The code is mainly referred from https://youtu.be/iqQgED9vV7k

HOCHSCHULE SRH HEIDELBERG

SRH University Heidelberg

Task 1: Computer Vision

Handwritten Digit Classification based on a Convolutional Neural Network

Name:	Shrey Kothavade
Matriculation Number:	11018044
Email Address:	11018044@stud.hochschule-heidelberg.de
Supervisor:	Prof. Dr.-Ing. Milan Gnjatovic
Project Begin Date:	23rd November 2022
Project End Date:	7th December 2022

The code is mainly referred from https://youtu.be/iqQgED9vV7k

Abstract:

Currently, handwritten number images may be classified in a range of ways using machine learning and deep learning models. Due to the uncertainty in learning techniques, researchers have focused their attention to handwritten digits recognition (HDR) using machine learning. By choosing the proper parameters and feature design, a number of researchers have made substantial efforts to enhance the recognition process to date. However, there is always space for advancement in traditional approaches. The most often used deep neural network is the convolutional neural network (CNN), which is used to study picture categorization, object identification, face recognition, etc. Robust CNN architecture is employed for feature extraction and classification in the HDR job. In this report, we used MNIST dataset, consisting of data derived from 70000 handwritten number images in CSV format.

I. Introduction:

The era of artificial intelligence has come with the rapid technological and scientific advancements over the past ten years, and computer vision is now widely employed in many aspects of our daily life. A multidisciplinary research topic called computer vision studies how advanced data can be extracted by computers from digital pictures or films. From an engineering standpoint, it aims to comprehend and automate operations that the human visual system is capable of performing.

Since the 1980s, handwritten digit recognition (HDR) has been employed in a variety of research fields, including online recognition, offline recognition, while processing bank checks, interpreting postal addresses, etc. [1-2]

Digits recognition is regarded as an important domain since it is used in so many different applications for handwritten character recognition. Convolutional neural networks (CNN) are the most popular deep learning architecture for the job of image recognition. Different convolution layers with fully linked convolutions are included in CNN.

II. Convolutional Neural Network

CNNs are deep learning neural networks specifically created for analyzing organized arrays of input, like photographs. The state-of-the-art for many visual applications, such as image classification, convolutional neural networks are widely employed in computer vision.

The patterns in the input picture, such as lines, gradients, circles, or even eyes and faces, are extremely well recognized by convolutional neural networks. Convolutional neural networks are extremely effective for computer vision because of this quality. Convolutional neural networks do not require any preparation and may function immediately on a raw picture, in contrast to older computer vision methods.

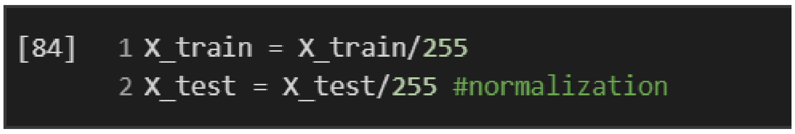
A convolutional neural network's architecture is a multi-layered feed-forward neural network created by sequentially stacking several hidden layers on top of one another. Convolutional neural networks may learn hierarchical features because of their sequential construction.

Convolutional layers are frequently followed by activation layers, some of which are then followed by pooling layers, as the hidden layers. [3]

