HANDWRITTEN DIGIT RECOGNITION

A MINI PROJECT REPORT

Submitted by

G SAI MANIKANTA [RA2011003011274]
M NIKHIL LKOESH REDDY [RA2011003011268]
K SUKRUTH[RA2011003011281]

Under the guidance of
Dr. Muralidharan C

in partial fulfillment for the award of the degree
Of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203
MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"HANDWRITTEN DIGIT RECOGNITION"** is the bonafide work of **"G SAI MANIKANTA[RA2011003011274], M NIKHIL LOKESH REDDY [RA2011003011268], K SUKRUTH [RA2011003011281]"**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE 11/5/22

Dr.Muralidharan C
GUIDE
Assistant Professor
Department of Computing Technologies

M. Puspalatha

Department of Computing Technology • SRM IST • Kattankulathur 603 203

SIGNATURE

Dr. M.PUSPHA LATHA
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

Table of Contents

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, **Dr. Muralidharan C** her valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing the project. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this project work.

G SAI MANIKANTA
M NIKHIL LOKESH REDDY
K SUKRUTH

ABSTRACT

Handwritten recognition has received considerable attention in the domain of pattern recognition, image processing, over the last few decades. As a consequence of this project, several algorithms were developed using different techniques. Particularly, Deep Learning has shown a remarkable capability to handle handwritten recognition in very recent years. The wellknown Deep learning techniques are the Convolutional Neuronal Networks (CNNs) and Recurrent Neuronal Networks (RNNs).

This project consists of two parts: creating a CNN to recognize hand written digits and create a GUI where user can draw a digit and can view the predicted output. The Network proposed here is experimented on the well-known MNIST data set. Pre – processing is done on the Dataset to get better accuracy. Handwriting identification system can be used to fix many complicated issues and facilitate the job of beings. We can create an artificial intelligence system that can automatically split individual digits from photos and discover their matching labels. The Neural Network built has good accuracy and low loss.

CHAPTER 1

1 .INTRODUCTION

This project performs hand written digit recognition. To build the project, the technologies used are:

Neural Networks
Deep Learning
Convolutional Neural Network
Tensorflow
Keras

Neural Networks:
Neural networks (NN) aren't stand-alone algorithms. Rather, they provide a structure or framework for combining machine learning algorithms in order to solve certain tasks.

A neural network is essentially a collection of neurons connected via connections. A neuron is a function that has several inputs and only one output. Its job is to take all the numbers from the input, apply a function to them, and output the result.
Neurons communicate through connections, which act as channels. They link the outputs of one neuron to the inputs of another so that they can exchange digits. There is just one parameter for each connection: weight. It's similar to a signal's connection strength.

These weights tell the neuron to respond more to one input and less to another. Weights are adjusted when training.

Deep Learning:

Deep learning refers to a type of machine learning techniques that employ numerous layers to extract higher-level characteristics from raw data. Lower layers in image processing, for example, may recognize edges, whereas higher layers may identify human-relevant concepts like numerals, characters, or faces.

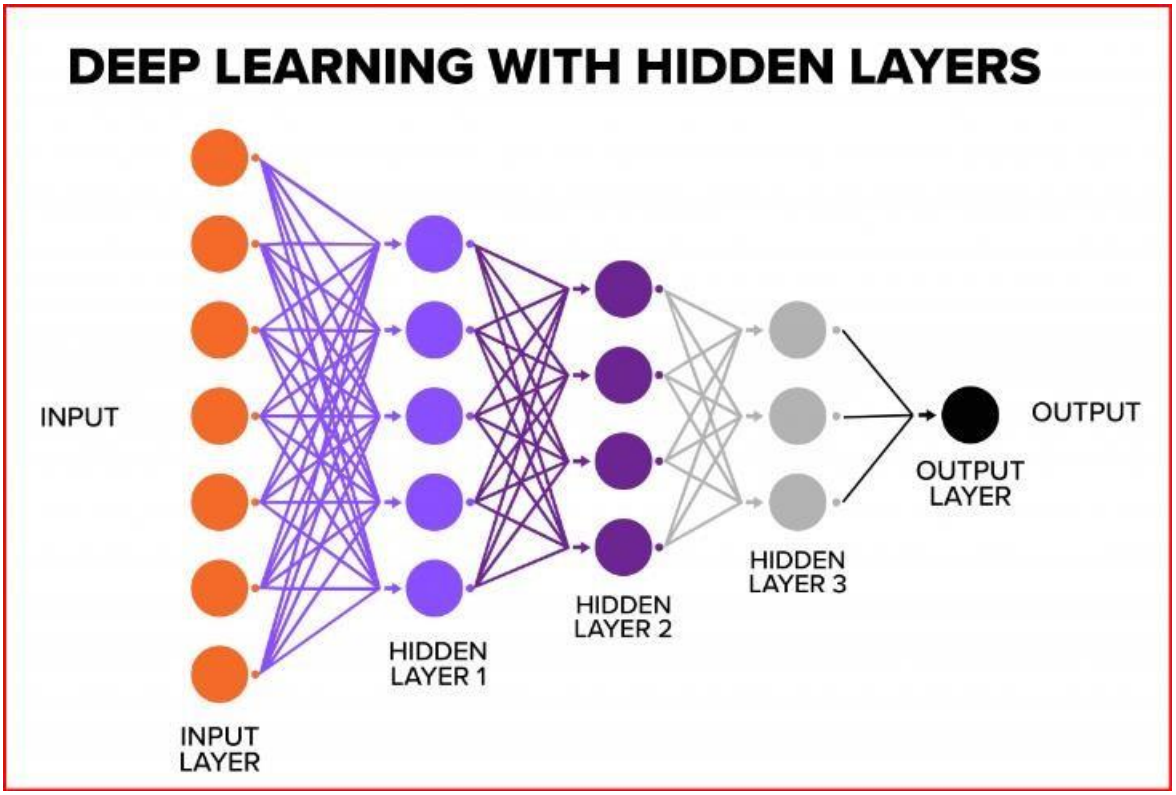


Fig ia-Deep learning with hidden layers

Convolutional Neural Network:

Convolutional Neural Network (CNN) used to search for objects on photos and in videos, face recognition, style transfer, generating and enhancing images, creating effects like slow-mo and improving image quality. CNNs are used in all cases that involve pictures and videos.

The defining feature of the CNN is that it performs the convolution operation in certain layers — hence, the name Convolutional Neural Network. The architecture varies slightly from the

traditional NN, starting with the makeup of the individual layers. Convolution can be represented as a layer of a neural network because each neuron can act as any function.

Tensorflow:

TensorFlow is a machine learning and artificial intelligence software library that is free and open-source. It may be used for a variety of applications, but it focuses on deep neural network training and inference.

The Google Brain team created TensorFlow for internal Google usage in research and production. In 2015, the first version was released under the Apache License 2.0. In September 2019, Google launched TensorFlow 2.0, an improved version of TensorFlow.

