AUTOMATION PROJECT

PR1101

REPORT

Institute of Engineering and Technology (IET), Jaipur

FACULTY GUIDE

Dr. Devika Kataria

PREPARED BY

DEVANAPALLY CHARAN TEJ(2020B.TECHCSE027)

MADIKANTI NAVEEN(2020BTECHCSE043)

RANERU VENKATESH(2020B.TECHCSE107)

E.VENKATA SAI SARAN(2020B.TECHCSE106)

NOVEMBER 2022

TABLE OF CONTENTS

ABSTRACT

Communication is the primary by which people communicate with one another. There has been a rapid increase in the number of deaf and dumb victims in recent years as a result of birth defects, accidents, and oral diseases. Because deaf and dumb people cannot communicate with normal people, they must on visual communication. We want to create a small intelligent hand gesture recognition system that can capture, localise, and translate hand gestures into text and speech. For this, we investigated existing technologies such as microcontroller and sensor-based gesture vocalizers. To understand gestures, we used image recognition.

PROBLEM STATEMENT

PROBLEM:

People with speech impairments and hearing loss rely on sign language to communicate, but because the general public is unaware of sign language, it is difficult to communicate with the able-bodied.

BACKGROUND:

Communication is the primary means by which people communicate with one another. There has been a rapid increase in the number of deaf and dumb victims in recent years as a result of birth defects, accidents, and oral diseases. Because deaf and dumb people cannot communicate with normal people, they must rely on visual communication.

In India, approximately 12.3 million people have moderate to severe hearing loss. Only four and a half million of these would be unable to succeed in a hearing school but could obtain an education in a deaf school if one was available. These deaf would then be exposed to sign language and could eventually become members of the Deaf community.

In most Indian metropolitan areas, Indian Sign Language (ISL) is a language of broader communication, and it may be the mother tongue of the Deaf in several of these areas. It was previously assumed that the Deaf of India spoke various dialects of ISL. However, current research among Bengali and other Deaf communities shows that the Deaf speak a completely different language in many places. Those who can try to use ISL with researchers because it is more prestigious, but the majority of these communities are unfamiliar with ISL.

There are 478 government-funded schools and approximately 372 private schools for the deaf spread across India. The "oral approach" is used in the majority of these schools' classrooms. It is an unusual school that employs signs in the classroom. Rural Indian Deaf people frequently do not receive an education because of the distance they must travel to school and the need for labourers at home.

Usually they use hand gesture and sign language to communicate. But very few people are trained in hand gesture understanding, thus they face communication gap between -special person and normal person. This huge challenge makes them uncomfortable and they feel discriminated in society. Because of miss communication Deaf and Dumb people feel not to communicate and hence they never able to express their feelings.

Objective:

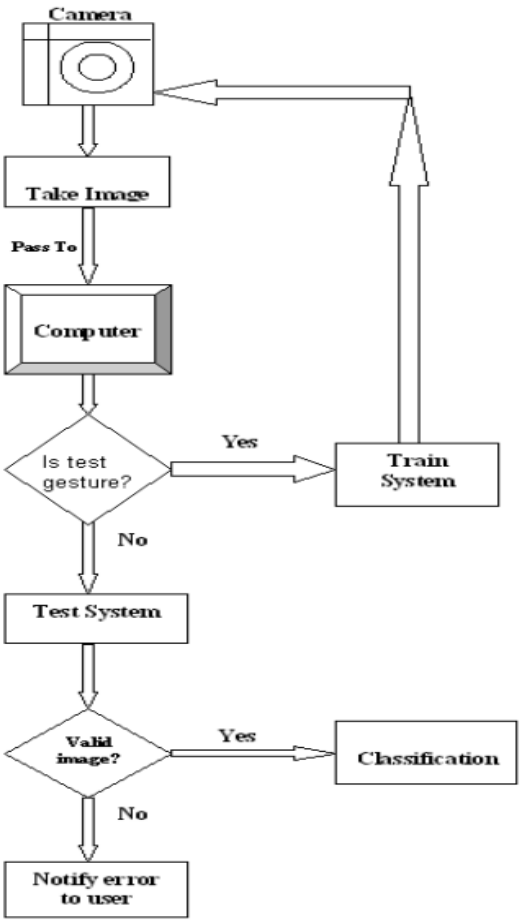
We aim to build a compact intelligent hand gesture recognition system that captures, localizes and translate hand gestures to appropriate text form and speech form.

