Machine learning and big data processing

OCR for handwriting

Author:
Mayowa Temiloluwa Akande

Teaching Assistants:
Remco Donovan Royen
Yuqing Yang

2020-2021

Contents

Introduction 3
Project description 3
Working OCR System 4
Pre-processing 4
Experimental Evaluation 5
Varying configurations 6
Model Accuracy 12
Conclusion 12

Introduction

This project entails the implementation of Optical Character Recognition for handwriting. The task is to infer the correct label of digit given an image of handwritten digit. An OCR system was built by employing neural network (ANN and CNN) models and using MNIST datasets. The datasets contains 60000 training and 10000 test data of 28×28-pixel grayscale images of handwritten single digits between 0 and 9.

Project description

A very simple ANN was defined where I fed a 2D image of 28*28 pixels into the input layer and the output layer has 10 neurons because I am classifying into 10 different classes (0 - 9) as shown in figure 1. This 2D images of 28*28 pixels were converted into 1D images of 784 input layer with each pixel represented by values 0 – 255. 0 means black and 255 means white. The pixel values were divided by 255 for scaling purpose to have a range of (0 – 1 pixel values). Scaling helps to improve accuracy of the model. After creating the neural network, the model was trained. I compared ANN and CNN to evaluate the accuracy of my handwriting digits. I also varied other configurations to better understand how they affect my predictions.

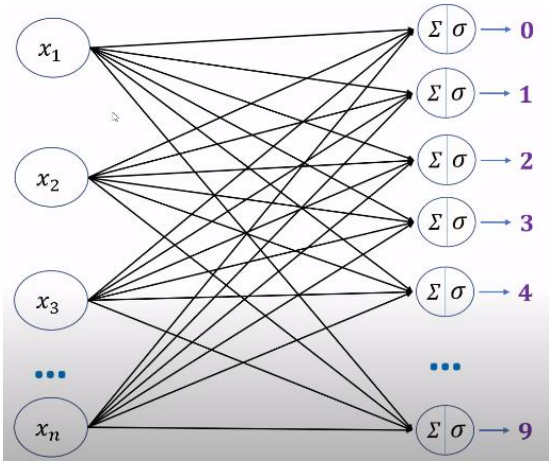

Figure 1: Neural network

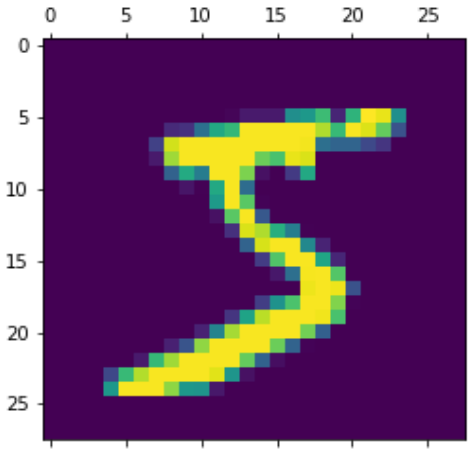

Figure 2: 28*28 pixel image in dataset

An example of a scaled 28*28 pixel image found in the dataset before conversion into 1D image is shown in figure 2.

Hidden layers were added to the network with the intention to improve accuracy. The confusion matrix were also plotted to describe the performance of classification model on a set of test data for which the predicted values are correct. Finally the model accuracies for ANN and CNN were evaluated.

Working OCR System

To achieve a working OCR system, the use of "Microsoft Paint" was employed with some pre-processing activities.

Pre-processing

1. The Input image was created with "Paint" making sure the foreground is white and background is black, similar to the pixel values in the MNIST dataset. The input image is shown in figure 3.
2. The image in Figure 3 has three colour components and a dimension different from 28*28 pixels. It is important to convert RGB pixel values to gray values as shown in figure 4.
3. The image was then resized to a dimension of 28*28 pixel values as shown in figure 5.
4. The image was also normalized by dividing by 255.

