Machine Learning CSE 574 Programming Assignment 2

Handwritten Digits Classification

Group 55
Pratibha Arjun Barsale – 50247005
Venkata Krishnan Anantha Raman - 50246287

1. Overview

The assignment aims at implementing a Multilayer Perceptron Neural Network for evaluating its performance for classifying handwritten digits and face recognition. Also, the performance of this Single Neural Network is compared with the Deep Neural Network and Convolution Neural Network using the TensorFlow library.

2. Implementation

The Neural Network implemented is evaluated on the real data. The data is loaded from the **MNIST** dataset containing the 10 matrices for training set and testing data. The training data is further split into training and validation data. Thus, the experiments have been carried out on the three datasets namely, training, validation and testing.

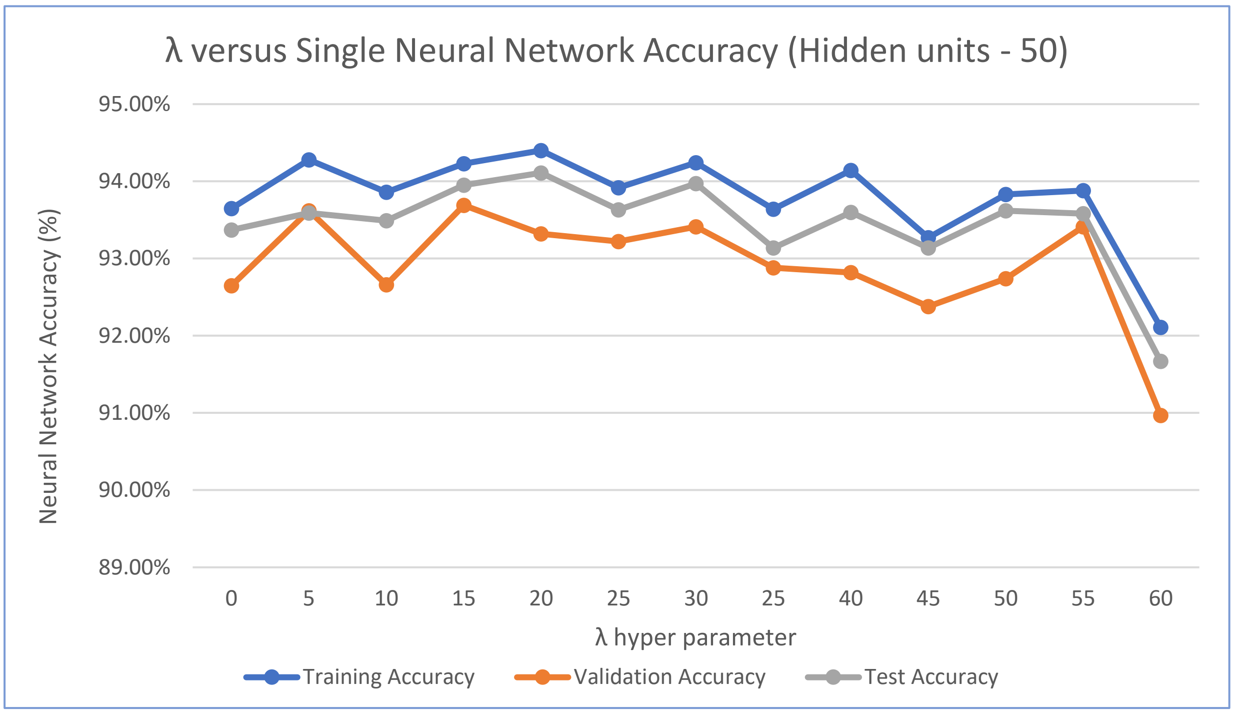
Feature selection is done on the input data. All features that don't carry any value are deleted. We are using the numpy.ptp function, which will give us the peak to peak value. A zero peek to peek value implies that the feature doesn't carry any value and can be deleted. A total of **719 features** were selected after this from the initial feature of **784**.