TensorFlow is compatible with a broad range of programming languages, including Python, Javascript, C++, and Java. This adaptability lends itself to a wide range of applications in a variety of industries.

Keras:

Keras is an open-source high-level Neural Network library written in Python that can be used with Theano, TensorFlow, and CNTK. Francois Chollet, a Google developer, was the one who created it. It has been made user-friendly, expandable, and modular to allow for speedier deep neural network research. It not only supports individual Convolutional and Recurrent Networks, but also their combination.

It uses the Backend library to resolve low-level calculations because it can't handle them. The backend library wraps the low-level API in a high-level API, allowing it to operate on TensorFlow, CNTK, or Theano.

1.1 OBJECTIVE

The project has two objectives:

1. Build a CNN Model which classifies handwritten digits from 0 – 9 with good accuracy.
2. Build a GUI where the user can draw the digit with freehand and the model built recognizes the digit. The output is shown in the same GUI along with the accuracy of recognition.

1.2 PROBLEM STATEMENT

Handwriting identification systems can be handy in many tasks. They can aid in the development of Artificial Intelligence systems which automatically split individual digits from photos and discover their matching labels. The work previously done on this subject has problems which may include one or more of: low accuracy, no environment where the user can interact with the system, failure to make right predictions on real life data. This project aims to solve all these problems.

1.3 PROPOSED SOLUTION

The problems mentioned above are solved in this project by building a digit recognition system using deep learning. The model built is a CNN model built using Tensorflow which takes the image of the digit and discovers to which label it belongs to from 10 given labels.

Finally, we build a user interface where the user can draw the digit with freehand, the model built recognizes the digit and output is shown in the same GUI along with the accuracy of recognition.

CHAPTER - 2

2.1 LITERATURE REVIEW

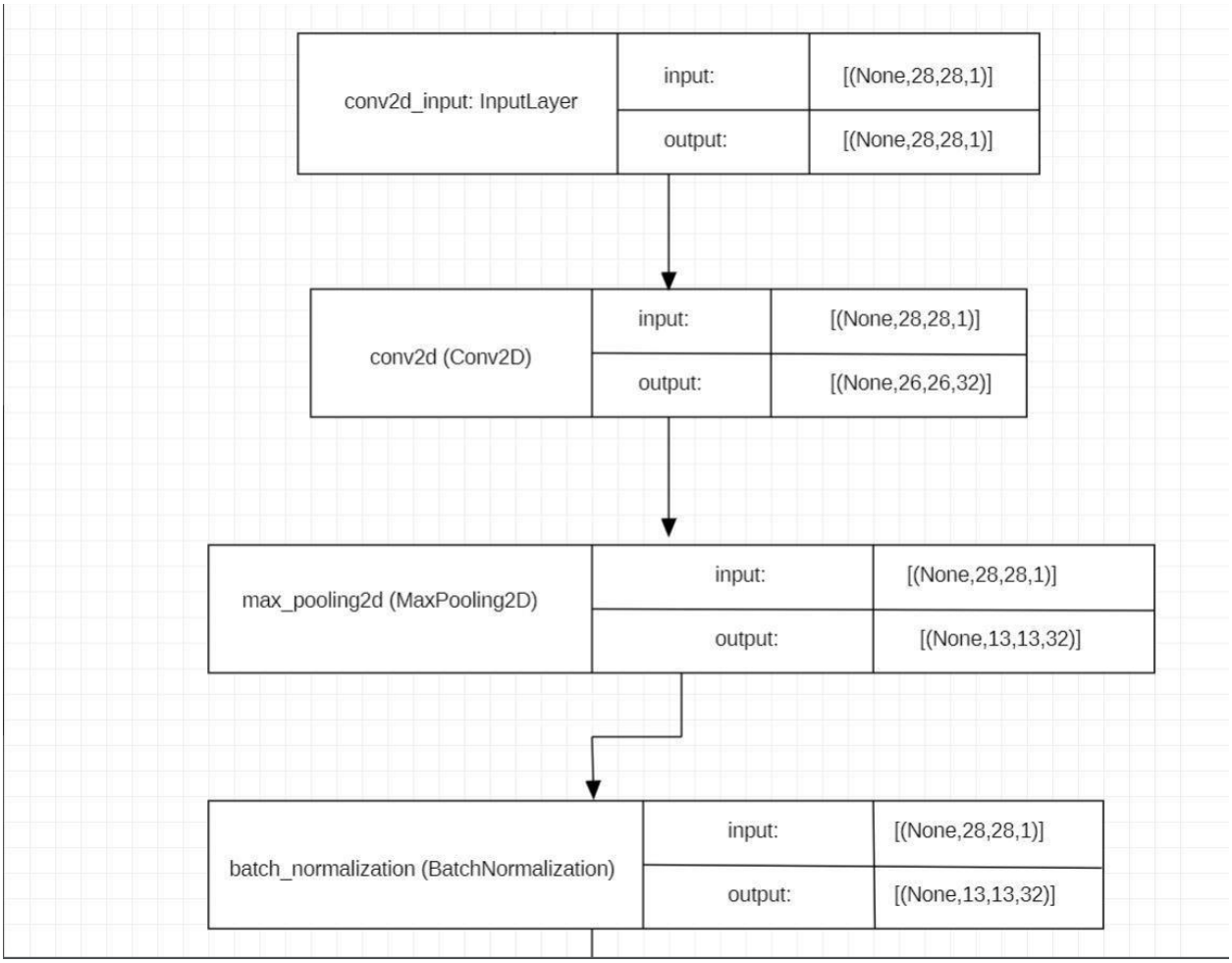
S. No.	Project name	Publishing year	Journal name	Author name
1 [1]	Handwritten Digit Recognition Using Machine Learning: A Review	2019	IEEE	Anchit Shrivastava, Isha Jaggi, Sheifali Gupta, Deepali Gupta
2 [2]	Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)	2021	IEEE	Drishti Beohar; Akhtar Rasool
3 [3]	Handwritten Digit Recognition System Based on Convolutional Neural Network	2020	IEEE	Jinze Li; Gongbo Sun; Leiye Yi; Qian Cao; Fusen Liang; Yu Sun
4 [4]	Hand Written Digit Recognition using Machine Learning	2020	IEEE	Rohan Sethi; Ila Kaushik
5 [5]	Handwritten Digit Recognition Based on	2020	IEEE	Chao Zhang; Zhiyao Zhou; Lan Lin
	Convolutional Neural Network			

