**CHAPTER 1**

**INTRODUCTION**

## 1.1 Introduction

When compared to prior knowledge or experience, awareness recognises or distinguishes an object or person. Comparably, digital recognition is simply the process of identifying or recognising numbers in any document. The framework of digital recognition is simply the operation of the machine to prepare or interpret digits. Handwritten digit recognition is the ability of computers to read handwritten digits from a number of sources, including text messages, bank checks, papers, pictures, etc.

Machine Learning offers a variety of ways in which human effort can be reduced to seeing handwritten digits. In-depth reading is a machine learning method that trains computers to do what most people can easily access: learning by example. With the use of in-depth learning methods, human efforts can be reduced in perception, learning, perception and in too many regions. Using in-depth learning, the computer learns to perform distinctive functions in images or content anywhere accuracy, in addition to the performance of the human level. The digital recognition model uses large data sets to detect digits from different sources.

### 1.1.1 Problem Statement

The handwritten digits are not always of the same size, width, orientation and justified to margins as they differ from writing of person to person, so the general problem would be while classifying the digits due to the similarity between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc.

This problem is faced more when many people write a single digit with a variety of different handwritings. Lastly, the uniqueness and variety in the handwriting of different individuals also influence the formation and appearance of the digits. Now we introduce the concepts and algorithms of deep learning and machine learning.

The uniqueness and diversity in handwriting of various people also contribute to the growth and existence of digits. In addition, people create the same digit with diverse concepts.

## 1.2 Aim of the project

The aim of a handwriting digit recognition system is to convert handwritten digits into machine readable formats, to ensure effective, error free and reliable approaches for recognition of handwritten digits and to get the accurate information of what we want.

## 1.3 Project Domain

With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and many more areas. The domain of the project is Deep learning. The progress of deep learning techniques has been challenging when it comes computer vision and Image processing. Deep learning uses various algorithms based on the requirement of the project.

Handwritten digit recognition has gained so much popularity from the aspiring beginner of machine learning and deep learning to an expert who has been practicing for years. Developing such a system includes a machine to understand and classify the images of handwritten digits as 10 digits (0–9). The dataset is considered as too simple for testing the modern, very complex deep learning models with up to billions of parameters. However, the dataset is still useful. For instance it is useful for quickly testing new implementation of algorithms. If the tested model achieves a high accuracy on MNIST chances are that the implementation is correct. If the algorithm doesn’t work on MNIST, it won’t work at all.

No preprocessing is required to play with that dataset as all digits are size-normalized and centered in the image.

## 1.4 Scope of the Project

The scope of this work is to recognise digits of different size, width, orientation of digits that are hand written and to get the accurate and error free information of what we want.

## 1.5 Methodology

Each research task requires some measurement, in order to measure the accuracy and performance of the handwritten digits, the MNIST database is used for such purposes. MNIST is a widely used standard for handwritten digital recognition. MNIST is a large and standard website of handwritten digits. The MNIST database is often used as a test for the division of algorithms in the digital manuscript recognition framework.

The first step that will be done is to set up the database, which can be done successfully using the Keras system interface. The MNIST database, an extension of the NIST database, is a low-complexity data collection of handwritten digits used to train and test various supervised machine learning algorithms. The database contains 70,000 28x28 black and white images representing the digits zero through nine.

# CHAPTER 2

# LITERATURE REVIEW

**[1] Shefali Aroraa & M.P.S Bhatia, “Handwriting recognition using Deep Learning in Keras”, International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018.**

In this paper, a Python library known as Keras, is used for classification of MNIST dataset, a database with images of handwritten images. Two architectures – feed forward neural networks and convolutional neural networks are used for feature extraction and training of model, which is optimized using Stochastic Gradient Descent. This paper gives an overview of multi-class classification of these images using these models, and their performance evaluation in terms of various metrics.

**[2] Rohan, Sethi, Ila Kaushik “Hand Written Digit Recognition using Machine Learning”, Publisher: IEEE Published in: 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT).**

Hand-written character and digit recognition have been one of the most exigent and engrossing field of pattern recognition and image processing. The vertical and horizontal projections methods are used for the purpose of segmentation in the proposed work. SVM is applied for recognition and classification, while Convex hull algorithm is applied for feature extraction.

**[3] R. Karakaya, S. Kazan, “Handwritten Digit Recognition Using Machine Learning”, Published 30 October 2020.**

In this study, handwriting digit recognition process has been done with algorithms having different working methods. These algorithms are Support Vector Machine (SVM), Decision Tree, Random Forest, Artificial Neural Networks (ANN), K-Nearest Neighbor (KNN) and K- Means Algorithm.

**[4] S. S.Shinde, D. Bhattacharyya, “Handwritten Character Recognition Using Machine Learning Algorithms”, Conference paper, First Online on 08 June 2021.**

Study of handwritten character recognition phases and various strategies and methods for machine learning and deep learning in this paper. The primary oblique of this paper is to ascertain efficient and trustworthy motion to handwritten character recognition.

**[5]** **Liu, W., Wei, J., & Meng, Q. (2020), “Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition”. 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA).**

**doi:10.1109/aeeca49918.2020.9213482.**

Through the use of Python and the corresponding machine learning library scikit-learn and deep learning framework Tensorflow[16], the KNN, SVM, BP neural network and the CNN were experimentally simulated, and the recognition rate under the best parameters was obtained. For **KNN** algorithm, recognition rate is 94.6. For **SVM** algorithm, recognition rate is 94.1. For **BP** algorithm, recognition rate is 96.6. For **CNN** algorithm, recognition rate is 97.7.

**[6]Yellapragada SS Bharadwaj , Rajaram P , Sriram V.P, Sudhakar S, Kolla Bhanu Prakash. “Effective Handwritten Digit Recognition using Deep Convolution Neural Network”.** **Volume 9 No.2, March-April 2020.IJRASET. https://doi.org/10.30534/ijatcse/2020/66922020**

This paper proposed a simple neural network approach towards handwritten digit recognition using convolution. With machine learning algorithms like KNN,SVM/SOM, recognizing digits is considered as one of the unsolvable tasks due to its distinctiveness in the style of writing. In this paper, Convolution Neural Networks are implemented with an MNIST dataset of 70000 digits with 250 distinct forms of writings. The proposed method achieved 98.51% accuracy for real-world handwritten digit prediction with less than 0.1 % loss on training with 60000 digits while 10000 under validation.

