CSE 574 : Introduction to Machine Learning

Assignment 2 - Group 3

Nupur Sunil Agrawal **Syed Aqhib Ahmed** **Venkata Sai Abhishekh**

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

MNIST Dataset:

Preprocessing:

The 'mnist all.mat' dataset consisted of 10 matrices for testing set and 10 matrices for training set, which corresponded to 10 digits. We split the training sets into two sets of 50000 randomly sampled training examples and 10000 validation examples. Each row of this matrix represents the feature vector of a particular image.

Feature Selection:

In feature selection, we were to remove the columns which had the same value for all the data points. Thus, 67 features were dropped and 717 total features were selected.

Choosing Hyperparameters:

In neural network we generally deal with three kinds of hyper-parameters namely regularization parameters (λ), number of hidden nodes in a hidden layer and number of hidden layers. We tried to regularize the neural network using different values of hyper_parameters (lambda and the number of hidden nodes) and chose the hyper-parameters which provides us with the best results. Following are the results obtained for different combinations of the hyper-parameters:

We can see from the table that we attain the maximum validation accuracy when the lambda value is 10 and the number of hidden nodes is 50 .The model takes 0.04859 seconds to train on the hand written images data set and gives a validation accuracy of 94.8%, training set accuracy of 95.406% and test set accuracy of 94.78%.

We plot the graphs or various lambda(λ) values against the accuracy of Neural Network for each value of the hidden layer, to study the relation between lambda and the performance of our neural network.

Table 1.1 : Various Lambda values for 4 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
4	0	71.074	70.57	70.59	0.0339642231464386
4	5	79.622	78.99000000000001	78.85	0.02912790584564209
4	10	70.923	70.32000000000001	70.15	0.03175808095932007
4	15	63.217	62.86000000000001	62.94	0.030309142112731932
4	20	70.472	70.78999999999999	70.49	0.03184402513504028
4	25	65.684	65.06	65.39	0.03134999299049378
4	30	56.95	57.17	57.25	0.030428142070770263
4	35	71.186	70.08	70.61	0.0314738597869873
4	40	43.662	44.87	42.80	0.02827137804031372
4	45	66.07	65.42	65.8	0.029986742973327637
4	50	78.802	78.16	78.3	0.02860669207572937
4	55	65.658	65.01	65.45	0.02755639100074768
4	60	66.214	65.98	66.7	0.028879892349243163

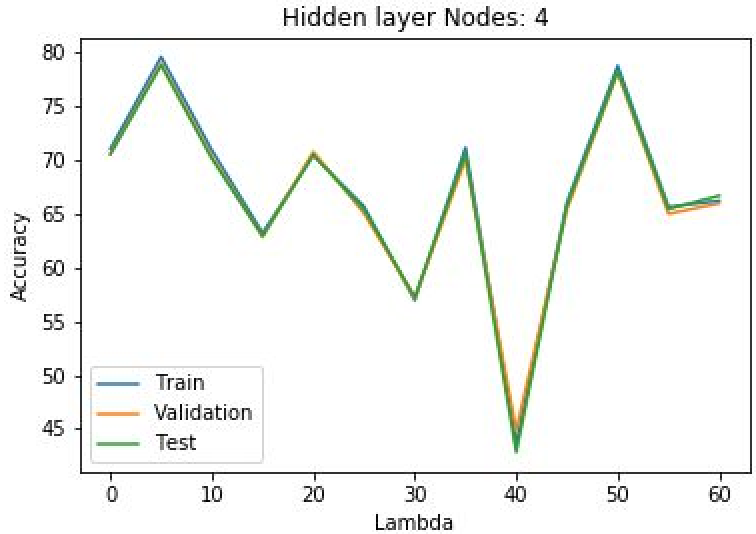


Fig 1.1 : Lambda Value vs Accuracy for 4 hidden nodes

Table 1.2 : Various Lambda values for 8 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
8	0	89.132	88.19	88.99000000000001	0.03160162901878357
8	5	89.38000000000001	88.46000000000001	89.39	0.03075276803970337
8	10	89.414	88.53999999999999	89.13	0.12233197999000549
8	15	89.376	88.35	89.32	0.048823246002197264
8	20	85.848	85.59	85.94000000000001	0.035013916015625
8	25	88.44	87.47	88.71	0.03504850006103516
8	30	86.752	85.86	87.05000000000001	0.043923337936401366
8	35	90.128	89.53999999999999	89.89	0.042266528844833375
8	40	88.92	87.97	89.2	0.040074446201324464
8	45	88.90400000000001	87.96000000000001	88.52	0.03696504092216492
8	50	87.358	86.25	88.01	0.03949703121185303
8	55	88.70599999999999	88.01	88.63	0.03574119186401367
8	60	88.154	87.32	87.83	0.03687878108024597