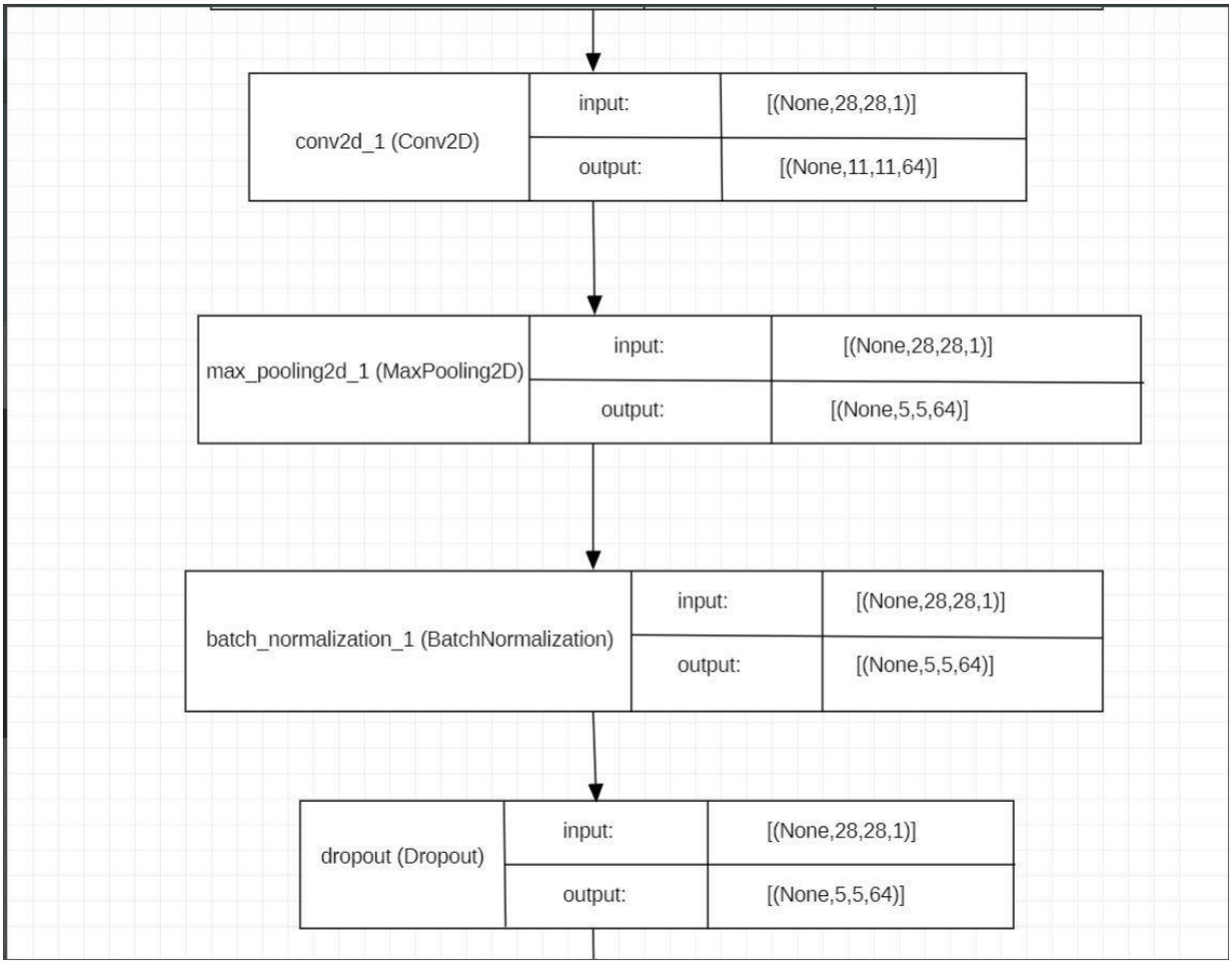
6 [6]	Handwritten Digit Recognition using Convolution Neural Networks	2022	IEEE	Shailesh S Rajput; Yoonsuk Choi
7 [7]	Handwritten Digit Recognition Using CNN	2021	IEEE	Mayank Jain; Gagandeep Kaur; Muhammad Parvez Quamar; Harshit Gupta
8 [8]	Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers	2019	IEEE	Fathma Siddique; Shadman Sakib; Md. Abu Bakr Siddique
9 [9]	An Enhanced Handwritten Digit Recognition Using Convolutional Neural Network	2021	IEEE	Malathy. S; C N Vanitha; Nirdhum Narayan; Rajesh Kumar; Gokul. R
10 [10]	Recognition of Handwritten Digit Using Convolutional Neural Network	2020	IEEE	Peiyu Ma

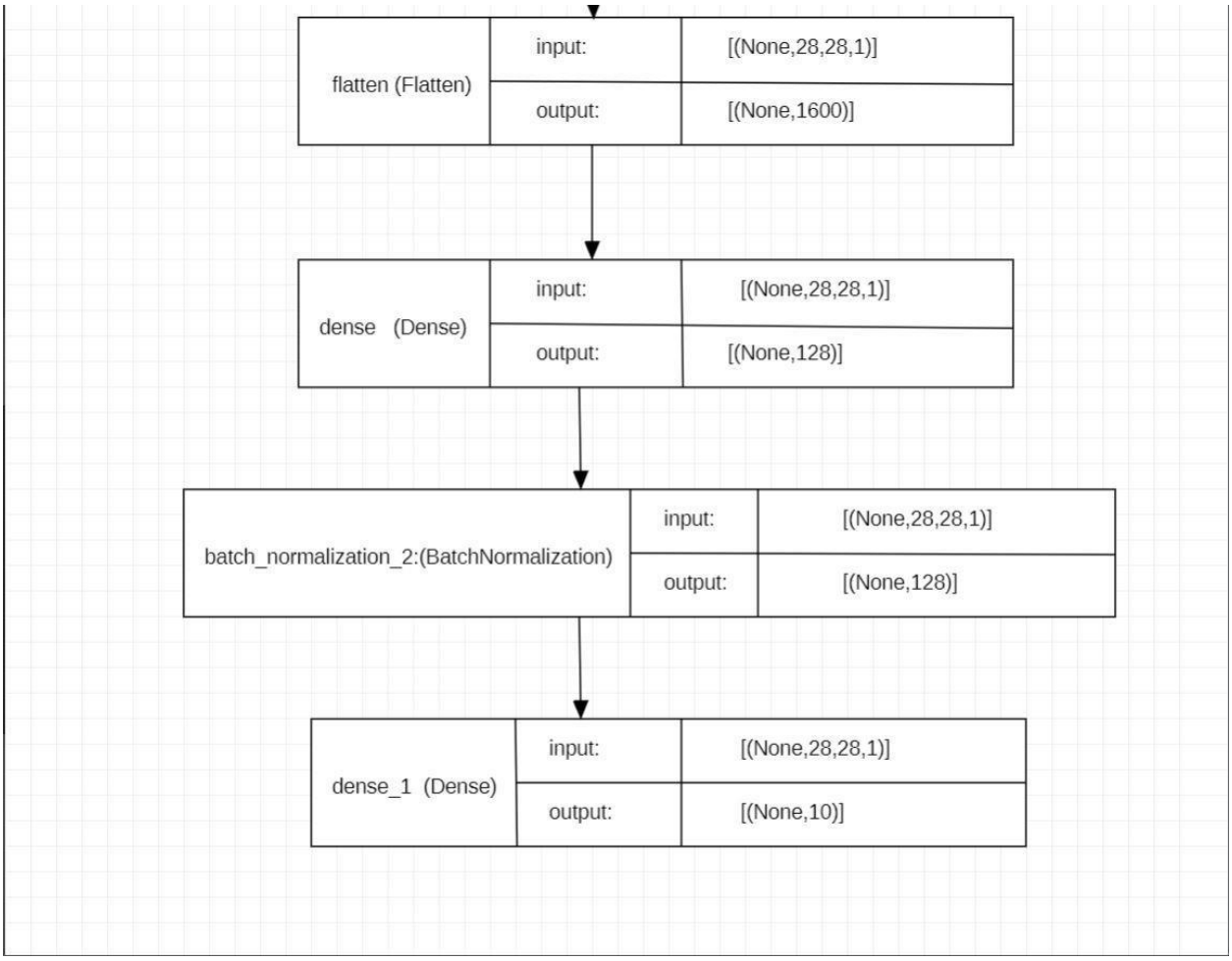
Fig(2.1)