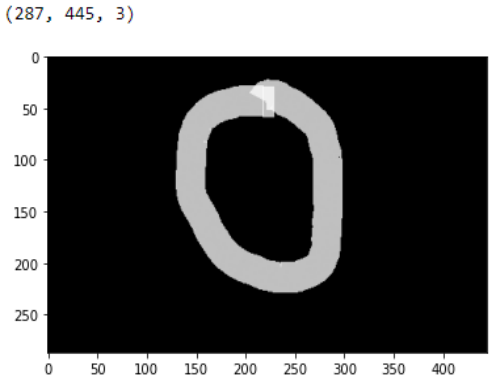

Figure 3: Input Image

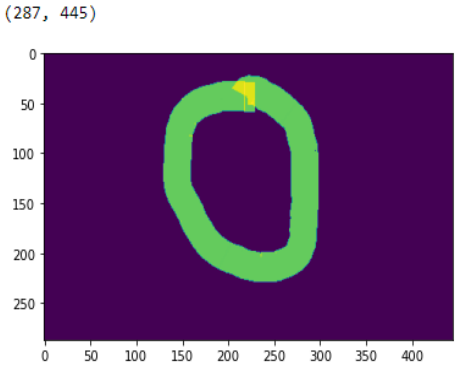

Figure 4: Gray pixel Values

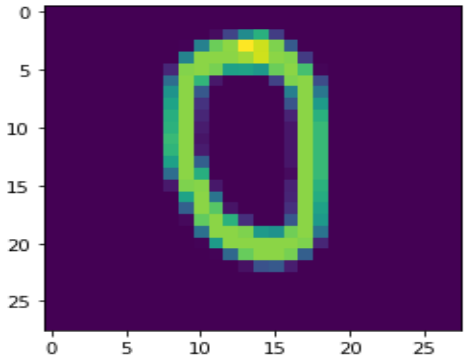

Figure 5: Resized Image

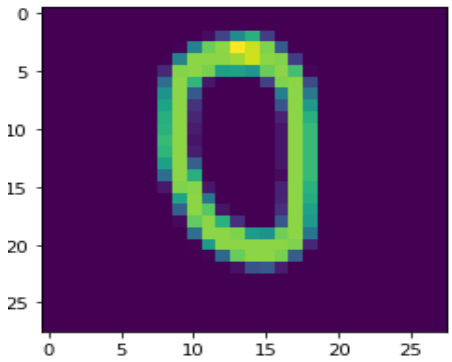

Figure 6: Normalized pixel values

Experimental Evaluation

First of all, the ANN model was trained without normalizing the pixel values in the dataset which gave a training accuracy of 88.65% and a test accuracy of 89.44% after 5 epochs. This was carried out without the addition of hidden layers. The ANN has 784 input neurons and 10 output neurons. The confusion matrix for this configuration is as shown in figure 7. From this figure it can be inferred that the model made a correct prediction of 3417 and an incorrect prediction of 6583.

From figure 7, it can be observed that the confusion matrix gave a very poor result. For an ideal result, the numbers on the diagonal should be very high and numbers outside the diagonal should be 0. For example, when the test data was 9, the model never predicted the correct value of 9, also when the test data was 8, the model predicted a correct value of 8 only once.

The next step was to improve the accuracy of my result by varying configurations.

