In [1]: **import os**

**import time**

**import numpy as np**

**from matplotlib import** pyplot **as** plt

**import cv2 as cv**

**import mediapipe as mp**

**import seaborn as sns**

**from sklearn.model\_selection import** train\_test\_split

**import tensorflow as tf**

**from tensorflow.keras import** models

**from tensorflow.keras import** layers

**from sklearn.metrics import** confusion\_matrix ,classification\_report **from gtts import** gTTS

**from playsound import** playsound

In [2]: tf.\_\_version\_\_

Out[2]: '2.1.0'

In [4]: tf.test.is\_gpu\_available('gpu')

Out[4]: True

In [39]: input\_types = ['palm', 'fist', 'thumbsup', 'gun', 'call']

In [40]: description\_types = ['01', '02', '03', '04', '05']

In [41]: defination\_types = ['a', 'b', 'c', 'd', 'e']

**Load Model**

In [6]: cnn = models.load\_model("cnn.h5")

In [7]: cnn.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #

================================================================= conv2d (Conv2D) (None, 96, 96, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ max\_pooling2d (MaxPooling2D) (None, 48, 48, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout (Dropout) (None, 48, 48, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_1 (Conv2D) (None, 48, 48, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ max\_pooling2d\_1 (MaxPooling2 (None, 24, 24, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_1 (Dropout) (None, 24, 24, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ flatten (Flatten) (None, 36864) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense (Dense) (None, 512) 18874880

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_2 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_1 (Dense) (None, 5) 2565

================================================================= Total params: 18,896,837

Trainable params: 18,896,837

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [8]: l = []

**for** i **in** range(5):

image\_matrix = plt.imread('Inputs/call/' + 'call\_' + str(i + 1) + '.jpg') l.append(image\_matrix)

l = np.array(l)

l = l / 255

In [9]: cnn.predict(l)

Out[9]: array([[2.6413172e-10, 3.2406274e-13, 5.9979181e-11, 1.3836636e-07, 9.9999988e-01],

[4.7895166e-10, 3.6021318e-12, 5.4874799e-10, 3.0129550e-07, 9.9999964e-01],

[1.0681169e-10, 2.1262924e-13, 1.7447507e-11, 3.5530383e-07, 9.9999964e-01],

[3.1814351e-09, 3.2657731e-11, 6.4703887e-09, 5.8453224e-08, 9.9999988e-01],

[2.3002328e-10, 2.8322423e-12, 2.8186053e-10, 2.9070209e-08, 1.0000000e+00]], dtype=float32)

In [10]: l.shape

Out[10]: (5, 96, 96, 3)

In [11]: **def** Predict(img):

class\_ = np.argmax(cnn.predict(img))

**return** input\_types[class\_]

In [12]: Predict(np.array([l[0]]))

Out[12]: 'call'

**New code**

In [13]: **def** saving\_sounds():

language = 'en'

**for** types **in** input\_types:

obj = gTTS(text = types, lang = language, slow = **False**) **if** os.path.isfile(str(types) + ".mp3") **is False**:

obj.save(str(types) + ".mp3")

In [14]: saving\_sounds()

In [15]: types = 'thumbsup'

playsound(str(types) + ".mp3")

In [16]: **def** playing\_sounds():

**for** types **in** input\_types:

playsound(str(types) + ".mp3")

In [18]: playing\_sounds()

**linkers**

In [42]: **def** message\_class(mirror1, class\_):

cv.putText(mirror1, str(class\_), (100, 100), cv.FONT\_HERSHEY\_PLAIN, 2, (255,0,0), 1) **return** mirror1

In [43]: **def** message\_sound(class\_):

*# if os.path.isfile(str(class\_) + ".mp3") is False:*

*# obj.save(str(class\_) + ".mp3")*

playsound(str(class\_) + ".mp3")

block = **True**

In [44]: message\_sound("palm")

In [45]: **def** message\_description(black1, class\_):

text1 = description\_types[input\_types.index(class\_)]

cv.putText(black1, str(text1), (100, 100), cv.FONT\_HERSHEY\_PLAIN, 5, (255,0,0), 1) **return** black1

In [46]: **def** message\_defination(black2, class\_):

text2 = defination\_types[input\_types.index(class\_)]

cv.putText(black2, str(text2), (100, 100), cv.FONT\_HERSHEY\_PLAIN, 5, (255,0,0), 1) **return** black2

**Real Time Acc**

In [22]: mp\_drawing = mp.solutions.drawing\_utils mp\_holistic = mp.solutions.holistic

In [23]: holistic = mp\_holistic.Holistic()

In [47]: R = 25

thickness = 2

webcam = 0

capture = cv.VideoCapture(webcam)

pre\_class = ''

fps = int(capture.get(cv.CAP\_PROP\_FPS))

print("fps is " + str(fps))

\_, frame = capture.read()

height, width, channel = frame.shape

**while** capture.isOpened():

*#time.sleep()*

**if** cv.waitKey(1) & 0xFF == 13:

**break**

black = np.zeros(shape = frame.shape)

black1 = np.zeros(shape = frame.shape)

black2 = np.zeros(shape = frame.shape)

\_, frame = capture.read()

frame\_rgb = cv.cvtColor(frame, cv.COLOR\_BGR2RGB)

result = holistic.process(frame\_rgb)

**try**:

hand\_landmarks = result.right\_hand\_landmarks.landmark **if** hand\_landmarks:

x\_max = 0

y\_max = 0

x\_min = width

y\_min = height

**for** i **in** range(0,21,1):

lm = hand\_landmarks[i]

x, y = int(lm.x \* width), int(lm.y \* height)

**if** x > x\_max:

x\_max = x

**if** x < x\_min:

x\_min = x

**if** y > y\_max:

y\_max = y

**if** y < y\_min:

y\_min = y

frame\_bgr = cv.cvtColor(frame\_rgb, cv.COLOR\_RGB2BGR)

mp\_drawing.draw\_landmarks(frame\_bgr, result.right\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS) cv.rectangle(frame\_bgr, (x\_min - R, y\_min - R), (x\_max + R, y\_max + R), (0, 255, 0), thickness)

result1 = frame\_bgr

mirror1 = cv.flip(result1, 1)

*'''*

*'''*

mp\_drawing.draw\_landmarks(black, result.right\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS)

croped = black[y\_min - R + thickness: y\_max + R - thickness, x\_min - R + thickness : x\_max + R - thickness ]

resized = cv.resize(croped, (96, 96))

mirror2 = cv.flip(resized, 1)

result2 = mirror2

img\_mat = np.array([result2])

class\_ = Predict(img\_mat)

*# msg 01*

message\_class(mirror1, class\_)

*#msg 02*

*# if pre\_class != class\_:*

*# message\_sound(class\_)*

*# pre\_class = class\_*

*#msg 03*

black1 = message\_description(black1, class\_) result3 = black1

*#msg 04*

black2 = message\_defination(black2, class\_) result4 = black2

cv.imshow("Frame2", result2)

cv.imshow("Frame3", result3)

cv.imshow("Frame4", result4)

**except**:

result1 = frame

mirror1 = cv.flip(result1, 1)

*#result2 = black*

**pass**

cv.imshow('frame1', mirror1)

capture.release()

cv.destroyAllWindows()

fps is 30

In [ ]: