CSE 574 - Introduction to Machine Learning

Programming Assignment 2

Handwritten Digits Classification

Sai krishna kanneti
Yashas Shamraju

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 backpropagation calculations. Performance of the neural network for varying λ and number of hidden nodes have been observed to check for optimal behavior.

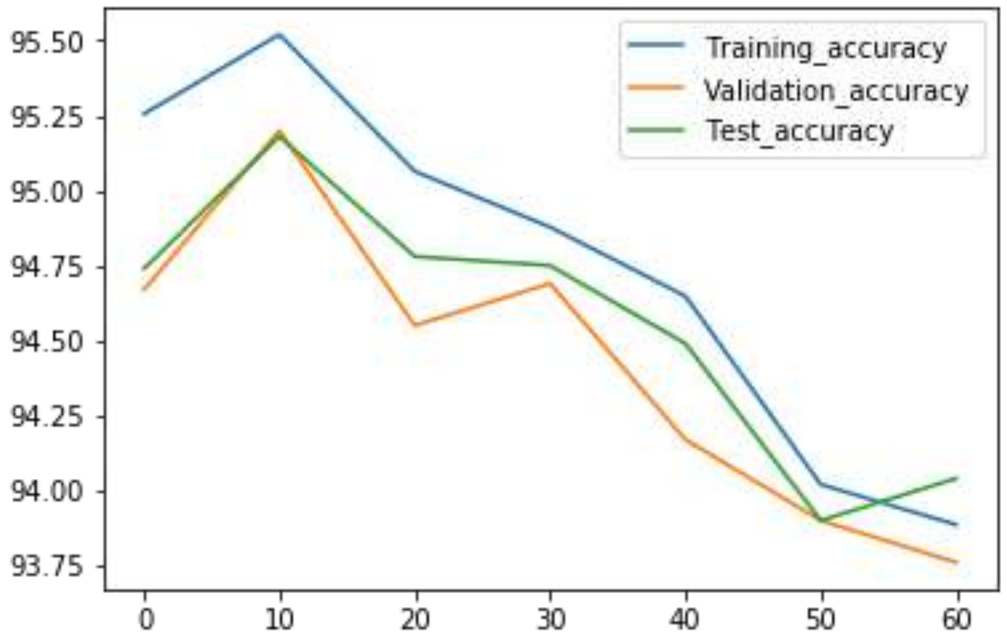
CHOOSING THE HYPER-PARAMETERS FOR NEURAL NETWORK

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 λ

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 graph below shows the accuracy obtained for different values of λ. when keeping the number of hidden nodes constant and changing the values for λ, and measuring the overall accuracy of our implementation.

