# CSE574 Introduction to Machine Learning
# Programming Assignment 2
# **Handwritten Digits Classification**

Group 7

1. Karan Nisar (karankir)
2. Tejas Dhrangadharia (tejassha)

## Neural Network

The following is the summary of the observations from handwritten digits classification

Training set Accuracy: 94.254%

Validation set Accuracy: 93.52%

Test set Accuracy: 93.89%

Optimal Regularization Hyper-parameter – 20

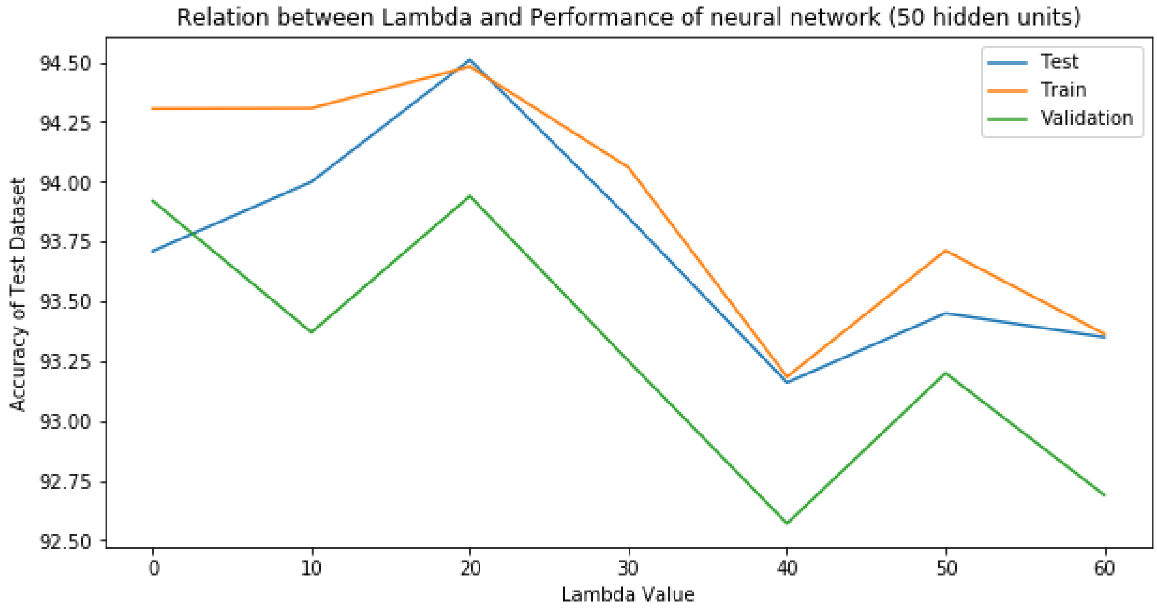
Optimal Number of nodes in hidden unit – 50

One major problem with Neural Networks is over-fitting. To avoid this we use regularization. The key here is to choose the optimal value of hyper-parameter to avoid both over-fitting and under-fitting. Hence, we iterate hyper-parameter lambda from 0 to 60 and calculate its effect in validation set accuracy.

On calculating prediction accuracy for different values of hyper-parameter, we obtain the following result

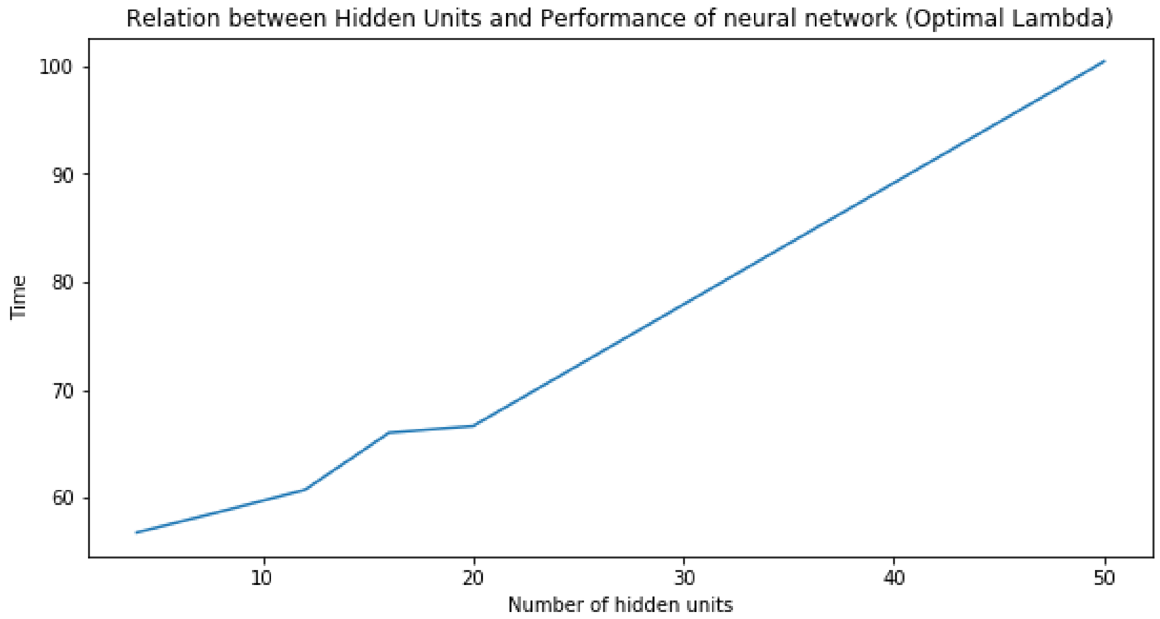
| Hyper-parameter values(lambda) | Test Set Accuracy | Validation Set Accuracy | Train Set Accuracy |
|---|---|---|---|
| **0** | 93.71 % | 93.92 % | 94.3 % |
| **10** | 94.0 % | 93.37 % | 94.3 % |
| **20** | 94.51 % | 93.94 % | 94.48 % |
| **30** | 93.85 % | 93.25 % | 94.06 % |
| **40** | 93.16 % | 92.57 % | 93.18 % |
| **50** | 93.45 % | 93.2 % | 93.71 % |
| **60** | 93.35 % | 92.69 % | 93.36 % |

