ELL409: Assignment 2

Neural Network for Handwritten Digits Recognition

Name : Aarushi Gupta

Entry Number: 2019MT60738

Name : Navya Arora

Entry Number: 2019MT10707

INTRODUCTION:

Neural Networks have become ubiquitous in their use for solving complicated Machine Learning problems involving large amounts of data due to their universal modelling properties. We explore the Fully Connected 3 Layer Neural Network, experimenting on different learning rates, regularisation parameters and hidden layer sizes, in this assignment by training it on hand written digits dataset.

BUILDING AND ANALYSING THE NEURAL NETWORK:

EXPLORATION DONE:

- Found an optimum number of neurones for the hidden layer
- Found an optimum learning parameter for the two weight matrices separately.
- Found an optimum regularisation parameter for both the layers (Layer1 → Layer2, Layer1 → Layer2) separately.

METHODOLOGY USED:

- [ ] Forward Propagation
- [ ] Backward Propagation
- [ ] Gradient Descent
- [ ] Cross-Validation

ERROR FUNCTION:

$$E(\vec{w}) = \frac{-1}{N}\left[\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)}\log\left(h(x^{(i)})_k\right) + \left(1-y_k^{(i)}\right)\log\left(1-h(x^{(i)})_k\right)\right] + \frac{\lambda}{2N}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

Aim: To minimize the sum of this error function for each each example for each of the weights.

PROCEDURE:

1. Divided the Dataset randomly into Training and Testing points: Trained on 75% (4250 points) and tested on 15% (750 points) of the dataset.
   The randomness was implemented using `random.shuffle(Data)`
2. Implementing Cross Validation in the following steps:

- Depending on the folds (we implemented using 3) , we divide this Training set into k further subsets. Each of this set was taken as a validation set in one iteration and a model was trained on the (k-1) sets. The error was estimated on this validation set.
- After obtaining validation error on all of the k validation sets, the final cross validation error was average of all of these.

The above procedure was used to explore the first two points mentioned under the heading "Exploration Done".

FOR OPTIMUM LEARNING RATES:

Since we implemented a 3 layer model, we had two set of weight matrices to optimise. Both of these weight matrices had a corresponding learning rate as well:

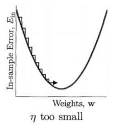
```
w1 = w1 - learning_rate_1*dJdw1
w2 = w2 - learning_rate_1*dJdw2
```

Our Hypothesis Set (with different learning rates):

Learning Rate Set = [0.001, 0.01, 0.1, 1,5]

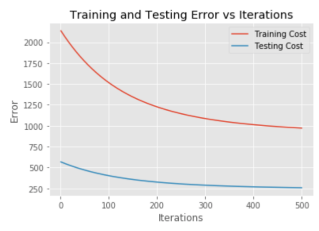
The set was chosen so to understand all the three possible scenarios:

- **LEARNING WITH A SMALL STEP-SIZE**



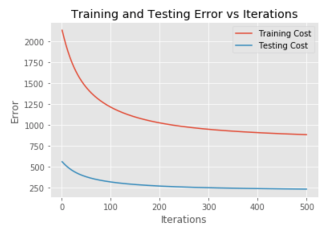
Results(Cross-validation accuracies) obtained:

```
For learning_rate 1= 0.001 and learning_rate 2= 0.001
```



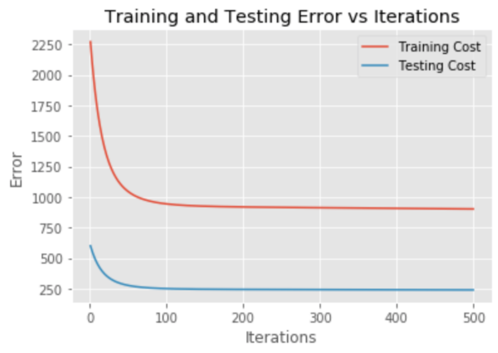
```
Validation Accuracy:% 12.994350282485879
Train Accuracy:% 12.80875088214538
Test Accuracy:% 14.000000000000002
```

```
For learning_rate 1= 0.01 and learning_rate 2= 0.001
```



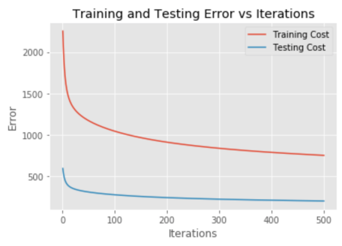
```
Validation Accuracy:% 39.26553672316384
Train Accuracy:% 40.96683133380381
Test Accuracy:% 36.8
```

```
For learning_rate 1= 0.001 and learning_rate 2= 0.01
```