CHAPTER - 3
PROPOSED METHODOLOGY

3.1 ARCHITECTURE DIAGRAM



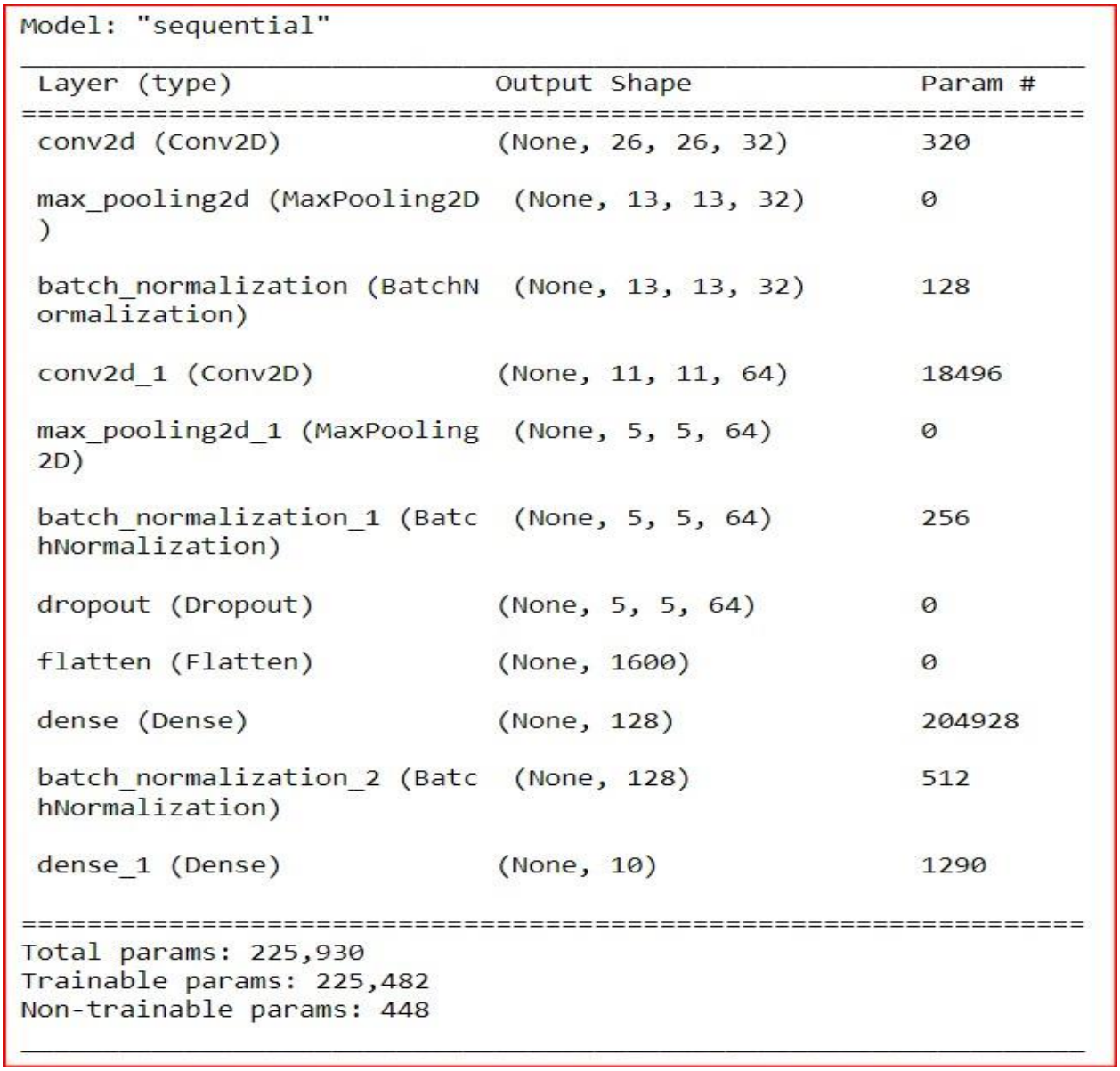

conv2d_1 (Conv2D)
input:
[(None,28,28,1)]
output:
[(None,11,11,64)]
max_pooling2d_1 (MaxPooling2D)
input:
[(None,28,28,1)]
output:
[(None,5,5,64)]
batch_normalization_1 (BatchNormalization)
input:
[(None,28,28,1)]
output:
[(None,5,5,64)]
dropout (Dropout)
input:
[(None,28,28,1)]
output:
[(None,5,5,64)]


flatten (Flatten)
input:
[(None,28,28,1)]
output:
[(None,1600)]
dense (Dense)
input:
[(None,28,28,1)]
output:
[(None,128)]
batch_normalization_2:(BatchNormalization)
input:
[(None,28,28,1)]
output:
[(None,128)]
dense_1 (Dense)
input:
[(None,28,28,1)]
output:
[(None,10)]

3.2 DESCRIPTION OF PROPOSED MODEL: -

The model consists of two convolution layers, one dense layer and an output layer which has 10 nodes. The 10 nodes represent 10 output classes, digits 0 – 9. The first layer has filter size (3,3) and 32 filters and a max pooling layer is added to it. The activation function used is ReLU. The second layer has filter size (3,3) and 64 filters and a max pooling layer is added to it. The activation function used is ReLU. The dense layer has 128 nodes and the activation function used is ReLU. The activation function used in output layer is softmax.

