Report for Assignment 2

Vishesh Singh Thakur
50322513, vthakur@buffalo.edu

Ashwin Nair
50321006, anair3@buffalo.edu

Divya Gawande
50340326, divyagaw@buffalo.edu

Group 2
CSE 574 Introduction to Machine Learning
October 2020

Abstract

In this assignment, we want to implement Multilayered Perceptron Neural Network, and evaluate its performance in classifying handwritten digits. We use the same network to analyze a celebrity face dataset and compare the performance of the neural network against the deep neural network and a convolutional neural network using the TensorFlow library

PREPROCESSING

We have 60000 training examples and 10000 testing examples. We split the data into two parts of training (50000 examples) and validation (10000 examples). We assign 1000 random examples to the validation set, and since there are a total of 10 keys, our validation set becomes a total of 10000 examples. The keys are specified in the dataset and are used to assign labels to the examples

FEATURE SELECTION

We can observe that there are many features in the data set that have the exact same value for all the data points. These points add no information to increase the accuracy of the model as they have no variation in the values. Thus, these points can be removed while training the model. The following features were removed during the pre-processing

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 52, 53, 54, 55, 56, 57, 82, 83, 84, 85, 111, 112, 140, 168, 476, 560, 644, 671, 672, 673, 699, 700, 701, 727, 728, 729, 730, 754, 755, 756, 757, 758, 759, 780, 781, 782, 783]

The implementation of Neural Network requires forward pass and back propagation

FORWARD PASS

For the forward pass, we use the following formula and feed it into the sigmoid function which is called as the activation function. Here a_j is the linear combination of the input data, w_{jp} represents the weights for the p^{th} input feature, for the j^{th} hidden layer. The bias of the nodes is directly set to 1

$$a_j = \sum_{p=1}^{d+1} w_{jp}^{(1)} x_p$$

$$z_j = \sigma(a_j) = \frac{1}{1 + \exp(-a_j)}$$

The final output is calculated by the linear combination of hidden layer output z with the weights, which is then parsed through a sigmoid function. We want to classify a hand-written digit image to its corresponding class, so we use the one-vs-binary classification in which each output unit represents the probability of an image belonging to a particular digit. The total number of output unit is set as k = 10. We use the following function to calculate the final output. Here w_{lj} represents the weights of j^{th} hidden layer for the l^{th} output layer, combined with z_j which is the output from the hidden layer calculated in the previous step

$$b_l = \sum_{j=1}^{m+1} w_{lj}^{(2)} z_j$$

$$o_l = \sigma(b_l) = \frac{1}{1 + \exp(-b_l)}$$

ERROR AND BACK PROPAGATION

For the error and back propagation, we use the following function. The error in this case is the negative log-likelihood function. Here, y_{il} represents the l^{th} target value in 1-of-K coding scheme of input data i and o_{il} is the output at l^{th} output node for the i^{th} data

$$J_i(W^{(1)}, W^{(2)}) = -\sum_{l=1}^{k}(y_{il} \ln o_{il} + (1 - y_{il}) \ln(1 - o_{il}))$$