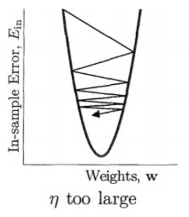
```
Validation Accuracy:% 22.669491525423723
Train Accuracy:% 21.5243472124206
Test Accuracy:% 19.999999999999996
```

```
For learning_rate 1= 0.1 and learning_rate 2= 0.001
```

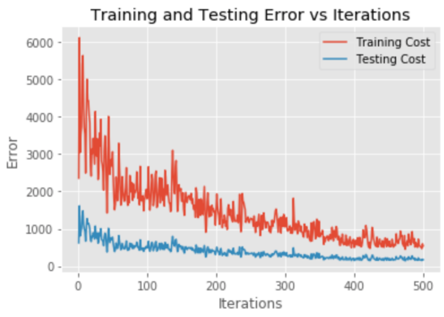


```
Validation Accuracy:% 64.61864406779661
Train Accuracy:% 66.3726182074806
Test Accuracy:% 66.4
```

- **LEARNING WITH A BIG STEP-SIZE**

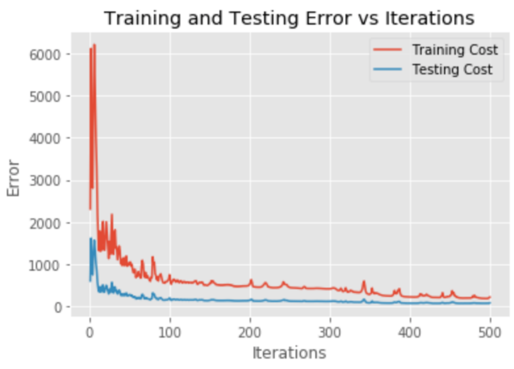


Results(Cross-validation accuracies) obtained:



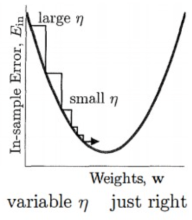
```
Validation Accuracy:% 58.89830508474576
Train Accuracy:% 62.9498941425547
Test Accuracy:% 57.333333333333336
```

```
For learning_rate 1= 5 and learning_rate 2= 5
```



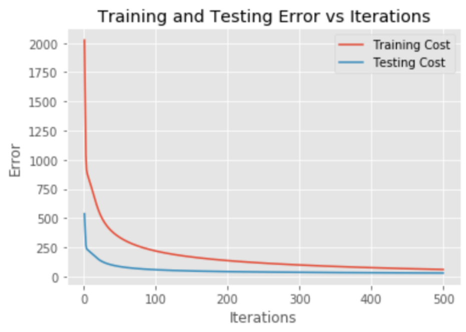
```
Validation Accuracy:% 72.31638418079096
Train Accuracy:% 78.44036697247707
Test Accuracy:% 76.0
```

- **LEARNING WITH STEP-SIZE JUST RIGHT**



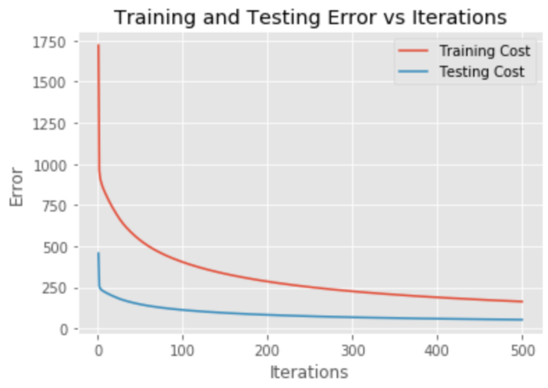
<u>Results(Cross-validation accuracies) obtained</u>:

```
For learning_rate 1= 1 and learning_rate 2= 1
```



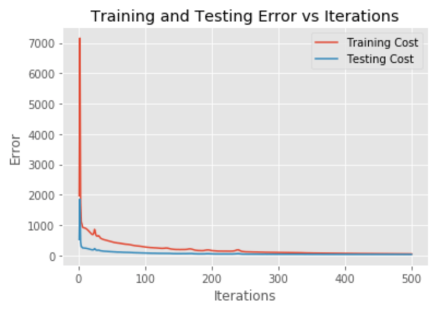
```
Validation Accuracy:% 95.62146892655367
Train Accuracy:% 98.1651376146789
Test Accuracy:% 92.93333333333334
```

```
[illegible]
For learning_rate 1= 1 and learning_rate 2= 0.1
```

