - val_loss: 1.1679 - val_accuracy: 0.7744
Epoch 2/15
500/500 - 2s - loss: 1.2585 - accuracy: 0.6581 - val_loss: 0.8517 - val_accuracy: 0.8092
Epoch 3/15
500/500 - 2s - loss: 1.0600 - accuracy: 0.6974 - val_loss: 0.7079 - val_accuracy: 0.8532
Epoch 4/15
500/500 - 2s - loss: 0.9599 - accuracy: 0.7297 - val_loss: 0.6330 - val_accuracy: 0.8598
Epoch 5/15
500/500 - 2s - loss: 0.8923 - accuracy: 0.7496 - val_loss: 0.5814 - val_accuracy: 0.8738
Epoch 6/15
500/500 - 2s - loss: 0.8558 - accuracy: 0.7570 - val_loss: 0.5506 - val_accuracy: 0.8781
Epoch 7/15
500/500 - 2s - loss: 0.8195 - accuracy: 0.7629 - val_loss: 0.5283 - val_accuracy: 0.8792
Epoch 8/15
500/500 - 2s - loss: 0.7942 - accuracy: 0.7730 - val_loss: 0.5059 - val_accuracy: 0.8819
Epoch 9/15
500/500 - 2s - loss: 0.7802 - accuracy: 0.7849 - val_loss: 0.5109 - val_accuracy: 0.8733
Epoch 10/15
500/500 - 3s - loss: 0.7471 - accuracy: 0.7884 - val_loss: 0.4908 - val_accuracy: 0.8810
Epoch 11/15
500/500 - 3s - loss: 0.7381 - accuracy: 0.7879 - val_loss: 0.4742 - val_accuracy: 0.8838
Epoch 12/15
500/500 - 3s - loss: 0.7408 - accuracy: 0.7850 - val_loss: 0.4722 - val_accuracy: 0.8842
Epoch 13/15
500/500 - 3s - loss: 0.7365 - accuracy: 0.7904 - val_loss: 0.4643 - val_accuracy: 0.8854
Epoch 14/15
500/500 - 2s - loss: 0.7215 - accuracy: 0.7925 - val_loss: 0.4586 - val_accuracy: 0.8846
Epoch 15/15
500/500 - 2s - loss: 0.6951 - accuracy: 0.8081 - val_loss: 0.4478 - val_accuracy: 0.8889
```

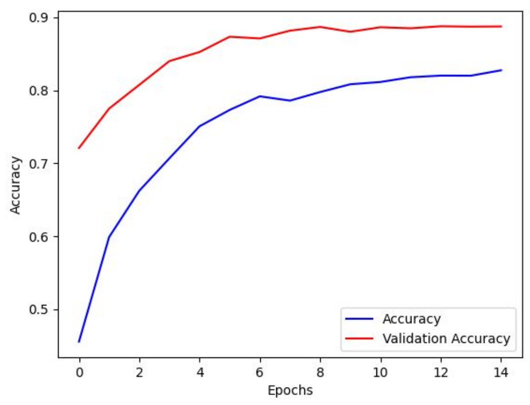
MODEL 5 :

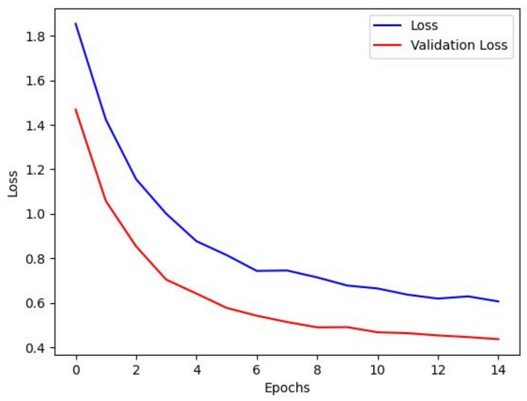
Architecture :

```
Layer (type)              Output Shape         Param #     Activation
=====================================================================
flatten (Flatten)           (None, 784)          0

dense (Dense)                (None, 20)          15700       tanh

dense_1 (Dense)              (None, 15)          315         tanh

dense_2 (Dense)              (None, 10)           160        sigmoid
===================================================================
Total params: 16,175
Trainable params: 16,175
Non-trainable params: 0
```

Training Accuracy	89.03 %
Training Loss	0.43
Testing Accuracy	90.40 %

Plots :





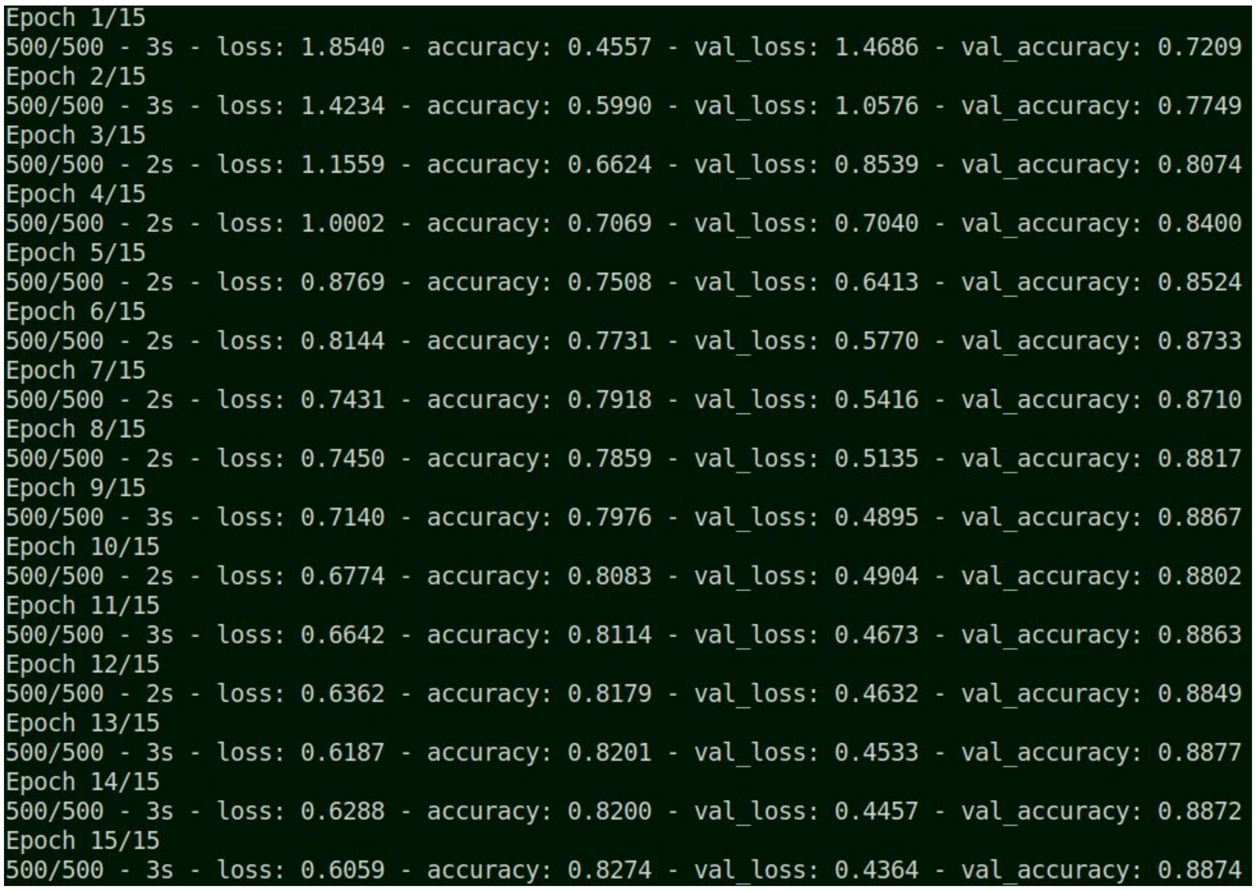
Conclusions:

As expected on increasing the dense layers from two to three, the accuracy of the model increased by about 2%.

Possible Improvements:

As we have seen from the last two models, the **tanh** activation function did not work as expected. So we can try using **softmax** for the last layer and later on try using the **relu** function as well.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.8540 - accuracy: 0.4557 - val_loss: 1.4686 - val_accuracy: 0.7209
Epoch 2/15
500/500 - 3s - loss: 1.4234 - accuracy: 0.5990 - val_loss: 1.0576 - val_accuracy: 0.7749
Epoch 3/15
500/500 - 2s - loss: 1.1559 - accuracy: 0.6624 - val_loss: 0.8539 - val_accuracy: 0.8074
Epoch 4/15
500/500 - 2s - loss: 1.0002 - accuracy: 0.7069 - val_loss: 0.7040 - val_accuracy: 0.8400
Epoch 5/15
500/500 - 2s - loss: 0.8769 - accuracy: 0.7508 - val_loss: 0.6413 - val_accuracy: 0.8524
Epoch 6/15
500/500 - 2s - loss: 0.8144 - accuracy: 0.7731 - val_loss: 0.5770 - val_accuracy: 0.8733
Epoch 7/15
500/500 - 2s - loss: 0.7431 - accuracy: 0.7918 - val_loss: 0.5416 - val_accuracy: 0.8710
Epoch 8/15
500/500 - 2s - loss: 0.7450 - accuracy: 0.7859 - val_loss: 0.5135 - val_accuracy: 0.8817
Epoch 9/15
500/500 - 3s - loss: 0.7140 - accuracy: 0.7976 - val_loss: 0.4895 - val_accuracy: 0.8867
Epoch 10/15
500/500 - 2s - loss: 0.6774 - accuracy: 0.8083 - val_loss: 0.4904 - val_accuracy: 0.8802
Epoch 11/15
500/500 - 3s - loss: 0.6642 - accuracy: 0.8114 - val_loss: 0.4673 - val_accuracy: 0.8863
Epoch 12/15
500/500 - 2s - loss: 0.6362 - accuracy: 0.8179 - val_loss: 0.4632 - val_accuracy: 0.8849
Epoch 13/15
500/500 - 3s - loss: 0.6187 - accuracy: 0.8201 - val_loss: 0.4533 - val_accuracy: 0.8877
Epoch 14/15
500/500 - 3s - loss: 0.6288 - accuracy: 0.8200 - val_loss: 0.4457 - val_accuracy: 0.8872
Epoch 15/15
500/500 - 3s - loss: 0.6059 - accuracy: 0.8274 - val_loss: 0.4364 - val_accuracy: 0.8874
```

