A MINOR PROJECT REPORT

ON

HANDWRITTEN DIGIT RECOGNITION

Submitted by

M.Dhanaraj 18A51A0596

B .Tech – Computer Science and Engineering (5th sem)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(An Autonomous Institution)

K. Kotturu , Tekkali-532201 , Srikakulam dist. (A.P)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Minor Project work entitled as **"Handwritten Digit Recognition"** is carried out by**V.Lalitha(18A51A05B9), M.Dhanaraj(18A51A0596),V.Divya(18A51A05B8), R.Mani (18A51A05A6),R.Abitha(18A51A05A5) , K.Jeevana jyothi(18A51A0591) ,** a Students of B.Tech 3rd Year 1st Semester during the academic year 2020-21 in the **Department of Computer Science and Engineering.** The work is carried out by her under my guidance and supervision.

Signature of the Head Of the Department

Dr .G .S .N Murthy M. Tech Ph . D,

Department of CSE

HANDWRITTEN DIGIT RECOGNITION

USING PYTHON

ABSTRACT

Digit Recognition is a noteworthy and important issue. As the manually written digits are not of a similar size, thickness, position and direction, in this manner, various difficulties must be considered to determine the issue of handwritten digit recognition. The uniqueness and assortment in the composition styles of various individuals additionally influence the example and presence of the digits. It is the strategy for perceiving and arranging transcribed digits. It has a wide range of applications, for example, programmed bank checks, postal locations and tax documents and so on. The aim of this project is to implement a classification algorithm to recognize the handwritten digits. The after effects of probably the most broadly utilized Machine Learning Algorithms like SVM, KNN and RFC and with Deep Learning calculation like multilayer CNN utilizing Keras with Theano and Tensorflow. Utilizing these, the accuracy of 98.70% utilizing CNN (Keras + Theano) when contrasted with 97.91% utilizing SVM, 96.67% utilizing KNN, 96.89% utilizing RFC was obtained.

KEYWORDS:KNN,SVM,RFC,CNN.

INDEX

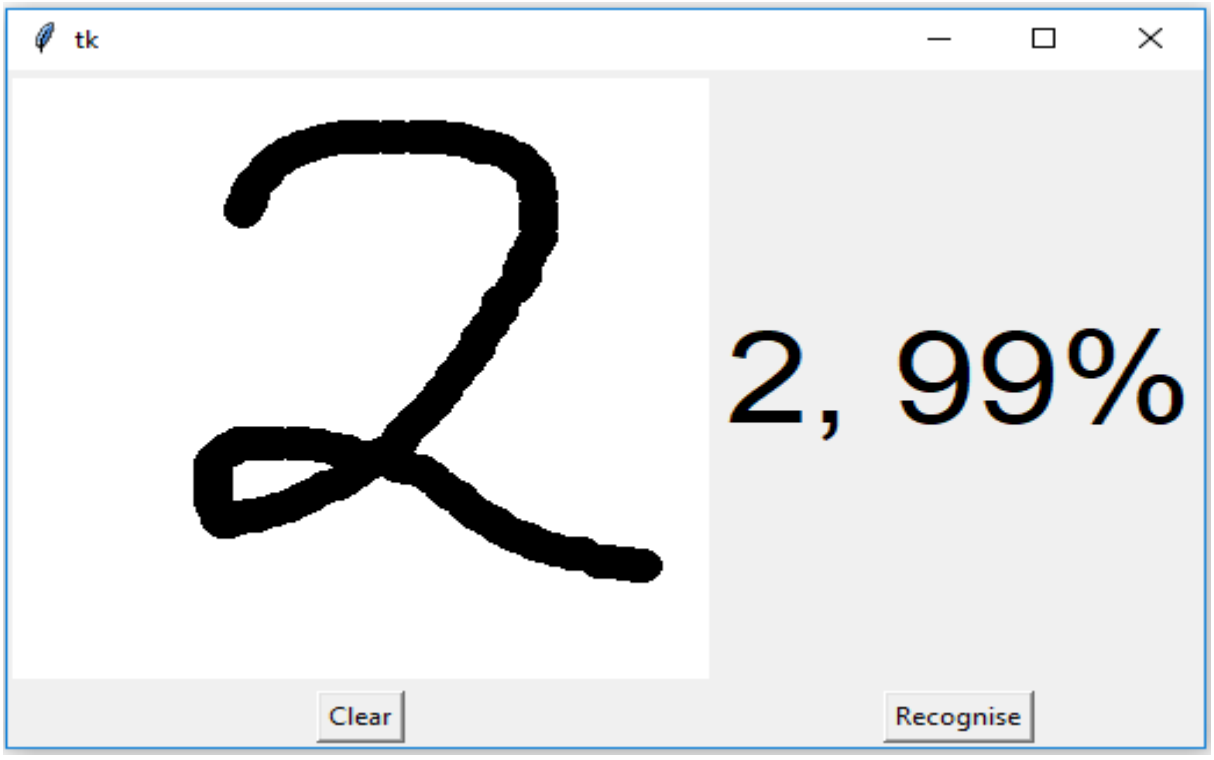
TABLE OF CONTENTS Page no.