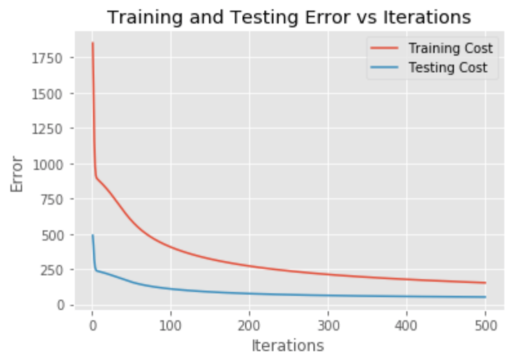


```
Validation Accuracy:% 91.80790960451978
Train Accuracy:% 94.46012702893437
Test Accuracy:% 91.2
```

```
For learning_rate 1= 1 and learning_rate 2= 5
```

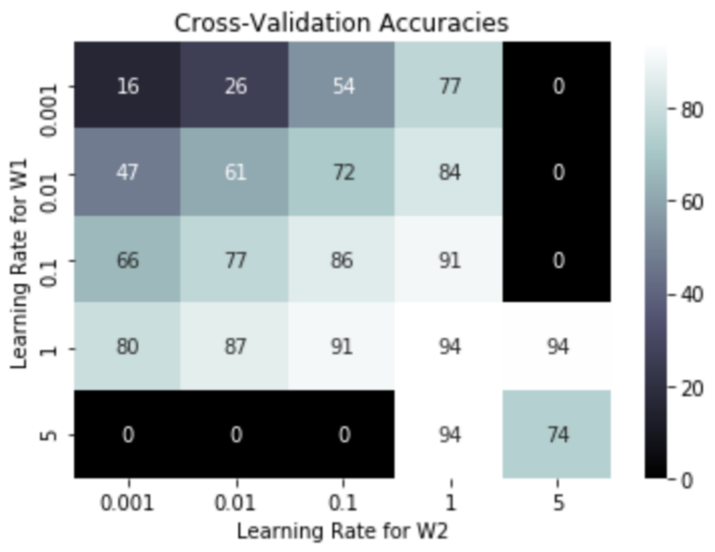


```
Validation Accuracy:% 94.27966101694916
Train Accuracy:% 98.97671136203246
Test Accuracy:% 93.60000000000001
```



```
Validation Accuracy:% 91.454802259887
Train Accuracy:% 94.0677966101695
Test Accuracy:% 90.53333333333333
```

Deductions:



	0.001	0.01	0.1	1	5
0.001	16	26	54	77	0
0.01	47	61	72	84	0
0.1	66	77	86	91	0
1	80	87	91	94	94
5	0	0	0	94	74

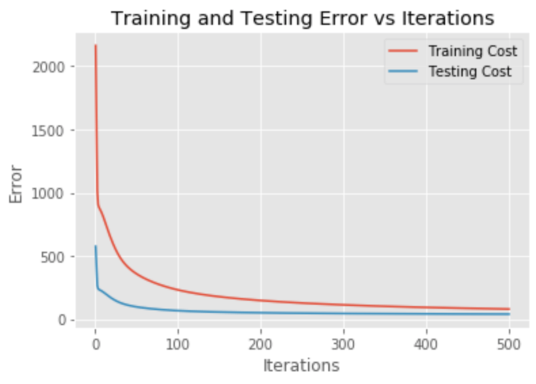
- The above pairs of values of alpha1, aplha2, learning rates are very small, so the model has a small step size, therefore it's inefficient to reach the optimum values for weights, iterations required will be very high to achieve this.
- It is clearly visible from the heatmap, that as the learning rates(both) start increasing, the colour gets fainter, showing that the results are getting better. Thus, we are able to reach the optimum values for weights quicker with this increase in step size.
- We attain the best values of cross-validation set error for alpha1=5 and alpha2=1 and alpha1=1, alpha2=1 with an accuracy of 94%. But we do not

choose (5,1) as our optimum value for learning rates as we see that the graph is a little bumpy showing that the step size might be a little large which leads to bouncing around the minimum.

