CSE 574 : Introduction to Machine Learning

Programming Assignment 2: Handwritten Digits Classification

Group 4

Shubham Sharma (Person No.: 50290293
Aditya Sahay(Person No.: 50292761)
Katyayni Shankar Kaushik(Person No.: 50289158)

April 14, 2019

The dataset provided to us, i.e. *mnist_all.mat* (original dataset from MNIST has 10 matrices for testing set and 10 matrices for training set, corresponding to 10 digits. The training set data (60000 examples) is split into training (50000 examples) and validation data (10000 examples) using *train_test_split* function in *scipy* library. Also, as a part of the pre-processing step we ignore the columns(features) which give no new information, i.e. for which there is no variation between the data points (standard deviation = 0). The Python scipy function *scipy.optimize.minimize* is used to solve the optimization problem of minimizing the weights using the option, conjugate gradient descent.

Choice of Hyperparameters for the Neural Network (number of hidden units, regularization term λ)

The validation data is used to estimate the hyperparameters. The purpose of adding the regularization term, λ is to avoid over-fitting so that our model generalizes well and makes fewer mistakes on the test data. Algorithms that do not learn anything from the training data are the ones which have 0 weights except for the bias term, in the context of the neural networks. In these models the effect of the inputs i.e. X would be nothing and we get the same result everytime. This model has no variance and high bias, i.e. even before training it knows what needs to be done. On the other hand, a model has very high variance and low bias if it follows the training data perfectly. Tuning of the regularization parameter, λ is required to get the best bias-variance trade-off. The new objective function after adding the regularization term is given by the following equation:

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

Clearly, if $\lambda = 0$ we just look at the first term in the equation, i.e. $J(W^{(1)}, W^{(2)})$ and we are only concerned with minimizing our mistakes on the training data and hence, we are over-fitting. If λ is very high we try to get models with very low weights because if weights are high the second term in the above equation becomes very high. Models having large weights will over-fit because if weights are large , we get a large change in output for changes in X. On the other hand, if the weights are lower it is less sensitive to the data.

As is clear from figure 3, the optimal λ can be seen as the one for which we get the highest accuracy which is for $\lambda = 0$. Although, there is not much difference in accuracy as we increase λ for 20 hidden layers. The plot is obtained by plotting accuracy by varying the values of λ from 0 to 60 in steps of 10. Also, we observe the accuracy for different number of hidden layers, i.e. 4, 8, 12, 16, 20 and observe that as the hidden layers increase, the accuracy increases. This is because the hidden nodes represent the number of learned features and it is also observed that at higher number of hidden layers the accuracy doesn't change much.

For training set, as λ is increased the accuracy will decrease, however for validation as well as test set the accuracy should ideally increase. This is not perfectly followed in our graph although we see the required trend for higher number of hidden layers. We have also plotted the one standard error bar for different values of λ.

The accuracy of our implementation increases as the number of hidden nodes increases but after a limit test and validation accuracy will decreases because of possible chances of overfitting. However, we do not observe this here since we only check till 20 hidden layers. 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 much.

Comparison of number of hidden nodes with the training time

As is clear from figure 4, as the number of nodes in the hidden layer is increased, the learning time also increases as expected since with more nodes the matrix of weights become larger and more computation is required. The number of weight updation increases as we add more nodes and this value is being computed several times before arriving at the optimal weights. Although, we see a dip in the figure this might be because of other operations running on the system parallely while recording the run time.

The summary for the Handwritten digits dataset and CelebA dataset for the single layer neural network architecture is provided in table 1.

Parameter	Handwritten digits dataset	CelebA dataset
λ	0	10
Number of hidden nodes	20	256
Time required for training	18 secs	54 secs
Training Set Accuracy	93.858 %	84.436 %
Validation Set Accuracy	93.01 %	83.715 %
Test Set Accuracy	93.78 %	84.898 %

Table 1: Summary for the Handwritten digits dataset and CelebA dataset for the single layer Neural Network

We observe that as we increase the nodes in the hidden layer, the accuracy improves, but there is a trade-off between the training time and the accuracy as we increase the number of nodes, the training time also increases. The optimal value of λ for which we get the best accuracy is: 0 as the accuracy decreases on further increasing the value for 20 hidden layers as observed in figure 3.

