Summer Project On

Real Time Sign Language Interpreter for Video Conferencing Applications

By

Suraj Prakkash Nair (2021510037)
Jay Umesh Oswal (2021510040)

Under the guidance of

Internal Supervisor

Prof. Harshil Kanakia

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University

2021-22

CERTIFICATE OF APPROVAL

This is to certify that the following student

Suraj Prakkash Nair (2021510037)
Jay Umesh Oswal (2021510040)

Have satisfactorily carried out work on the project entitled

"Real Time Sign Language Interpreter for Video Conferencing Applications"

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2021-22.

Project Guide:
Prof. Harshil Kanakia

PROJECT APPROVAL CERTIFICATE

This is to certify that the following student

Suraj Prakkash Nair (2021510037)
Jay Umesh Oswal (2021510040)

Have successfully completed the Project report on

"Real Time Sign Language Interpreter for Video Conferencing Applications",

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER EXTERNAL EXAMINER

HEAD OF DEPARTMENT PRINCIPAL

Contents

Abstract

There is a challenge for mute and deaf people to actively participate and communicate during video conferencing. By using our application, hearing challenged people can communicate using sign language and the opposite party can understand after the sign is translated into text and audio in real time. Using this program, mute people can communicate using American Sign Language during video conferencing.

The user will have the ability to use our application as a virtual camera and microphone in any existing video conferencing applications like Meet, Zoom, Teams, WebEx etc. Thus making exchange of Information easier for them.

This will provide one-way communication as a person who doesn't understand sign language will get the meaning of the hand signs shown to them. Also, to make two-way communication possible, this application also presents text audio is played to sign language conversion.

Objectives

This cross platform application is used

- To detect sign language from video captured by camera, to classify detection into correct gesture and convert it to text and audio output.
- Deaf and hard hearing people can exchange information efficiently in video conferences without the need of human sign language interpreters.

List of Figures

List of Tables

1 Introduction

1.1 Problem Definition

To eliminate need of sign language interpreter and provide a solution for deaf and hard hearing people to exchange information effectively in video conference application.

1.2 Objectives and Scope

1.2.1 Objectives

This cross platform application is used

- To detect sign language from video captured by camera, to classify detection into correct gesture and convert it to text and audio output.
- Deaf and hard hearing people can exchange information efficiently in video conferences without the need of human sign language interpreters.

1.2.2 Scope

The user will effortlessly integrate our application into video meets and actively participate in it.

They can also train the model with their custom hand signs and gestures for superior experience.

This application will play identified gesture's audio through microphone which will played through speakers for other users.

This application will put the identified gesture's text on the video feed of user which will be available as camera's feed for other users.

1.3 Existing System

Currently no as such complete solution is available for the solution of mentioned problem, although solutions for sign language interpreter are available but they are not able to integrate with external video conferencing applications.
Following points are recognized as limitations for already existing solutions:

- Solutions just for Sign Language Interpreter
  Various project papers and research publications have provided solutions for just interpreting.
- Standalone application
  They are supposed to be used just as standalone application and cannot be integrate with other applications like zoom, meet, teams etc.

1.4 Proposed System

A desktop application to achieve two things namely, interpret sign language and integrate it with conferencing applications to provide audio and text of interpreted signs.

This will be achieved by using Python, OpenCV, Tensorflow and Keras to detect hand in a video feed and classify it into correct gesture.

User will just select camera and microphone as our application and use the conferencing apps as usual.

Some of the advantages of our application are as follows :

- Interpret Sign Language
  American Sign Language gestures are able to be recognized by this app.
- Compatible for all Conferencing applications
  Video Conferencing applications which give users the ability to choose their own video and microphone sources can use this application to achieve the aforementioned result.
- No User training required
  Just via integrating this app with video meets, users are ready to interpret sign language.
- Select and use interface.
  No additional hardware resource required.

1.5 System Requirements

Table 1.5.1: System Requirements

Operating System	Windows, MacOS, Linux
Hardware	Functioning Camera
Additional Software	OBS Studio, Video Conferencing Applications, Python 3.7.0
Additional Drivers	VB Virtual Cable

2 Software Requirement Specification (SRS)

2.1 Purpose

The project aims at building a machine learning model that will be able to classify the various hand gestures used for finger spelling in sign language. In this user independent model, classification machine learning algorithms are trained using a set of image data and testing is done on a completely different set of data.

2.2 Definition

To provide complete solution for deaf people to actively participate in video conferences.

2.3 Overall Description

2.3.1 Product Functions

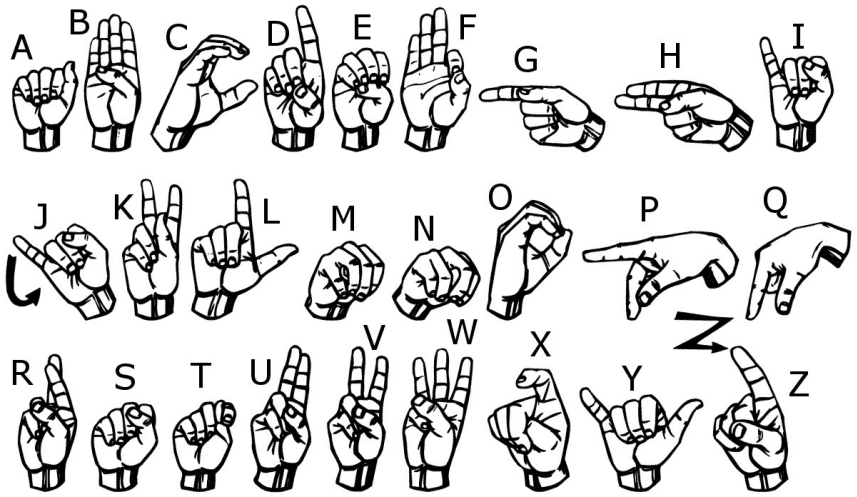
The product function includes:

1. The most common sign language is American sign language. The only means of communication for DM persons are sign languages since their single communication-related handicap prevents them from using spoken languages.
2. The process of communicating involves exchanging ideas and messages via a variety of means, including voice, signalling, behaviour, and pictures.
3. People who are Deaf and Mute (Dumb) (DM) use their hands to make a variety of gestures to communicate with others.
4. The nonverbal communication that takes place through gestures is comprehended visually. Sign language is the nonverbal form of communication used by the deaf and the dumb.

