CSE574 Introducion to Machine Learning

By Prof: Sargur Srihari

Project 3 Report.

Submitted by

Sai Varun Alapati
Ubit : saivarun
Personal Number :50290571

Table of Contents

1. Introduction : 4
2. Problem Definition : 4
3. Dataset Description and Pre Processing : 4
   3.1 Source of Dataset : 4
      3.1.1 MNIST Data 4
      3.1.2 USPS Data 5
   3.2 Data Preprocessing steps : 5
      3.2.1 Extracting Feature Values and labels form the data : 5
      3.2.2 Target Values : 5
      3.2.3 Feature vectors : 5
   3.3 Data Division: 5
4. Classifiers 5
   4.1 Training Logistic Regression Model : 5
      4.1.1 The learning system : 5
      4.1.2 Logistic Regression using gradient descent (SGD) : 6
      4.1.3 Experimental Values : 6
      4.1.4 Observations : 7
   4.2 Neural Networks Model : 8
      4.2.1 The learning system : 8
      4.2.2 Experimental Values : 9
      4.2.3 Observations 16
   4.3 Support Vector Machines 17
      4.3.1 Learning System 17
      4.3.2 Tuning parameters 17
      4.3.2.2 Regularization : 17
      4.3.2.3 Gamma 17
      4.3.2.4 Margin 17
      4.3.3 Experimental Values : 17
      4.3.4 Observations : 18
   4.4 Random Forest Classifier 19
      4.4.1 Learning System 19
      4.4.2 Tuning parameters 19

4.4.3 Experimental Values : 19
4.4.4 Observations : 20
4.5 Mini Batch Stochastic gradient descent with Logistic Regression Model : 21
4.5.1 The learning system : 21
4.5.2 Logistic Regression using mini batch stochastic gradient descent (SGD) : 21
4.5.3 Experimental Values : 22
4.5.6 Observations : 23
4.6 Ensmeble Classifier 24
4.6.1 Majority/Hard Voting : 24
4.6.2 Observations : 25
5 Conclusions and Summary : 26

1. Introduction :

The aim of the project is to implement machine learning methods for the task of classification. The classification task will be that of recognizing a 28×28 grayscale handwritten digit image and identifying it as a digit among 0, 1, 2, ... , 9.

2. Problem Definition :

In this project, recognizing a 28×28 grayscale handwritten digit image and identifying it as a digit among 0, 1, 2, ... , 9. We train the following five classifiers using MNIST digit images.

1. Logistic regression, which implemented using Stochastic Gradient Descent and tuned hyper parameters.
2. A publicly available multilayer perceptron neural network, trained it on the MNIST digit images and tuned hyper parameters.
3. A publicly available Random Forest package, trained it on the MNIST digit images and tuned hyper parameters.
4. A publicly available SVM package, trained it on the MNIST digit images and tuned hyper parameters.
5. Logistic regression, which implemented using Mini Batch Stochastic Gradient Descent and tuned hyper parameters.

3. Dataset Description and Pre Processing :

3.1 Source of Dataset :

We used two data sets, MNIST Data and USPS data.

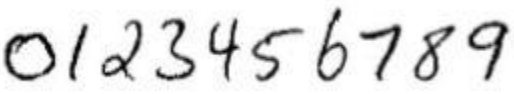
3.1.1 MNIST Data

For both training and testing of our classifiers, we used the MNIST dataset. The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.

The database contains 60,000 training images and 10,000 testing images. The dataset could be downloaded from here: http://yann.lecun.com/exdb/mnist/ The original black and white (bilevel) images from MNIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

3.1.2 USPS Data

We use USPS handwritten digit as another testing data for this project to test whether your models could be generalize to a new population of data. Examples of each of the digits are given below.

Figure 1: Examples of each of the digits 

Each digit has 2000 samples available for testing. These are segmented images scanned at a resolution of 100ppi and cropped. Resize or fill the images to 28x28 like MNIST digits and feed this into your trained model and compare the result on USPS data and MNIST test data.

3.2 Data Preprocessing steps :

3.2.1 Extracting Feature Values and labels form the data :

We extracted all the the 28x28 centered MNIST Data set images which now have 28*28=784 feature values and 1 target value.

The usps data set images are resized to the 28*28 centered image and extracted 28*28=784 feature values and 1 target value.

3.2.2 Target Values :

The target values of each data sample are in the range 0-9 whose values says the digit that data sample belongs to.

3.2.3 Feature vectors :

Each sample image have 784 feature vectors.

3.3 Data Division:

We divided the MNIST data into a training set, a validation set and a testing set. The training set takes around 80% of the total. The validation set takes about 10% . The testing set takes the rest without overlapping with the above. The USPS data set of 20000 samples are used for testing along with MNIST test data.

4. Classifiers

4.1 Training Logistic Regression Model :

4.1.1 The learning system :

we train a linear regression model dataset using the following methods :

Suppose we use 1-of-K coding scheme $t = [t_1,\ldots, t_K]$ for our multiclass classification task. Our multiclass logistic regression model could be represented in the form,

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)},$$

where the activation a_k are given by $a_k = w^T x + b_k$. The cross-entropy error function for multiclass classification problem in terms of a training sample x is,

$$E(x) \quad = -\sum_{k=1}^{k}(t_k ln y_k(x))$$

4.1.2 Logistic Regression using gradient descent (SGD) :

- It is the process of mininmizing a function by following the gradients of the cost function.
- This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.
- In Machine learning we can use a similar technique called stochastic gradient descent to minimize the error of a model on our training data.
- The gradient descent algorithm takes a random initial value. Then it updates the value of w using
  $W^{(\tau+1)}=W^{(\tau)}+\Delta W^{(\tau)}$

where $\Delta w(\tau) = -\eta^{(\tau)} *X*(y-y^{l})$ is called the weight updates. It goes along the opposite direction of the gradient of the error. η(τ) is the learning rate, deciding how big each update step would be. Because of the linearity of differentiation and y is the actual output where as y^{l} is the predicted output.

