**TEAM NAME : QUADSQUAD**

**PROJECT NAME : சைகை துணை - SIGN LANGUAGE SUBTITLE GENERATOR**

**Code for ML model generation using tensorflowa and web app using streamlit :**

```
# import cv2

# import numpy as np

# import av

# import mediapipe as mp

# from streamlit_webrtc import webrtc_streamer,
WebRtcMode, RTCConfiguration

# mp_drawing = mp.solutions.drawing_utils

# mp_drawing_styles = mp.solutions.drawing_styles

# mp_hands = mp.solutions.hands

# hands = mp_hands.Hands(

#     model_complexity=0,

#     min_detection_confidence=0.5,

#     min_tracking_confidence=0.5

# )
```

```
# def process(image):
#     image.flags.writeable = False
#     image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
#     results = hands.process(image)
# # Draw the hand annotations on the image.
#     image.flags.writeable = True
#     image = cv2.cvtColor(image,
cv2.COLOR_RGB2BGR)
#     if results.multi_hand_landmarks:
#       for hand_landmarks in
results.multi_hand_landmarks:
#         mp_drawing.draw_landmarks(
#             image,
#             hand_landmarks,
#             mp_hands.HAND_CONNECTIONS,
#
mp_drawing_styles.get_default_hand_landmarks_style(
),
```

```
#
mp_drawing_styles.get_default_hand_connections_styl
e())
#     return cv2.flip(image, 1)
# RTC_CONFIGURATION = RTCConfiguration(
#     {"iceServers": [{"urls":
["stun:stun.l.google.com:19302"]}]}
# )
# class VideoProcessor:
#     def recv(self, frame):
#         img = frame.to_ndarray(format="bgr24")
#         img = process(img)
#         return av.VideoFrame.from_ndarray(img,
format="bgr24")
# webrtc_ctx = webrtc_streamer(
#     key="WYH",
#     mode=WebRtcMode.SENDRECV,
#     rtc_configuration=RTC_CONFIGURATION,
#     media_stream_constraints={"video": True,
"audio": False},
```

```
#     video_processor_factory=VideoProcessor,
#     async_processing=True,
# )


import streamlit as st
from streamlit_webrtc import webrtc_streamer,
VideoProcessorBase
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
from tensorflow import keras
import av


mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
```

```
modelF= keras.models.load_model('rec_0.h5')









class OpenCamera (VideoProcessorBase):

    def __init__(self) -> None :

        self.sequence = []

        self.sentence = []

        self.threshold = 0.4

        # self.actions = np.array(['hello',
'thanks', 'i love you', 'stop', 'please', 'walk',
'argue', 'yes', 'see', 'good'])

        self.acitons =
np.array(['வணக்கம்','நன்றி','நான் உன்னை
காதலிக்கிறேன்', 'நிறுத்து', 'ஆம்'])
```

```
    def mediapipe_detection(self,image, model):
        self.image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
        self.image.flags.writeable = False
        self.results = model.process(image)
        print(self.results)
        self.image.flags.writeable = True
        self.image = cv2.cvtColor(image,
cv2.COLOR_RGB2BGR)
        return self.image, self.results



    def draw_styled_landmarks(self,image, results):



        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)
                                )

        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(self, results):

        self.key1 = 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)

        self.key2 = 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)

        self.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)

        self.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([self.key1,
self.key2, self.lh, self.rh])




    def recv(self, frame):

        img=frame.to_ndarray(format="bgr24")

        with
mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
```

```
            image, results =
self.mediapipe_detection(img,holistic)

            self.draw_styled_landmarks(image,
results)

            # 2. Prediction logic

            keypoints =
self.extract_keypoints(results)



            self.sequence.append(keypoints)

            self.sequence = self.sequence[-30:]


            if len(self.sequence) == 30:

                res =
modelF.predict(np.expand_dims(self.sequence,
axis=0))[0]




            #3. Viz logic

                if res[np.argmax(res)] >
self.threshold:
```

```
                    if len(self.sentence) > 0:
                        if
self.actions[np.argmax(res)] != self.sentence[-1]:
self.sentence.append(self.actions[np.argmax(res)])
                    else:
self.sentence.append(self.actions[np.argmax(res)])

                if len(self.sentence) > 1:
                    self.sentence =
self.sentence[-1:]

                    # Viz probabilities
                    # image = prob_viz(res,
actions, image, colors)

            cv2.rectangle(image, (0,0), (640, 40),
(245, 117, 16), -1)
            cv2.putText(image, '
'.join(self.sentence),
```

```
(3,30),cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255,
255), 2, cv2.LINE_AA)



    #    av.VideoFrame.from_ndarray(image,
format="bgr24")

        return av.VideoFrame.from_ndarray(image,
format="bgr24")



st.sidebar.title('Quadsquad')

app_mode = st.sidebar.selectbox('Select Page',
['Home', 'Demo'])



if app_mode == 'Home':

    st.title('About Our Project')

    st.markdown("")

elif app_mode == 'Demo':

    st.header('Real-Time Hand Gesture Recognition
Using Mediapipe & LSTM')

    st.markdown('To start detecting your ASL
gesture click on the "START" button')

    ctx = webrtc_streamer(
```

```
    key="example",
    video_processor_factory=OpenCamera,
    rtc_configuration={ # Add this line
        "iceServers": [{"urls":
["stun:stun.l.google.com:19302"]}]
    }, media_stream_constraints={"video": True,
"audio": False,}, async_processing=True
)
```