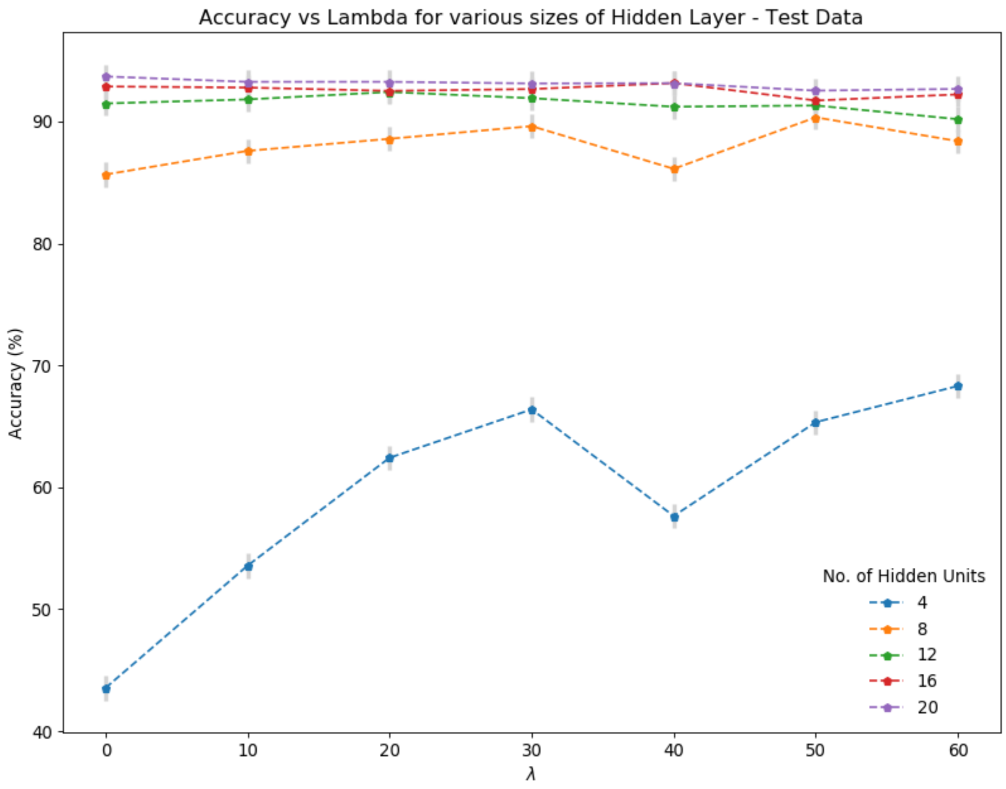

Figure 1: Plot for Accuracy vs λ for various sizes of Hidden Layer for test data

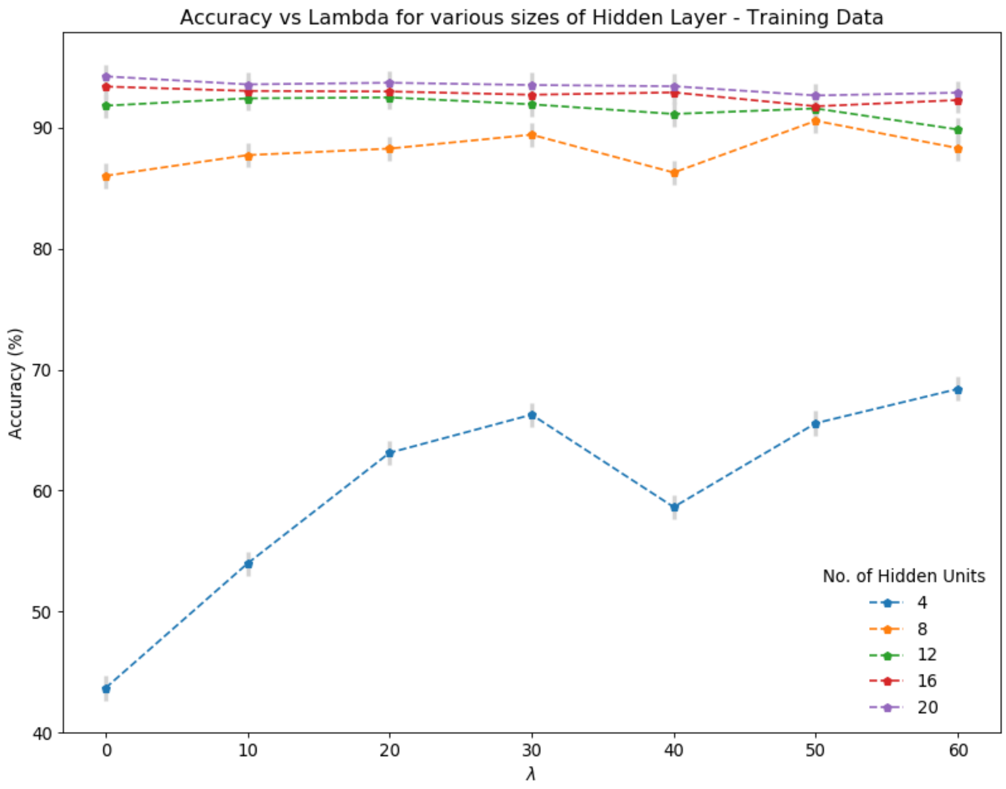

Figure 2: Plot for Accuracy vs λ for various sizes of Hidden Layer for training data