4.1.3 Experimental Values :

4.1.3.1 Hyperparameters Used :

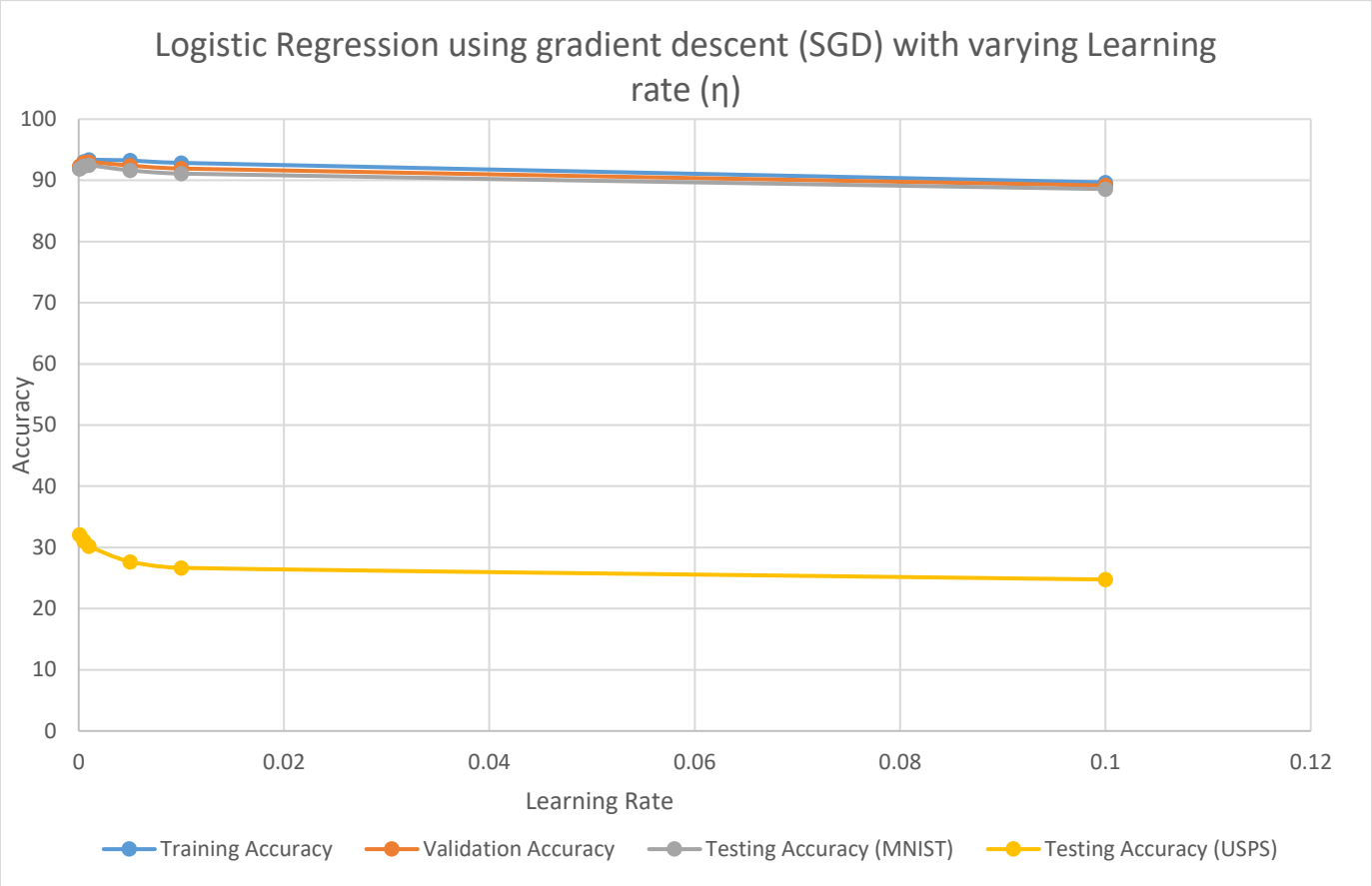
The following are the hyper parameters used in this model :

- Learning rate (η) – Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope.

4.1.3.2 Varying Learning Rate, (η) :

The Accuracy measures are calculated for different learning rate values :

Learning Rate	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
0.0001	91.966	92.32	91.79	32.897
0.0005	93.046	92.88	92.3	31.087
0.001	93.34	93.01	92.46	30.212
0.005	93.252	92.42	91.64	27.661
0.01	92.85	91.93	91.1	26.666
0.1	89.67	89.16	88.57	24.756



4.1.4 Observations :

We observed the accuracies are better for less learning rate 0.0001 whose mnist test accuracy is 91.87 and USPS test accuracy is 32.897

4.1.4.1 Confusion Matrices for Logistic Regression SGD :

Logistic SGD Confusion matrix MNIST Test data is

```
[[ 958    0    0    3    0    3    8    3    5    0]
 [   0 1110    2    2    0    3    4    1   13    0]
 [   5   12  907   21    8    4   16   12   41    6]
 [   3    1   17  920    1   25    2    9   23    9]
 [   2    3    7    1  910    0   13    3    9   34]
 [  11    3    7   38   10  753   19    9   34    8]
 [  11    3    4    2   10   12  910    3    3    0]
 [   2    6   23    8    7    1    0  946    3   32]
 [  10    7    6   22    9   34   10   12  861    3]
 [   9    8    1    8   37    7    1   25    9  904]]
```

Logistic SGD Confusion matrix USPS
[[399 3 251 103 148 367 100 70 116 443]
 [70 280 217 267 192 154 28 532 237 23]
 [105 17 1158 191 39 247 75 51 90 26]
 [49 3 174 1169 14 441 11 58 58 23]
 [44 42 46 69 776 151 64 299 301 208]
 [83 8 203 179 37 1234 101 54 82 19]
 [167 6 512 82 69 436 618 15 54 41]
 [147 136 149 597 36 109 16 441 255 114]
 [243 22 158 260 85 680 127 54 283 88]
 [29 83 106 524 108 73 18 471 367 221]]

4.1.4.2 Strengths and Weaknesses of Logistic Regression:

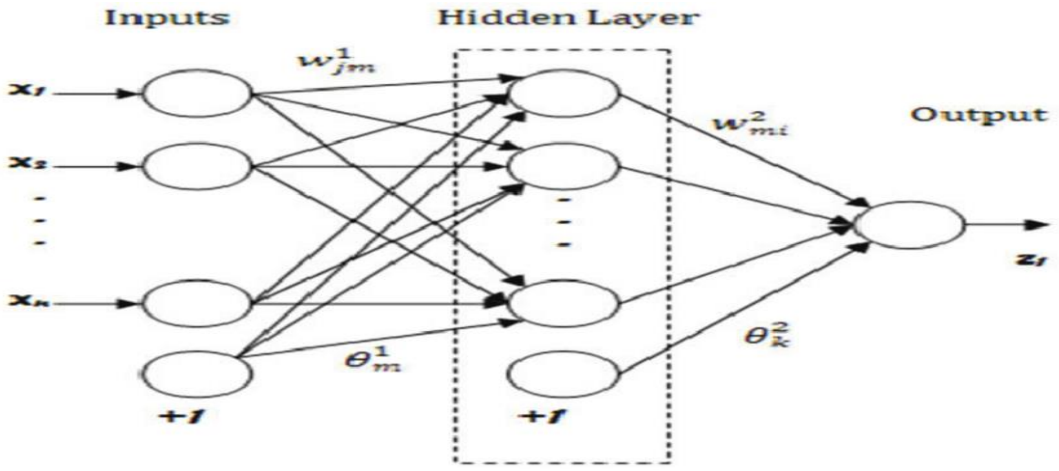
Strengths of Logistic Regression: Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent.

Weaknesses of Logistic Regression: Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

4.2 Neural Networks Model :

4.2.1 The learning system :

Making a model helps to explain a system and to study the effects of different components, and to make predictions about behavior. The model is created in keras which uses high level API built on Tensor Flow. It is more user-friendly and easy to use as compared to Tensor flow**.** Using Sequential model which is just a linear stack of layers. The Artificial Neural Network model will have inputs which are features, one dense hidden layer consists of several nodes, activation function and 2nd layer as outputs as shown in below figure.



The activation function introduce non linear properties to the network and converts input signal of a node to an output signal which can be used as a input to the next dense layer. Later To prevent Neural network from overfitting we are adding dropouts to the hidden layer which drops random weights which are causing overfitting to the model. To reduce the influence of extreme values in the data without removing them from the data set we are adding softmax activation function. Later Categorical crossentropy Loss function is used to calculate the amount of inaccuracy.

4.2.2 Experimental Values :

4.2.2.1 Hyper parameters Used:

The following are the hyper parameters used in this model.

- Number of Neuron/Nodes : These are the number of nodes/neurons in the first dense layer.
- Optimiser : This the function name of the optimizer used
- Number of hidden layers : These are number of hidden layers in the overall neural network used.
- Input batch size : This the number of data samples sent into model per single batch
- Number of epochs :This is the number of iterations the model should run for the given data set.
- Validation data split – This is the percentage of the data samples used for validating the model in each epoch.