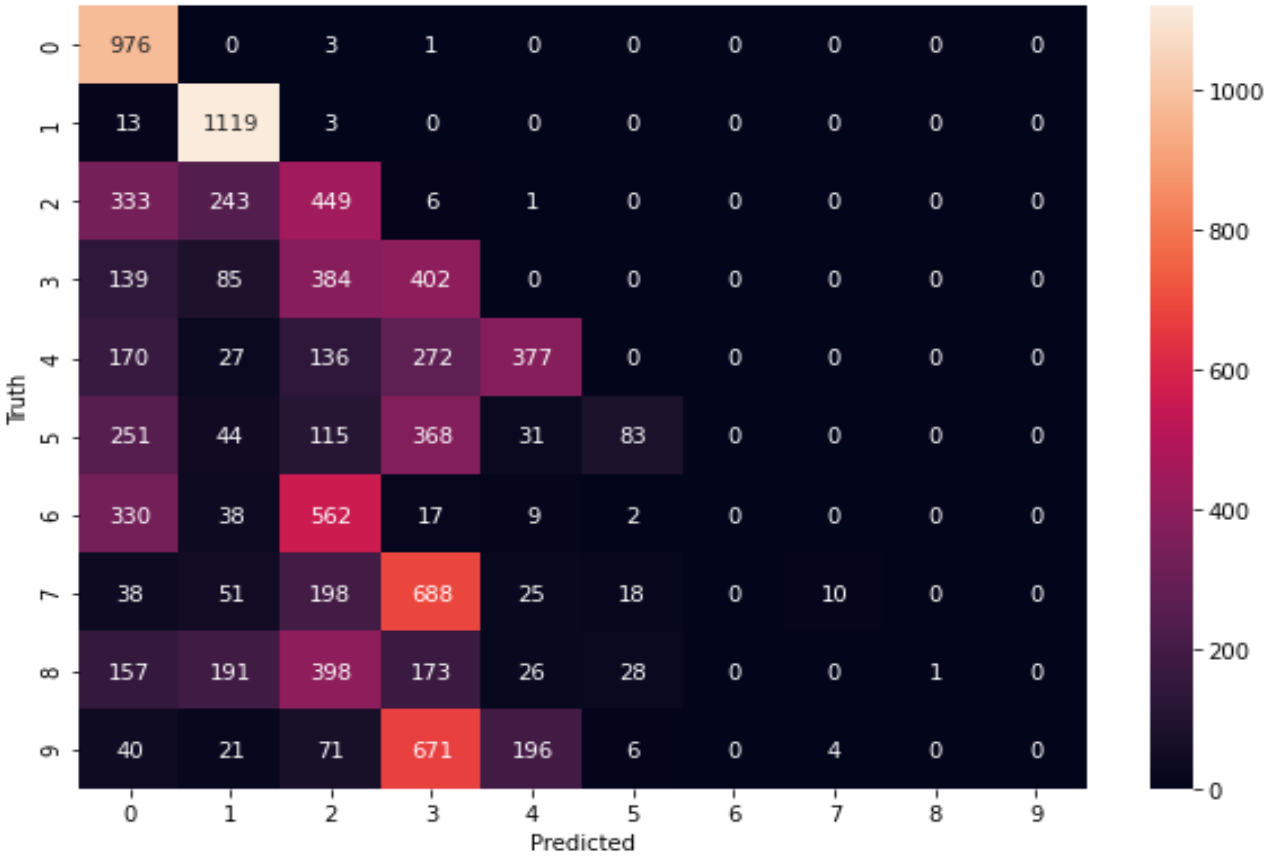

Figure 7: Confusion Matrix (ANN); No normalization, No hidden layer, 5 Epochs

Varying configurations

After normalizing the datasets, I observed some improvements in the training accuracy result of 92.51% and test accuracy result of 92.44. The confusion matrix of figure 8 also has some improvements as I have higher numbers across the diagonal and less numbers outside the diagonal when compared with figure 7.

From figure 8, 9244 correct predictions were made and 756 incorrect predictions were made which results in a significant improvement due to normalization.