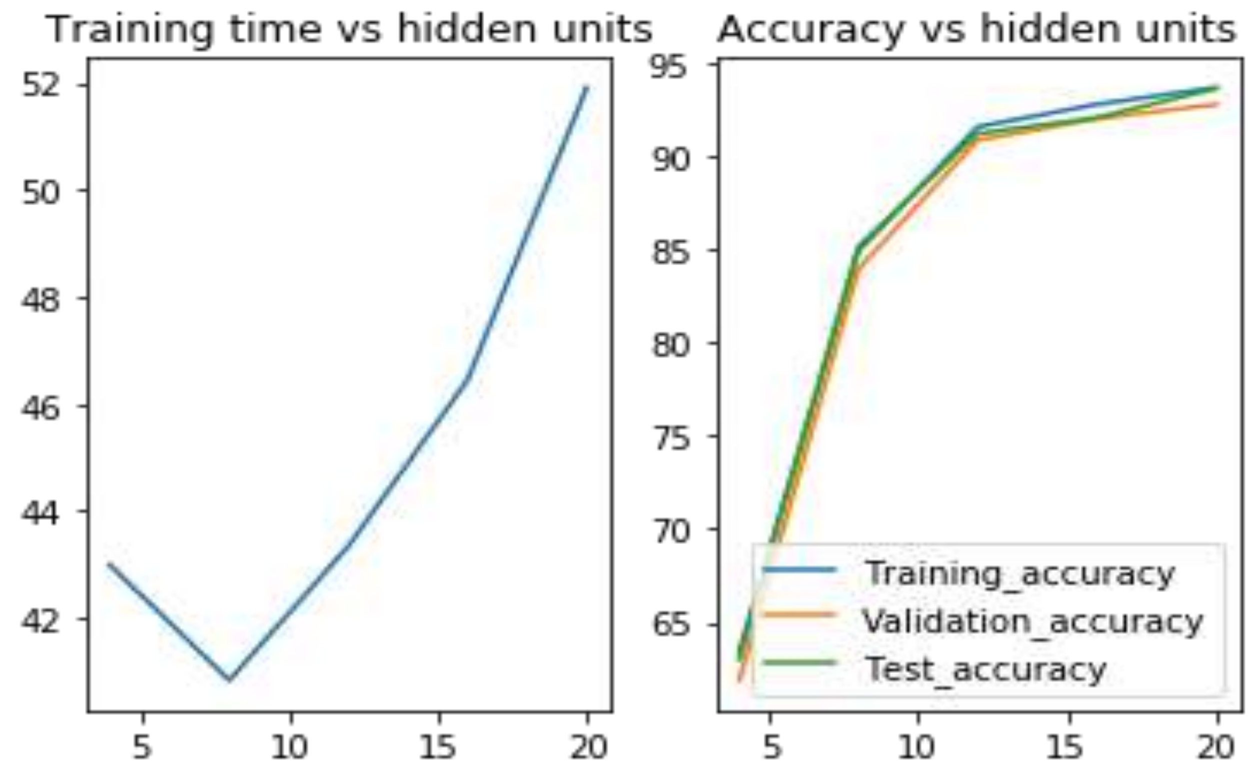


The above graph shows the variation in accuracy of our implementation with variation in λ in the range 0 to 60 in steps of 10. Ideally we expect increase in accuracy of testing data and decrease in accuracy of training data with increasing lambda value. However, this is not observed in our implementation. This might be because of sampling techniques used in training samples or over-fitting problem. In particular, the similarity between training and test data may cause the performance on the test set not to be affected by overfitting on the training set, and vice versa. The optimal value of lambda is observed at: 10.

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 10.

The graph below shows us the values that we observed for accuracy and training time by changing the number of hidden nodes and keeping the value for λ constant.



The below graph shows the variation in accuracy of our implementation with variation in number of hidden nodes in the range of [4,8,12,16,20].

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 is not getting increased.

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 updation increases 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:

$\lambda = 10$

Number of hidden nodes = 20
Time taken for computation = 51 seconds
Training Set Accuracy = 93.65%
Validation Set Accuracy = 92.76%
Test Set Accuracy = 93.61%

CONCLUSION

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. As we increase the nodes, the accuracy improves, yet there is a trade-off between the training time vs accuracy as we increase the number of nodes, the training time also increases. We conclude that the hyper-parameter values that we have considered are optimal for our implementation of the assignment. The optimal value of lambda is considered as: 10 as the further increase in lambda does overfitting of the model

DEEP NEURAL NETWORKS

INTRODUCTION

For this part, we need to test the accuracy of the celeb dataset by using the single layered neural network we built for the MNIST dataset. The essence here is to run deep neural network code provided in the assignment and compare the results with the single layered neural network. The 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 **85.2701421801%**
Validation set accuracy on the CelebA dataset using the single layered neural network code developed for MNIST data is **84.277673546%**
Test set accuracy on the CelebA dataset using the single layered neural network code developed for MNIST data is **85.352006056%**

EVALUATING THE ACCURACY OF DEEP NEURAL NETWORK ON CELEBA DATASET

The code for deep neural network on CelebA dataset with 1 hidden layers. After making few code changes to run this code for 3,5,7 hidden layers, the accuracy and time are taken for this code.