1.INTRODUCTION

1.1)What is Handwritten Digit Recognition?

The 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 be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

ABOUT THE PYTHON DEEP LEARNING PROJECT:



In this article, we are going to implement a handwritten digit recognition app using the MNIST dataset. We will be using a special type of deep neural network that is *Convolutional Neural Networks*. In the end, we are going to build a GUI in which you can

draw the digit and recognize it straight away.

Prerequisites:

The interesting Python project requires you to have basic knowledge of Python programming, deep learning with Keras library and the Tkinter library for building GUI.

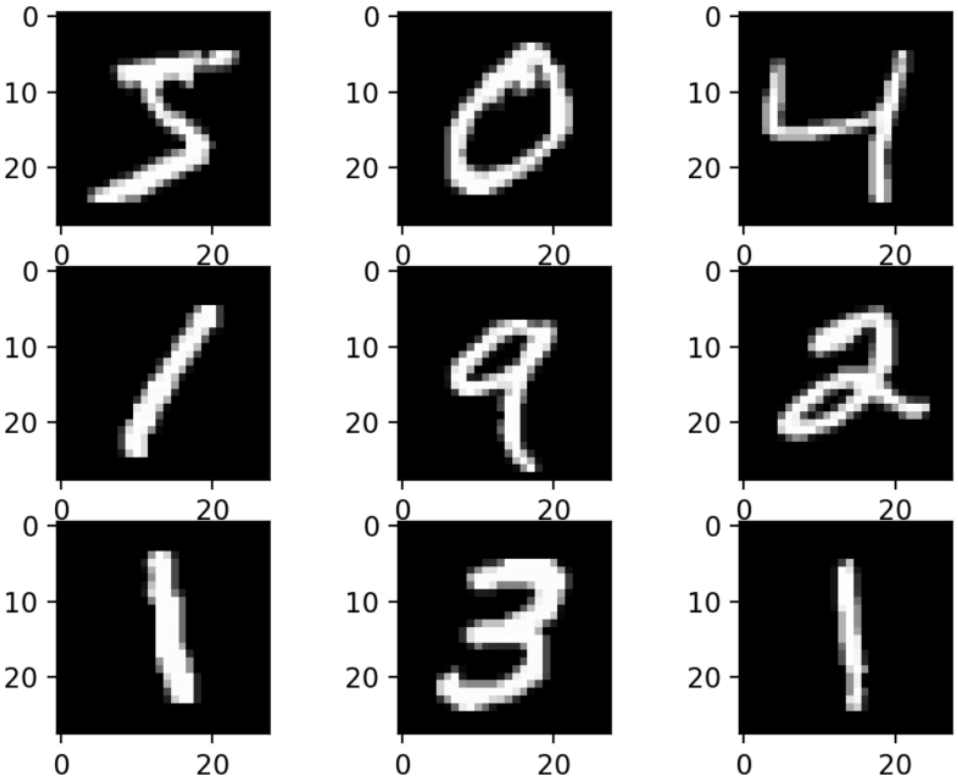
Install the necessary libraries for this project using this command:

```
pip install numpy, tensorflow, keras
```

1.2)The MNIST dataset

This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The mnist dataset contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. So, the MNIST dataset has 10 different classes. The handwritten digits images are represented as a 28×28 matrix where each cell contains grayscale pixel value.
It is a widely used and deeply understood dataset and, for the most part, is "*solved*." Top-performing models are deep learning convolutional neural networks that achieve a classification accuracy of above 99%, with an error rate between 0.4 %and 0.2% on the hold out test dataset.
The example below loads the MNIST dataset using the Keras API and creates a plot of the first nine images in the training dataset.

A plot of the first nine images in the dataset is also created showing the natural handwritten nature of the images to classified.



1.3)Model Evaluation Methodology

Although the MNIST dataset is effectively solved, it can be a useful starting point for developing and practicing a methodology for solving image classification tasks using convolutional neural networks.

Instead of reviewing the literature on well-performing models on the dataset, we can develop a new model from scratch.

The dataset already has a well-defined train and test dataset that we can use.

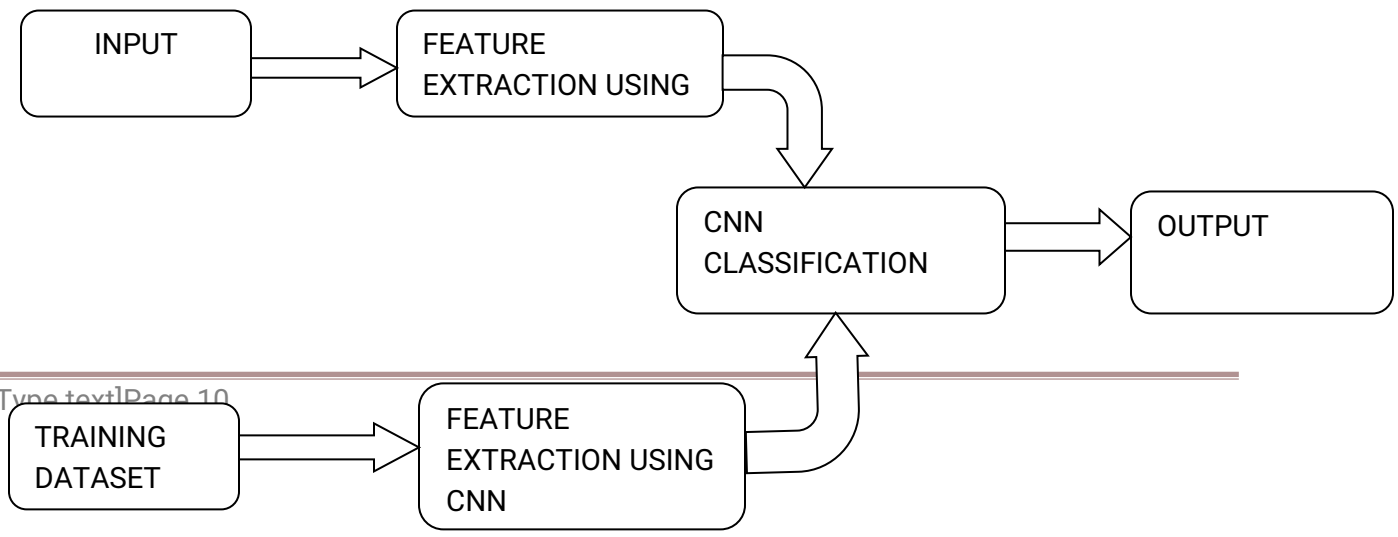
In order to estimate the performance of a model for a given training run, we can further split the training set into a train and

validation dataset. Performance on the train and validation dataset over each run can then be plotted to provide learning curves and insight into how well a model is learning the problem.

In order to estimate the performance of a model on the problem in general, we can use k-fold cross-validation, perhaps five-fold cross-validation. This will give some account of the models variance with both respect to differences in the training and test datasets, and in terms of the stochastic nature of the learning algorithm. The performance of a model can be taken as the mean performance across k-folds, given the standard deviation, that could be used to estimate a confidence interval if desired.
We can use the KFold class from the scikit-learn API to implement the k-fold cross-validation evaluation of a given neural network model. There are many ways to achieve this, although we can choose a flexible approach where the *KFold* class is only used to specify the row indexes used for each spit.

2)ARCHITECTURE

The reason behind this document is to look into the design possibilities of the proposed system, such as architecture design, block diagram, sequence diagram, data flow diagram and user interface design of the system in order HANDWRITTEN NUMBER ANALYSIS to define the steps such as pre-processing, feature extraction, segmentation, classification and recognition of digits.



3.PERFORMANCE REQUIREMENTS:

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system.