The code is mainly referred from https://youtu.be/iqQgED9vV7k

III. Pre-Processing of the data:

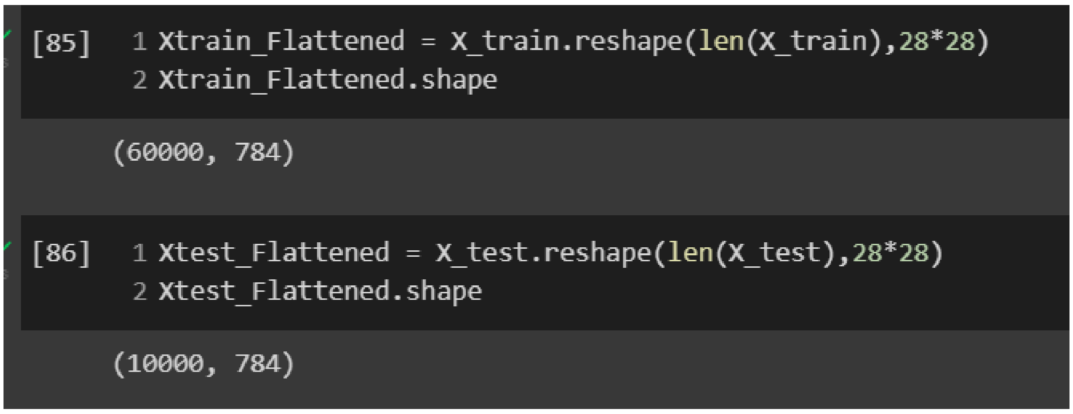
In the Pre-processing of the data, The X_train and X_test is normalized first. Model accuracy is significantly increased by normalization. Each variable is given equal weights/importance so that no one variable may influence model performance only because they have larger numbers. [4] As we know, highest pixel is 255, therefore we divide the X_train and X_test by 255.



```
[84]   X_train = X_train/255
       X_test = X_test/255 #normalization
```

Image1: Normalization of the data

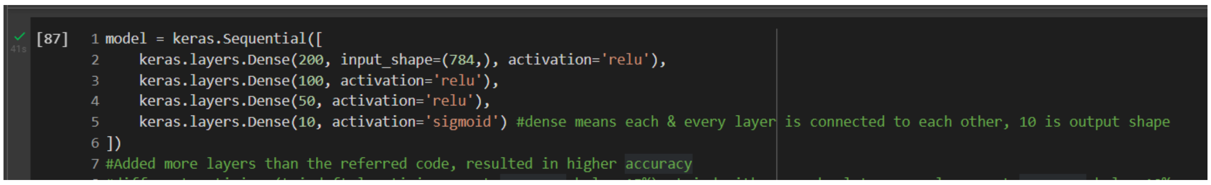
The data fetched is in 2-dimensional data i.e., 28*28. To convert this data in single dimensional data or in a single array we need to flattened the data. With the reshape command, we flattened the data.



```
[85]   Xtrain_Flattened = X_train.reshape(len(X_train),28*28)
       Xtrain_Flattened.shape

     (60000, 784)

[86]   Xtest_Flattened = X_test.reshape(len(X_test),28*28)
       Xtest_Flattened.shape

     (10000, 784)
```

Image2: Flattened X_train and X_test

Now, as we know, CNN is a multi-layered feed-forward neural network. With the help of "keras.Sequential" we create layers. In this we use dense function, which means each & every layer is connected to each other.



```
[87]   model = keras.Sequential([
           keras.layers.Dense(200, input_shape=(784,), activation='relu'),
           keras.layers.Dense(100, activation='relu'),
           keras.layers.Dense(50, activation='relu'),
           keras.layers.Dense(10, activation='sigmoid') #dense means each & every layer is connected to each other, 10 is output shape
       ])
       #Added more layers than the referred code, resulted in higher accuracy
```

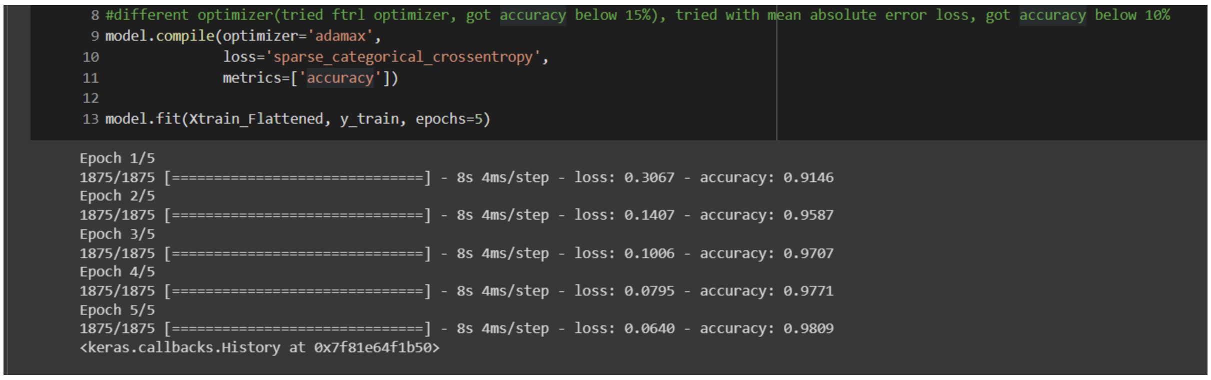
Image3: Initializing multiple layers of CNN