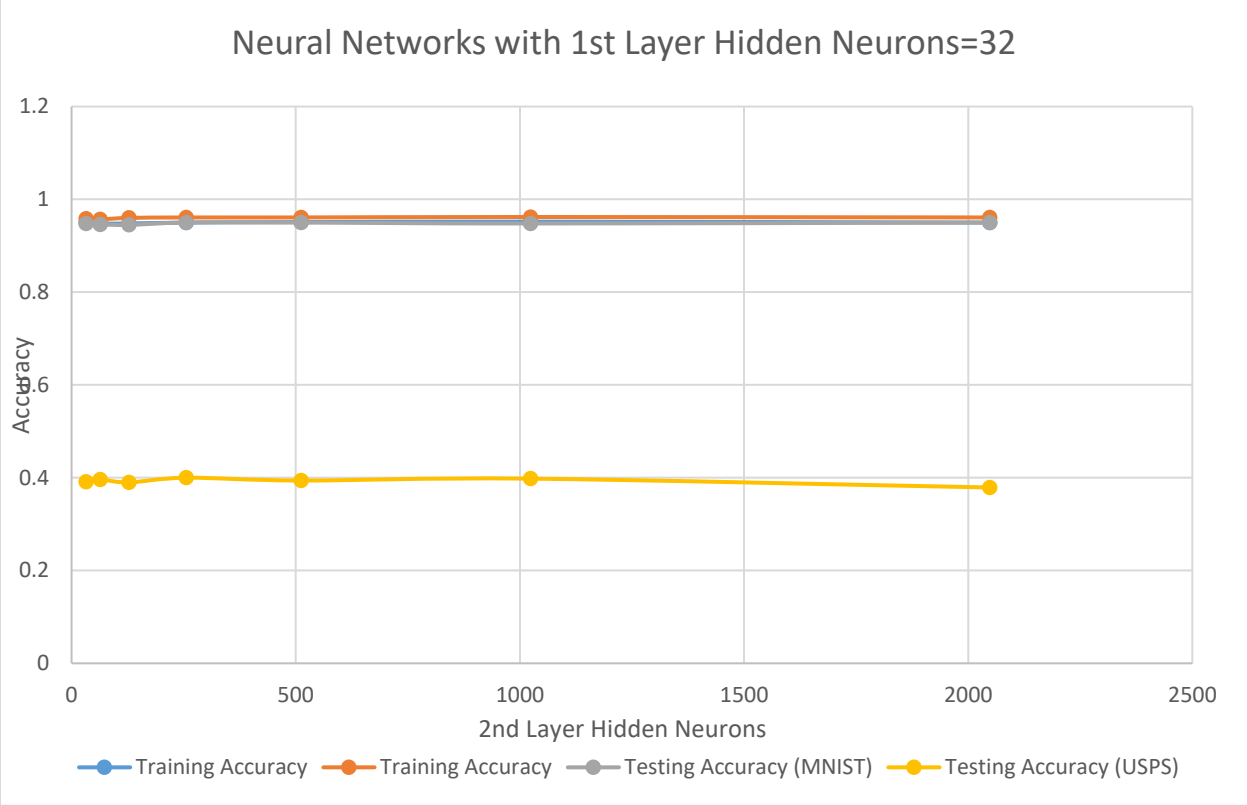
4.2.2.2 Model values are :

Based on the experimental results of varying hyper parameters we have chosen the following hyp er parameters.

Number of Neurons	32-2048
Activation function	Relu +sigmoid
Optimiser	sgd
Number of hidden layers	2
Input batch size	128
Number of epochs	100
Validation data split	0.1

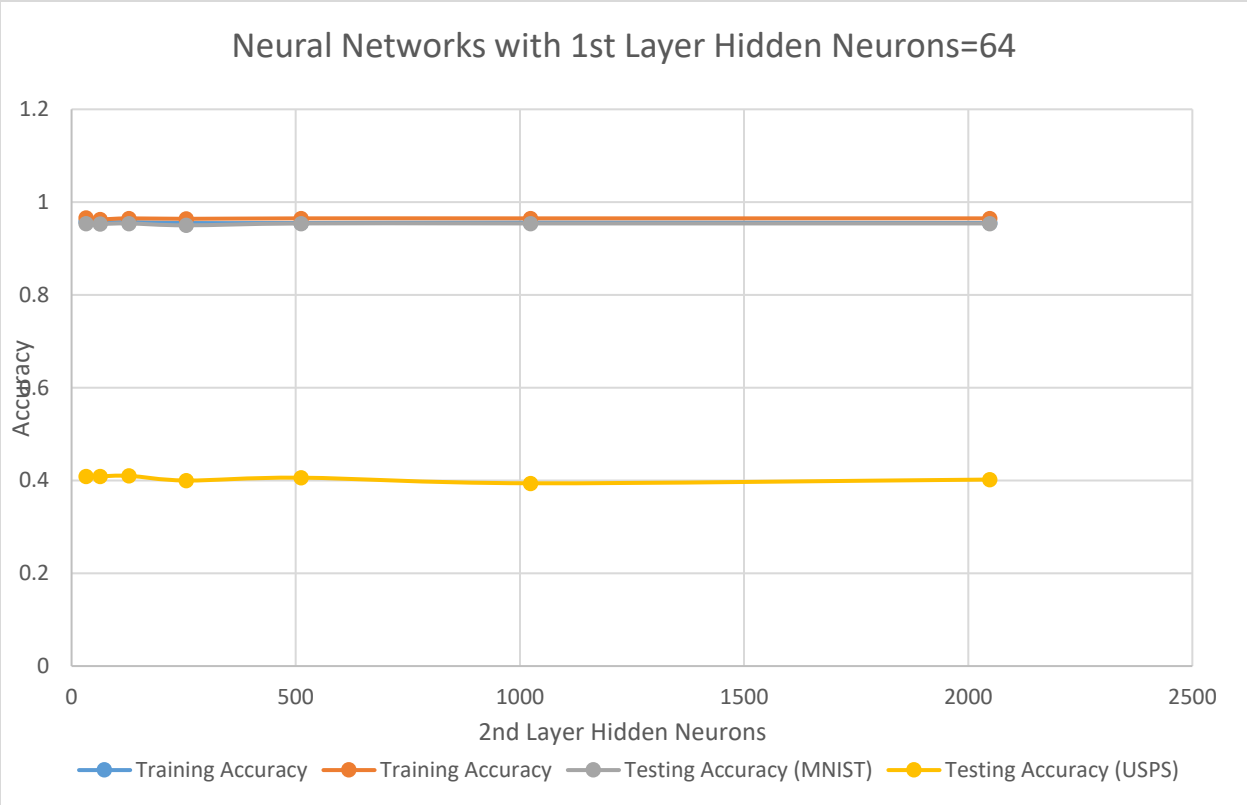
4.2.2.4 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 32

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.95	0.958	0.948	0.391
64	0.948	0.957	0.946	0.396
128	0.948	0.96	0.945	0.39
256	0.95	0.961	0.95	0.4
512	0.951	0.961	0.95	0.394
1024	0.952	0.962	0.948	0.398
2048	0.95	0.961	0.95	0.379