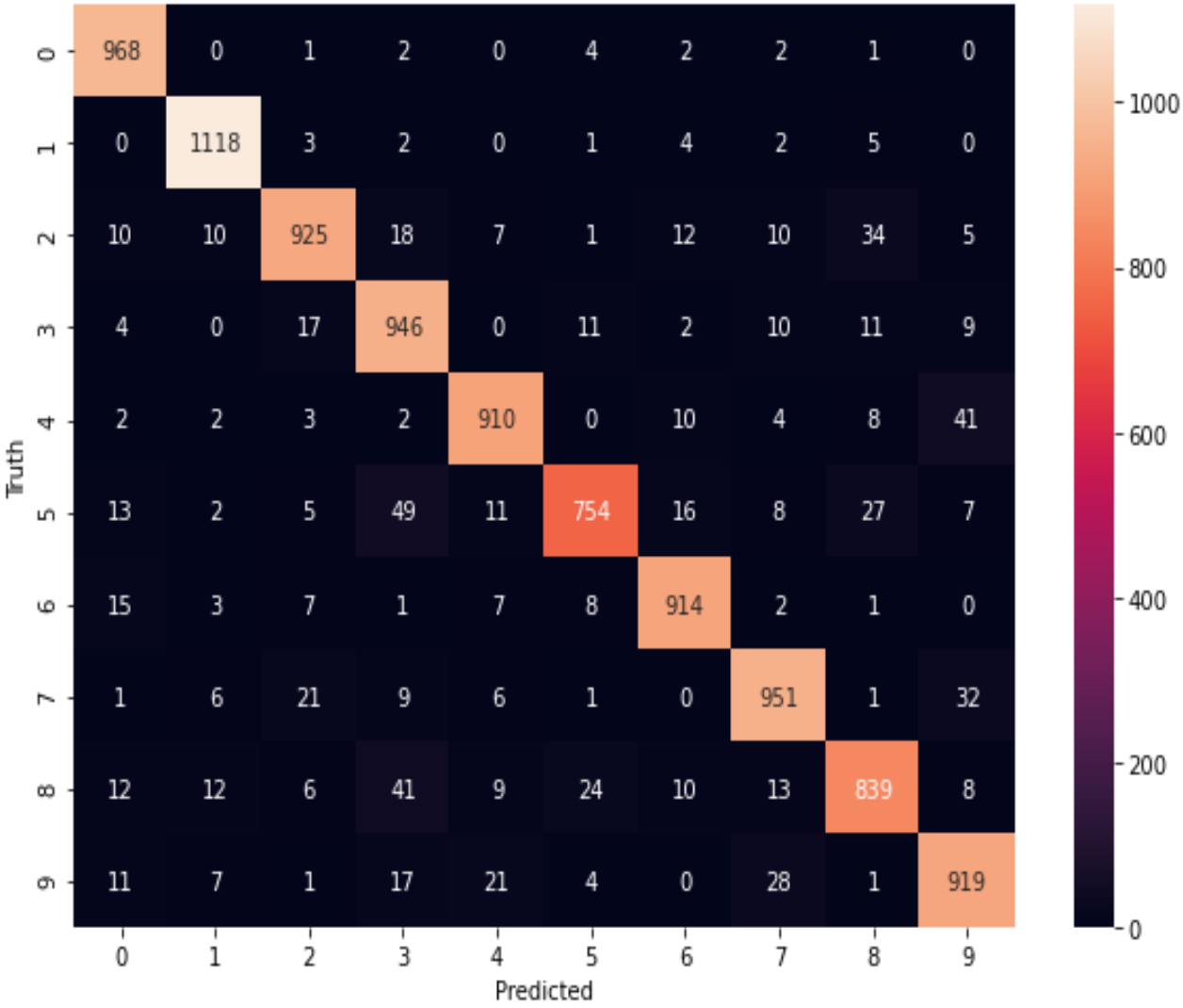

Figure 8: Confusion Matrix(ANN); Normalization, No hidden layer, 5 Epochs

Next, I added one hidden layer to the ANN to see further improvements. As a result, I observed some improvements in the training accuracy result of 97.96% and test accuracy result of 97.22%. The confusion matrix of figure 9 also shows some improvements as I have some higher numbers across the diagonal and some lesser numbers outside the diagonal when compared with figure 8.

9722 correct predictions were made and 278 incorrect predictions were made. This is an improvement over a network without hidden layers.

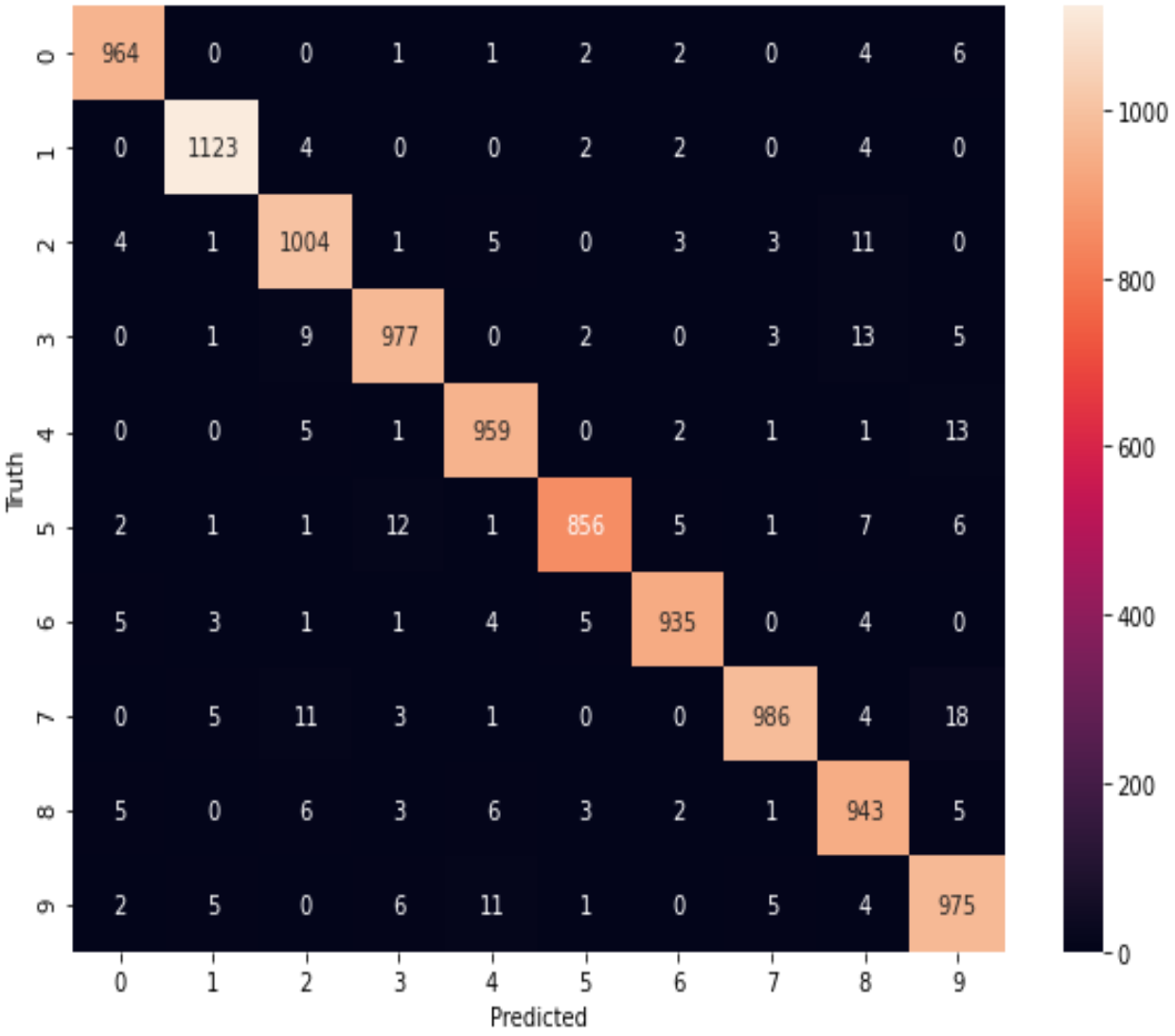

Figure 9: Confusion Matrix(ANN); Normalization, 1 hidden layer, 5 Epochs

To see more improvements, I made use of 1 hidden layer and 20 epochs. I observed that the training accuracy gave an improved result of 99.65% and test accuracy result of 97.18%