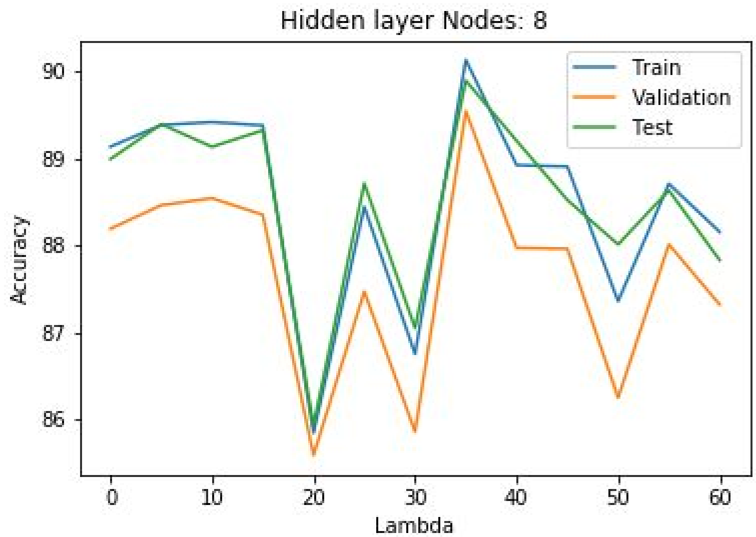


Fig 1.2 : Lambda Value vs Accuracy for 8 hidden nodes

Table 1.3 : Various Lambda values for 12 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
12	0	89.558	88.94999999999999	89.07000000000001	0.03773949599266052
12	5	91.8	90.9	91.86	0.03336129307746887
12	10	90.938	90.18	90.4	0.037192891836166385
12	15	91.824	90.97	91.4	0.03631894588470459
12	20	91.4	90.51	90.98	0.03986833620071411
12	25	91.39	90.46	91.4	0.0377582790851593
12	30	91.97	91.25999999999999	91.85	0.0373305070400238
12	35	91.56	91.21000000000001	91.73	0.03948582696914673
12	40	91.846	90.97	91.75999999999999	0.035465338945388794
12	45	91.69	90.95	91.53999999999999	0.03698310708999634
12	50	90.874	89.91	90.59	0.03943149495124817
12	55	92.132	91.4	91.94	0.03961651110649109
12	60	90.48	89.72	90.34	0.035771684885025024

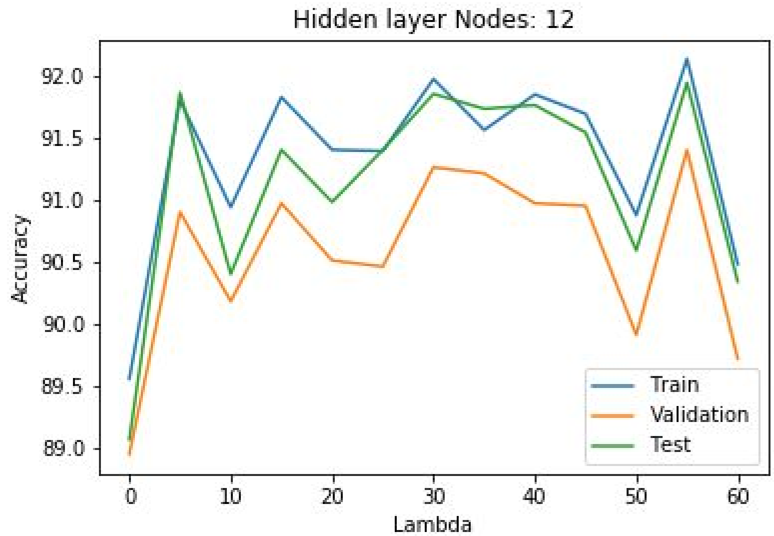


Fig 1.3 : Lambda Value vs Accuracy for 12 hidden nodes

Table 1.4 : Various Lambda values for 16 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
16	0	92.958	92.17	92.64	0.037927417039871215
16	5	92.862	91.91	92.4	0.037777651071548464
16	10	90.066	89.23	89.55	0.0402405641078949
16	15	93.002	92.08	93.08	0.03710684990882874
16	20	92.874	92.36	92.89	0.042200666904449465
16	25	93.496	92.67999999999999	93.08	0.03846309494972229
16	30	92.72	91.84	92.29	0.03892770385742188
16	35	93.01	92.14	92.67	0.04212060618400574
16	40	93.022	92.03	92.92	0.033626872062683104
16	45	92.542	91.67	92.21000000000001	0.035213000297546385
16	50	92.862	92.03	92.54	0.032454561948776245
16	55	91.72200000000001	90.64999999999999	91.49000000000001	0.03572228217124939
16	60	92.36	91.69	92.34	0.034605353355407716

