RNN

import cv2

import numpy as np

import os

from matplotlib import pyplot as plt

import time

import mediapipe as mp

mp\_holistic=mp.solutions.holistic

mp\_drawing=mp.solutions.drawing\_utils

def mediapipe\_detection(image,model):

image=cv2.cvtColor(image,cv2.COLOR\_BGR2RGB)

image.flags.writeable=False

results=model.process(image)

image.flags.writeable=True

image=cv2.cvtColor(image,cv2.COLOR\_RGB2BGR)

return image,results

def draw\_styled\_landmarks(image,results):

mp\_drawing.draw\_landmarks(image,results.face\_landmarks,mp\_holistic.FACEMESH\_CONTOURS,

mp\_drawing.DrawingSpec(color=(80,110,10),thickness=1 ,circle\_radius=1),

mp\_drawing.DrawingSpec(color=(80,256,121),thickness=1,circle\_radius=1))

mp\_drawing.draw\_landmarks(image,results.pose\_landmarks,mp\_holistic.POSE\_CONNECTIONS,

mp\_drawing.DrawingSpec(color=(80,22,10),thickness=1, circle\_radius=1),

mp\_drawing.DrawingSpec(color=(80,44,121),thickness=1,circle\_radius=1))

mp\_drawing.draw\_landmarks(image,results.left\_hand\_landmarks,mp\_holistic.HAND\_CONNECTIONS,

mp\_drawing.DrawingSpec(color=(121,22,76),thickness=1, circle\_radius=1),

mp\_drawing.DrawingSpec(color=(121,44,250),thickness=1,circle\_radius=1))

mp\_drawing.draw\_landmarks(image,results.right\_hand\_landmarks,mp\_holistic.HAND\_CONNECTIONS,

mp\_drawing.DrawingSpec(color=(245,117,66),thickness=1 ,circle\_radius=1),

mp\_drawing.DrawingSpec(color=(245,66,230),thickness=1,circle\_radius=1))

cap=cv2.VideoCapture(0)

with mp\_holistic.Holistic(min\_detection\_confidence=0.5,min\_tracking\_confidence=0.5)as holistic:

while cap.isOpened():

ret,frame=cap.read()

image,results=mediapipe\_detection(frame,holistic)

print(results)

draw\_styled\_landmarks(image,results)

cv2.imshow('OpenCV Feed',image)

if cv2.waitKey(10)&0xFF==ord('q'):

break

cap.release()

cv2.destroyAllWindows

def extract\_keypoints(results):

pose = (np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose\_landmarks.landmark]).flatten() if results.pose\_landmarks else np.zeros(33\*4))

face = (np.array([[res.x, res.y, res.z] for res in results.face\_landmarks.landmark]).flatten() if results.face\_landmarks else np.zeros(468\*3))

rh = (np.array([[res.x, res.y, res.z] for res in results.right\_hand\_landmarks.landmark]).flatten() if results.right\_hand\_landmarks else np.zeros(21\*3))

lh = (np.array([[res.x, res.y, res.z] for res in results.left\_hand\_landmarks.landmark]).flatten() if results.left\_hand\_landmarks else np.zeros(21\*3))

return np.concatenate([pose,face,lh,rh])

FOLDER CREATION-TO STORE DATASET

import os

import numpy as np

DATA\_PATH = r'C:\Users\DELL\HOP\_Data'

actions = np.array(['Hello','Thanks','Love You','Yes','No','Help','Please','Food',])

no\_sequences = 30

sequence\_length=30

start\_folder=30

for action in actions:

for sequence in range(no\_sequences):

try:

os.makedirs(os.path.join(DATA\_PATH, action, str(sequence)))

print(f"Created directory: {os.path.join(DATA\_PATH, action, str(sequence))}")

except Exception as e:

print(f"Failed to create directory: {os.path.join(DATA\_PATH, action, str(sequence))}")

print(f"Error: {e}")

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

from tensorflow.keras.utils import to\_categorical

label\_map={label:num for num,label in enumerate(actions)}

#DATAPOINT COLLECTION.

cap=cv2.VideoCapture(0)

with mp\_holistic.Holistic(min\_detection\_confidence=0.5,min\_tracking\_confidence=0.5)as holistic:

for action in actions:

for sequence in range(no\_sequences):

for frame\_num in range(sequence\_length):

ret,frame=cap.read()

image,results=mediapipe\_detection(frame,holistic)

if frame\_num==0:

cv2.putText(image,'Starting\_collection',(120,200),

cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,255,0),4,cv2.LINE\_AA)

cv2.putText(image,'Collecting\_frames for {} video Number{}'.format(action,sequence),(15,12),

cv2.FONT\_HERSHEY\_SIMPLEX,0.5,(0,0,255),4,cv2.LINE\_AA)

cv2.waitKey(500)

else:

cv2.putText(image,'Collecting frames for {} video number{}'.format(action,sequence),(15,12),

cv2.FONT\_HERSHEY\_SIMPLEX,0.5,(0,0,255),4,cv2.LINE\_AA)

cv2.waitKey(100)

key\_points=extract\_keypoints(results)

npy\_path=os.path.join(DATA\_PATH,action,str(sequence),str(frame\_num))

np.save(npy\_path,key\_points)

draw\_styled\_landmarks(image,results)

cv2.imshow('OpenCV Feed',image)

if cv2.waitKey(10)&0xFF==ord('q'):

break

cap.release()

cv2.destroyAllWindows

sequences,labels=[],[]

for action in actions:

for sequence in range(no\_sequences):

window=[]

for frame\_num in range(sequence\_length):

res=np.load(os.path.join(DATA\_PATH,action,str(sequence),"{}.npy".format(frame\_num)))

window.append(res)

sequences.append(window)

labels.append(label\_map[action])

X=np.array(sequences)

y=to\_categorical(labels).astype(int)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.08)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM,Dense

from tensorflow.keras.callbacks import TensorBoard

model=Sequential()

model.add(LSTM(64,return\_sequences=True,activation='relu',input\_shape=(30,1662)))

model.add(LSTM(128,return\_sequences=True,activation='relu'))

model.add(LSTM(64,return\_sequences=False,activation='relu'))

model.add(Dense(64,activation='relu'))

model.add(Dense(32,activation='relu'))

model.add(Dense(actions.shape[0],activation='softmax'))

model.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=['categorical\_accuracy'])

model.fit(X\_train,y\_train,epochs=1000,validation\_data=(X\_test,y\_test))

res=model.predict(X\_test)

model.save('Signmain.h5')

model.load\_weights('Signmain.h5')

colors=[(245,117,16),(117,245,16),(16,117,245),(16,117,245),(117,245,16),(117,245,16),(117,245,16),(117,245,16)]

def prob\_viz(res,actions,input\_frame,colors):

output\_frame=input\_frame.copy()

for num,prob in enumerate(res):

cv2.rectangle(output\_frame,(0,60+num\*40),(int(prob\*100),90+num\*40),colors[num],-1)

cv2.putText(output\_frame,actions[num],(0,85+num\*40),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,255),2,cv2.LINE\_AA)

return output\_frame

sequence=[]

sentence=[]

predictions=[]

threshold=0.45

model.load\_weights('Signmain.h5')

cap=cv2.VideoCapture(0)

with mp\_holistic.Holistic(min\_detection\_confidence=0.5,min\_tracking\_confidence=0.5)as holistic:

while cap.isOpened():

ret,frame=cap.read()

image,results=mediapipe\_detection(frame,holistic)

print(results)

draw\_styled\_landmarks(image,results)

#prediction

keypoints=extract\_keypoints(results)

sequence.append(keypoints)

sequence=sequence[-30:]

if len(sequence)==30:

res=model.predict(np.expand\_dims(sequence,axis=0))[0]

print(actions[np.argmax(res)])

predictions.append(np.argmax(res))

#RESULT PRINTING ON SCREEN

image=prob\_viz(res,actions,image,colors)

cv2.rectangle(image,(0,0),(640,40),(245,117,16),-1)

cv2.putText(image,''.join(sentence),(3,30),

cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,255),2,cv2.LINE\_AA)

cv2.imshow('OpenCV Feed',image)

if cv2.waitKey(10)&0xFF==ord('q'):

break

cap.release()

cv2.destroyAllWindows