**[7] Priyanshu Singh , Pranali Pawar, Nikhil Raj, “Handwritten Digit Recognition”. International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022- Available at www.ijraset.com**

The purpose of this project is to use the classification algorithm to identify handwritten digits. Background results are probably the most widely used Machine Learning Algorithms such as SVM, KNN and RFC and in-depth reading calculations like CNN multilayer using Keras and Theano and Tensorflow. Using these, 98.70% accuracy was used by CNN (Keras + Theano) compared to 97.91% using SVM, 96.67% using KNN, 96.89% using RFC was obtained.

**[8] Ciresan, D. et al. “Multi-column deep neural networks for image classification.” 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012): 3642-3649.**

Traditional methods of computer vision and machine learning cannot match human performance on tasks such as the recognition of handwritten digits or traffic signs. Our biologically plausible deep artificial neural network architectures can. Small (often minimal) receptive fields of convolutional winner-take-all neurons yield large network depth, resulting in roughly as many sparsely connected neural layers as found in mammals between retina and visual cortex. Only winner neurons are trained. Several deep neural columns become experts on inputs preprocessed in different ways; their predictions are averaged. Graphics cards allow for fast training. On the very competitive MNIST handwriting benchmark, our method is the first to achieve near-human performance.

[9] **Niu n Xiao-Xiao and Y. Suen Ching, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits", Elsevier, vol. 45, no. 4, pp. 1318-1325, April 2012.**

This paper presents a hybrid model of integrating the synergy of two superior classifiers: Convolutional Neural Network (CNN) and Support Vector Machine (SVM), which have proven results in recognizing different types of patterns. In this model, CNN works as a trainable feature extractor and SVM performs as a recognizer. This hybrid model automatically extracts features from the raw images and generates the predictions. Experiments have been conducted on the well-known MNIST digit database. Comparisons with other studies on the same database indicate that this fusion has achieved better results: a recognition rate of 99.81% without rejection, and a recognition rate of 94.40% with 5.60% rejection. These performances have been analyzed with reference to those by human subjects.

**[10] Ahlawat S, Choudhary A, Nayyar A, Singh S, Yoon B. “Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)”. Sensors. 2020; 20(12):3344. https://doi.org/10.3390/s20123344**

Convolutional neural networks (CNNs) are very effective in perceiving the structure of handwritten characters/words in ways that help in automatic extraction of distinct features and make CNN the most suitable approach for solving handwriting recognition problems. Our aim in the proposed work is to explore the various design options like number of layers, stride size, receptive field, kernel size, padding and dilution for CNN-based handwritten digit recognition. In addition, we aim to evaluate various SGD optimization algorithms in improving the performance of handwritten digit recognition.

**[11] Shikha Gupta. “Newbie’s Deep Learning Project to Recognize Handwritten Digit” Published on November 26, 2021.**

In this article, we are going to use the MNIST dataset for the implementation of a handwritten digit recognition app. To implement this we will use a special type of deep neural network called *Convolutional Neural Networks*. In the end, we will also build a Graphical user interface(GUI) where you can directly draw the digit and recognize it straight away.

**[12] Keerthana Buvaneshwaran. “MNIST Handwritten Digit Classification using Deep Learning” Published on June 12, 2022**

In this article, we are gonna build a project which is the MNIST handwritten digit classification using deep learning. The handwritten digit classification is the ability of the model to recognize the human handwritten digits. The handwritten digits may have different size and orientation since it differs from person to person in writing style. Hence it is a hard task for machines to recognize handwritten digits. Handwritten digit classification is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

**[13] Milind Soorya. “ MNIST Handwritten digit classification using tensorflow.” Published on September 16 2021.**

In this article, we will look at the MNIST dataset and create a simple neural network using TensorFlow and Keras. Later we will also add a hidden layer to make the model more accurate. Handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can vary from person to person. Handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

**[14] Drishti Beohar, Akhtar Rasool. “ Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network(ANN) and Convolutional Neural Network(CNN).”. Publisher: IEEE.**

Handwritten digit recognition is an intricate assignment that is vital for developing applications, in computer vision digit recognition is one of the major applications. There has been a copious exploration done in the Handwritten Character Recognition utilizing different deep learning models. Deep learning is rapidly increasing in demand due to its resemblance to the human brain. The two major Deep learning algorithms Artificial Neural Network and Convolutional Neural Network which have been compared in this paper considering their feature extraction and classification stages of recognition. The models were trained using categorical cross-entropy loss and ADAM optimizer on the MNIST dataset.

**[15] Ritik Dixit, Rishika Kushwah, Samay Pashine. “ Handwritten Digit Recognition using Machine and Deep Leaarning Algorithms”.**

This paper provides a reasonable understanding of machine learning and deep learning algorithms like SVM, CNN, and MLP for handwritten digit recognition. It furthermore gives you the information about which algorithm is efficient in performing the task of digit recognition. In further sections of this paper will be discussing the related work that has been done in this field followed by the methodology and implementation of all the three algorithms for the fairer understanding of them.

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended dataset

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended dataset.

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended dataset.

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended dataset.

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended dataset

The MNIST digit dataset is used to

evaluate the experiments, which are then compared to

different methods. On the MNIST dataset, 99.47 percent test

accuracy was attained, which is superior to other

approaches. The research was then expanded upon by the

addition of a new dataset for recognizing English capital

letters. 98.94 percent accuracy was achieved on this

extended datase

**CHAPTER 3**

# PROJECT DESCRIPTION

## 3.1 Existing System

Image Recognition is an important research area for most used apps. In general, in the field of pattern recognition, one of the most difficult tasks is the precise computerization of human handwriting. Without a doubt, this is a very difficult subject because there are so many variations of handwriting from person to person. Despite the fact that, this difference does not cause problems for humans, however, it is becoming increasingly difficult to instruct computers to interpret common handwriting. In the case of image recognition, for example, classification by hand, it is important to know how the information is displayed in the pictures.

In addition, people create the same digit with different ideas, the diversity and diversity in the handwriting of different people also contributes to the development and existence of digits.

