A Project Report

On

Comparative Analysis of Different ML and DL Models

(Assignment-1)

BY

KESHAV BERIWAL 2017B4A71301H

RADHESH SARMA 2017B4A70886H

SIMRAN SAHNI 2017B5A70856H

UNDER THE SUPERVISION OF

DR. N.L.BHANU MURTHY

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF

CS F441 : SELECTED TOPICS FROM COMPUTER SCIENCE

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

HYDERABAD CAMPUS

OCTOBER - NOVEMBER, 2020

Contents

1 Objectives **1**

2 Data Used and Preprocessing **1**

3 Methodology **1**

4 Machine Learning Models **2**

4.1 Logistic Regression 2

4.2 Support Vector Machine 3

4.3 Decision Tree 4

4.4 Random Forest 6

4.5 Summary of all Machine Learning Models 7

5 Deep Learning Models **7**

5.1 Model 1 7

5.2 Model 2 9

5.3 Model 3 10

5.4 Model 4 12

5.5 Model 5 13

5.6 Model 6 15

5.7 Model 7 16

5.8 Model 8 18

5.9 Model 9 19

5.10 Model 10 21

5.11 Model 11 22

5.12 Model 12 24

6 Comparison of Different Models **25**

7 Conclusion **28**

List of Figures

1 Classification Report for Logistic Regression 2
2 Confusion Matrix of Logistic Regression 3
3 Classification Report for SVM 4
4 Confusion Matrix of SVM 4
5 Classification Report for Decision Tree 5
6 Confusion Matrix for Decision Tree 5
7 Classification Report for Random Forests 6
8 Confusion Matrix for Random Forest 6
9 Summary of Different Machine Learning Models 7
10 Loss vs Epochs and Accuracy vs Epochs 8
11 Confusion Matrix 8
12 Loss vs Epochs and Accuracy vs Epochs 9
13 Confusion Matrix 10
14 Loss vs Epochs and Accuracy vs Epochs 11
15 Confusion Matrix 11
16 Loss vs Epochs and Accuracy vs Epochs 12
17 Confusion Matrix 13
18 Loss vs Epochs and Accuracy vs Epochs 14
19 Confusion Matrix 14
20 Loss vs Epochs and Accuracy vs Epochs 15
21 Confusion Matrix 16
22 Loss vs Epochs and Accuracy vs Epochs 17
23 Confusion Matrix 17
24 Loss vs Epochs and Accuracy vs Epochs 18
25 Confusion Matrix 19
26 Loss vs Epochs and Accuracy vs Epochs 20
27 Confusion Matrix 20
28 Loss vs Epochs and Accuracy vs Epochs 21
29 Confusion Matrix 22

30 Loss vs Epochs and Accuracy vs Epochs 23
31 Confusion Matrix 23
32 Loss vs Epochs and Accuracy vs Epochs 24
33 Confusion Matrix 25
34 Summary of Different Deep Learning Models 27
35 Accuracy of all models 28

1 Objectives

The main focus of this assignment is to perform a comparative study of different machine learning and deep learning architectures on the MNIST handwritten data set.The four machine learning classifiers are namely Logistics Regression, Support Vector Classification Model, Decision Tree and Random Forest classifiers. Experiments with the design of neural network is done by varying the activation functions, number of hidden layers, number of neurons in a particular layer and optimisers, which may influence the performance of our model.
These models are implemented in this assignment in python using numpy, pandas, matplotlib, sklearn, keras and tensorflow libraries.

2 Data Used and Preprocessing

This report uses MNIST handwritten digit database. MNIST handwritten digit database can be taken from (Lecun, n.d.). It has become a standard for fast-testing theories of pattern recognition and machine learning algorithms. It contains 60,000 handwritten digit images for the classifier training and 10,000 handwritten digit images for the classifier testing, both drawn from the same distribution. All these black and white digits are size normalized, and centered in a fixed-size image where the center of the intensity lies at the center of the image with 28 × 28 pixels. The dimensionality of each image sample vector is 28 * 28 = 784, where each element is binary.if the pixel value is '0' it indicates that the background is black and if it is '1' the background is white. We normalize the data set before using it for our models. To normalize the data set, we divide the each data point by 255(i.e. maximum RGB code - minimum RGB code).

3 Methodology

After preprocessing, the data was shuffled randomly.There are 70000 examples , out of which 10000 are for testing and 60000 are for training.12 Deep learning models are made by varying the number of hidden layers, number of hidden neurons and activation functions.The models were then trained with 50 iterations.Model accuracy, Model loss for each function is plotted

and accuracy of each model is measured. We have printed the confusion matrix of each model and measured the parameters like F1 score, cohen kappa score to analyse the performance.

4 Machine Learning Models

4.1 Logistic Regression

Model description:

- Preprocessing: Training data split in ratio 9:1
- This is a linear classifier, works best on binary classifcation.
- Optimizer used: lbfgs (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) for faster convergence.
- Accuracy: 92.65%
- Kappa: 0.9183

```
Cohen Kappa score is 0.9183056480251641
Accuracy is 0.9265
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       624
           1       0.95      0.97      0.96       654
           2       0.90      0.90      0.90       572
           3       0.90      0.90      0.90       589
           4       0.94      0.94      0.94       580
           5       0.89      0.89      0.89       551
           6       0.96      0.95      0.96       580
           7       0.95      0.94      0.94       633
           8       0.89      0.87      0.88       585
           9       0.92      0.92      0.92       632

    accuracy                           0.93      6000
   macro avg       0.93      0.93      0.93      6000
weighted avg       0.93      0.93      0.93      6000
```

Figure 1: Classification Report for Logistic Regression

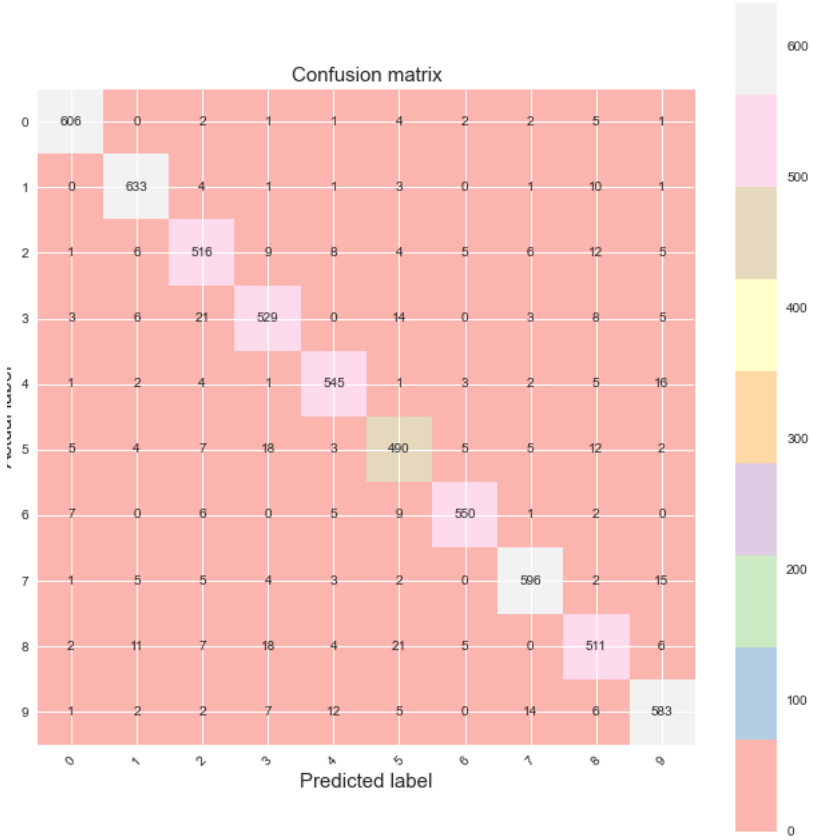

Figure 2: Confusion Matrix of Logistic Regression

4.2 Support Vector Machine

Model description:

- Preprocessing: Training data split in ratio 75:25
- Support Vector Classifier used : 'Linear' kernel
- Accuracy: 92.25%
- Kappa: 0.9138

	precision	recall	f1-score	support
0	0.95	0.97	0.96	1510
1	0.96	0.98	0.97	1728
2	0.89	0.91	0.90	1458
3	0.90	0.90	0.90	1548
4	0.91	0.93	0.92	1434
5	0.88	0.87	0.88	1342
6	0.96	0.95	0.95	1468
7	0.94	0.93	0.93	1535
8	0.91	0.87	0.89	1484
9	0.91	0.90	0.91	1493
accuracy			0.92	15000
macro avg	0.92	0.92	0.92	15000
weighted avg	0.92	0.92	0.92	15000