To compile the model, adam optimizer is used and as the given problem is a multi – class classification, sparse categorical cross – entropy function loss function is used. The metric being monitored is the accuracy.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

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

 batch_normalization (BatchN  (None, 13, 13, 32)       128
 ormalization)

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

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

 batch_normalization_1 (Batc  (None, 5, 5, 64)         256
 hNormalization)

 dropout (Dropout)           (None, 5, 5, 64)          0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 batch_normalization_2 (Batc  (None, 128)              512
 hNormalization)

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 225,930
Trainable params: 225,482
Non-trainable params: 448
_________________________________________________________________
```

FIGURE 3.2

CHAPTER - 4

TOOLS AND SOFTWARES USED

4.1 DATASET DESCRIPTION

The Dataset used is the Modified National Institute of Standards and Technology database (MNIST) Dataset. It can be imported using:

```
import tensorflow
tensorflow.keras.datasets.mnist.load_data()
```

This contains 70,000 images of handwritten digits 0-9. It has 60,000 images in training set and 10,000 images in testing set. The images in the dataset look like:

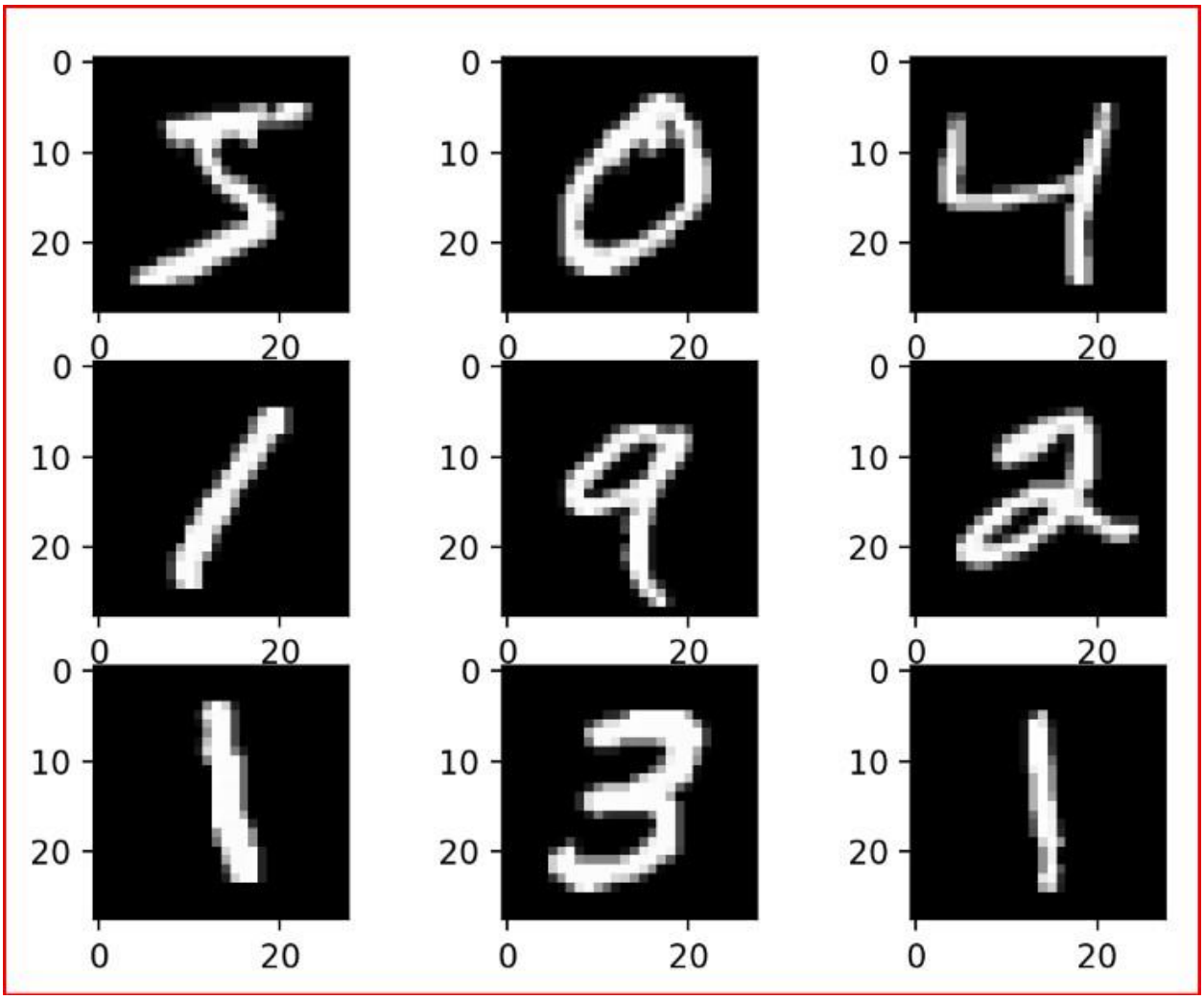


Fig 4.1-Data set

4.2 TOOLS DESCRIPTION

The tools and libraries used in the project are:

- NumPy Library
- Tkinter Library
- Pillow Library
- Matplotlib Library
- Seaborn Library
- Pandas Library

- Jupyter Notebook

NumPy:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Tkinter:

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Pillow library:

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is

also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Seaborn:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

Pandas:

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language.

Jupyter Notebook:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

CHAPTER - 5

RESULTS AND DISCUSSION

5.1 CODE IMPLEMENTATION: -

```
import tensorflow as tf import
matplotlib.pyplot as plt import
seaborn as sns
import pandas as pd

#Loading Data

df = tf.keras.datasets.mnist.load_data()

(X_train,y_train),(X_test,y_test)=df

# Normalization

X_train=(X_train.astype('float32').reshape(X_train.shape[0],28,28,1))/255.0
X_test=(X_test.astype('float32').reshape(X_test.shape[0],28,28,1))/255.0


# Building the model

from tensorflow.keras.layers import BatchNormalization model=tf.keras.models.Sequential([

  # Layer1
  tf.keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)),
  tf.keras.layers.MaxPooling2D(2,2),
  BatchNormalization(axis=-1),

  # Layer2
  tf.keras.layers.Conv2D(64,(3,3),activation='relu',input_shape=(28,28,1)),
  tf.keras.layers.MaxPooling2D(2,2),   BatchNormalization(axis=-
1),   #Flattening   tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Flatten(),

  #Dense Layer
```

```
  tf.keras.layers.Dense(128,activation='relu'),
  BatchNormalization(axis=-1),

  #Output Layer
  tf.keras.layers.Dense(10,activation='softmax')
])

model.summary()

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics= ['accuracy'])

#Fitting Data into the model
h = model.fit (X_train, y_train , epochs= 10,validation_split=0.2)

#saving the model
model.save("handwritten_digit_recognition_model.h5")

#Training                                    Accuracy
loss,acc=model.evaluate(X_train,y_train, verbose =0)
print("The train accuracy of model is :-",acc*100)

#Testing Accuracy
test_loss,test_acc=model.evaluate(X_test,y_test,verbose =0) print("The
test accuracy of model is :-",test_acc*100)

df0=pd.DataFrame(x.history) plt.figure(figsize
= (7,7))
sns.lineplot(data=df0[['accuracy','val_accuracy']],palette=['r', 'g'],linewidth=4) plt.show()

plt.figure(figsize = (7,7))
sns.lineplot(data=df0[['loss','val_loss']],palette=['b', 'k'],linewidth=4) plt.show()


# Loading saved model
#model = tf.keras.models.load_model(r'path\handwritten_digit_recognition_model.h5')

from keras.models import load_model
from tkinter import *
import  tkinter  as  tk
import win32gui
```