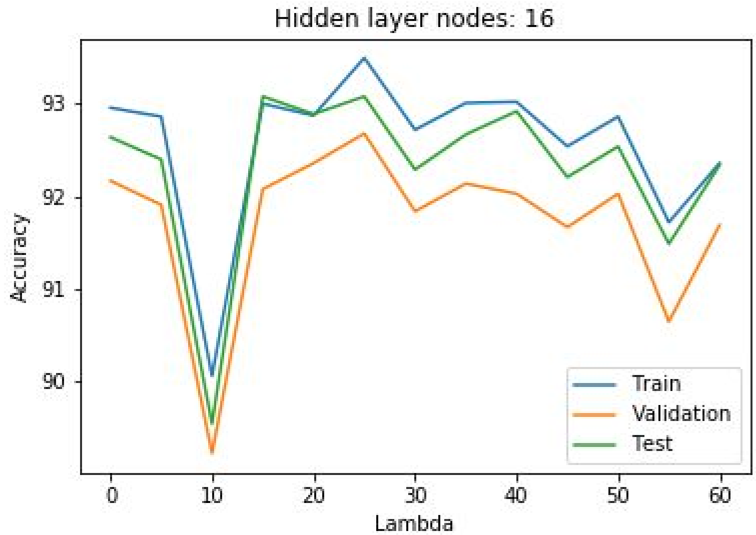


Fig 1.4 : Lambda Value vs Accuracy for 16 hidden nodes

Table 1.5 : Various Lambda values for 20 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
20	0	94.096	93	93.63	0.048918345928192136
20	5	93.7	92.75	93.19	0.037996577978134154
20	10	93.396	92.44	93.08	0.0425710301399231
20	15	93.498	92.67	93.07	0.039843378782272336
20	20	93.33200000000001	92.77	93.25	0.034958586931228636
20	25	93.12	92.13	92.77	0.04497658777236938
20	30	93.28800000000001	92.45	93.14	0.04388237738609314
20	35	93.43599999999999	92.61	92.97999999999999	0.035323023080825806
20	40	93.106	92.34	93.17	0.03706754517555237
20	45	93.108	92.08	92.92	0.037694058895111085
20	50	92.84	91.85	92.78999999999999	0.03643324685096741
20	55	93.04	92.21000000000001	93.02	0.03692982792854309
20	60	92.754	91.99000000000001	92.41	0.031226634979248047

