**Handwritten Digit Recognition Using Python**

**GAYATHRI B H 28/10/2021**

**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.

**Abstract**

The aim of a handwriting digit recognition system is to convert handwritten digits into machine readable formats. The main objective of this work is to ensure effective and reliable approaches for recognition of handwritten digits and make banking operations easier and error free.

**How accurate is handwriting recognition?**

Today, OCR technology provides **higher than 99% accuracy** with typed characters in high-quality images. However, the diversity in human writing types, spacing differences, and irregularities of handwriting causes less accurate character recognition.

**About the project**

We are going to implement a handwritten digit recognition app using the MNIST dataset. In the end, we are going to build a GUI in which we can draw the digit and recognize it straight away.

**Prerequisites**

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

The necessary libraries for this project using this command:

pip install numpy, tensorflow, keras, pillow,

**The MNIST dataset**

This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The [MNIST dataset](http://yann.lecun.com/exdb/mnist/) 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.

**Building handwritten digit recognition project**

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

**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 in MNIST also. 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

# the data, split between train and test sets

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

print(x\_train.shape, y\_train.shape)

### 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).

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

# convert class vectors to binary class matrices

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

print('x\_train shape:', x\_train.shape)

print(x\_train.shape[0], 'train samples')

print(x\_test.shape[0], 'test samples')

### 3. Create the model

Creating **CNN model**. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures and that’s 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.

batch\_size = 128

num\_classes = 10

epochs = 10

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3,3),activation='relu',input\_shape=input\_shape))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

### 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.

hist = model.fit(x\_train, y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test))

print("The model has successfully trained")

model.save('mnist.h5')

print("Saving the model as mnist.h5")

### 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 as it is new data for the 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])

### 6. Create GUI to predict digits

Now for the GUI, 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. Creating 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.

**For 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):

#resize image to 28x28 pixels

img = img.resize((28,28))

#convert rgb to grayscale

img = img.convert('L')

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')

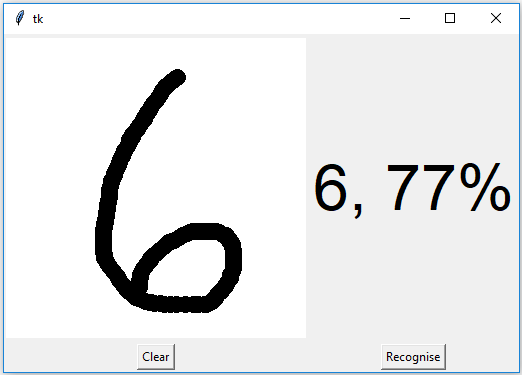
app = App()

mainloop()

**screenshots:**

Graphical user interface, application

Description automatically generated



**Summary**

We have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Convolutional neural network which is very effective for image classification purposes. Later on, we build the GUI where we draw a digit on the canvas then we classify the digit and show the results. You can build an app regarding the same following the steps.