CSE 574 – Introduction to Machine Learning

Programming Assignment 1

Handwritten Digits Classification

CSE 574 - Group 18

➢ Maheedhara Sastry Achalla
➢ Akash Yelaswarupu
➢ Amaan Akhtarali Modak

HANDWRITTEN DIGITS CLASSIFICATION

INTRODUCTION

In this assignment, our task was to implement a Multilayer Perceptron Neural Network and evaluate its performance in classifying handwritten digits. After implementing the feedforward pass with randomly initialized weights, we have normalized the error function by a factor λ and used it for back propagation calculations. Performance of the neural network for varying λ and number of hidden nodes have been observed to check for optimal behavior.

CHOOSING THE HYPER-PARAMETERS FOR NEURAL NETWORK

In order to implement our neural network, we needed to fine-tune the hyper-parameters involved, which were the number of units in the hidden layer and λ (lambda).

To select the optimal value for lambda $\boldsymbol{\lambda}$

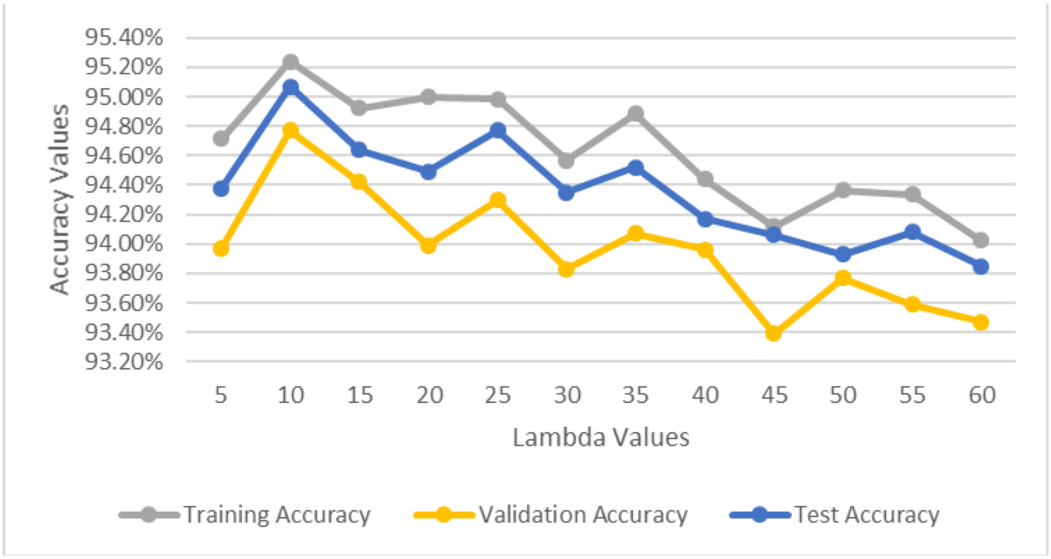
Firstly, we consider selecting the optimal value of lambda, λ, by calculating the accuracy of our implementation for each varying value of λ, and by keeping the number of hidden nodes at a predefined value.

The table below gives us the values that we obtained when keeping the number of hidden nodes constant and changing the values for λ, and measuring the overall accuracy of our implementation.

λ	Hidden Nodes	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
0	50	95.41	94.76	94.95
5	50	94.716	93.97	94.38
10	50	95.24	94.77	95.07
11	50	94.894	94.59	94.48
12	50	95.188	94.82	94.91
13	50	94.734	94.31	94.48
15	50	94.922	94.42	94.64
20	50	94.996	93.99	94.49
25	50	94.986	94.3	94.77
30	50	94.57	93.83	94.35
35	50	94.886	94.07	94.52
40	50	94.438	93.96	94.17
45	50	94.12	93.39	94.06
50	50	94.364	93.77	93.93

55	50	94.334	93.59	94.08
60	50	94.022	93.47	93.85

The below graph shows the variation in accuracy of our implementation with variation in λ in the range 0 to 60 in steps of 5. Ideally we expect increase in accuracy of testing data and decrease in accuracy of training data with increasing lambda value. However, this is not observed in our implementation. This might be because there are not enough training samples.