## 3.2 Proposed System

Digital recognition is also remarkable an important issue. As handwritten digits are not a same size, thickness, position and direction, in this case by the way, various difficulties should be considered find the handwritten digital recognition problem. I unique and a variety of creative styles for different people moreover have an influence on the model as well the presence of digits. It is a strategy to see again edit written digits. It has a wide variety applications, for example, scheduled bank checks, post offices and tax documents and so on. The purpose of this project is to use the classification algorithm to identify handwritten digits. Background results are probably the most widely used Machine and Deep Learning Algorithms such as SVM, KNN and RFC and in-depth reading calculations like CNN multilayer using Keras and Theano and Tensorflow. Using these, 98.70% accuracy was used by CNN (Keras + Theano) compared to 97.91% using SVM, 96.67% using KNN, 96.89% using RFC was obtained.

#### 3.2.1 Advantages

* To achieve more accurate information.
* Easy recognition of digits especially for bank check processing, numberplate digit recognition etc

**3.3 Feasibility Study**

A Feasibility study is carried out to check the viability of the project and to analyze the strengths and weaknesses of the proposed system. The application of usage of mask in crowd areas must be evaluated. The feasibility study is carried out in three forms

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

#### 3.3.1 Economic Feasibility

The proposed system does not require any high cost equipment.

This project can be developed within the available software.

#### 3.3.2 Technical Feasibility

The proposed system is completely a Deep learning model. The main tools used in this project are Anaconda prompt, Pycharm, Keras dataset, Jupyter Notebook and the language used to execute the process is Python. The above mentioned tools are available for free and technical skills required to use this tools are practicable. From this we can conclude that the project is technically feasible.

#### 3.3.3 Social Feasibility

Social feasibility is a determination of whether project will be acceptable or not. Our project is Eco-friendly for society and there is no social issues. Our project must not threatened by the system instead must accept it as a necessity. Since our project is applicable for every individual in the society to take care about the society and environment. The level of the acceptance of System is very high and it depends on the methods deployed in the system. our system is highly familiar with the society.

**3.4 System Specification**

#### 3.4.1 Hardware Specification

* 256 GB SSD
* CPU QUAD CORES

**3.4.2 Software Specification**

* Tensorflow
* Python 3.7
* Anaconda3 5.3.1
* Jupyter Notebook

**CHAPTER 4**

**MODULE DESCRIPTION**

* 1. **System Architecture**

The purpose is to look at the design possibilities of the proposed system, such as architectural design, block diagram, sequence diagram, data flow diagram and system user interface design to define steps such as pre-processing, output, partition, segmentation and digital recognition.

Figure shows a diagram of the properties of the proposed system. The proposed model consists of four stages for planning and obtaining digits:

1. Preliminary consideration
2. Separation
3. Feature Domain
4. Separation and Recognition

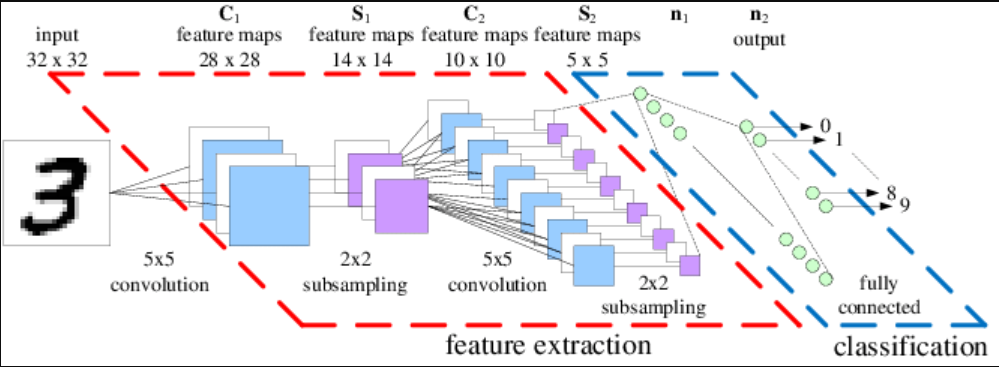


Fig.No:4.1 System Architecture

* 1. **Design Phase**

#### 4.2.1 Data Flow Diagram

Part of the pre-processing step that performs various functions in the input image. Enhances the image by making it a reason for the split. This method is to teach a set of images to be processed to reduce data, by combining it into a binary image. Figure shows a sample of photos taken on the MNIST website.

Once the pre-processing of the input images is complete, individual digital images are created from the sequence of images. Pre-processed digital images are subdivided into individual digital images, which are assigned to each digit. Each digit is resized to pixels. In this step the method of obtaining a limit on dividing a database image is used.

After the completion of the pre-processing phase and the separation phase, the pre-processed images are represented in the form of a matrix containing pixels of images of the largest size. In this way it will be useful to represent the digits in the images that contain the required information. This function is called feature removal. In the extraction feature the data reuse is deleted.

In the process of separation and recognition the vectors of the extracted element are considered the death of the individual in the subsequent segmentation. In order to model the operating system the output components are grouped and defined using the following three categories:

* K-Close Neighbor
* Random Forest Classifier
* Vector Support Machine

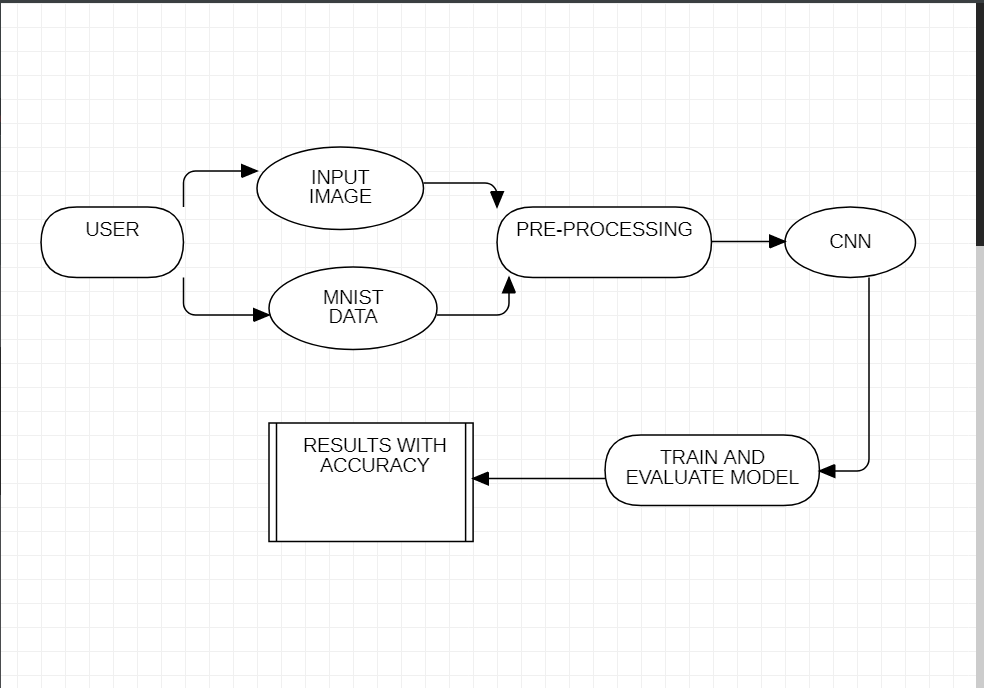
****

Fig.No:4.2.1 Data Flow Diagram

**4.2.2 Flowchart**

Figure 4.2.2 represent the UML diagram of our model. Data will be extracted from the data source and background noises will be removed. Next step is segmentation of image into partitions, after that the image will be classified and features will be extracted using character geometry or using gradient features. After extracting the digit from the image it will be displayed in the digit file.

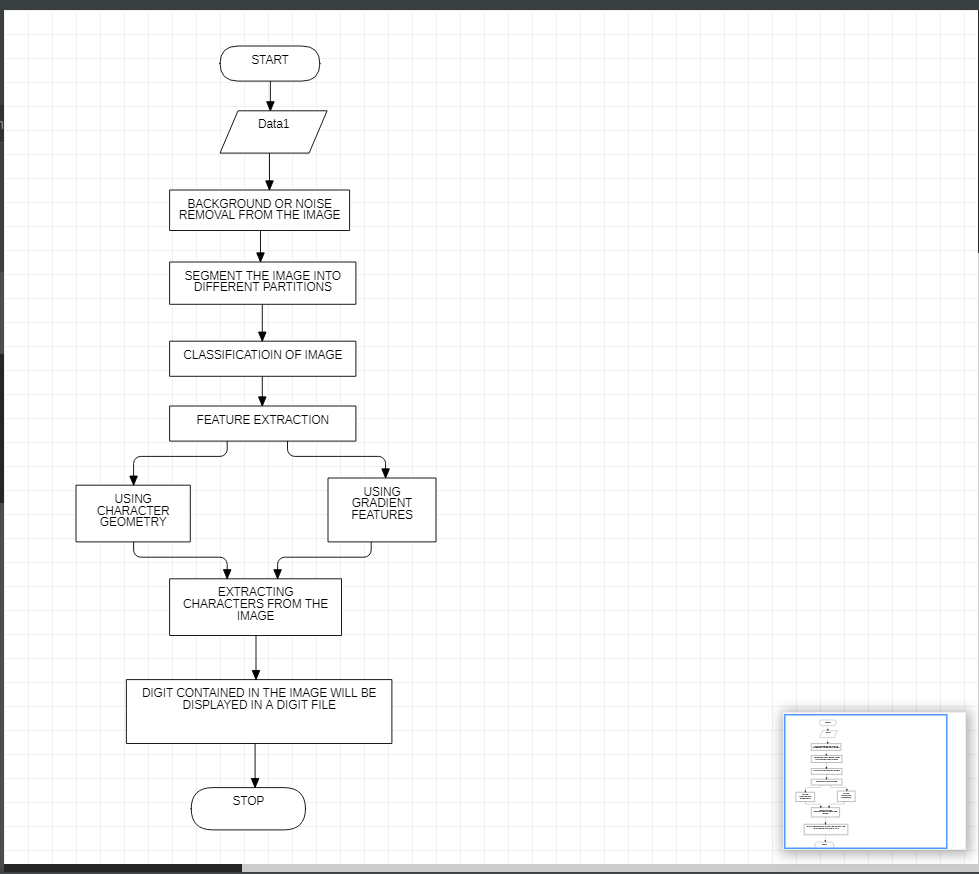


Fig.No:4.2.2 Flowchart

#### 4.2.3 Use Case Diagram

Figure 4.2.3 represents the Use Case diagram of our model. The image is uploaded by the user. The system then pre - process the image that includes converting to grey scale or grey scale to binary format or normalization. And further segmentation and recognition. Finally, output is generated.

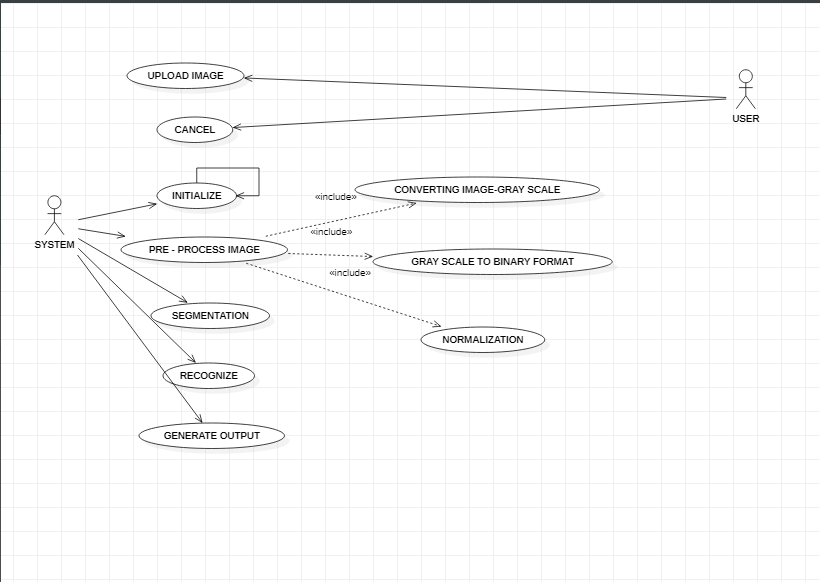


Fig.No:4.2.3 Use Case Diagram

**4.2.4 ACTIVITY DIAGRAM**

Figure 4.2.4 represents the Activity diagram of our model. When the process is initialized, the image is given as input. Then pre-processing of image is done then segmentation and clipping of image is done. After that feature extraction process starts then training and recognition for image takes place. That’s helps to generate output.