REGULARIZATION

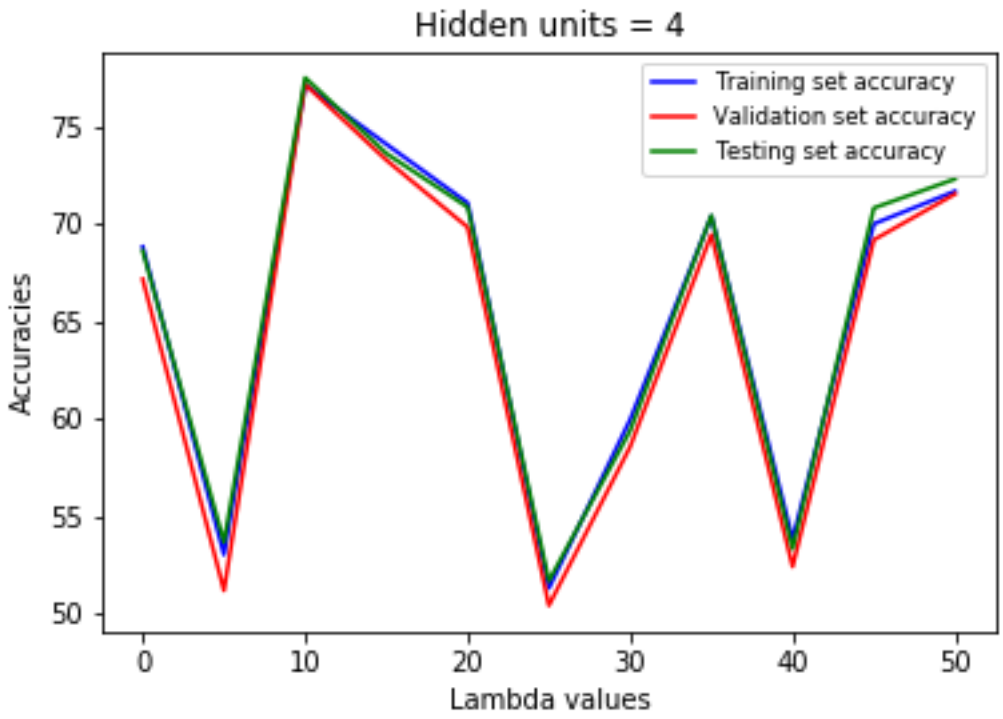
Overfitting is a major problem while learning a model, where the model fits best with the training data but gives poor generalization with the test and validation data. To avoid such a problem, we add a regularization term to our error function to control the magnitude of parameters in Neural Network. The following function is used as a regularization function. Where λ represents the regularization coefficient, and n is the size of the training data

$$\widetilde{J}(W^{(1)}, W^{(2)}) = J(W^{(1)}, W^{(2)}) + \frac{\lambda}{2n}\left(\sum_{j=1}^{m}\sum_{p=1}^{d+1}(w_{jp}^{(1)})^2 + \sum_{l=1}^{k}\sum_{j=1}^{m+1}(w_{lj}^{(2)})^2\right)$$

