A Project Activity Report
Submitted for Machine Learning (UML501)

Deep Learning:

Multiple Handwritten Digit Recognition with Canvas GUI

Using Keras ,CNN and Sequential Model

Submitted by:

Arpit Sagar (102003130)

Rupinderpal Singh (102003167)

Submitted to : Niyaz Wani Sir

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE and ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

July-Dec 2022

ABSTRACT

The Satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this project. We take this opportunity to express our gratitude to all those who have helped us in this project.

Our Sincere thanks to our guide **Niyaz Wani Sir** for giving us their support, guidance and constant encouragement for completion of the project successfully.

<u>INTRODUCTION</u>

In today’s tech world, developers are working hard on machines to make them more smart and intelligent by using machine learning and deep learning techniques so that they can perform tasks similar to humans. With the help of these techniques human effort can be reduced in recognizing, learning, predictions and many other areas.

The ability of computers to recognize human handwritten digits is known as **handwritten digit recognition** from sources such as paper documents, images, touch-screens etc. therefore carrying a wide range of applications in real world scenario.

We have developed a deep learning model to achieve high performance on the handwritten digit recognition task using the **MNIST** dataset and build a **GUI App** based on **Tkinter** where user can draw the digits (single as well as multiple) and recognize it straight away by draw a bounding box surrounding each digit. The model also predicts the corresponding accuracy. Two python files are thus configured : train.py for model training which is then saved as mnist.h5 file and gui.py for running the GUI app.

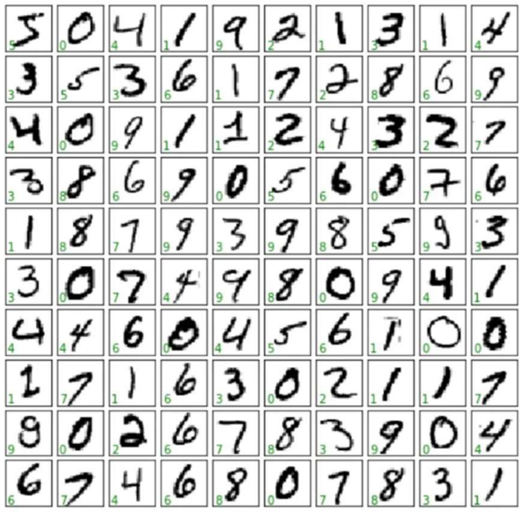
The challenge in handwritten digit recognition is mainly caused by the writing style variations of every single individual. So, it is not easy for the machine to recognize the handwritten digits accurately like the humans do. Hence, robust feature extraction is very important to improve the performance of machines.

ABOUT THE DATASET

MNIST Dataset

The MNIST dataset (Modified National Institute of Standards and Technology) is a large dataset of handwritten digits that is widely used for training and testing in the field of machine learning and deep learning.

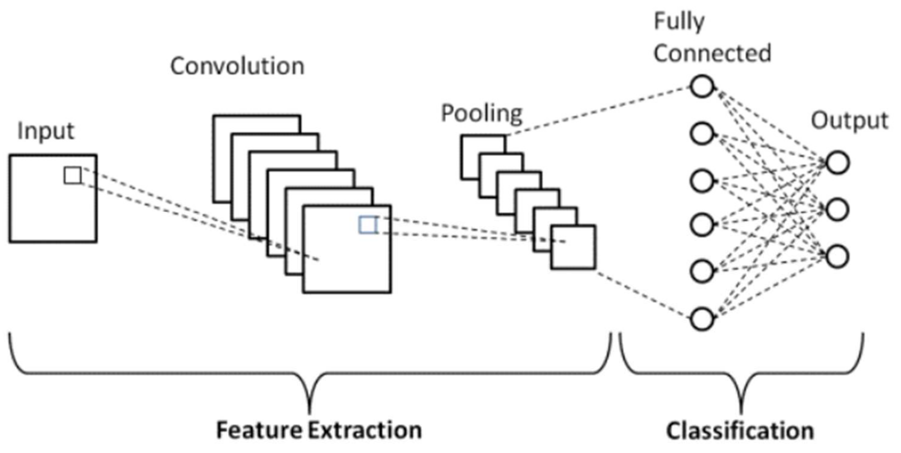
The MNIST dataset contains 60,000 training images and 10,000 testing images of handwritten digits from zero to nine(0 to 9). So, the MNIST dataset has 10 different classes. Each image is represented as a 28×28 matrix where each cell contains grayscale pixel value. In our model, we have imported this dataset using the keras open source library.