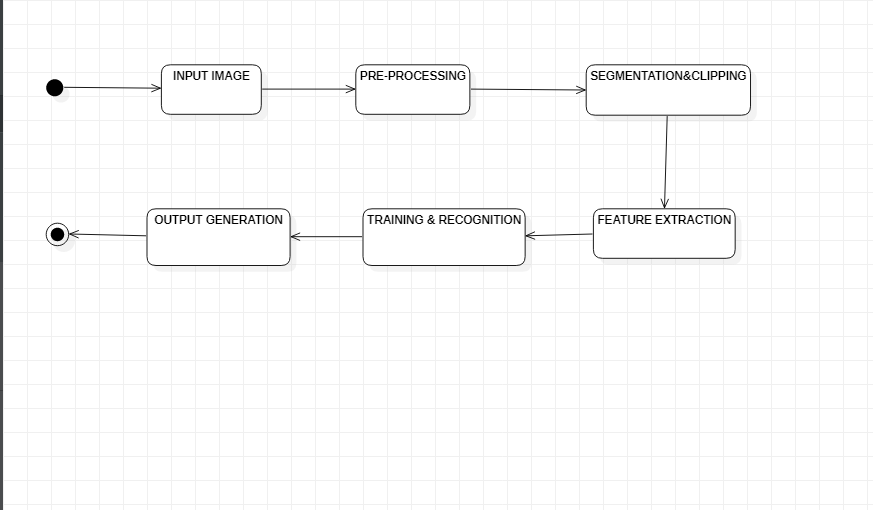


Fig.No:4.2.4 Activity Diagram

**4.3 Module Description**

In this module for real-world image classification prediction, we need to do an image pre-processing on the real-world images as model training was done with greyscale raster images. The steps of image pre-processing are

1. Loading image

2. Convert the image to greyscale

3. Resize the image to 28x28

4. Converting the image into a matrix form

5. Reshape the matrix into 28x28x1

After pre-processing, we predict the label of the image by passing the pre-processed image through the machine and deep learning algorithms. The output we get is a list of 10 activation values 0 to 9, respectively. The position having the highest value is the predicted label for the image.

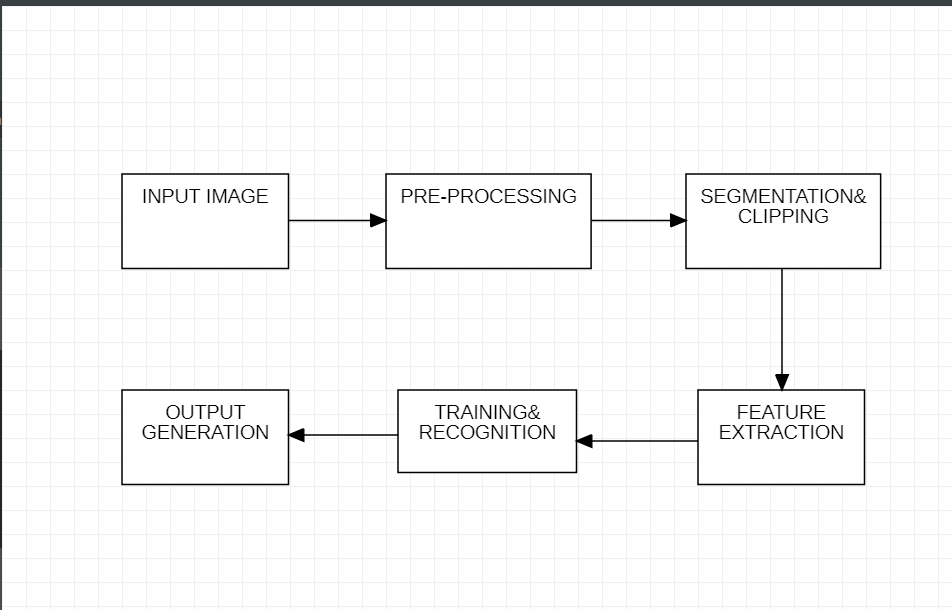
****

Fig.No:4.3 Module Description

**CHAPTER 5**

**IMPLEMENTATION AND TESTING**

**5.1 Input**

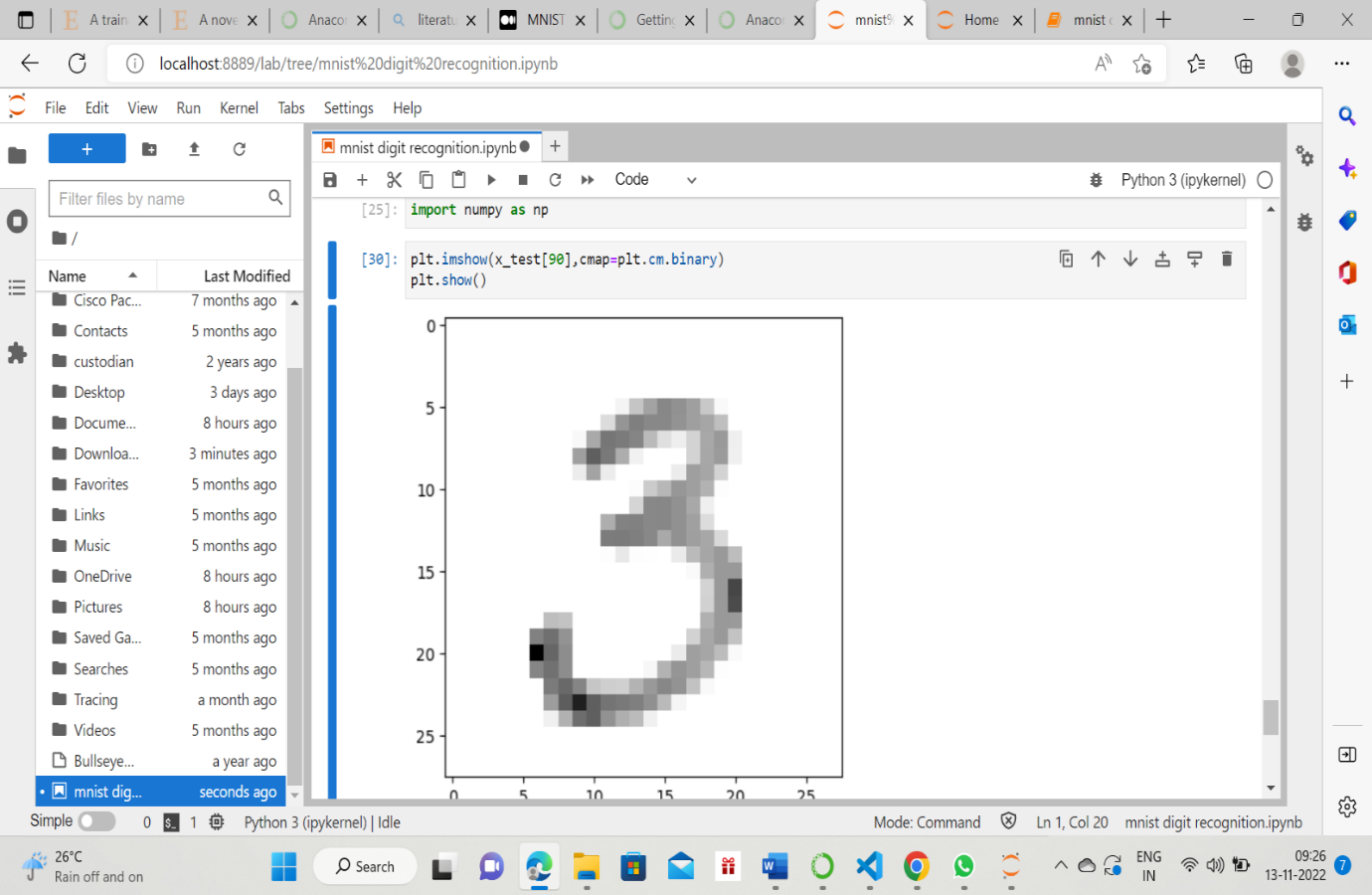
****

Fig.No:5.1 Input Image

**5.2 Testing**

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

**5.2.1 Types of Testing**

* Unit Testing
* Integration Testing
* Functional Testing
* White Box Testing
* Black Box Testing

**5.2.2 Unit testing**

Unit Testing is a beneficial software testing method where the units of source code is tested to check the efficiency and correctness of the program.

**Code for Unit testing**

import tensorflow astf

import matplotlib.pyplotasplt

mnist = tf.keras.datasets.mnist

(x\_train,y\_train),(x\_test,y\_test)=mnist.load\_data()

plt.imshow(x\_train[4],cmap=plt.cm.binary)

plt.show()

x\_train[0]

x\_train=tf.keras.utils.normalize(x\_train,axis=1)

x\_test=tf.keras.utils.normalize(x\_test,axis=1)

##### Test Results

Here, the above piece of code is able to load all the data from dataset and is able to categorize into training and testing. Later normalization of the data is also successfully done.

#### 5.2.3 Integration testing

It is defined as a type of testing where software modules are integrated logically and tested as a group.

**Code for Integration testing**

from tkinter import \*

import tkinter as tk

import win32gui

import PIL

from PIL import imageGrab, imagetk,image

import io

from keras.models import load\_model

model=load\_model(r'c:\users\vyshu\Desktop\digitrecognized\_model.h5')

##### Test Results

All the libraries are imported here which is very much necessary for integrating the model that is created for recognising the digits to this GUI code based on tkinter.

**5.2.4 Functional testing**

It is a type of software testing that validates the software system against the functional requirements/specifications.

**Code for Functional testing**

def drawrect(event):

x,y=event.x,event.y

r=8

canvas.createoval(x-r,y-r,x+r,y+r,fill="black")

classifybtn.configure(state=NORMAL)

6

def destroycanvas():

root.destroy()

defclearcanvas():

classifybtn.configure(state=DISABLED)

classifybtn.configure(state=DISABLED)

12canvas.delete("all")

try:

cleard by below destroy()function

label.destroy()

except:

1pass

def classify handwritting():

globallabq,ab2

im=imageGrab.grab(bbox=(921,130,1400,510))

im.save('./res/nitzz.png','PNG')

digit=predigit(im)

#im.show()

#digit,acc=predigit(im)

lab1=tk.label(root,text='predictedDigits:'+str(digit),width=24,height=2,fg="#3e7d75",bg="black",font=('Lucida Typewriter',16,'blod'))

lab1.place(x=10,y=420)

def preddigit(img):

image=cv2.imread('./res/nitzz.png')

grey=cv2.cvt color(image.copy(),cv2.COLOR BGR2GRAY)

returns,thresh=cv2.threshold(grey.copy(),75,255,cv2.THRESH BINARY INV)

contours,hierachy=cv2.findContour(thresh.copy(),cv2.RETR EXTERNAL,cv2.CHAIN APPROX SIMPLE)

pre processed digits=[]

##### Test Results:

##### 

##### All the functions that resulted to bring nice GUI were successfully built in all the aspects for the proper execution.

##### 

**5.2.5 White Box testing**

**Code for White Box testing**

model=tf.keras.models.sequential()

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(10,activation=tf.nn.softmax))

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

model.fit(xtrain,ytrain,epochs=10)

valloss,valacc=model.evalute(xtest,ytest)

valloss

valacc

model.save(r'c:\users\vyshu\Desktop\digitrecognized\_model.h5')

new\_model=tf.keras.models.loadmodel(r'c:\users\vyshu\Desktop\digitrecognized\_model.h5')

predicta=newmodel.predict(xtest)

predict[0]

import numpy as np

plt.imshow(x\_test[11],cmap=plt.cm.binary)

plt.show()

**Test Results:**

Since white box testing is testing of the internal structure and working of model. So above code is resulted as the main working which is based on algorithm and this is properly working and tested fine with random inputs.

**5.2.6. Black Box testing**

**Code for Black Box testing**

def drawrect(event):

x,y=event.x,event.y

r=8

canvas.creatoval(x-r,y-r,x+r,y+r,fill="black")

calssifybin configure(state=NORMAL)

def destroy cnvas():

root.destroy()

def clearcanvas():

classifybtn.configure(state=DISABLED)

canvas.delete("all")

try:

lab1.destroy()

except:

pass

**Test Results:**

Here this testing role is to examine a particular function. Therefore this set of functions are examined properly so that the output could be generated without any error.

**5.3 Testing Strategy**

* Unit testing: Unit testing verifies the bits of code to check the viability of the code.
* Integration testing: Integration testing is carried out to the efficiency of the model with functional requirements.
* Functional testing: The functional testing is done to verify the output with the provided input against the functional requirements.