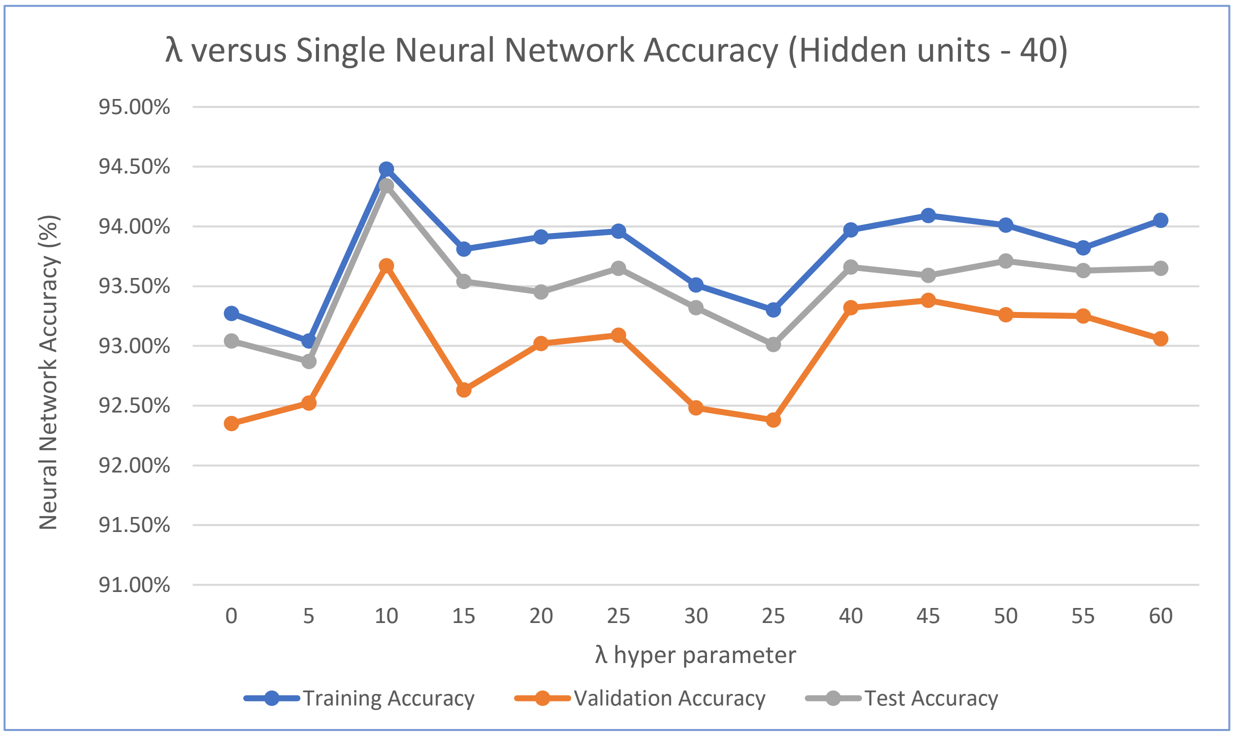
The Neural Network consists of three layers, the input layer, the hidden layer and the output layer. Feature selection is done. The two weight vectors namely w1 and w2 are assigned a random weight given the number of unit in the input and output layer. The data label is then predicted using the w1 and w2 weight vectors. Then the value of objective function and gradient are computed using the backpropagation algorithm.

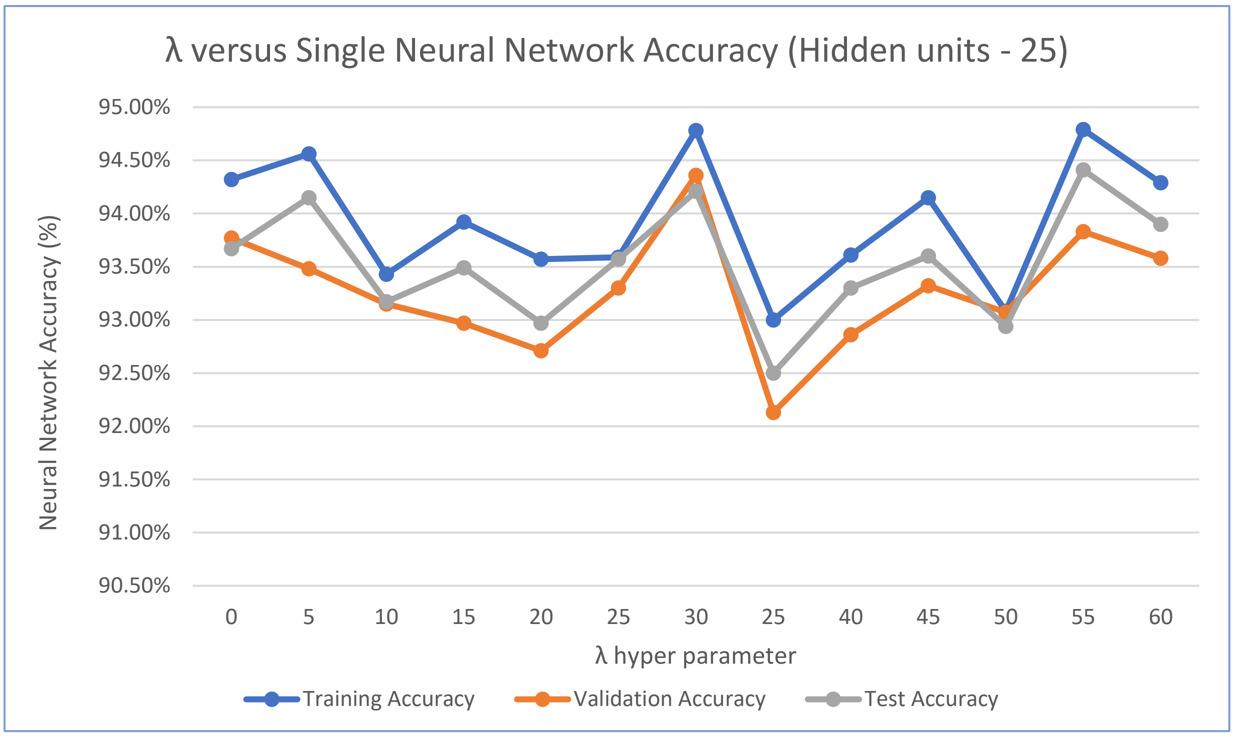
By setting different values to the regularization hyper-parameters of the network, namely lambda and number of hidden unit nodes, various comparisons have been made determining the accuracy and the runtime of the neural network for the three data sets.