Source : MNIST

MODEL CREATION

- ***CNN Model*** : In this project, we have trained our model on MNIST dataset using CNN (Convolutional Neural Network).We have created a CNN model with a double convolutional layer of the same size 3×3, max pooling layers and fully connected layers. The dropout layer is used to deactivate some of the neurons to reduce overfitting. Finally, the output layer has 10 neurons for the 10 classes. We then compile the model with the rmsprop optimizer.

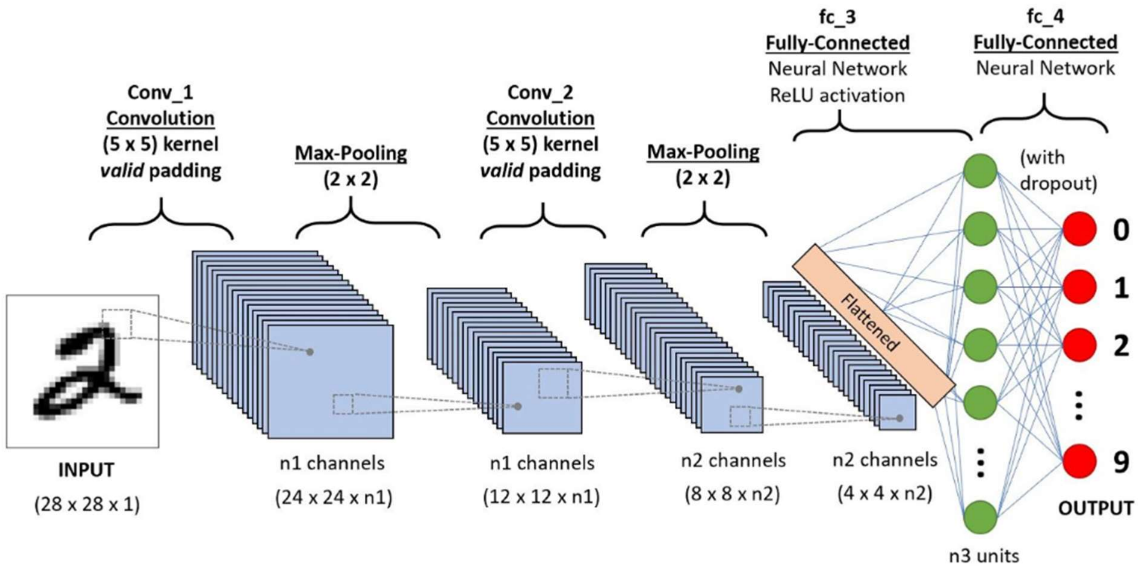


- ***Convolutions***

Convolutions are essentially the CNN model detecting patterns in the image by passing many different types of filters over the image. Each convolutional layer in a CNN is created using the Conv2D()class that simply performs the convolution operation in a two-dimensional space.The movement of the kernel (filter) happens on the input image across a two-dimensional space.

➢ ***Max Pooling***

The convoluted image or the feature map is a representation of the image with a certain feature extracted. Each filter that is applied to the image produces 1 new convoluted image, so if 32 or even 64 filters are used, we end up with 32 times the amount of data. This is a huge amount, thus to decrease computational power and to increase efficiency, the convoluted image is summarized and reduced in size. This process is called Max Pooling.