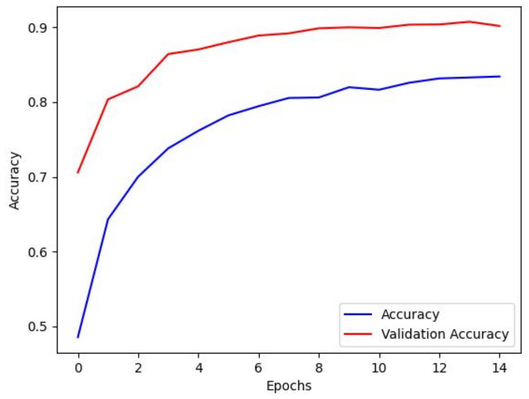
MODEL 6 :

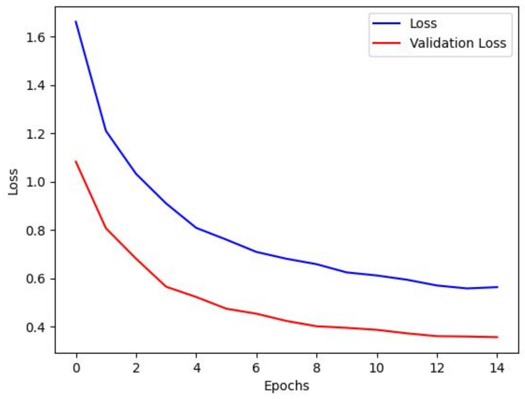
Architecture :

```
_________________________________________________________________
Layer (type)            Output Shape           Param #   Activation
=================================================================
flatten (Flatten)         (None, 784)           0
_________________________________________________________________
dense (Dense)              (None, 20)           15700       tanh
_________________________________________________________________
dense_1 (Dense)             (None, 10)           210        tanh
_________________________________________________________________
dense_2 (Dense)             (None, 10)           110        softmax
=================================================================
Total params: 16,020
Trainable params: 16,020
Non-trainable params: 0
_________________________________________________________________
```

Training Accuracy	90.82 %
Training Loss	0.37
Testing Accuracy	90.9 %

Plots :





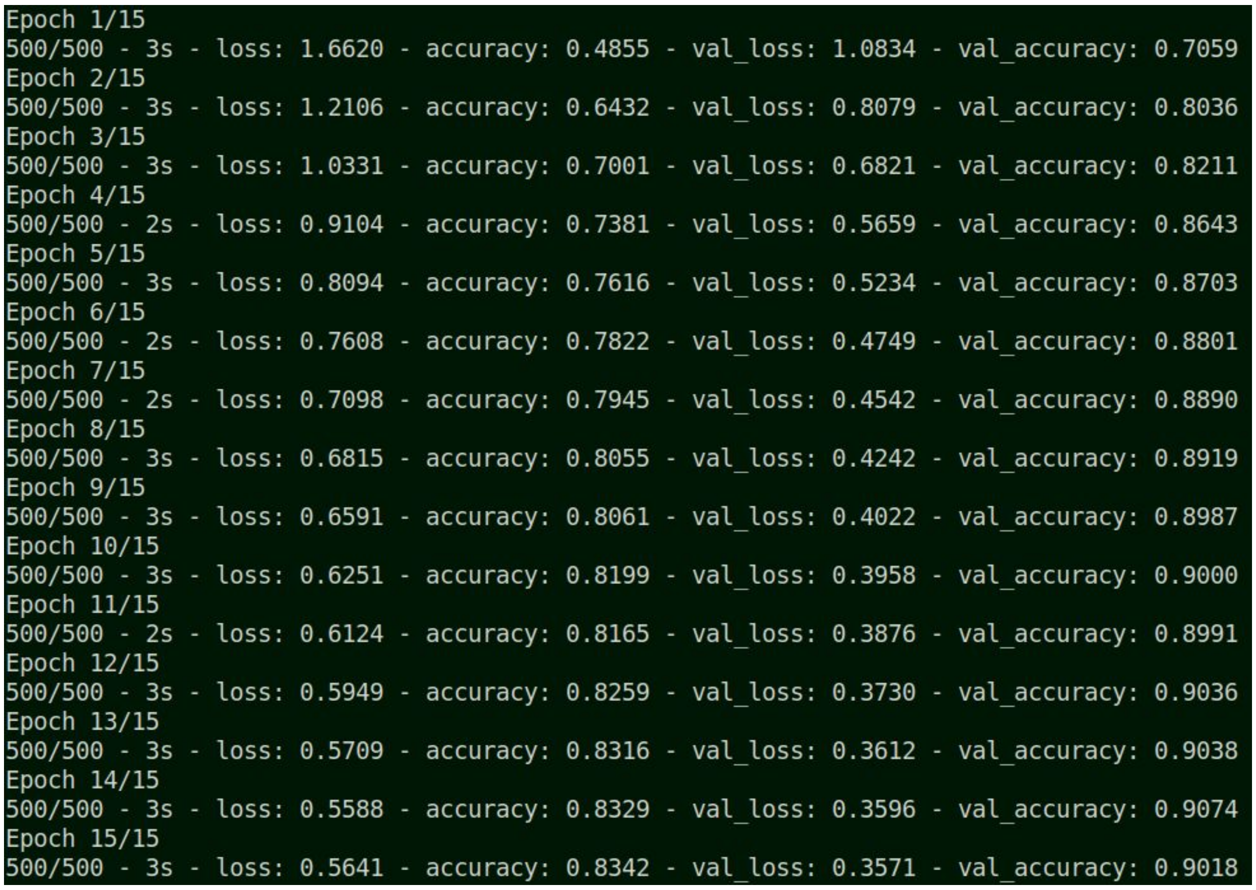
Conclusions:

On using **softmax** for the last layer, the accuracy did increase.

Possible Improvements:

We can now try replacing the **tanh** with **relu**. Since **relu** generally has been giving better accuracy results for the previous models.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.6620 - accuracy: 0.4855 - val_loss: 1.0834 - val_accuracy: 0.7059
Epoch 2/15
500/500 - 3s - loss: 1.2106 - accuracy: 0.6432 - val_loss: 0.8079 - val_accuracy: 0.8036
Epoch 3/15
500/500 - 3s - loss: 1.0331 - accuracy: 0.7001 - val_loss: 0.6821 - val_accuracy: 0.8211
Epoch 4/15
500/500 - 2s - loss: 0.9104 - accuracy: 0.7381 - val_loss: 0.5659 - val_accuracy: 0.8643
Epoch 5/15
500/500 - 3s - loss: 0.8094 - accuracy: 0.7616 - val_loss: 0.5234 - val_accuracy: 0.8703
Epoch 6/15
500/500 - 2s - loss: 0.7608 - accuracy: 0.7822 - val_loss: 0.4749 - val_accuracy: 0.8801
Epoch 7/15
500/500 - 2s - loss: 0.7098 - accuracy: 0.7945 - val_loss: 0.4542 - val_accuracy: 0.8890
Epoch 8/15
500/500 - 3s - loss: 0.6815 - accuracy: 0.8055 - val_loss: 0.4242 - val_accuracy: 0.8919
Epoch 9/15
500/500 - 3s - loss: 0.6591 - accuracy: 0.8061 - val_loss: 0.4022 - val_accuracy: 0.8987
Epoch 10/15
500/500 - 3s - loss: 0.6251 - accuracy: 0.8199 - val_loss: 0.3958 - val_accuracy: 0.9000
Epoch 11/15
500/500 - 2s - loss: 0.6124 - accuracy: 0.8165 - val_loss: 0.3876 - val_accuracy: 0.8991
Epoch 12/15
500/500 - 3s - loss: 0.5949 - accuracy: 0.8259 - val_loss: 0.3730 - val_accuracy: 0.9036
Epoch 13/15
500/500 - 3s - loss: 0.5709 - accuracy: 0.8316 - val_loss: 0.3612 - val_accuracy: 0.9038
Epoch 14/15
500/500 - 3s - loss: 0.5588 - accuracy: 0.8329 - val_loss: 0.3596 - val_accuracy: 0.9074
Epoch 15/15
500/500 - 3s - loss: 0.5641 - accuracy: 0.8342 - val_loss: 0.3571 - val_accuracy: 0.9018
```

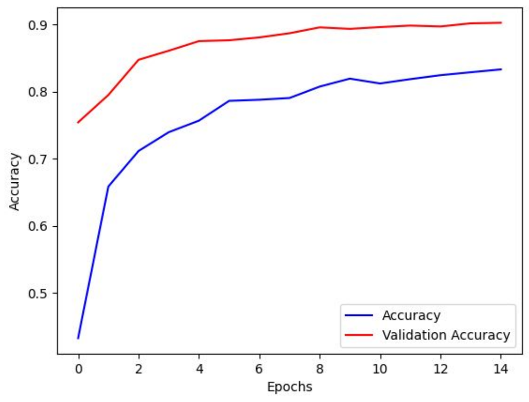
MODEL 7 :

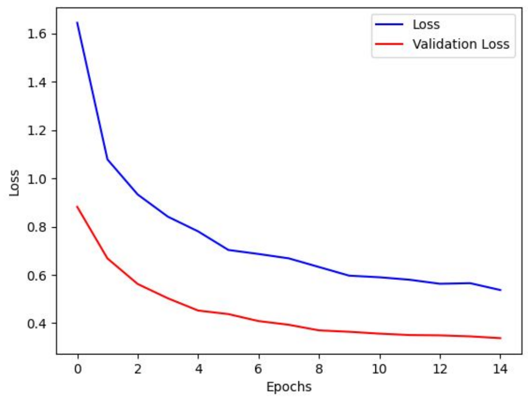
Architecture :

```
Layer (type)             Output Shape          Param #   Activation
=================================================================
flatten (Flatten)          (None, 784)          0

dense (Dense)               (None, 20)          15700        relu

dense_1 (Dense)              (None, 10)          210         relu

dense_2 (Dense)              (None, 10)          110         softmax
=================================================================
Total params: 16,020
Trainable params: 16,020
Non-trainable params: 0
```

Training Accuracy	90.22 %
Training Loss	0.36
Testing Accuracy	90.49 %

Plots :





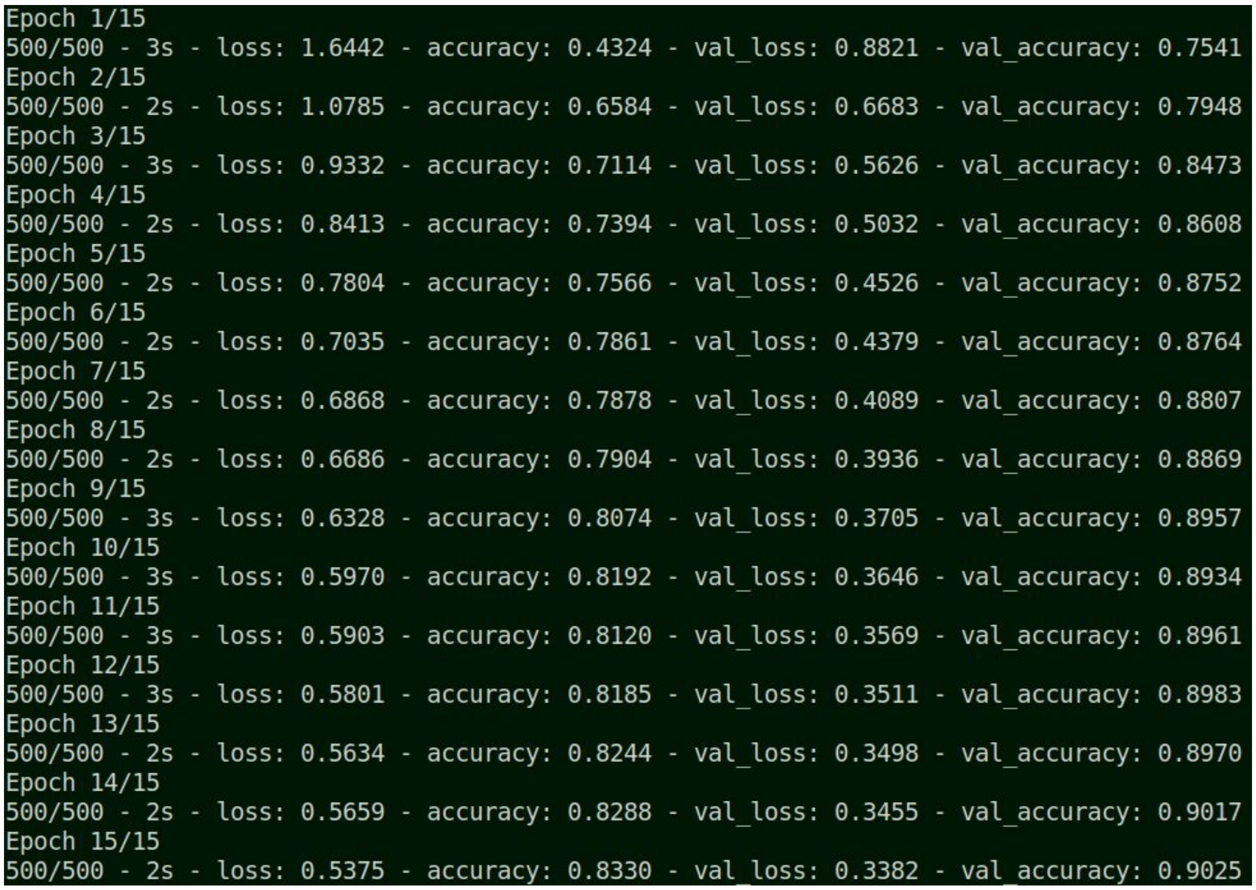
Conclusions:

Changing all the hidden layers' activation function to **relu** the accuracy increased by a small amount.

Possible Improvements:

We have tried changing the activation functions and number of layers till now with not much improvement in accuracy. Let us now try to change the width of the layers.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.6442 - accuracy: 0.4324 - val_loss: 0.8821 - val_accuracy: 0.7541
Epoch 2/15
500/500 - 2s - loss: 1.0785 - accuracy: 0.6584 - val_loss: 0.6683 - val_accuracy: 0.7948
Epoch 3/15
500/500 - 3s - loss: 0.9332 - accuracy: 0.7114 - val_loss: 0.5626 - val_accuracy: 0.8473
Epoch 4/15
500/500 - 2s - loss: 0.8413 - accuracy: 0.7394 - val_loss: 0.5032 - val_accuracy: 0.8608
Epoch 5/15
500/500 - 2s - loss: 0.7804 - accuracy: 0.7566 - val_loss: 0.4526 - val_accuracy: 0.8752
Epoch 6/15
500/500 - 2s - loss: 0.7035 - accuracy: 0.7861 - val_loss: 0.4379 - val_accuracy: 0.8764
Epoch 7/15
500/500 - 2s - loss: 0.6868 - accuracy: 0.7878 - val_loss: 0.4089 - val_accuracy: 0.8807
Epoch 8/15
500/500 - 2s - loss: 0.6686 - accuracy: 0.7904 - val_loss: 0.3936 - val_accuracy: 0.8869
Epoch 9/15
500/500 - 3s - loss: 0.6328 - accuracy: 0.8074 - val_loss: 0.3705 - val_accuracy: 0.8957
Epoch 10/15
500/500 - 3s - loss: 0.5970 - accuracy: 0.8192 - val_loss: 0.3646 - val_accuracy: 0.8934
Epoch 11/15
500/500 - 3s - loss: 0.5903 - accuracy: 0.8120 - val_loss: 0.3569 - val_accuracy: 0.8961
Epoch 12/15
500/500 - 3s - loss: 0.5801 - accuracy: 0.8185 - val_loss: 0.3511 - val_accuracy: 0.8983
Epoch 13/15
500/500 - 2s - loss: 0.5634 - accuracy: 0.8244 - val_loss: 0.3498 - val_accuracy: 0.8970
Epoch 14/15
500/500 - 2s - loss: 0.5659 - accuracy: 0.8288 - val_loss: 0.3455 - val_accuracy: 0.9017
Epoch 15/15
500/500 - 2s - loss: 0.5375 - accuracy: 0.8330 - val_loss: 0.3382 - val_accuracy: 0.9025
```

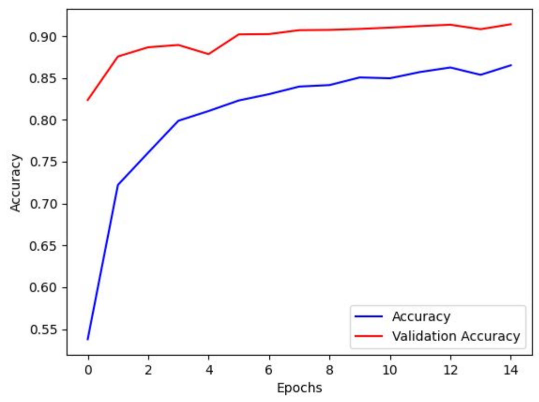
MODEL 8 :