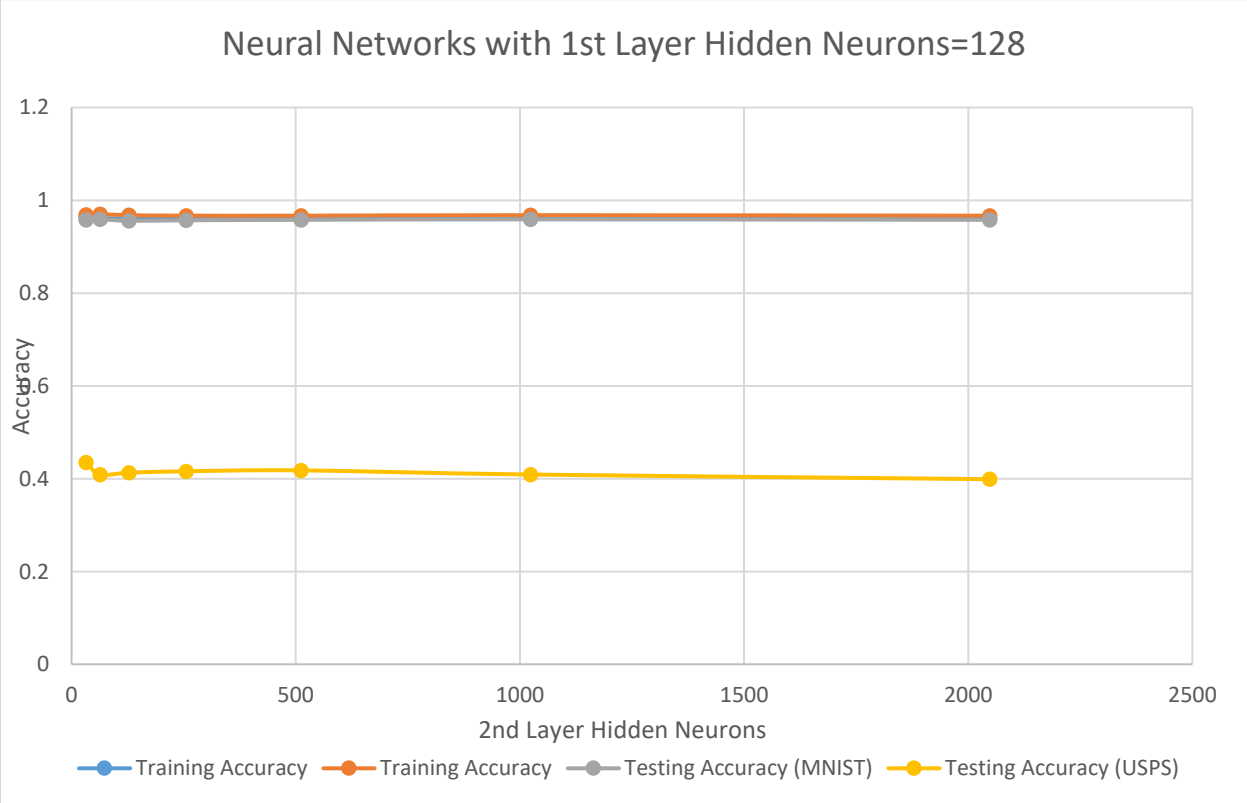
4.2.2.5 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 64

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.958	0.966	0.954	0.409
64	0.956	0.963	0.953	0.409
128	0.957	0.965	0.954	0.41
256	0.955	0.964	0.95	0.40
512	0.955	0.965	0.954	0.406
1024	0.956	0.965	0.954	0.394
2048	0.955	0.965	0.954	0.402



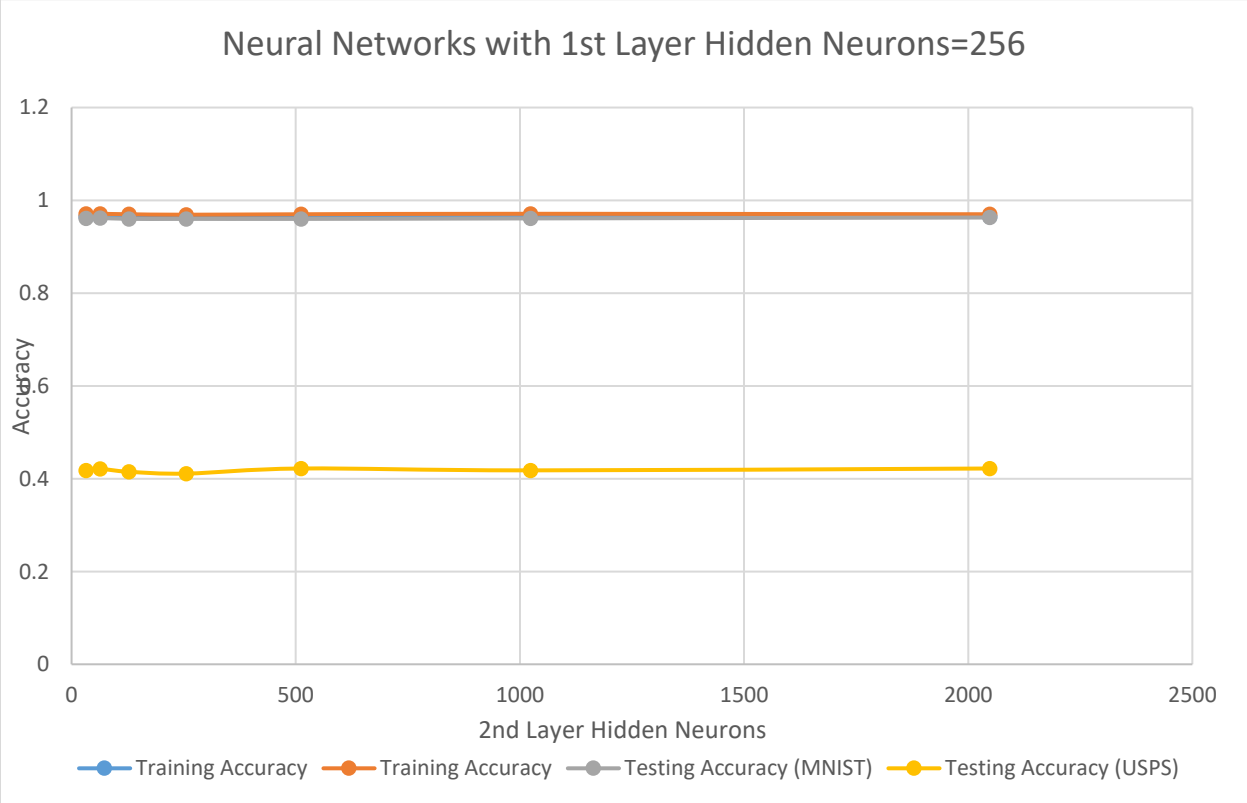
4.2.2.6 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 128

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.963	0.969	0.958	0.435
64	0.961	0.97	0.959	0.409
128	0.961	0.968	0.956	0.413
256	0.961	0.967	0.957	0.416
512	0.96	0.967	0.958	0.418
1024	0.963	0.968	0.959	0.409
2048	0.96	0.967	0.958	0.399