METHODOLOGY

There have been numerous researches in this field and several methodologies were proposed like Principle Component Analysis (PCA) method, gradient method, subtraction method etc. PCA relates to Linear transformation consist on statistical approach. This gives us powerful tool for pattern recognition and data analysis which mostly used in image processing techniques for data . Gradient method is also another image processing technique that detect colour patches applying low pass filters is also known as edge detection method. Subtraction method is very simple that subtract input image pixel to another image or constant value to provide output. I have also studied different approaches to hand gesture recognition and came to know that implementation of such techniques like PCA and Gradient method is complicated, we can produce same output as these techniques gives us by simple and easy implementation. So, I have tried four different algorithms and finally selected the one, which was most efficient i.e. diagonal sum algorithm. This algorithm is able to recognize maximum gestures correctly.

RECOGNITION:

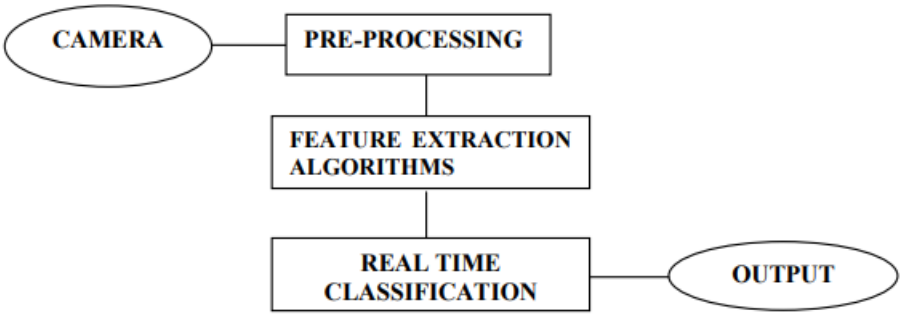
Hand detection and recognition have been significant subjects in the field of computer vision and image processing during the past 30 years. There have been considerable achievements in these fields and numerous approaches have been proposed. However, the typical procedure of a fully automated hand gesture recognition system.



Brief outline of the implemented system

Hand gesture recognition system can be divided into following modules:

- Preprocessing
- Feature extraction of the processed image
- Real time classification



The task of differentiating the skin pixels from those of the background is made considerably easier by a careful choice of lighting. According to Ray Lockton, if the lighting is constant across the view of the camera then the effects of self-shadowing can be reduced to a minimum.

We propose a vision-based approach to accomplish the task of hand gesture detection. As discussed above, the task of hand gesture recognition with any machine learning technique suffers from the variability problem. To reduce the variability in hand recognition task we assume the following assumptions:

- Single colored camera mounted above a neutral colored desk.
- User will interact by gesturing in the view of the camera.
- Training is must.
- Hand will not be rotated while image is capturing

THE WEBCAM SYSTEM (USB PORT)

In my project web cam was attached via USB port of the computer. The web cam worked by continually capturing the frames. In order to capture a particular frame, the user just need to select the particular Algorithm METHOD button on the interface and the hand was detected in the particular frame. The web cam took color pictures, which were then converted into grayscale format. The main reason of sticking to grayscale was the extra amount of processing required to deal with color images,