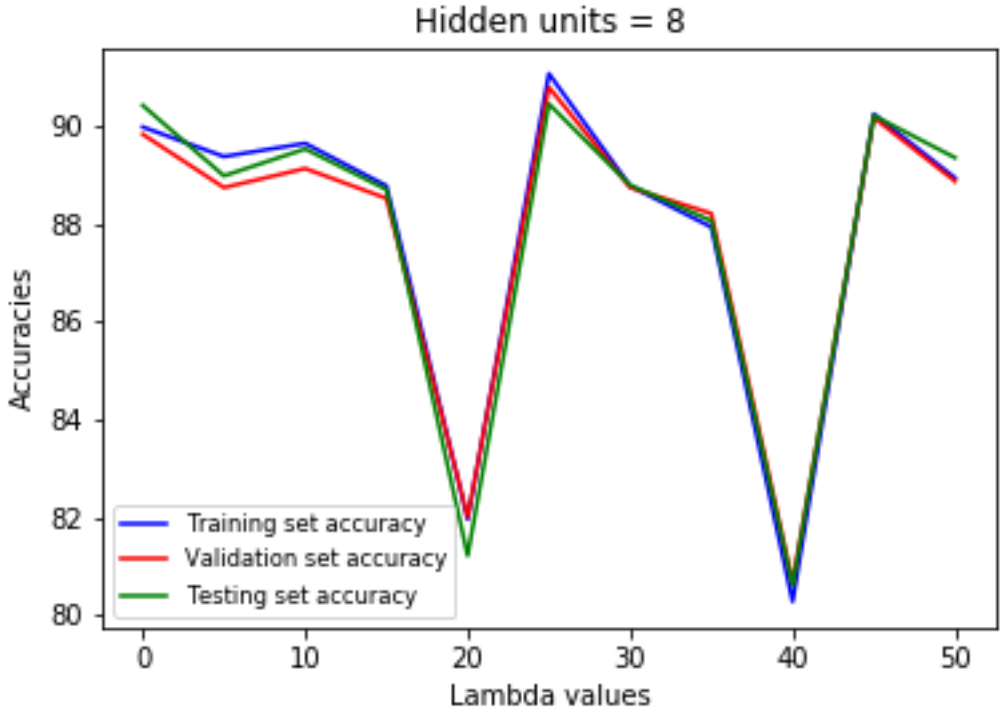
TUNING THE HYPERPARAMETERS

In Machine learning, a hyperparameter is a parameter whose value is used to control the learning process, that in theory has no influence on the performance of the model, but affect the speed and quality of the learning process. For the tuning, we vary our hyperparameters, both λ and number of hidden units. We vary λ from 0 to 60 in the intervals of 5, and vary the number of hidden units as 4, 8, 12, 16, and 20. For the comparison of different performances, we compare the accuracies of training, validation and testing set for different values of λ while keeping the number of hidden units fixed at a time. We tune and select our hyperparameters based on the accuracies achieved for the validation set. Training on validation set prevents both underfitting and overfitting.

