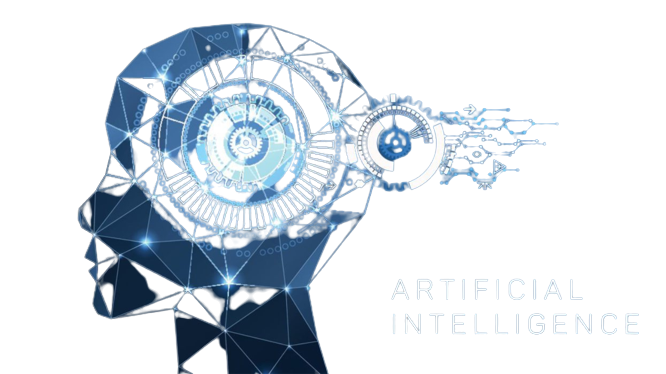
AI sign language detection project



Participates:-

GADDIPATHI MADAN BABU

Student ID:STU6329e1e974f781663689193

Internship ID:INTERNSHIP\_166003801862f22b826becb

Student AICTE Email ID: gaddiparthimadanbabu@gmail.com

AICTE Student ID: STU6329e1e974f781663689193

Index:-

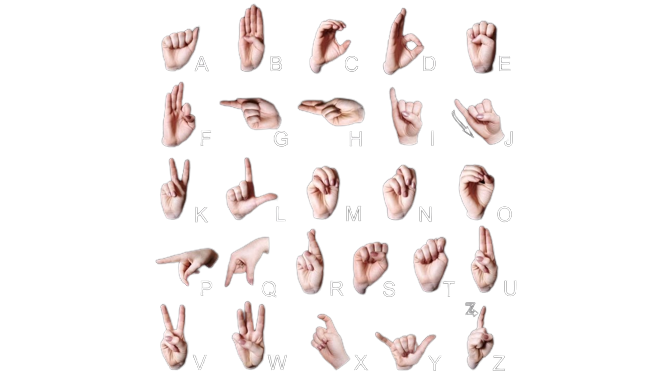
|  |  |
| --- | --- |
| S.no | Subject |
| 1 | Introduction |
| 2 | History |
| 3 | Sign language detection |
| 4 | The requirement libraries |
| 5 | My python program |

Introduction

Sign language (SL) is the main means of communication between hearing-impaired people and other communities and it is expressed through manual (i.e., body and hand motions) and non-manual (i.e., facial expressions) features. These features are combined to form utterances that convey the meaning of words or sentences. Being able to capture and understand the relation between utterances and words is crucial for the Deaf community to guide us to an era where the translation between utterances and words can be achieved automatically. The research community has long identified the need for developing sign language technologies to facilitate the communication and social inclusion of hearing-impaired people. Although the development of such technologies can be challenging due to the existence of numerous sign languages and the lack of large annotated datasets, the recent advances in AI and machine learning have played a significant role in automating and enhancing such technologies.

Sign language translation (SLT) involves the translation between different sign languages, as well as the translation between sign and speaking languages. SLT methods employ sequence-based machine learning algorithms and aim to bridge the communication gap between people signing or speaking different languages. The difficulties in SLT lie in the lack of multilingual sign language datasets, as well as the inaccuracies of SLR methods, considering that gloss recognition (performed by SLR methods) is the initial step of the SLT methods. Finally, sign language representation involves the accurate representation and reproduction of signs using realistic avatars or signed video approaches. Currently, avatar movements are deemed unnatural and hard to understand by the Deaf community due to inaccuracies in skeletal pose capturing and the lack of life-like features in the appearance of avatars.

History

ISL uses both hands similar to British Sign Language and is similar to International Sign Language.

ISL alphabet is derived from British Sign Language and French Sign Language alphabets.

Unlike its American counterpart which uses one hand, uses both hands to represent alphabets.

Sign Language Recognition

Sign language recognition (SLR) is the task of recognizing sign language glosses from video streams. It is a very important research area since it can bridge the communication gap between hearing and Deaf people, facilitating the social inclusion of hearing-impaired people. Moreover, sign language recognition can be classified into isolated and continuous based on whether the video streams contain an isolated gloss or a gloss sequence that corresponds to a sentence.

The requirement libraries:-

Python (3.7.4)

IDE (Jupyter)

Numpy (version 1.16.5)

cv2 (openCV) (version 3.4.2)

Keras (version 2.3.1)

Tensorflow (as Keras uses TensorFlow the in backend and for image preprocessing) (version 2.0.0)

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming

Click the download for the Python in your system

<https://www.python.org/downloads/>

Jupyter

This page briefly introduces the main components of the *Jupyter Notebook* environment. For a more complete overview see [References](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#references).