4.HARDWARE REQUIREMENTS:

Processor	Intel core i5
Ram	12 GB
Hard disk	1TB

5.SOFTWARE USED:

Software requirements	Version
Windows	10

DEPENDENCIES USED:

Requirements	Version
Numpy	1.18.2

Pandas	1.0.3
Open cv	4.3.0

6)Technical Description:

6.1 Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for

the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to game.

Python Features:

Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read – Python code is more clearly defined and visible to the eyes.

Easy-to-maintain – Python's source code is fairly easy-to-maintain.

A broad standard library – Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Mac into sh.

Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable – Python provides a better structure and support for large programs than shell scripting.

Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient. Databases – Python provides interfaces to all major commercial databases

7.TKINTER:

Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps – • Import the Tkinter module. • Create the GUI application main window. • Add one or more of the above-mentioned widgets to the GUI application. • Enter the main event loop to take action against each event triggered by the user.

Example :

```
#!/usr/bin/python

importTkinter top = Tkinter.Tk()

# Code to add widgets will go here...

top.mainloop() This would create a  window
```

8) Building Python Deep Learning Project on Handwritten Digit Recognition

Below are the steps to implement the handwritten digit recognition project:

8.1. Import the libraries and load the dataset:

First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it.
The mnist.load_data() method returns us the training data, its labels and also the testing data and its labels.

```
import keras
from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

8.2. Preprocess the data:

The image data cannot be fed directly into the model so we need to **perform some operations and process the data** to make it ready for our neural network. The dimension of the training data is (60000,28,28). The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1).

8.3. Create the model:

Now we will create our CNN model in Python data science project. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces offer fitting of the model. We will then compile the model with the Adadelta optimizer.

8.4.Train the model

The **model.fit() function** of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size.
It takes some time to train the model. After training, we save the weights and model definition in the 'mnist.h5' file.

8.5.Evaluate the model

We have 10,000 images in our dataset which will be used to evaluate how good our model works. The testing data was not involved in the training of the data therefore, it is new data for our model. The MNIST dataset is well balanced so we can get around 99% accuracy.

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print(Test accuracy:', score[1])
```

8.6.Create GUI to predict digits:

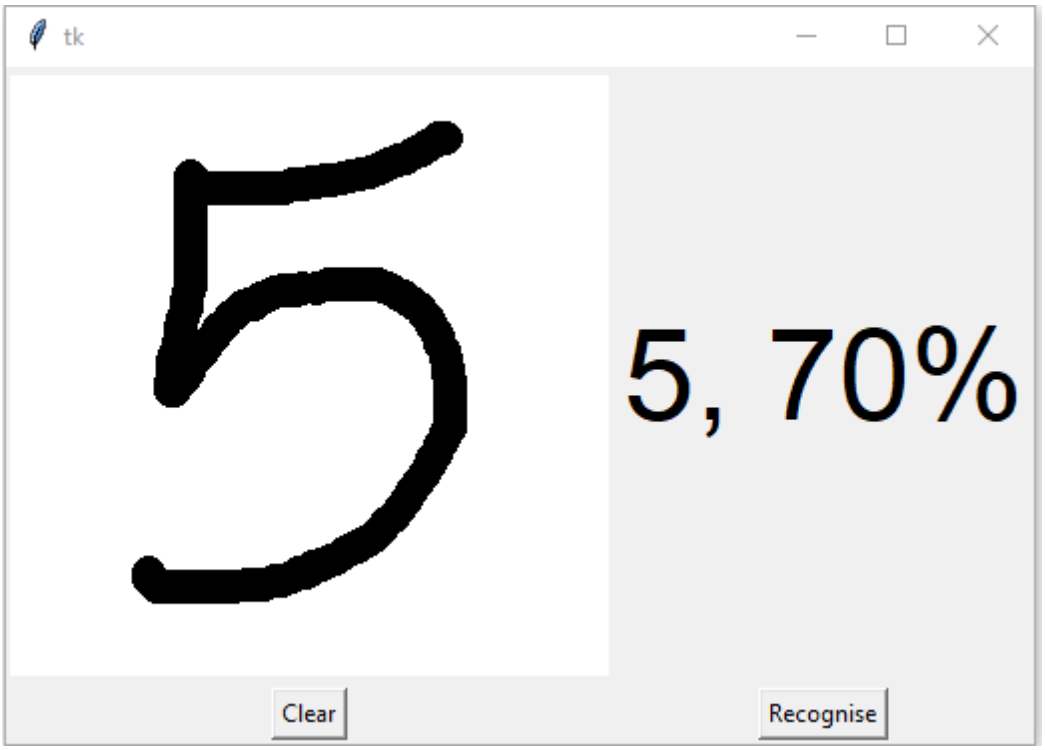
Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function **predict_digit()** that takes the image as input and then uses the trained model to predict the digit.
Then we create the App class which is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict_digit() function and display the results.