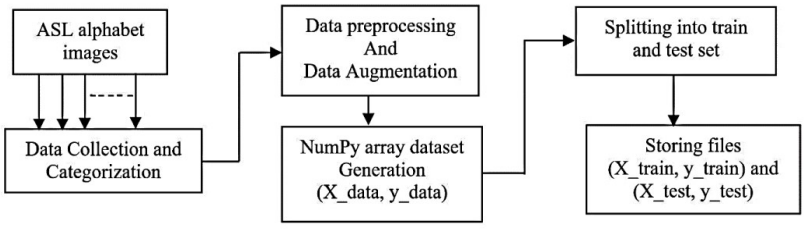
3 Data Acquisition

3.1 Description

The images of various alphabets of American Sign Language were collected using different webcams from different laptops. At first, a data collection program was created using OpenCV library packages in Python. This program has two versions: first one being the manual image capture version, in which all the images were captured manually with varying background and hand poses. Since it was time consuming, we ended up making a second version,video capture and split frames version, which would capture the video of the hand gesture in different background and automatically split it into the desired number of frames. The dataset is created by placing respective images for various alphabets inside a folder named after that alphabet, for instance, all the images collected for a category "Okay" are placed inside a folder named "Data/Okay", the folder name acts as the labels for training the dataset so this is important. With these things in mind, the current dataset is being created. And finally, captured images are also resized to 300 by 300 pixels.