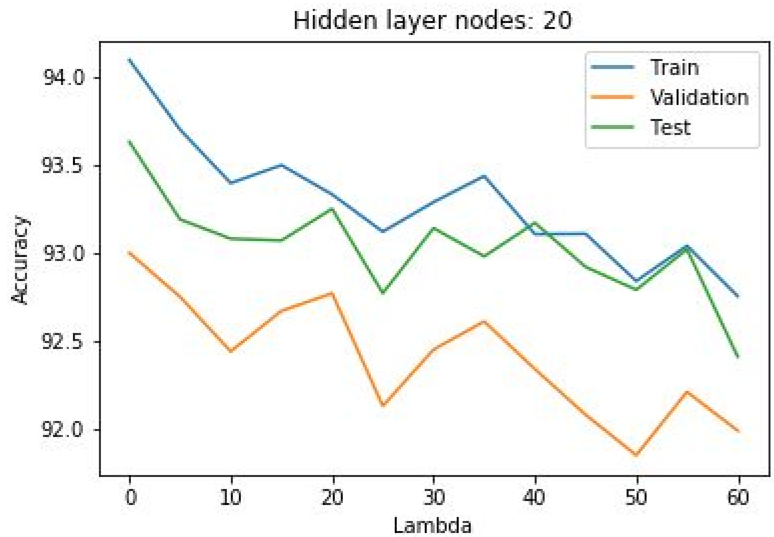


Fig 1.5 : Lambda Value vs Accuracy for 20 hidden nodes

Table 1.6 : Various Lambda values for 20 hidden nodes

Hidden Nodes	Lambda	Training accuracy	Validation accuracy	Testing accuracy	Time taken
50	0	95.036	94.17	94.54	0.045667265176773074
50	5	95.204	94.39999999999999	94.69	0.047291740894317626
50	10	95.406	94.8	94.78	0.04506685066223144
50	15	95.082	94.19999999999999	94.76	0.04197260022163391
50	20	95.086	94.27	94.91000000000001	0.0434632260799408
50	25	95.03200000000001	94.45	94.84	0.04085705590248108
50	30	95.11	94.28999999999999	94.91000000000001	0.04859129405021668
50	35	94.616	93.96	94.17999999999999	0.043591243982315064
50	40	94.622	93.74	94.39999999999999	0.04439829993247986
50	45	94.478	93.58999999999999	94.15	0.044081570148468016
50	50	94.15599999999999	93.5	94.01	0.042773808240890505
50	55	94.014	93.24	93.99	0.0411388738155365
50	60	93.78999999999999	92.97999999999999	93.75	0.05185982704162598

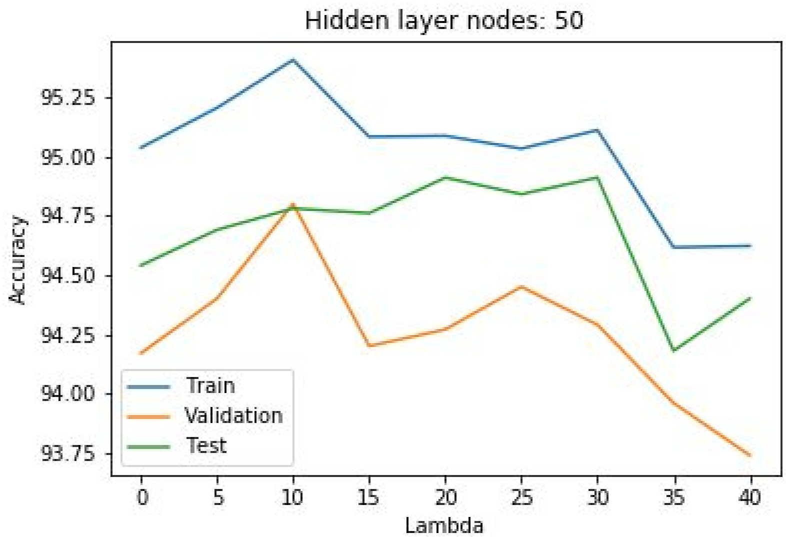


Fig 1.6 : Lambda Value vs Accuracy for 20 hidden nodes

Inference: The graphs provides us with a more comprehensive result and from these graphs we can see that the highest value of validation accuracy for hidden layer nodes and lambda takes the values 50 and 10 respectively. We achieve a accuracy of 94.8% on the validation set.

Furthermore, we plot the graph for the hidden layer nodes with respect to accuracy for lambda value 10 to get a comprehensive understanding of the effect of the number of hidden layer nodes on the performance of the neural network.

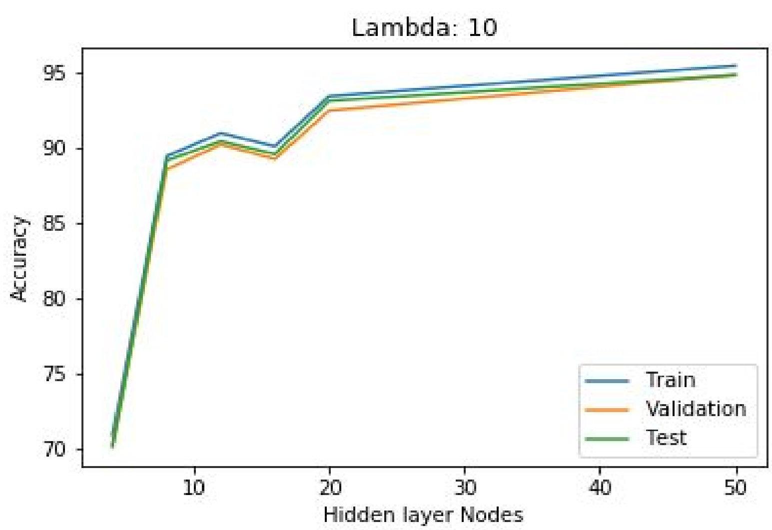
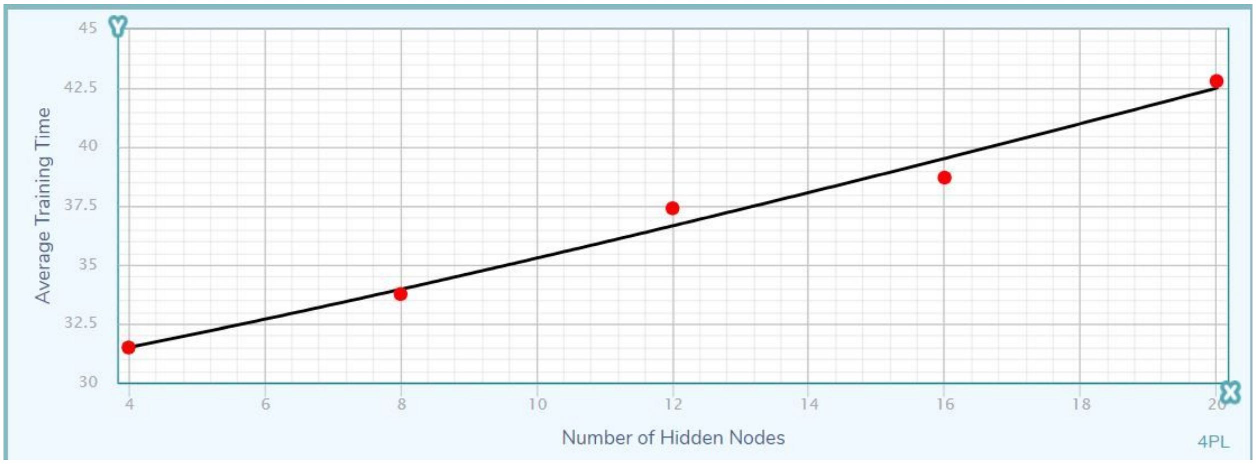


Fig 1.7 : Hidden Layer Nodes vs Accuray for Lambda Value 10

Thus we can observe that the performance of the neural network keeps getting better as we add more number of hidden layer neurons and it performs the best at hidden layer nodes=50.

Average Training time:



r 2

As expected, the average training time over the range of all lambda values increases as a function of the number of hidden nodes.

CelebFaces Attributes Dataset:

The dataset consisted of data for 26407 face images, split into two classes. One class had images in which the individual is wearing glasses and the other class had iimages in which the individual is not wearing glasses. Each image is a 54×44 matrix, flattened into a vector of length 2376.

We used the TensorFlow library to evaluate the accuracy of single hidden layer Neural Network on CelebA dataset to distinguish between two classes - wearing glasses and not wearing glasses.

We got the following values for training , validation and test accuracy:

```
Training set Accuracy:84.4739336492891l%

Validation set Accuracy:83.33958724202627%

Test set Accuracy:84.29220287660864%
```

Comparison of deep neural network and neural network with one hidden layer on the CelebA dataset:

In this section we analyze and compare the results we have acquired from our single layer neural network and deep neural network with multiple hidden layers. We evaluate the performance of our model on the CelebA dataset. The goal of the model is to distinguish between two classes –wearing glasses and not wearing glasses. When we train the using single layer neural network, we attain an accuracy of 84.29% on the test set and 83.33% on the validation set , further the model takes 177.38 seconds to train. Furthermore, we try to predict the same using deep neural network with various hidden layers with an aim to achieve better accuracy. However, we notice that as we increase the number of hidden layers in our models our accuracy drops. We can see this in the following table.

Table 1.7 :

Number of Hidden Layers	Number of Units in each layer	Time taken for Training	Validation Accuracy
One	50	67.87	84.55%
Three	256	88.31	78.75%
Three	512	133	80.35%
Three	1024	133	80.35%
Five	256	91.04	75.09%
Five	512	116.3	79.06%
Five	1024	173	79.03%
Seven	256	100.41	74.48%
Seven	512	128.8	78.5%
Seven	1024	215	80.88%

In general, increasing the number of hidden layers may or may not increase the accuracy, this depends on the complexity of the problem that we are trying to solve. As we increase the number of hidden nodes, we learn more complex features of the data, which initially leads to an increase in accuracy, but the gains in accuracy plateau after a point, leaving us to deal with the problem of overfitting the data which we cause by adding the additional hidden nodes.. We experimented with various number of hidden units in each layer .As the problem statement is not relatively complex , we see that we get the best result from the single layer neural network, from the results we can interpret that this might be the optimal number of neural network for this problem.