Resolution: 640x480

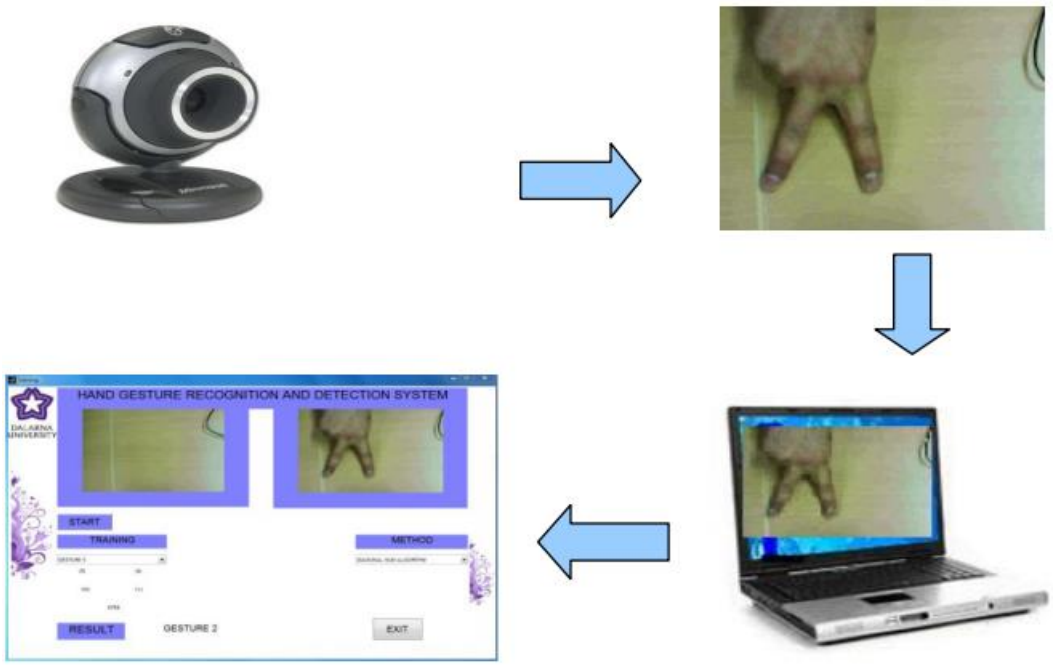
Video frame rate: 30fps @640x480

Pixel depth: Minimum 1.3-mega pixels

Connection port: USB

REAL TIME CLASSIFICATION

shows the concept for real time classification system. A hand gesture image will be passed to the computer after being captured through camera at run time and the computer will try to recognize and classify the gesture through computer vision.



JETSON NANO(HARDWARE)

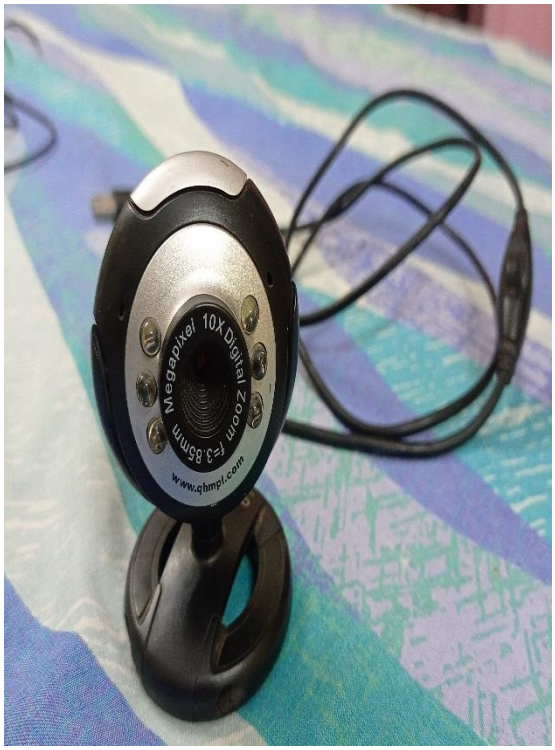
The Jetson Nano module is a small AI computer that has the performance and power efficiency needed to run modern AI workloads, multiple neural networks in parallel, and process data from several high-resolution sensors simultaneously.

This makes it the perfect entry-level option to add advanced AI to embedded products.The Jetson software stack begins with NVIDIA JetPack™ SDK, which provides Jetson Linux, developer tools, and CUDA-X accelerated libraries and other NVIDIA technologies.

Jetson Nano is a small, powerful computer designed to power entry-level edge AI applications and devices. Get started quickly with the comprehensive NVIDIA JetPack™ SDK, which includes accelerated libraries for deep learning, computer vision, graphics, multimedia, and more. Jetson Nano brings real-time computer vision and inferencing across a wide variety of complex Deep Neural Network (DNN) models. These capabilities enable multi-sensor autonomous robots, IoT devices with intelligent edge analytics, and advanced AI systems.

HARDWARE:

- Minimum 2.8 GHz processor Computer System or latest
- JETSON NANO Module
- Web cam (For real-time hand Detection)



WEB CAMERA

JETSON NANO

Software:

- PYTHON 3.6
- TENSOR FLOW
- KERAS
- JETSANNANO
- CAMERA

Dataset:

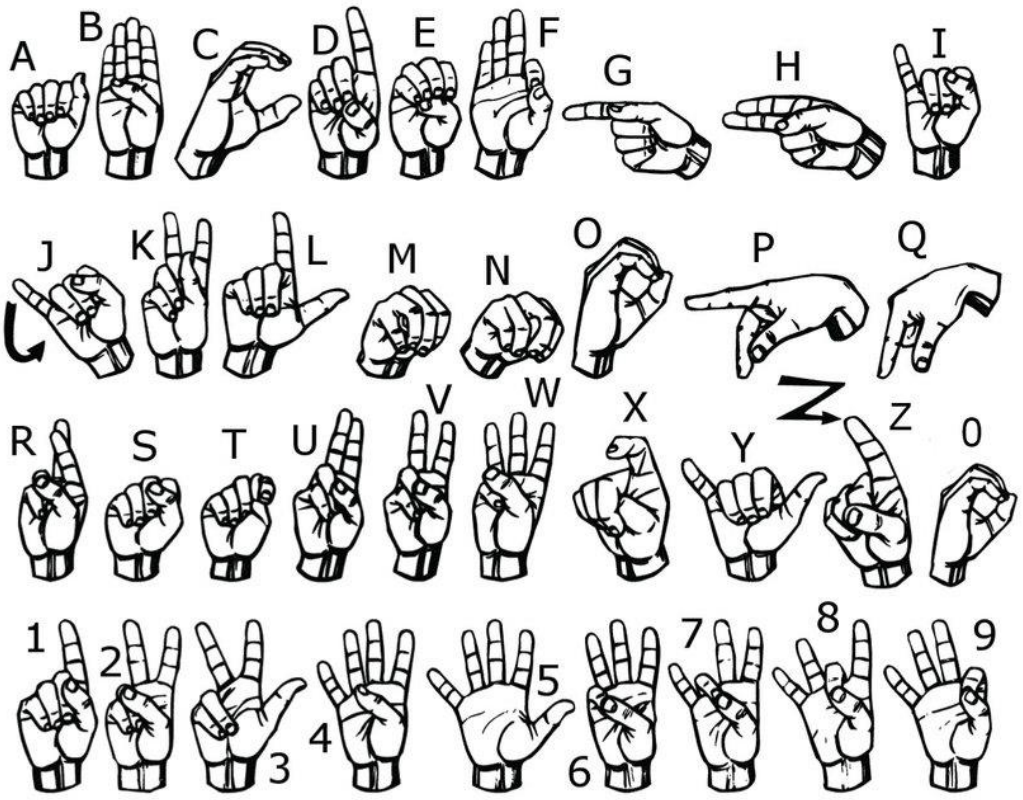
We choose American Sign Language for our dataset. In that we choose all the 26 Alphabets and related gestures.

Creating dataset:

If a Computer Vision (CV) related application deals with detecting or tracking a specific object, then it is necessary to determine the range of HSV (Hue, Saturation, and Value) values of that object. This range is required to be specified as part of the coding to detect that object. If the correct range is not specified, the CV algorithm may pick-up noises as well, besides the actual object, leading to false detection and tracking.

Using OpenCV images were captured, cropped and stored. These RGB images were converted to HSV style.Thresholding and masking was done in HSV color space.

The exact HSV range was determined programmatically using OpenCV for an object to be identified or tracked. We created slide bars in code to adjust ranges and get the desired area we want. Then we took the corresponding HSV range and use them in our code .



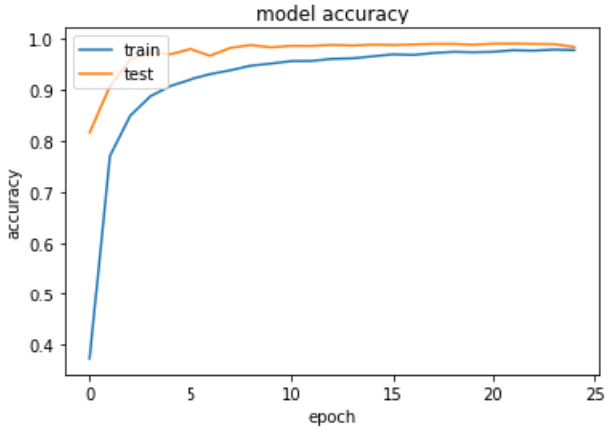
Deployment:

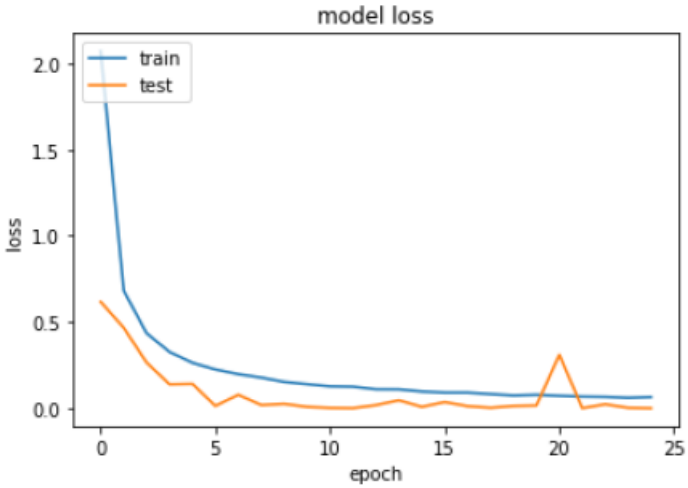
Model in .h5 format was stored with weights. This model will be deployed on RaspberryPi.

Due to technical errors (frequently crashes on boot) on Raspberry Pi, we didn't deployed it. But we have prepared code for deployement anytime board gets corrected and tested it on other linux system.

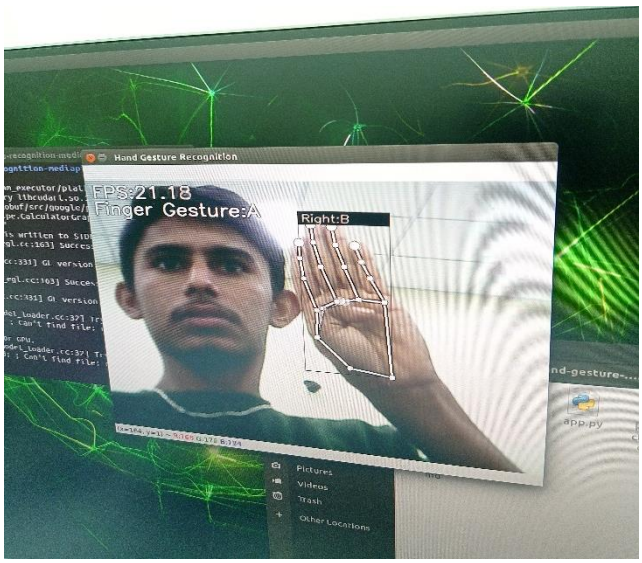
Raspberry Pi board with OpenCV and Keras installed will be suitable for deployment