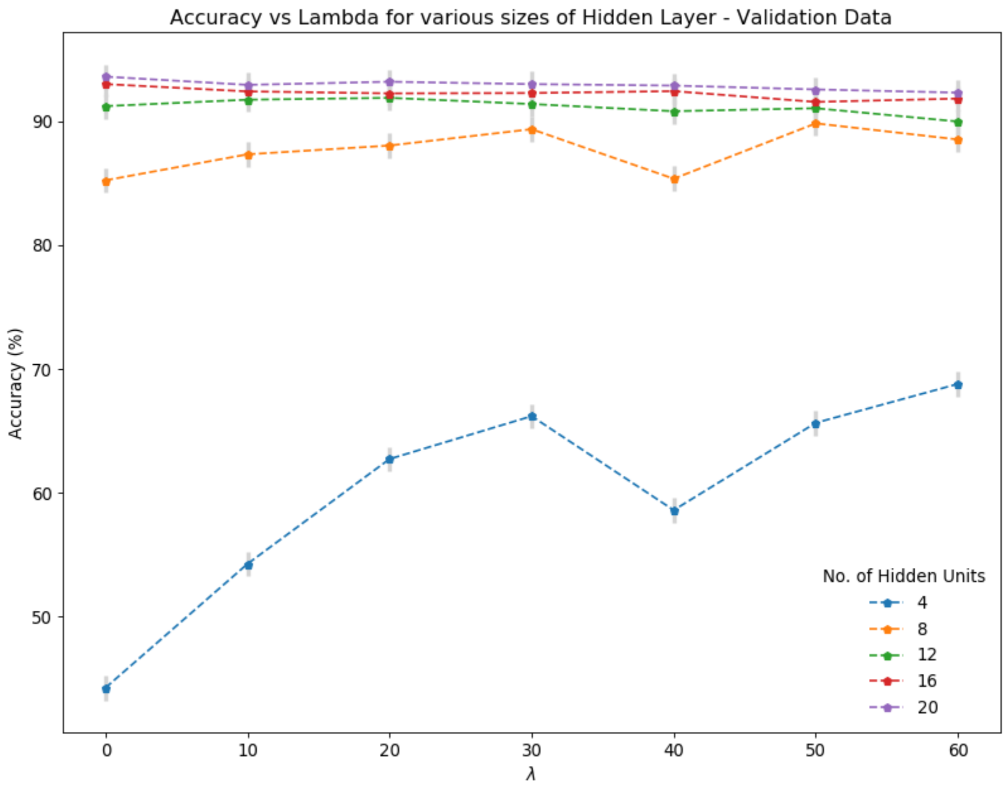

Figure 3: Plot for Accuracy vs λ for various sizes of Hidden Layer for Validation data

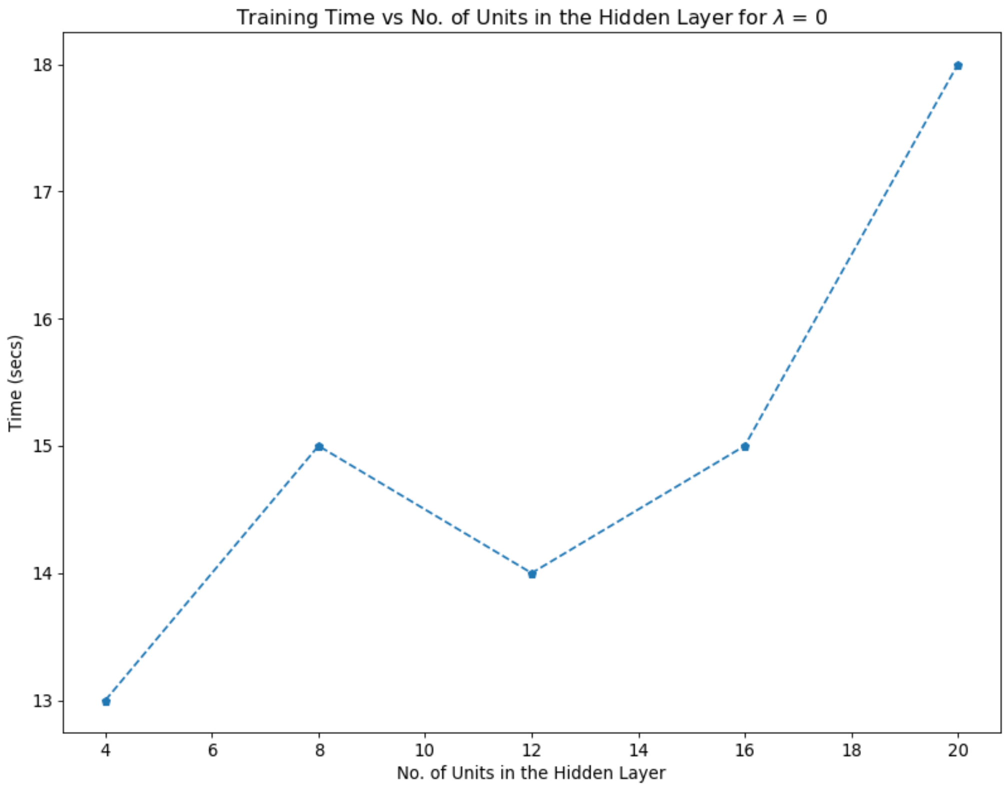

Figure 4: Variation of training time vs no. of hidden units in the hidden layer for $\lambda = 0$

Comparison of our Neural Network with a Deep Neural Network (using TensorFlow) in terms of accuracy and training time

The CelebFaces Attributes Dataset (CelebA) has 26407 face images and the labels are 1 for glasses and 0 for not glasses.