Accuracy on the CelebA dataset using the deep neural network code provided is **80.658%**
Time taken is **296.7259 seconds**

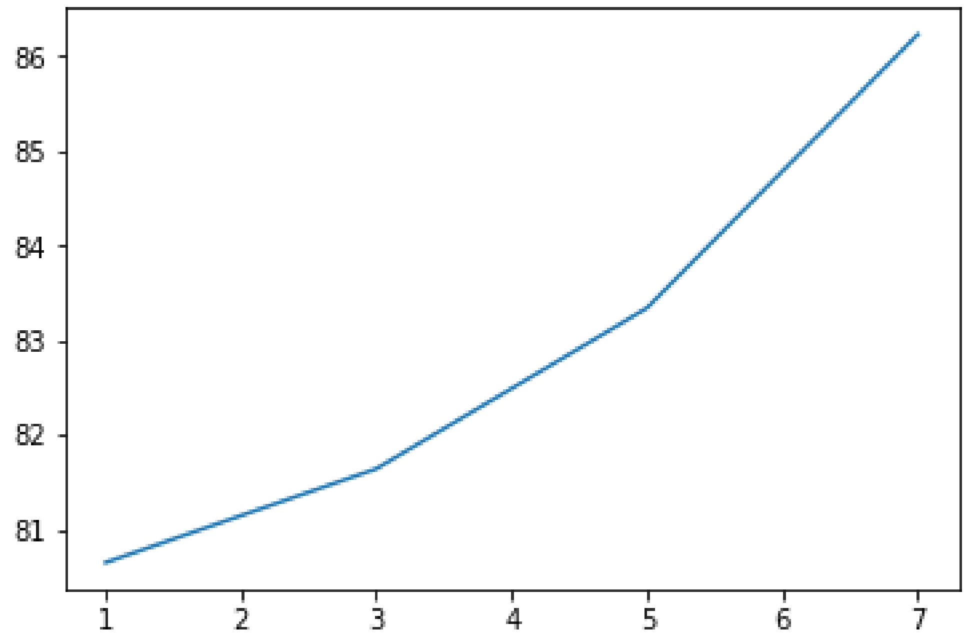
COMPARING SINGLE VS DEEP NEURAL NETWORKS

Calculated 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:

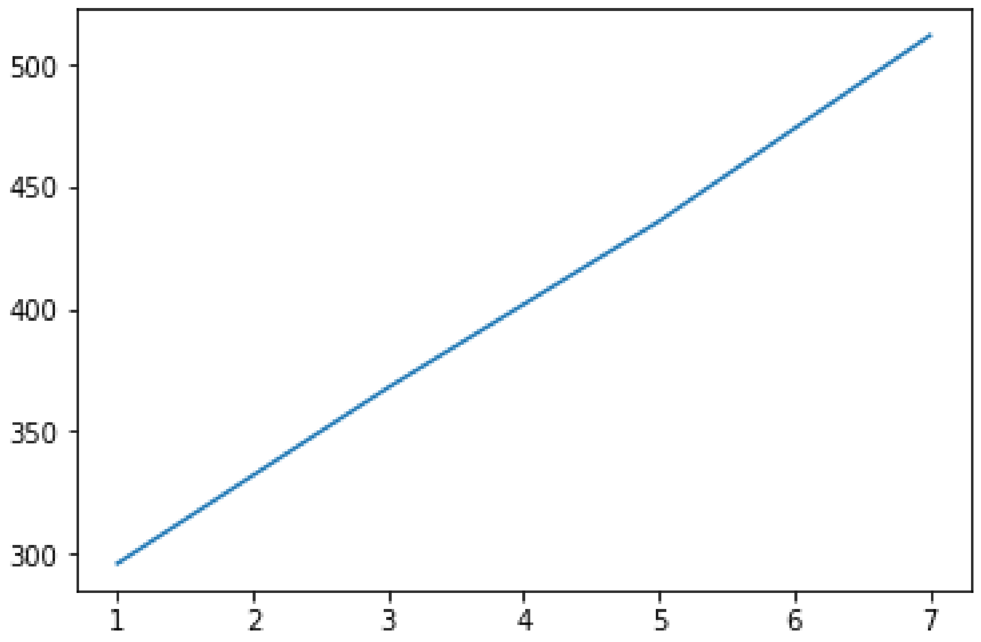
Hidden Layers	Accuracy (in percentage)	Time taken (in sec)
1	80.658	296.0259
3	81.6427	368.756

5	83.346	436.1480
7	86.6185	512.6439

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

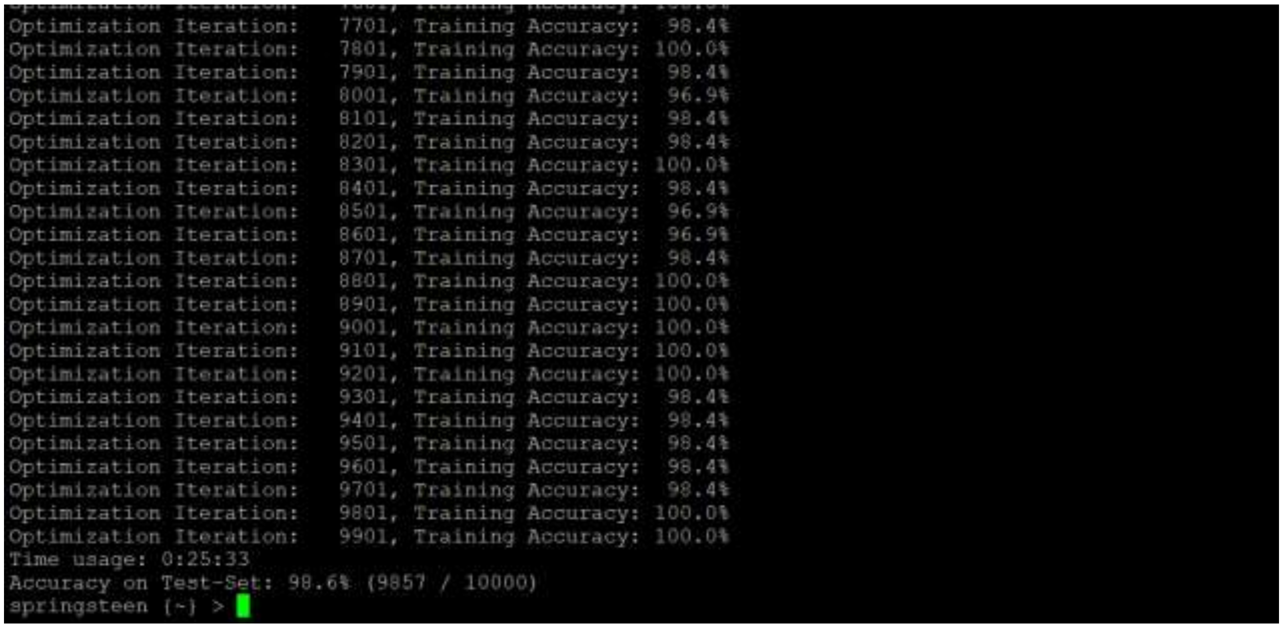


Here, it's observed that the accuracy increases as the number of hidden layers increases. Also as the number of hidden layers increases, the time taken for running the code also increases at constant pace.