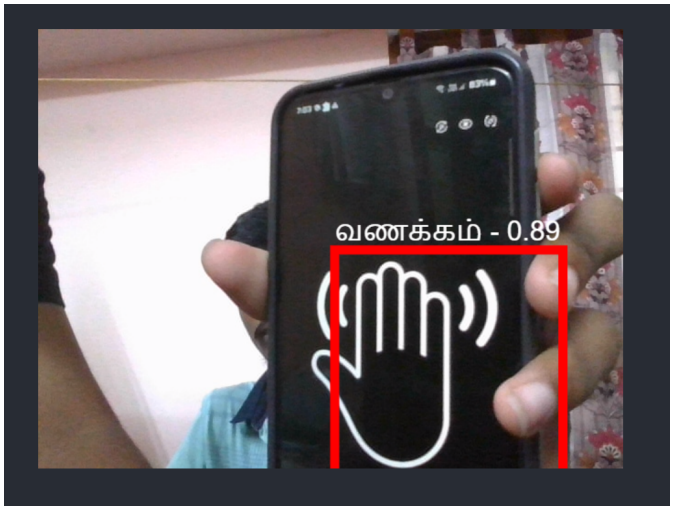
## Sign Language for ‘NO’:

## Sign Language for ‘HELLO’:

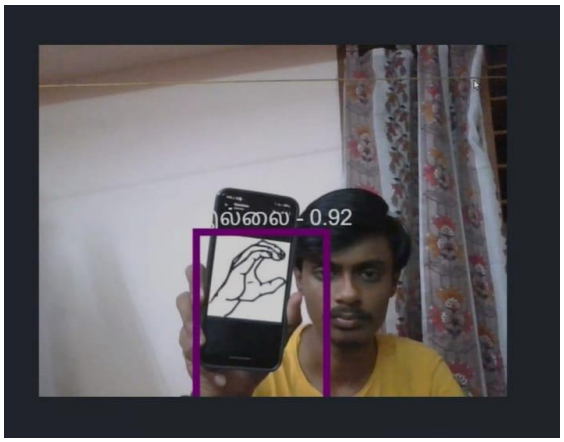
## Sign Language for ‘I LOVE YOU’:

## Sign Language for ‘YES’:

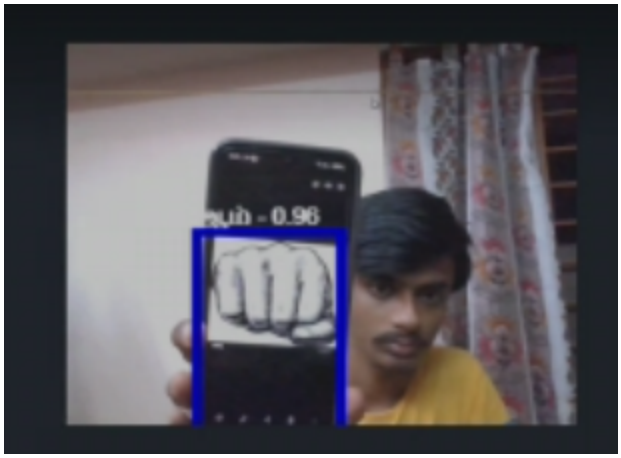
## Detection by web app:



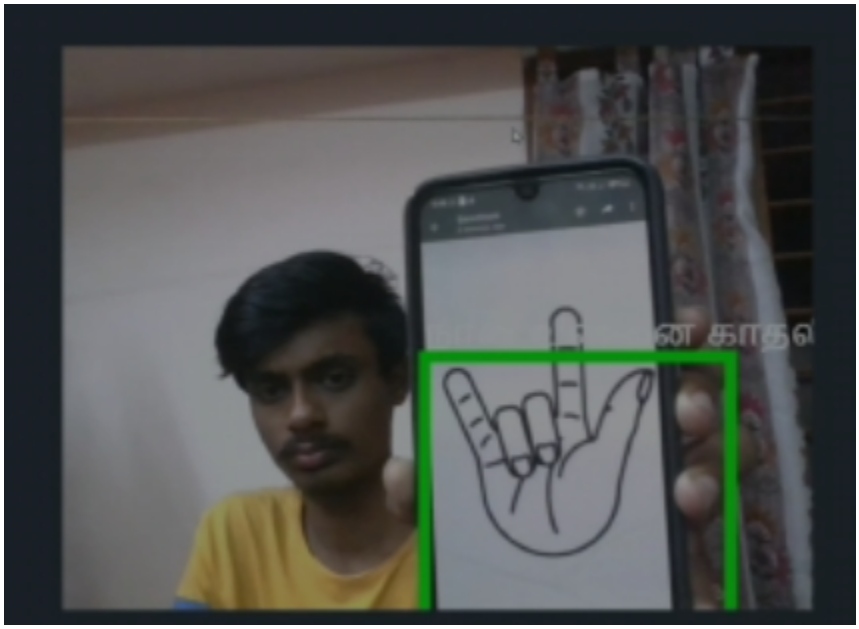

**(Here, the word Hello - வணக்கம் is detected by our ML model with 89% accuracy)**




**(Here, the word No - இல்லை is detected by our ML model with 92% accuracy)**




**(Here, the word Yes - ஆம் is detected by our ML model with 96% accuracy)**



**(Here, the word I Love You - நான் உன்னை காதலிக்கிறேன் is detected by our ML model with 95% accuracy)**

**Demo video drive link:**

https://drive.google.com/file/d/1ftwVloOLIkboVC9Sr17N6KN1pX6TpYVq/view?usp=sharing

**Ppt link:**

https://docs.google.com/presentation/d/1tPqlxvEhbyJ-jcFwEivxOOR7MrBQIKr-/edit?usp=sharing&ouid=102750549371353706278&rtpof=true&sd=true