```
# importing the required module

import keras
from keras.models import load_model
import imutils
from collections import deque
import cv2
import pandas as pd
from keras import layers
import numpy as np
from keras.layers import Input,Dense,Activation,ZeroPadding2D,BatchNormalization,Flatten,Conv2D
from keras.layers import AveragePooling2D,MaxPooling2D,Dropout,GlobalMaxPooling2D,GlobalAveragePooling2D
from keras.utils import np_utils,print_summary
import pandas as pd
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint
import keras.backend as k
#read the csv file database
data = pd.read_csv("data.csv")          # also we use the full path of that excel database csv file
dataset=np.array(data)
np.random.shuffle(dataset)
X=dataset
Y=dataset
X=X[:,0:1024]
Y=Y[:,1024]
X_train=X[0:70000,:]
```

```
X_train=X_train/255.
X_test=X[70000:72001,:]
X_test=X_test/255.
Y=Y.reshape(Y.shape[0],1)
Y_train=Y[0:70000,:]
Y_train=Y_train.T
Y_test=Y[70000:72001,:]
Y_test=Y_test.T
image_X=32
image_Y=32


train_Y=np_utils.to_categorical(Y_train)
test_Y=np_utils.to_categorical(Y_test)
train_Y=train_Y.reshape(train_Y.shape[1],train_Y.shape[2])
test_Y=test_Y.reshape(test_Y.shape[1],test_Y.shape[2])
X_train=X_train.reshape(X_train.shape[0],image_X,image_Y,1)
X_test=X_test.reshape(X_test.shape[0],image_X,image_Y,1)
print("X_train shape:"+str(X_train.shape))
print("Y_train shape:"+str(train_Y.shape))




def keras_model(image_X,image_Y):
    num_of_classes=37
    model=Sequential()
    model.add(Conv2D(32,(5,5),input_shape=(image_X,image_Y,1),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2),padding='same'))
    model.add(Conv2D(64,(5,5),activation='relu'))
    model.add(MaxPooling2D(pool_size=(5,5),strides=(5,5),padding='same'))
    model.add(Flatten())

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

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

    filepath="devnagari.h5"

checkpoint1=ModelCheckpoint(filepath,monitor='var_acc',verbose=1,save_best_only
=True,mode='max')

    callbacks_list=[checkpoint1]


    return model,callbacks_list


model, callbacks_list= keras_model(image_X, image_Y)

model.fit(X_train,train_Y,validation_data=(X_test,test_Y),epochs=1,batch_size=64,call
backs=callbacks_list)

scores=model.evaluate(X_test,test_Y,verbose=0)

print("CNN Error:%.2f%%"%(100-scores[1]*100))

print_summary(model)

model.save('devanagari.h5')



model1 = load_model('devanagari.h5')

print(model1)






def main():

    letter_count = {0: 'CHECK', 1: '01_ka', 2: '02_kha', 3: '03_ga', 4: '04_gha', 5: '05_kna',
6: 'character_06_cha',

                    7: '07_chha', 8: '08_ja', 9: '09_jha', 10: '10_yna',

                    11: '11_taamatar',
          12: '12_thaa', 13: '13_daa', 14: '14_dhaa', 15: '15_adna', 16: '16_tabala',
17: '17_tha',

          18: '18_da',

          19: '19_dha', 20: '20_na', 21: '21_pa', 22: '22_pha',

          23: '23_ba',

          24: '24_bha', 25: '25_ma', 26: '26_yaw', 27: '27_ra', 28: '28_la', 29:
'29_waw', 30: '30_motosaw',

          31: '31_petchiryakha',32: '32_patalosaw', 33: '33_ha',

          34: '34_chhya', 35: '35_tra', 36: '36_gya', 37: 'CHECK'}



cap = cv2.VideoCapture(2)  # for opening the webcam via opencv



Lower_green = np.array([110, 50, 50])

Upper_green = np.array([130, 255, 255])

pred_class=0

pts = deque(maxlen=512)

blackboard = np.zeros((480, 640, 3), dtype=np.uint8)

digit = np.zeros((200, 200, 3), dtype=np.uint8)

while (True):

  ret,img = cap.read()

  img=cv2.flip(img,1)

  if ret:

    imgHSV =cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    mask = cv2.inRange(imgHSV, Lower_green, Upper_green)

    blur = cv2.medianBlur(mask, 15)

    blur = cv2.GaussianBlur(blur, (5, 5), 0)

    thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

    contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[0]

    center = None
```

```
        if len(contours) >= 1:
            contour = max(contours, key=cv2.contourArea)
            if cv2.contourArea(contour) > 250:
                ((x, y), radius) = cv2.minEnclosingCircle(contour)
                cv2.circle(img, (int(x), int(y)), int(radius), (0, 255, 255), 2)
                cv2.circle(img, center, 5, (0, 0, 255), -1)
                M = cv2.moments(contour)
                center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))
                pts.appendleft(center)
                for i in range(1, len(pts)):
                    if pts[i - 1] is None or pts[i] is None:
                        continue
                    cv2.line(blackboard, pts[i - 1], pts[i], (255, 255, 255), 10)
                    cv2.line(img, pts[i - 1], pts[i], (0, 0, 255), 5)
        elif len(contours) == 0:
            if len(pts) != []:
                blackboard_gray = cv2.cvtColor(blackboard, cv2.COLOR_BGR2GRAY)
                blur1 = cv2.medianBlur(blackboard_gray, 15)
                blur1 = cv2.GaussianBlur(blur1, (5, 5), 0)
                thresh1 = cv2.threshold(blur1, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)[1]
                blackboard_cnts = cv2.findContours(thresh1.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[0]
                if len(blackboard_cnts) >= 1:
                    cnt = max(blackboard_cnts, key=cv2.contourArea)
                    print(cv2.contourArea(cnt))
                    if cv2.contourArea(cnt) > 2000:
                        x, y, w, h = cv2.boundingRect(cnt)
                        digit = blackboard_gray[y:y + h, x:x + w]
                        # newImage = process_letter(digit)
                        pred_probab, pred_class = keras_predict(model1, digit)
                    print(pred_class, pred_probab)


            pts = deque(maxlen=512)
            blackboard = np.zeros((480, 640, 3), dtype=np.uint8)
        cv2.putText(img, "Conv Network :  " + str(letter_count[pred_class]), (10, 470),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.imshow("Frame", img)
        cv2.imshow("Contours", thresh)
        if cv2.waitKey(27) & 0xFF==ord('q'):
            break


cap.release()
cv2.destroyAllWindows()


def keras_predict(model, image):
    processed = keras_process_image(image)
    print("processed: " + str(processed.shape))
    pred_probab = model.predict(processed)[0]
    pred_class = list(pred_probab).index(max(pred_probab))
    return max(pred_probab), pred_class




def keras_process_image(img):
    image_x = 32
    image_y = 32
    img = cv2.resize(img, (image_x, image_y))
    img = np.array(img, dtype=np.float32)

    img = np.reshape(img, (-1, image_x, image_y, 1))

    return img



import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

keras_predict(model1, np.zeros((32, 32, 1), dtype=np.uint8))

main()
```