**CHAPTER 6**

**RESULTS AND DISCUSSIONS**

* 1. **Efficiency of the Proposed System**

* MNIST Handwritten digit recognition is used to develop In-depth learning strategies.
* Many widely used machine and deep learning algorithms, KNN, SVM, RFC and CNN trained and tested on the same data into finding comparisons between dividers.
* Using these deeper learning methods, the higher the level of accuracy can be found.
* Compared to other research methods, this method focuses on which category works best for developing more than 99% separation accuracy models.
* Use Keras as backend and Tensorflow as software, CNN
* The model is able to provide about 98.72% accuracy.

**Accuracy using Machine and Deep Learning Algorithms:**

i) K Nearest Neighbors: 96.67%

ii) SVM: 97.91%

iii) Random Forest Classifier: 96.82%

**6.2 Comparison of Existing and Proposed System**

In Existing System various algorithms used for implementing handwritten digit recognition systems consist of Proximal Support Vector Machine (PSVM), Multilayer Perceptron, Support Vector Machine (SVM), Random forest and many more.

According to previous research work, accuracies provided by these algorithms are of the order of:

* Proximal SVM - 98%
* Random Forest - 85%
* SVM - 87%
* Multilayer Perceptron – 90%

Even though these algorithms may prove to be useful in some of the applications based on this technology, many other applications such as banking industry applications require better results which can be achieved using other algorithms as compared to the algorithms that are mentioned above.

In proposed System as we concentrated mainly on Deep learning, the algorithms used are K Nearest Neighbors, SVM, Random Forest Classifier

Accuracy using Machine and Deep Learning Algorithms:

* K Nearest Neighbors: 96.67%
* SVM: 97.91%
* Random Forest Classifier: 96.82%

So, in proposed system we gey more accurate results than in existing system. And this is also useful in many applications such as banking industry applications that did not provide accurate information in existing system.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**7.1 Conclusion**

Handwritten digit recognition using deep learning methods has been implemented. The most widely used algorithm CNN has been trained and tested on a MNIST dataset. Utilizing these deep learning techniques, the high amount of accuracy is obtained. Using Keras as backend and Tensorflow as a software, a CNN model is able to give accuracy of about 99.5% in the training and 97.6% in testing.

The goal of this project is to explore the field of deep learning and try to come up with some techniques that could be used without going into deep computations, and even if the final result is not very reliable, it still provides an accuracy way better than random.

**7.2 Future Enhancements**

Further implementations is to make the model test with the robotics or use in creating a artificial brain. As this model could be a very small part of it. Digit recognition is an excellent prototype problem for learning about neural networks and it gives a great way to develop more advanced techniques of deep learning.

In future, we are planning to develop a real time handwritten digit recognition system.

**CHAPTER 8**

**SOURCE CODE & POSTER PRESENTATION**

**8.1 Sample Code**

pip install numpy

pip install tensorflow

pip install keras

pip install pillow

from keras.models import Sequential

import tensorflow as tf

import matplotlib.pyplot as plt

mnist = tf.keras.datasets.mnist

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

plt.imshow(x\_train[4],cmap=plt.cm.binary)

plt.show()

x\_train[0]

x\_train= tf.keras.utils.normalize(x\_train,axis=1)

x\_test= tf.keras.utils.normalize(x\_test,axis=1)

x\_train[0]

model=tf.keras.models.sequential()

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(10,activation=tf.nn.softmax))

model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])

model.fit(x\_train,y\_train,epochs=10)

val\_loss,val\_acc=model.evaluate(x\_test,y\_test)

val\_loss

val\_acc

model.save(r'c:\vtu14343(6th sem)\minor project\digitrecognized\_model.h5')

new\_model=tf.keras.models.load\_model(r'c:\vtu14343(6th sem)\minor project\digitrecognized\_model.h5')

predicta=new\_model.predict(x\_test)

predicta[0]