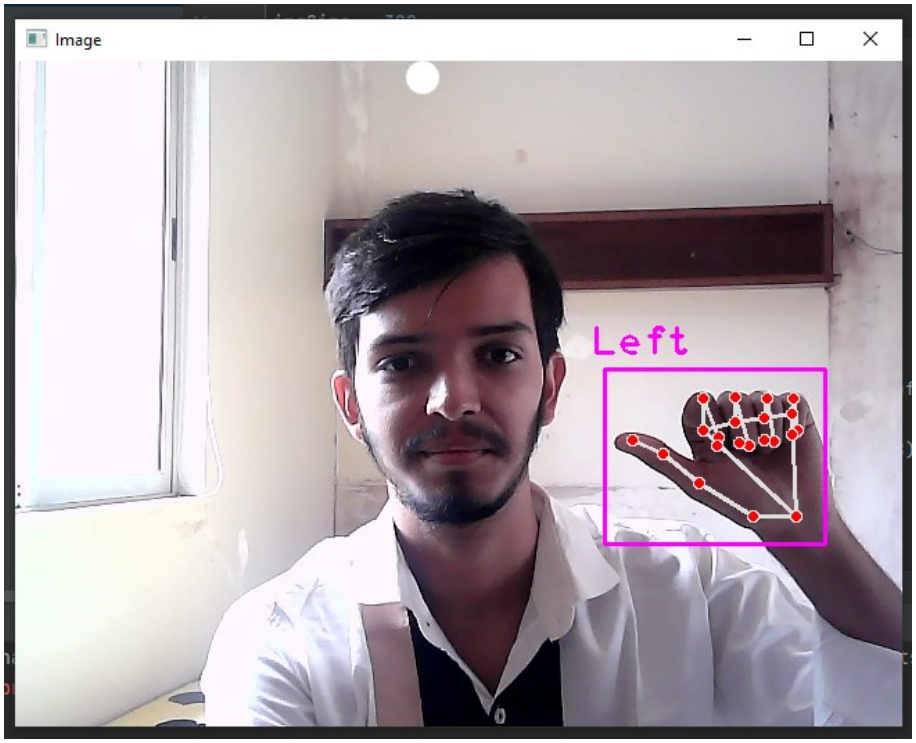
3.1.1: ASL Alphabets



3.1.2: Block diagram of data acquisition and train-test dataset generation

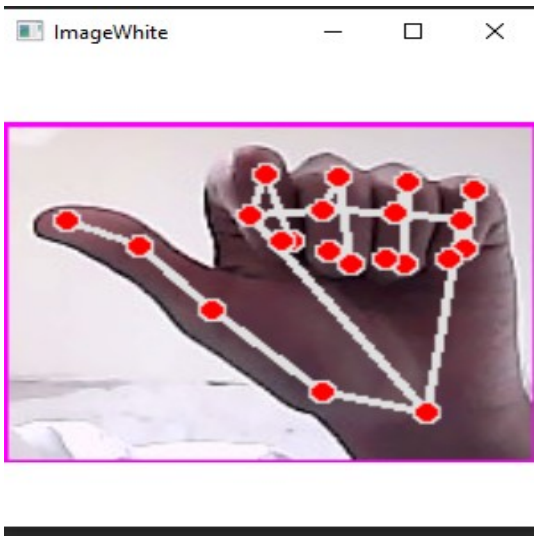
3.2 Images Collection

3.2.1 Detect Hand in video feed



3.2.1: Detech hand in video feed

3.2.2 Crop only hand with skeleton

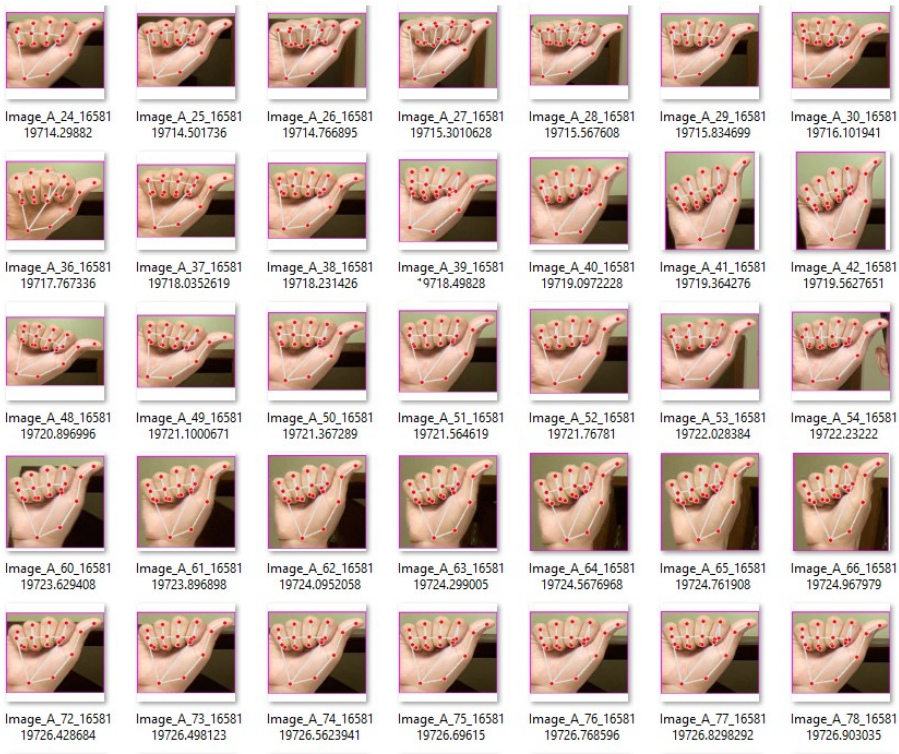


3.2.2: Crop only hand with skeleton

3.2.3 Save images within 300 by 300 by adding borders for gap

3.2.3: Save images within 300 by 300 by adding borders for gap

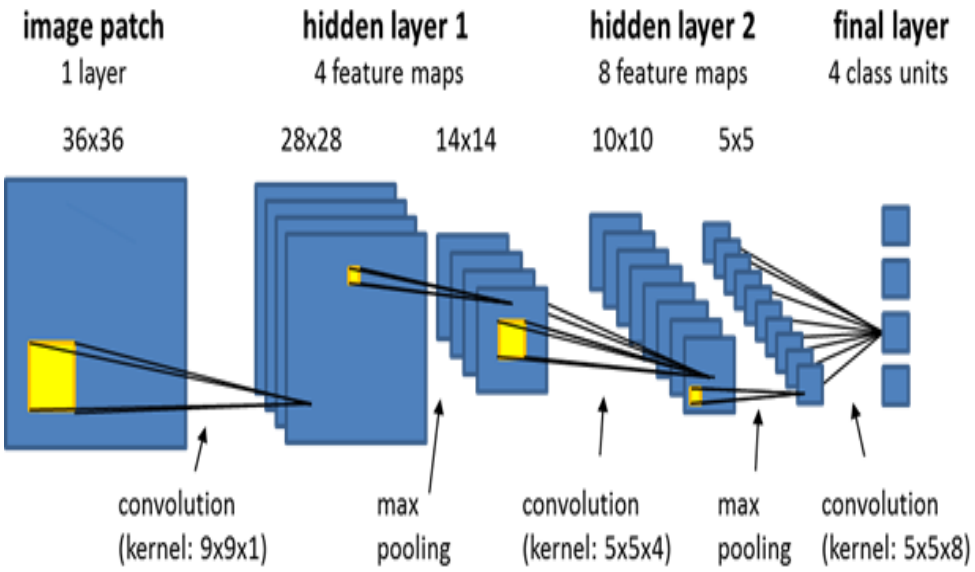
3.2.4 Save all cropped images



3.2.4: Save all cropped images

4 Training and Keras Model Creation Using CNN

4.1 Convolutional Neural Network(CNN)



4.1.1: Convolutional Neural Network

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth.

The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner.

Moreover, the final output layer would have dimensions(number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

1. Convolutional Layer: In convolution layer I have taken a small window size [typically of length 5*5] that extends to the depth of the input matrix.

The layer consists of learnable filters of window size. During every iteration I slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.

As I continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position.

That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour.

2. Pooling Layer: We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters.

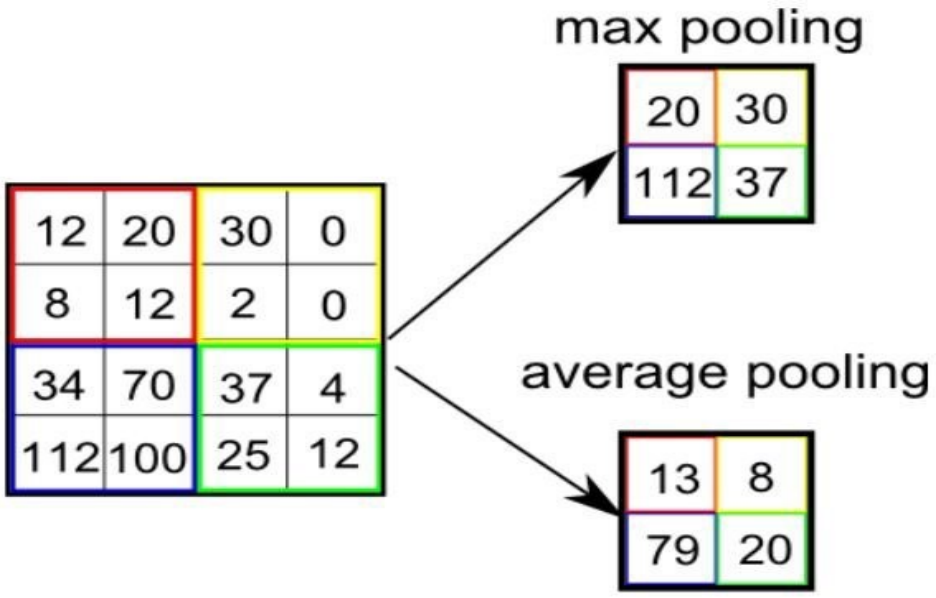
There are two types of pooling:

a. Max Pooling: In max pooling we take a window size [for example window of size 2*2], and only taken the maximum of 4 values.

Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.

b. Average Pooling: In average pooling we take average of all Values in a window.