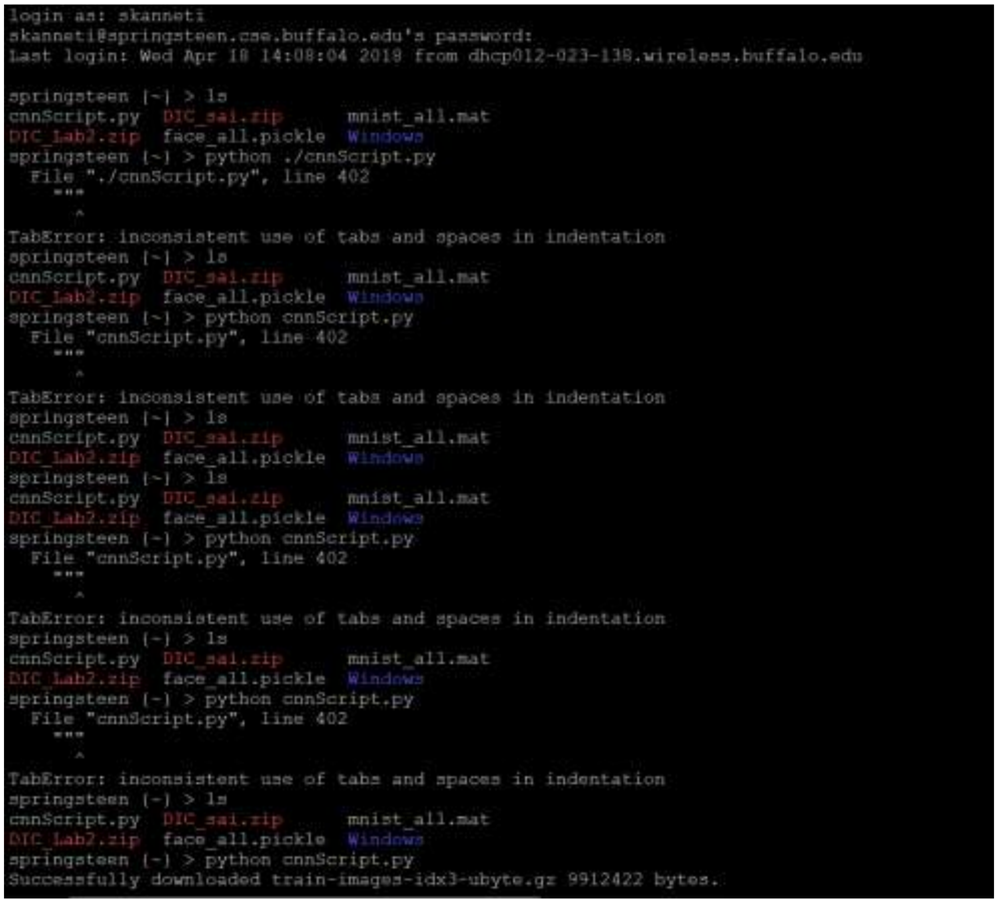
CONCLUSION:

Upon observing the variations in the accuracy for different hidden layers 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. The time taken for running the code also played a significant role in the selection criterion.
For the given CelebA dataset, the single layered neural network works efficiently than the given multilayered neural network with increasing hidden layers.

RESULTS OF CONVOLUTIONAL NEURAL NETWORK IN TERMS OF ACCURACY& TRAINING TIME:

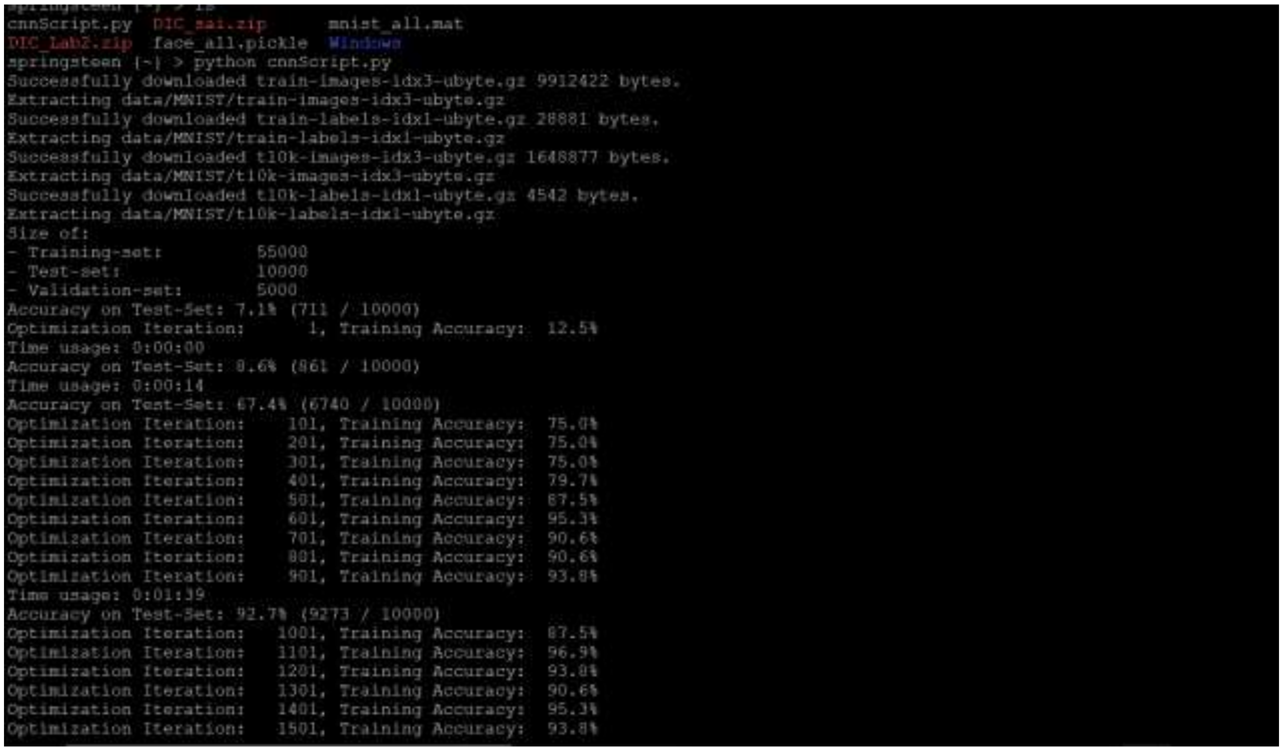


```
Optimization Iteration:   7701, Training Accuracy:  98.4%
Optimization Iteration:   7801, Training Accuracy: 100.0%
Optimization Iteration:   7901, Training Accuracy:  98.4%
Optimization Iteration:   8001, Training Accuracy:  96.9%
Optimization Iteration:   8101, Training Accuracy:  98.4%
Optimization Iteration:   8201, Training Accuracy:  98.4%
Optimization Iteration:   8301, Training Accuracy: 100.0%
Optimization Iteration:   8401, Training Accuracy:  98.4%
Optimization Iteration:   8501, Training Accuracy:  96.9%
Optimization Iteration:   8601, Training Accuracy:  96.9%
Optimization Iteration:   8701, Training Accuracy:  98.4%
Optimization Iteration:   8801, Training Accuracy: 100.0%
Optimization Iteration:   8901, Training Accuracy: 100.0%
Optimization Iteration:   9001, Training Accuracy: 100.0%
Optimization Iteration:   9101, Training Accuracy: 100.0%
Optimization Iteration:   9201, Training Accuracy: 100.0%
Optimization Iteration:   9301, Training Accuracy:  98.4%
Optimization Iteration:   9401, Training Accuracy:  98.4%
Optimization Iteration:   9501, Training Accuracy:  98.4%
Optimization Iteration:   9601, Training Accuracy:  98.4%
Optimization Iteration:   9701, Training Accuracy:  98.4%
Optimization Iteration:   9801, Training Accuracy: 100.0%
Optimization Iteration:   9901, Training Accuracy: 100.0%
Time usage: 0:25:33
Accuracy on Test-Set: 98.6% (9857 / 10000)
springsteen (~) >
```