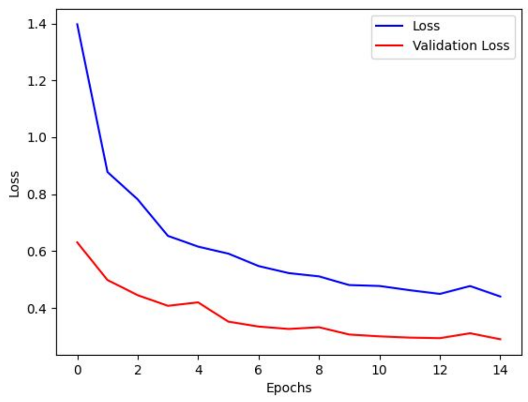
Architecture :

```
Layer (type)           Output Shape          Param #          Activation
==============================================================
flatten (Flatten)       (None, 784)          0

dense (Dense)            (None, 20)          15700        relu

dense_1 (Dense)           (None, 40)          840         relu

dense_2 (Dense)           (None, 10)          410         softmax
==============================================================
Total params: 16,950
Trainable params: 16,950
Non-trainable params: 0
```

Training Accuracy	92.52 %
Training Loss	0.27
Testing Accuracy	92.52 %

Plots :





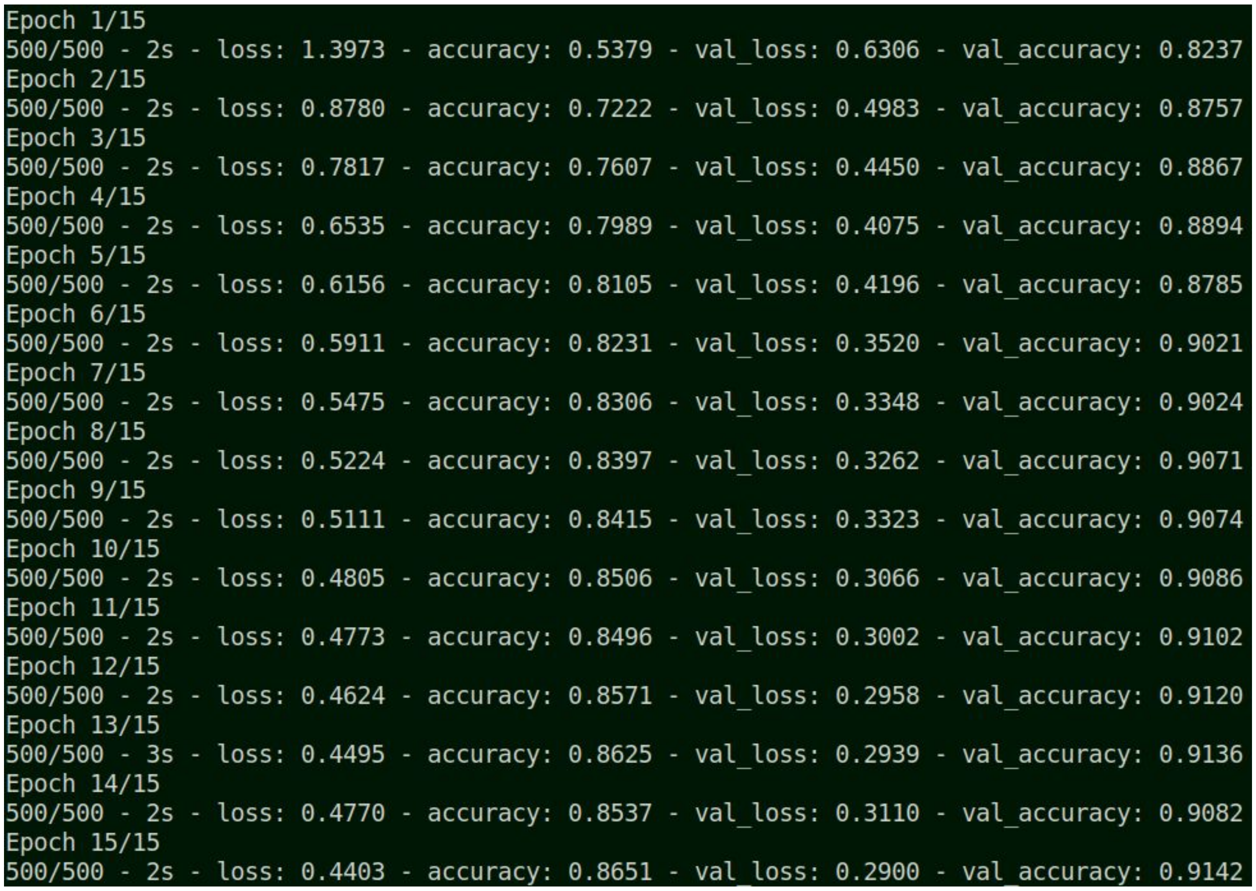
Conclusions:

The accuracy improved by a good amount of 2% after increasing the number of nodes in the hidden layer. This is because more hyperparameters can capture more features from the input and hence increase the accuracy.

Possible Improvements:

We can still try increasing the number of nodes for the hidden layers of the model.

Training Screenshot:



```
Epoch 1/15
500/500 - 2s - loss: 1.3973 - accuracy: 0.5379 - val_loss: 0.6306 - val_accuracy: 0.8237
Epoch 2/15
500/500 - 2s - loss: 0.8780 - accuracy: 0.7222 - val_loss: 0.4983 - val_accuracy: 0.8757
Epoch 3/15
500/500 - 2s - loss: 0.7817 - accuracy: 0.7607 - val_loss: 0.4450 - val_accuracy: 0.8867
Epoch 4/15
500/500 - 2s - loss: 0.6535 - accuracy: 0.7989 - val_loss: 0.4075 - val_accuracy: 0.8894
Epoch 5/15
500/500 - 2s - loss: 0.6156 - accuracy: 0.8105 - val_loss: 0.4196 - val_accuracy: 0.8785
Epoch 6/15
500/500 - 2s - loss: 0.5911 - accuracy: 0.8231 - val_loss: 0.3520 - val_accuracy: 0.9021
Epoch 7/15
500/500 - 2s - loss: 0.5475 - accuracy: 0.8306 - val_loss: 0.3348 - val_accuracy: 0.9024
Epoch 8/15
500/500 - 2s - loss: 0.5224 - accuracy: 0.8397 - val_loss: 0.3262 - val_accuracy: 0.9071
Epoch 9/15
500/500 - 2s - loss: 0.5111 - accuracy: 0.8415 - val_loss: 0.3323 - val_accuracy: 0.9074
Epoch 10/15
500/500 - 2s - loss: 0.4805 - accuracy: 0.8506 - val_loss: 0.3066 - val_accuracy: 0.9086
Epoch 11/15
500/500 - 2s - loss: 0.4773 - accuracy: 0.8496 - val_loss: 0.3002 - val_accuracy: 0.9102
Epoch 12/15
500/500 - 2s - loss: 0.4624 - accuracy: 0.8571 - val_loss: 0.2958 - val_accuracy: 0.9120
Epoch 13/15
500/500 - 3s - loss: 0.4495 - accuracy: 0.8625 - val_loss: 0.2939 - val_accuracy: 0.9136
Epoch 14/15
500/500 - 2s - loss: 0.4770 - accuracy: 0.8537 - val_loss: 0.3110 - val_accuracy: 0.9082
Epoch 15/15
500/500 - 2s - loss: 0.4403 - accuracy: 0.8651 - val_loss: 0.2900 - val_accuracy: 0.9142
```

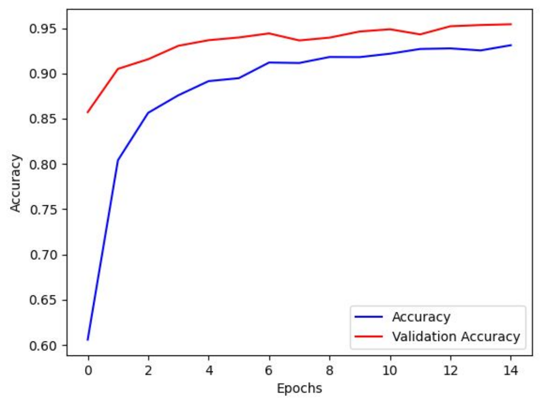
MODEL 9 :