```
from PIL import ImageGrab, ImageOps import
numpy as np
model = load_model('handwritten_digit_recognition_model.h5')

def predict_digit(img):
#resize image to 28x28 pixels
img = img.resize((28,28))
#convert rgb to grayscale img
= img.convert('L') img =
ImageOps.invert(img) img =
np.array(img)
   #reshaping to support our model input and normalizing
img = img.reshape(1,28,28,1)

   img = img/255.0
#predicting the class res =
model.predict([img])[0]
   return np.argmax(res), max(res)

class App(tk.Tk): def
__init__(self):
tk.Tk.__init__(self)
      self.x = self.y = 0

      # Creating elements
      self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")
self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
self.classify_btn = tk.Button(self, text = "Recognise", command =
self.classify_handwriting)
      self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)

      # Grid structure
      self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
self.label.grid(row=0, column=1,pady=2, padx=2)
self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
self.button_clear.grid(row=1, column=0, pady=2)
#self.canvas.bind("<Motion>", self.start_pos)
      self.canvas.bind("<B1-Motion>", self.draw_lines)

   def clear_all(self):
self.canvas.delete("all")
```

```
    def classify_handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas         rect =
win32gui.GetWindowRect(HWND) # get the coordinate of the canvas         im =
ImageGrab.grab(rect)         digit, acc = predict_digit(im)
        self.label.configure(text= str(digit)+', '+ str(int(acc*100))+'%')

    def draw_lines(self, event):
        self.x = event.x
self.y = event.y         r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