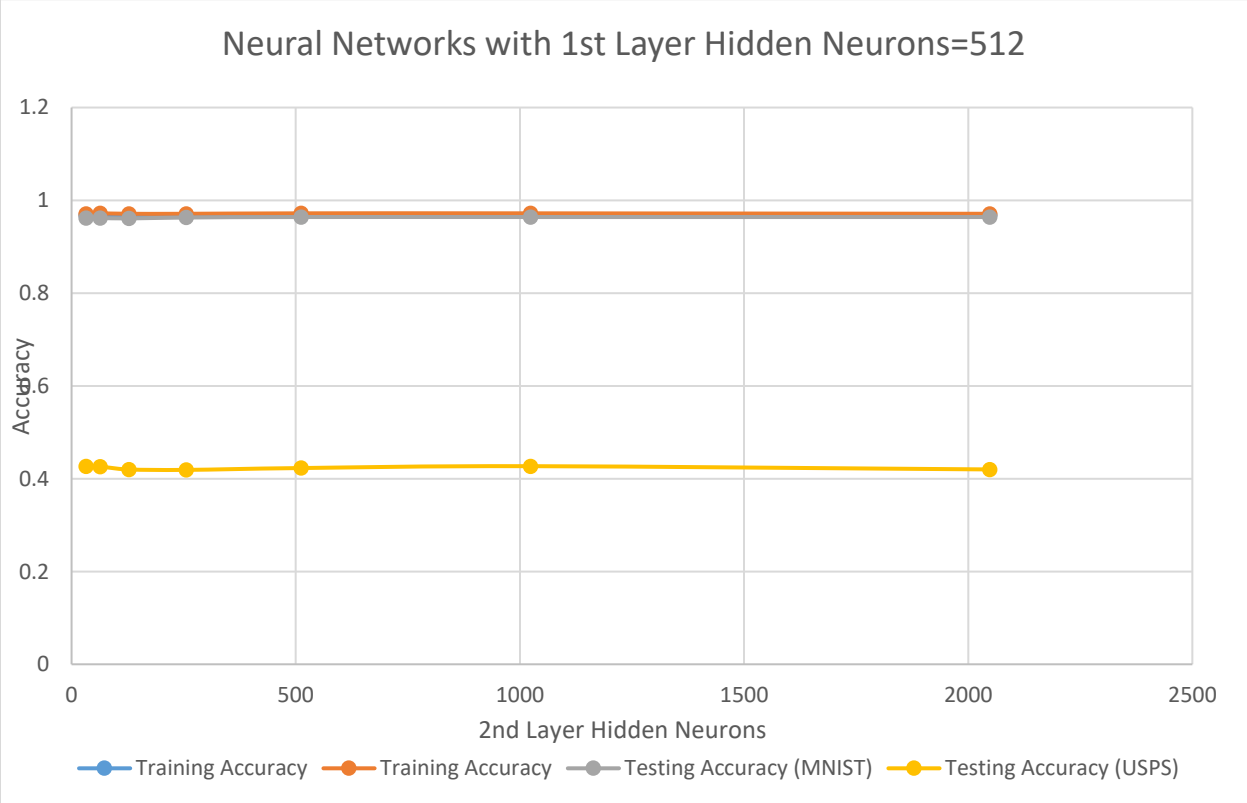
4.2.2.7 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 256

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.965	0.971	0.961	0.418
64	0.964	0.971	0.962	0.421
128	0.963	0.97	0.96	0.415
256	0.963	0.969	0.96	0.411
512	0.964	0.97	0.96	0.422
1024	0.964	0.971	0.961	0.418
2048	0.965	0.97	0.963	0.422



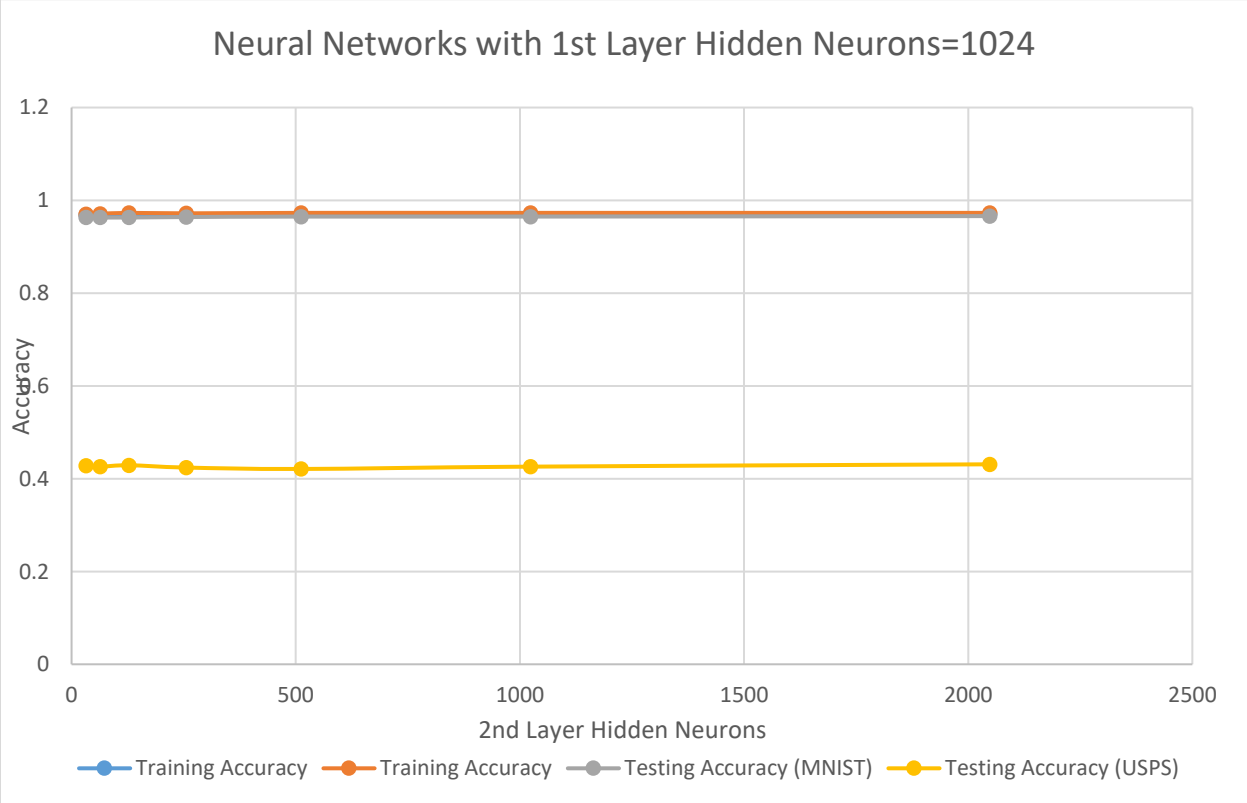
4.2.2.8 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 512

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.968	0.971	0.962	0.427
64	0.967	0.972	0.962	0.426
128	0.966	0.971	0.961	0.42
256	0.967	0.971	0.963	0.419
512	0.967	0.972	0.964	0.423
1024	0.967	0.972	0.964	0.427
2048	0.968	0.971	0.964	0.42



4.2.2.9 Varying 2nd Layer Hidden Neurons form 32 to 2048 with 1st Layer Hidden Neurons = 1024

2nd Layer Hidden No	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
32	0.969	0.97	0.963	0.428
64	0.968	0.971	0.963	0.426
128	0.967	0.973	0.963	0.429
256	0.969	0.972	0.964	0.424
512	0.969	0.973	0.965	0.421
1024	0.97	0.973	0.965	0.426
2048	0.97	0.973	0.966	0.431



4.2.3 Observations

We observed that the accuracies are better i.e testing accuracy for mnist is 0.966 and testing accuracy for USPS data is 0.431 for the following hyper parameters :

Number of Neuron	1024 (1st layer) + 2048 (2nd Layer)
Activation function	Relu +sigmoid
Optimiser	sgd
Number of hidden layers	2
Input batch size	128
Number of epochs	100
Validation data split	0.1