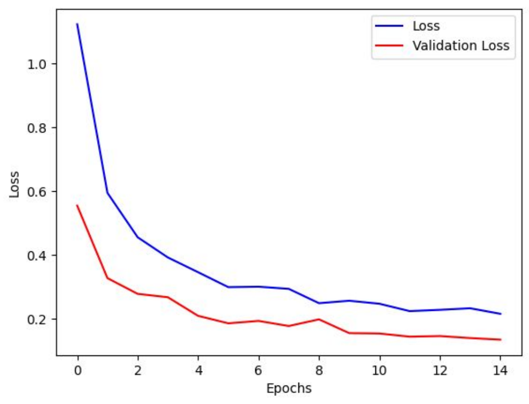
Architecture :

```
Layer (type)              Output Shape          Param #   Activation
=========================================================================
flatten (Flatten)           (None, 784)         0

dense (Dense)                (None, 50)         39250       relu

dense_1 (Dense)               (None, 100)        5100       relu

dense_2 (Dense)               (None, 10)         1010       softmax
=========================================================================
Total params: 45,360
Trainable params: 45,360
Non-trainable params: 0
```

Training Accuracy	95.34 %
Training Loss	0.16
Testing Accuracy	95.57 %

Plots :





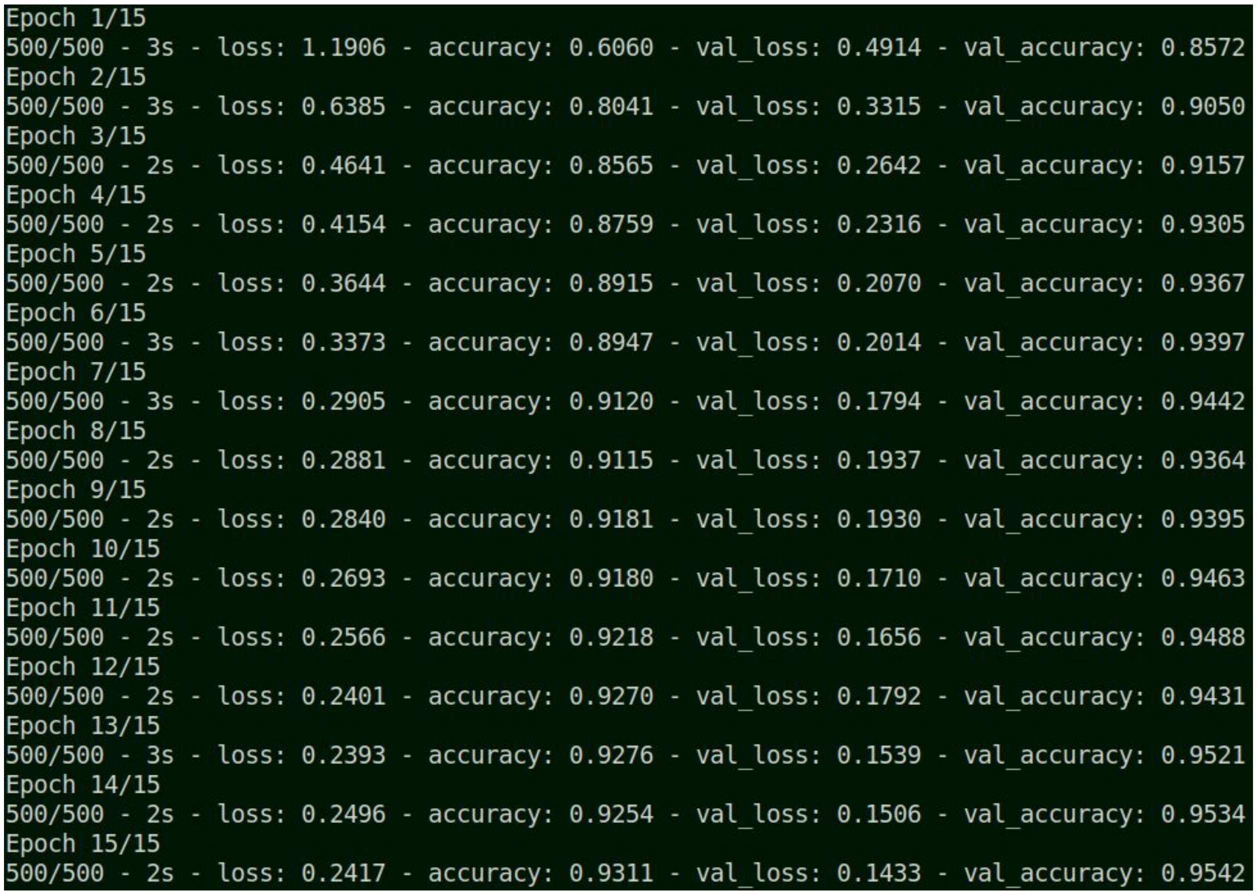
Conclusions:

Increasing the number of nodes to 50 and 100 increases the accuracy by a whopping amount. Hence it is more apparent now that increasing the number of nodes in the hidden layers can help increase the accuracy of the model.

Possible Improvements:

Since more layers and more nodes has been increasing the accuracy of the model till now, we can do both of them at the same time now. We try adding another layer with 200 nodes.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.1906 - accuracy: 0.6060 - val_loss: 0.4914 - val_accuracy: 0.8572
Epoch 2/15
500/500 - 3s - loss: 0.6385 - accuracy: 0.8041 - val_loss: 0.3315 - val_accuracy: 0.9050
Epoch 3/15
500/500 - 2s - loss: 0.4641 - accuracy: 0.8565 - val_loss: 0.2642 - val_accuracy: 0.9157
Epoch 4/15
500/500 - 2s - loss: 0.4154 - accuracy: 0.8759 - val_loss: 0.2316 - val_accuracy: 0.9305
Epoch 5/15
500/500 - 2s - loss: 0.3644 - accuracy: 0.8915 - val_loss: 0.2070 - val_accuracy: 0.9367
Epoch 6/15
500/500 - 3s - loss: 0.3373 - accuracy: 0.8947 - val_loss: 0.2014 - val_accuracy: 0.9397
Epoch 7/15
500/500 - 3s - loss: 0.2905 - accuracy: 0.9120 - val_loss: 0.1794 - val_accuracy: 0.9442
Epoch 8/15
500/500 - 2s - loss: 0.2881 - accuracy: 0.9115 - val_loss: 0.1937 - val_accuracy: 0.9364
Epoch 9/15
500/500 - 2s - loss: 0.2840 - accuracy: 0.9181 - val_loss: 0.1930 - val_accuracy: 0.9395
Epoch 10/15
500/500 - 2s - loss: 0.2693 - accuracy: 0.9180 - val_loss: 0.1710 - val_accuracy: 0.9463
Epoch 11/15
500/500 - 2s - loss: 0.2566 - accuracy: 0.9218 - val_loss: 0.1656 - val_accuracy: 0.9488
Epoch 12/15
500/500 - 2s - loss: 0.2401 - accuracy: 0.9270 - val_loss: 0.1792 - val_accuracy: 0.9431
Epoch 13/15
500/500 - 3s - loss: 0.2393 - accuracy: 0.9276 - val_loss: 0.1539 - val_accuracy: 0.9521
Epoch 14/15
500/500 - 2s - loss: 0.2496 - accuracy: 0.9254 - val_loss: 0.1506 - val_accuracy: 0.9534
Epoch 15/15
500/500 - 2s - loss: 0.2417 - accuracy: 0.9311 - val_loss: 0.1433 - val_accuracy: 0.9542
```

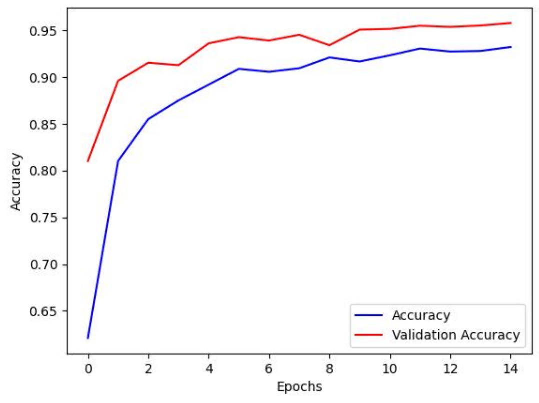
MODEL 10 :

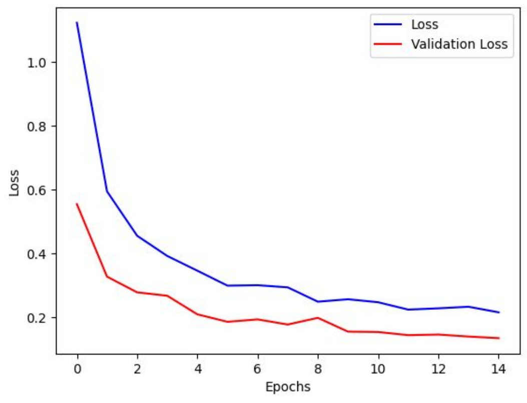
Architecture :

```
Layer (type)              Output Shape          Param #   Activation
==================================================================
flatten (Flatten)           (None, 784)           0

dense (Dense)                (None, 50)           39250       relu

dense_1 (Dense)               (None, 100)           5100       relu

dense_2 (Dense)               (None, 200)            20200     relu

dense_3 (Dense)               (None, 10)            2010       softmax
==================================================================
Total params: 66,560
Trainable params: 66,560
Non-trainable params: 0
```

Training Accuracy	96.01 %
Training Loss	0.12
Testing Accuracy	96.19 %

Plots :





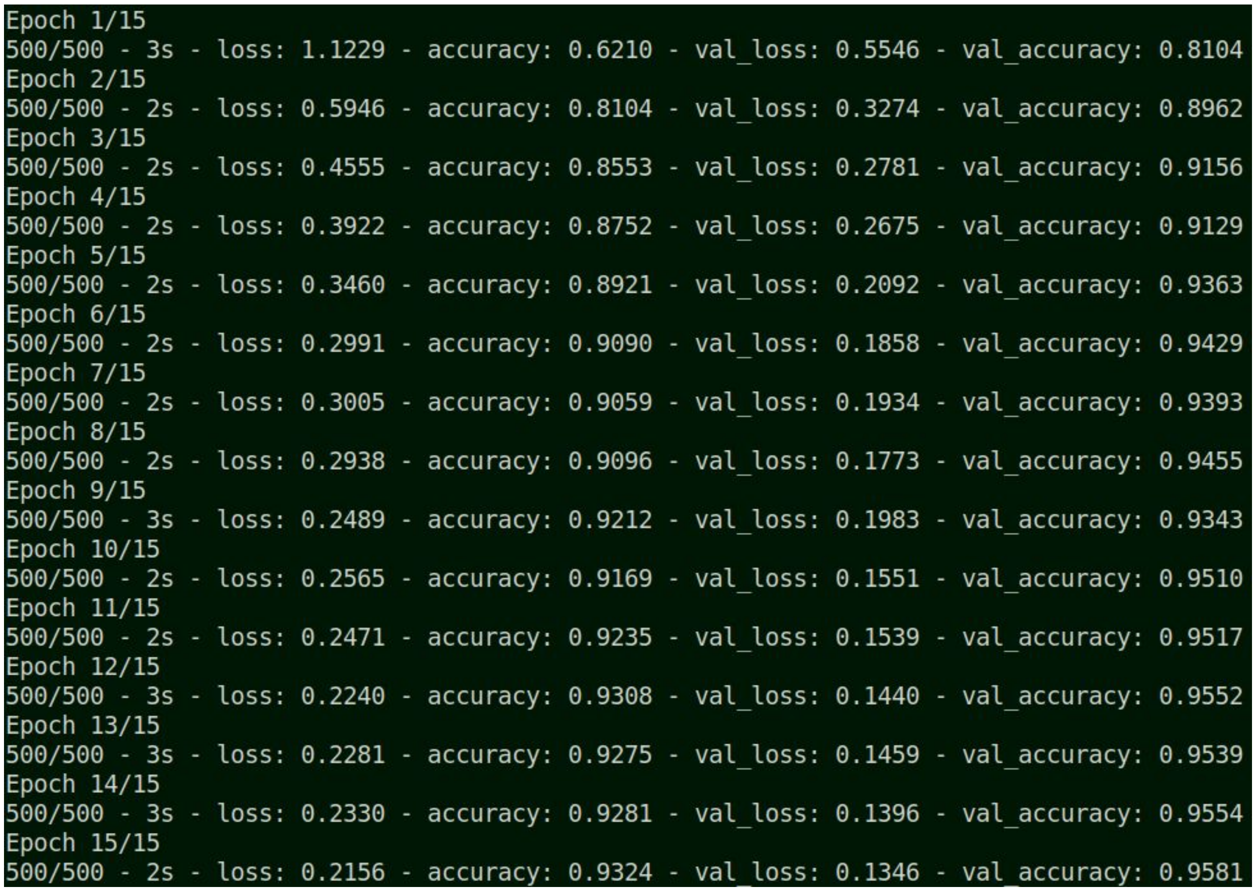
Conclusions:

Adding the fourth layer with 200 nodes, increases the accuracy as expected by around 1%.

Possible Improvements:

We can still try adding one more layer with around 200 nodes to increase the accuracy of the model.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.1229 - accuracy: 0.6210 - val_loss: 0.5546 - val_accuracy: 0.8104
Epoch 2/15
500/500 - 2s - loss: 0.5946 - accuracy: 0.8104 - val_loss: 0.3274 - val_accuracy: 0.8962
Epoch 3/15
500/500 - 2s - loss: 0.4555 - accuracy: 0.8553 - val_loss: 0.2781 - val_accuracy: 0.9156
Epoch 4/15
500/500 - 2s - loss: 0.3922 - accuracy: 0.8752 - val_loss: 0.2675 - val_accuracy: 0.9129
Epoch 5/15
500/500 - 2s - loss: 0.3460 - accuracy: 0.8921 - val_loss: 0.2092 - val_accuracy: 0.9363
Epoch 6/15
500/500 - 2s - loss: 0.2991 - accuracy: 0.9090 - val_loss: 0.1858 - val_accuracy: 0.9429
Epoch 7/15
500/500 - 2s - loss: 0.3005 - accuracy: 0.9059 - val_loss: 0.1934 - val_accuracy: 0.9393
Epoch 8/15
500/500 - 2s - loss: 0.2938 - accuracy: 0.9096 - val_loss: 0.1773 - val_accuracy: 0.9455
Epoch 9/15
500/500 - 3s - loss: 0.2489 - accuracy: 0.9212 - val_loss: 0.1983 - val_accuracy: 0.9343
Epoch 10/15
500/500 - 2s - loss: 0.2565 - accuracy: 0.9169 - val_loss: 0.1551 - val_accuracy: 0.9510
Epoch 11/15
500/500 - 2s - loss: 0.2471 - accuracy: 0.9235 - val_loss: 0.1539 - val_accuracy: 0.9517
Epoch 12/15
500/500 - 3s - loss: 0.2240 - accuracy: 0.9308 - val_loss: 0.1440 - val_accuracy: 0.9552
Epoch 13/15
500/500 - 3s - loss: 0.2281 - accuracy: 0.9275 - val_loss: 0.1459 - val_accuracy: 0.9539
Epoch 14/15
500/500 - 3s - loss: 0.2330 - accuracy: 0.9281 - val_loss: 0.1396 - val_accuracy: 0.9554
Epoch 15/15
500/500 - 2s - loss: 0.2156 - accuracy: 0.9324 - val_loss: 0.1346 - val_accuracy: 0.9581
```

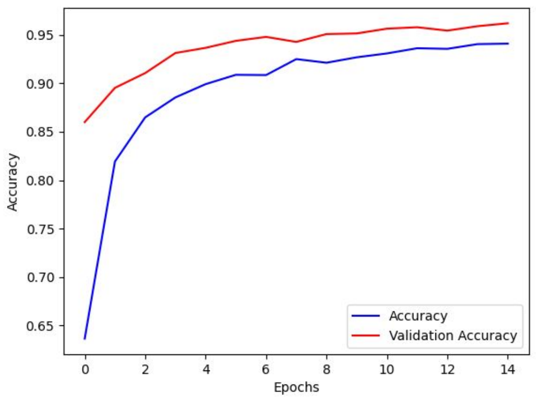
MODEL 11 :

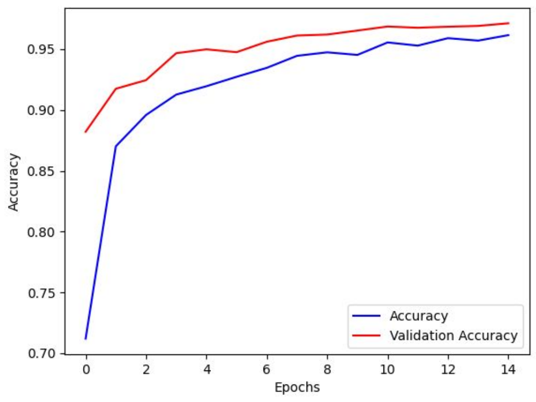
Architecture :

```
Layer (type)              Output Shape          Param #   Activation
==========================================================
flatten (Flatten)          (None, 784)          0

dense (Dense)               (None, 50)          39250       relu

dense_1 (Dense)             (None, 100)          5100       relu

dense_2 (Dense)             (None, 200)          20200      relu

dense_3 (Dense)             (None, 400)          80400      relu

dense_4 (Dense)             (None, 10)           4010       softmax
==========================================================
Total params: 148,960
Trainable params: 148,960
Non-trainable params: 0
```

Training Accuracy	96.0 %
Training Loss	0.13
Testing Accuracy	96.22 %

Plots :