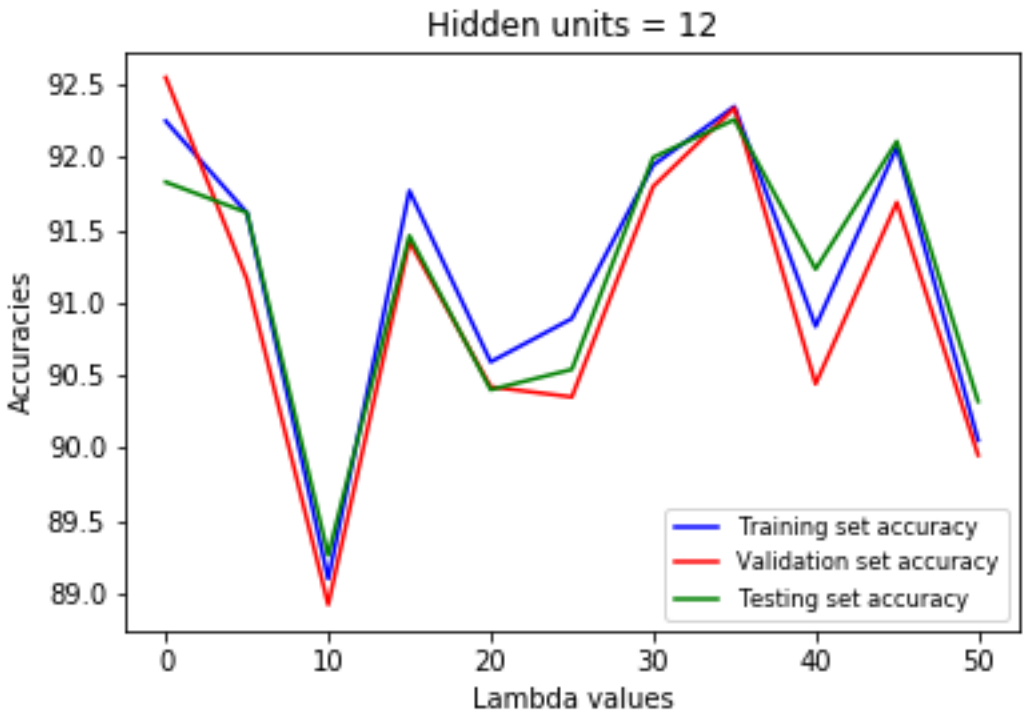
For number of hidden units as 4



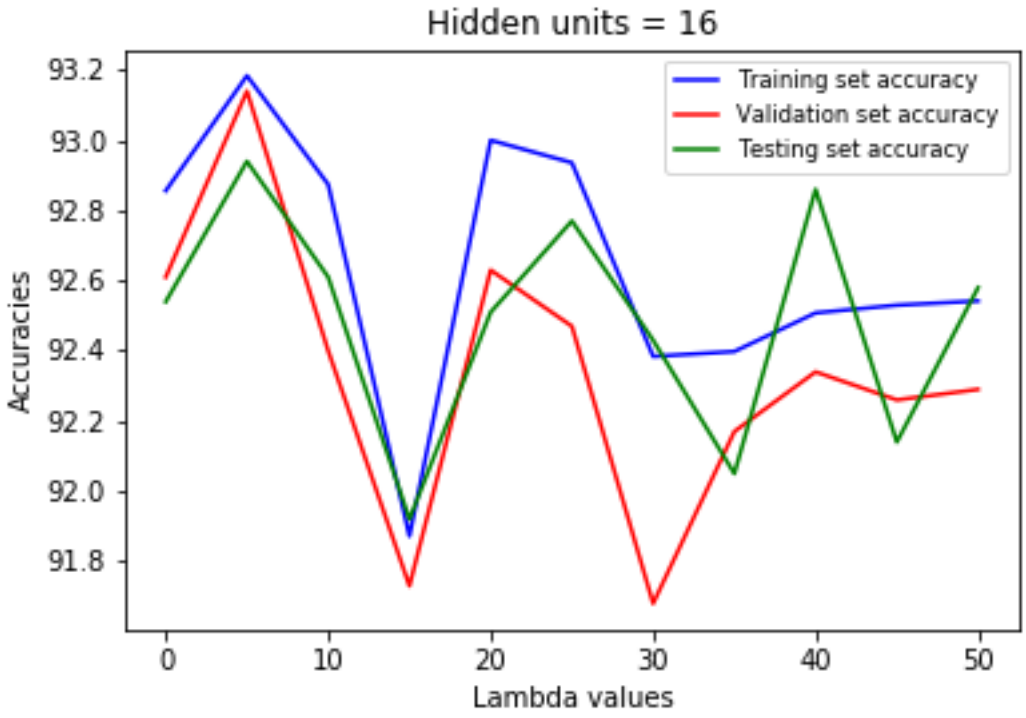
Here, we observe that accuracies decrease for smaller values of λ, and then maximizes for value 10, and then continues fluctuating. The maximum validation set accuracy is achieved at the λ value of 10



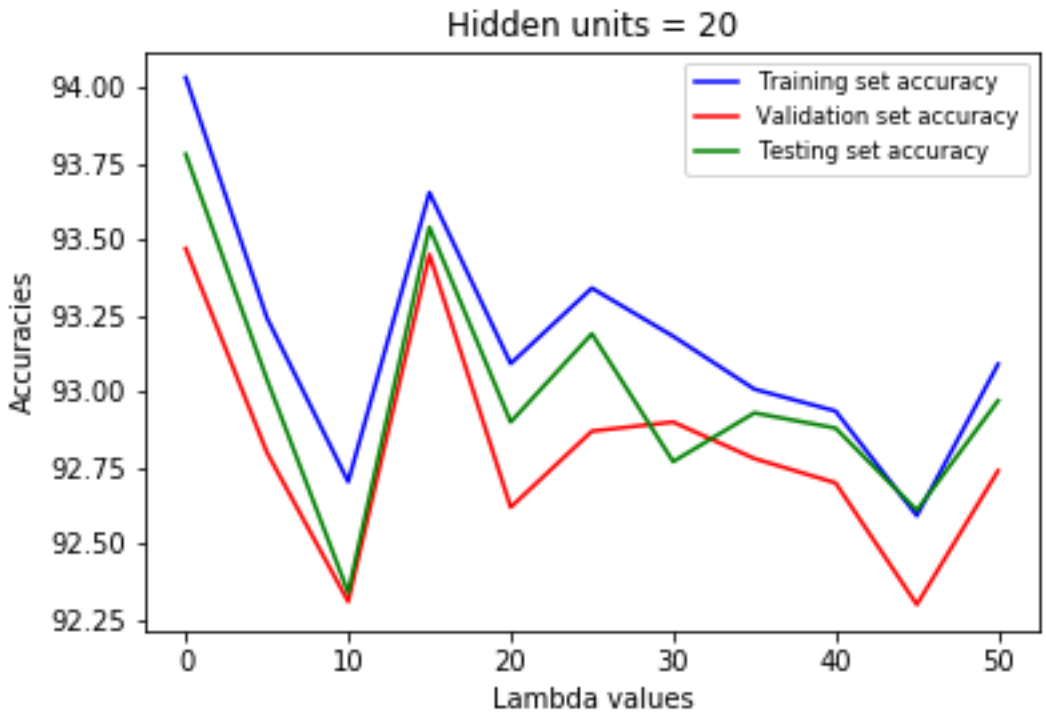
Here, we observe that for 8 hidden units, the accuracies start high and decreases continuously for λ value of 20, then maximizes for λ value of 25, and continues fluctuating between high and low