From the image, we can see, we used activation function Relu and Sigmoid. In artificial neural networks, an activation function is a function that outputs a smaller value for small inputs and a higher value if its inputs are greater than a threshold. The activation function "fires" if the inputs are big enough; otherwise, nothing changes.[5] The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. Sigmoid function takes any real value as input and outputs values in the range of 0 to 1. [6-7]

The code is mainly referred from https://youtu.be/iqQgED9vV7k

IV. Training Data:

To minimize the error and improve the accuracy of predictions and classification, Optimization is needed. The learning and development of algorithms would not be possible without the optimization process. So, function optimization is a fundamental component of machine learning. [8]



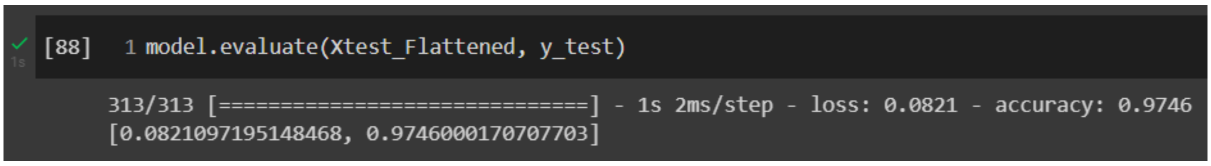
```
#different optimizer(tried ftrl optimizer, got accuracy below 15%), tried with mean absolute error loss, got accuracy below 10%
model.compile(optimizer='adamax',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(Xtrain_Flattened, y_train, epochs=5)

Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3067 - accuracy: 0.9146
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1407 - accuracy: 0.9587
Epoch 3/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1006 - accuracy: 0.9707
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0795 - accuracy: 0.9771
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0640 - accuracy: 0.9809
<keras.callbacks.History at 0x7f81e64f1b50>
```

Image4: Compilation and Training of Dataset

We use compile function to define the loss function, optimizer and metrics for the model. This compiled data is now trained with ".fit" command. Here, I am using epoch function to reiterate the data training on the same data multiple times. It improves the accuracy of the model.

V. Results:

Now this model is evaluated on flattened dataset of X_test to get the accuracy of the model.