app = App()
mainloop()
```

5.2 PERFORMANCE EVALUATION & METRICS: -

Optimizer used is “Adam” and the loss function used is sparse categorical cross – entropy function. The metric being monitored is the accuracy.

Accuracy and loss plots:

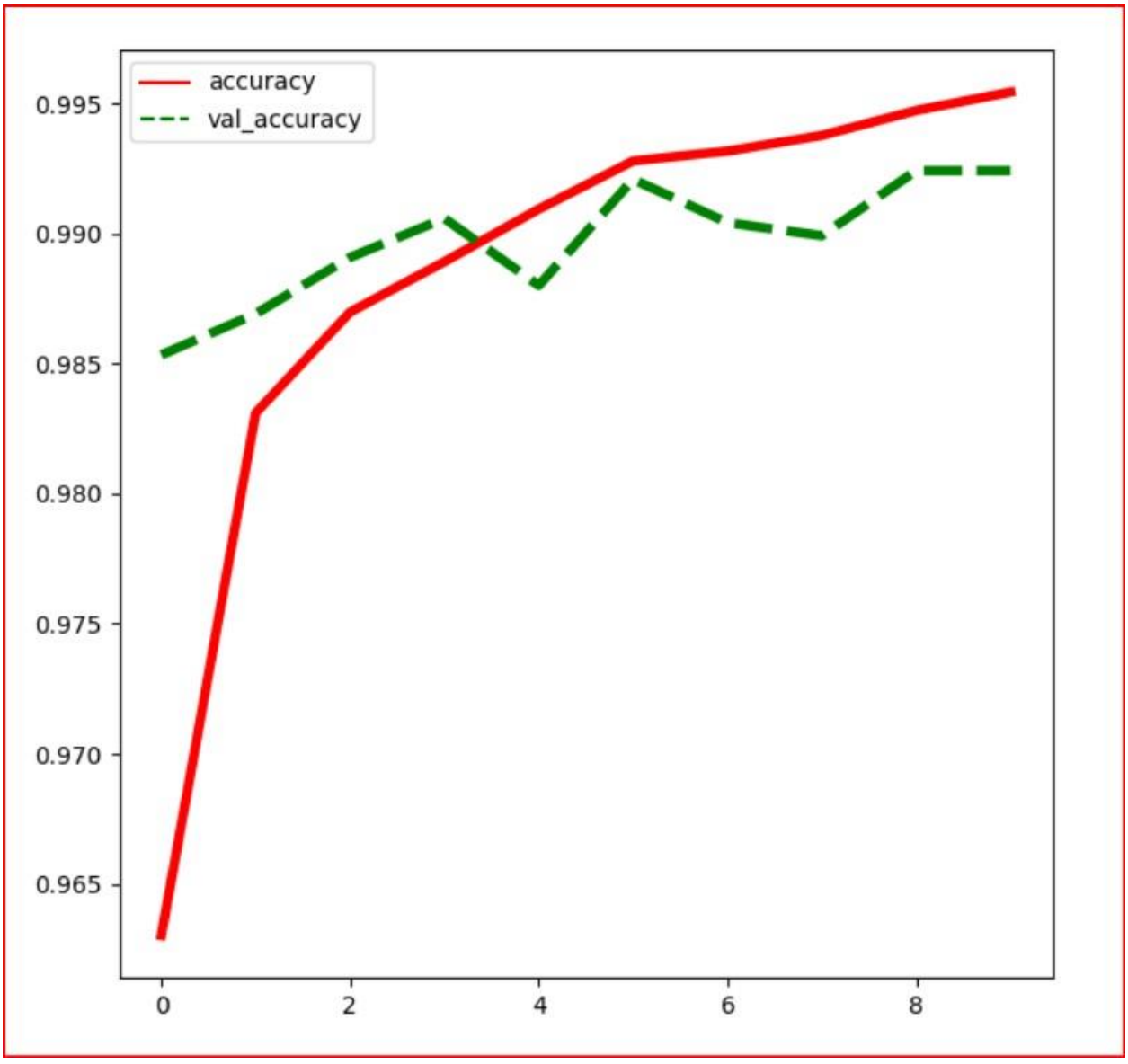


Fig 5.1a Accuracy Plot

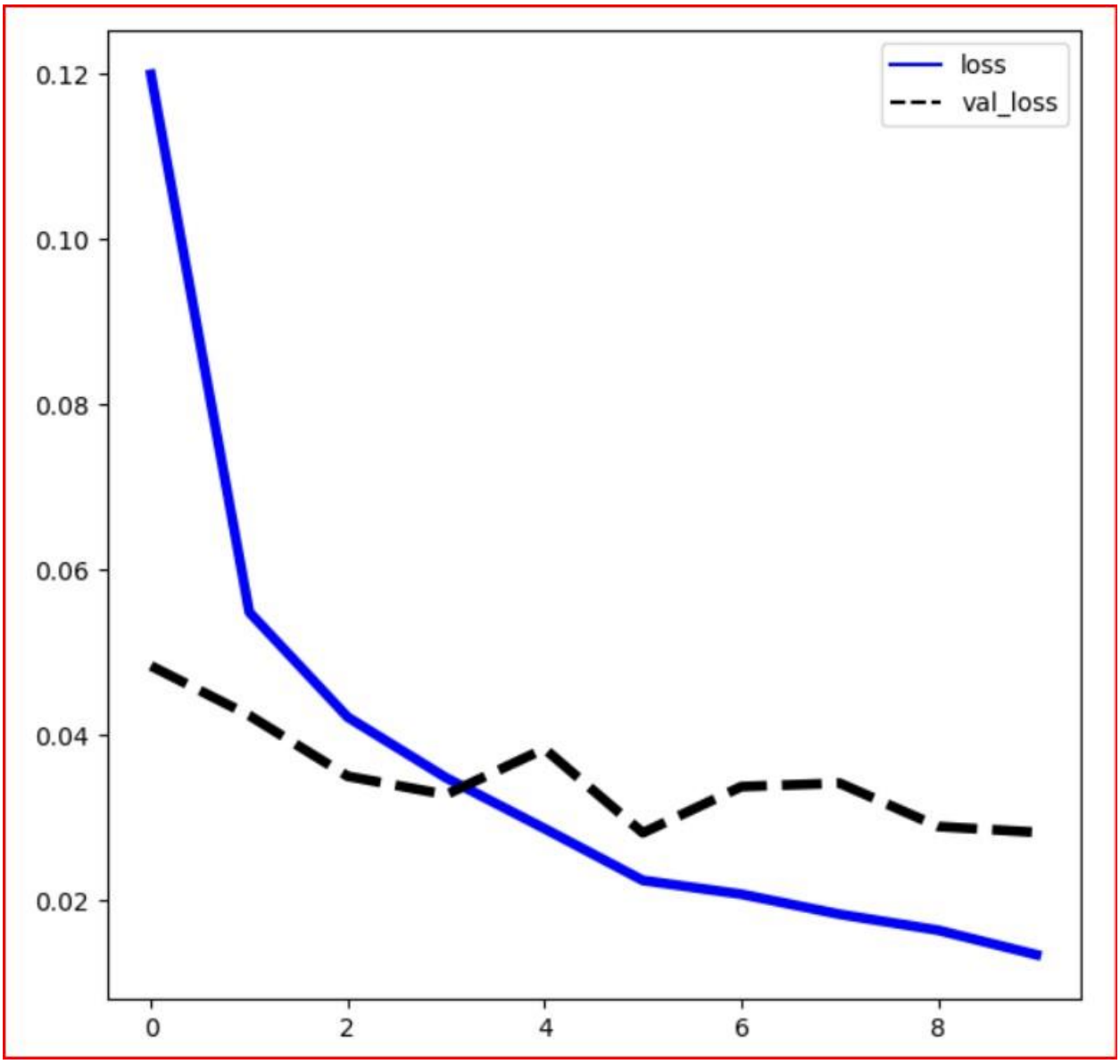


Fig 5.1b Loss Plot

5.3 RESULTS: -

The accuracy obtained is:

• On training set: 99.64%

• On testing set: 99.06%

The output shown for all 10 labels:

5.31 Figures of Digits there predicted value and accuracy:

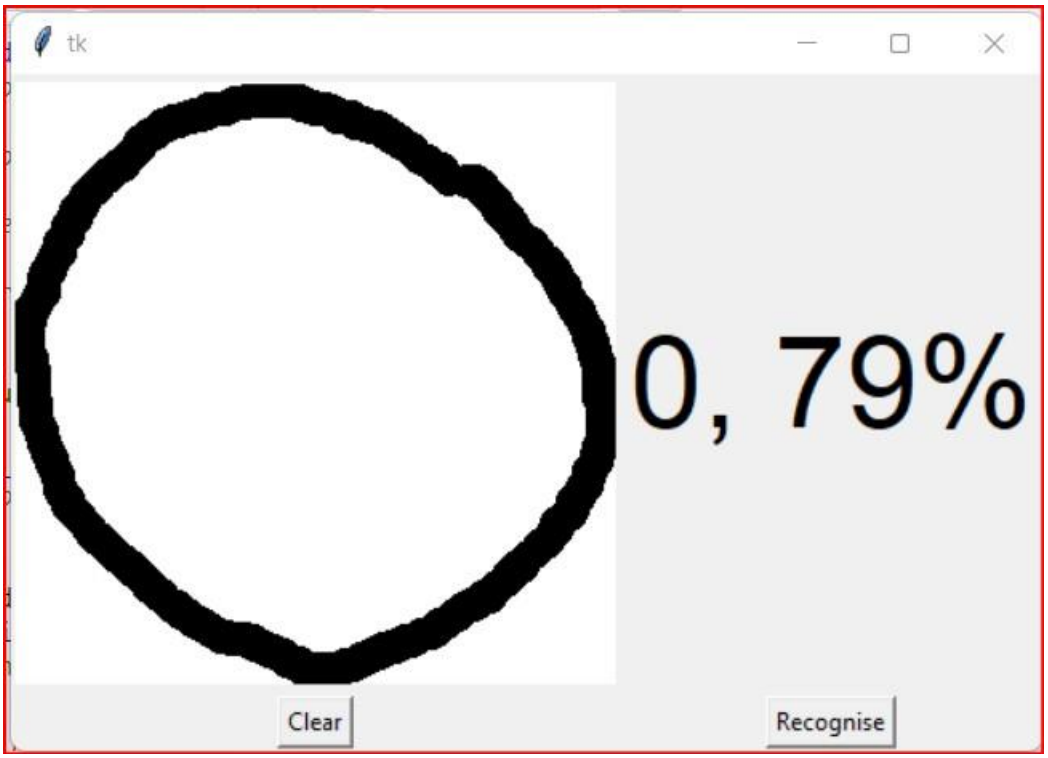


Fig 5.31a

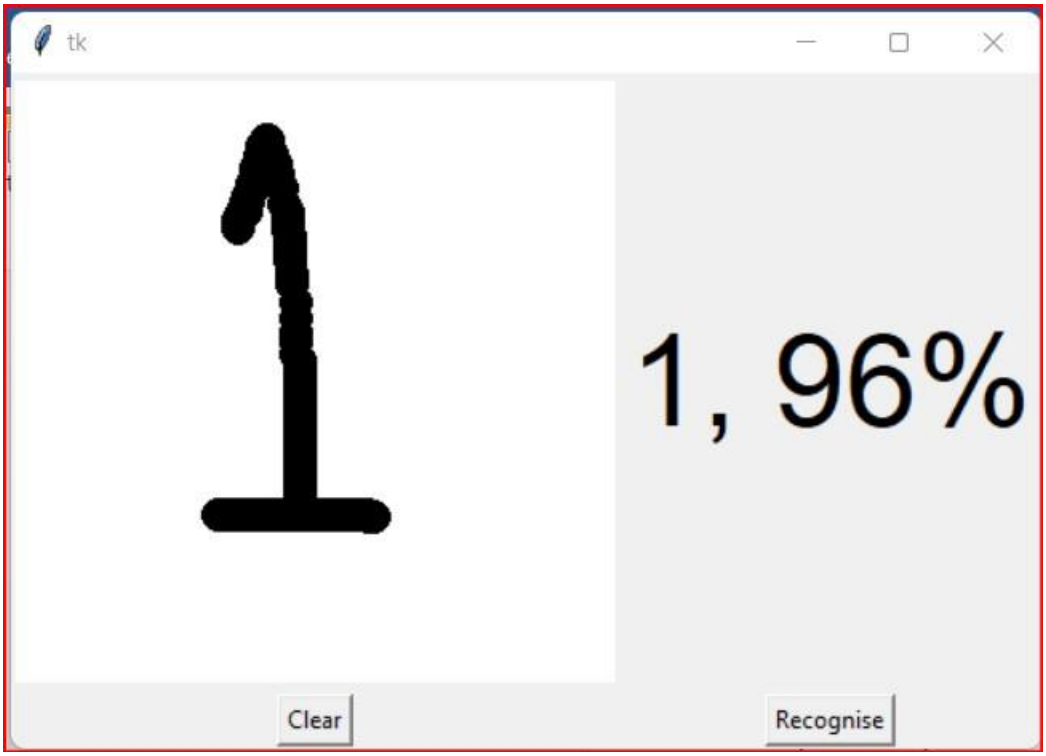


Fig 5.31b

Fig 5.31c

Fig 5.31d

Fig 5.31e

Fig 5.31f

Fig 5.31f

Fig 5.31g

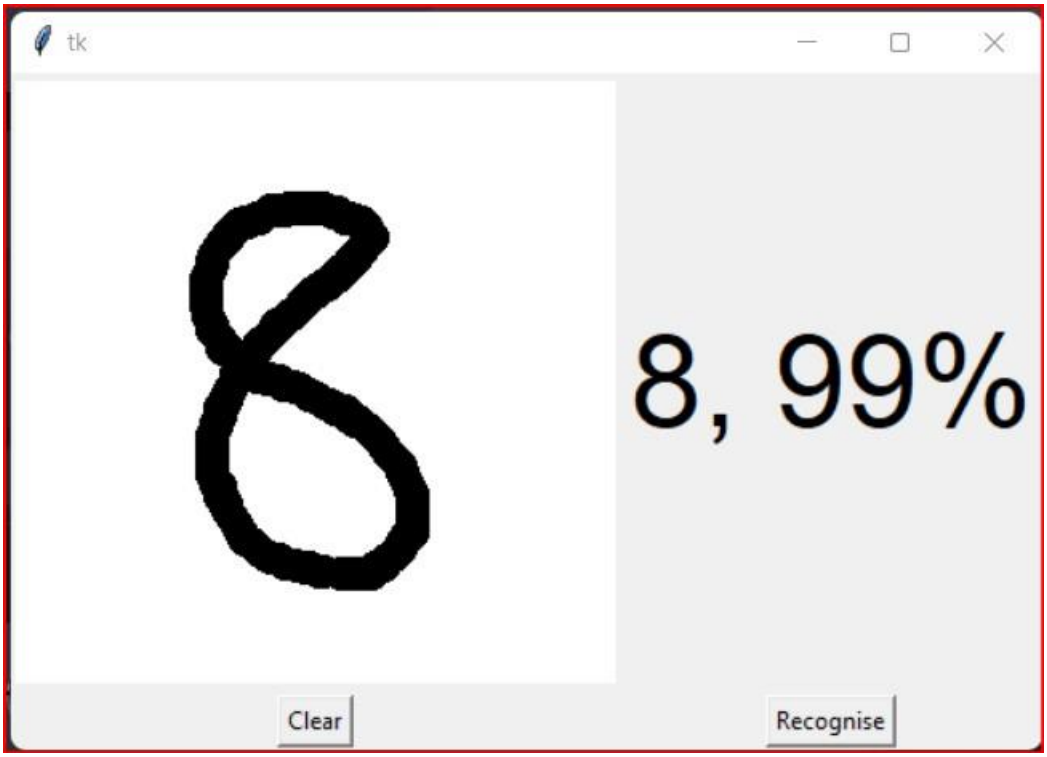


Fig 5.31i

Fig-5.31j

5.4 DISCUSSION: -

The built CNN model recognizes the given handwritten digit with test accuracy of 99.06%. The project can be further improved by improving accuracy of the model by increasing number of layers or epochs or both. Also, another functionality can be added which, if multiple digits are given, i.e, if a number with 2 or more digits is given, splits the number into single digits and sends input to the model so that we can recognize these numbers too.

CHAPTER - 6

6.1 CONCLUSION

In addition, ongoing research and development in the field is likely to lead to even better accuracy and performance in the future.Overall, handwritten recognition technology has the potential to greatly improve efficiency and accuracy in a wide range of industries, from finance to healthcare to education.

Hence, Handwritten recognition project is implemented. The built project recognizes the digit written by the user on the GUI.

6.2 REFERENCES: -

[1] A. Shrivastava, I. Jaggi, S. Gupta and D. Gupta, "Handwritten Digit Recognition Using Machine Learning: A Review," 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC), 2019, pp. 322-326, doi: 10.1109/PEEIC47157.2019.8976601.