Figure 3: Classification Report for SVM

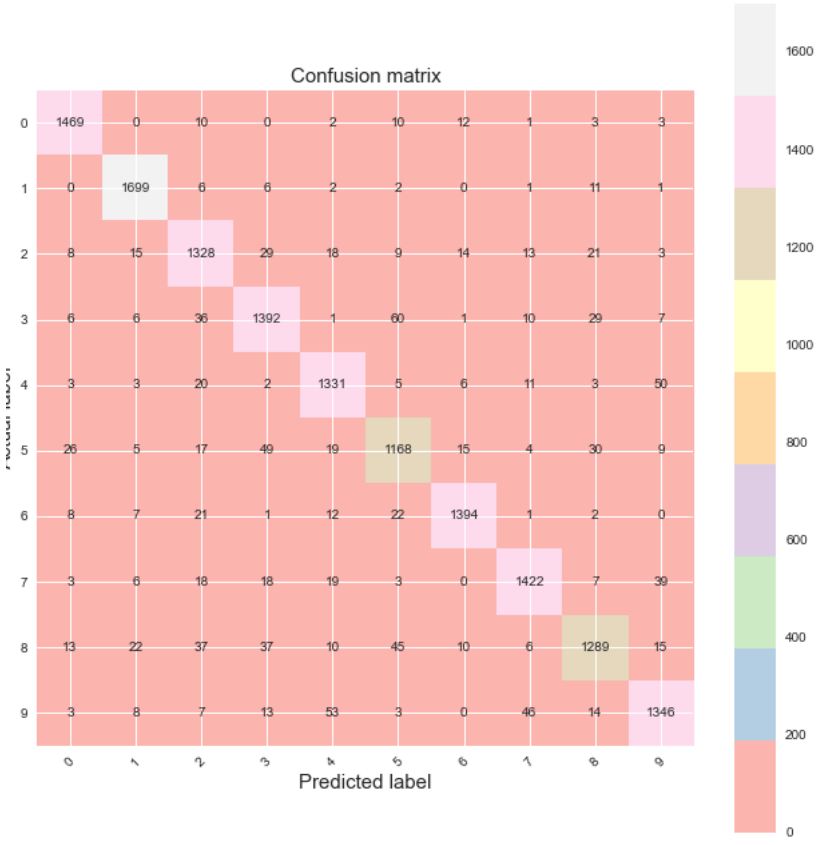

Figure 4: Confusion Matrix of SVM

4.3 Decision Tree

Model description:

- Preprocessing: Training data split in ratio 9:1

- This classifier uses an if-else construct to arrive at a conclusion, hence is slower or not that accurate for this huge data set.
- Accuracy:87.1%
- Kappa: 0.856

Cohen Kappa score is 0.8566264601239718
Accuracy is 0.871

	precision	recall	f1-score	support
0	0.93	0.91	0.92	624
1	0.93	0.96	0.94	654
2	0.86	0.87	0.86	572
3	0.83	0.83	0.83	589
4	0.87	0.87	0.87	580
5	0.79	0.83	0.81	551
6	0.89	0.91	0.90	580
7	0.91	0.89	0.90	633
8	0.84	0.79	0.82	585
9	0.84	0.85	0.85	632
accuracy			0.87	6000
macro avg	0.87	0.87	0.87	6000
weighted avg	0.87	0.87	0.87	6000

Figure 5: Classification Report for Decision Tree

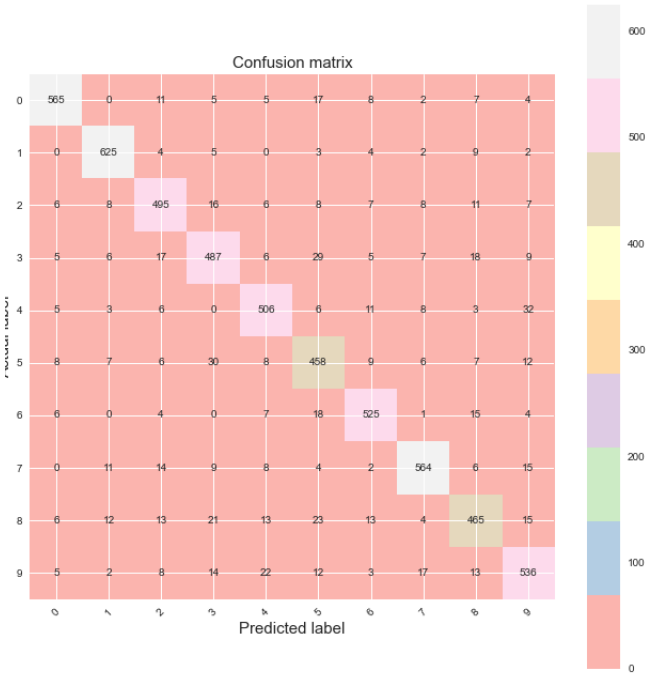

Figure 6: Confusion Matrix for Decision Tree

4.4 Random Forest

Model description:

- Preprocessing: Training data split in ratio 9:1
- This classifier is an advanced accurate model. It is perceived as a collection of Decision trees
- Accuracy: 97.08%
- Kappa: 0.965

	precision	recall	f1-score	support
0	0.99	0.98	0.98	587
1	0.98	0.99	0.98	630
2	0.97	0.97	0.97	600
3	0.97	0.97	0.97	627
4	0.97	0.96	0.97	595
5	0.97	0.95	0.96	549
6	0.97	0.99	0.98	571
7	0.98	0.98	0.98	668
8	0.96	0.94	0.95	597
9	0.94	0.97	0.96	576
accuracy			0.97	6000
macro avg	0.97	0.97	0.97	6000
weighted avg	0.97	0.97	0.97	6000

Figure 7: Classification Report for Random Forests

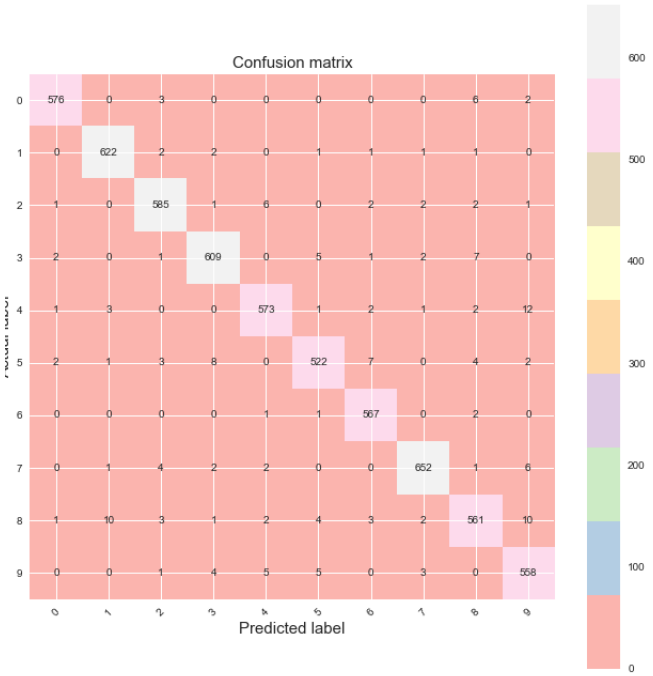

Figure 8: Confusion Matrix for Random Forest

4.5 Summary of all Machine Learning Models

Algorithm	Accuracy	Kappa	F1 Score
Logistics Regression	92.65%	0.9183	0.93
Decision Tree Classifier	87.1%	0.8566	0.87
Random Forest Classifier	97.08%	0.9675	0.97
Support Vector Machine Classifier	92.25%	0.9138	0.92

Figure 9: Summary of Different Machine Learning Models

It can be clearly observed that the Random Forest Classifier is significantly better when compared to Logistics Regression, Decision Tree, and Support Vector Machine classifier. A lot of other hyperparamters of these solvers can be varied to get the desired accuracy for the respective dataset,using GridSearchCV searcher modules of sklearn. But they fall short after a certain point, and start resulting in good deal of misclassifications. Also training these simple ML models on such huge 70k entries dataset of MNIST is very time consuming.

5 Deep Learning Models

5.1 Model 1

- Layer Wise Description: (784, input) $->$ (512, relu) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 96.1%
- Kappa: 0.9709

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 10.

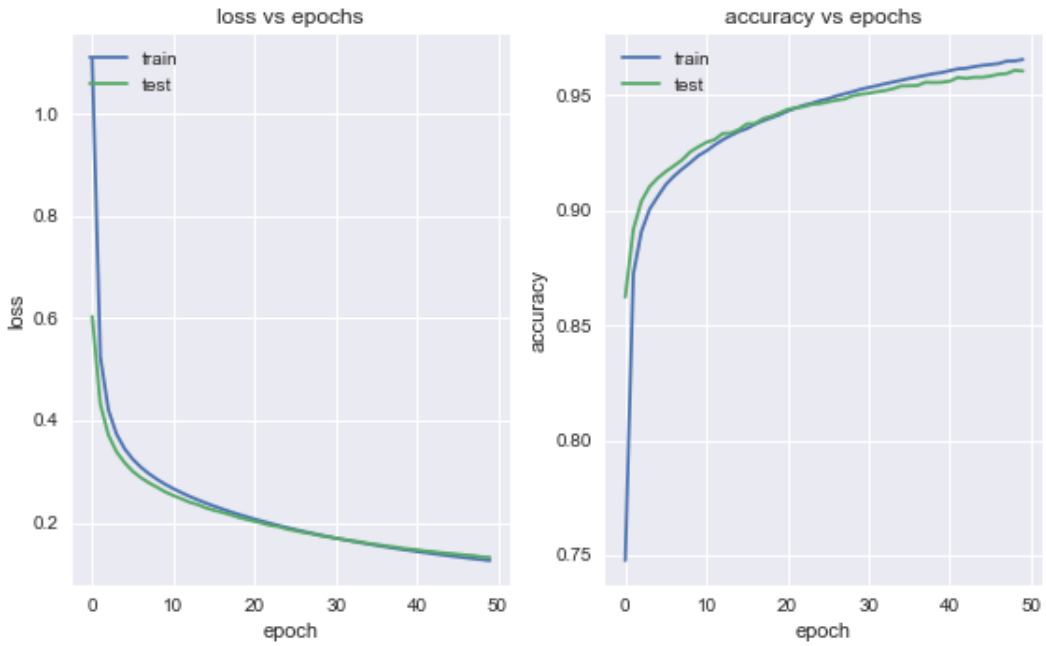

Figure 10: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 11.