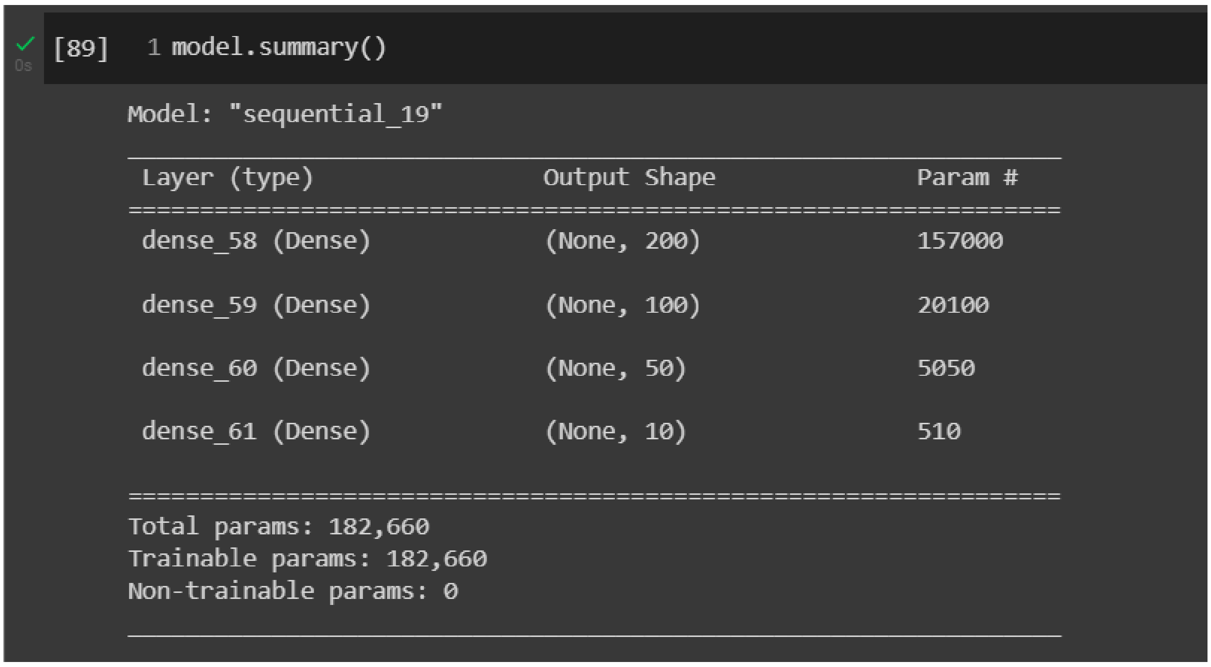


```
[88]  model.evaluate(Xtest_Flattened, y_test)

313/313 [==============================] - 1s 2ms/step - loss: 0.0821 - accuracy: 0.9746
[0.0821097195148468, 0.9746000170707703]
```

Image5: Accuracy of the model

The accuracy of the model is 97.46%

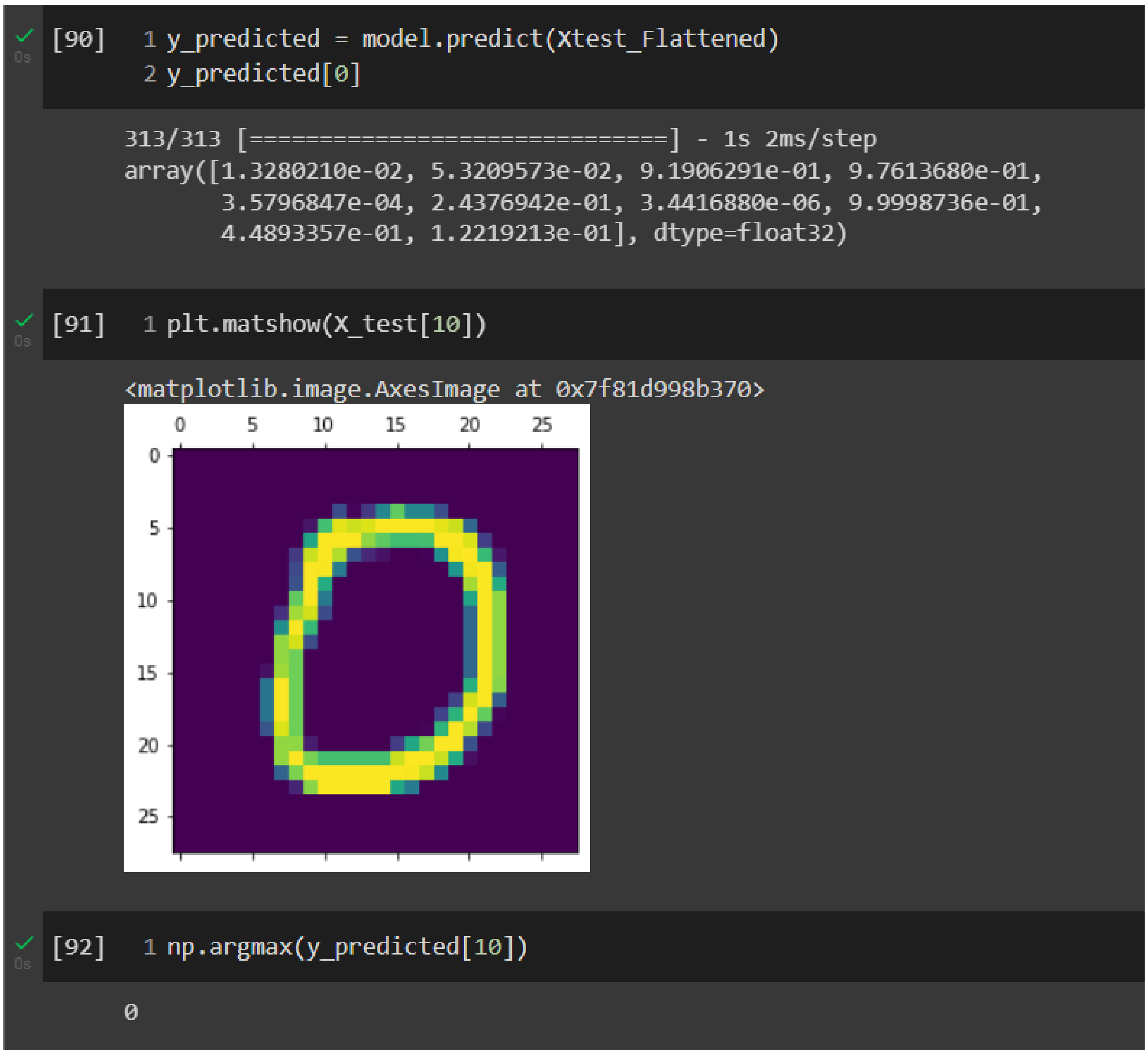
The summary of the model is shown in the code.



```
[89]  model.summary()

Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_58 (Dense)            (None, 200)               157000

 dense_59 (Dense)            (None, 100)               20100

 dense_60 (Dense)            (None, 50)                5050

 dense_61 (Dense)            (None, 10)                510

=================================================================
Total params: 182,660
Trainable params: 182,660
Non-trainable params: 0
_________________________________________________________________
```