✓ BEST LEARNING RATES = 1, 1 with ACCURACY = **94.32%**

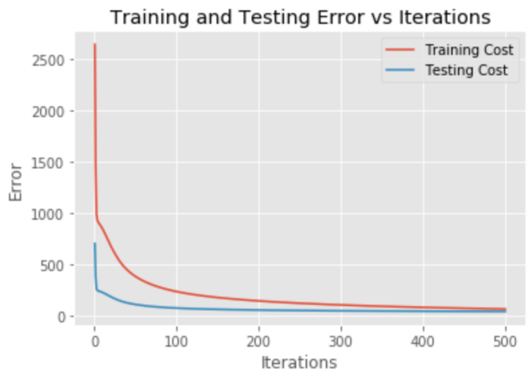
FOR OPTIMUM REGULARISATION PARAMETER:

```
For lambda 1= 0.5 and lambda 2= 5
```



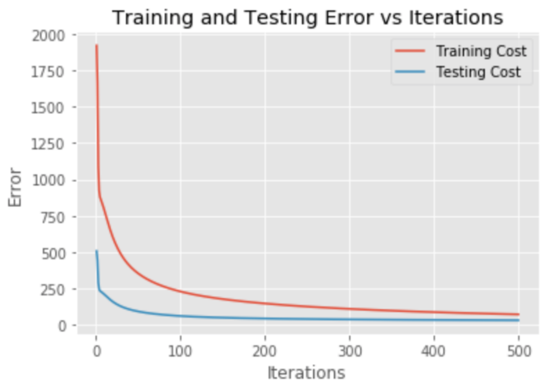
```
Validation Accuracy:% 92.93785310734464
Train Accuracy:% 98.30628087508822
Test Accuracy:% 93.73333333333333
```

```
For lambda 1= 0.1 and lambda 2= 0.5
```



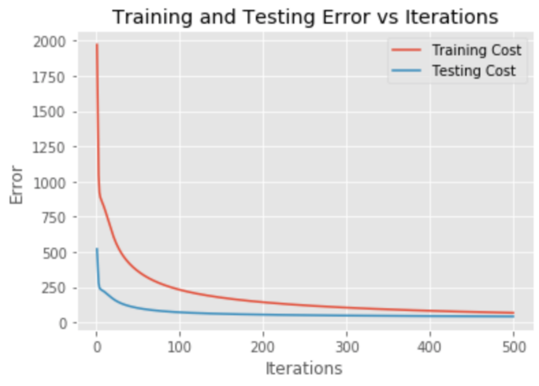
```
Validation Accuracy:% 95.76271186440678
Train Accuracy:% 98.72971065631616
```

```
For lambda 1= 1 and lambda 2= 1
```



```
Validation Accuracy:% 93.99717514124293
Train Accuracy:% 98.02399435426959
Test Accuracy:% 94.0
```

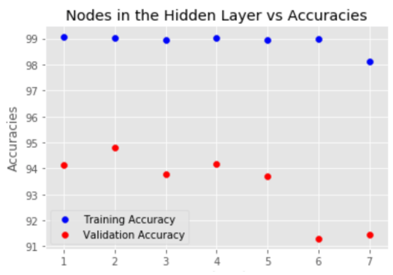
```
For lambda 1= 1 and lambda 2= 0.5
```

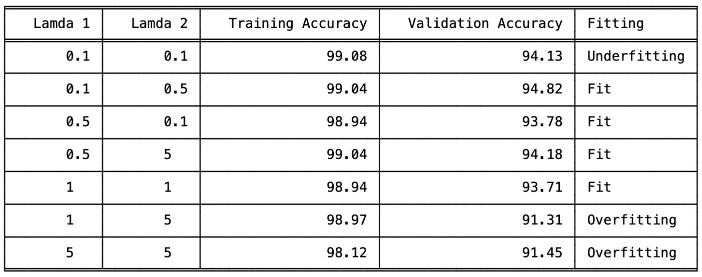


```
Validation Accuracy:% 94.42090395480226
Train Accuracy:% 98.55328158080452
Test Accuracy:% 92.26666666666667
```