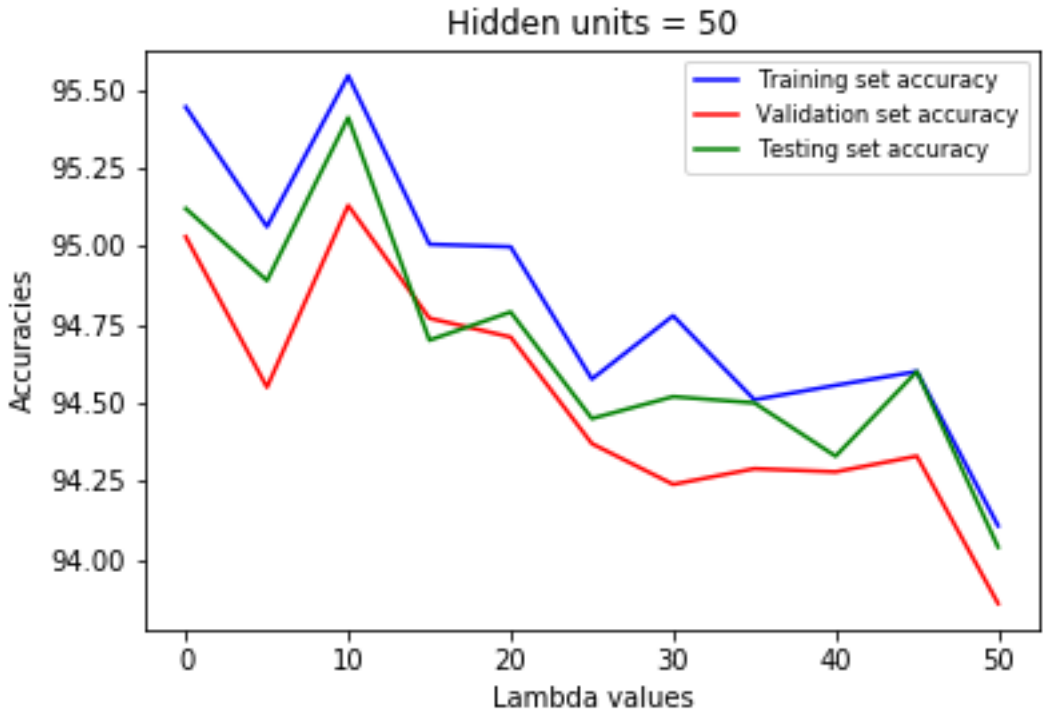
For 12 hidden units, the accuracies fluctuate between high and low values, with maximum validation set accuracy achieved at λ value of 0



For 16 hidden units, the validation set accuracy maximizes at λ value of 5, and then starts fluctuating between high and low



For the value of hidden nodes set to 20, the validation set accuracy maximizes at λ value of 0, and then starts fluctuating between high and low values



For the default value of 50 hidden units given by the instructor, we get the best validation accuracy at λ value of 10

After comparison between all the values, the maximum value of validation set accuracy is found at λ value of 0, and number of hidden nodes set to 20.

If we observe closely at the minimum and maximum accuracies at each graph, we can see that for models having low number of hidden units have lower average accuracies, and it increases as we increase the number of hidden units. The reason is undertraining, as the more hidden units the model has, the better it is able to train the model. The best validation accuracy is observed in the given example nnScript.py, which has 50 hidden units, and a λ value of 10

ACCURACY OF CLASSIFICATION OF HANDWRITTEN DIGITS TEST DATA

From the experiment, we calculated the optimal hyperparameters for our neural network, with λ value of 0, and 20 hidden units. The highest validation set accuracy was found to be 93.47% at the optimal values of hyperparameters. We know that on changing the value of λ, we change the amount of regularization in the objective function. On increasing the value of λ for a fixed set of hidden units, the accuracies of training, validation, and testing sets all start fluctuating, reaching the respective maximum testing accuracies at lower levels of λ. We get the best validation set accuracy of 95.13% at 50 hidden units and λ value of 10