The following plot is obtained for the above table



With increase in the value of hyper-parameter, the accuracy decreases because of under-fitting. This happens because high values give more importance to the weight than the errors that makes it difficult to converge. Similarly, lower value of hyper-parameter results in over-fitting. So, a value which is not too small and not too large can avoid both these problems. Also, we can see in the figure that lower values result in over-fitting resulting in higher train accuracy. For lambda value 20 we can see that the values converge. Hence 20 will be the optimal hyper-parameter for this Neural Network.

Another factor affecting the performance of Neural Network is the number of hidden units. Higher the number of hidden units, higher is the accuracy, but the time taken also increases. Hence, we need to choose an optimal value of hidden unit that maximizes the performance and also does not take too long for execution. The relation of number of hidden units with performance and time can be understood from the following graph.



## CelebA Dataset

The following is the classification accuracy on CelebA dataset

Training set Accuracy: 85.5260663507%

Validation set Accuracy: 84.5403377111%

Test set Accuracy: 85.3141559425%

## Neural Network vs Deep NN

The following is the observation of the Deep neural network

| Layer | learning_rate | Accuracy | Time |
|---|---|---|---|
| **3** | 0.001 | 0.809614 | 2141.822 |
| **5** | 0.001 | 0.840432 | 2276.718 |
| **7** | 0.001 | 0.837464 | 2597.307 |

We can observe that we get a higher accuracy for 5 hidden layers, so we will use 5 hidden layers.

| Evaluation on Test Data | Accuracy | Time |
|---|---|---|
| **Single NN** | 93.89 % | 100.42 |
| **Deep NN** | 84.04 % | 2276.718 |

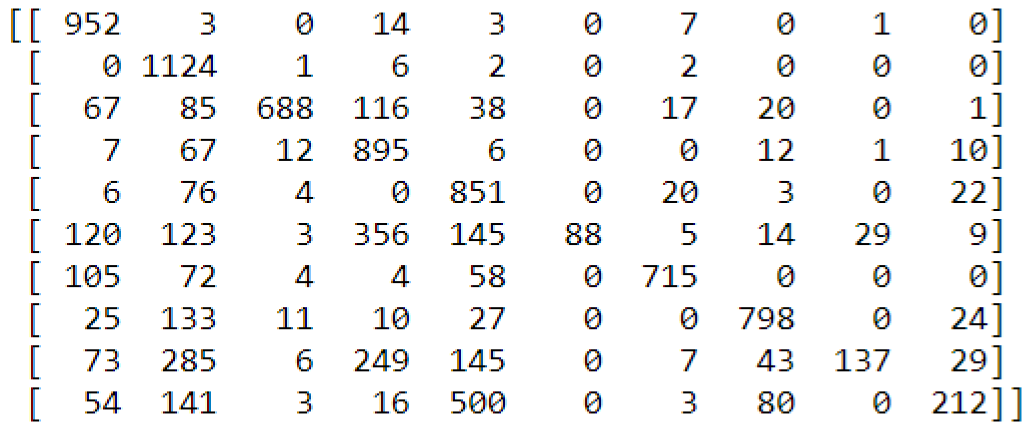
Here neural network outperforms Deep Neural network in terms of both time and accuracy.