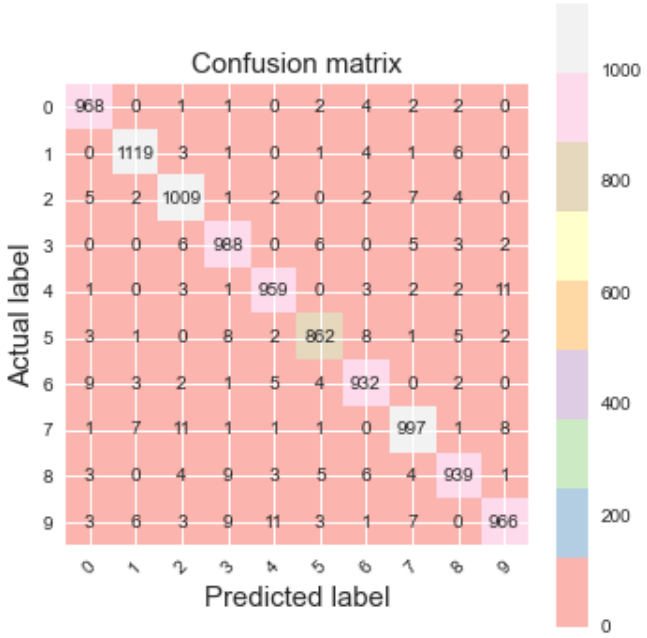

Figure 11: Confusion Matrix

5.2 Model 2

- Layer Wise Description: (784, input) $->$ (512, sigmoid) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 91.5%
- Kappa: 0.9058

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 12.

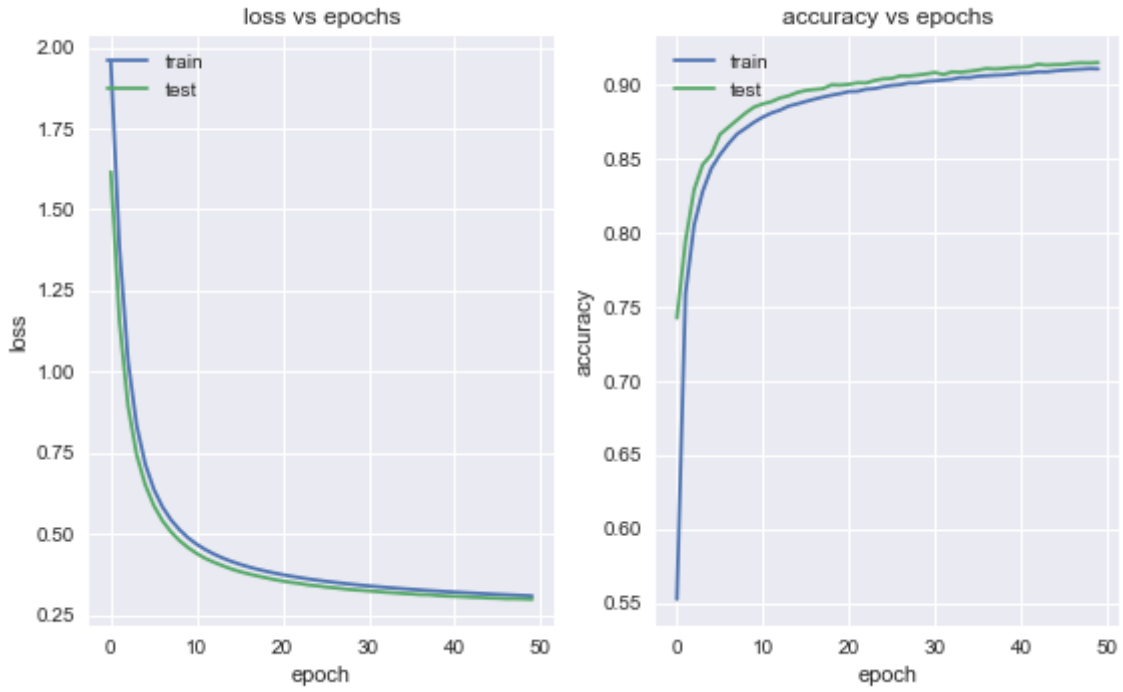

Figure 12: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 13.

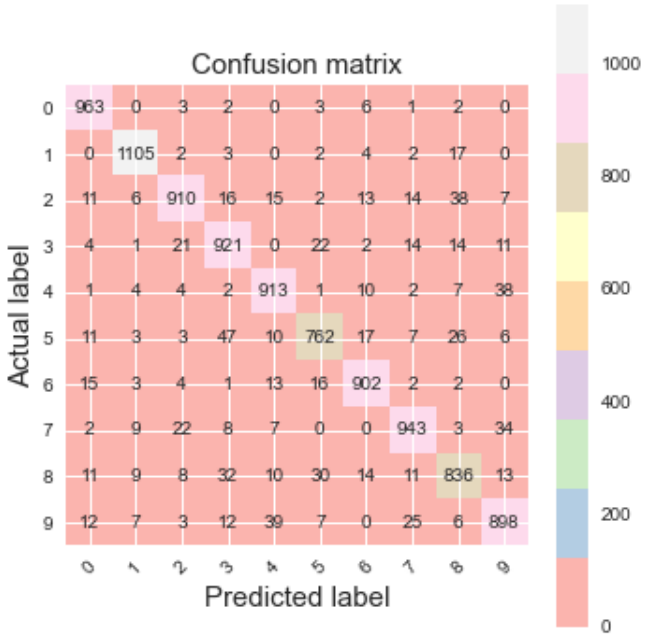

Figure 13: Confusion Matrix

5.3 Model 3

- Layer Wise Description: (784, input) $->$ (512, softplus) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 92.8%
- Kappa: 0.9199

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 14.

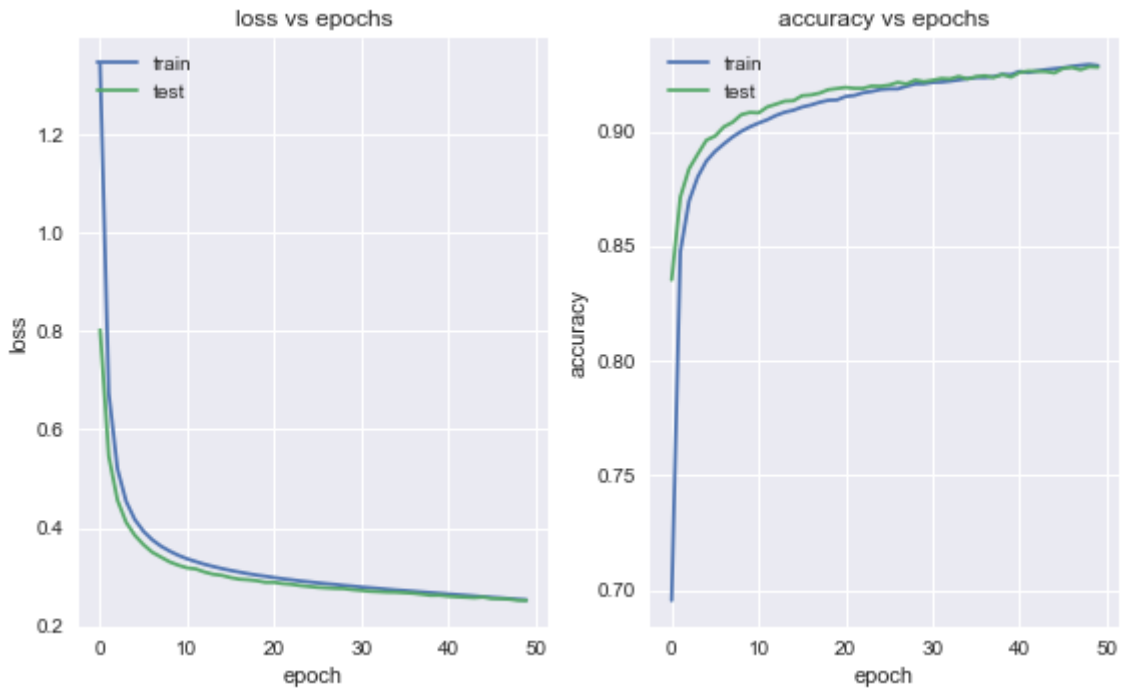

Figure 14: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 15.