To select the optimal value for the number of hidden nodes

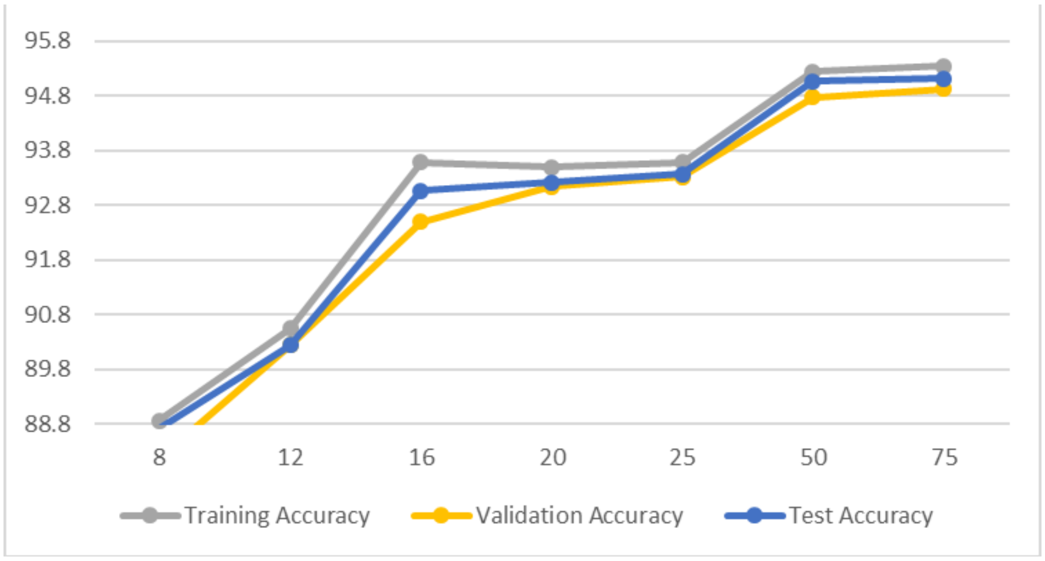
Now, we consider selecting the optimal number of units hidden, by calculating the accuracy of our implementation for each varying value of the number of hidden nodes, and by keeping the value of λ at a predefined value, that is 50.

The table below gives us the values that we obtained when changing the number of hidden nodes and keeping the value for λ constant, and measuring the overall accuracy of our implementation.

λ	Hidden Nodes	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Time (s)
10	4	58.6	57.06	58.47	139.39
10	8	88.868	88.17	88.71	148.25
10	12	90.55	90.25	90.25	157.25
10	16	93.59	92.5	93.07	165.75
10	20	93.5	93.14	93.22	167.1

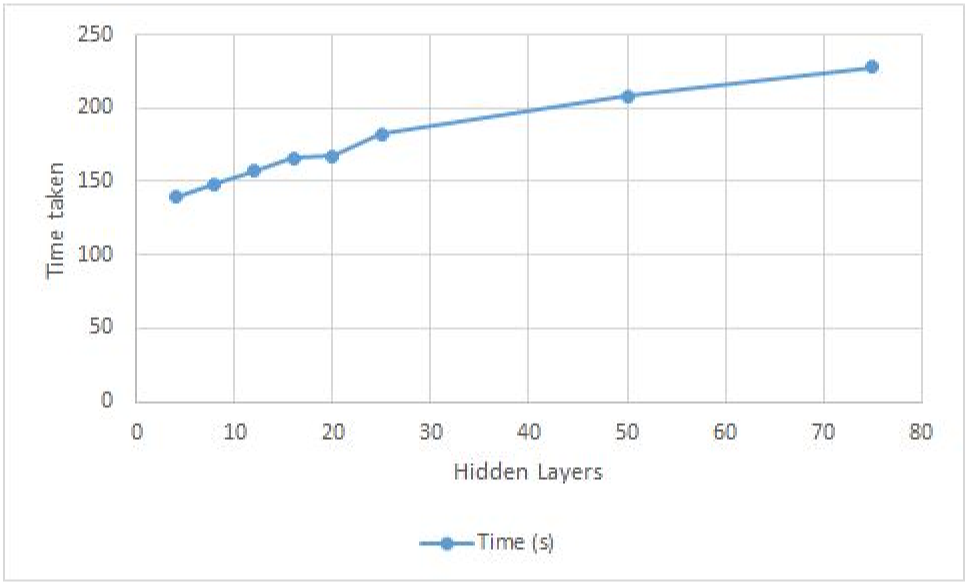
10	25	93.59	93.32	93.37	182.34
10	50	95.24	94.77	95.07	208.34
10	75	95.338	94.92	95.12	228.05

The below graph shows the variation in accuracy of our implementation with variation in number of hidden nodes in the range 0 to 75.