ACCURACY OF CLASSIFICATION METHOD ON THE CELEBA DATASET

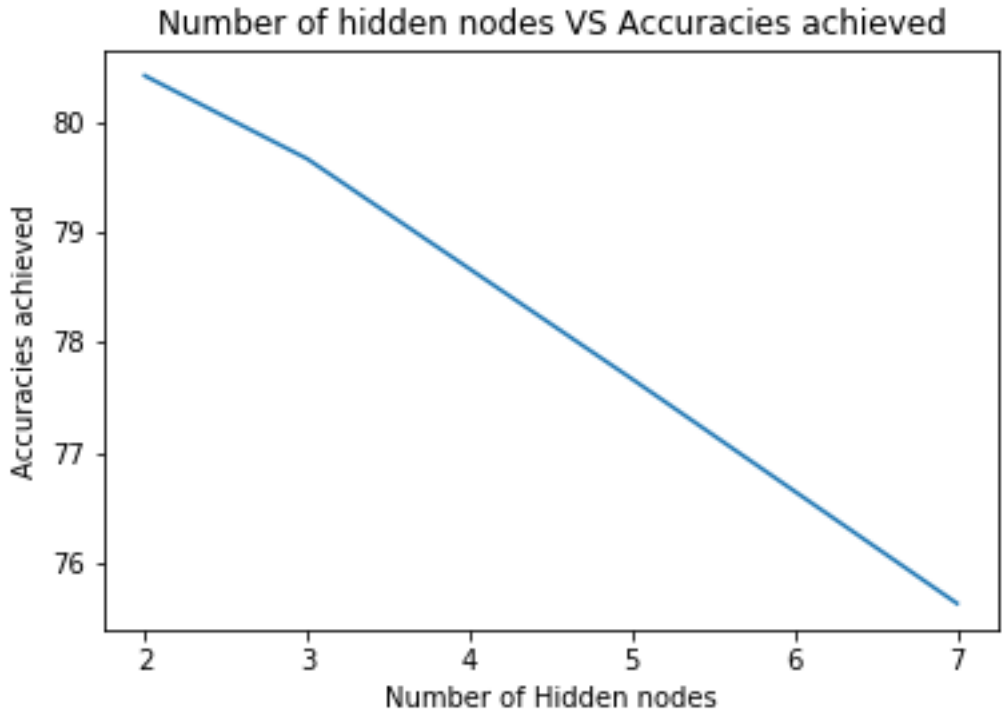
For the facennScript, we initialize the sigmoid function and use it to calculate the sigmoid outputs after each layer. After implementing the optimization and adding regularization, we calculate the accuracy of single layered neural network on CelebA dataset to distinguish between the two classes of wearing glasses and not wearing glasses. The following snippet shows the result of single layered neural network

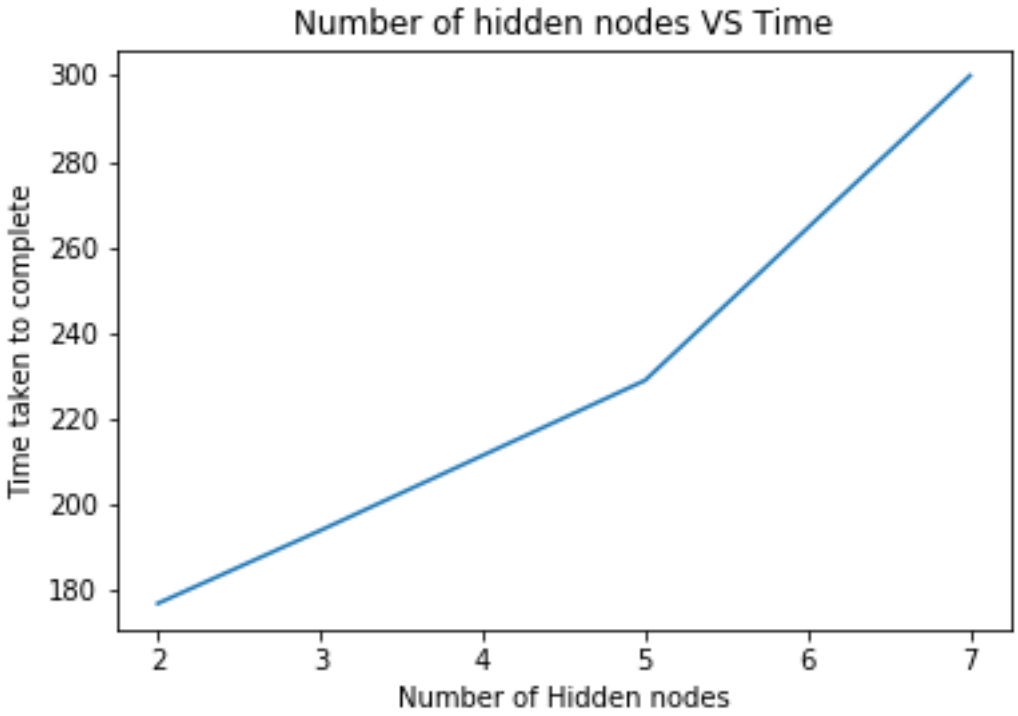
```
Training set Accuracy:85.59715639810427%

Validation set Accuracy:84.35272045028142%

Test set Accuracy:85.91975775927327%
```