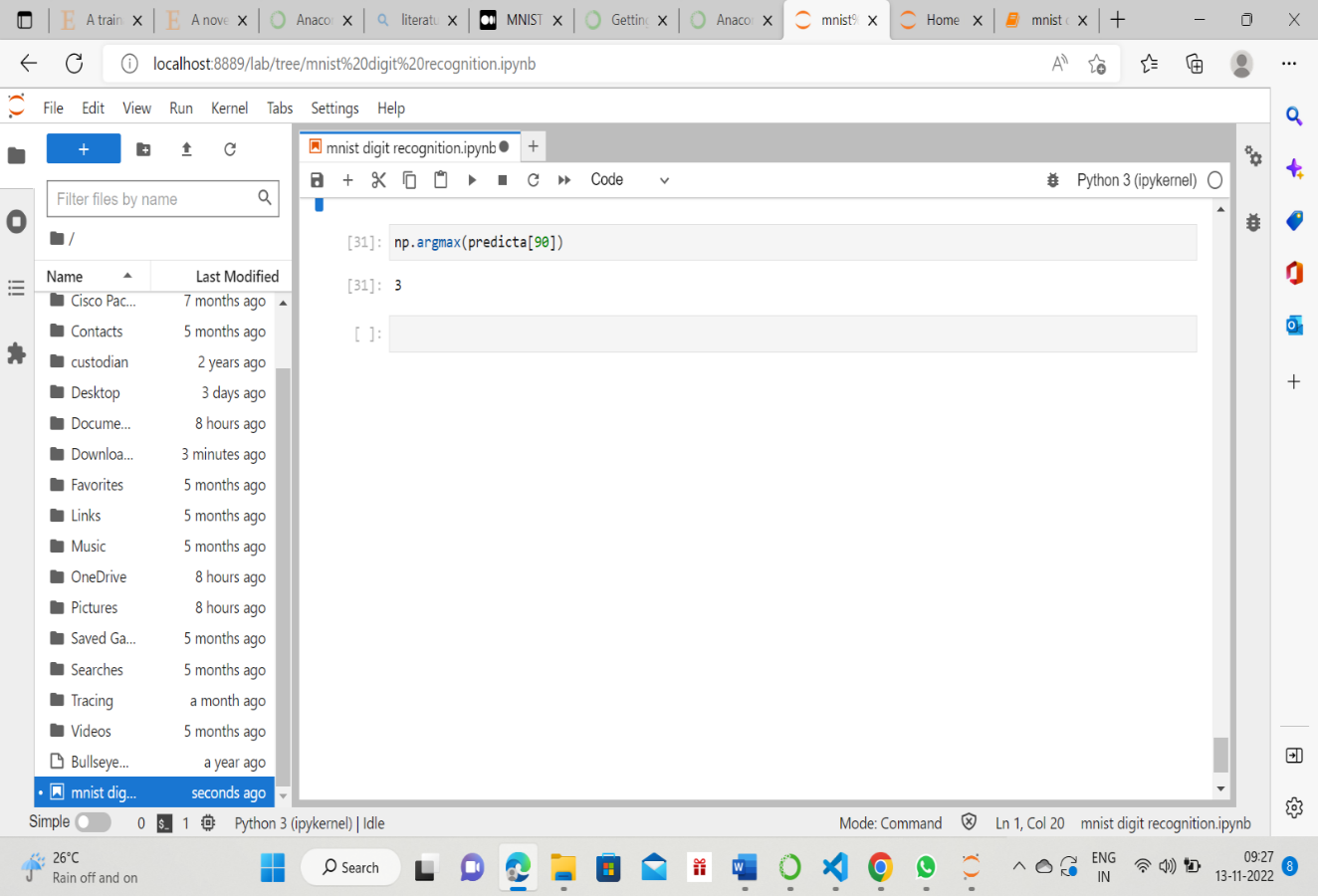
import numpy as np

plt.imshow(x\_test[90],cmap=plt.cm.binary)

plt.show()

np.argmax(predicta[90])

**Output:**

Fig.No:8.1 Generated Output

**8.2 Implementation using GUI tkinter module:**

**Source code:**

from keras.models import load\_model  
from tkinter import \*  
import tkinter as tk  
#import win32gui  
from PIL import ImageGrab, Image  
import numpy as np  
model = load\_model(r'C:\Users\vyshu\Desktop\digitrecognized\_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 = np.array(img)  
 #reshaping for model normalization  
 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=200, height=200, bg = "black", cursor="cross")  
 self.label = tk.Label(self, text="Analyzing..", font=("Helvetica", 48))  
 self.classify\_btn = tk.Button(self, text = "Searched", command = self.classify\_handwriting)  
 self.button\_clear = tk.Button(self, text = "Dlt", 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("", self.start\_pos)  
 self.canvas.bind("", self.draw\_lines(self))  
 def clear\_all(self):  
 self.canvas.delete("all")  
def classify\_handwriting(self):  
 Hd = self.canvas.winfo\_id() # to fetch the handle of the canvas  
 rect = win32gui.GetWindowRect(Hd) # to fetch the edges 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(slf, event):  
 slf.x = event.x  
 slf.y = event.y  
 r=8  
 slf.canvas.create\_oval(slf.x-r, slf.y-r, slf.x + r, slf.y + r, fill='black')  
app = App()  
mainloop()

**Output:**

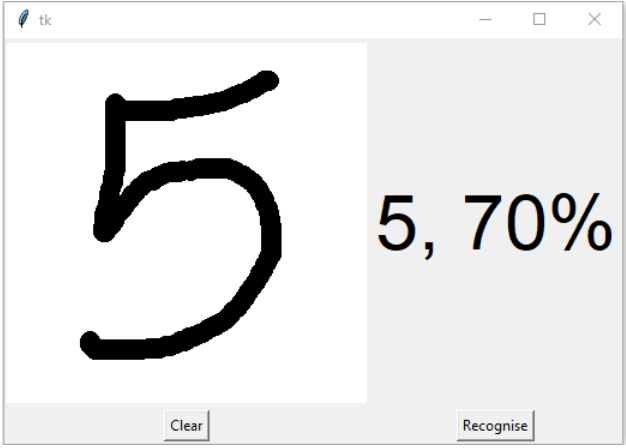
****

Fig.No: 8.2 GUI Module Output

**REFERENCE:**

1. Shefali Aroraa & M.P.S Bhatia, “Handwriting recognition using Deep Learning in Keras”, International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018.
2. Rohan, Sethi, Ila Kaushik “Hand Written Digit Recognition using Machine Learning”, Publisher: IEEE Published in: 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT).
3. R. Karakaya, S. Kazan, “Handwritten Digit Recognition Using Machine Learning”, Published 30 October 2020.
4. S. S.Shinde, D. Bhattacharyya, “Handwritten Character Recognition Using Machine Learning Algorithms”, Conference paper, First Online on 08 June 2021.
5. Liu, W., Wei, J., & Meng, Q. (2020), “Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition”. 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA).
6. Yellapragada SS Bharadwaj , Rajaram P , Sriram V.P, Sudhakar S, Kolla Bhanu Prakash. “Effective Handwritten Digit Recognition using Deep Convolution Neural Network”. Volume 9 No.2, March-April 2020.IJRASET.
7. Ciresan, D. et al. “Multi-column deep neural networks for image classification.” 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012): 3642-3649.
8. Niu n Xiao-Xiao and Y. Suen Ching, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits", Elsevier, vol. 45, no. 4, pp. 1318-1325, April 2012.
9. Priyanshu Singh , Pranali Pawar, Nikhil Raj, “Handwritten Digit Recognition”. International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022- Available at www.ijraset.com
10. Shikha Gupta. “Newbie’s Deep Learning Project to Recognize Handwritten Digit” Published on November 26, 2021.
11. Keerthana Buvaneshwaran. “MNIST Handwritten Digit Classification using Deep Learning” Published on June 12, 2022
12. Milind Soorya. “ MNIST Handwritten digit classification using tensorflow.” Published on September 16 2021.
13. Drishti Beohar, Akhtar Rasool. “ Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network(ANN) and Convolutional Neural Network(CNN).”. Publisher: IEEE.
14. Ritik Dixit, Rishika Kushwah, Samay Pashine. “ Handwritten Digit Recognition using Machine and Deep Leaarning Algorithms”.
15. https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/#:~:text=Learning%20with%20Anaconda-,MNIST%20Handwritten%20Digit%20Classification%20Dataset,digits%20between%200%20and%209
16. <https://towardsdatascience.com/mnist-handwritten-digits-classification-from-scratch-using-python-numpy-b08e401c4dab>
17. <https://www.analyticsvidhya.com/blog/2021/11/newbies-deep-learning-project-to-recognize-handwritten-digit\>
18. <https://medium.com/analytics-vidhya/handwritten-digit-recognition-gui-app-46e3d7b37287>
19. <https://dev.to/milindsoorya/mnist-handwritten-digit-classification-using-tensorflow-3k7d>