Image6: Summary of the Model

In CNN, every digit or image is compared with the trained dataset of the digit images. The probability of each digit is stored and the highest probability is predicted as what image shows. To choose the highest probability, we use NumPy tools “np.argmax”.

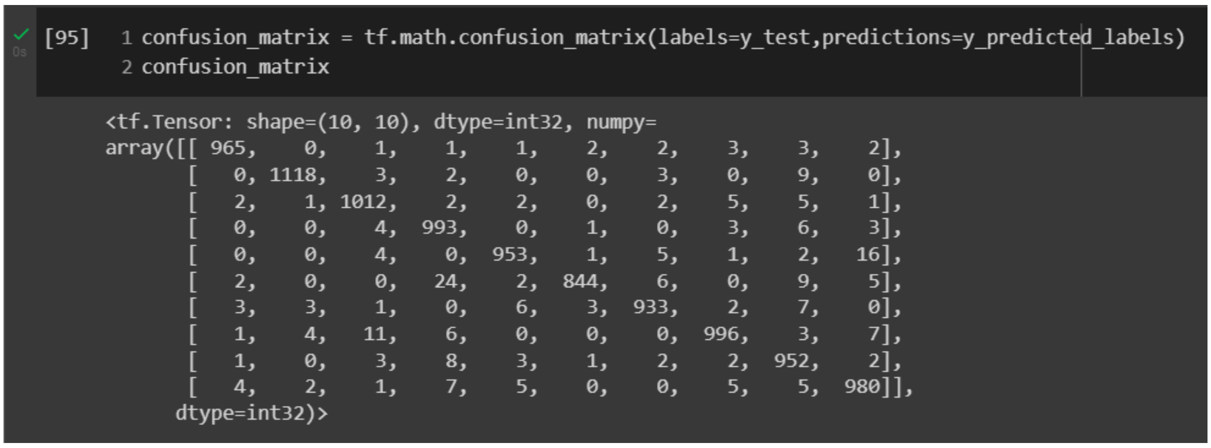


```
[90]  y_predicted = model.predict(Xtest_Flattened)
      y_predicted[0]

313/313 [==============================] - 1s 2ms/step
array([1.3280210e-02, 5.3209573e-02, 9.1906291e-01, 9.7613680e-01,
       3.5796847e-04, 2.4376942e-01, 3.4416880e-06, 9.9998736e-01,
       4.4893357e-01, 1.2219213e-01], dtype=float32)

[91]  plt.matshow(X_test[10])

<matplotlib.image.AxesImage at 0x7f81d998b370>
```

```
[92]  np.argmax(y_predicted[10])

0
```