## Convolution Neural Network Output

Following is the output of CNN for three different iterations viz 99, 900 and 9000.

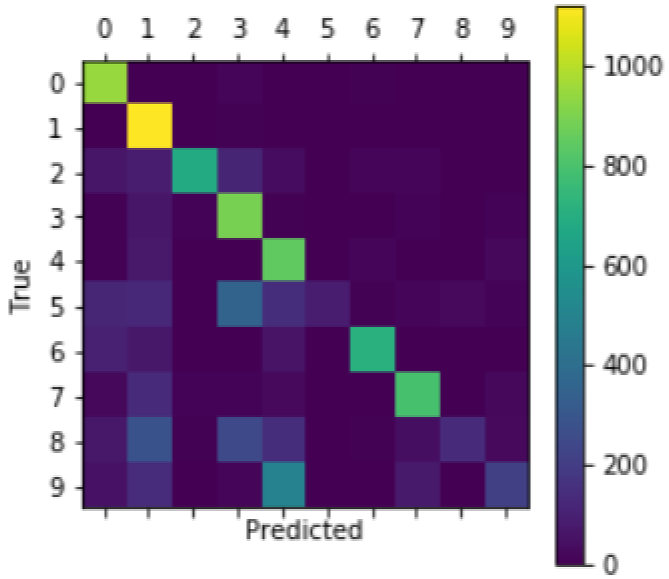
**CNN (99 iterations)**

Confusion matrix:



```
[[ 952    3    0   14    3    0    7    0    1    0]
 [   0 1124    1    6    2    0    2    0    0    0]
 [  67   85  688  116   38    0   17   20    0    1]
 [   7   67   12  895    6    0    0   12    1   10]
 [   6   76    4    0  851    0   20    3    0   22]
 [ 120  123    3  356  145   88    5   14   29    9]
 [ 105   72    4    4   58    0  715    0    0    0]
 [  25  133   11   10   27    0    0  798    0   24]
 [  73  285    6  249  145    0    7   43  137   29]
 [  54  141    3   16  500    0    3   80    0  212]]
```

Plot:

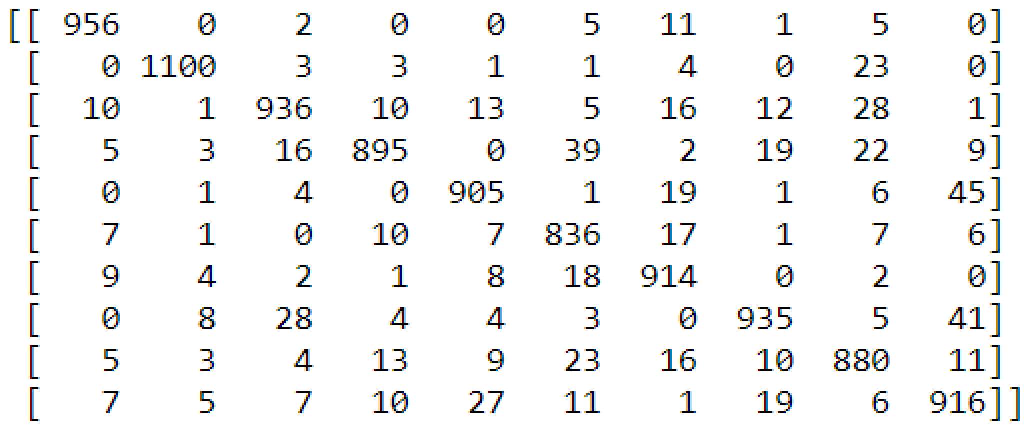


Time usage: 0:00:57 (57 sec)

Accuracy on Test-Set: 64.6% (6460 / 10000)