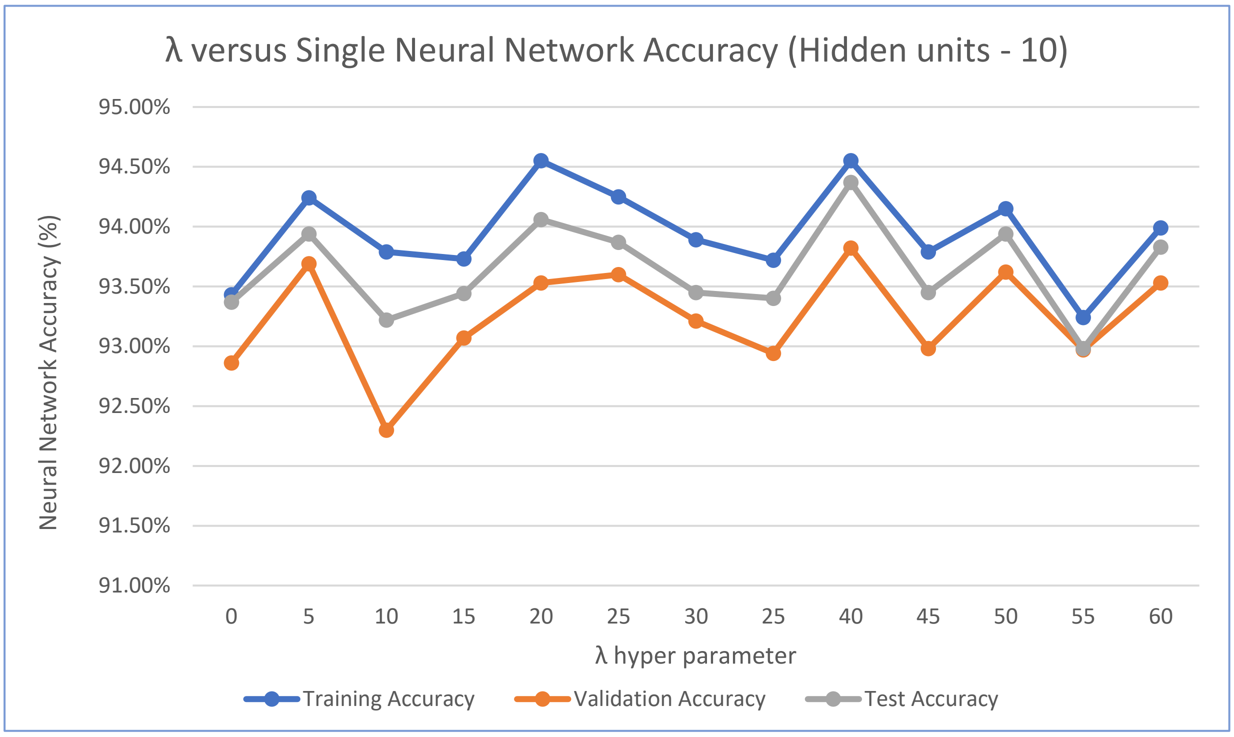
3. Choosing Hyper parameters:

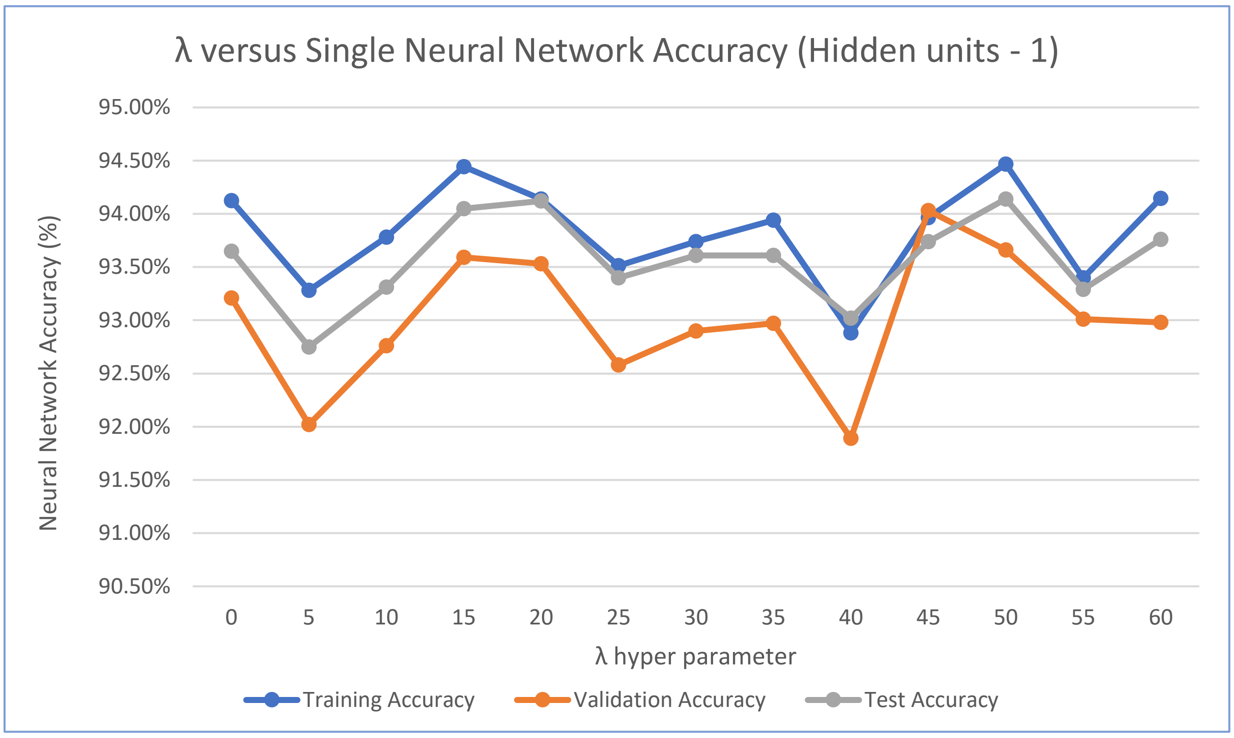
In this we choose λ from 0 to 60 for various hidden units ranging from 1 to 50. λ versus the accuracy for hidden units like 50,40,20,25,10,1 is obtained and plotted.