- ***Sequential Model:***

We use the Sequential API in Keras to build CNN model It is built by using the Sequential() class where we sequentially add layers to the model using the add()method(hence the name Sequential). A CNN can be instantiated as a Sequential model because each layer has exactly one input and output and is stacked together to form the entire network.

GUI CREATION

- **Tkinter Package :** We build GUI App using Tkinter (standard Python interface) of the Tk GUI toolkit which provides a variety of common GUI elements – such as buttons, menus and various kinds of entry fields and display areas.

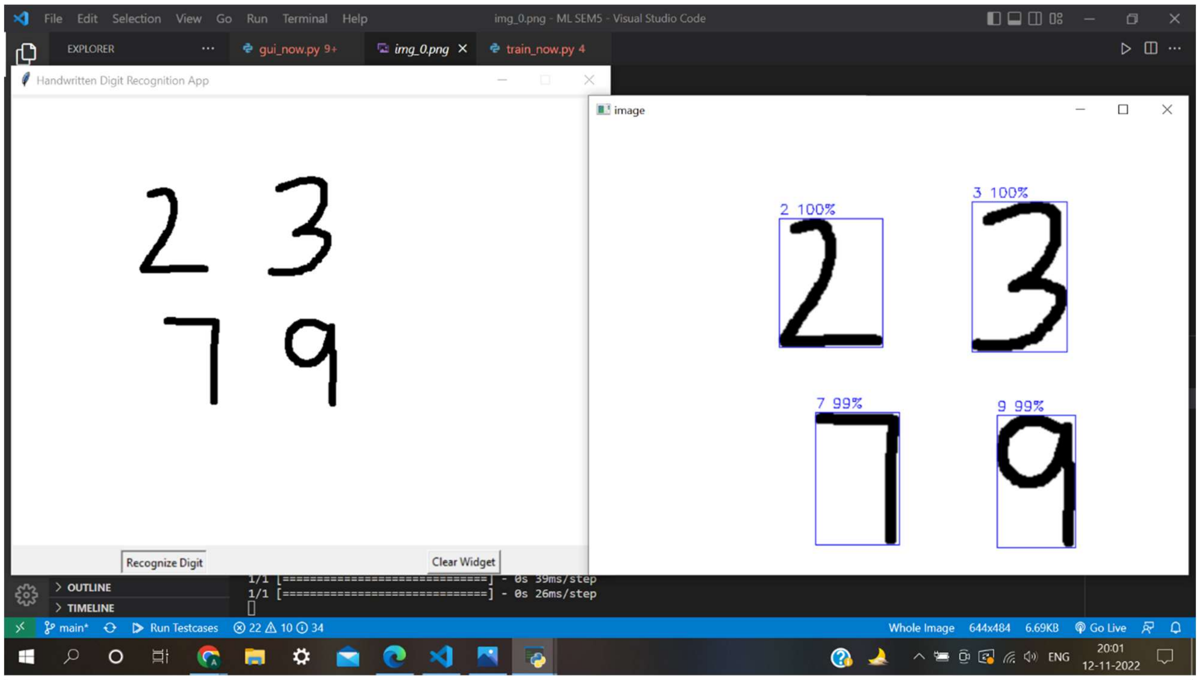
- **Canvas:** Canvas is a widget for drawing graphics which is used to create custom widgets so we can draw anything we like inside it. In our application we use the widget to draw the digit (single as well as multiple).

- **Button :**A Button usually maps directly onto a user action. In our application we use two buttons named as “Recognize Digit” and “Clear Widget”. when the user clicks on a button, it triggers the functions as assigned.