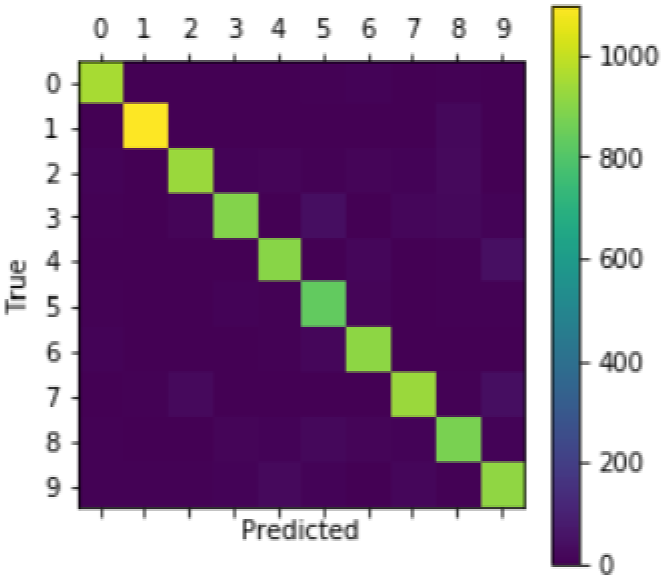
**CNN (900 iterations)**

Confusion Matrix:



```
[[ 956    0    2    0    0    5   11    1    5    0]
 [   0 1100    3    3    1    1    4    0   23    0]
 [  10    1  936   10   13    5   16   12   28    1]
 [   5    3   16  895    0   39    2   19   22    9]
 [   0    1    4    0  905    1   19    1    6   45]
 [   7    1    0   10    7  836   17    1    7    6]
 [   9    4    2    1    8   18  914    0    2    0]
 [   0    8   28    4    4    3    0  935    5   41]
 [   5    3    4   13    9   23   16   10  880   11]
 [   7    5    7   10   27   11    1   19    6  916]]
```

Plot:

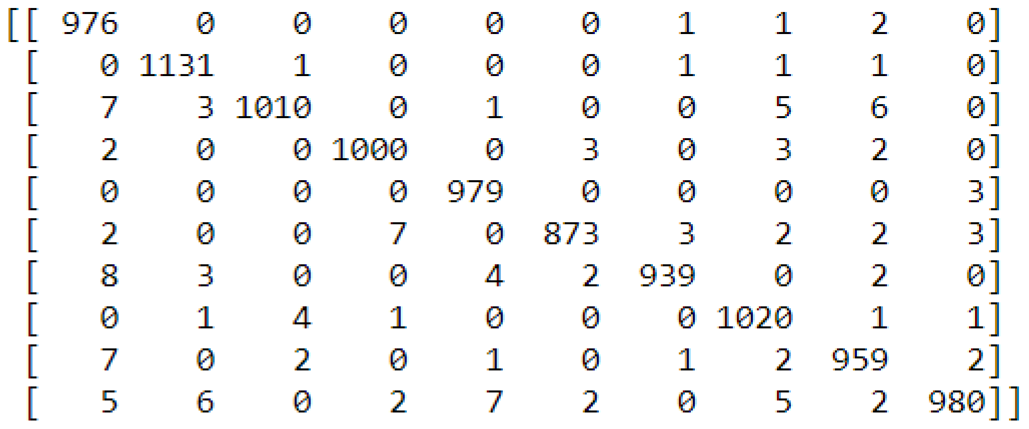


Time usage: 0:09:19 (559 sec)

Accuracy on Test-Set: 92.7% (9273 / 10000)

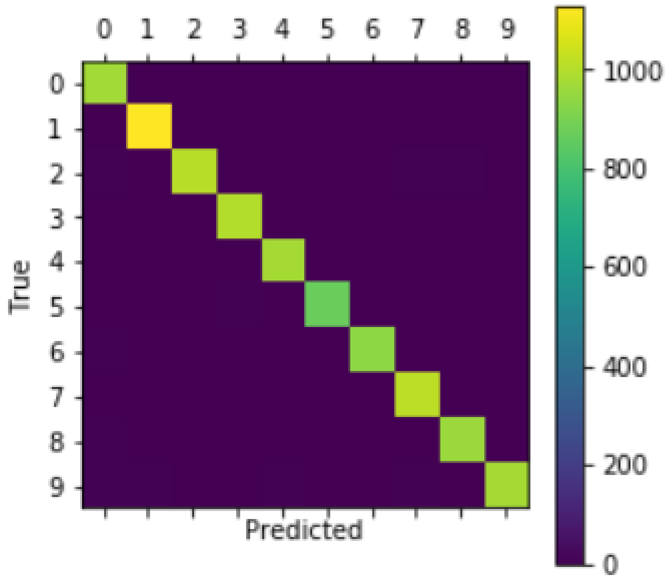
**CNN (9000 iterations)**

Confusion matrix:



```
[[ 976    0    0    0    0    0    1    1    2    0]
 [   0 1131    1    0    0    0    1    1    1    0]
 [   7    3 1010    0    1    0    0    5    6    0]
 [   2    0    0 1000    0    3    0    3    2    0]
 [   0    0    0    0  979    0    0    0    0    3]
 [   2    0    0    7    0  873    3    2    2    3]
 [   8    3    0    0    4    2  939    0    2    0]
 [   0    1    4    1    0    0    0 1020    1    1]
 [   7    0    2    0    1    0    1    2  959    2]
 [   5    6    0    2    7    2    0    5    2  980]]
```

Plot:



Time usage: 0:59:45 (1815 sec)

Accuracy on Test-Set: 98.7% (9867 / 10000)

We can see that the Accuracy is directly related to the number of iterations. The accuracy shoots from 64.6% to 98.7%. This can be seen clearly from the plots of confusion matrix. However, the time taken also increases considerably.