4.2.3.1 Confusion Matrices for Neural Network

Confusion matrix MNIST Test data

```
[[ 966    0    1    2    0    4    4    2    1    0]
 [   0 1119    4    0    0    1    4    2    5    0]
 [   5    1 1004    3    3    0    3    7    5    1]
 [   1    0   10  970    0   10    0   10    6    3]
 [   1    1    6    0  944    0    7    2    2   19]
 [   8    1    0    9    2  853    9    1    5    4]
 [   8    3    0    0    7    8  928    1    3    0]
 [   1    9   12    4    2    1    0  988    1   10]
 [   5    2    3    8    4    6    9    8  925    4]
 [   6    6    1    8   17    5    1    7    3  955]]
```

Confusion matrix USPS test data :

```
[[ 569    1  195   72  145  157   52  103  120  586]
 [  74  379  327  114   87   78   29  760   91   61]
 [  65    8 1485  107   25  144   62   50   41   12]
 [  44    3  142 1427    3  261    5   45   60   10]
 [  23   52   59   33  924  131   21  410  220  127]
 [ 110    4  159  143   20 1365   51   42   92   14]
 [ 142    7  448   56   61  273  907   20   46   40]
 [  66  134  365  444   20  125    6  697  116   27]
 [ 180   12  223  282   65  550   73   92  471   52]
 [  20   93  113  408   95   62    3  610  317  279]]
```

4.2.3.2 Strengths and Weaknesses of *Neural Network* :

Strengths: Deep learning performs very well when classifying for audio, text, and image data.
Weaknesses: As with regression, deep neural networks require very large amounts of data to train, so it's not treated as a general-purpose algorithm.

4.3 Support Vector Machines

4.3.1 Learning System

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane i.e given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

4.3.2 Tuning parameters

4.3.2.1 Kernel :

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B(0) + sum(ai * (x,xi))$$

4.3.2.2 Regularization :

The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much we want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

4.3.2.3 Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close' i.e with low gamma, points far away from plausible seperation line are considered in calculation for the seperation line. Where as high gamma means the points close to plausible line are considered in calculation.

4.3.2.4 Margin

A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. A good margin allows the points to be in their respective classes without crossing to other class.

4.3.3 Experimental Values :

4.3.3.1 Hyper parameters Used:

- Gamma : The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

4.3.3.2 Values for Various Settings :

The Experimental values of SVM classifer for various setting mentioned in the Appendix are :

Kernel	gamma	TrA	VaA	TeA	TeA_USPS
linear	auto	0.972	0.942	0.939	0.291
rbf	default	0.94	0.945	0.944	0.385
rbf	1	1.0	0.182	0.176	0.1

4.3.4 Observations :

We observed that for SVM classifier, the testing accuracy is more for setting whose kernel is 'rbf' and gamma is 'default' is 94.4% for Mnist data and 38.5% for USPS data.

4.3.4.1 Confusion Matrices of SVM :

Confusion matrix for MNIST Test Data

```
[[ 966    0    1    0    1    5    5    1    1    0]
 [   0 1118    3    2    0    1    4    1    6    0]
 [   9    2  934   10   14    2   18   14   27    2]
 [   2    2   19  932    1   21    1   11   17    4]
 [   1    4    7    0  919    0   10    3    2   36]
 [   7    6    5   36    9  795   14    4   11    5]
 [  10    3    5    1    6   14  918    0    1    0]
 [   3   17   23    4   10    1    0  939    4   27]
 [   4    8    8   18    8   28   11    8  873    8]
 [  10    9    1   12   39    7    1   13    6  911]]
```

Confusion matrix for USPS Test data

```
[[ 565    4  409   18  336  223   77   37   11  320]
 [  98  455  240  124  417  190   46  397   19   14]
 [ 107   23 1345   70   48  213   67   84   31   11]
 [  71    7  164 1118   15  506    6   63   27   23]
 [  18   95   62   16 1238  245   16  162   64   84]
 [ 137   20  188  100   34 1371   69   52   18   11]
 [ 236    8  441   31  115  377  740   14   13   25]
 [  49  262  393  245   86  407   28  440   57   33]
 [ 114   32  193  186  119  927   97   40  249   43]
 [  33  197  189  255  277  163   13  486  205  182]]
```

4.3.4.2 Strengths and Weaknesses of State Vector Machines :

- **Strengths :** SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.
- **Weaknesses**: However, SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets. Currently in the industry, random forests are usually preferred over SVM's.

4.4 Random Forest Classifier

4.4.1 Learning System

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.

4.4.2 Tuning parameters

4.4.2.1 Trees :

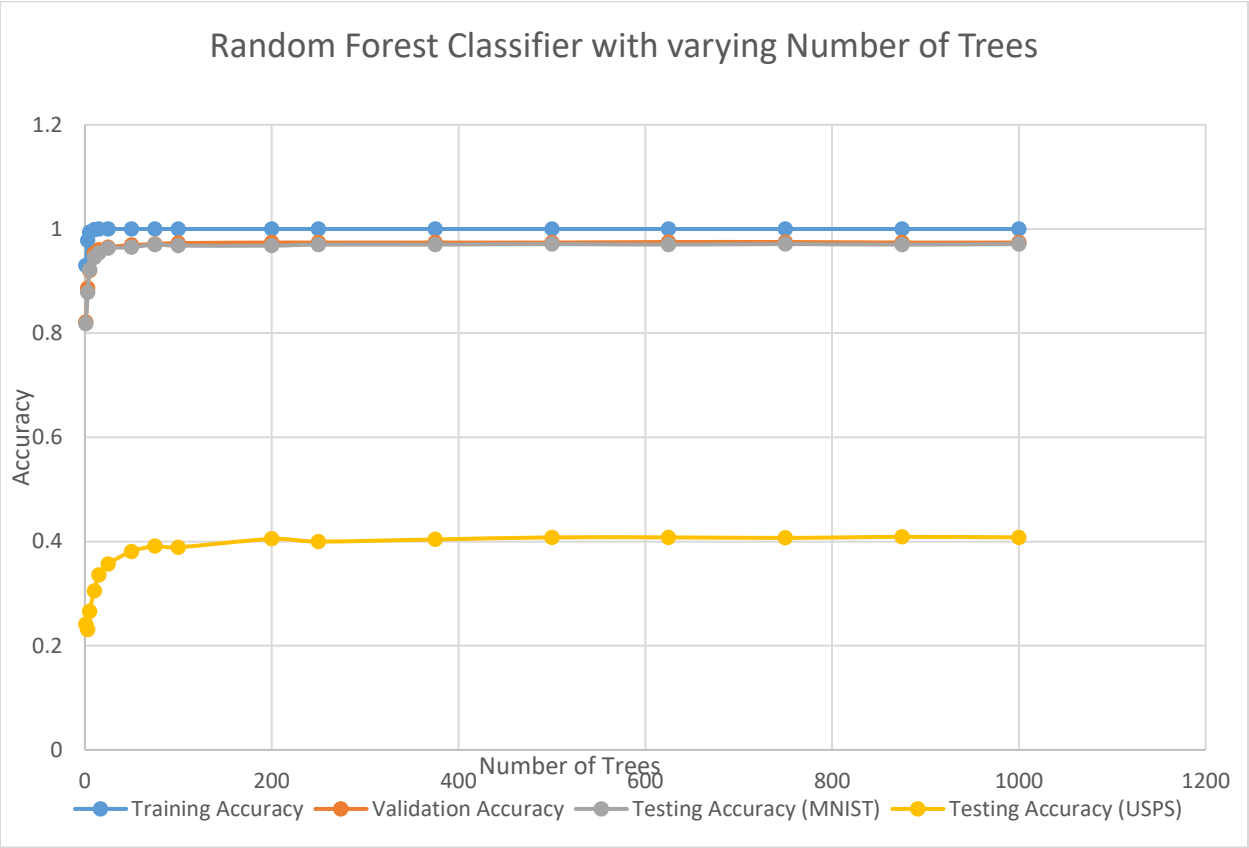
These are the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

4.4.3 Experimental Values :

4.4.3.1 Hyper parameters Used:

- n_estimators : These are the number of trees.