From the confusion matrix of figure 10, 9720 correct predictions were made and 280 incorrect predictions were made. Therefore, increasing the number of epochs increases training accuracy.

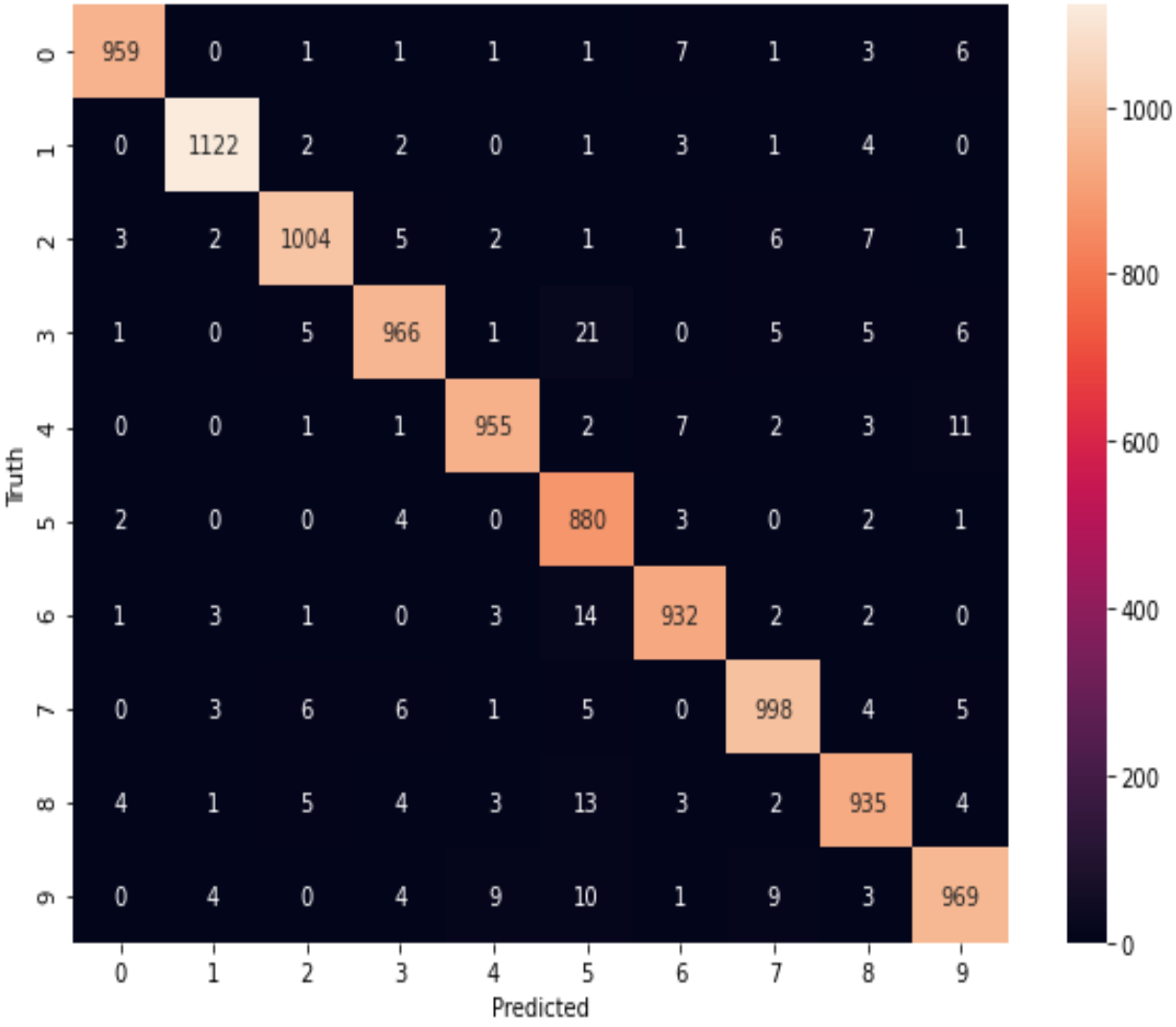

Figure 10: Confusion Matrix(ANN); Normalization, 1 hidden layer, 20 Epochs

With the hope to see more improvements, I made use of 2 hidden layers and 20 epochs. I observed a training accuracy result of 99.43% and test accuracy result to 97.21%. The confusion matrix of figure 11 also shows a correct prediction of 9700 and incorrect prediction of 300. Making use of 2 hidden layers did not show any significant improvement so I thought of varying the numbers of neurons in each hidden layer with the hope to get better results.