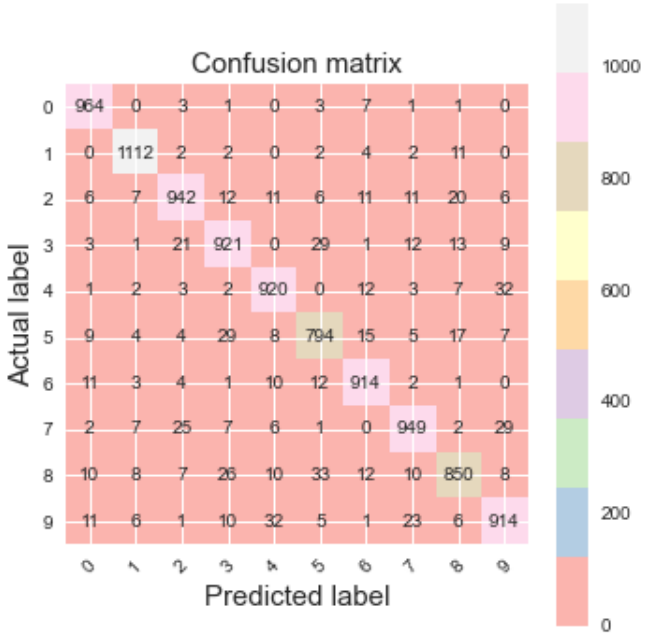

Figure 15: Confusion Matrix

5.4 Model 4

- Layer Wise Description: (784, input) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 91.8%
- Kappa: 0.9090

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 16.

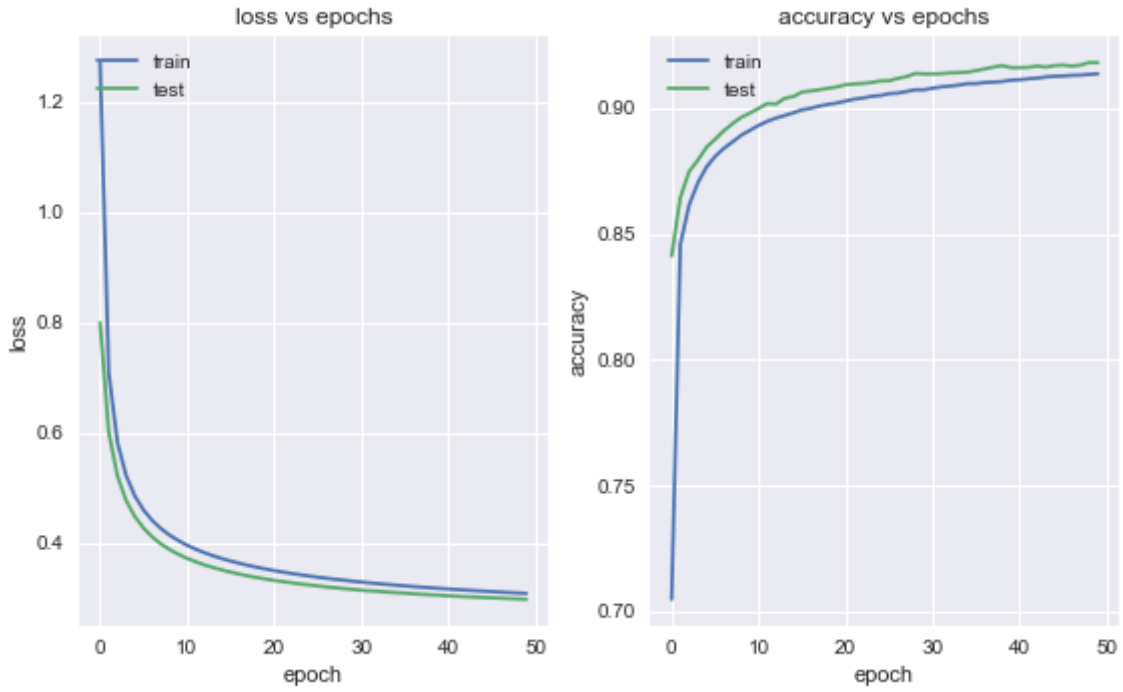

Figure 16: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 17.

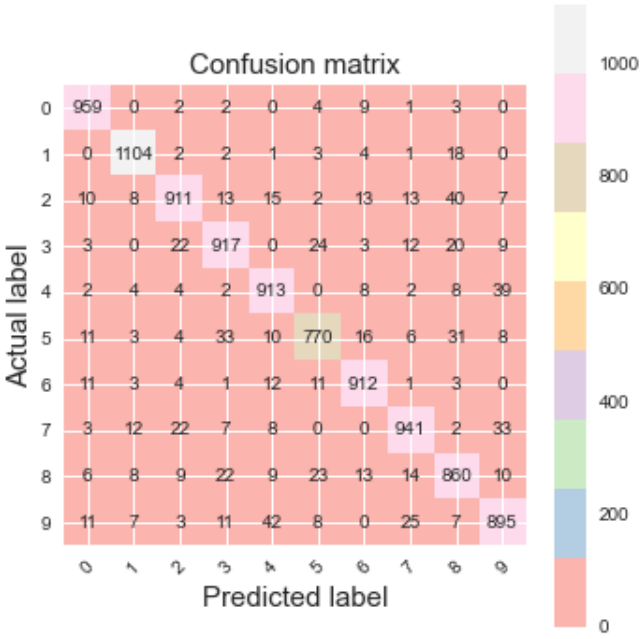

Figure 17: Confusion Matrix

5.5 Model 5

- Layer Wise Description: (784, input) $->$ (256,Relu) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 96.1%
- Kappa: 0.9567

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 18.

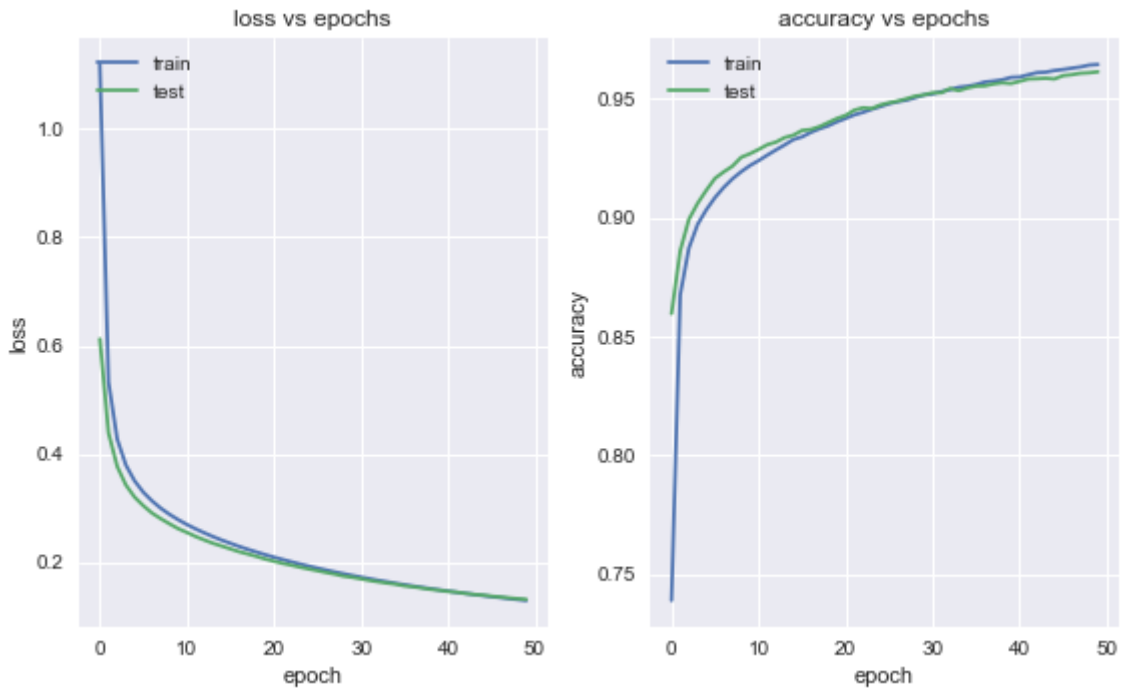

Figure 18: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 19.

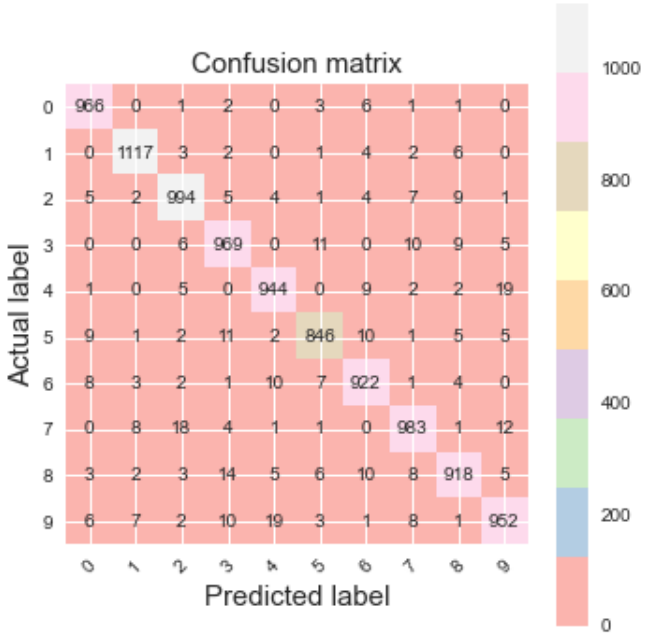

Figure 19: Confusion Matrix

5.6 Model 6

- Layer Wise Description: (784, input) $->$ (256,Relu) $->$ (256,Relu) $->$(10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 97.2%
- Kappa: 0.9686

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 20.