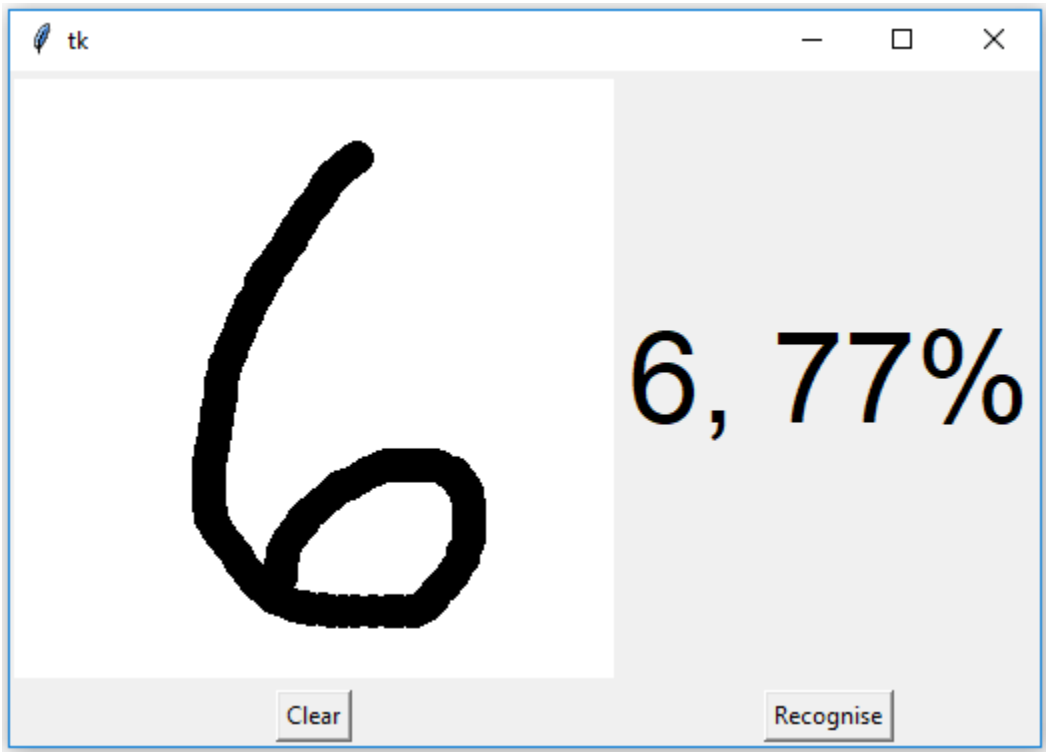
Here's the full code for our gui_digit_recognizer.py file:

```
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('mnist.h5')
def predict_digit(img):
img = img.resize((28,28))
img = img.convert('L')
img = np.array(img)
img = img.reshape(1,28,28,1)
img = img/255.0
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
```

```
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)
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("<B1-Motion>", self.draw_lines)
def clear_all(self):
self.canvas.delete("all")
def classify_handwriting(self):
HWND = self.canvas.winfo_id()
rect = win32gui.GetWindowRect(HWND)
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()
```

8.7.Screenshots:


tk
5, 70%
Clear
Recognise


tk
6, 77%
Clear
Recognise

9.LIMITATION:

The main difficulty in the handwritten digits recognition is different handwritten style which is a very personal behavior where there are a lot of models for numbers based on angles,length of the segments,stress on some parts of numbers,etc.

10.CONCLUSION:

In this paper an extensive review of recent advancement in the field of handwritten numeric digit classification and recognition has been presented. The review presented covers all the aspects for handwritten as well as printed digit recognition like off-line and on-line recognition, different features used, and finally various types of classifies recently used for digit classification. Moreover, all the important and recent works have been discussed with their ssadvantages and limitations. It has been also discussed, most of the available systems were developed for particular database and yet to analyze over real time handwritten digits.