B.TECH. PROJECT ON

Communication Assistant for People having Hearing and Speech Impairment

Submitted By:

Lakshay Verma 2019UEC2508
Mohd Zunaid Al Ameen Ansari 2019UEC2517
Kunal Nayak 2019UEC2527
Archit Bikram 2019UEC2537

Under the Guidance of:

Dr. Sukhbir Singh

Project-II in partial fulfillment of requirement for the award of

B.Tech. in
Electronics & Communication Engineering

Division of Electronics & Communication Engineering

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY
NEW DELHI-110078

April-May 2023

Certificate

Certified that Lakshay Verma 2019UEC25208, Mohd Zunaid Al Ameen Ansari 2019UEC2517, Kunal Nayak 2019UEC2527, Archit Bikram 2019UEC2537 has carried out their project work presented in the project entitled " Communication assistant for people having hearing and speech impairment " for the award of Bachelor of Technology, Department of Electronics and Communication, Netaji Subhas University of Technology, New Delhi, under my supervision. The project embodies results of original work, and studies are carried out by the student himself/herself and the contents of the project do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

(Signature)

Dr. Sukhbir Singh
Assitant Professor, ECE department
Netaji Subhas University of Technology

Date:

Acknowledgement

We would like to express our sincere gratitude and respect towards NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY, for giving us the chance to make this project possible.

We would like to express our gratitude towards our supervisor **Dr. Sukhbir Singh**, who provided us with the golden opportunity to do this engaging project of **Communication Assistant for people having hearing and speech impairment** who is also guiding us in attempting this project. We have learnt many new technologies. We are really thankful to them.

Also, we would like to thank the Electronics and Communication department and the Head of the department for giving us this opportunity of working on this project.

Lakshay Verma

Mohd Zunaid Al Ameen Ansari

Kunal Nayak

Archit Bikram

BTP report

ORIGINALITY REPORT

16%	12%	11%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Michail Doukas, Sotirios Xydis, Dimitrios Soudris. "Dataflow Acceleration of scikit-learn Gaussian Process Regression", Proceedings of the 8th Workshop and 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms - PARMA-DITAM '17, 2017 Publication	2%
2	Submitted to University of Lancaster Student Paper	2%
3	Submitted to National Institute of Technology, Rourkela Student Paper	2%
4	ir.aiktclibrary.org:8080 Internet Source	1%
5	wikimili.com Internet Source	1%

6	medium.com Internet Source	1 %
7	Submitted to iGroup Student Paper	1 %
8	sol.sbc.org.br Internet Source	1 %
9	www.semanticscholar.org Internet Source	1 %
10	link.springer.com Internet Source	1 %
11	jyx.jyu.fi Internet Source	1 %
12	T. Prem Jacob, A. Pravin, K. Mohana Prasad, G T. Judgi, R. Rajakumar. "Hand Sign Language Interpreter using Machine Learning Model", 2022 International Conference on Electronics and Renewable Systems (ICEARS), 2022 Publication	1 %
13	Submitted to Kaplan International Colleges Student Paper	<1 %
14	Mohammed S. Al-Samarraay, Mahmood M. Salih, Mohamed A. Ahmed, A. A. Zaidan et al. "A new extension of FDOSM based on Pythagorean fuzzy environment for evaluating and benchmarking sign language	<1 %

recognition systems", Neural Computing and Applications, 2022
Publication

15	Prima Dewi Purnamasari, Muhammad Taqiyuddin, Anak Agung Putri Ratna. "Performance comparison of text-based sentiment analysis using recurrent neural network and convolutional neural network", Proceedings of the 3rd International Conference on Communication and Information Processing - ICCIP '17, 2017 Publication	<1 %
16	dergipark.org.tr Internet Source	<1 %
17	escholarship.org Internet Source	<1 %
18	Saggio, Giovanni, Francesco Riillo, Laura Sbernini, and Lucia Rita Quitadamo. "Resistive flex sensors: a survey", Smart Materials and Structures, 2016. Publication	<1 %
19	Sepp Hochreiter, Jürgen Schmidhuber. "Long Short-Term Memory", Neural Computation, 1997 Publication	<1 %
20	Submitted to Liverpool John Moores University Student Paper	<1 %

21	Mellit, A.. "Artificial intelligence techniques for photovoltaic applications: A review", Progress in Energy and Combustion Science, 200810 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Abstract

Impairment in hearing and speech can potentially hinder the communication of any individual with others around them. For such situations the introduction of sign language has led to ease among such individuals to communicate fairly, but the lack of knowledge about such a system and less motivation towards knowing such a language still has turned into a communication gap for these individuals and people who do not have such impairments.

For such a case we have designed a communication assistant so that communication among people from the above-mentioned two groups can be achieved with relative ease.

The model developed takes real-time video feed and finds key points and landmarks, which are then processed by a set of neural networks to obtain the final output as a textual message which is a translation of the signs/gestures that were input to the model.

List of Contents

Chapter 1: Introduction 6
Chapter 2: Literature Review.....7
Chapter 3: Tools and Libraries Used.....8
Chapter 4: Specification & Design 11
Chapter 5: Conclusion.....21
References.....22
Appendix.....24

List of Figures

Figure 1.1: Basic Sign Language Gestures 6
Figure 3.1: Mediapipe hand key points 8
Figure 3.2: Mediapip Body Pose key points 9
Figure 4.1: Recurrent neural network 11
Figure 4.2: Zunaid Gesturing “Hello” in Sign Language 15
Figure 4.3: Zunaid Gesturing for “Please” in Sign Language 16
Figure 4.4: Output image for Done 17
Figure 4.5: Output image for Help 18
Figure 4.6: Output image for Please 18
Figure 4.7: Output image for Learn 19
Figure 4.8: Output image for No 19
Figure 4.9: Output image for I Love You 20
Figure 4.10 Confusion Matrix 21
Figure 4.11 Accuracy Score 21
References 26
Appendix 1a Complete Source Code 27

Chapter 1: Introduction

The goal of the project is to create a model for effortless and rapid communication among people with speech and/or hearing impairment and others. The method of communication used by people having such an impairment is generally sign language, which is a system of communication using visual gestures and signs. Such gestures and signs can be abundant and learning such a system for every individual who can speak and hear be hectic. This model is an attempt at solving this issue.

The model takes the gestures and motions of an individual as input. The input is then converted to the appropriate format and passed through the first model, in here, important information such as key points and landmarks on the individuals face, body, right and left arms are detected. These are the location whose motion determines the gesture which when combined can be entire messages. To determine the motion, we need multiple instances of images to detect changes in the position of these important points. To do that multiple images are taken as input at different small instances of time one after another.

figure 1.1: Basic Sign Language Gestures

Chapter 2: Literature Review

Hand gesture recognition is an area within Human-Computer Interaction (HCI) which facilitates non-verbal communication among humans and machines. Real-time gesture decoding systems have been explored in detail within the past few decades. Agrawal, S., et al [1] proposed a system that utilizes MediaPipe for extracting hand landmarks and recognizing the gesture. The proposed system was evaluated using a self-made dynamic dataset of certain everyday gestures, achieving an accuracy of 91.4% in recognizing different hand gestures. The system also showed robustness and stability in recognizing hand gestures in real time.

Several research works have been conducted to recognize hand gestures using different approaches. Yang, W., et al [2] proposed a hand gesture recognition system using a deep learning approach. The system employed the VGG-16 network for feature extraction and a convolutional neural network (CNN) for classification. The system achieved high accuracy in recognizing different hand gestures. Liu, J., Huang, Z., et al [3] proposed a deep learning-based hand gesture recognition system using a combination of CNN and Recurrent Neural Network (RNN). The system employed depth maps and RGB images for hand gesture recognition. The system achieved high accuracy in recognizing different hand gestures.

The proposed system can be used in various applications, including human-computer interaction, virtual reality, gaming, and robotics. Future work may include testing the system on a larger dataset and exploring the possibility of using the system with a larger set of hand gestures. Overall, the proposed system is a promising solution for real-time hand gesture recognition and has the potential to contribute significantly to the field of HCI.

Another paper "Measurement of the Flexible Bending Force of the Index and Middle Fingers for Virtual Interaction" by N. H. Adnan, K. Wan, and A. B. Shahriman was published in 2012 in the journal "Vol. 41, no. Iris" tries to deal with the given issue but in a different set of tools equipped. The study presented in the paper aimed to measure the flexible bending force of the index and middle fingers for virtual interaction, which is an important aspect of human-computer interaction. Using a specially-designed device the writers of this paper have attempted to measure the force required to move the Middle/Forefinger and use this information to assist in certain basic forms of communication. The study results showed that the device was able to measure the flexible bending force of the fingers accurately and that the force required to bend the fingers varied depending on the degree of bending and the type of finger movement. It was suggested that this device can be used for the design of virtual interaction devices.

Chapter 3: Tools and Libraries Used

3.1 Machine Learning: It is a subset of artificial intelligence that helps software applications to become more correct at predicting outcomes without explicitly being programmed to do that. Machine learning algorithms use previous data in the input to predict new output values.

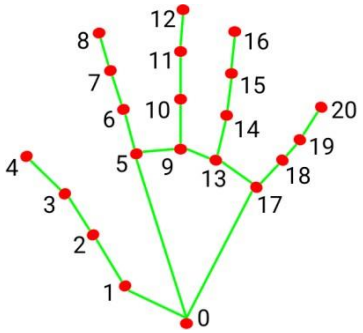
3.2 Recurrent Neural Network: Recurrent Neural Networks are one of the forms of feed-forward neural networks. Every one of the neurons in the hidden layer would get input after a certain delay. The RNN mostly obtains the previous information. For example, to guess the next word in any sentence, one must have knowledge of the words that were used previously.

Application of RNN:

1. Time Series Prediction
2. Speech Recognition
3. Speech Synthesis
4. Music Composition

3.3 Mediapipe Holistic: It is a Framework used for building machine learning pipelines for the time series processing like video, audio, etc. It is cross-platform and can be used on Desktop/Server, Android, iOS, and even embedded devices.

Mediapipe holistic is a pipelines which contains optimized hands,pose and face components that will allow it for holistic tracking. Hence making the model sufficient to simultaneously detect the body and poses along with face tracking by drawing landmarks in real-time.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

figure 3.1: Mediapipe hand key points

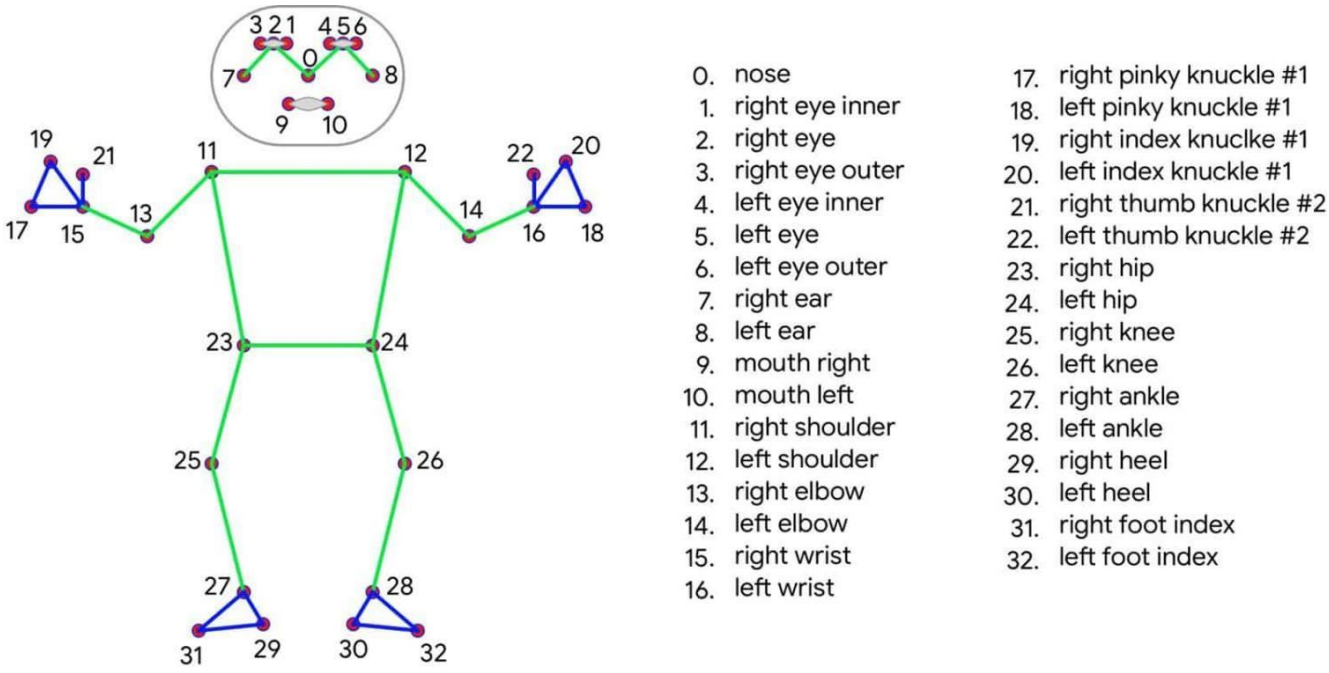


Figure 3.2: Mediapip Body Poses

3.4 Scikit-learn: Scikit learn is a free machine learning software library, developed for the Python programming language. It has various features for classifications, regression, and clustering algorithms and is designed to co-work with the Python numerical and scientific libraries NumPy and SciPy.

3.5 LSTM Model: LSTM stands for long short-term memory. Although a recurrent neural network can be used in tasks where memory is expected from the system, due to new values being dependent on older values owing to the feedback mechanism feeding previous outputs as a parameter to the system, traditional RNNs fail to accomplish these tasks where a memory serving for longer than a few iterations may be required. Here a system that can be trained to save states that are relevant to the context it is trained for.

The LSTM model achieves this by the use of 3 types of gates that decide what information gets stored, what is tossed away, and what is to be given as the output; namely Input Gates, Output Gates.

3.6 Numpy: Numpy is a library provided in Python. It adds support for large, multi–dimensional arrays and matrices along with a large collection of high-level mathematical functions to work on multidimensional arrays and matrices.

3.7 TensorFlow: TensorFlow is an open-source software library for machine learning and artificial intelligence. The method can be applied to a variety of tasks but is specifically useful for inferring and training deep neural networks.

3.8 OpenCV: OpenCV or cv2 is a library provided in Python. It provides common infrastructures for applications involving computer vision tasks. It allows us to perform

image processing and computer vision tasks with ease.

3.9 Matplotlib: Matplotlib is a plotting library for the Python language and its mathematics extension NumPy. It creates different types of visual data/ Graphs for best interpretation of the data.

Chapter 4: Specification & Design

4.1 Specification

This project attempts to detect and convert real-time sign language into textual message, which can then be displayed on a screen. To achieve this first the motion is captured by a camera which is then converted into multiple image files that are then analyzed into groups, to make sense of a connected set of motions/poses into a single word/phrase which can then further be combined to make sentences.

4.2 Design

OpenCV is used to obtain the snapshots of the web cam feed of the motion, which are then stacked together into sets of 30 to produce the video feed. After that, using Mediapipe, we import the holistic model that we used to obtain landmarks. As a result, landmarks are identified on the face, body, and right and left hand.

To process all the keypoints received from mediapipe we will be using neural network since it tries to replicate real human like processing of image information processing.

Our project requires a neural network which can process information of gestures which vary with time. So we need a kind of neural network with memory one such type of neural network is RNN (Recurrent neural network).

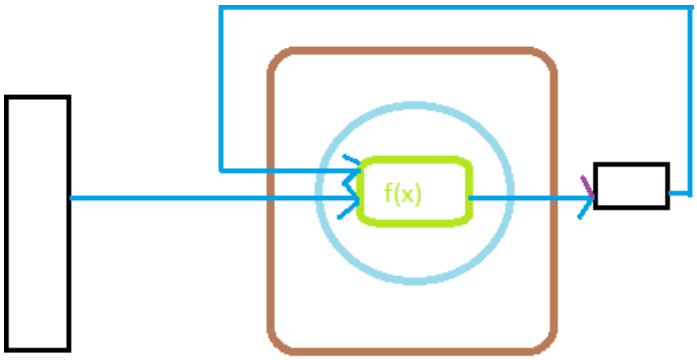


Figure 4.1: A Recurrent neural network

Although Recurrent neural network has the capability to store information (it has memory) but this neural network has the issue of vanishing and exploding gradients which results in RNN not being able to retain previous input for a long duration.

This problem can be solved in three ways:

1. Weight initialization

2. Choosing the right activation function
3. LSTM (Long short-term memory)

Among these ways Long Short Term Memory is the most convenient solution for our purposes.

Using numpy we convert the landmarks into numpy arrays which we can then flatten to obtain another array with fewer dimensions.

Following this, we collect data for different actions in the sign language, using the camera feed to train our model. We obtain 30 samples of actions for each action (such as “Hello”, “Bye”, and “Thanks”) to train our model, each sample containing 30 frames. The motion is then saved a converted into numpy array flattened and saved for later training and testing.

Using the above 3 * 30 samples, the data are divided into training and testing data that are mutually exclusive from one another. The Data is then fed into the given neural network. The Neural Network is designed with 5 hidden layers which take 1662 input units to output them down to the number of actions that we are trainin

4.3 Implementation

1. Import and Install Dependencies

In [ ]:
```
!pip install tensorflow==2.4.1 tensorflow-gpu==2.4.1 opencv-python mediapipe sklearn matplotlib
```

In [1]:
```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
```

2. Keypoints using MP Holistic

In [2]:
```
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities
```

In [3]:
```
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False                  # Image is no longer writeable
    results = model.process(image)                 # Make prediction
    image.flags.writeable = True                   # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return image, results
```

In [4]:
```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS) # Draw face connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS) # Draw pose connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw right hand connections
```

In [5]:
```
def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                              )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )
```

In [ ]:
```
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

In [30]:
```
draw_landmarks(frame, results)
```

In [31]:
```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Finding Landmarks and Key points

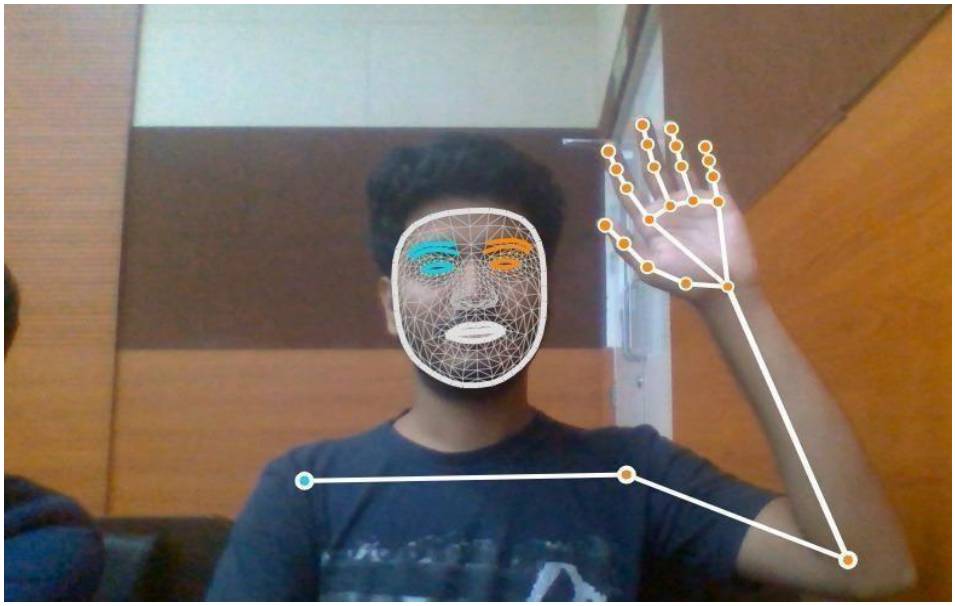


Figure 4.2: Zunaid Ansari Gesturing “Hello” in Sign Language

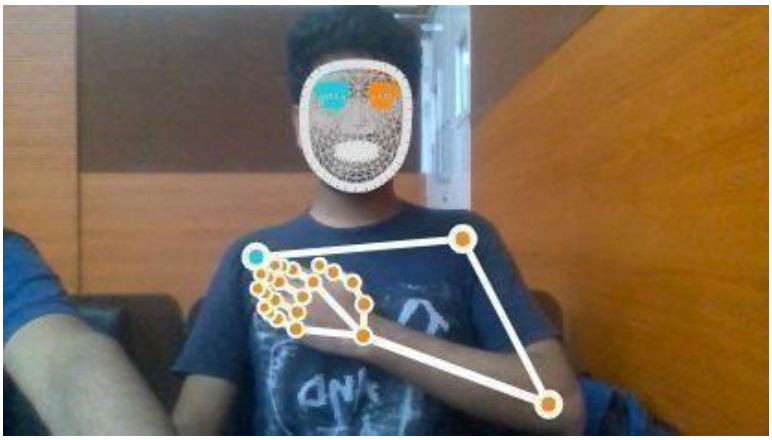


figure 4.3: Zunaid Gesturing for “Please” in Sign Language

3. Extract Keypoint Values

In [60]:
```
pose = []
for res in results.pose_landmarks.landmark:
    test = np.array([res.x, res.y, res.z, res.visibility])
    pose.append(test)
```

In [105…
```
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.zerc
face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if results.face_landmarks else np.zeros(1404)
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)
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
```

In [109…
```
face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten()
    if results.face_landmarks
    else np.zeros(1404)
```

Out[109…
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [6]:
```
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.
    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)
    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
    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(
    return np.concatenate([pose, face, lh, rh])
```

In [122…
```
result_test = extract_keypoints(results)
```

Extracting Key points

4. Setup Folders for Collection

In [7]:
```
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('MP_Data')

# Actions that we try to detect
actions = np.array(['hello', 'thanks', 'iloveyou'])

# Thirty videos worth of data
no_sequences = 30

# Videos are going to be 30 frames in length
sequence_length = 30

# Folder start
start_folder = 30
```

In [34]:
```
for action in actions:
    dirmax = np.max(np.array(os.listdir(os.path.join(DATA_PATH, action))).astype(int))
    for sequence in range(1,no_sequences+1):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(dirmax+sequence)))
        except:
            pass
```

Collecting Data and Saving in a file

Figure 4.4: Output image for Done

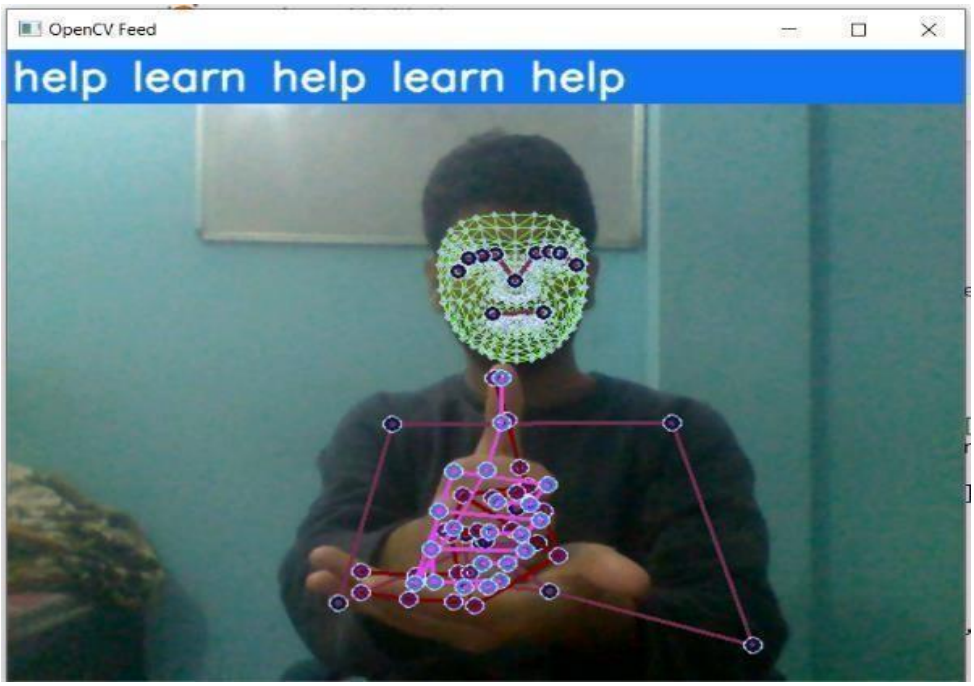


figure 4.5: Output image for Help

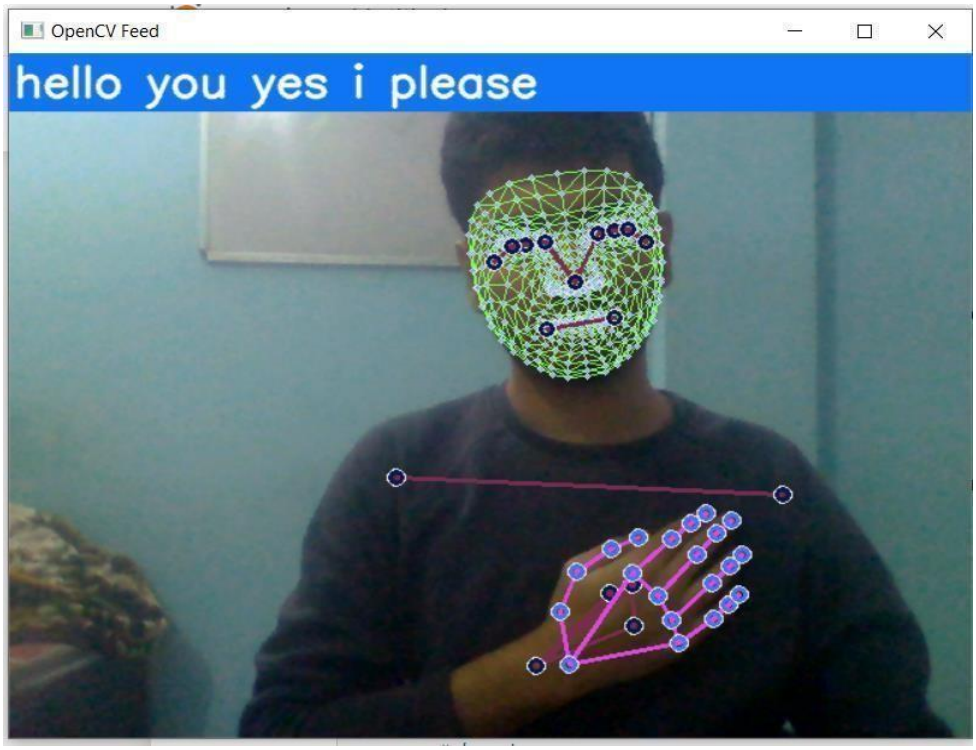


Figure 4.6: Output image for Please

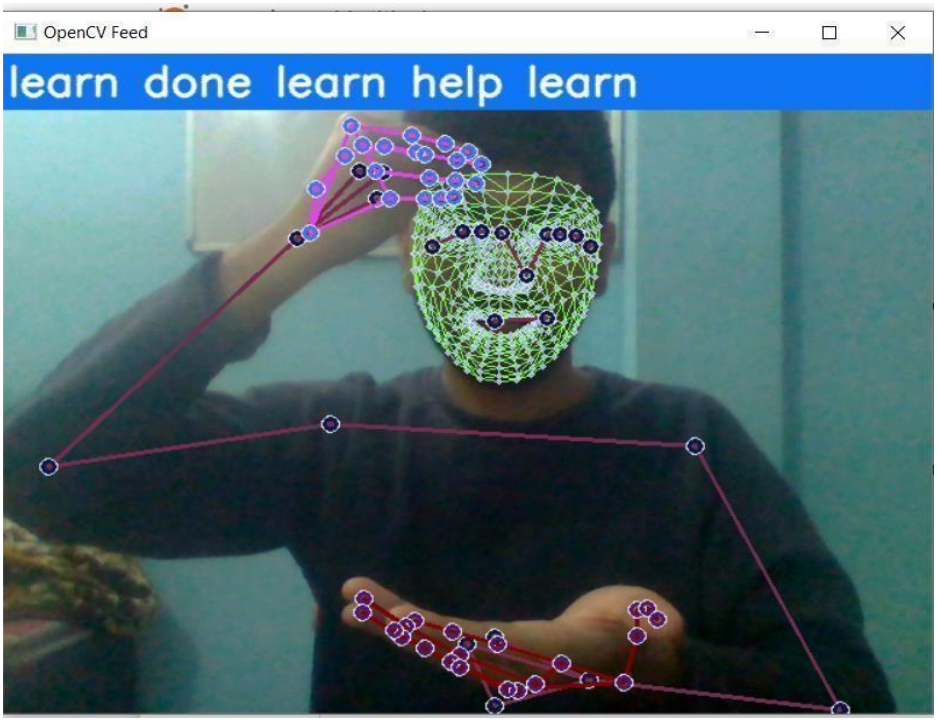


figure 4.7: Output image for Learn

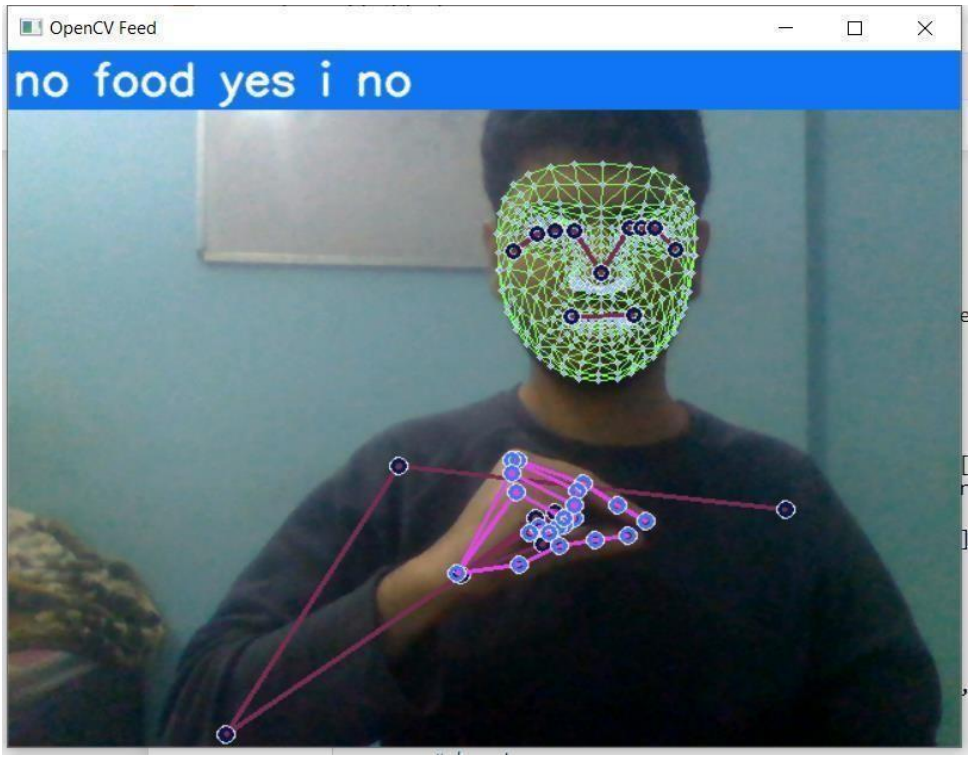


figure 4.8: Output image for No

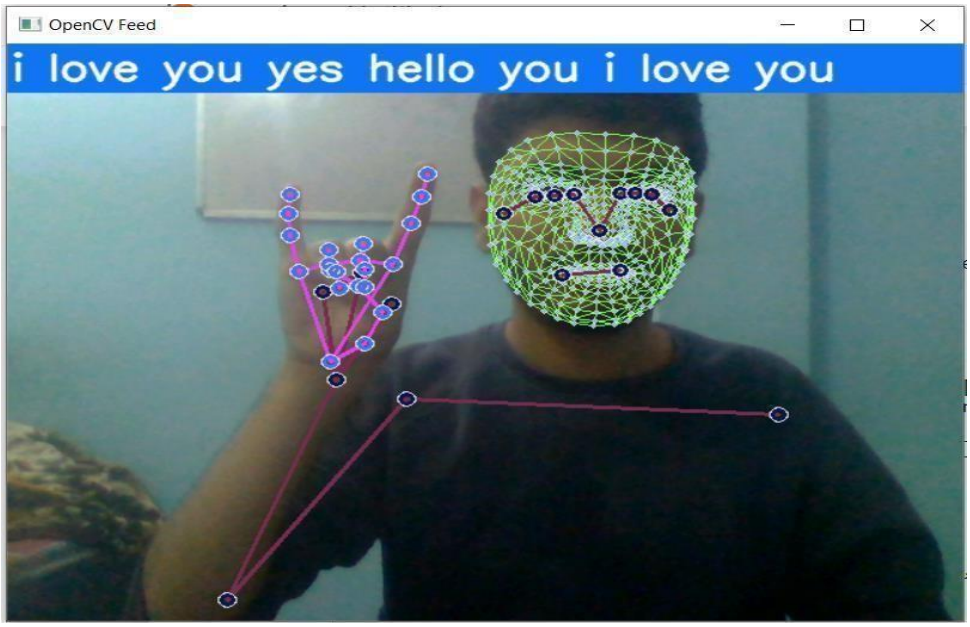


figure 4.9: Output image for I Love You

Chapter 5: Conclusion

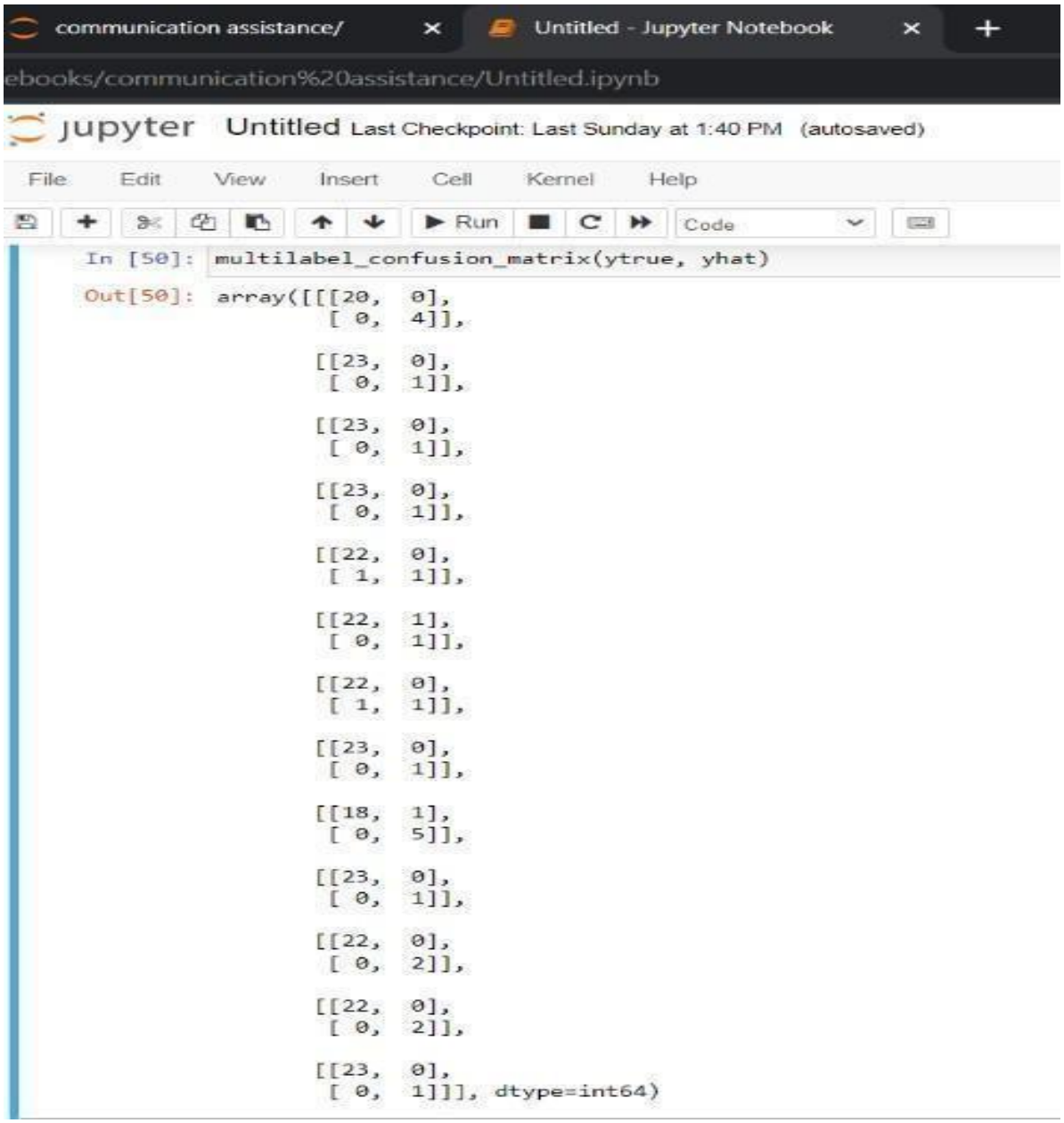


Figure 4.10 Confusion Matrix

```
In [51]: accuracy_score(ytrue, yhat)

Out[51]: 0.9166666666666666
```

Figure 4.11 Accuracy score of model

Our final project is a software system that helps in the communication between people with hearing and speech impairment and people who do not have such restrictions. It utilizes a web camera feed that captures information regarding the gestures of the sign language user.

We are really grateful to Dr. Sukhbir Singh who has been our supervisor, for his unhinged faith and guidance which encouraged and helped us to bring this project to conclusion.

References

[1] Agrawal ;S. & Chakraborty; (2021). Real-Time Hand Gesture Recognition System Using MediaPipe and LSTM. arXiv preprint arXiv:2106.02768.

[2] Yang, W., & Li, Y. (2020). A deep learning-based hand gesture recognition system for human-machine interaction. IEEE Access, 8, 177682-177692.

[3] Liu, J., Huang, Z., & Tang, W. (2019). Deep learning-based hand gesture recognition using depth maps and RGB images. Multimedia Tools and Applications, 78(22), 32227-32243.

Appendix

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
from tkinter import *
from  PIL import Image, ImageTk


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_style_landmarks(image,results):
#       #draw face connections
#       mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACEMESH_TESSELATION,
#
        mp_drawing.DrawingSpec(color=(244,164,96),thickness=1,circle_radius=1),
#
        mp_drawing.DrawingSpec(color=(80,256,121),thickness=1,circle_radius=1)
#                       )
        #draw pose connections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,

        mp_drawing.DrawingSpec(color=(80,22,10),thickness=2,circle_radius=4),

        mp_drawing.DrawingSpec(color=(80,44,121),thickness=2,circle_radius=2)
                        )
        #draw left hand connections
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,

        mp_drawing.DrawingSpec(color=(121,22,76),thickness=2,circle_radius=4),
```

```
        mp_drawing.DrawingSpec(color=(0,0,128),thickness=2,circle_radius=2)
                        )
        #draw right hand connections
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,

        mp_drawing.DrawingSpec(color=(245,117,66),thickness=2,circle_radius=4),

        mp_drawing.DrawingSpec(color=(245,66,230),thickness=2,circle_radius=2)
                        )


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




actions = np.array(['sign','more','learn','yes','no','thank you','sorry','hello','i love
you','please','again','food','done','help','i','you'])
no_sequences = 30

sequence_length = 30

from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM, Dense



# Producting The LSTM model neural network
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,258)))
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'))

res = [0.7, 0.2, 0.1]
actions[np.argmax(res)]

# Compiling the m0del
model.compile(optimizer='Adam', loss='categorical_crossentropy',
metrics=['categorical_accuracy'])

# Loading Old weights or model we trained into the model
model.load_weights(r"C:\Users\virbk\Downloads\BTP_Project\BTP_Project\action.h5")

from sklearn.metrics import multilabel_confusion_matrix, accuracy_score



# tkinter window generation
def window(frame,prediction):
        frame = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        img = ImageTk.PhotoImage(Image.fromarray(frame))
        l1["image"] = img
        l2.config(text="Current Prediction: "+ str(prediction))
        x.update()


# on press key q, quit the file
def onKeyPress(event):
        x.destroy()
        exit()


# This is the actual function, flow starts in here
def detection():
        sequence = []
        sentence = []
        threshold = 0.80

        cap = cv2.VideoCapture(0)
        #set mediapipe model
        with
mp_holistic.Holistic(min_detection_confidence=0.5,min_tracking_confidence=0.5) as
holistic:
        while cap.isOpened():
        ret, frame = cap.read()
        image1, results = mediapipe_detection(frame, holistic)
```

```
        print(results)
        draw_style_landmarks(image1, results)
        keypoints = extract_keypoints(results)
        #sequence.insert(0,keypoints)
        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)])

        #3. Viz logic
                if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                if actions[np.argmax(res)] != sentence[-1]:
                        sentence.append(actions[np.argmax(res)])
                else:
                sentence.append(actions[np.argmax(res)])

                if len(sentence) > 5:
                sentence = sentence[-5:]


                cv2.rectangle(image1, (0,0), (640,40), (245,117,16), -1)
                image1 = cv2.putText(image1, ' '.join(sentence), (3,30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,  cv2.LINE_AA)
                window(image1,actions[np.argmax(res)]) # put image1 here



                #cv2.imshow('OpenCV Feed', image1)
                #Break gracefully
                if cv2.waitKey(10) & 0xFF == ord('q'):
                break
        cap.release()
        cv2.destroyAllWindows()

x = Tk()
x.geometry("900x650")
# x.bind('<KeyPress>', onKeyPress)
x.bind('q', onKeyPress)
x.configure(bg="light blue")
i = 0
l1 = Label(x,font=("times new roman",24))
l1.pack()
l2 = Label(x,font=("times new roman",24))
l2.pack()
l1.place(relx=0.5,rely=0.5,anchor="center")
```

```
l2.place(x=1,y=1)


detection()
cv2.destroyAllWindows()


#Start the window
x.mainloop()
```