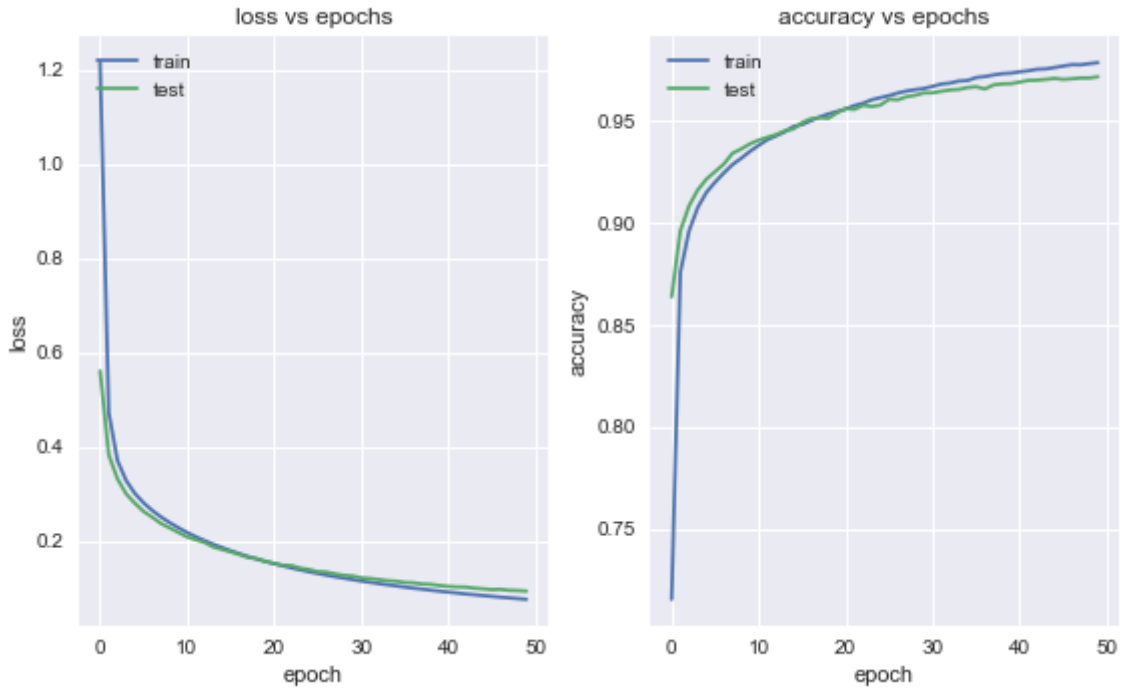

Figure 20: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 21.

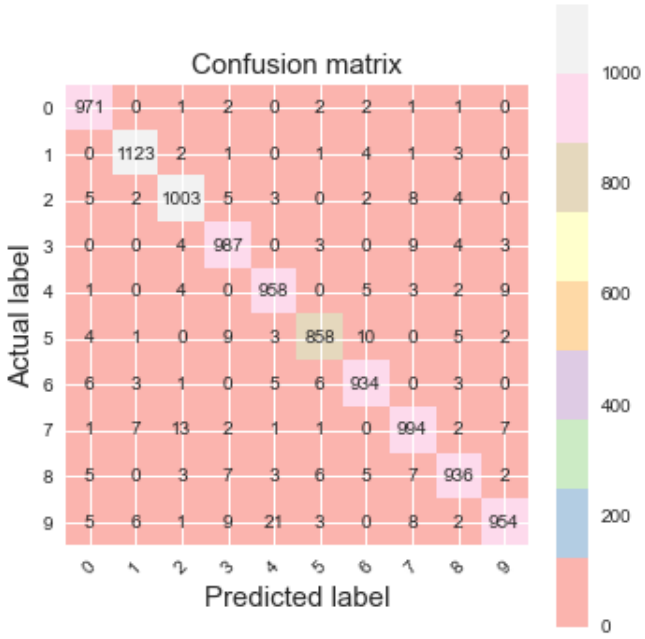

Figure 21: Confusion Matrix

5.7 Model 7

- Layer Wise Description: (784, input) $->$ (256,Relu) $->$ (256,Relu) $->$ (256,Relu) $->$(10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 97.5%
- Kappa: 0.9726

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 22.

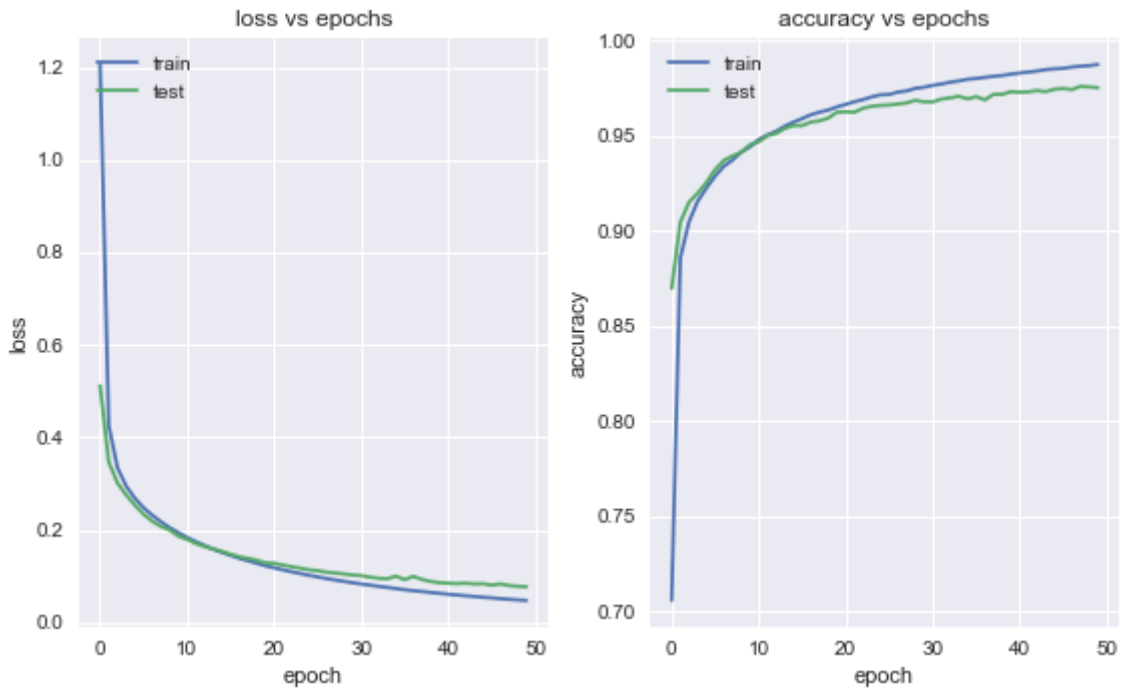

Figure 22: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 23.

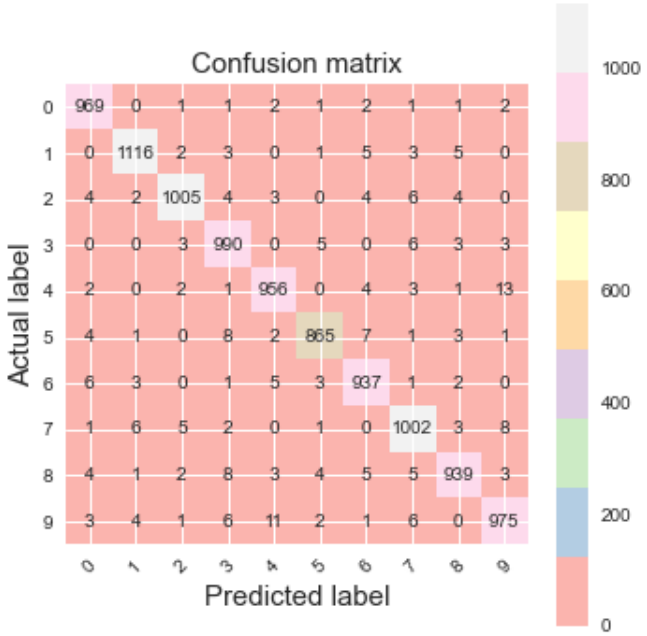

Figure 23: Confusion Matrix

5.8 Model 8

- Layer Wise Description: (784, input) $->$ (1024,Relu) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 96.4%
- Kappa: 0.9594

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 24.