After the implementation of single layered neural network, we decided to implement multi-layered deep neural network on the CelebA dataset. The comparison of the accuracies of Deep Neural Network for different number of hidden layers is shown in the following table

NUMBER OF HIDDEN LAYERS	ACCURACY (IN %)	TIME TAKEN FOR THE OPTIMIZATION TASK (IN HR:MIN:SEC)
2	80.431	0:02:57
3	79.67499	0:03:14
5	77.668434	0:03:49
7	75.624536	0:05:00

We add the original code provided for 2 hidden layers and the accuracy calculated in the process, and for comparison, we plot the following two graphs





We can infer from the graphs, that as we increase the number of hidden layers the complexity of the process increases which in turn increases the time taken for optimization and also decreases the accuracy of the model

COMPARISON BETWEEN DEEP NN AND NORMAL NN

In the normal neural network, we use only one hidden node, whereas in deep neural network we use multiple hidden nodes. We calculated the accuracies of having 2, 3, 5, and 7 hidden nodes, and the best accuracy, which is calculated at 2 hidden nodes, is selected for comparison

The following table compares the testing accuracy of the best deep neural network and for the normal neural network with 256 hidden nodes, and the λ value of 10

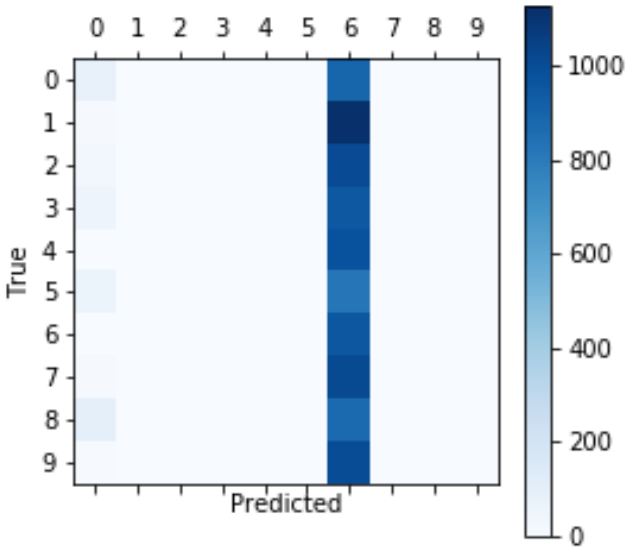
VALUES	DEEP NEURAL NETWORK	FACE NEURAL NETWORK
ACCURACY IN %	80.431	85.049
TIME TAKEN IN HR:MIN:SEC	0:02:57	0:01:58