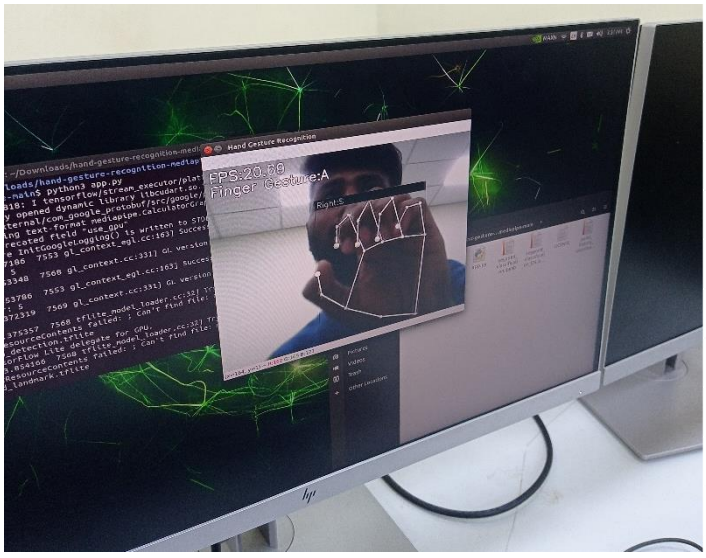
TESTING AND EFFICIENCY

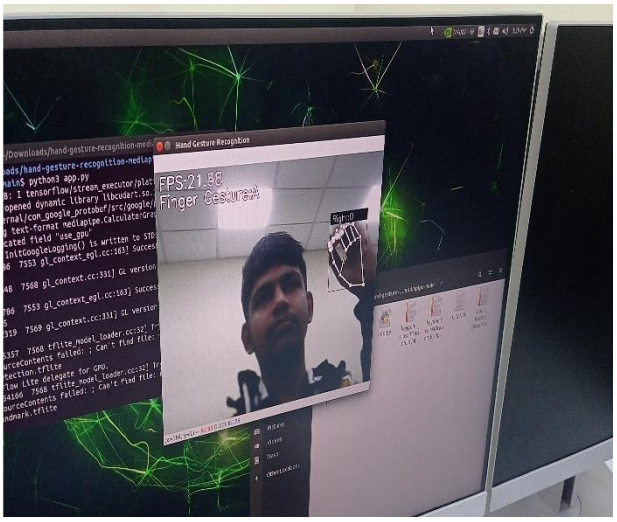


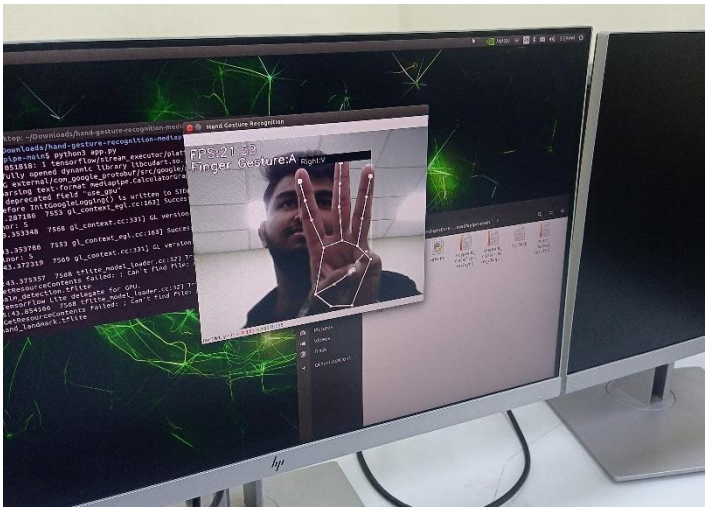


ACCURACY:94.57%









Demonstration Video Link: [Understanding Sign Language Demonstration Video.mp4]

CONCLUSIONS AND FUTURE SCOPE

We conclude that through this project we people can connect all people together. At least this can be done remotely. In the future, this project can be improved by the Advancements

1. This can be connected to any video calling app like zoom to help in online.
2. This can be converted to captions a series of words and sentences.
3. This can also help the captions to be spelled out as speech.

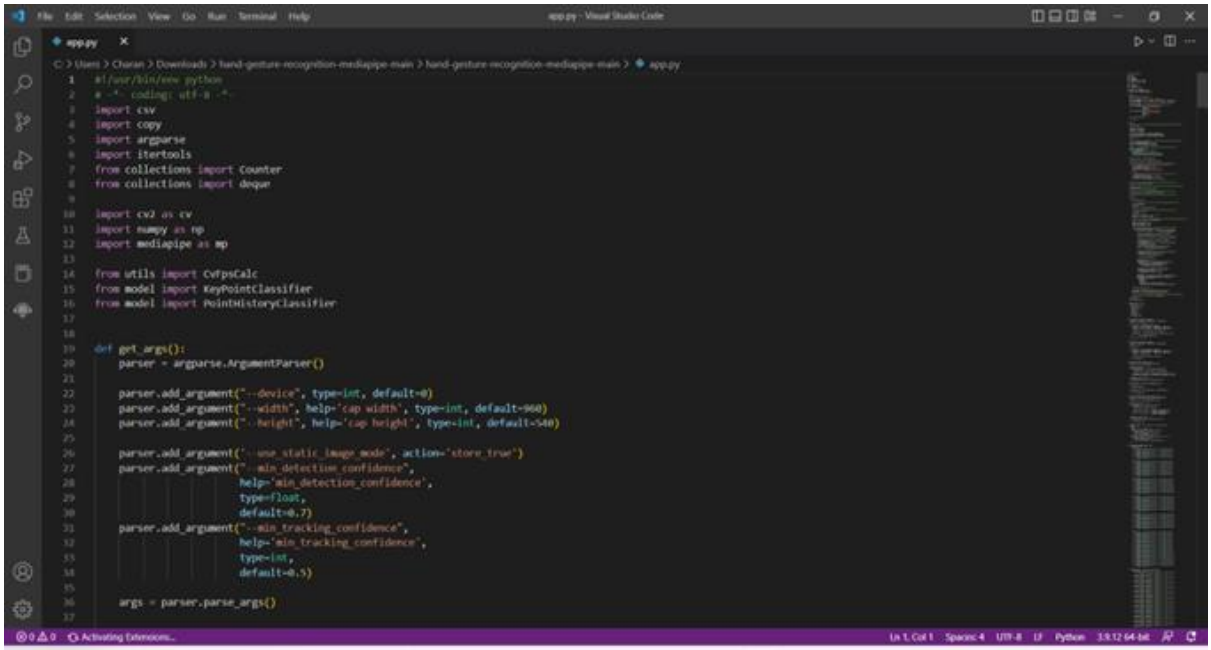
REFERENCES

A. Voskou, K. P. Panousis, D. Kosmopoulos, D. N. Metaxas and S. Chatzis, "Stochastic Transformer Networks with Linear Competing Units: Application to end-to-end SL Translation," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 11926-11935, doi: 10.1109/ICCV48922.2021.01173.