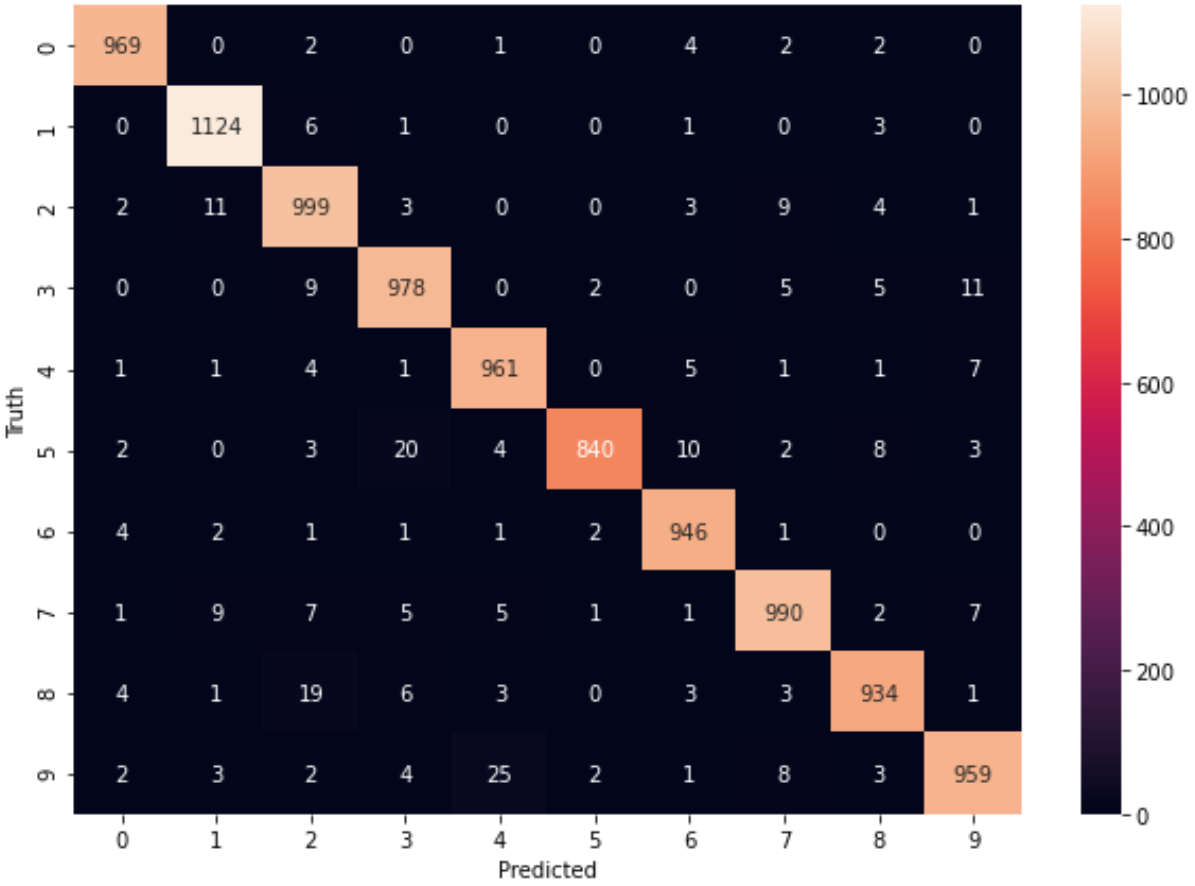

Figure 11: Confusion Matrix(ANN); Normalization, 2 hidden layers, 20 Epochs

Input	Number of Correct Predictions	Number of Incorrect Predictions	Total
0	969	11	980
1	1124	11	1135
2	999	33	1032
3	978	32	1010
4	961	21	982
5	840	52	892
6	946	12	958
7	990	38	1028
8	934	40	974
9	959	50	1009
Total:	9700	300	10000

Table 1: Normalization, 2 hidden layers, 20 Epochs

Table 2 and 3 shows the result of varying configurations of the number of neurons at each stage of hidden layers

No of hidden layers	Epoch	Training Accuracy(%)	Test Accuracy(%)	No of incorrect elements in confusion matrix	Correct prediction of the input image "2" created by paint
0	5	92.56	92.48	752	Yes
0	20	93.36	92.61	739	Yes
1	5	97.89	97.32	268	Yes
1	20	99.66	97.47	256	Yes
1	50	99.94	97.38	340	No
2	5	98.05	97.05	295	Yes
2	20	99.56	97.00	474	Yes

Table 2: Hidden layers of 64 neurons and 32 neurons

The distribution of neurons in the hidden layers also affected the accuracy and confusion matrix.