We also notice that training time of the deep neural network tends to increase exponentially as we scale the number of hidden layers in the deep learning model. This can be explained by the number of gradients that need to be calculate to adjust the weights for each node,

Convolutional Neural Networks

Introduction

Convolutional Neural Networks use a variation of multilayer perceptrons system which was designed to require minimal preprocessing They are also known as **shift invariant artificial neural networks**, due to their shared-weights based architecture and their complete translation invariance properties.

Why are CNNs good for Image Processing?

When data has a lot of features, like image data, by using a CNN, you can get comparable results with far fewer parameters, which usually results in far less training time. This is because CNNs take advantage of structural information in data. If you simply arrange your features into a matrix and use a CNN, you'll find it does very poorly.

The CNNs have many different filters/kernels which consist of trainable parameters, which can convolve on a given input (the first input is the training image) spatially to create activation maps at each layer. During this process, (through back-propagation) they learn by adjusting those initial values to capture the correct magnitude of a spatial feature on which they are convolving. These high number of filters learn to capture spatial features from the input volumes based on the learned magnitude. Hence they can successfully represent a given image into a highly abstracted representation which is easy for predicting.

Using CNNs on Handwriting Dataset

When we run the Convolutional Neural Network on out Handwriting Dataset, at the end of 10,000 iterations, we get a test set accuracy of 98.6%, which is higher than both our single layer neural network, as expected.

We run the optimizer, in order to update weights, for a total of 10,000 iterations. For the reasons specified above, we reach optimal training accuracy after just 1301 iterations. We reach this accuracy of 100% on the training data, which might seem like overfitting the image data, but the performance on the test sets after 100, 1000, 5000, and 10,000 iteration suggests that the model generalizes fairly well and in fact, doesn't overfit the data. The graph for test set accuracy as a function of number of iterations is shown below