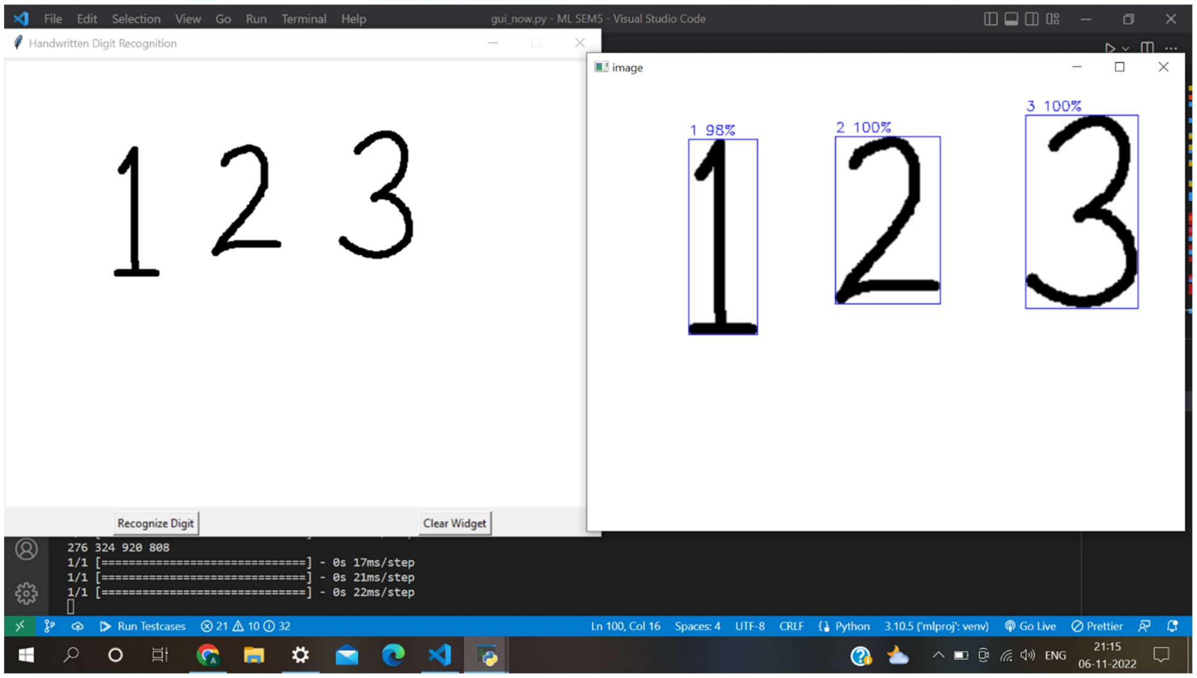
- **OpenCV and PIL :** ImageGrab module has been used to copy the contents of the screen or the clipboard to a PIL (Python Imaging Library) image memory. So, it takes a snapshot of the screen and it is saved in png format. With the help of OpenCV library we find contours of an image .Contours are simply a curve joining all the continuous points (along the boundary), having the same color or intensity.

EXECUTION SCREENSHOTS:

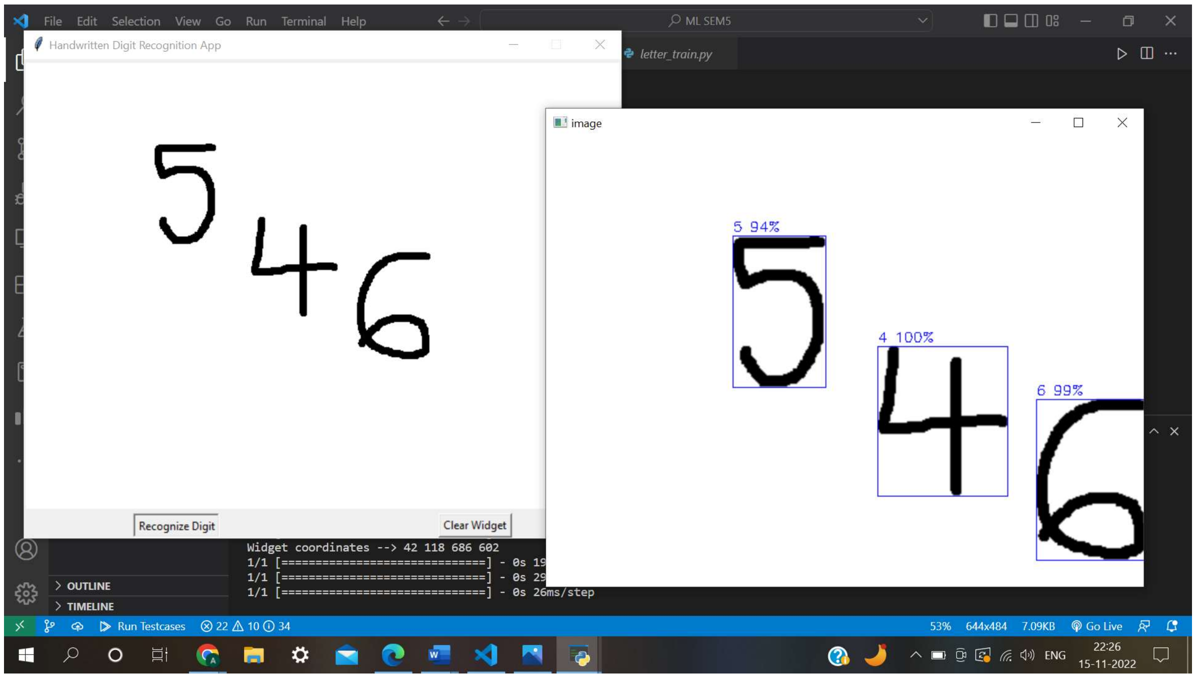
- **Case (1) :**
  When any digit is drawn : Digit style is matched with training images and appropriate result is predicted .img_0_png stores the screenshot of the canvas.



```
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 26ms/step
```

