BHAKTI VARADKAR **ROLL NO.: 512**

Machine & Deep Learning – Mini Project

AIM: *Implementation of handwritten digit recognition using MNIST dataset.*

Handwritten digit recognition

This is the process to provide the ability to machines to recognize human handwritten digits. It is not an easy task for the machine because handwritten digits are not perfect, vary from person-to-person, and can be made with many different flavours.

Commands to install the necessary libraries for this project:

- pip install numpy
- pip install keras
- pip install tensorflow
- pip install pillow

The MNIST dataset

MNIST is the most popular dataset for enthusiasts of machine learning and deep learning. Above 60,000 plus training images of handwritten digits from zero to nine and more than 10,000 images for testing are present in the MNIST dataset.

Steps to build Handwritten Digit Recognition System

1. ***Import libraries and dataset***

Import all the needed modules for training our model. Keras library already contains many datasets and MNIST is one of them. We call mnist.load_data() function to get training data with its labels and also the testing data with 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()
```

2. ***The Data Pre-Processing***

Model cannot take the image data directly so we need to perform some basic operations and process the data to make it ready for our neural network. The dimension of the training data is (60000*28*28). One more dimension is needed for the CNN model 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, 10)
y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

3. ***Create the model***

A convolutional layer and pooling layers are the two wheels of a CNN model. The reason behind the success of CNN for image classification problems is its feasibility with grid structured data. We will use the Adadelta optimizer for the model compilation.

```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
```

```
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optim
izers.Adadelta(),metrics=['accuracy'])
```

4. ***Train the model***

To start the training of the model we can simply call the model.fit() function of Keras. It takes the training data, validation data, epochs, and batch size as the parameter.

The training of model takes some time. After succesful model training, we can 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***

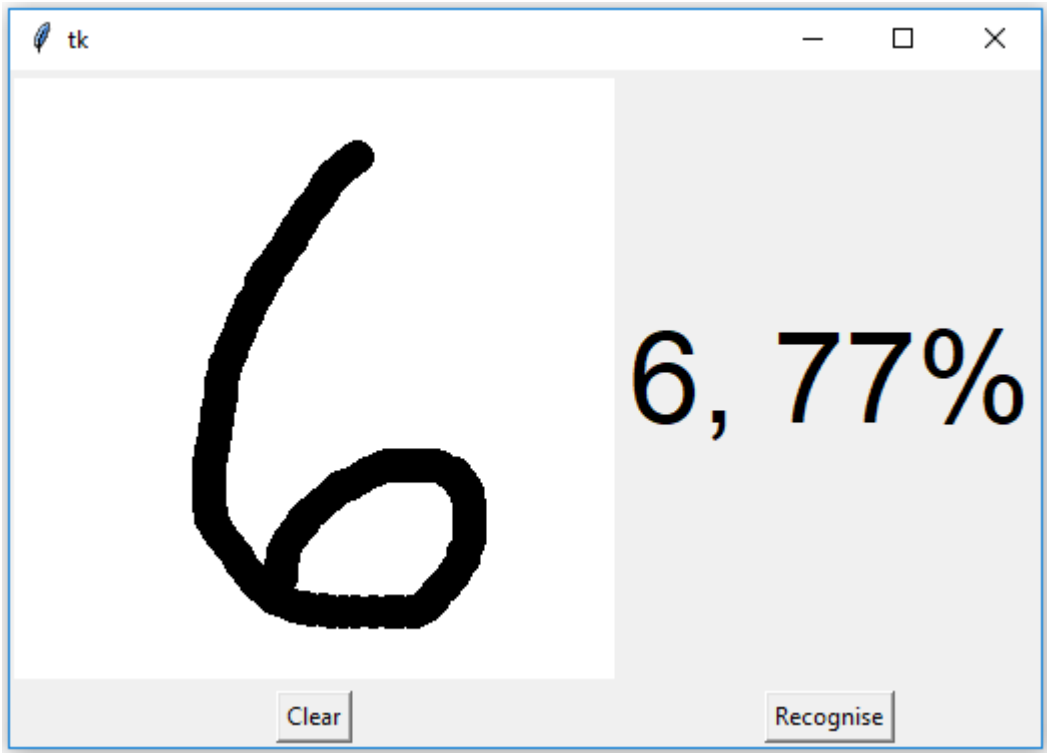
To evaluate how accurate our model works, we have around 10,000 images in our dataset. In the training of the data model, we do not include the testing data that’s why it is new data for our model. Around 99% accuracy is achieved with this well-balanced MNIST dataset.

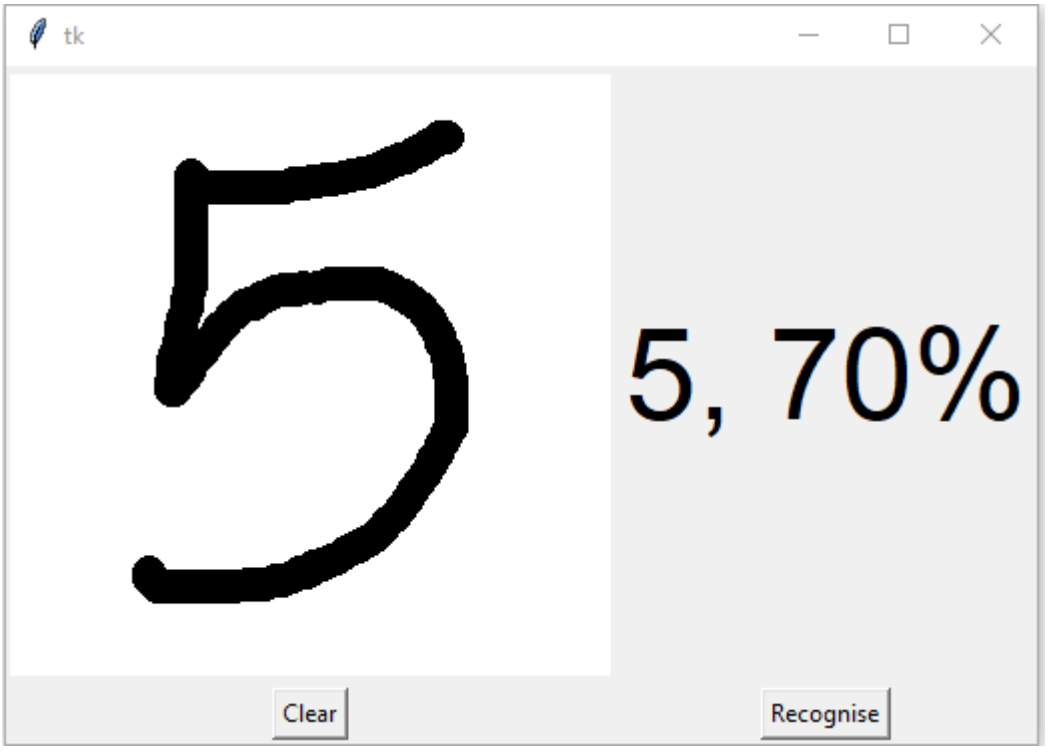
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

6. ***Create GUI to predict digits***

```
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="Draw..", 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
        a,b,c,d = rect
        rect=(a+4,b+4,c-4,d-4)
        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()
```


tk
6, 77%
Clear
Recognise


tk
5, 70%
Clear
Recognise