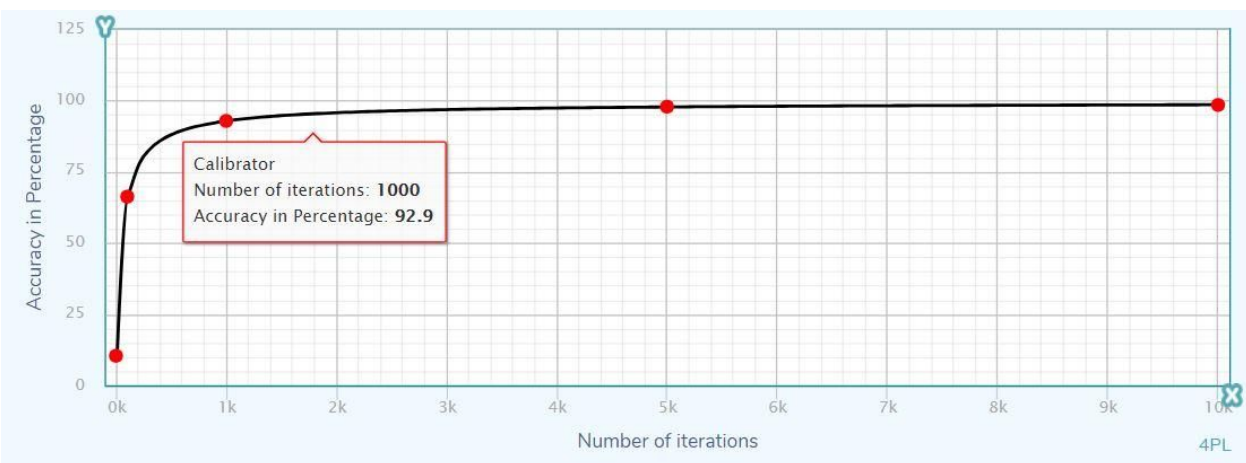


Figure: CNN Number of iterations VS accuracy

First Hundred Iterations

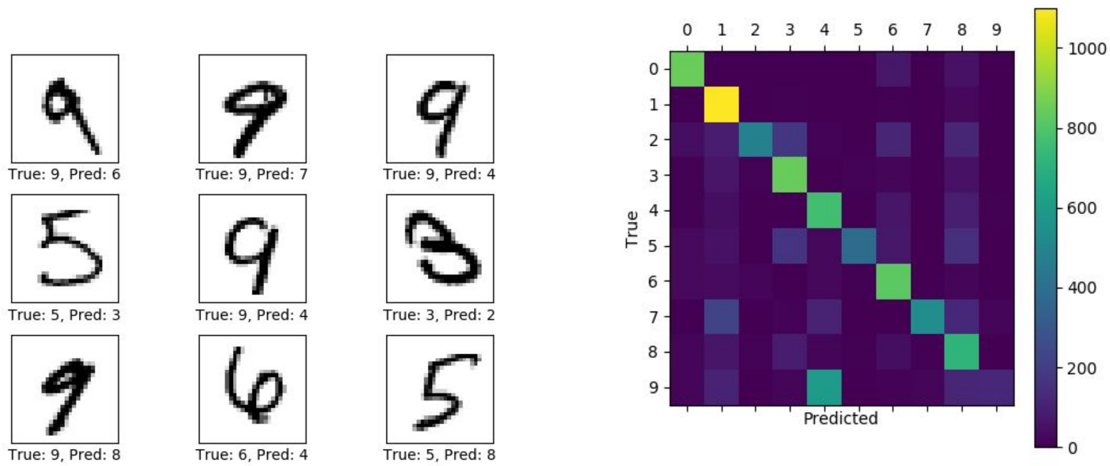


Figure: Example errors and Confusion heat-map

With the first hundred weight updates, we reach an accuracy of 66.3%, which is quite impressive, given that we were getting a test-accuracy of 10.2% with random weight initialization.

We proceed with further iteration to see if this growth in accuracy continues, which we expect it to. We also look at possible overfitting problems that might occur.

First Thousand Iterations

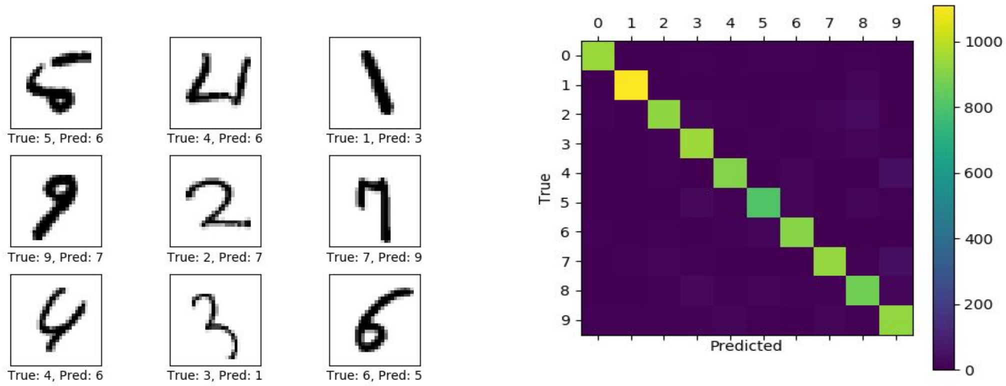


Figure: Example errors and Confusion heat-map

With the first thousand iterations, we get a test accuracy of 92.9 %. Also, during training, we hit a training accuracy of a 100%, which goes to show how fast a CNN can fit the training data. This can be a problem which leads to overfitting in some cases, but our model seems to be generalizing well from the looks of the increasing test set accuracy.

Next, we look at the model after it has been optimized over five thousand iterations.

Five Thousand Iterations

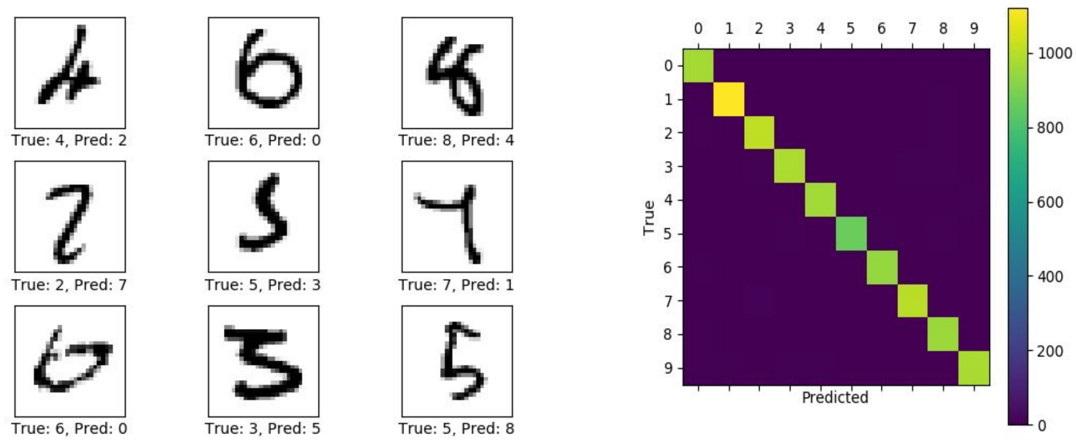


Figure: Example errors and Confusion heat-map

After five thousand iterations, we land on a test set accuracy of 97.2%. We start noticing that the samples on which the model fails to classify accurately are the ones which are the most ambiguous(0/6). This means that it has not yet learned the spatial features that distinguish these numbers.