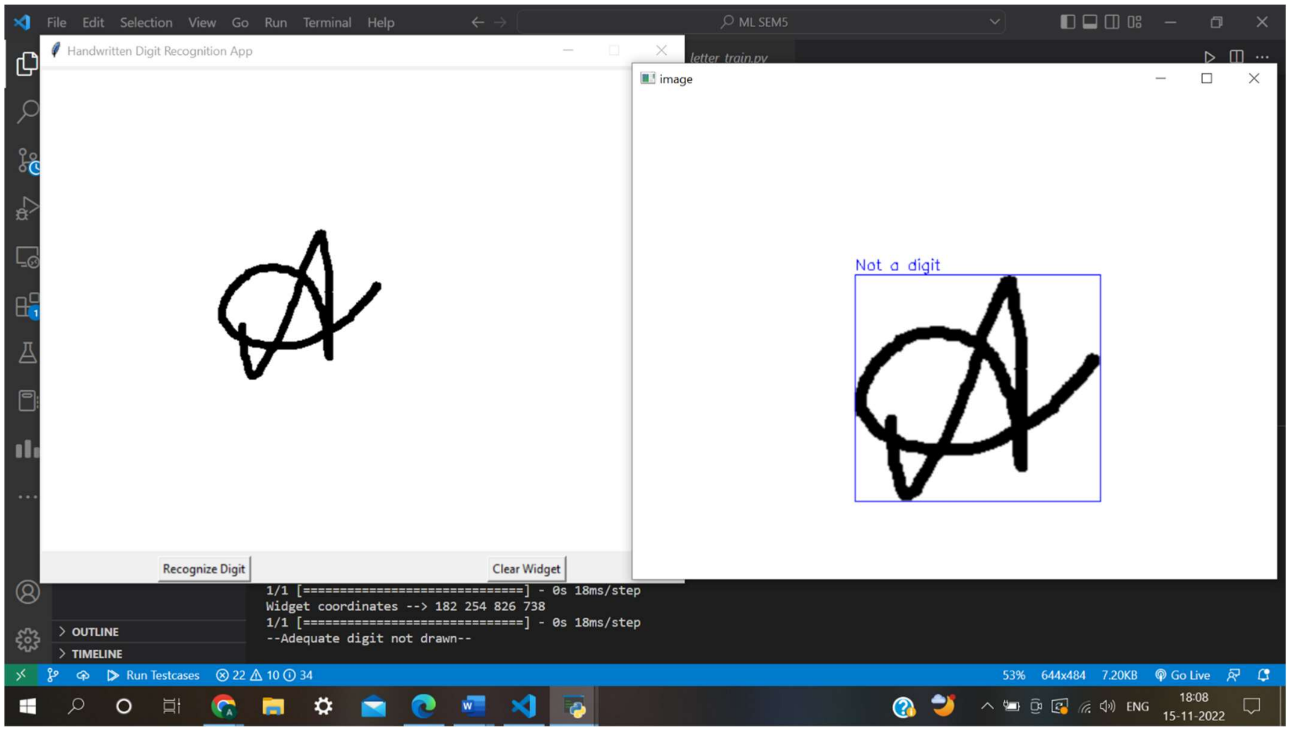

File Edit Selection View Go Run Terminal Help
gui_now.py - ML SEM5 - Visual Studio Code
Handwritten Digit Recognition
image
1 98%
2 100%
3 100%
Recognize Digit
Clear Widget
Run Testcases
21 10 32
Ln 100, Col 16 Spaces: 4 UTF-8 CRLF Python 3.10.5 ('mlproj': venv) Go Live Prettier
ENG
21:15
06-11-2022

```
276 324 920 808
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
```

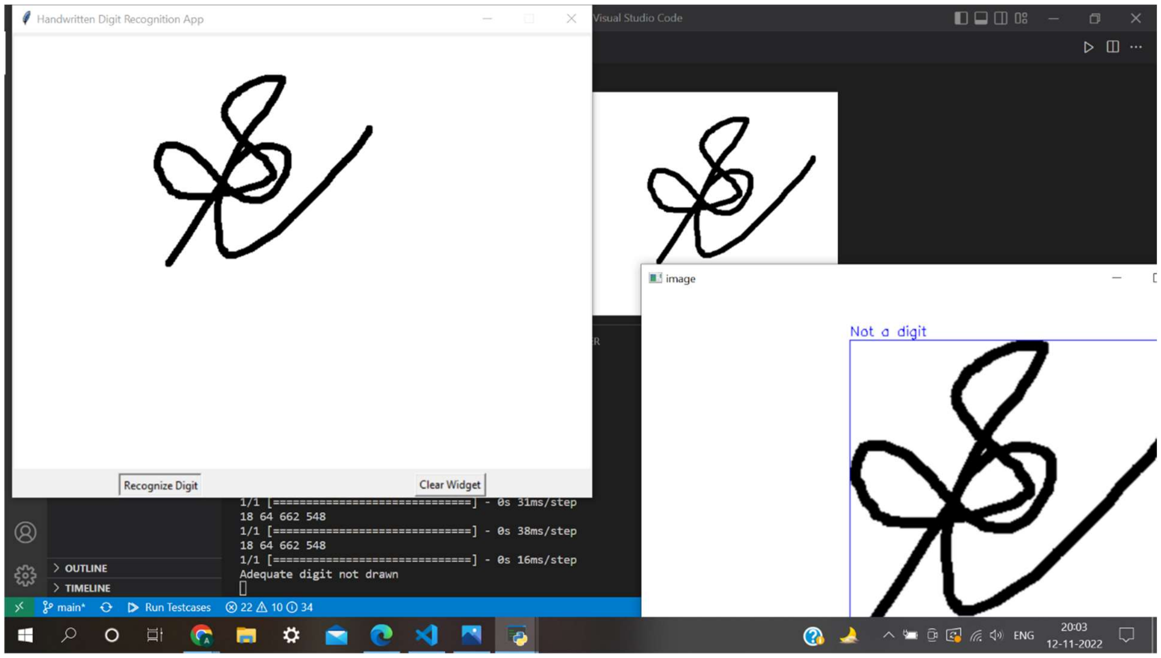


```
Widget coordinates --> 42 118 686 602
1/1 [==============================] - 0s 19
1/1 [==============================] - 0s 29
1/1 [==============================] - 0s 26ms/step
```