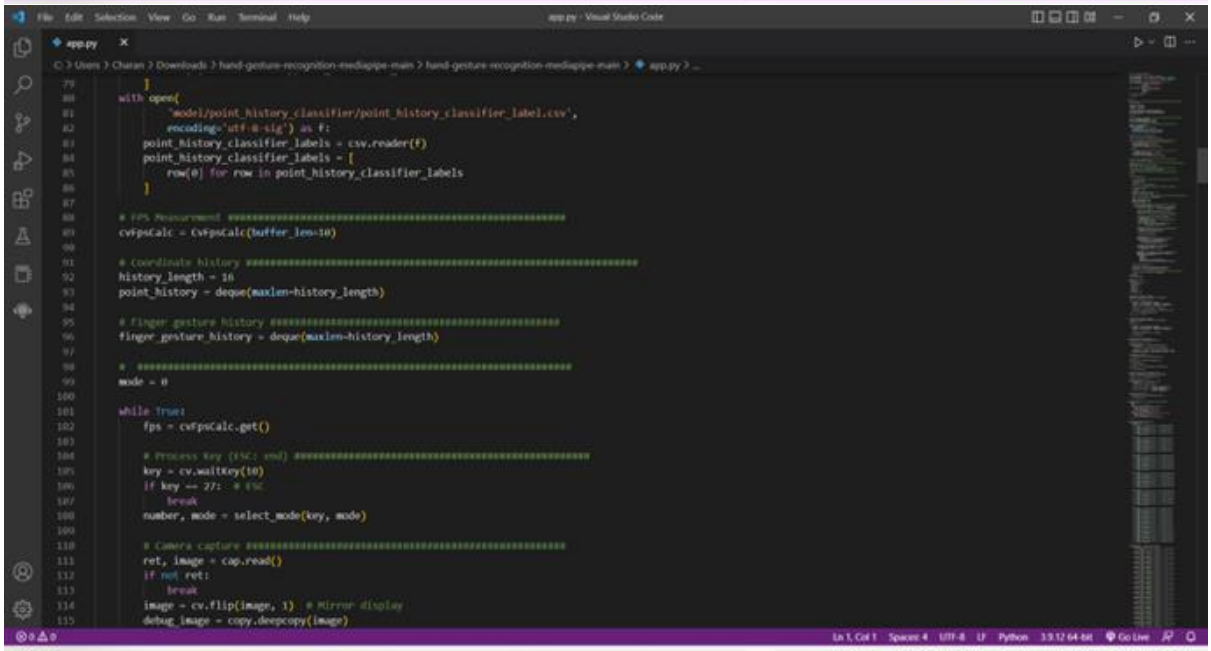
www.youtube.com

APPENDICES

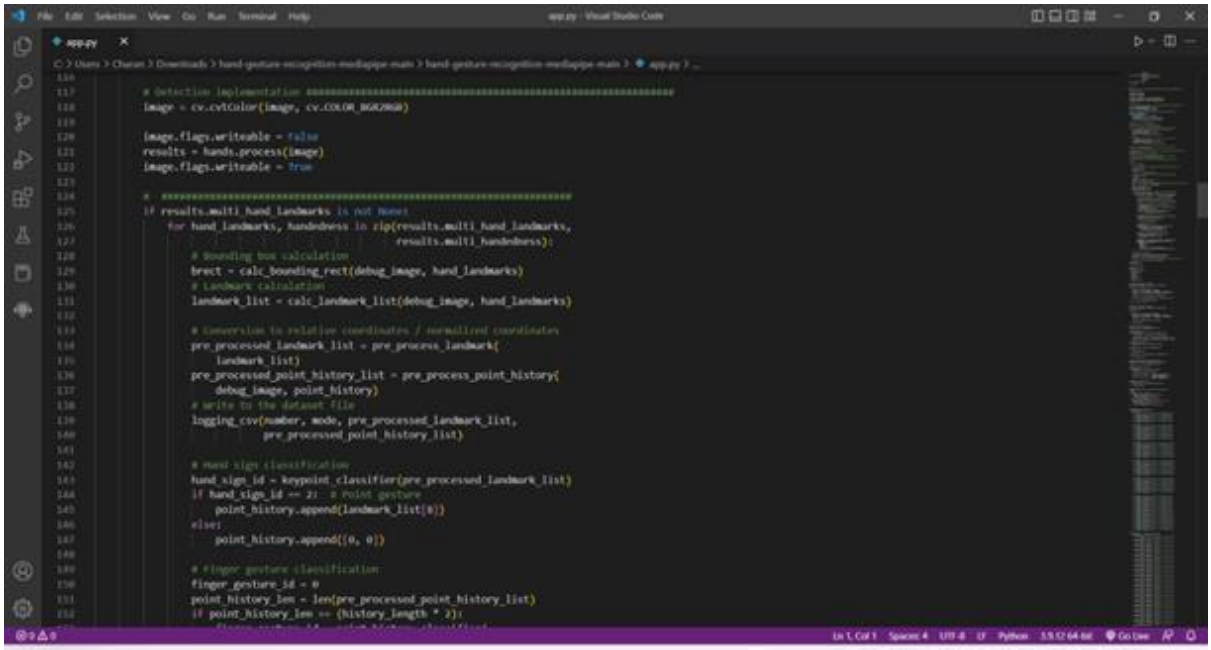
Sample codes:



```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import csv
import copy
import argparse
import itertools
from collections import Counter
from collections import deque

import cv2 as cv
import numpy as np
import mediapipe as mp

from utils import CvFpsCalc
from model import KeyPointClassifier
from model import PointHistoryClassifier


def get_args():
    parser = argparse.ArgumentParser()

    parser.add_argument("--device", type=int, default=0)
    parser.add_argument("--width", help='cap width', type=int, default=960)
    parser.add_argument("--height", help='cap height', type=int, default=540)

    parser.add_argument('--use_static_image_mode', action='store_true')
    parser.add_argument("--min_detection_confidence",
                        help='min_detection_confidence',
                        type=float,
                        default=0.7)
    parser.add_argument("--min_tracking_confidence",
                        help='min_tracking_confidence',
                        type=int,
                        default=0.5)

    args = parser.parse_args()
```



```python
        ]
    with open(
            'model/point_history_classifier/point_history_classifier_label.csv',
            encoding='utf-8-sig') as f:
        point_history_classifier_labels = csv.reader(f)
        point_history_classifier_labels = [
            row[0] for row in point_history_classifier_labels
        ]

    # FPS Measurement ########################################################
    cvFpsCalc = CvFpsCalc(buffer_len=10)

    # Coordinate history #################################################################
    history_length = 16
    point_history = deque(maxlen=history_length)

    # Finger gesture history ################################################
    finger_gesture_history = deque(maxlen=history_length)

    #  ########################################################################
    mode = 0

    while True:
        fps = cvFpsCalc.get()

        # Process Key (ESC: end) #################################################