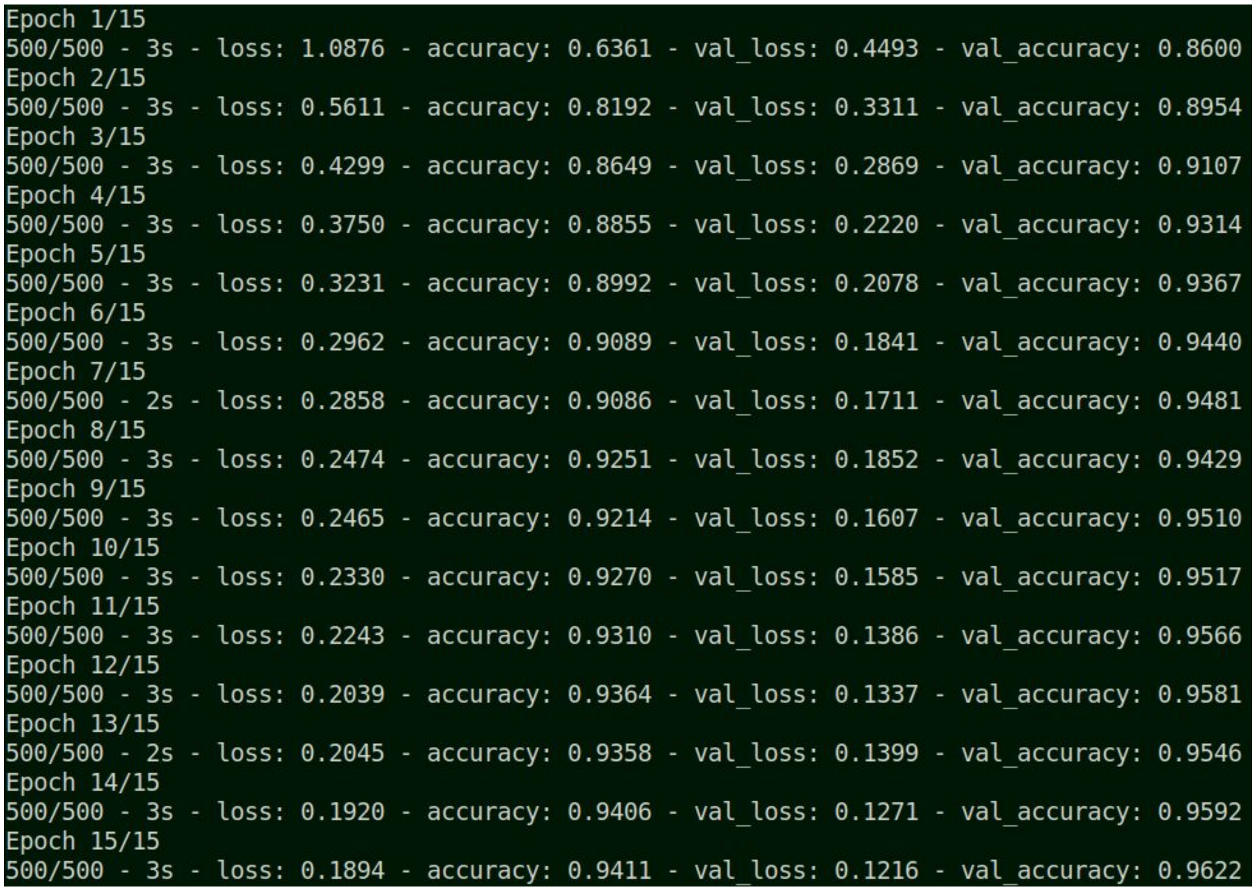
Conclusions:

Increasing the number of layers till 4 was increasing the accuracy by a good amount. But adding another layer did not help much. The accuracy remained almost the same. Hence adding a fifth layer leads to **overfitting.**

Possible Improvements:

We can still try to increase the number of nodes in the hidden layers keeping the number of layers to be the same i.e 4.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 1.0876 - accuracy: 0.6361 - val_loss: 0.4493 - val_accuracy: 0.8600
Epoch 2/15
500/500 - 3s - loss: 0.5611 - accuracy: 0.8192 - val_loss: 0.3311 - val_accuracy: 0.8954
Epoch 3/15
500/500 - 3s - loss: 0.4299 - accuracy: 0.8649 - val_loss: 0.2869 - val_accuracy: 0.9107
Epoch 4/15
500/500 - 3s - loss: 0.3750 - accuracy: 0.8855 - val_loss: 0.2220 - val_accuracy: 0.9314
Epoch 5/15
500/500 - 3s - loss: 0.3231 - accuracy: 0.8992 - val_loss: 0.2078 - val_accuracy: 0.9367
Epoch 6/15
500/500 - 3s - loss: 0.2962 - accuracy: 0.9089 - val_loss: 0.1841 - val_accuracy: 0.9440
Epoch 7/15
500/500 - 2s - loss: 0.2858 - accuracy: 0.9086 - val_loss: 0.1711 - val_accuracy: 0.9481
Epoch 8/15
500/500 - 3s - loss: 0.2474 - accuracy: 0.9251 - val_loss: 0.1852 - val_accuracy: 0.9429
Epoch 9/15
500/500 - 3s - loss: 0.2465 - accuracy: 0.9214 - val_loss: 0.1607 - val_accuracy: 0.9510
Epoch 10/15
500/500 - 3s - loss: 0.2330 - accuracy: 0.9270 - val_loss: 0.1585 - val_accuracy: 0.9517
Epoch 11/15
500/500 - 3s - loss: 0.2243 - accuracy: 0.9310 - val_loss: 0.1386 - val_accuracy: 0.9566
Epoch 12/15
500/500 - 3s - loss: 0.2039 - accuracy: 0.9364 - val_loss: 0.1337 - val_accuracy: 0.9581
Epoch 13/15
500/500 - 2s - loss: 0.2045 - accuracy: 0.9358 - val_loss: 0.1399 - val_accuracy: 0.9546
Epoch 14/15
500/500 - 3s - loss: 0.1920 - accuracy: 0.9406 - val_loss: 0.1271 - val_accuracy: 0.9592
Epoch 15/15
500/500 - 3s - loss: 0.1894 - accuracy: 0.9411 - val_loss: 0.1216 - val_accuracy: 0.9622
```

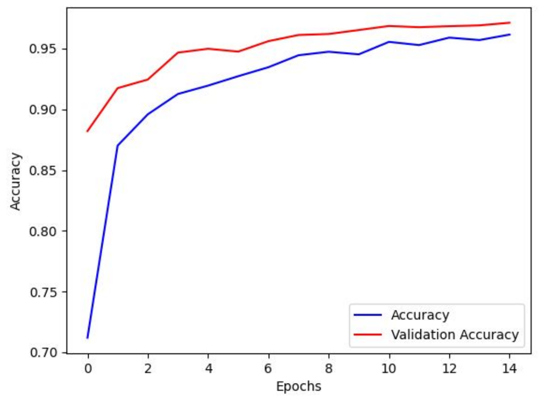
MODEL 12 :

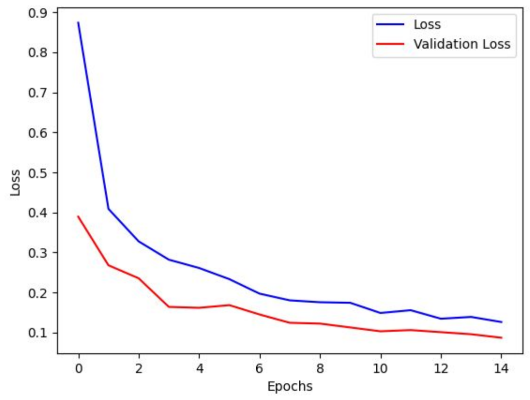
Architecture :