Image7: use of np.argmax command

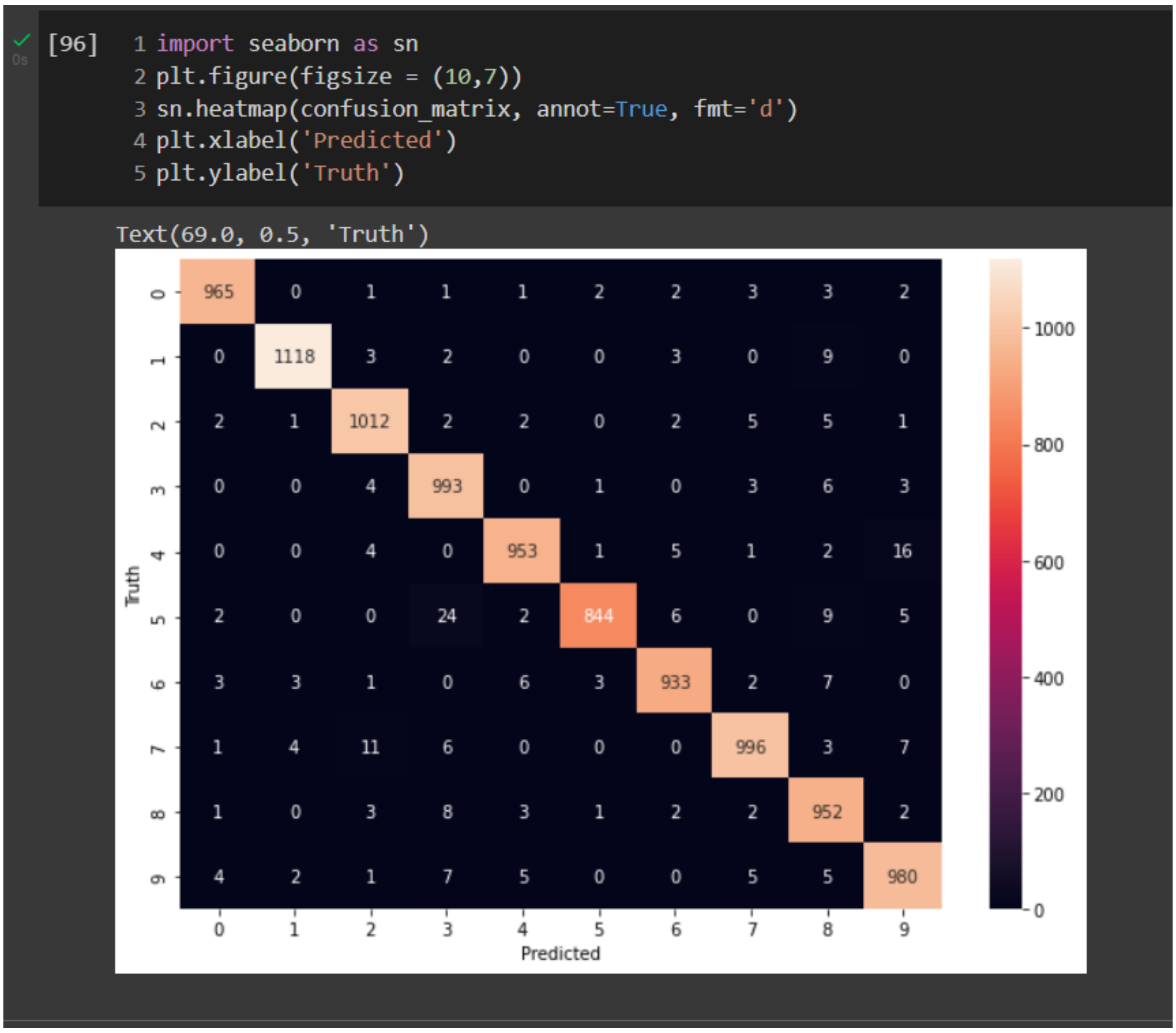
Confusion matrix is generated using tensorflow’s math function in the code.



```
[95]  confusion_matrix = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
      confusion_matrix

<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 965,    0,    1,    1,    1,    2,    2,    3,    3,    2],
       [   0, 1118,    3,    2,    0,    0,    3,    0,    9,    0],
       [   2,    1, 1012,    2,    2,    0,    2,    5,    5,    1],
       [   0,    0,    4,  993,    0,    1,    0,    3,    6,    3],
       [   0,    0,    4,    0,  953,    1,    5,    1,    2,   16],
       [   2,    0,    0,   24,    2,  844,    6,    0,    9,    5],
       [   3,    3,    1,    0,    6,    3,  933,    2,    7,    0],
       [   1,    4,   11,    6,    0,    0,    0,  996,    3,    7],
       [   1,    0,    3,    8,    3,    1,    2,    2,  952,    2],
       [   4,    2,    1,    7,    5,    0,    0,    5,    5,  980]],
      dtype=int32)>
```