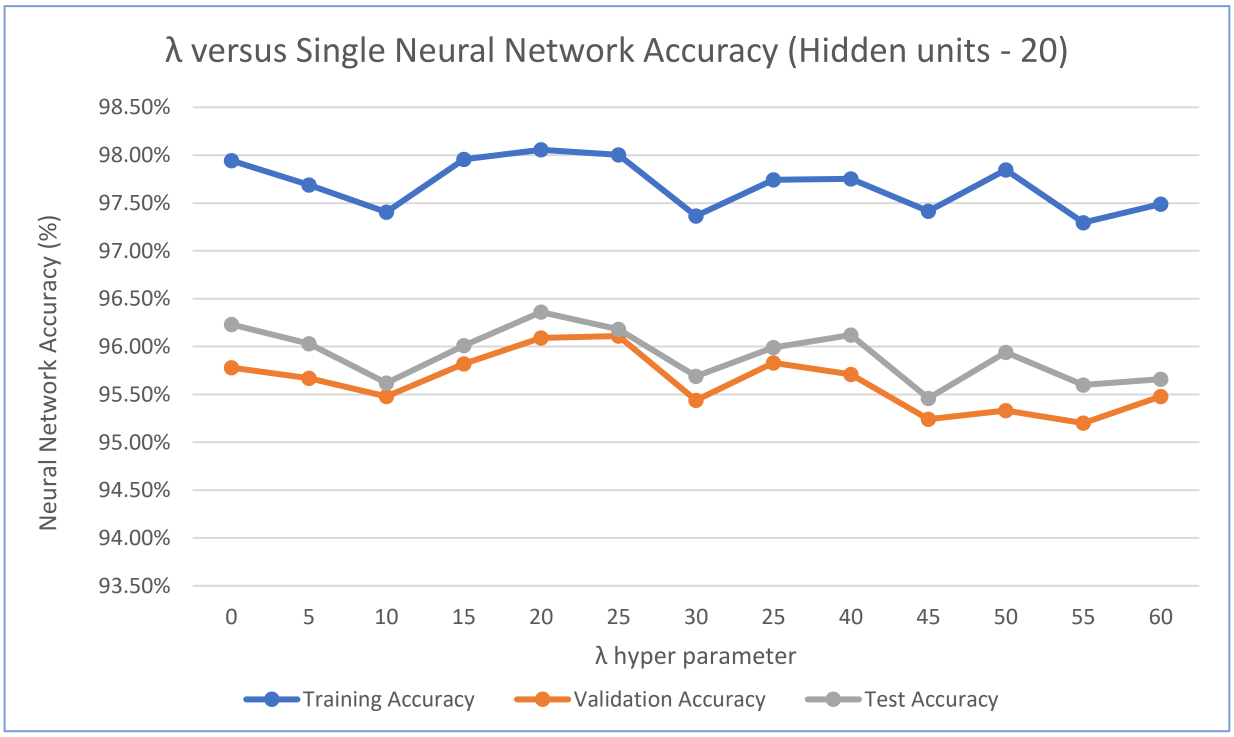





λ versus Single Neural Network Accuracy (Hidden units - 25)
Neural Network Accuracy (%)
95.00%
94.50%
94.00%
93.50%
93.00%
92.50%
92.00%
91.50%
91.00%
90.50%
0
5
10
15
20
25
30
25
40
45
50
55
60
λ hyper parameter
Training Accuracy
Validation Accuracy
Test Accuracy

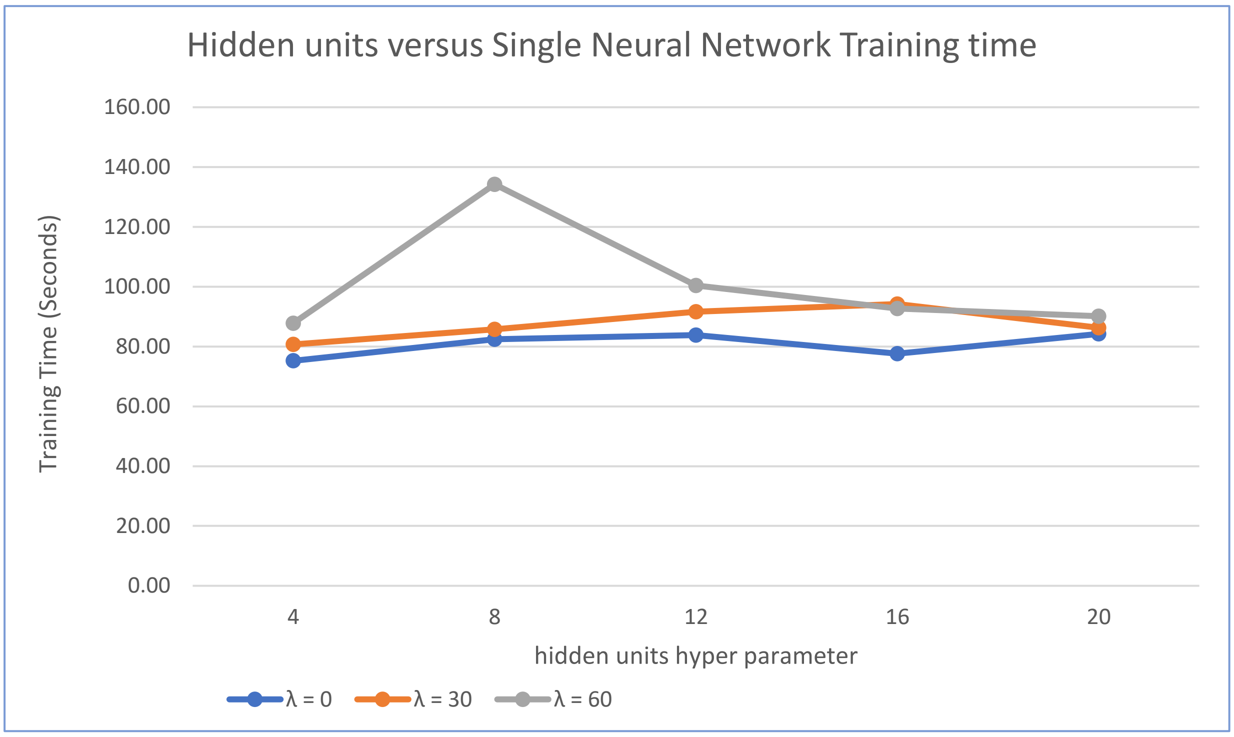

λ versus Single Neural Network Accuracy (Hidden units - 10)
Neural Network Accuracy (%)
95.00%
94.50%
94.00%
93.50%
93.00%
92.50%
92.00%
91.50%
91.00%
0
5
10
15
20
25
30
25
40
45
50
55
60
λ hyper parameter
Training Accuracy
Validation Accuracy
Test Accuracy