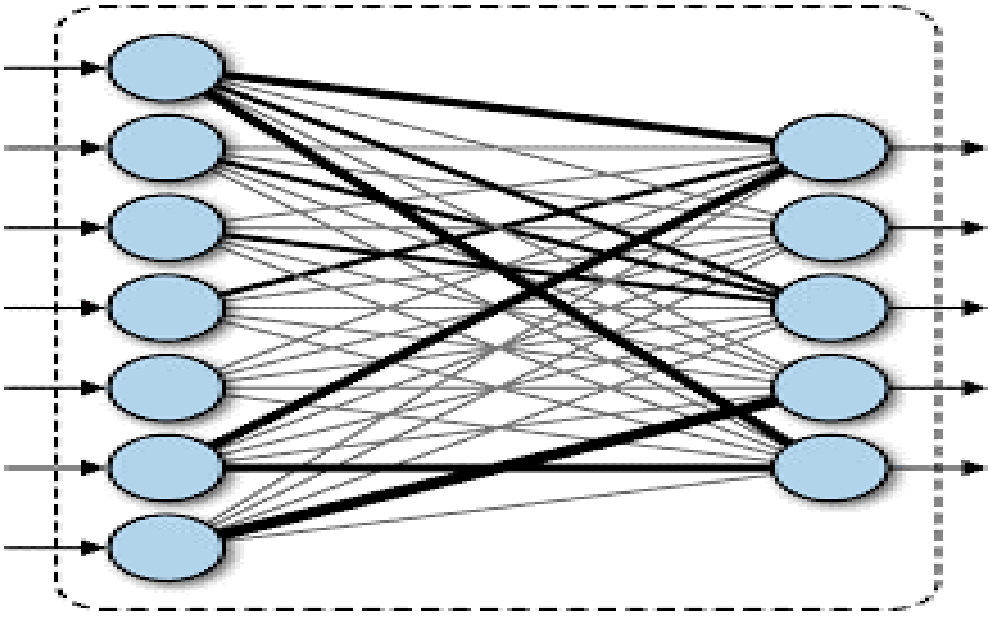
4.2 CNN Pooling



4.2.1: CNN Pooling

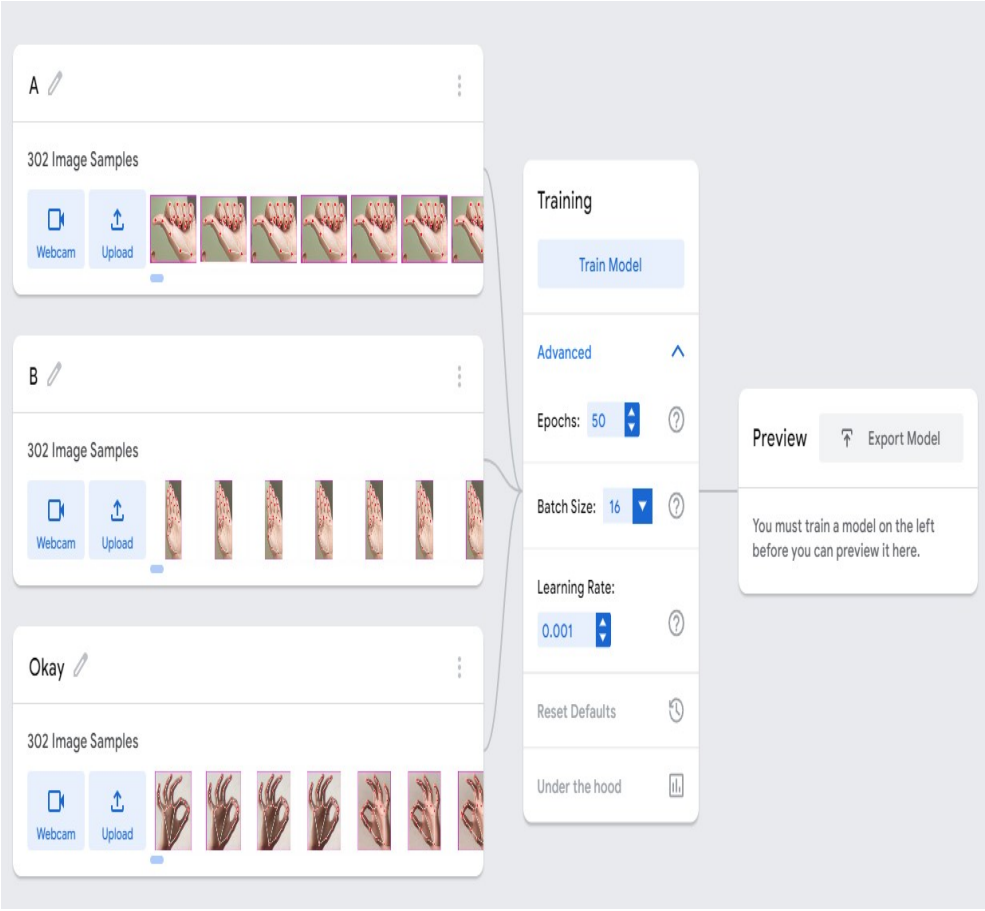
3. Fully Connected Layer: In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