- **Case (2) :**
  When anything other than digit is drawn : Message is prompted on terminal "Adequate digit not drawn" with the adequate prediction.



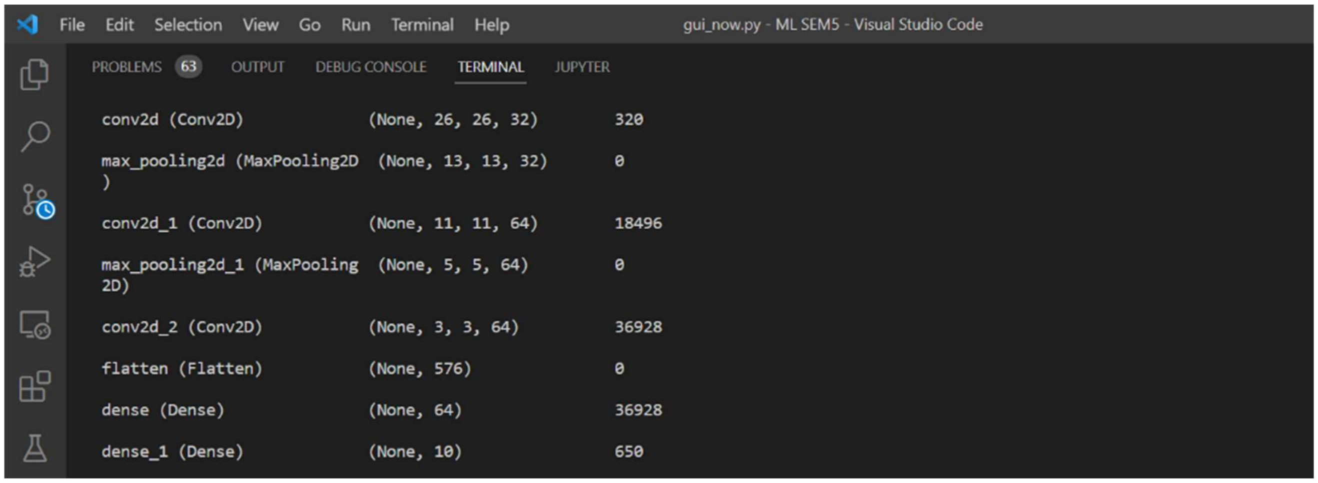
```
1/1 [==============================] - 0s 18ms/step
Widget coordinates --> 182 254 826 738
1/1 [==============================] - 0s 18ms/step
--Adequate digit not drawn--
```