        key = cv.waitKey(10)
        if key == 27:  # ESC
            break
        number, mode = select_mode(key, mode)

        # Camera capture #####################################################
        ret, image = cap.read()
        if not ret:
            break
        image = cv.flip(image, 1)  # Mirror display
        debug_image = copy.deepcopy(image)
```



```
        # Detection implementation #############################################################
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

        image.flags.writeable = False
        results = hands.process(image)
        image.flags.writeable = True

        #  ####################################################################
        if results.multi_hand_landmarks is not None:
            for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
                                                  results.multi_handedness):
                # Bounding box calculation
                brect = calc_bounding_rect(debug_image, hand_landmarks)
                # Landmark calculation
                landmark_list = calc_landmark_list(debug_image, hand_landmarks)

                # Conversion to relative coordinates / normalized coordinates
                pre_processed_landmark_list = pre_process_landmark(
                    landmark_list)
                pre_processed_point_history_list = pre_process_point_history(
                    debug_image, point_history)
                # Write to the dataset file
                logging_csv(number, mode, pre_processed_landmark_list,
                            pre_processed_point_history_list)

                # Hand sign classification
                hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
                if hand_sign_id == 2:  # Point gesture
                    point_history.append(landmark_list[8])
                else:
                    point_history.append([0, 0])

                # Finger gesture classification
                finger_gesture_id = 0
                point_history_len = len(pre_processed_point_history_list)
                if point_history_len == (history_length * 2):
```