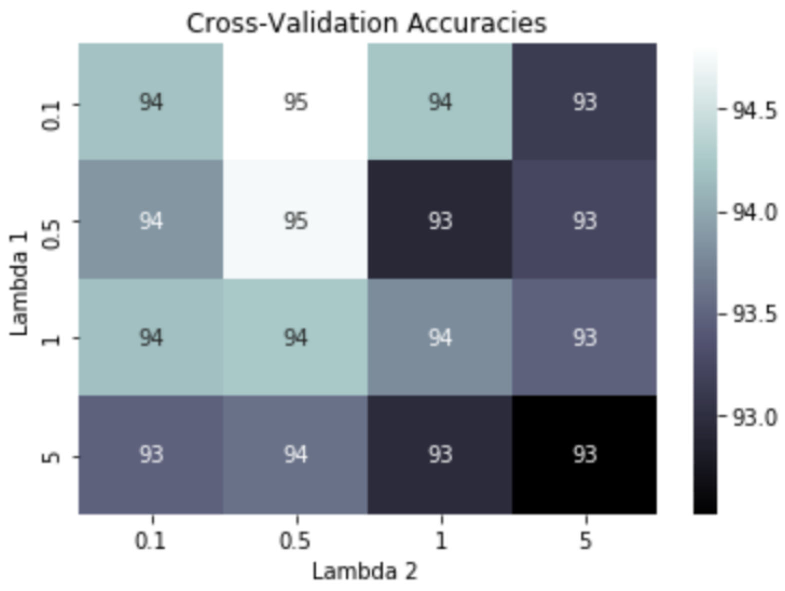
__Deductions__:

Differentiating between undercutting, overfitting and best fit:





Lamda 1	Lamda 2	Training Accuracy	Validation Accuracy	Fitting
0.1	0.1	99.08	94.13	Underfitting
0.1	0.5	99.04	94.82	Fit
0.5	0.1	98.94	93.78	Fit
0.5	5	99.04	94.18	Fit
1	1	98.94	93.71	Fit
1	5	98.97	91.31	Overfitting
5	5	98.12	91.45	Overfitting



- We got slightly better cross-validation set accuracies after using regularization. The difference is not much as some factors already didn't let our neural network overfit much. The contributing reasons to this are:

1. Stopping early: We did only 500 iterations while experimenting with the regularization parameters.
2. Fewer neurons in the hidden layer: We made the hidden layer with 50 neurons, which considering the size of the input layer (784) is small.

- For some lambda values, the accuracy was less than what we achieved without regularization. These cases are mostly over-fitted due to the large value of lambda, corresponding to lambda1 and lamba2 values to 5.

BEST REGULARISATION PARAMETER: 0.1, 0.5 with ACCURACY = **94.82%**

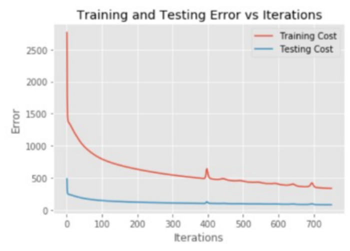
FOR OPTIMUM NEURONS IN THE HIDDEN LAYER:

We now analyse the model (from the training and the testing accuracy) as a function of number of neurones in the hidden layer. We started with 5 neurones in the hidden layer and went upto 2000.

We fixed the learning and regularisation parameters to the best ones achieved from cross validation. We now went for a different approach than cross validation, to see how the results depend on the hidden layers of the neural network.

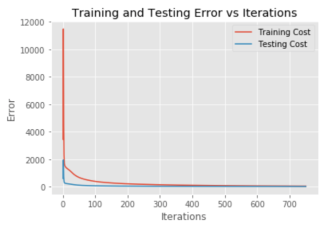
Our analysis has been presented below:

```
For Hidden layer size= 5
```



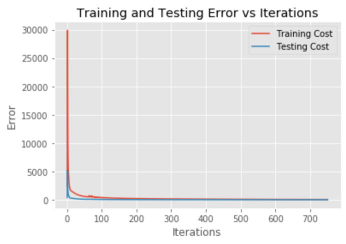
```
Train Accuracy:% 90.37647058823529
Test Accuracy:% 84.0
```

```
For Hidden layer size= 250
```



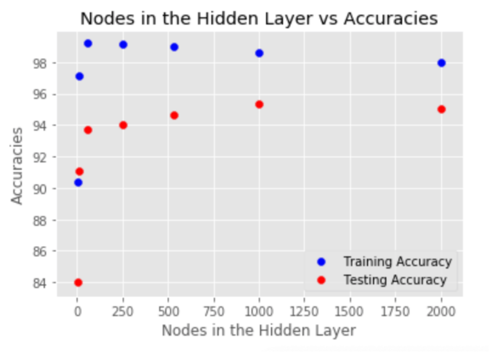
```
Train Accuracy:% 99.12941176470588
Test Accuracy:% 94.0
```

```
For Hidden layer size= 1000
```



```
Train Accuracy:% 98.61176470588235
Test Accuracy:% 95.33333333333334
```

Deductions:



Nodes in the Hidden Layer	Training Accuracy	Testing Accuracy
5	90.37	84
9	97.1294	91.0667
60	99.2	93.7333
250	99.1294	94
532	99.0118	94.6667
1000	98.6118	95.3333
2000	98.0235	95.0666

- The cross-validation set accuracy seems to increase with the increase in the number of neurons in the hidden layer. Though the accuracy increases, our model becomes more complex.
- The value for 5 neurons in the hidden layer is considerably lower than the other (84.0%) as such less number of neurons cause underfitting in our model.