[2] D. Beohar and A. Rasool, "Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021, pp. 542-548, doi: 10.1109/ESCI50559.2021.9396870.

[3] J. Li, G. Sun, L. Yi, Q. Cao, F. Liang and Y. Sun, "Handwritten Digit Recognition System Based on Convolutional Neural Network," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020, pp. 739-742, doi: 10.1109/AEECA49918.2020.9213619.

[4] R. Sethi and I. Kaushik, "Hand Written Digit Recognition using Machine Learning," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 49-54, doi: 10.1109/CSNT48778.2020.9115746.

[5] C. Zhang, Z. Zhou and L. Lin, "Handwritten Digit Recognition Based on Convolutional Neural Network," 2020 Chinese Automation Congress (CAC), 2020, pp. 7384-7388, doi: 10.1109/CAC51589.2020.9326781.

[6] S. S. Rajput and Y. Choi, "Handwritten Digit Recognition using Convolution Neural Networks," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022, pp. 0163-0168, doi: 10.1109/CCWC54503.2022.9720854.

[7] M. Jain, G. Kaur, M. P. Quamar and H. Gupta, "Handwritten Digit Recognition Using CNN," 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), 2021, pp. 211-215, doi: 10.1109/ICIPTM52218.2021.9388351.

[8] F. Siddique, S. Sakib and M. A. B. Siddique, "Recognition of Handwritten Digit using

Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers," 2019 5th International Conference on Advances in Electrical Engineering (ICAEE), 2019, pp. 541-546, doi: 10.1109/ICAEE48663.2019.8975496.

[9] M. S, C. N. Vanitha, N. Narayan, R. Kumar and G. R, "An Enhanced Handwritten Digit Recognition Using Convolutional Neural Network," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021, pp. 1724-1727, doi: 10.1109/ICIRCA51532.2021.9544669.

[10]P. Ma, "Recognition of Handwritten Digit Using Convolutional Neural Network," 2020 International Conference on Computing and Data Science (CDS), 2020, pp. 183-190, doi: 10.1109/CDS49703.2020.00044.