Hence, we continue running the optimizer for five thousand more iterations.

Ten Thousand Iterations

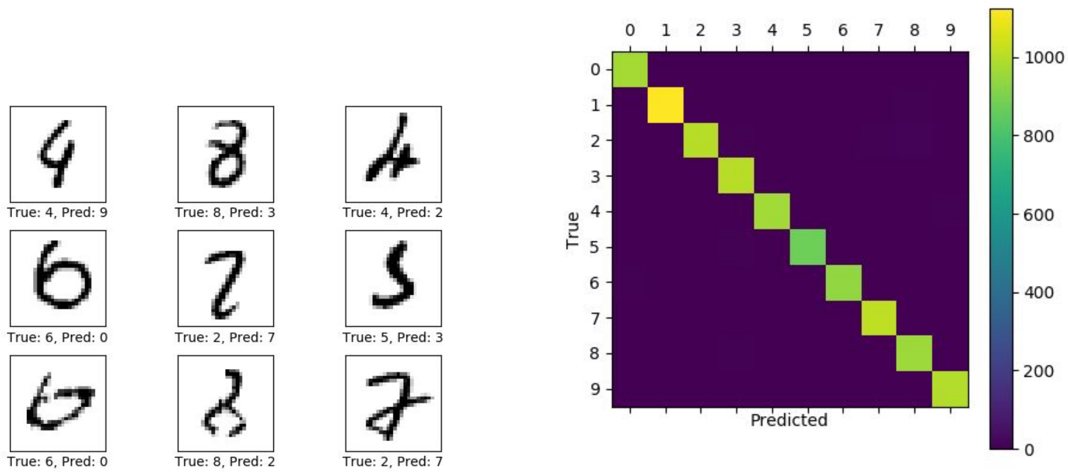


Figure: Example errors and Confusion heat-map

At the end of ten-thousand iteration, we have a test-set accuracy of 98.5%, which is higher than the accuracies obtained from any of the other models that we ran on the handwriting dataset. The reasons for this are listed in the document under ‘Why are CCNs good for Image Processing?’.

Training Time

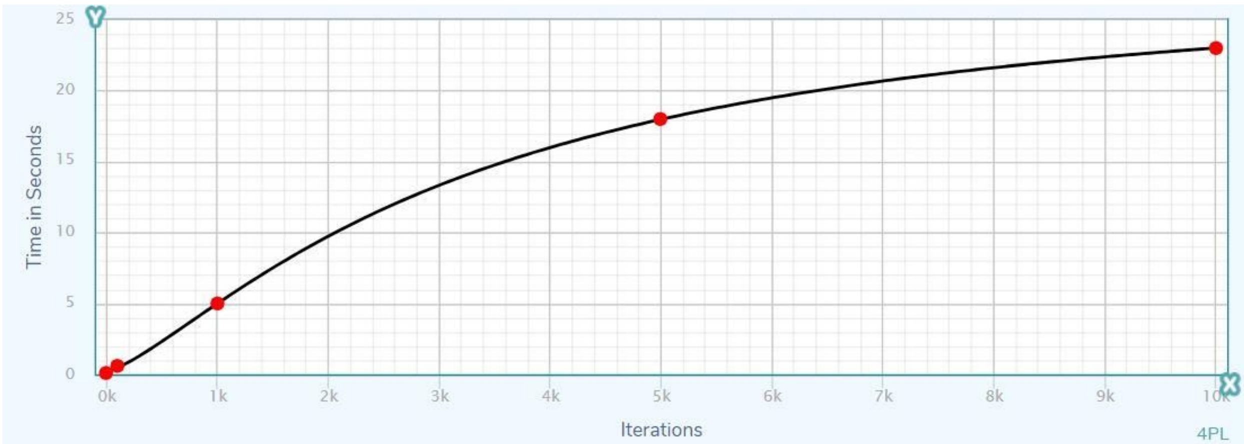


Figure: Training time of CNN w.r.t. iterations

CNNs take advantage of structural information in data which leads to them requiring fewer features to train well. This, combined with the fact that we are using tensorflow with a GPU, resulted in a total training time of just 23 seconds for ten-thousand iterations.

Conclusion

From the results that we have gained from running different types of neural networks on the handwriting data with various parameters, and from the inferences that we have gained from performing these experiments, we can say with confidence that Convolutional Neural Networks are best suited for this task of image classification.