```
________________________________________________________________
Layer (type)              Output Shape           Param #   Activation
================================================================
flatten (Flatten)          (None, 784)          0
________________________________________________________________
dense (Dense)               (None, 200)          157000     relu
________________________________________________________________
dense_1 (Dense)              (None, 300)          60300     relu
________________________________________________________________
dense_2 (Dense)              (None, 300)          90300     relu
________________________________________________________________
dense_3 (Dense)             (None, 10)           3010        softmax
================================================================
Total params: 310,610
Trainable params: 310,610
Non-trainable params: 0
________________________________________________________________
```

Training Accuracy	96.78 %
Training Loss	0.08
Testing Accuracy	97.31 %

Plots :





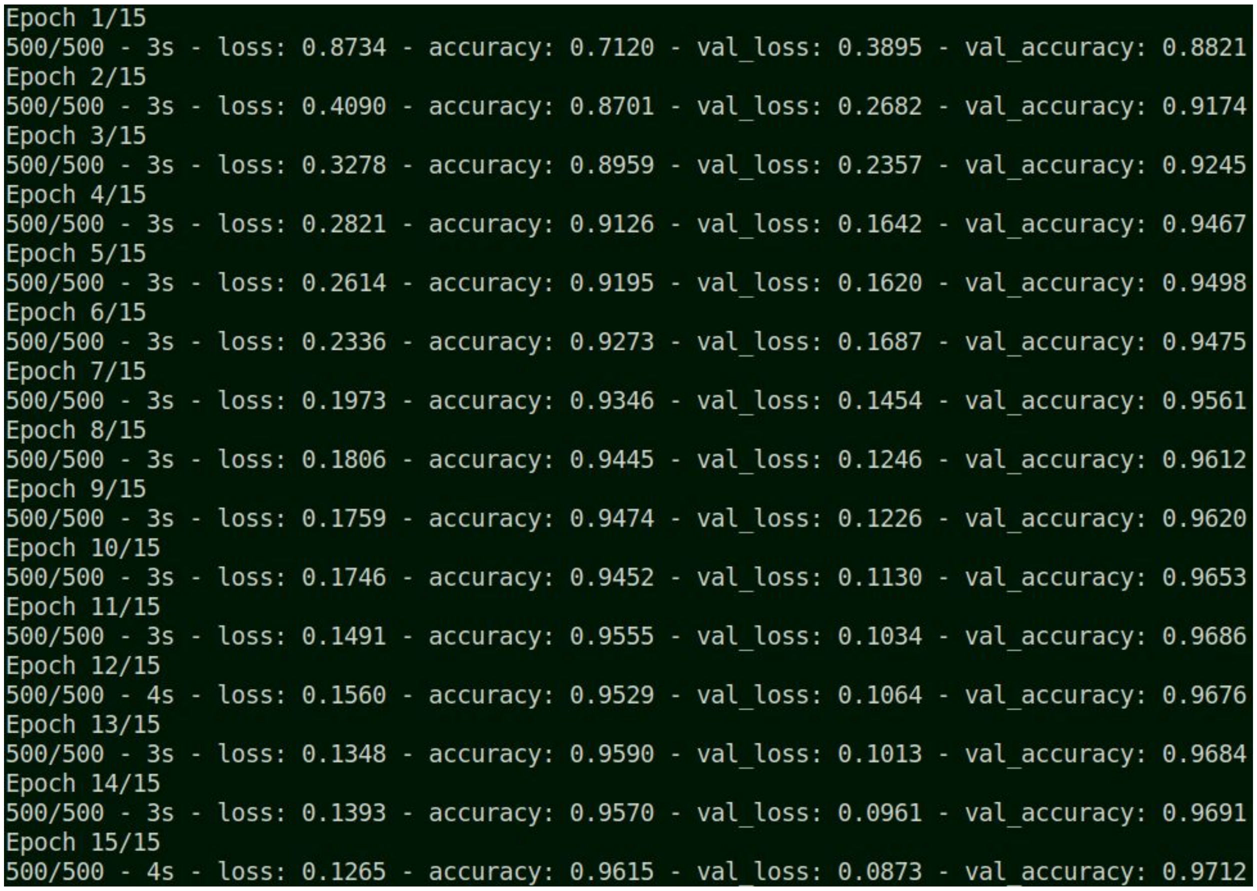
Conclusions:

As expected, increasing the number of nodes again increased the capturing capacity of the model, hence leading to a higher accuracy of 97.31 %. But increasing the width of the model by a larger amount can also lead to **overfitting** as we can see the accuracy gap between training and testing accuracy is significant.

Possible Improvements:

This model with 4 layers and wide structure gave us a commendable accuracy of 97.31 %. But to increase the accuracy further, we can try using the **convolutional layers**. Also generally it is seen that **CNNs** perform really well on image datasets.

Training Screenshot:



```
Epoch 1/15
500/500 - 3s - loss: 0.8734 - accuracy: 0.7120 - val_loss: 0.3895 - val_accuracy: 0.8821
Epoch 2/15
500/500 - 3s - loss: 0.4090 - accuracy: 0.8701 - val_loss: 0.2682 - val_accuracy: 0.9174
Epoch 3/15
500/500 - 3s - loss: 0.3278 - accuracy: 0.8959 - val_loss: 0.2357 - val_accuracy: 0.9245
Epoch 4/15
500/500 - 3s - loss: 0.2821 - accuracy: 0.9126 - val_loss: 0.1642 - val_accuracy: 0.9467
Epoch 5/15
500/500 - 3s - loss: 0.2614 - accuracy: 0.9195 - val_loss: 0.1620 - val_accuracy: 0.9498
Epoch 6/15
500/500 - 3s - loss: 0.2336 - accuracy: 0.9273 - val_loss: 0.1687 - val_accuracy: 0.9475
Epoch 7/15
500/500 - 3s - loss: 0.1973 - accuracy: 0.9346 - val_loss: 0.1454 - val_accuracy: 0.9561
Epoch 8/15
500/500 - 3s - loss: 0.1806 - accuracy: 0.9445 - val_loss: 0.1246 - val_accuracy: 0.9612
Epoch 9/15
500/500 - 3s - loss: 0.1759 - accuracy: 0.9474 - val_loss: 0.1226 - val_accuracy: 0.9620
Epoch 10/15
500/500 - 3s - loss: 0.1746 - accuracy: 0.9452 - val_loss: 0.1130 - val_accuracy: 0.9653
Epoch 11/15
500/500 - 3s - loss: 0.1491 - accuracy: 0.9555 - val_loss: 0.1034 - val_accuracy: 0.9686
Epoch 12/15
500/500 - 4s - loss: 0.1560 - accuracy: 0.9529 - val_loss: 0.1064 - val_accuracy: 0.9676
Epoch 13/15
500/500 - 3s - loss: 0.1348 - accuracy: 0.9590 - val_loss: 0.1013 - val_accuracy: 0.9684
Epoch 14/15
500/500 - 3s - loss: 0.1393 - accuracy: 0.9570 - val_loss: 0.0961 - val_accuracy: 0.9691
Epoch 15/15
500/500 - 4s - loss: 0.1265 - accuracy: 0.9615 - val_loss: 0.0873 - val_accuracy: 0.9712
```

4 TRADITIONAL MODELS

MODEL 13 :

Support Vector Machine (Polynomial)	Accuracy: 0.9629

MODEL 14 :

Support Vector Machine (Gaussian)	Accuracy: 0.9695

MODEL 15 :

Single Decision Tree	Accuracy: 0.8762

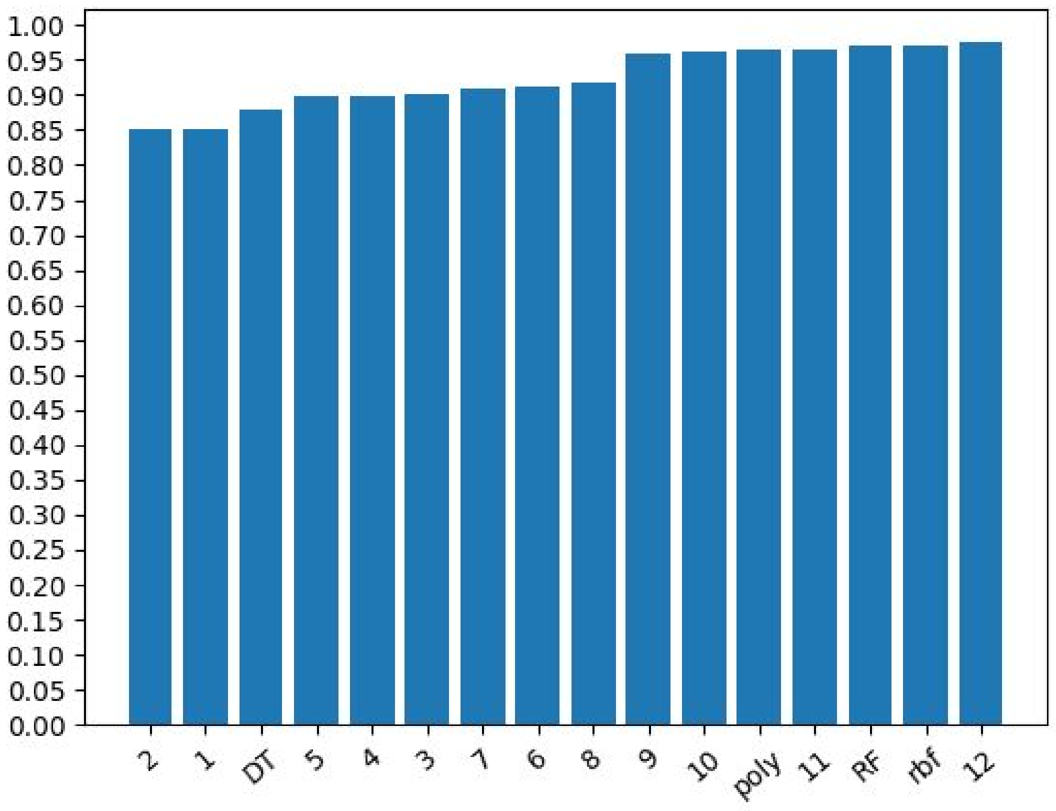
MODEL 16 :

Random Forest	Accuracy:0.9684

Accuracy Comparison of all 16 Models

Label Notations :

1 - 12 : Dense Models
poly : Support Vector Machine (Polynomial)
rbf : Support Vector Machine (Gaussian)
RF : Random Forest
DT : Single Decision Tree



CONCLUSION

We can conclude that these features can help in making a good performing model on MNSIT datasets.

1. More hidden layers till a limit. Adding excessive layers can lead to overfitting too.
2. Increasing the number of nodes in the hidden layers help in increasing the capturing capacity of the model leading to higher accuracy.
3. **relu** was seen to perform better as an activation function than **tanh** for the hidden layers, along with **softmax** function.
4. We need to use trial and error to find the limit of the above given parameters, till the model hits overfitting.

We can see the accuracy gap between training and testing accuracy, this is the situation of overfitting

Here is the animation showing how the accuracy on the training and validation datasets improves upon applying the above given four steps to the models.