λ versus Single Neural Network Accuracy (Hidden units - 1)
Neural Network Accuracy (%)
95.00%
94.50%
94.00%
93.50%
93.00%
92.50%
92.00%
91.50%
91.00%
90.50%
0
5
10
15
20
25
30
35
40
45
50
55
60
λ hyper parameter
Training Accuracy
Validation Accuracy
Test Accuracy


λ versus Single Neural Network Accuracy (Hidden units - 20)
Neural Network Accuracy (%)
98.50%
98.00%
97.50%
97.00%
96.50%
96.00%
95.50%
95.00%
94.50%
94.00%
93.50%
0
5
10
15
20
25
30
25
40
45
50
55
60
λ hyper parameter
Training Accuracy
Validation Accuracy
Test Accuracy

Hidden Units vs Training Time:

The graph plots the Training time with the number of hidden nodes, for various lambda values.



We can infer from the above graph, as we increase the hidden units, the training time of the neural network increases. This is because more hidden units will result in more gradient and weights computations which will eventually result in more time.

Picking optimal Values of lambda and hidden units.

From the above observations it can be seen, choosing lambda as 30 with 25 hidden nodes will result in more accuracy and less training time. The training accuracy is 94.36% which is the highest among all the observations.

Number of Layer(Face and Deep NN) versus Accuracy on Celeb Dataset:

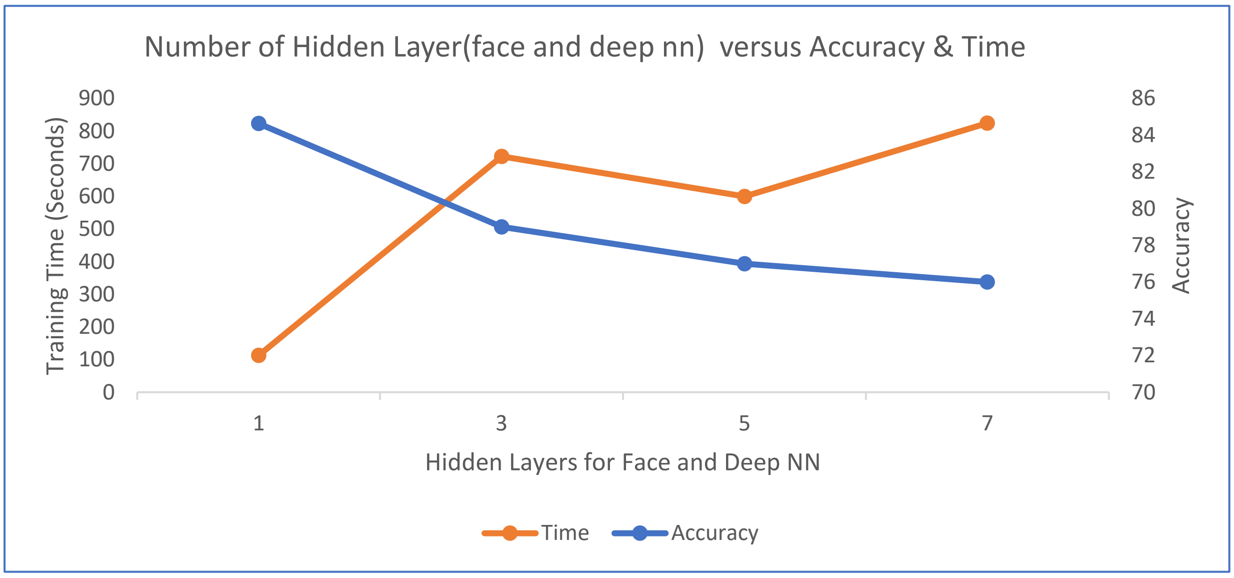
Single Layer Neural Network (lambda = 10, hidden units = 256)

Training Accuracy	:	84.57
Validation Accuracy	:	83.64
Test Accuracy	:	84.63
Time taken	:	112.63

Comparision of Single Layer Neural Network with Deep Neural Network:

Number of Layers	**Accuracy (%)**	**Training Time(seconds)**
1	84.63	112.63
3	79.14	721.89
5	77.25	599.86
7	76.08	824.16

The number of hidden layers versus Accuracy & Time for Celeb dataset:



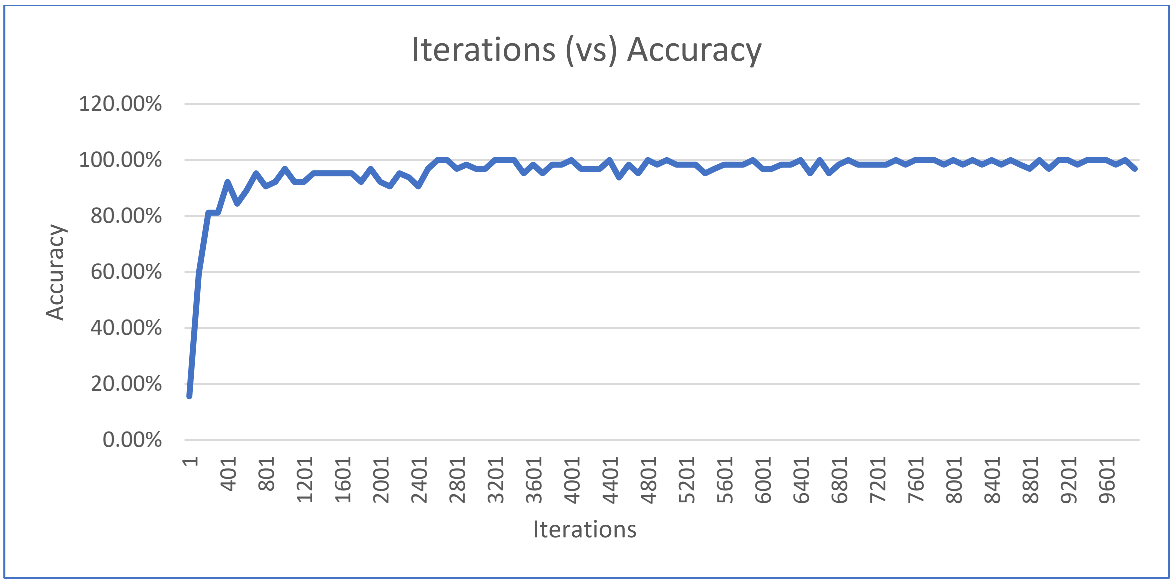
The above plot compares neural network with single layer (1 hidden layer) and deep neural network with 3,5,7 hidden layers. It could be seen that accuracy drops from 84.63 for single layer to 79%, 77% and 76% for 3,5,7 Hidden Layers. This is because with the addition of more layers we are overfitting the problem and this results in reduced testing accuracy.

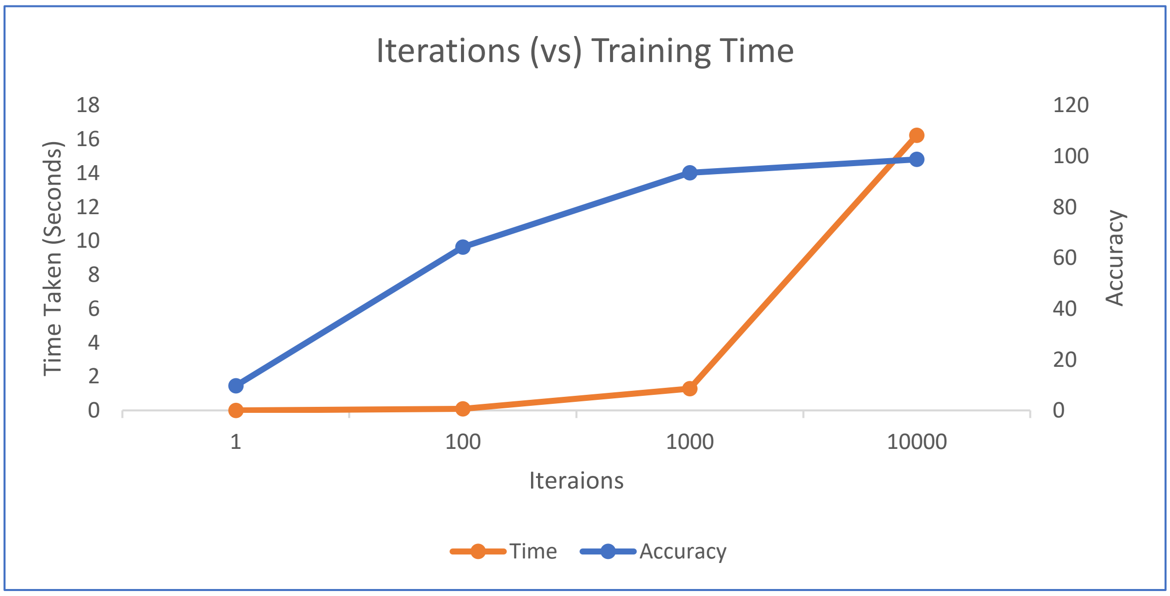
Usually increasing hidden nodes will try to fit the dataset better and result in better efficiency. But in this case with 1 hidden node the accuracy is better, increasing the hidden node results in overfitting and hence contributing to the drop in accuracy and increase in time taken.