No. of trees	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
1	0.93	0.821	0.818	0.241
3	0.978	0.886	0.878	0.231
5	0.994	0.92	0.922	0.266
10	0.999	0.953	0.945	0.305
15	1.0	0.96	0.954	0.336
25	1.0	0.965	0.963	0.357
50	1.0	0.969	0.965	0.381
75	1.0	0.971	0.97	0.391
100	1.0	0.973	0.968	0.389
200	1.0	0.974	0.968	0.405
250	1.0	0.974	0.97	0.4
375	1.0	0.974	0.97	0.404
500	1.0	0.974	0.971	0.408
625	1.0	0.975	0.97	0.408
750	1.0	0.975	0.971	0.407
875	1.0	0.974	0.97	0.409
1000	1.0	0.974	0.971	0.408



4.4.4 Observations :

We observed that the accuracies are increased wrt to no. of trees until 200 and from there it was almost saturated. The best test accuracy observed is 0.971 for MNIST data and 0.409 for USPS data

4.4.4.1 Confusion matrix for MNIST test data for Random Forest with no. of trees =200

```
[[ 969   0   0   0   0   2   3   1   4   1]
 [  0 1124   2   3   0   2   2   1   1   0]
 [  6   0 999   7   2   0   4   8   6   0]
 [  0   0   9 972   0   9   0   9   9   2]
 [  1   0   1   0 957   0   4   1   3  15]
 [  2   0   1  15   4 858   5   1   4   2]
 [  7   3   0   0   2   4 938   0   4   0]
 [  1   4  16   1   1   0   0 992   1  12]
 [  3   0   6   8   6   4   5   3 927  12]
 [  7   5   2   8  14   5   1   5   3 959]]
```

Confusion matrix for USPS test data for Random Forest with no. of trees =200

```
[[ 649   12  270   60  418  159   61  117    3  251]
 [  44  566  127  105   43   95   23  977   18    2]
 [  85   30 1283   81   55  168   18  272    5    2]
 [  36   10   93 1268   51  326    3  187    5   21]
 [  12  215   66   22 1061  180   12  378   30   24]
```

```
[ 129   31  134   90   26 1437   20  121    5    7]
[ 309   45  265   27   86  332  780  141    4   11]
[  44  344  355  253   35  231   32  693    6    7]
[  44   50  156  210   89 1064   74  108  185   20]
[  13  260  253  317  236  129   10  598   83  101]]
```

4.4.4.2 Strengths and Weaknesses of Random Forest Classifier:

Strengths of Random Forest Classifier: As with regression, classification tree ensembles also perform very well in practice. They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries thanks to their hierarchical structure.

Weaknesses of Random Forest Classifier: Unconstrained, individual trees are prone to overfitting, but this can be alleviated by ensemble methods.

4.5 Mini Batch Stochastic gradient descent with Logistic Regression Model :

4.5.1 The learning system :

we train a linear regression model dataset using the following methods :

Suppose we use 1-of-K coding scheme t = [t_1,…, t_K] for our multiclass classification task. Our multiclass logistic regression model could be represented in the form,

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)},$$

where the activation a_k are given by $a_k = w^T x + b_k$. The cross-entropy error function for multiclass classification problem in terms of a training sample x is,

$$E(x) \quad = -\sum_{k=1}^{k}(t_k ln y_k(x))$$

4.5.2 Logistic Regression using mini batch stochastic gradient descent (SGD) :

- It is the process of minimizing a function by following the gradients of the cost function.
- This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.
- In Machine learning we can use a similar technique called stochastic gradient descent to minimize the error of a model on our training data.
- The mini batch stochastic gradient descent algorithm takes a random initial value. Then it updates the value of w using
  $W^{(\tau+1)}=W^{(\tau)}+\Delta W^{(\tau)}$

where $\Delta w(\tau) = -\eta^{(\tau)} * X * (y-y^l)$ is calculated using mini batch of size 'n' data samples and will update the weights for every n data samples and this process is called the weight updates. It goes along the opposite direction of the gradient of the error. η(τ) is the learning rate, deciding how big each

update step would be. Because of the linearity of differentiation and y is the actual output where as y^{l} is the predicted output.

4.5.3 Experimental Values :

4.5.3.1 Hyperparameters Used :

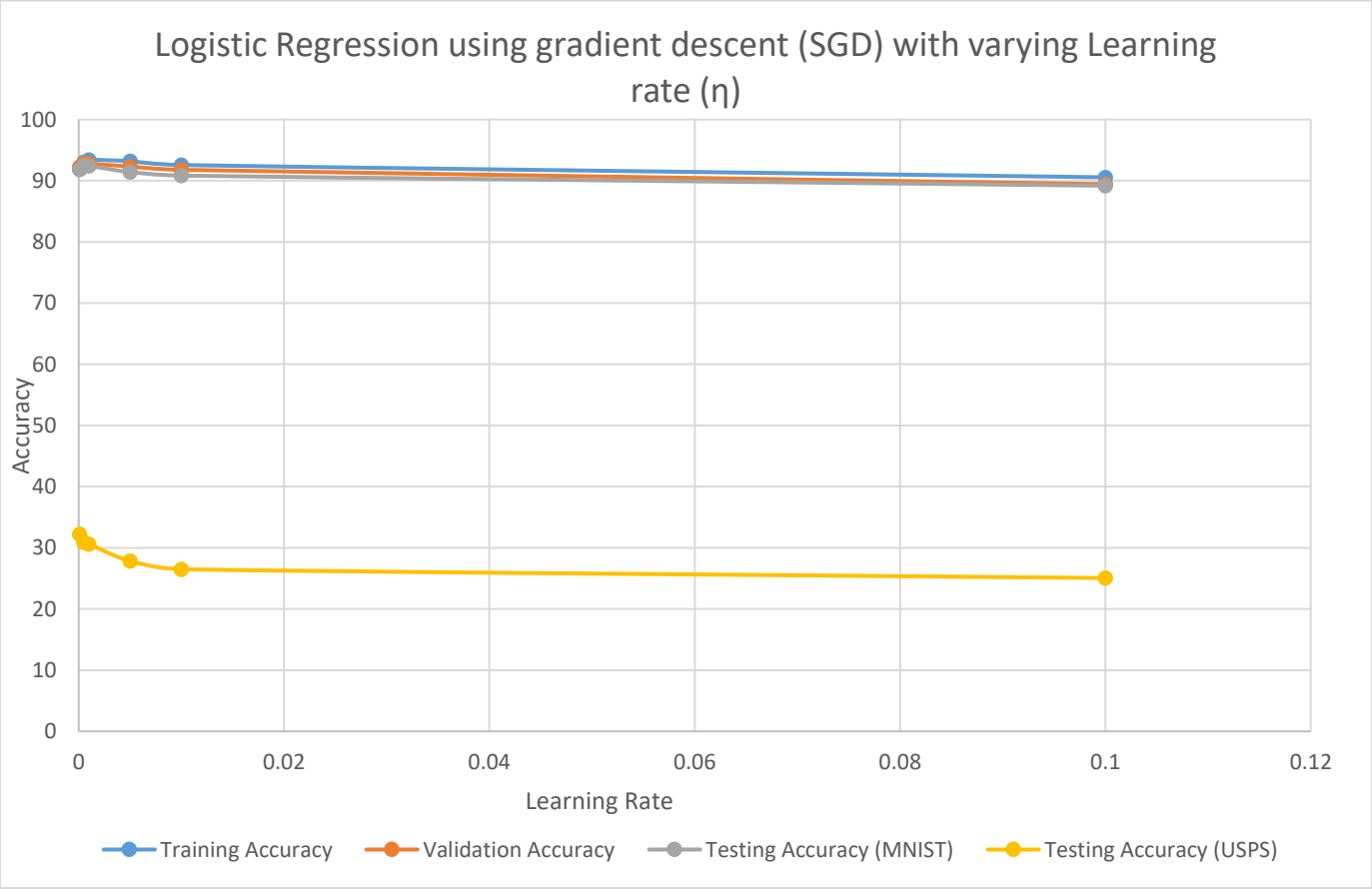
The following are the hyper parameters used in this model :