The training times for deep neural network with 3,5,7 hidden layers each having 256 nodes are 80 secs, 96 secs and 114 secs respectively whereas the accuracy for the same is 78.9 % , 77.7 % and 75.1 % respectively.

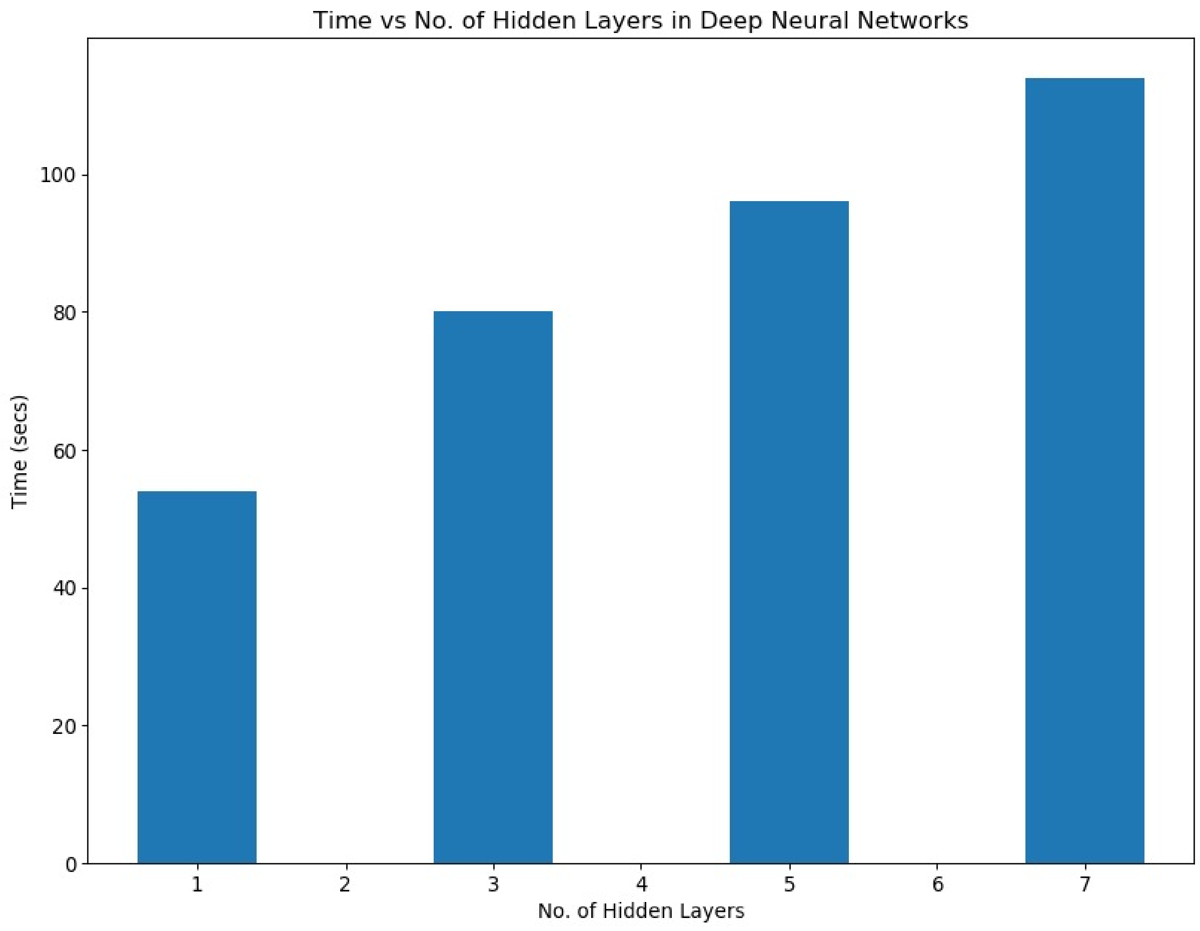

Figure 5: Time vs No. of Hidden Layers in Deep Neural Networks and Single Layer Neural Network

The accuracy and training time for our single layer neural network and Deep Neural Network with hidden layers: 3,5,7 is shown in figures 5 and 6. We observe that as we vary the number of nodes in the hidden layers in the nodes the accuracy changes significantly. It is an accepted trend to take the number of hidden neurons as 2/3 the size of the input layer, plus the size of the output layer and the number of hidden neurons less than twice the size of the input layer. We observe that we get an accuracy of 83.8 % on the test set and a training time of 36 secs when we use 64, 32 and 16 nodes in the 3 layer deep neural network architecture. For 256, 128, 64, 32, 16 node 5 layer deep neural architecture, we get an accuracy of 78.9 % and takes 73 seconds to train. For 7 layers (all 128 nodes) we get an accuracy of 79 % and a training time of 66 seconds.