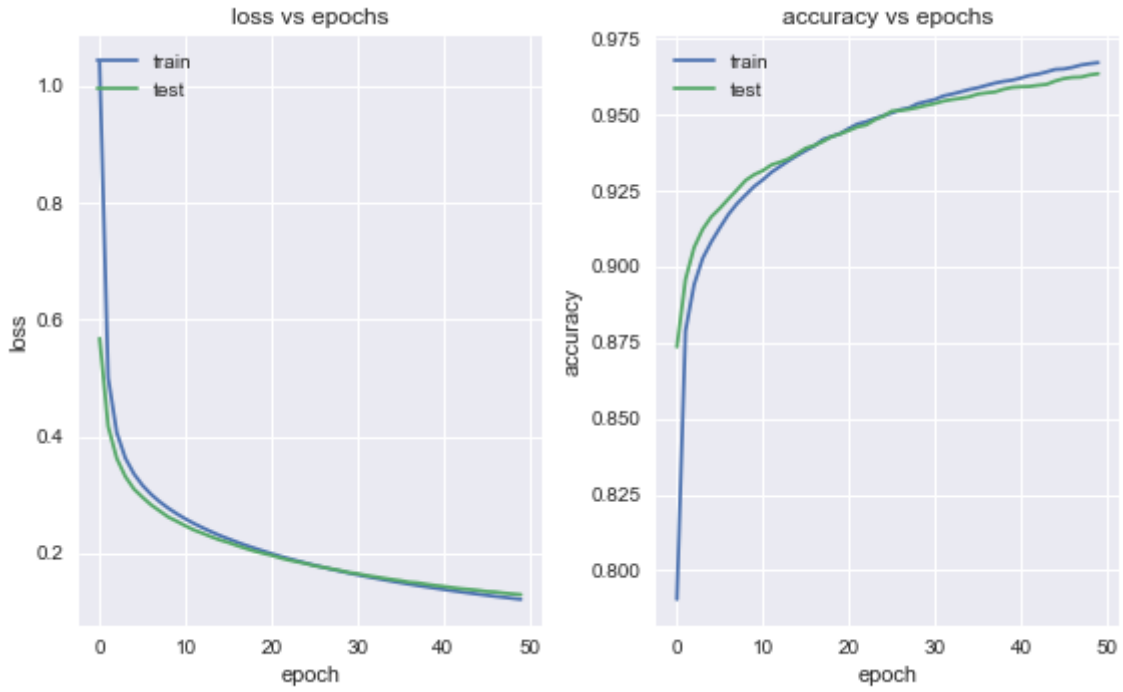

Figure 24: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 25.

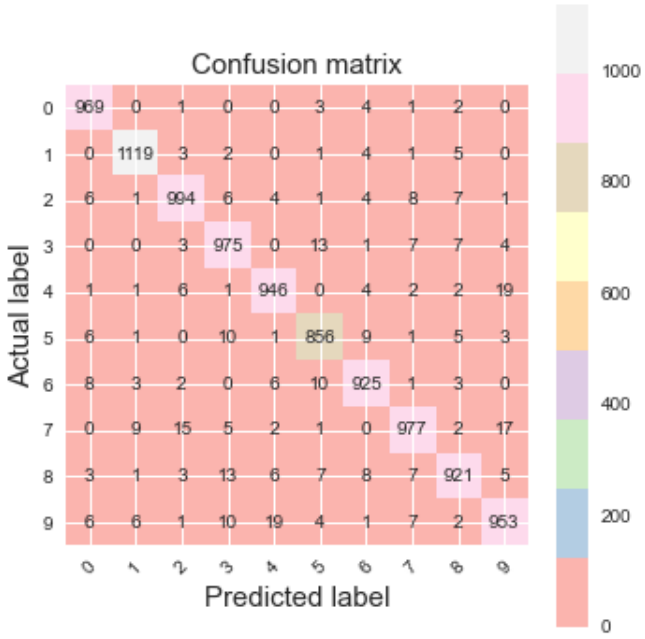

Figure 25: Confusion Matrix

5.9 Model 9

- Layer Wise Description: (784, input) $->$ (512,Relu) $->$ (512,Softplus) $->$(10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 96.9%
- Kappa: 0.9655

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 26.

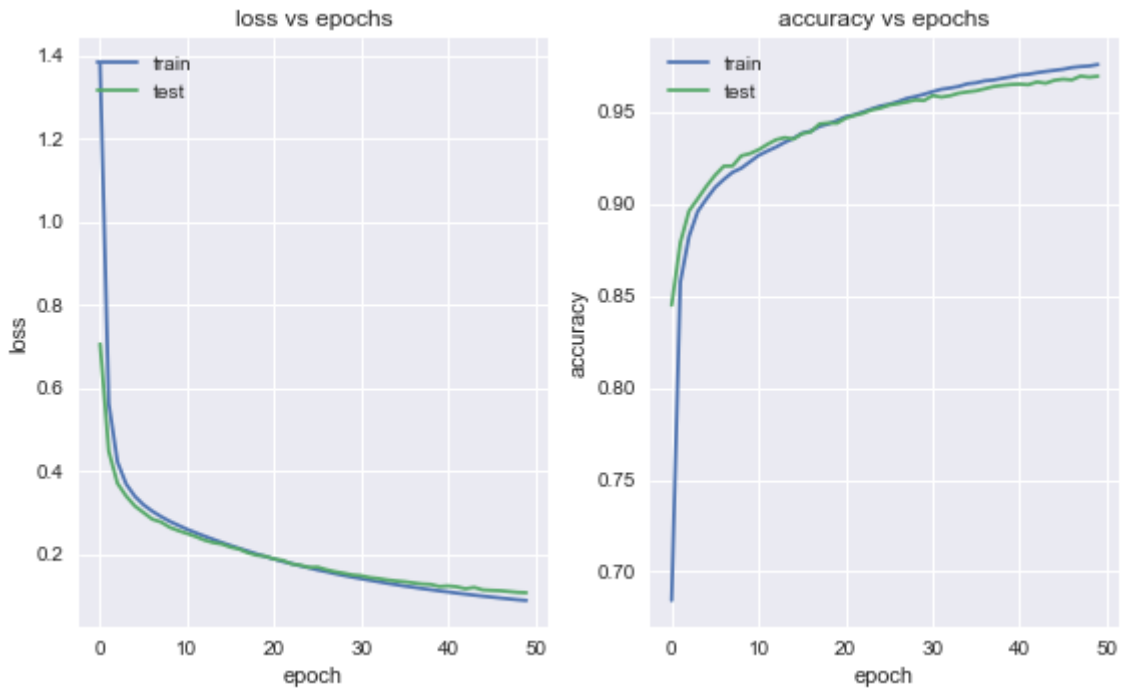

Figure 26: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 27.

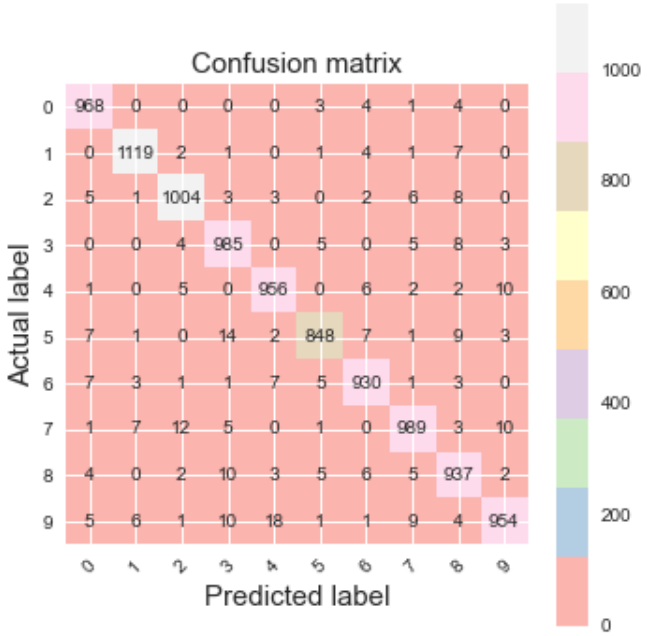

Figure 27: Confusion Matrix

5.10 Model 10

- Layer Wise Description: (784, input) $->$ (512,Softplus) $->$ (512,Relu) $->$(10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Stochastic Gradient Descent
- Accuracy: 95.5%
- Kappa: 0.9495

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 28.