No of hidden layers	No of neurons in hidden layers	Training Accuracy(%)	Test Accuracy(%)	No of incorrect elements in confusion matrix	Correct prediction of the input image "2" created by paint
1	100	99.81	97.83	213	yes
1	200	99.81	98.12	202	yes
1	400	99.83	98.00	204	yes
2	400, 200	99.71	98.22	178	yes
2	400, 64	99.78	98.02	235	yes
2	100, 32	99.60	97.59	340	yes
3	400, 200, 100	99.69	98.05	195	yes
4	400, 200, 100, 50	99.69	98.34	166	yes
5	400, 200, 100, 50, 25	99.70	98.29	171	yes

Table 3: For epoch = 20

To further see improvements in the predictions of my handwritten digits, I employed the use of CNN which makes use of Convolution layer for feature extraction, Pooling layer to reduce the image size and to retain more valuable pixels. The CNN gave a pleasing result as seen in figure 12 and table 4. With 5 epochs I achieved an accuracy of 98.68% and 196 incorrect predictions from the confusion matrix.

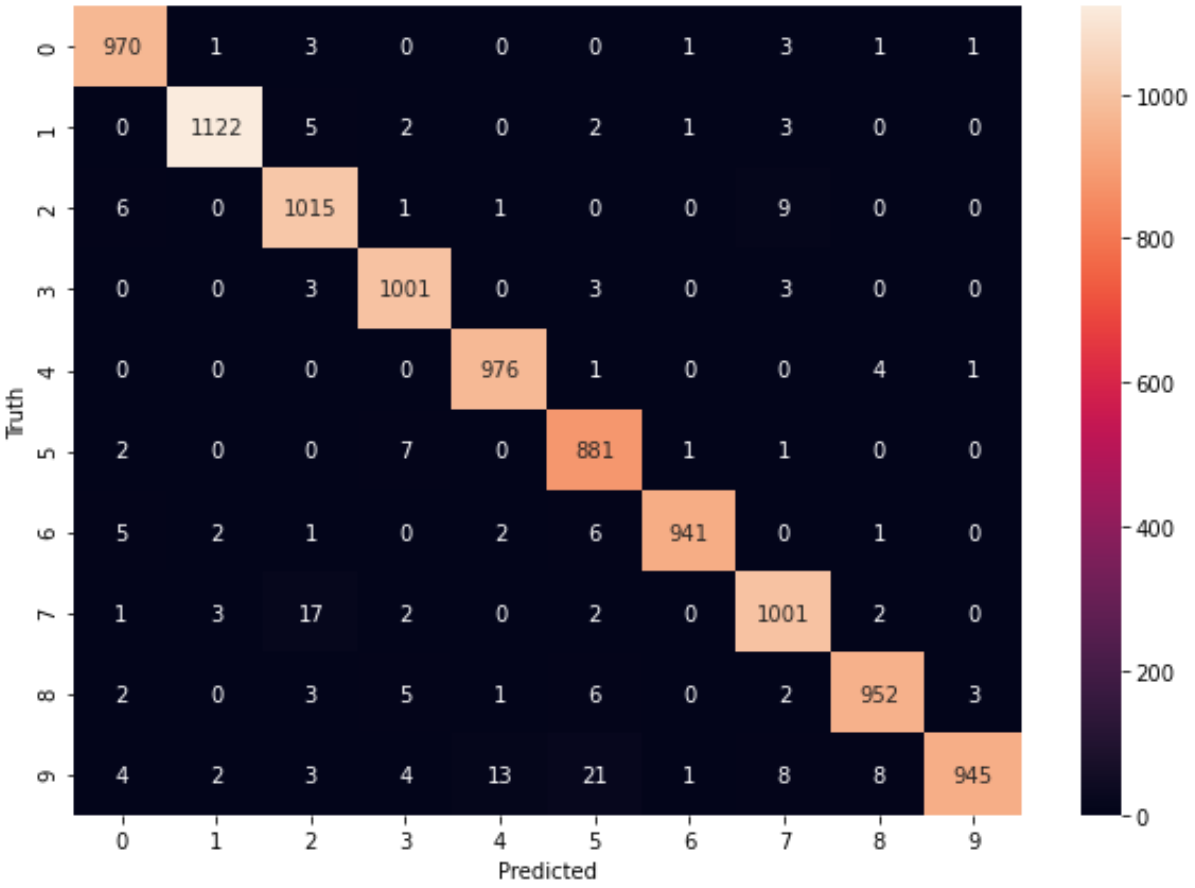

Figure 12: Confusion Matrix(CNN); Normalization, 5 Epochs

The CNN was able to predict correctly more of my hand written digits because of its ability to extract features from images.