Image8: Confusion Matrix1

The code is mainly referred from https://youtu.be/iqQgED9vV7k



```
[96]
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(confusion_matrix, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```

	0	1	2	3	4	5	6	7	8	9
0	965	0	1	1	1	2	2	3	3	2
1	0	1118	3	2	0	0	3	0	9	0
2	2	1	1012	2	2	0	2	5	5	1
3	0	0	4	993	0	1	0	3	6	3
4	0	0	4	0	953	1	5	1	2	16
5	2	0	0	24	2	844	6	0	9	5
6	3	3	1	0	6	3	933	2	7	0
7	1	4	11	6	0	0	0	996	3	7
8	1	0	3	8	3	1	2	2	952	2
9	4	2	1	7	5	0	0	5	5	980

Image9: Confusion Matrix2

VI. Challenges faced during project

1. Similar to ML_Task1, I faced issues in loading the MNIST dataset in the code with ‘pd.read_csv’ command. As the dataset was missing heading, I was not able to load the data properly.
2. At the start, I did not normalize the data. Due to which the accuracy was below 90%. But after normalization the accuracy crossed 97%
3. I was trying to find out suitable loss function than “sparse_categorical_crossentropy” but was not able to find suitable alternative.

VII. Conclusion:

Handwritten Digit Classification has been successfully implemented with Convolutional Neural Network on the Google colab platform using Python language. Tensorflow has been used for the execution of the model. The accuracy of the model is 97.46%.

The code is mainly referred from https://youtu.be/iqQgED9vV7k

VIII. References:

[1] S. Ali, Z. Sakhawat, T. Mahmood, M. S. Aslam, Z. Shaukat, and S. Sahiba, “A robust CNN model for handwritten digits recognition and classification,” 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), Aug. 2020, doi: 10.1109/aeeca49918.2020.9213530.

[2] Shah, F.T. and K. Yousaf. Handwritten Digit Recognition Using Image Processing and Neural Networks. in World Congress on Engineering. 2007.

[3] https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

[4] https://towardsdatascience.com/data-normalization-in-machine-learning-395fdec69d02#:~:text=The%20short%20answer%20is%20%E2%80%94%20it,because%20they%20are%20bigger%20numbers.

[5] https://deepai.org/machine-learning-glossary-and-terms/activation-function

[6] https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=Sigmoid%20%2F%20Logistic%20Activation%20Function,to%200.0%2C%20as%20shown%20below.

[7] https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[8] https://www.seldon.io/machine-learning-optimisation#:~:text=Without%20the%20process%20of%20optimisation,to%20achieving%20an%20accurate%20model

[9] https://youtu.be/iqQgED9vV7k

[10] https://youtu.be/BEx0KLyrIFQ