4.3 Final CNN



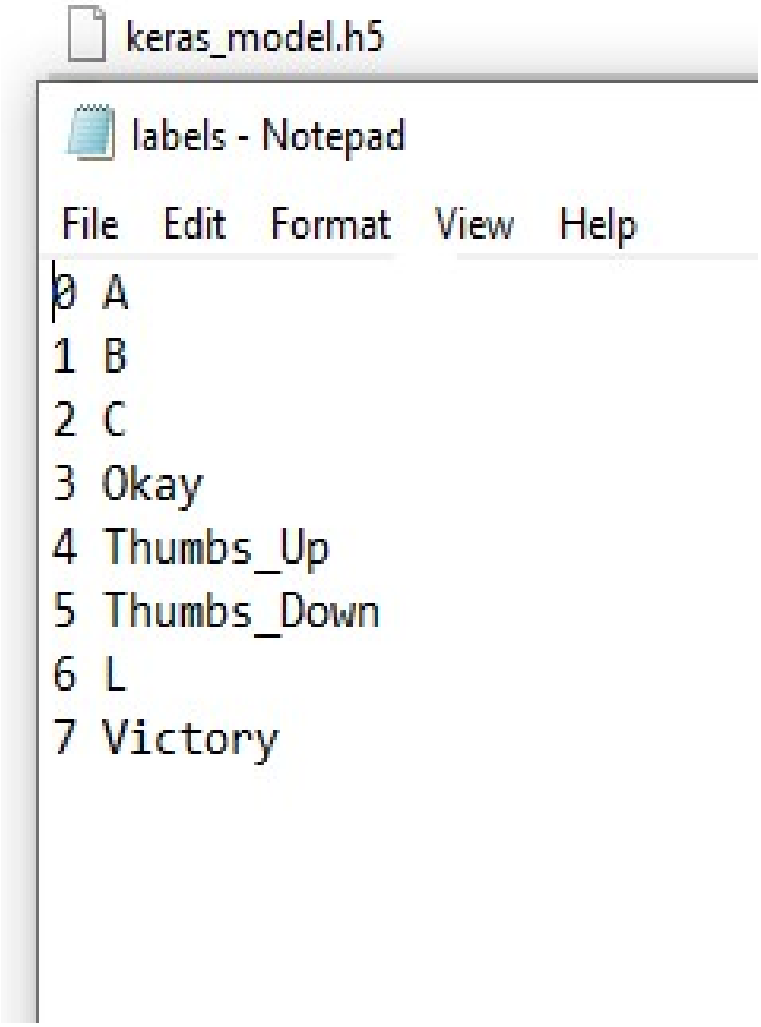
4.3.1: Final CNN

4.4 Google Teachable Machine



4.4.1: Google Teachable Machine

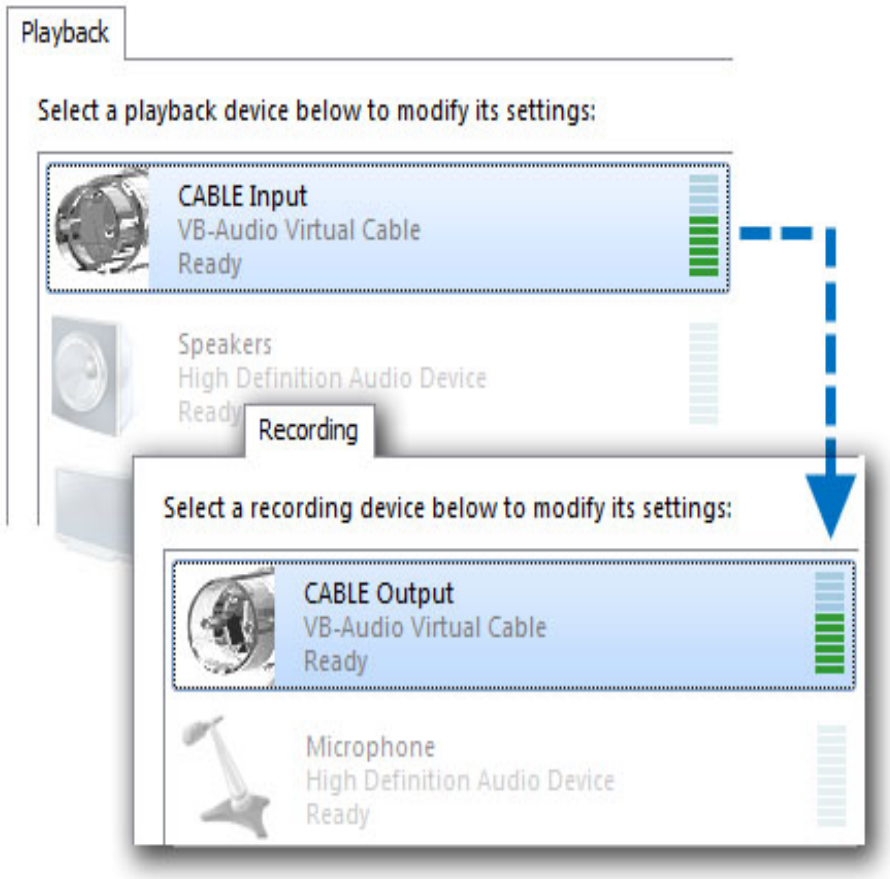
4.5 Keras Model File with Labels



4.5.1: Keras Model File with Labels

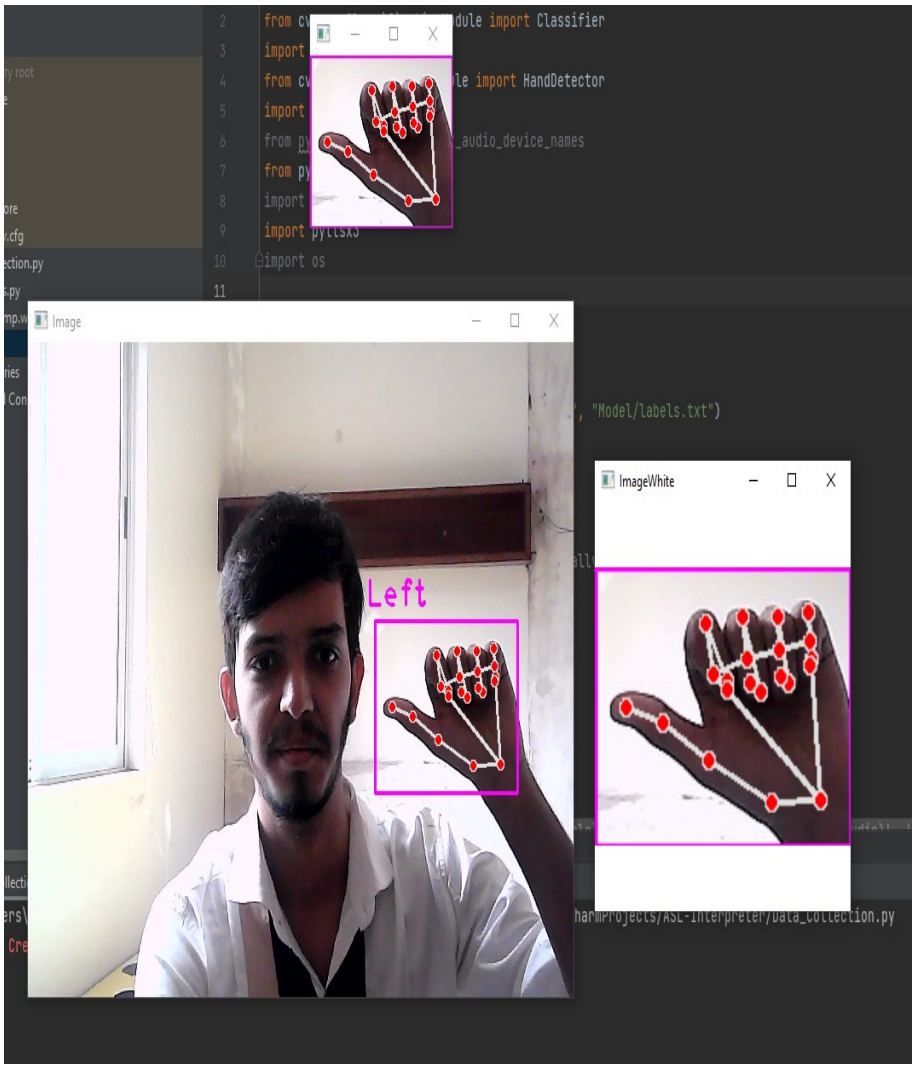
5 Setup

5.1 Install VB Cabel Virtual Audio Drivers



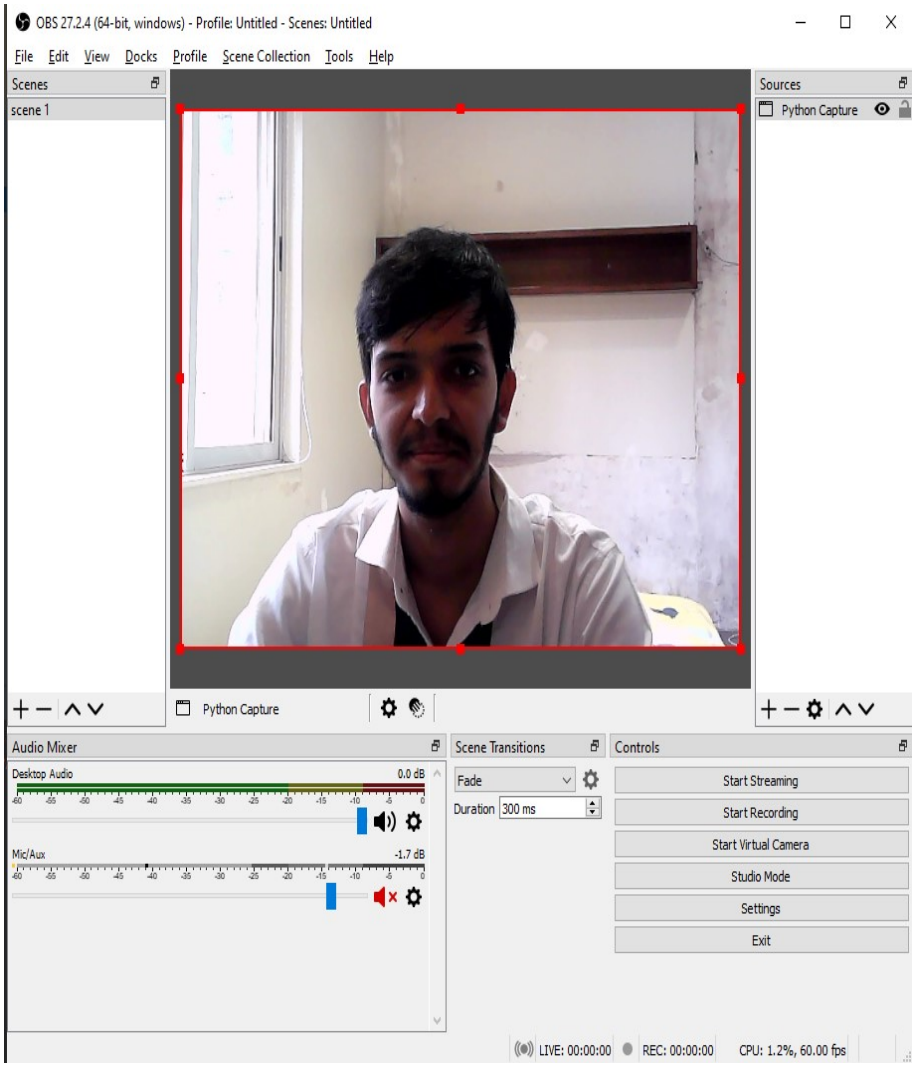
5.1.1: Install VB Cabel Virtual Audio Drivers