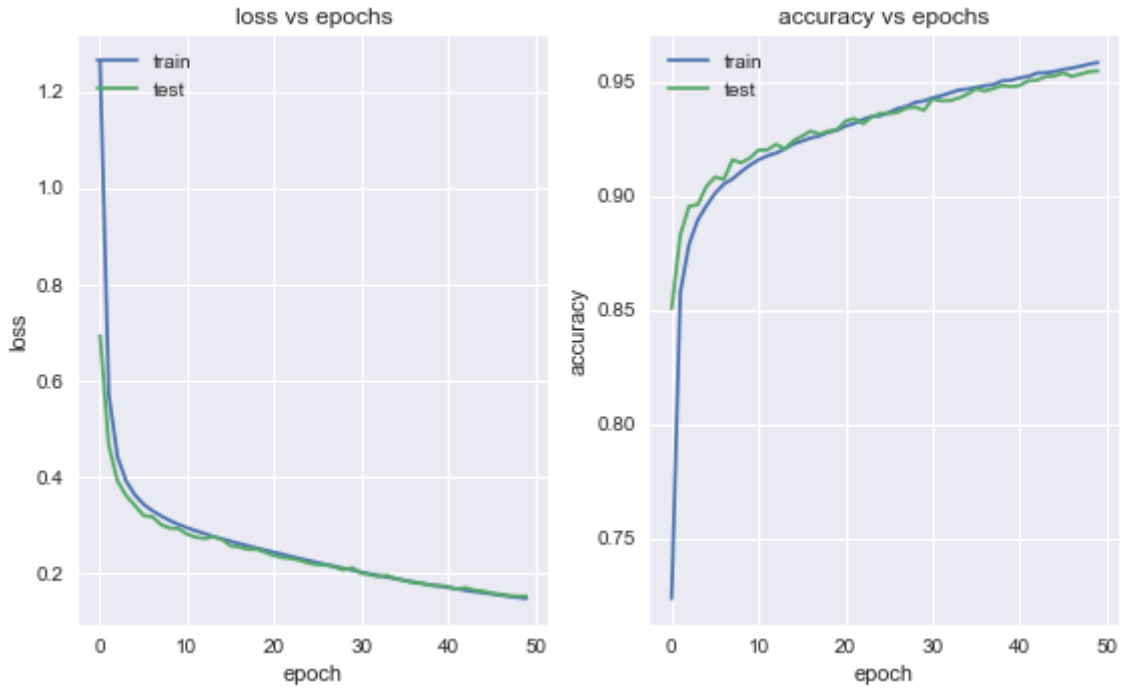

Figure 28: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 29.

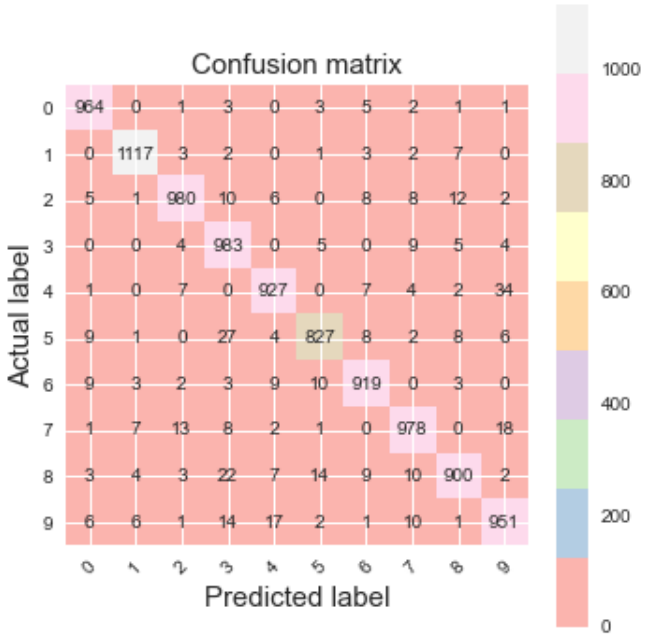

Figure 29: Confusion Matrix

5.11 Model 11

- Layer Wise Description: $(784, \text{input}) -> (512, \text{relu}) -> (10, \text{softmax})$
- Loss Function: Sparse categorical Crossentropy
- Optimizer: Adam
- Accuracy: 97.4%
- Kappa: 0.9709

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 30.

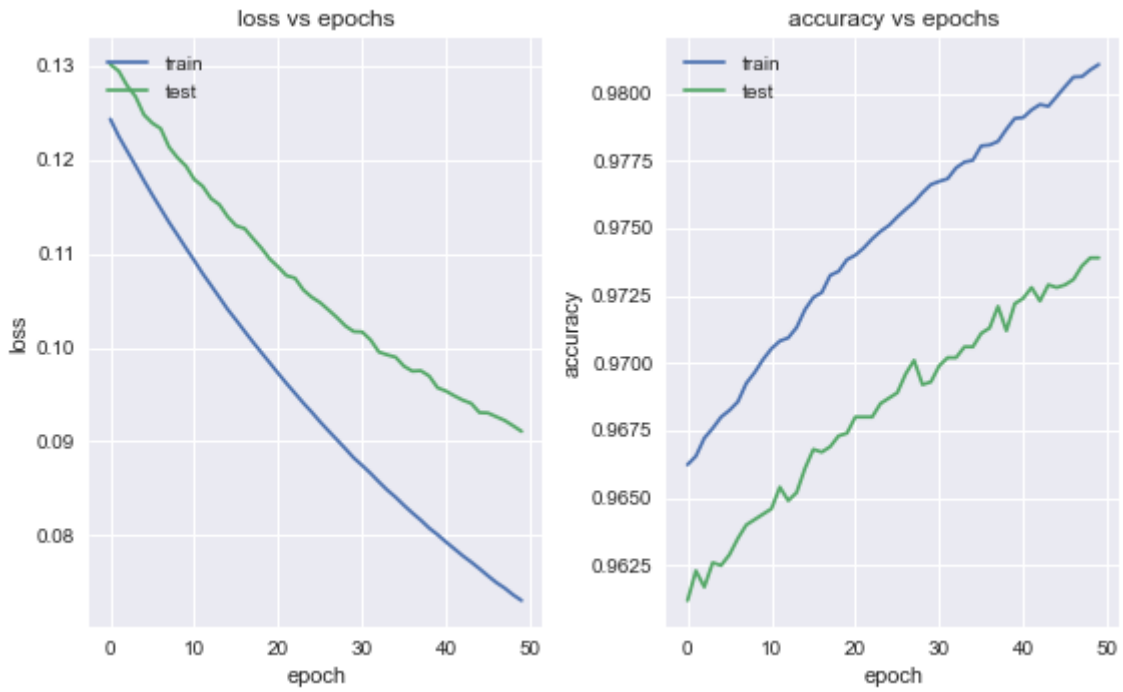

Figure 30: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 31.

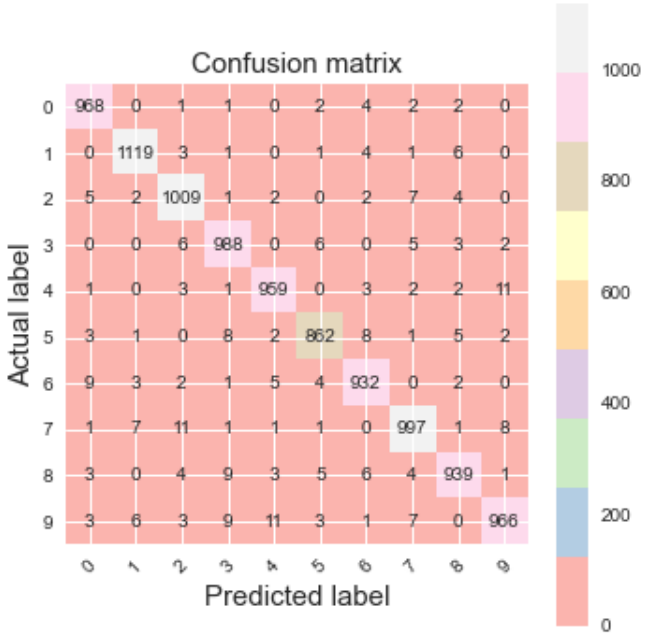

Figure 31: Confusion Matrix

5.12 Model 12

- Layer Wise Description: (784, input) $->$ (512, relu) $->$ (10, softmax)
- Loss Function: Sparse categorical Crossentropy
- Optimizer: RMSprop
- Accuracy: 98.4%
- Kappa: 0.9821

Loss vs Epcochs and accuracy vs Epochs plots for both training and testing data are shown in Fig 32.

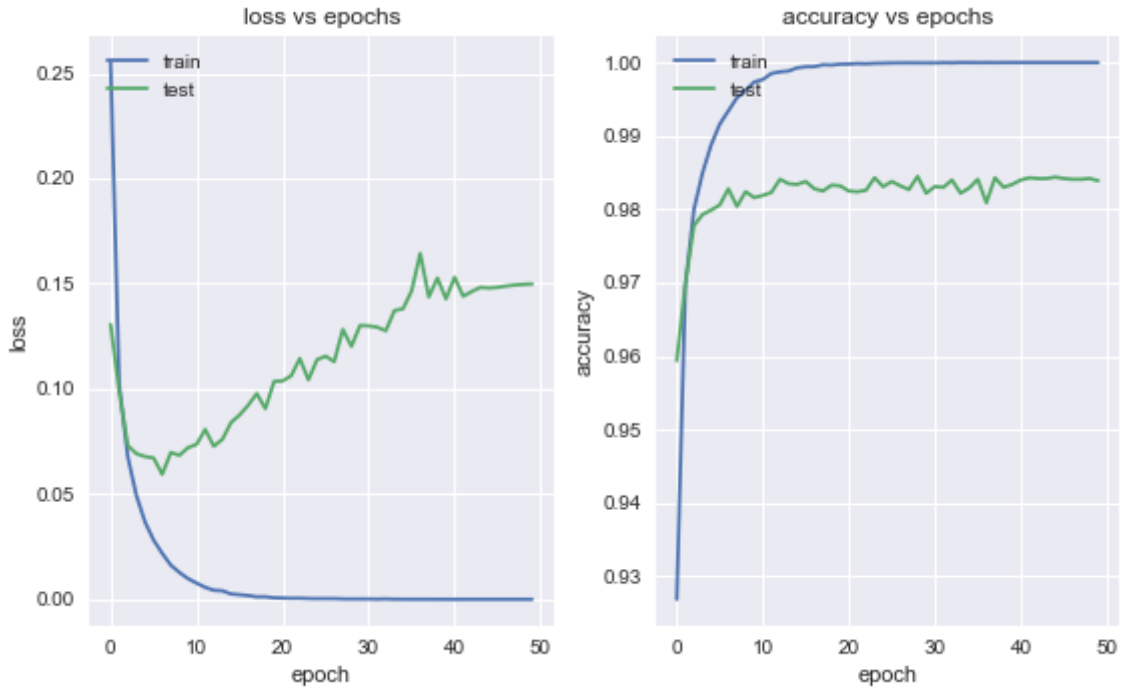

Figure 32: Loss vs Epochs and Accuracy vs Epochs

The performance of the classification model has been summarised in the form of confusion matrix in Fig 33.

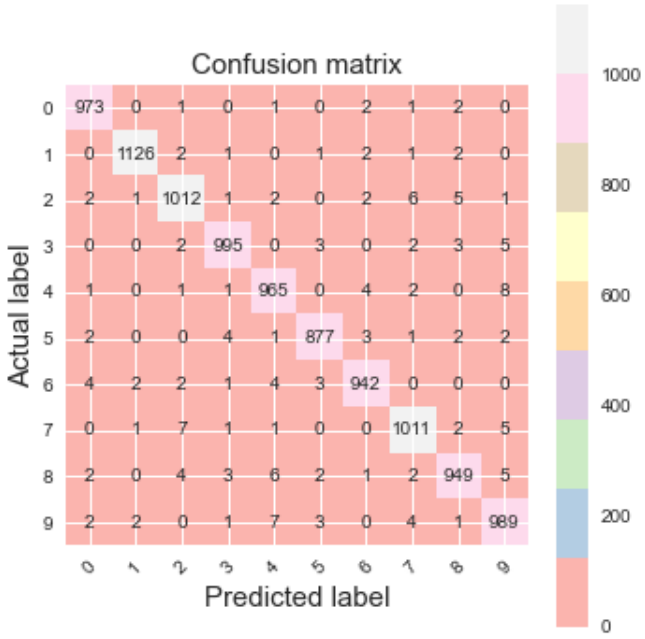

Figure 33: Confusion Matrix

6 Comparison of Different Models

From the above results we observe the following things-

- The **first 3 DL models have different activation functions** for hidden layer and rest parameters are same. We observe that **ReLu Activation function (96.1%) performs significantly better** when compared to softplus (92.8%) and sigmoid (91.5%).
  - This is because sigmoid functions saturates (i.e. the curve becomes parallel to x-axis) for a large positive or large negative number. Thus, the gradient at these regions is almost zero. During backpropagation, if the local gradient is very small, it'll kill the the gradient and the network will not learn. Also, sigmoid outputs are not zero-centered, which is undesirable because it can indirectly introduce undesirable zig-zagg dynamics in the gradient updates for the weights.
  - Outputs produced by sigmoid functions have upper and lower limits whereas softplus function produces outputs in scale of $(0, +inf)$, and its gradient doesn't vanish while back propagating. Hence, it performs slightly better than sigmoid functions.

  - Relu function performs best beacuse it overcomes all the issues mentioned above. The problem of vanishing gradient is solved by ReLU. Both the ReLU and Softplus are largely similar, except near 0 where the softplus is enticingly smooth and differentiable. It's much easier and efficient to compute ReLU and its derivative than for the softplus function which has log(.) and exp(.) in its formulation. Hence, By switching to ReLU, the forward and backward passes are much faster while retaining the non-linear nature of the activation function required for deep neural networks to be useful.

- On comparing the **models 1, 5 and 8** we observe that as we **increase the number of hidden units per layer**, with relu activation function and 2 hidden layers, **the accuracy slightly increases.**
  - This is because model gets more adaptive, so it can learn smaller details.
  - However, this may not always be true. The model may get more affected by overfitting hence, generalization of classification model might also decrease.
- On comparing the **models 4, 5, 6 and 7** we observe that as we **increase the number of hidden layers** with the relu activation function, the model accuracy significantly increases.
  - This is because complexity of network will increase, thus resulting in increased efficiency.
  - However, increasing the number of hidden layers may lead to overfitting because it will make it easier for the neural network to memorize the training set, that is to learn a function that perfectly separates the training set but that does not generalize to unseen data.

Model Number	Accuracy	Kappa	F1 Score
Model 1	96.1%	0.9709	0.97
Model 2	91.5%	0.9058	0.92
Model 3	92.8%	0.9199	0.93
Model 4	91.8%	0.9090	0.92
Model 5	96.1%	0.9567	0.96
Model 6	97.2%	0.9686	0.97
Model 7	97.5%	0.9726	0.98
Model 8	96.4%	0.9594	0.96
Model 9	96.9%	0.9655	0.97
Model 10	95.5%	0.9495	0.95
Model 11	97.4%	0.9709	0.97
Model 12	98.4%	0.9821	0.99

Figure 34: Summary of Different Deep Learning Models

7 Conclusion

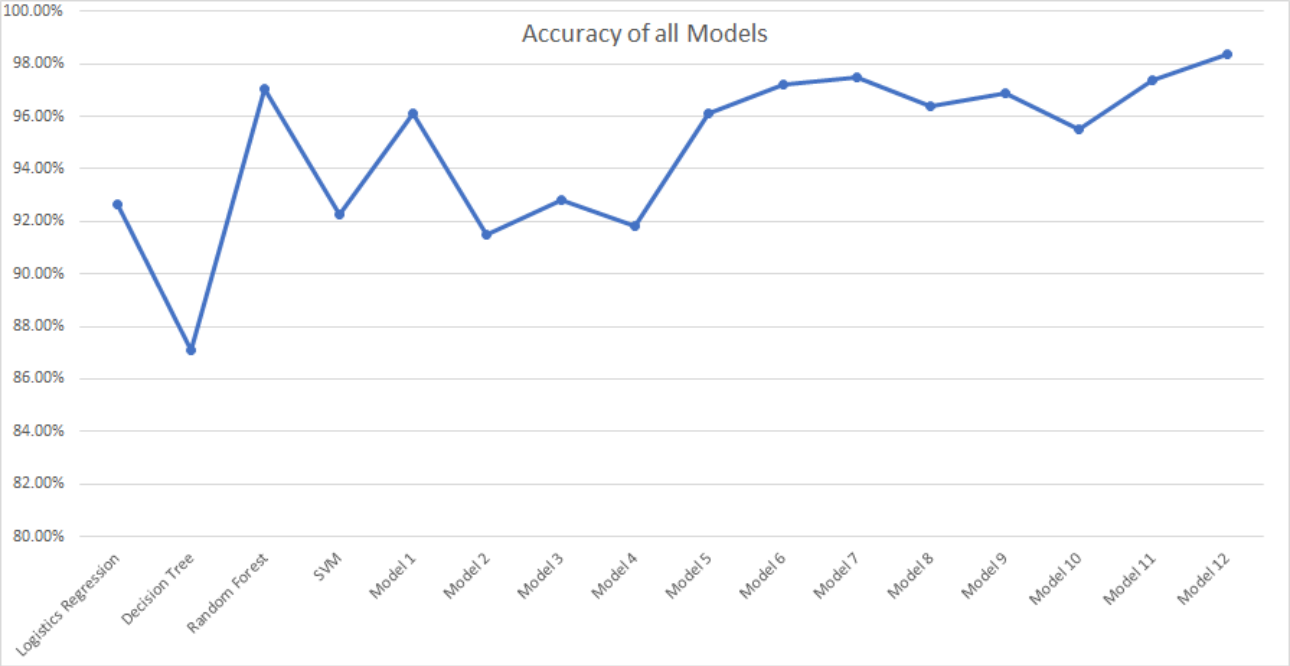

Figure 35: Accuracy of all models

Sensitivity, applicability and comparative performance of 12 different Deep Learning models and 4 Machine Learning Algorithms, for the classification of MNIST data set, were evaluated. DL models in general perform better or at par with the ML models, if chosen the correct parameters depending on the data set.

Among Machine Learning Models, Random Forest Classifier performed better (97.08%) when compared to Logistics Regression, Decision Tree, and Support Vector Machine classifier. Decision Tree gave us the lowest classification accuracy of 87.1%.

Among Deep Learning Models, results demonstrate that the Deep Learning model, with 2 hidden layers, Relu function and RMSProp optimiser(98.4%), outperformed other classification models whilst the Deep Learning model, with 2 hidden layers, sigmoid function and SGD optimiser, obtained the lowest classification accuracy of 91.5%. Moreover, Deep Learning model, with 3 hidden layers, Relu function and SGD optimiser, gave the second highest overall classification accuracy of 97.4%.

From the above results, we can also conclude that ReLu activation function performs the best while sigmoid gives the worst performance. Also, on increasing the number of hidden layers and number of units per layer, we get slightly better results.