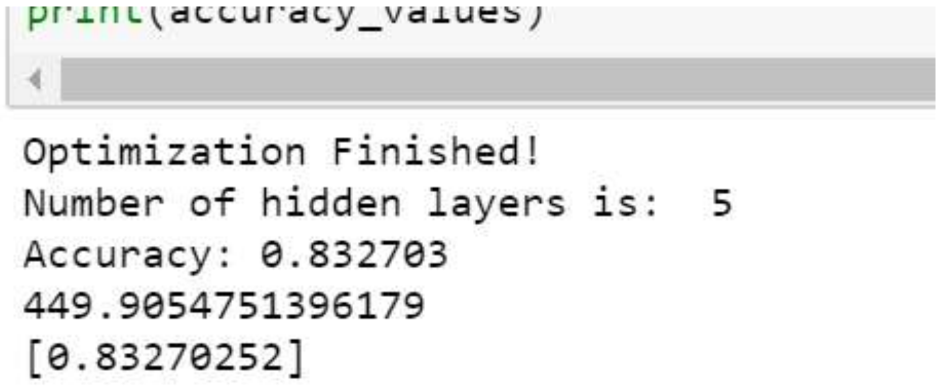
```
login as: skanneti
skanneti@springsteen.cse.buffalo.edu's password:
Last login: Wed Apr 18 14:08:04 2018 from dhcp012-023-138.wireless.buffalo.edu

springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python ./cnnScript.py
  File "./cnnScript.py", line 402
    """
      ^
TabError: inconsistent use of tabs and spaces in indentation
springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python cnnScript.py
  File "cnnScript.py", line 402
    """
      ^
TabError: inconsistent use of tabs and spaces in indentation
springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python cnnScript.py
  File "cnnScript.py", line 402
    """
      ^
TabError: inconsistent use of tabs and spaces in indentation
springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python cnnScript.py
  File "cnnScript.py", line 402
    """
      ^
TabError: inconsistent use of tabs and spaces in indentation
springsteen {~} > ls
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python cnnScript.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
```



```
cnnScript.py  DIC_sai.zip      mnist_all.mat
DIC_Lab2.zip  face_all.pickle  Windows
springsteen {~} > python cnnScript.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting data/MNIST/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting data/MNIST/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting data/MNIST/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/MNIST/t10k-labels-idx1-ubyte.gz
Size of:
- Training-set:         55000
- Test-set:             10000
- Validation-set:       5000
Accuracy on Test-Set: 7.1% (711 / 10000)
Optimization Iteration:      1, Training Accuracy:  12.5%
Time usage: 0:00:00
Accuracy on Test-Set: 8.6% (861 / 10000)
Time usage: 0:00:14
Accuracy on Test-Set: 67.4% (6740 / 10000)
Optimization Iteration:    101, Training Accuracy:  75.0%
Optimization Iteration:    201, Training Accuracy:  75.0%
Optimization Iteration:    301, Training Accuracy:  75.0%
Optimization Iteration:    401, Training Accuracy:  79.7%
Optimization Iteration:    501, Training Accuracy:  87.5%
Optimization Iteration:    601, Training Accuracy:  95.3%
Optimization Iteration:    701, Training Accuracy:  90.6%
Optimization Iteration:    801, Training Accuracy:  90.6%
Optimization Iteration:    901, Training Accuracy:  93.8%
Time usage: 0:01:39
Accuracy on Test-Set: 92.7% (9273 / 10000)
Optimization Iteration:   1001, Training Accuracy:  87.5%
Optimization Iteration:   1101, Training Accuracy:  96.9%
Optimization Iteration:   1201, Training Accuracy:  93.8%
Optimization Iteration:   1301, Training Accuracy:  90.6%
Optimization Iteration:   1401, Training Accuracy:  95.3%
Optimization Iteration:   1501, Training Accuracy:  93.8%
```

The Test accuracy obtained after 10,000 iterations from CNN is: 98.60%
The training time is: 25 minutes and 33 sec.



```
print(accuracy_values)
```

```
Optimization Finished!
Number of hidden layers is:  5
Accuracy: 0.832703
449.9054751396179
[0.83270252]
```