The accuracy of our implementation increases as the number of hidden nodes increases. This can be attributed to the fact that the hidden nodes represent the number of learned features. We also see that after a certain value of hidden nodes, the accuracy does not increase anymore. Beyond this point, accuracy cannot increase as the number of features to learn is a fixed value.

To compare the number of hidden nodes with runtime



We see here that as we increase the number of hidden nodes, the learning time of the neural network increases. The number of weight updations increase with each additional node, and this value is being computed several times before arriving at optimal weights. Hence, more hidden units imply more complexity and therefore more time.

<u>BEST OBSERVED OUTPUT OF OUR IMPLEMENTATION</u>

Based on observation we obtain the following values:

λ = 10

Number of hidden nodes = 50
Time taken for computation = 153 seconds
Training Set Accuracy = 95.24%
Validation Set Accuracy = 94.77%
Test Set Accuracy = 95.07%

<u>CONCLUSION</u>

Based on the given programming assignment we were able to work on the concepts of how a neural networks, the feed forward and back propagation methods to implement a neural network. We were able to perform real life machine learning operations on the given dataset, and generate the desired accuracy levels by tweaking certain parameters of our proposed neural network. With lesser nodes in the hidden layers, it seems like we have underfitting problem. As we increase the nodes, the accuracy improves. We conclude that the hyper-parameter values that we have considered are optimal for our implementation of the assignment.

DEEP NEURAL NETWORKS

INTRODUCTION

For this part, we need to test the accuracy of the celeb dataset by using the single layererd neural network we built for the MNIST dataset. The basic idea here is to run deep neural network code provided in the assignment and compare the results with the single layered neural network we developed. The building deep neural network is developed using tensorflow library.
Training set accuracy on the CelebA dataset using the single layered neural network code developed for MNIST data is **85.3886255924%**
Validation set accuracy on the CelebA dataset using the single layered neural network code developed for MNIST data is **83.8273921201%**
Test set accuracy on the CelebA dataset using the single layered neural network code developed for MNIST data is **85.8062074186%**

EVALUATING THE ACCURACY OF DEEP NEURAL NETWORK ON CELEBA DATASET

We were provided the code for deep neural network on CelebA dataset with two hidden layers. We need to simply run this code and check the accuracy and time taken for running this code.

Accuracy on the CelebA dataset using the deep neural network code provided is **80.999%**
Time taken is **292.0259 seconds**

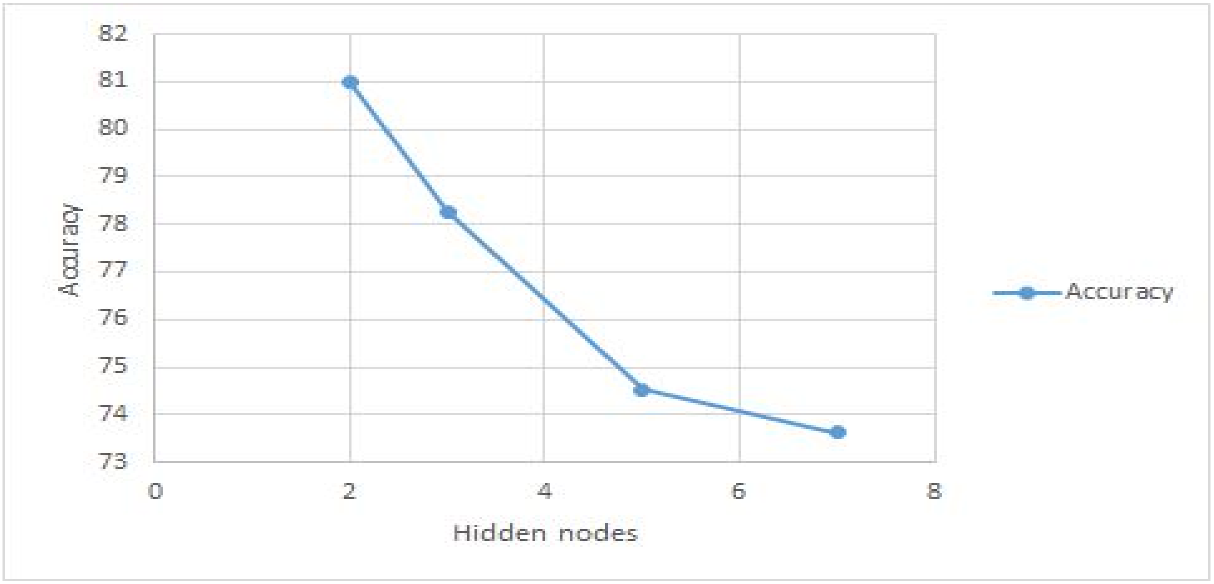
COMPARING SINGLE VS DEEP NEURAL NETWORKS