CONVOLUTIONAL NEURAL NETWORK

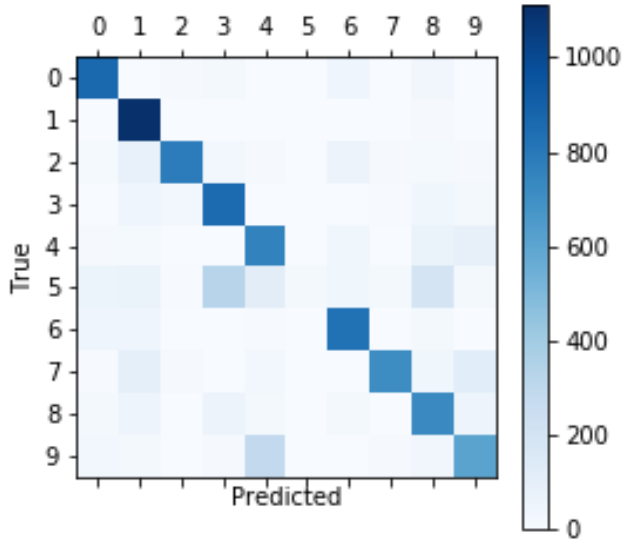
A Convolutional Neural Network (CNN) is a deep learning method that can be used for the purpose of feature extraction as well as classification. Hence, CNN acts as a feed-forward network that extracts topological properties from images. It extracts features from raw images (i.e. contain the intended pattern) in its first layers, and then classifies the pattern with its last layers.

The confusion matrices for different number of iterations of the convolutional neural network can be observed below:

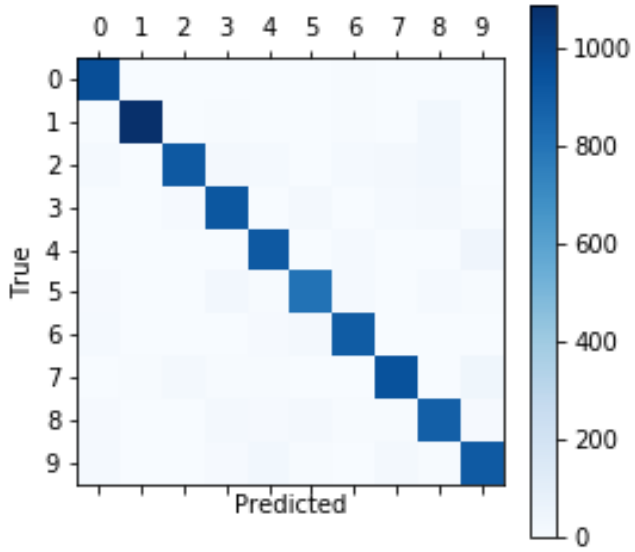
For number of iterations = 1



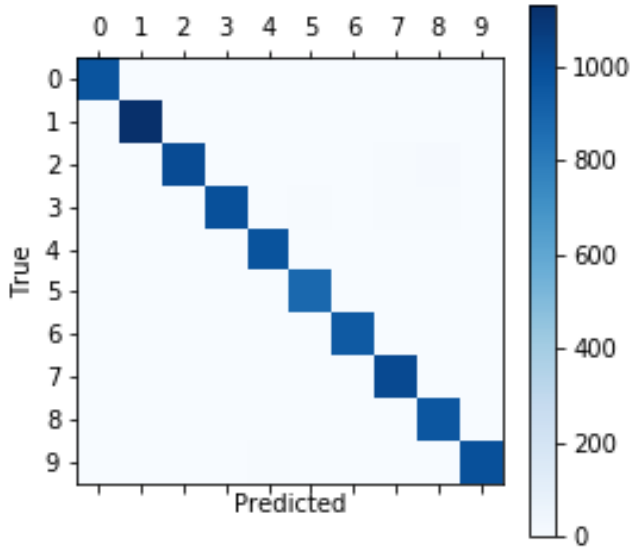
For number of iterations = 99



For number of iterations = 900



For number of iterations = 9000



NUMBER OF ITERATIONS	TIME TAKEN IN HR:MIN:SEC	TEST ACCURACY IN %
1	0:00:03	8.4
99	0:00:09	63.9
900	0:00:53	93.6
9000	0:08:43	98.9

We can observe that the testing accuracy increases as we increase the number of iterations, however the time taken for the optimization also increases. From the confusion matrix, we can observe that the number of errors decrease with the increase in iterations