**HandWritten.ipynb**

import numpy as np

import matplotlib.pyplot as plt

import keras

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout

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

x\_train.shape , y\_train.shape , x\_test.shape , y\_test.shape

((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))

def plot\_input\_img(i):

plt.imshow(x\_train[i], cmap='binary')

plt.title(y\_train[i])

plt.show()

for i in range(10):

plot\_input\_img(i)

#normalizing the image to [0,1] range

x\_train = x\_train.astype(np.float32)/255

x\_test = x\_test.astype(np.float32)/255

# Re shape the dimensions of image to (28,28,1)

x\_train = np.expand\_dims(x\_train, -1)

x\_test = np.expand\_dims(x\_test, -1)

#cpnvert classes to one hot vector

y\_train = keras.utils.np\_utils.to\_categorical(y\_train)

y\_test = keras.utils.np\_utils.to\_categorical(y\_test)

from keras.layers.convolutional import Conv2D

model = Sequential()

model.add(Conv2D(32, (3,3), input\_shape = (28,28,1), activation='relu'))

model.add(MaxPool2D((2,2)))

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

model.add(MaxPool2D((2,2)))

model.add(Flatten())

model.add(Dropout(0.25))

model.add(Dense(10, activation='softmax'))

model.summary()

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

#callbacks

from keras.callbacks import EarlyStopping, ModelCheckpoint

#Earlyatopping

es = EarlyStopping(monitor='val acc', min\_delta= 0.01, patience= 4, verbose= 1 )

#Model Check Point

mc = ModelCheckpoint("./bestmodel.h5", monitor='val acc', verbose= 1, save\_best\_only= True)

cb = [es,mc]

import os

import tensorflow as tf

from tensorflow import keras

his = model.fit(x\_train, y\_train, epochs= 5, validation\_split= 0.03, callbacks=cb )

model.save("bestmodel.h5")

model\_s = keras.models.load\_model("C:/Users/shala/bestmodel.h5")

score = model\_s.evaluate(x\_test, y\_test )

print(f" the model accuracy is {score[1]}")

**App.ipynb**

import pygame,sys

from pygame.locals import \*

import numpy as np

from keras.models import load\_model

import cv2

#initialize or pygame

pygame.init()

WINDOWSIZEX= 640

WINDOWSIZEY= 480

BOUNDARYINC = 5

WHITE = (255, 255, 255)

BLACK = (0,0,0)

RED = (255,0,0)

IMAGESAVE = False

MODEL = load\_model("bestmodel.h5")

LABELS = {0:"Zero", 1:"One",

2:"Two", 3:"Three",

4:"Four", 5:"Five",

6:"Six", 7:"Seven",

8:"Eight", 9:"Nine"}

pygame.font.init()

FONT = pygame.font.SysFont('Raleway Bold', 18)

#myfont = pygame.font.SysFont('Raleway Bold', 72)

DISPLAYSURF = pygame.display.set\_mode((WINDOWSIZEX, WINDOWSIZEY))

pygame.display.set\_caption("Digit Board")

iswriting = False

number\_xcord = []

number\_ycord = []

img\_cnt = 1

PREDICT =True

while True:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == MOUSEMOTION and iswriting:

xcord, ycord = event.pos

pygame.draw.circle(DISPLAYSURF, WHITE, (xcord, ycord), 4,0)

number\_xcord.append(xcord)

number\_ycord.append(ycord)

if event.type == MOUSEBUTTONDOWN:

iswriting = True

if event.type == MOUSEBUTTONUP:

iswriting = False

number\_xcord = sorted(number\_xcord)

number\_ycord = sorted(number\_ycord)

rect\_min\_x , rect\_max\_x = max(number\_xcord[0]- BOUNDARYINC, 0), min(WINDOWSIZEX, number\_xcord[-1]+BOUNDARYINC)

rect\_min\_y , rect\_max\_y = max(number\_ycord[0]- BOUNDARYINC, 0), min(number\_ycord[-1]+BOUNDARYINC, WINDOWSIZEX)

number\_xcord = []

number\_ycord = []

img\_arr = np.array(pygame.PixelArray(DISPLAYSURF))[rect\_min\_x:rect\_max\_x, rect\_min\_y:rect\_max\_y].T.astype(np.float32)

if IMAGESAVE:

cv2.imwrite("image.png")

imag\_cnt +=1

if PREDICT:

image = cv2.resize(img\_arr, (28,28))

image = np.pad(image, (10,10), 'constant', constant\_values = 0)

image = cv2.resize(image ,(28,28))/255

label = str(LABELS[np.argmax(MODEL.predict(image.reshape(1,28,28,1)))])

textSurface = FONT.render(label , True, RED, WHITE)

textRecObj = textSurface.get\_rect()

textRecObj.left , textRecObj.bottom = rect\_min\_x, rect\_max\_y

DISPLAYSURF.blit(textSurface, textRecObj)

if event.type == KEYDOWN:

if event.unicode == "n":

DISPLAYSURF.fill(BLACK)

pygame.display.update()