Now we need to calculate the accuracy of deep neural network by adding the hidden layers. The expectation is that the accuracy increases with increased number of hidden layers. The accuracy and the time taken for different number of hidden layers has been calculated and formulated below:

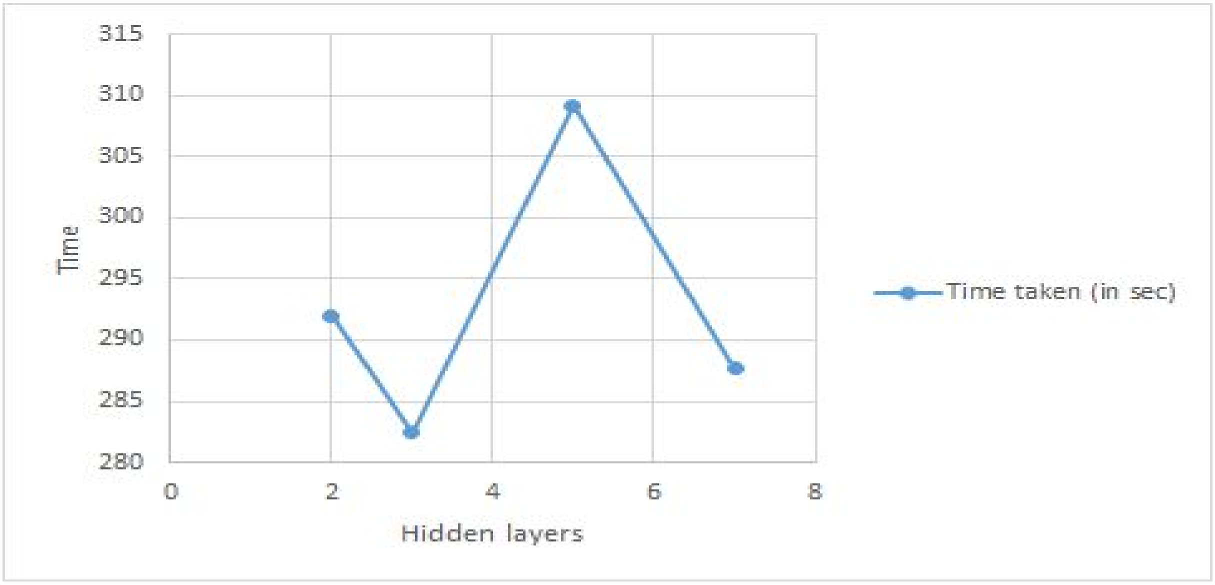
Hidden Nodes	**Accuracy (in percentage)**	**Time taken (in sec)**
2	80.999	292.0259
3	78.274	282.4274

5	74.526	309.1480
7	73.6185	287.6439

The below graph shows the variation of accuracy with increasing hidden nodes



The below graph shows the variation of time taken for running the code for different number of hidden layers



Here, it's observed that the accuracy goes down as the number of hidden layers increases, which is contrary to what we have expected. Also as the number of hidden layers increases, the time taken for running the code fluctuates instead of increasing at constant pace. This might be due to overfitting.

<u>CONCLUSION:</u>

Upon observing the variations in the accuracy for different hidden nodes in deep neural network vs single neural network, we can assume that efficiently implemented single layered neural network tends to perform better than multilayered neural network which is not efficiently implemented. A lot of factors are to be considered while choosing proper neural network for your dataset. The time taken for running the code also plays a major role in the selection criterion. We need to ensure that a poorly performed neural network is not considered for insignificant improvement in accuracy.
For the given CelebA dataset, the single layered neural network works efficiently than the given multilayered neural network with increases hidden nodes.