5.2 Run test.py file



5.2.1: Run test.py file

5.3 Select Video output as Virtual Cam in OBS



5.3.1: Select Video output as Virtual Cam in OBS

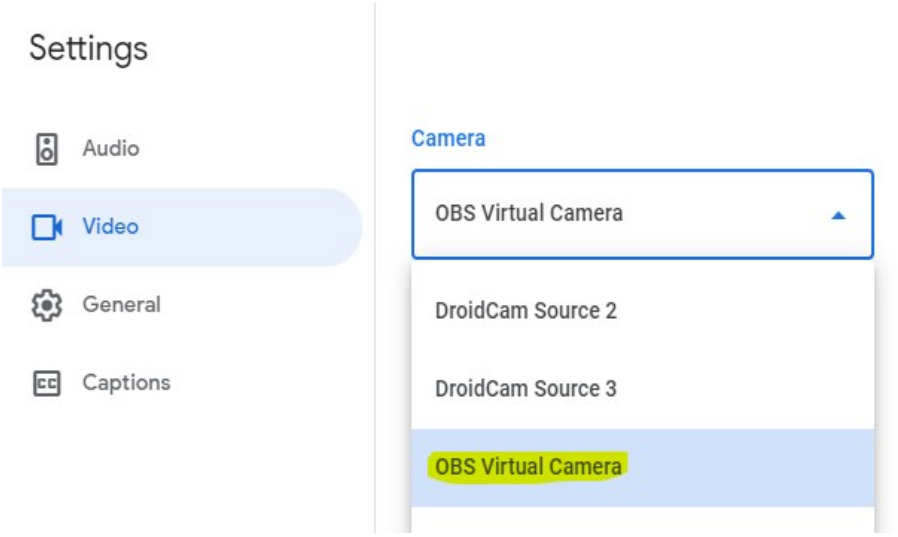
6 Using Application

6.1 Step 1 – Join a Video Conference

We will be using Google Meet Application

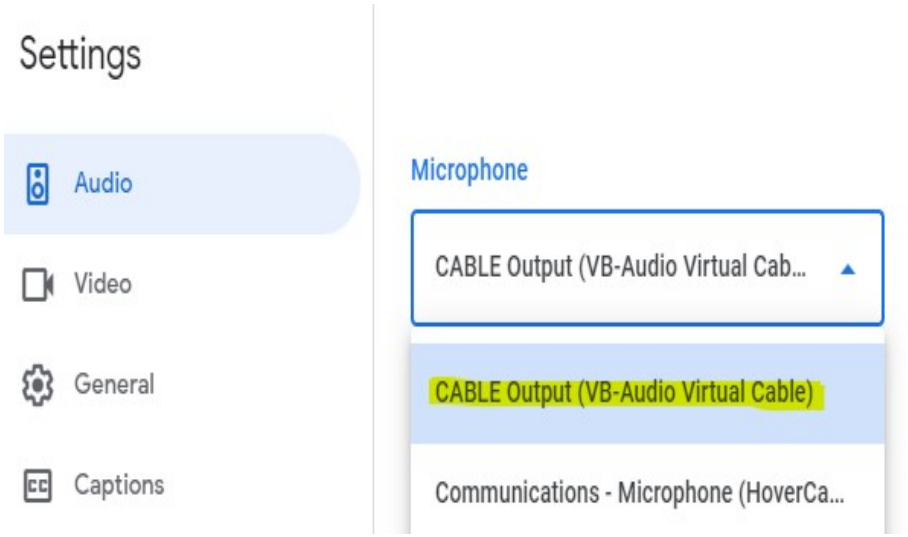
6.2 Step 2 – Run test.py file

6.3 Step 3 – Select OBS Virtual Camera as camera



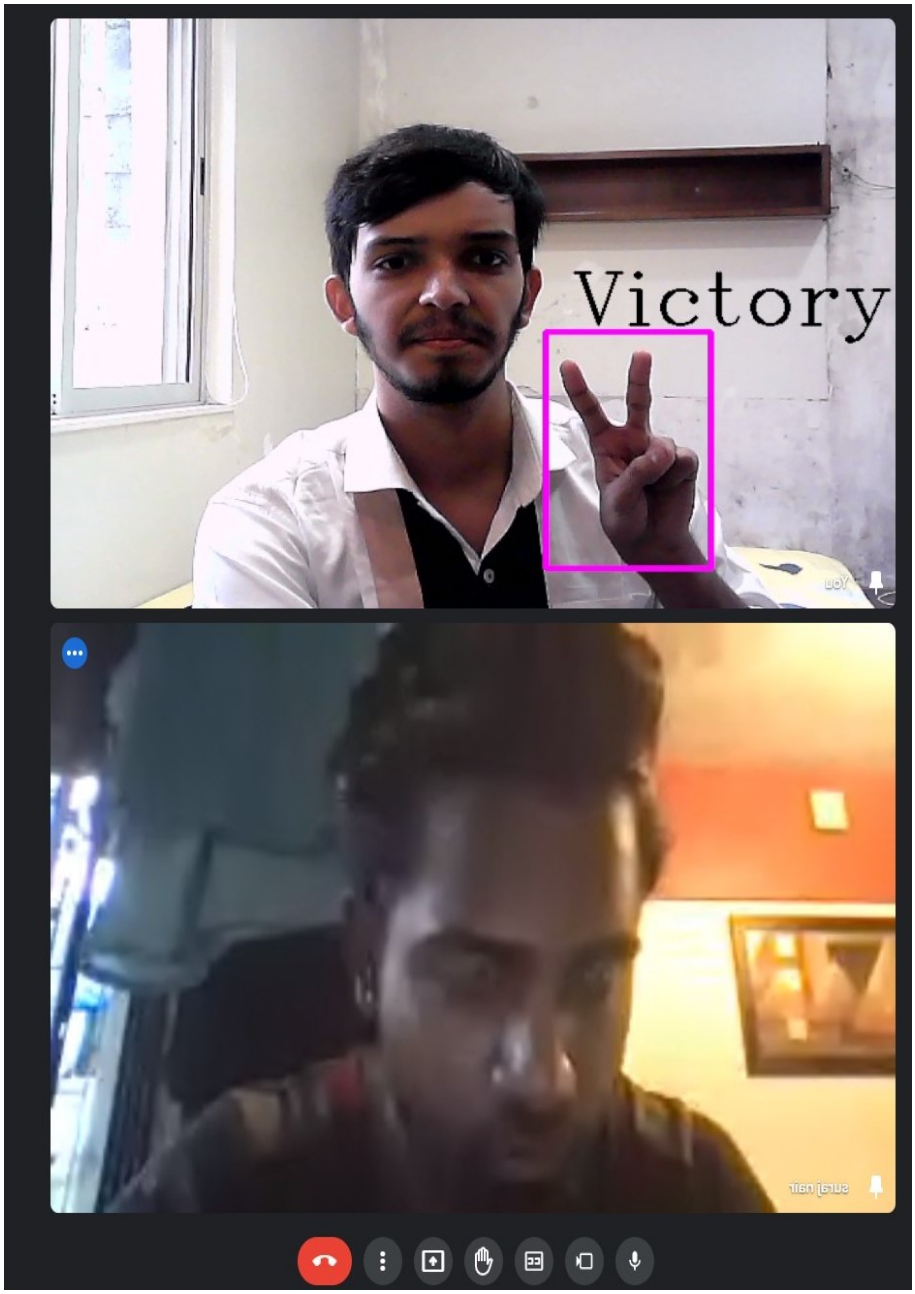
6.3.1: Select OBS Virtual Camera

6.4 Step 4 – Select VB Audio Cabel as microphone

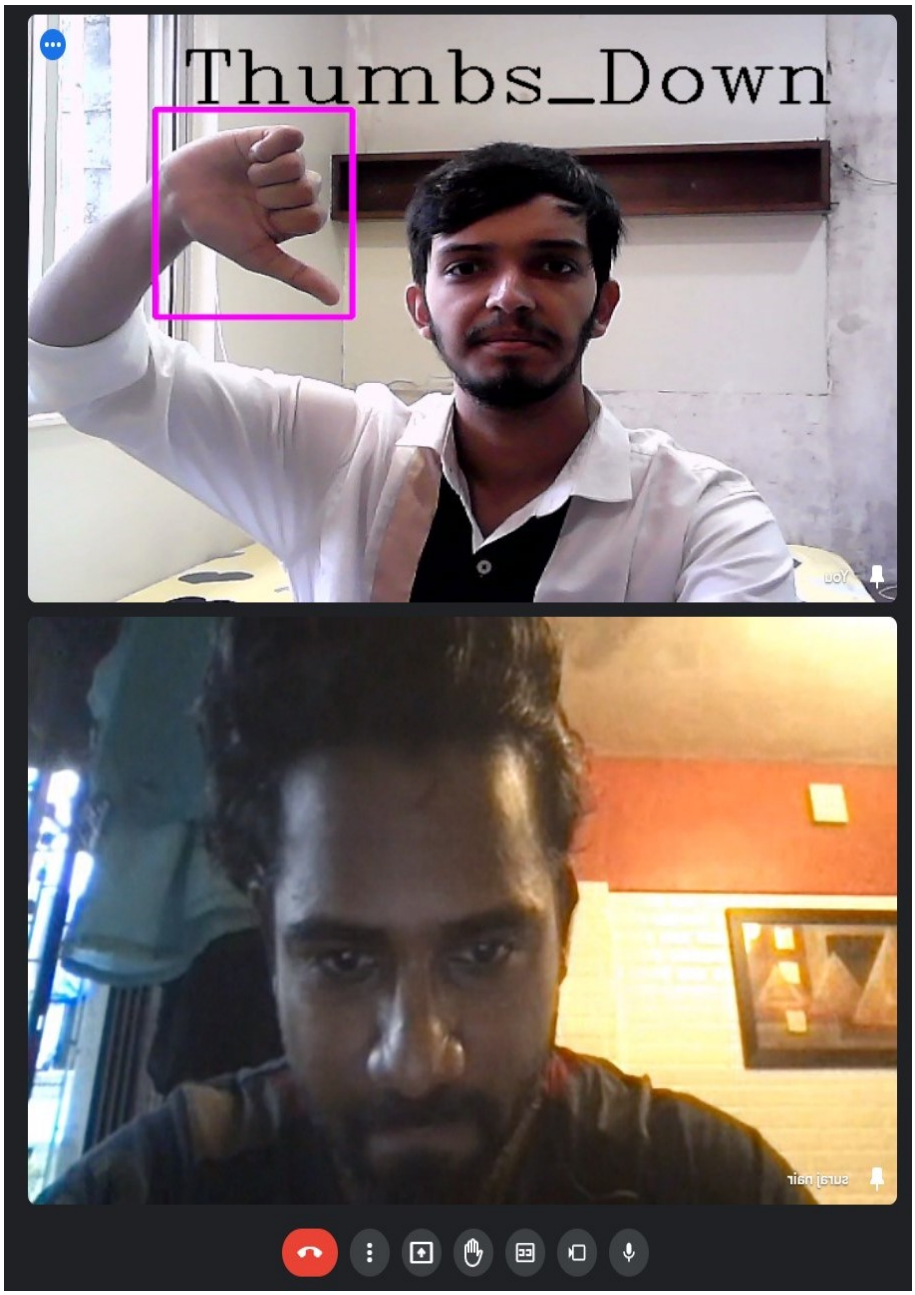


6.4.1: Select OBS Virtual Camera

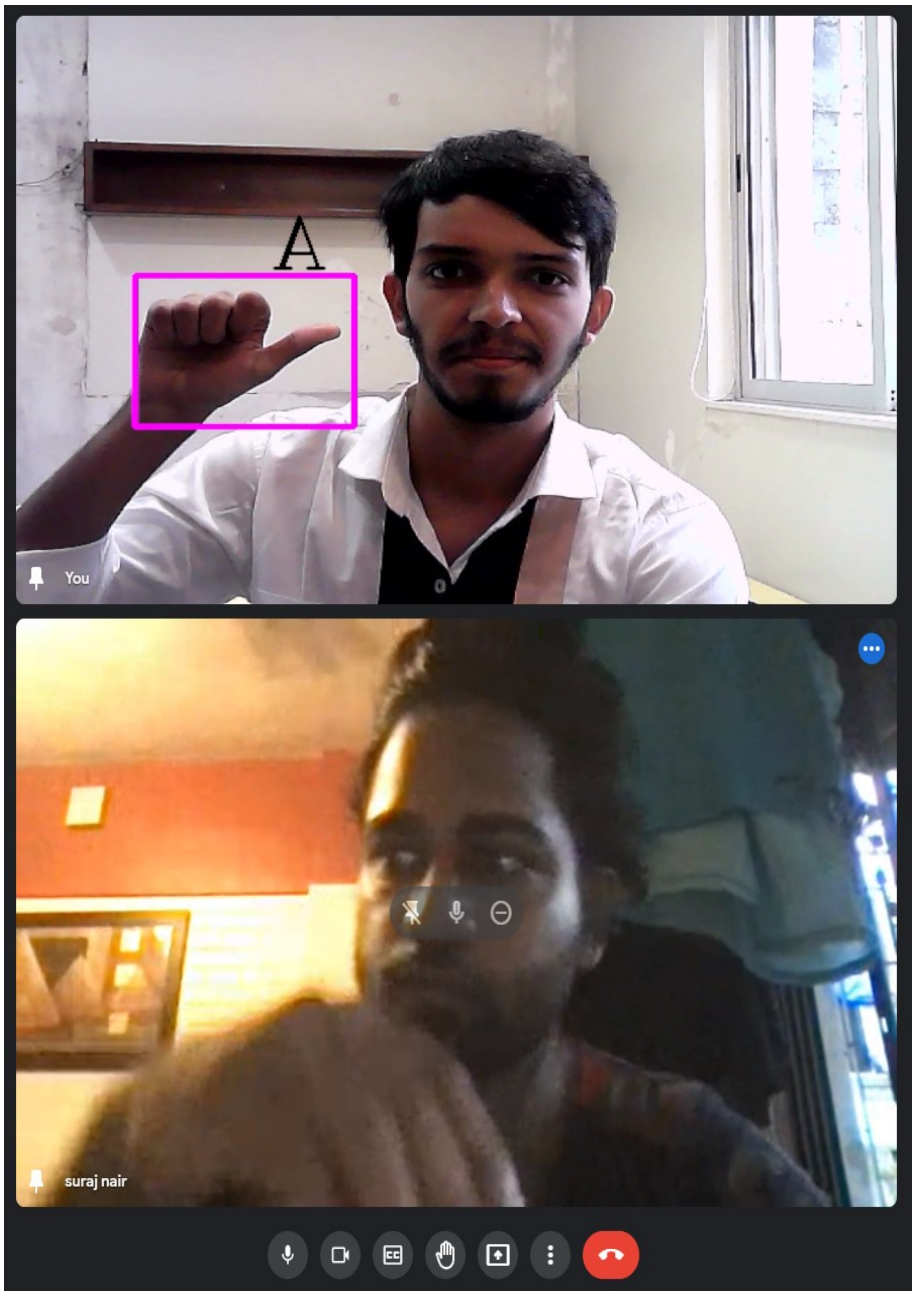
6.5 Step 5 – Participate in Video Meet



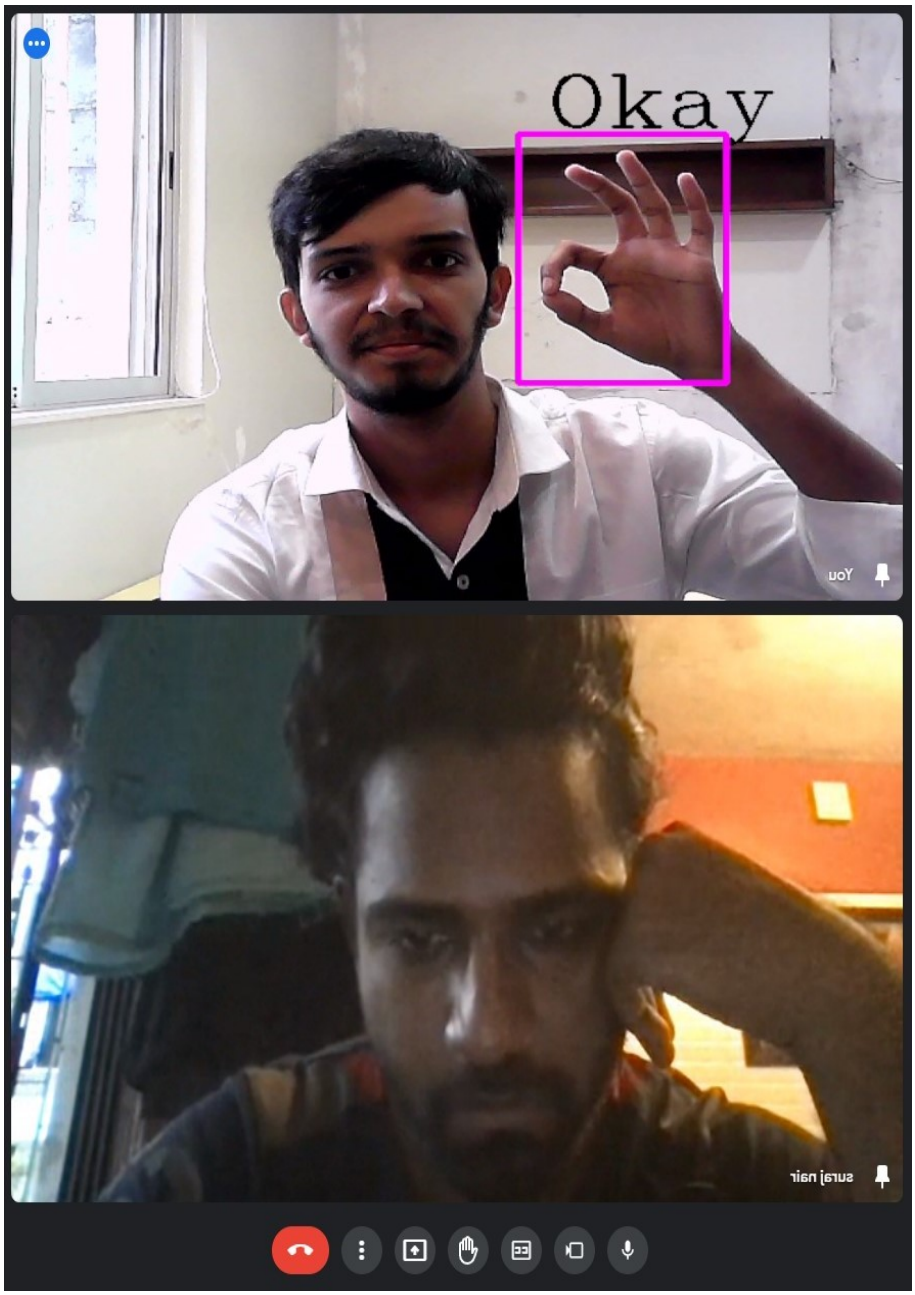