As is clear from figures 5 and 6, the single layer neural network outperforms the multilayered neural network implemented here in terms of both accuracy on the test set as well as the time taken for training.

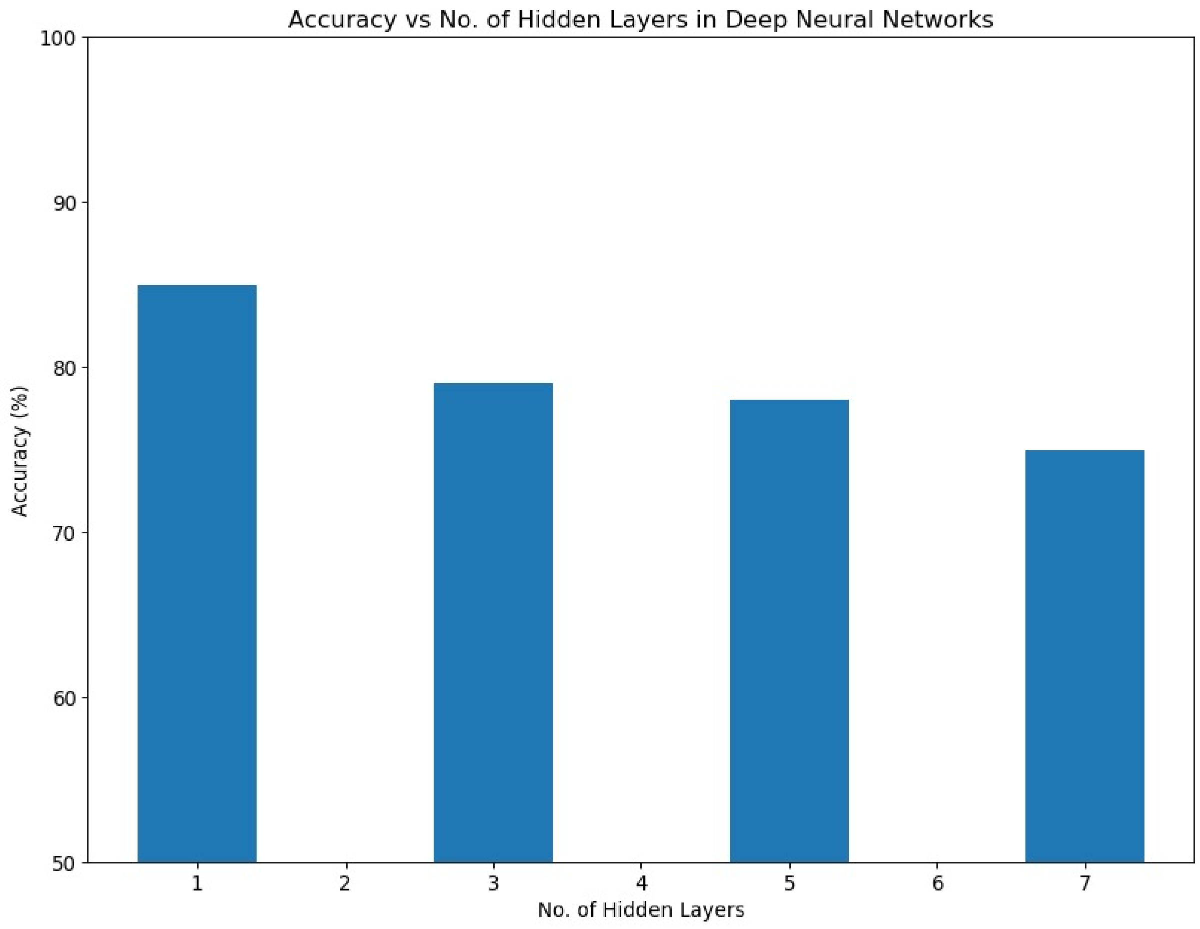

Figure 6: Accuracy vs No. of Hidden Layers in Deep Neural Networks and Single Layer Neural Network

Results from the Convolutional Neural Network in terms of accuracy and training time

The training time and accuracy for test set(10000 examples) is summarized in table 2. As is clear from the table 2, we observe that as the number of iterations increase the accuracy increases as well as the training time which is intuitive. Since, with more iterations our weights get learned more number of times and are nearer to the truth. The confusion matrix for number of iterations 1, 99, 900, 9000 is shown in figures 7, 8, 9 and 10. Clearly, the confusion matrix has large values along the diagonals(green color) which show more number of correct predictions for more iterations as is shown in the table as well.

No. of iterations	Test set accuracy(in %)	Training Time(in secs)
1	12.7	0
99	72.8	3
900	92.8	25
9000	98.7	247

Table 2: Summary for the training time and accuracy for test set of Handwritten digits dataset

The time required for fully connected neural network is more than the convolutional neural network in some cases since here more number of weights(parameters) need to be learned where as in convolutional neural network the main image matrix is reduced to a matrix of lower dimension in the first layer itself through convolution. Networks having large number of parameter face several problems, for e.g. slower training time, chances of overfitting etc.

All computations and recording of run times is done on Macbook Pro (Core-i7 8th Generation 16 GB RAM)

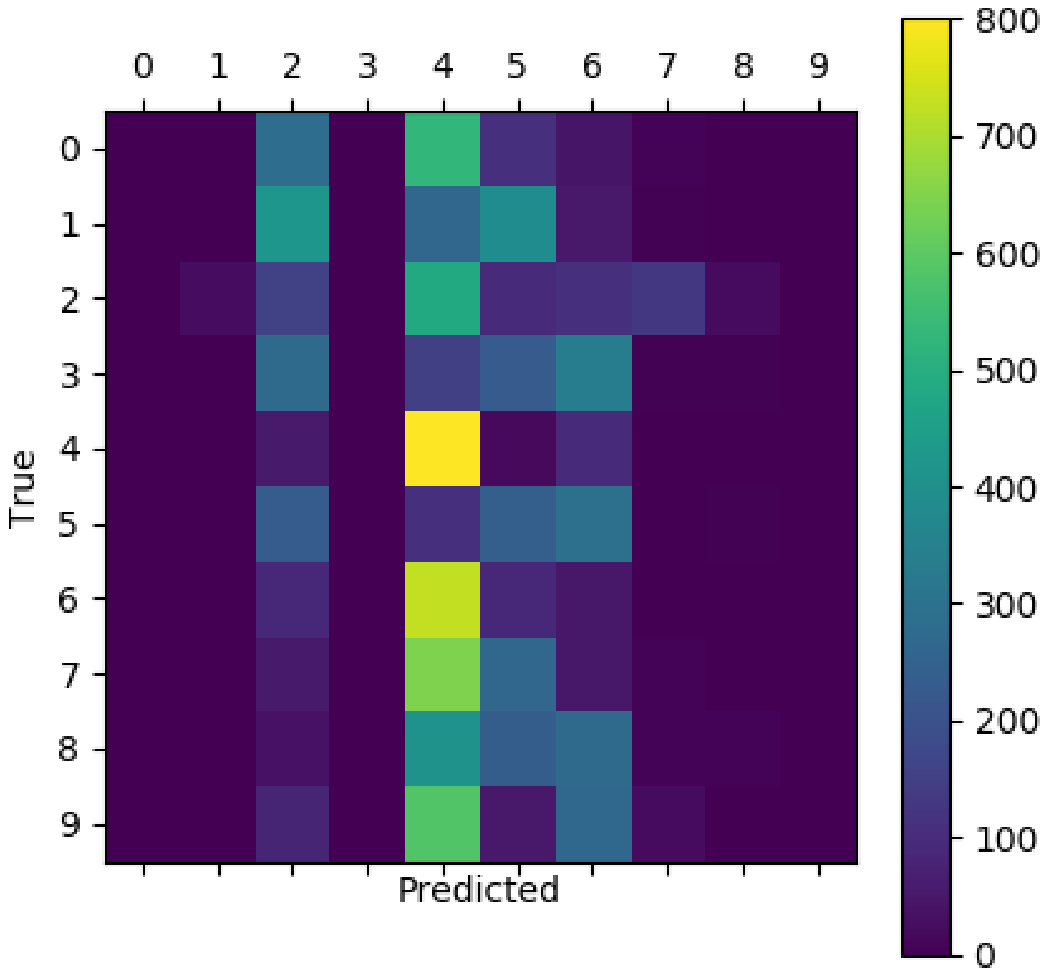

Figure 7: Confusion Matrix for 1 iteration

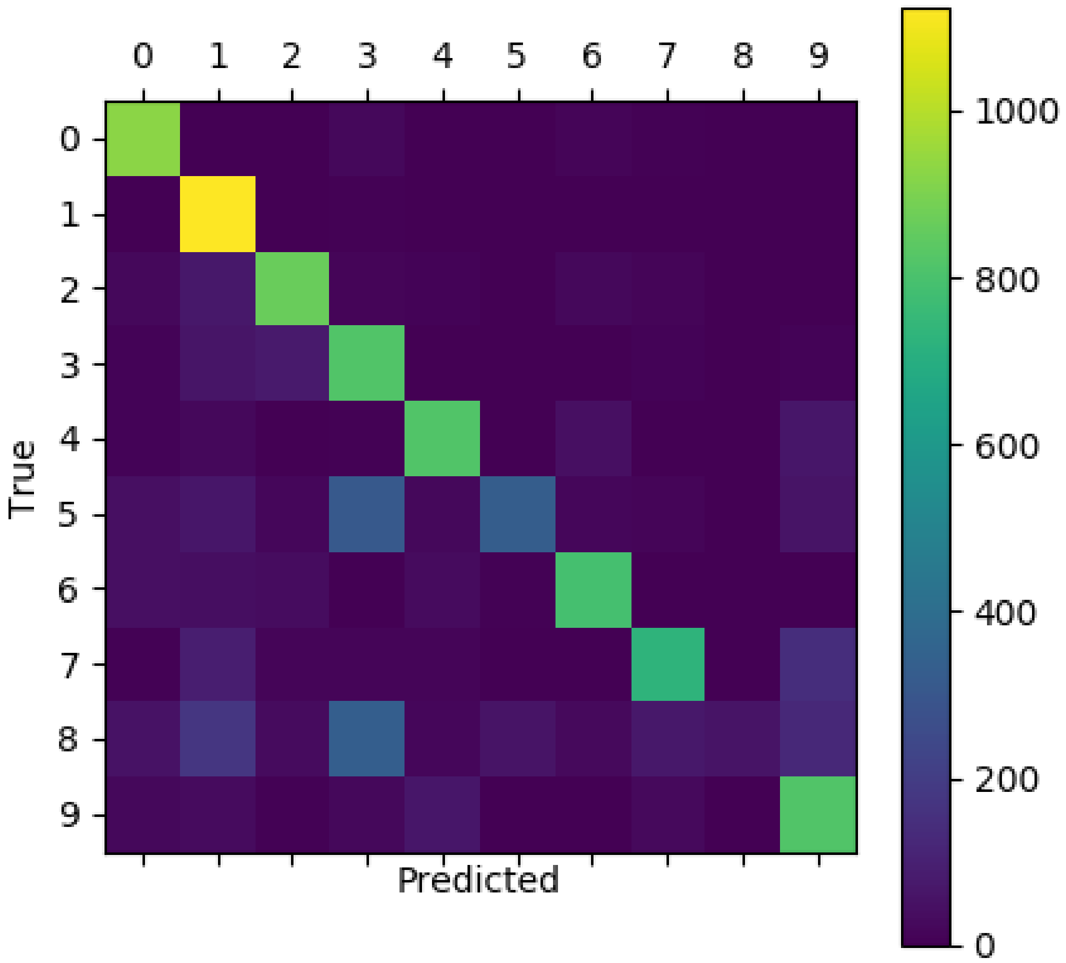

Figure 8: Confusion Matrix for 99 iteration

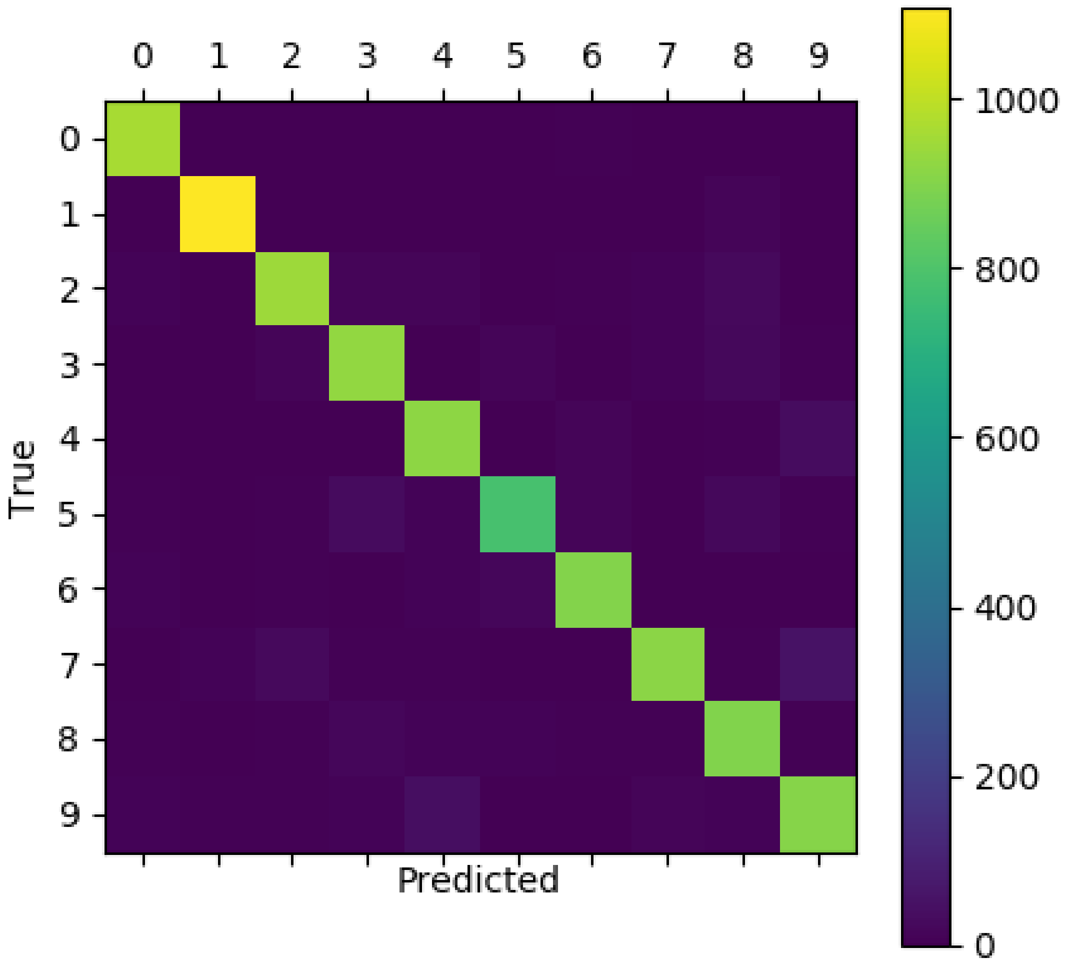

Figure 9: Confusion Matrix for 900 iteration

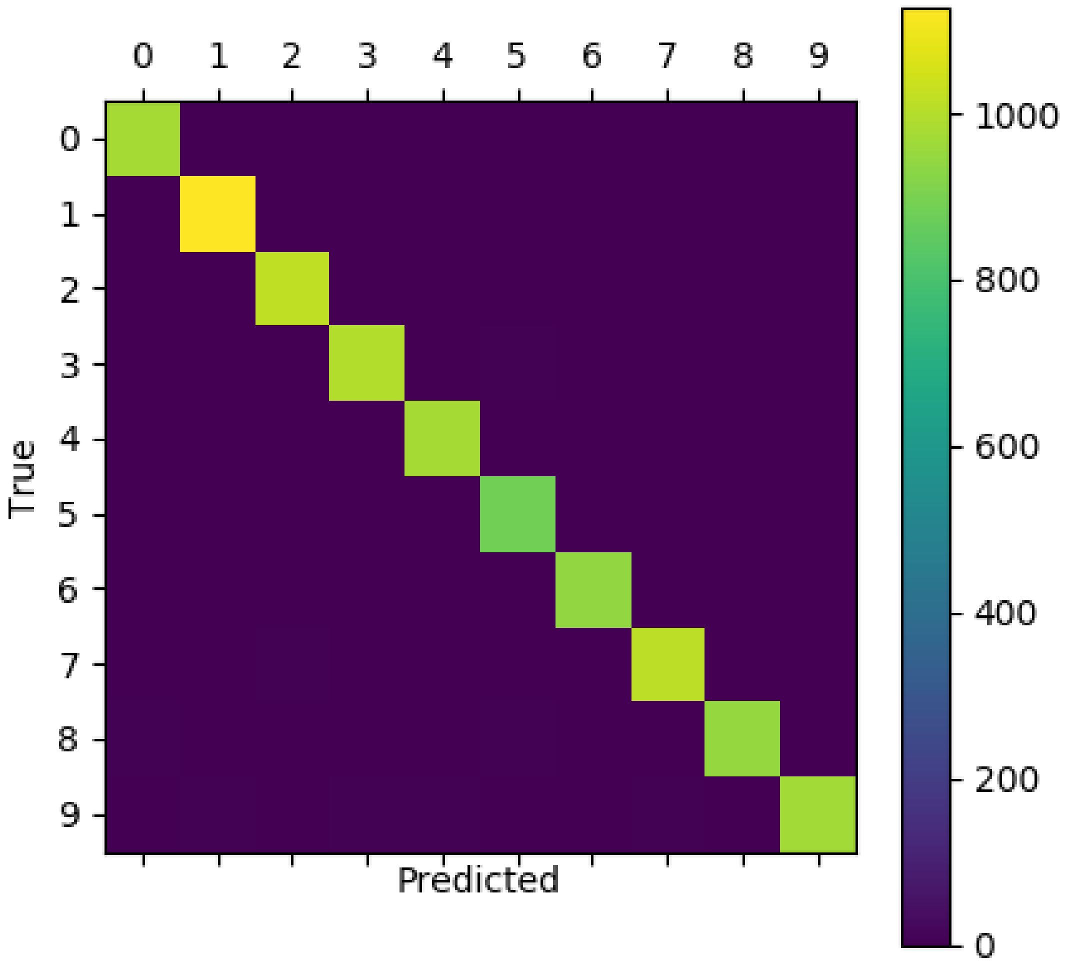

Figure 10: Confusion Matrix for 9000 iteration