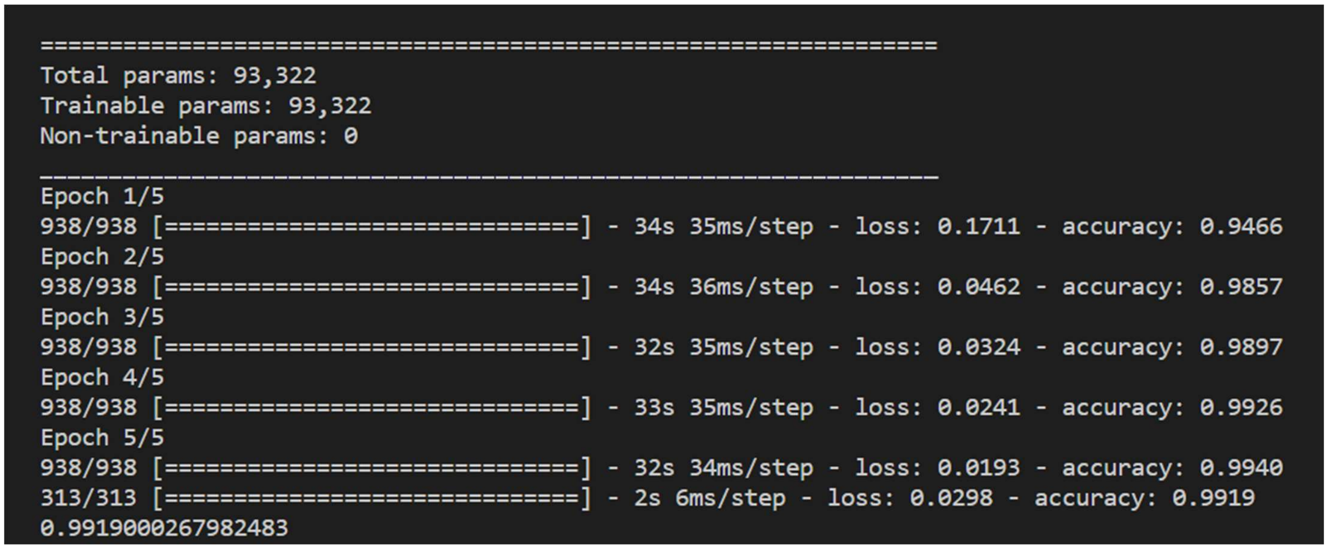
```
1/1 [==============================] - 0s 31ms/step
18 64 662 548
1/1 [==============================] - 0s 38ms/step
18 64 662 548
1/1 [==============================] - 0s 16ms/step
Adequate digit not drawn
```

OUTPUT AND ACCURACY

The model is trained with 5 epochs each having 938 iterations.Succesfully trained model gives the total test accuracy of near about 99.19 % .


File Edit Selection View Go Run Terminal Help
gui_now.py - ML SEM5 - Visual Studio Code
PROBLEMS 63
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER

```
conv2d (Conv2D)             (None, 26, 26, 32)        320

max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
)

conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
2D)

conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

flatten (Flatten)           (None, 576)               0

dense (Dense)               (None, 64)                36928

dense_1 (Dense)             (None, 10)                650
```



```
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
938/938 [==============================] - 34s 35ms/step - loss: 0.1711 - accuracy: 0.9466
Epoch 2/5
938/938 [==============================] - 34s 36ms/step - loss: 0.0462 - accuracy: 0.9857
Epoch 3/5
938/938 [==============================] - 32s 35ms/step - loss: 0.0324 - accuracy: 0.9897
Epoch 4/5
938/938 [==============================] - 33s 35ms/step - loss: 0.0241 - accuracy: 0.9926
Epoch 5/5
938/938 [==============================] - 32s 34ms/step - loss: 0.0193 - accuracy: 0.9940
313/313 [==============================] - 2s 6ms/step - loss: 0.0298 - accuracy: 0.9919
0.9919000267982483
```

So we have built a Python deep learning project on multiple handwritten digit recognition with an interactive GUI App . We have trained the Convolutional neural network which is very effective (CNN error < 1%). User can draw a digit (single as well as multiple) on the canvas and the predicted result (value and percentage) is displayed on the top of the bounding box that surrounds each digit present in the image.

REFERENCES

- https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/
- https://towardsdatascience.com/coding-a-convolutional-neural-network-cnn-using-keras-sequential-api-ec5211126875
- https://keras.io/api/datasets/
- https://www.geeksforgeeks.org/handwritten-digit-recognition-using-neural-network/
- https://ieeexplore.ieee.org/document/9796722/
- https://www.kaggle.com/competitions/digit-recognizer