Input Image	ANN Correct Predictions	CNN Correct Predictions
0	Yes	Yes
1	No	Yes
2	Yes	Yes
3	Yes	Yes
4	No	Yes
5	No	Yes
6	Yes	Yes
7	No	Yes
8	No	Yes
9	No	No
No of correct predictions	4	9

Table 4: Comparing correct predictions made by ANN and CNN

Model Accuracy

After carefully conducting various experiments, I was able to come up with an optimal graph model for both ANN and CNN. As expected CNN outperformed ANN. With the CNN model, I was able to achieve better predictions of my handwritten images because CNN was able to capture the spatial features (pixel relationships and arrangement) from my input image.

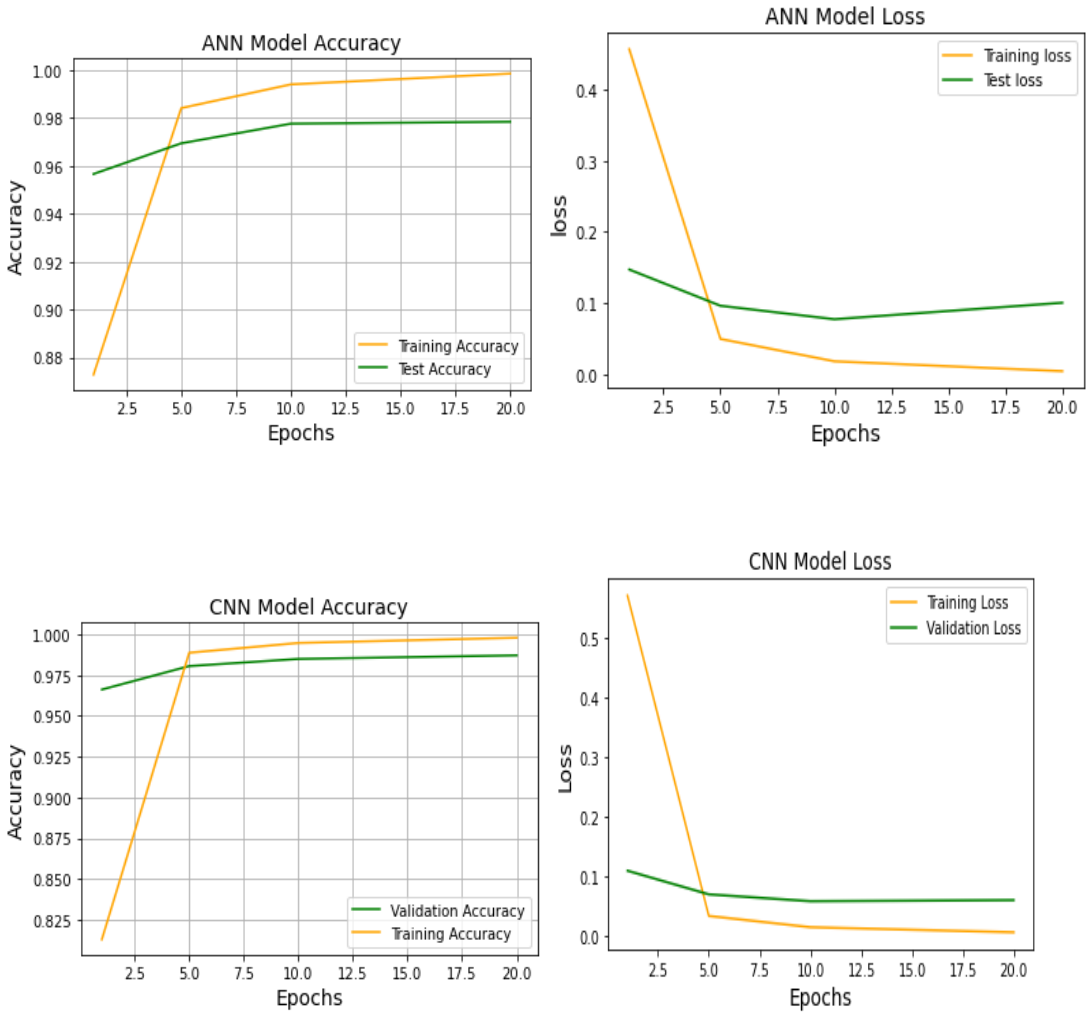

Figure 13: Model Accuracies

Conclusion

The Optical Character Recognition for handwriting was successfully implemented. I was able to evaluate both the accuracies of using ANN and CNN. Making use of CNN, I was able to achieve better predictions of my handwritten images because CNN was able to consider the context information in the small neighbourhood. This feature made it important to achieve a better image prediction of the input image by learning fewer parameters and reducing the chances of overfitting.