6.5.1: Victory Gesture



6.5.2: Thumbs Up Gesture



6.5.3: Letter A



6.5.4: Okay Gesture

7 Limitations

- Currently this application can be used on desktop only.
- Model is trained with limited dataset.
- For now limited gestures to interpret.
- Needs adequate lightning and simple backdrop.

8 Future Enhancements

- Making this application for mobile phones.
- Add more types of Sign Languages and gestures.
- Continuous training - while the app is being used and user is happy with predictions made, it will account that in predicting next gestures.
- By experimenting with different background removal methods, we hope to obtain improved accuracy even with complicated backdrops.
- We are considering enhancing the preprocessing to more accurately anticipate motions in dimly lit environments.

9 User Manual

Step 1 – Join a Video Conference

Step 2 – Run test.py file

Step 3 – Select OBS Virtual Camera as camera

Step 4 – Select VB Audio Cabel as microphone

10 Bibliography

10.1 References

[1.] https://edu.authorcafe.com/academies/6813/sign-language-recognition
[2.] https://www.irojournals.com/iroiip/V2/I2/01.pdf
[3.] https://github.com/emnikhil/Sign-Language-To-Text-Conversion
[4.] https://github.com/luvk1412/Sign-Language-to-Text/blob/master/Project%20Report.pdf

10.2 Resources

[1.] https://teachablemachine.withgoogle.com/train/image
[2.] https://obsproject.com/
[3.] https://vb-audio.com/Cable/
[4.] https://teachablemachine.withgoogle.com/train/image