3. *Analysis of Convolutional Neural Network:*

Iterations	**Accuracy**	**Time in sec**
1	9.7	0.10
100	64.2	0.10
1000	93.4	1.28
10000	98.7	16.22

The graph of iterations versus Accuracy & time for the Convolutional Neural Network are shown below:





The plot is done between number of hidden layers and the Time Taken & Accuracy. It could be seen that the Accuracy starts with 9.7% for 1 iteration, increases and reaches 98.7% with 10k Iterations. It can also be observed that the time taken increases with the increasing the number of iterations.

Confusion Matrix:

Confusion Matrix for Iteration: 1

```
[[    0     0     0   979     0     0     0     1     0     0]
 [    0     0     0  1134     0     0     0     1     0     0]
 [    0     0     0   972     0    56     0     4     0     0]
 [    0     0     0   997     0    12     0     1     0     0]
 [    0     0     0   976     0     5     0     1     0     0]
 [    0     0     0   880     0    10     0     2     0     0]
 [    0     0     0   942     0     0     0    16     0     0]
 [    0     0     0  1027     0     1     0     0     0     0]
 [    0     0     0   971     0     1     0     2     0     0]
 [    0     0     0  1006     0     3     0     0     0     0]]
```

Confusion Matrix for Iteration: 100

```
[[925   0   1  27   1   0   9   6  11   0]
 [  0 960   1  18   1   0   6   0 144   5]
 [ 23   3 815  77  23   0  24  28  37   2]
 [ 12   5  37 852   2   3   5  45  40   9]
 [  1   2   8   0 840   0  22   6  31  72]
 [ 28  21  21 242  64 355  21  62  55  23]
 [ 36  14  15   3  64   4 777   1  44   0]
 [  0  12  19   7  11   0   0 915  23  41]
 [ 14   6  19 130  24   6   7  43 703  22]
 [ 12   5  11  17 200   0   0 140  23 601]]
```

Confusion Matrix for Iteration: 1000

```
[[ 958     0     1     2     0     4     8     3     4     0]
 [    0 1122     1     3     1     0     3     0     5     0]
 [   13   17   902    28    19     1    12    20    18     2]
 [    2     8     7   939     0    19     0    16    11     8]
 [    1     3     3     0   937     0     9     1     2    26]
 [    6     3     0    27     8   817    18     2     7     4]
 [    7     7     1     2    22    14   903     0     2     0]
 [    0    13    22     7     6     0     0   937     2    41]
 [    6    11     3    27    14    16     7    11   859    20]
 [    8     8     2    11    39     7     1    15     3   915]]
```

Confusion Matrix for Iteration: 10000

```
[[ 975     0     1     0     0     0     0     1     3     0]
 [    0 1125     4     0     0     0     2     2     2     0]
 [    2     1  1022     0     1     0     0     1     5     0]
 [    1     0     0  1001     0     3     0     2     3     0]
 [    0     0     1     0   976     0     0     1     0     4]
 [    2     0     0     5     0   879     2     1     3     0]
 [   10     2     0     0     4     3   938     0     1     0]
 [    1     2     9     2     0     0     0  1010     1     3]
 [    5     0     2     1     1     1     0     2   960     2]
 [    5     3     2     2     7     3     0     3     1   983]]
```

Fig. Confusion Matrix

In the above confusion matrix, we could see how the matrix identifies the digits for different iterations. For iteration 1, the accuracy of identifying the data correctly is less. As the number of iterations increases, the data accuracy level also increases.

4. *Conclusion and Inferences:*

- **Feature selection** on input data is important because this will help us avoiding unnecessary computations.
- Too many hidden units may lead to the slow training phase while too few hidden units may cause the under-fitting problem. Therefore, we have chosen optimal hidden units (25) and optimal lambda (30) to avoid overfitting. These values are obtained from the handwritten number dataset. The code submission is set with these optimal values.
- We could see form the celeb dataset with single and multilayer neural network that with the increasing the hidden unit the accuracy decreases. This is a typical case of overfitting.
- Generally, with the increase in the hidden nodes the Accuracy should improve, stabilize and decrease if it is increased beyond a specific level.
- On a convolutional neural network, with the increasing the number of iterations we will get better accuracy.