- Learning rate (η) – Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope.

4.5.3.4 Varying Learning Rate, (η) :

The Accuracy measures are calculated for different learning rate values :

Learning Rate	Training Accuracy	Validation Accuracy	Testing Accuracy (MNIST)	Testing Accuracy (USPS)
0.0001	91.904	92.26	91.84	32.242
0.0005	93.058	92.87	92.45	30.942
0.001	93.414	92.81	92.42	30.612
0.005	93.188	92.32	91.42	27.831
0.01	92.564	91.78	90.84	26.501
0.1	90.558	89.43	89.18	25.061



4.5.6 Observations :

We observed the accuracies are better for less learning rate 0.0001 whose mnist test accuracy is 91.84 and USPS test accuracy is 32.242

4.5.6.1 Confusion Matrices for Logistic Regression mini batch SGD :

Confusion matrix for MNIST data :

```
[[ 956    0    1    2    0    7   11    1    2    0]
 [   0 1111    2    2    0    2    4    1   13    0]
 [   8    9  914   18    9    1   12   11   41    9]
 [   1    0   18  916    0   29    2   12   25    7]
 [   2    3    4    2  919    0    7    2    7   36]
 [  10    3    5   43   10  756   16    6   37    6]
 [   8    3    5    2   10   17  909    2    2    0]
 [   2    5   20   10   10    0    0  947    6   28]
 [   7    7    9   23    9   31    9   14  861    4]
 [  10    6    1   10   38    7    0   27    5  905]]
```

Confusion matrix for USPS data :

```
[[ 478    2  324   94  119  293   94  156  205  235]
 [  85  266  289  191  208  105   20  594  222   20]
 [ 131   11 1266  128   30  177   75   63   94   24]
 [  69    3  247 1068   17  417    7   86   66   20]
 [  50   41   64   54  772  154   73  321  293  178]
 [  76   11  307  167   32 1150   75   65  103   14]
 [ 184    5  528   92   61  457  599   12   39   23]
 [ 154  111  187  608   67  133   10  409  253   68]
 [ 243   19  137  299  127  662  106   55  290   62]
 [  25   87   95  543  124  128   11  455  334  198]]
```

4.5.6.2 Strengths and Weaknesses of Logistic Regression:

Strengths of Logistic Regression: Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent. When compared to mini batch stochastic gradient descent it won't update over each data sample instead update a batch size which decreases the computation time

Weaknesses of Logistic Regression: Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

4.6 Ensmeble Classifier

4.6.1 Majority/Hard Voting :

In majority voting, the predicted class label for a particular sample is the class label that represents the majority (mode) of the class labels predicted by each individual classifier.
E.g., if the prediction for a given sample is

- classifier 1 -> class 1
- classifier 2 -> class 1
- classifier 3 -> class 2
- classifier 4 -> class 1
- classifier 5 -> class 2

The VotingClassifier (with voting='hard') would classify the sample as “class 1” based on the majority class label.
In the cases of a tie, the VotingClassifier will select the class based on the ascending sort order. E.g., in the following scenario

- classifier 1 -> class 2
- classifier 2 -> class 1

the class label 1 will be assigned to the sample.

The accuracies with majority voting for following classifiers and hyper parameters are :

1. Logistic Regression with Stochastic Gradient descent with Learning rate 0.0001.
2. Neural Networks Hyper parameters are :

Number of Neuron	1024 (1st layer) + 2048 (2nd Layer)
Activation function	Relu +sigmoid
Optimiser	sgd
Number of hidden layers	2
Input batch size	128
Number of epochs	100
Validation data split	0.1

3. SVM classifier with kernel = ‘rbf’ and gamma=’default’
4. Random Forest classifier with NO. of trees = 200
5. Logistic Regression with Mini Batch Stochastic Gradient descent with Learning rate 0.0001.

Majority Voting MNIST Test accuracy is 0.9675
Majority Voting USPS Test accuracy is 0.4224

4.6.1.2 Confusion matrix for Majority_Voting :

Confusion matrix for MNIST dataset :

```
[[ 968    0    1    1    0    3    4    1    2    0]
 [   0 1123    2    2    0    1    3    1    3    0]
 [   7    1  997    5    3    0    3    8    7    1]
```

```
[  0   0   9 973   0   7   0  11   8   2]
[  1   0   3   0 955   0   6   2   2  13]
[  7   1   0  12   2 854   7   1   6   2]
[  7   3   0   0   4   8 933   1   2   0]
[  1   7  15   2   2   1   0 989   0  11]
[  4   1   4   9   5   5   8   8 926   4]
[  8   6   1   9  14   3   1   7   3 957]]
```

Confusion matrix for USPS data :

```
[[ 586    2  245   69  209  191   57  103   78  460]
 [  65  413  270  114   56   87   24  851   83   37]
 [  77   10 1478   93   29  149   43   86   28    6]
 [  38    3  131 1411    4  290    4   63   48    8]
 [  17   77   56   22  991  150   17  367  202  101]
 [  99   11  181  124   15 1418   37   51   56    8]
 [ 172    9  423   48   55  325  886   26   28   28]
 [  61  190  324  441   24  164   10  667   97   22]
 [ 136   18  180  274   67  750   74   93  365   43]
 [  17  124  117  413  116   78    5  596  301  233]]
```

4.6.2 Observations :

The overall combined performance i.e majority voting is almost similar to that of best individual classifier and is better than worst classifiers.

5 Conclusions and Summary :

- We observed that the Neural Network model gave the best accuracy compared to other classifiers. The hyper parameters used are :

Number of Neuron	1024 (1st layer) + 2048 (2nd Layer)
Activation function	Relu +sigmoid
Optimiser	sgd
Number of hidden layers	2
Input batch size	128
Number of epochs	100
Validation data split	0.1

- "No Free Lunch" theorem states that there is no one algorithm works best for every problem, and it's especially relevant for supervised learning (i.e. predictive modeling).
- We proved "No Free Lunch" theorem by calculating test accuracy of USPS data in every classifier which is just giving 30-45% of accuracy and MNIST data which got around 95.
- The overall combined performance is almost similar to that of best individual classifier and is better than worst classifiers.