- As the model got complex with the increase in the number of neurons in the hidden layer, we computed only till max 2000 neurons in the hidden layer. We achieved the maximum accuracy for **1000 neurons** with the test set accuracy of **95.33%**.
- The results were a little better for 1000 than 2000 neurons probably as the number of neurons are so high that the model is reading too many features from the given dataset. Since we require our models to generalize well, such a high count of neurons is rejected.
- When we tried computing for 5000 neurons the system started showing runtime and overflow errors owing to the complexity.
- Taking into account both complexity of the model and accuracy of the solution, we came to a conclusion that the model with 60 nodes in hidden layer is the best hypothesis. We can see that between **60** and 1000, the increase in number of neurons is way higher than that of change in accuracy (1.6%). Of course, if this change is relevant to the problem we develop on, we would prefer the higher accuracy model.

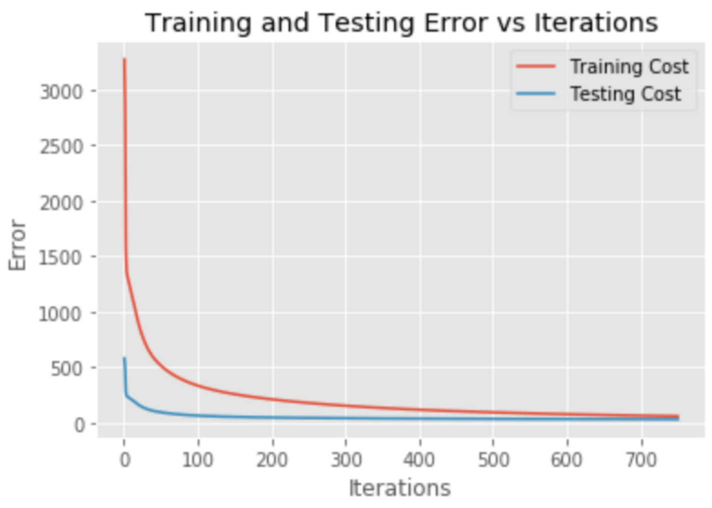
BEST HYPOTHESIS:

> Learning Rates: 1, 1

> Regularisation parameters: 0.1, 0.5

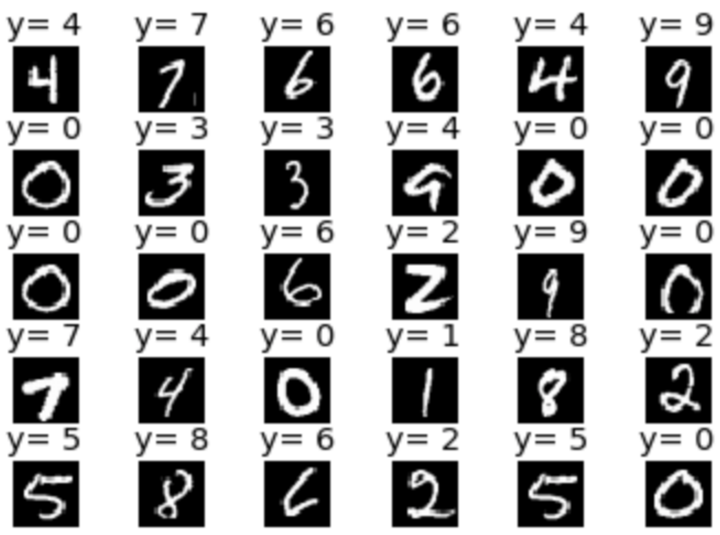
> No. of nuerons in the hidden layer: 60

```
For Hidden layer size= 60
```



```
Train Accuracy:% 99.2
Test Accuracy:% 93.73333333333333
```

Visualising 30 random samples from the test set with their predictions by the model written above them:



CONCLUSION:

This Assignment was a true test of ML theory, where we were deriving concepts learnt in class and revising them to make the most efficient neural network. Through this assignment, we were truly able to appreciate the effort it takes to ensure well functioning algorithms. Given the time taken to implement already existing algorithms, this further intrigues us about the work involved in developing something from scratch.