Click the link to download the Jupiter in your system

[Project Jupyter | Installing Jupyter](https://jupyter.org/install)

Follow the steps then we download the Jupiter to your system

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

When we download the Jupiter, then already NumPy is installed on your computer

Cv2

Cv2 mean OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

Click the link

[OpenCV-python · PyPI](https://pypi.org/project/opencv-python/)

Keras

Keras is a high-level, deep-learning API developed by Google for implementing neural networks. It is written in Python and is used to implement neural networks easily. It also supports multiple backend neural network computations.

Requirements which is used for creating an environment in your device

Firstly we follow the steps

1. Creating the dataset
2. Training a CNN on the captured dataset
3. Predicting the data

Creating the Dataset



With the help this code

import cv2

import uuid

import os

import time

IMAGES\_PATH = 'Tensorflow/workspace/images/collectedimages'

labels = ['hello', 'thanks', 'yes', 'no' ,'I\_love\_you']

number\_imgs = 10

for label in labels:

!mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}

cap = cv2.VideoCapture(0)

print('Collecting images for {}'.format(label))

time.sleep(5)

for imgnum in range(number\_imgs):

ret, frame = cap.read()

imgname = os.path.join(IMAGES\_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))

cv2.imwrite(imgname,frame)

cv2.imshow('frame',frame)

time.sleep(2)

if cv2.waitKey(1) and 0xFF == ord('q'):

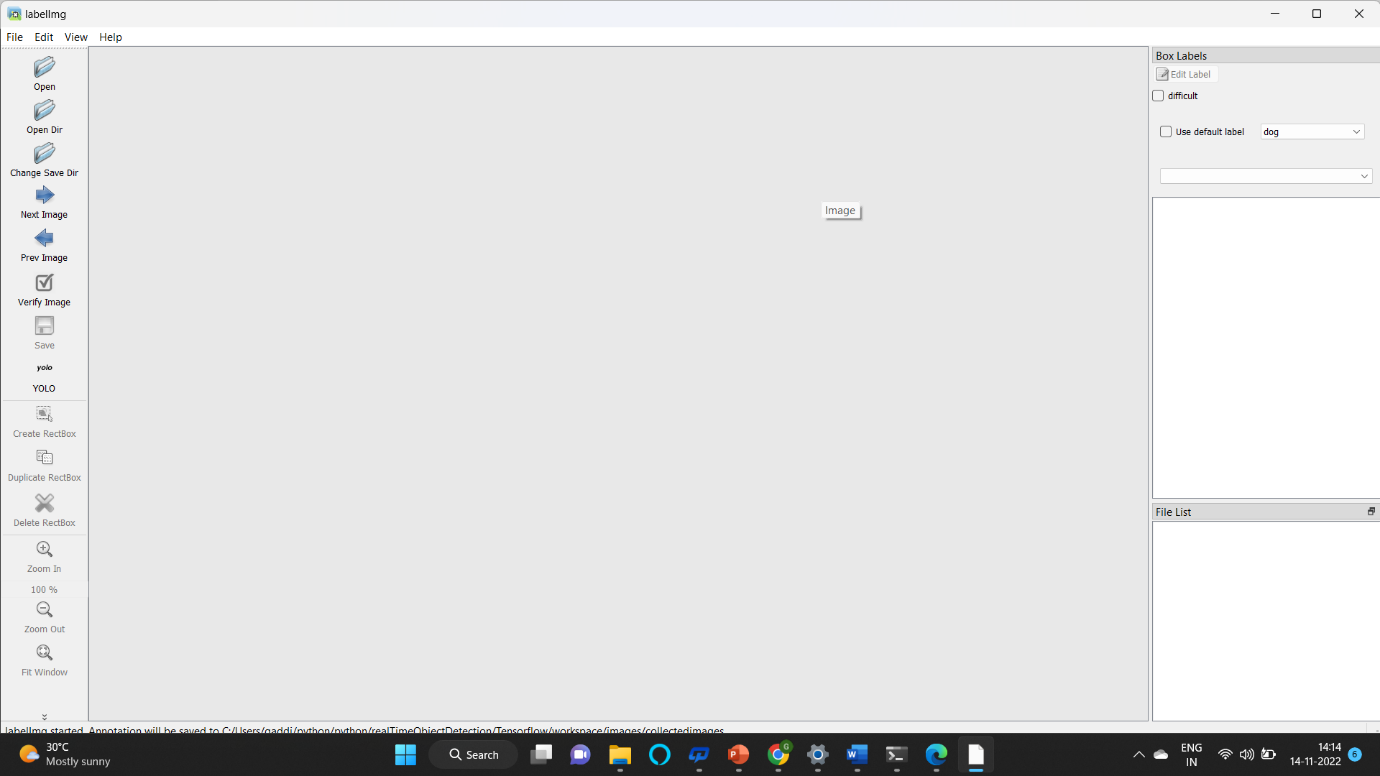
break

cap.release()

Labelling to the given images

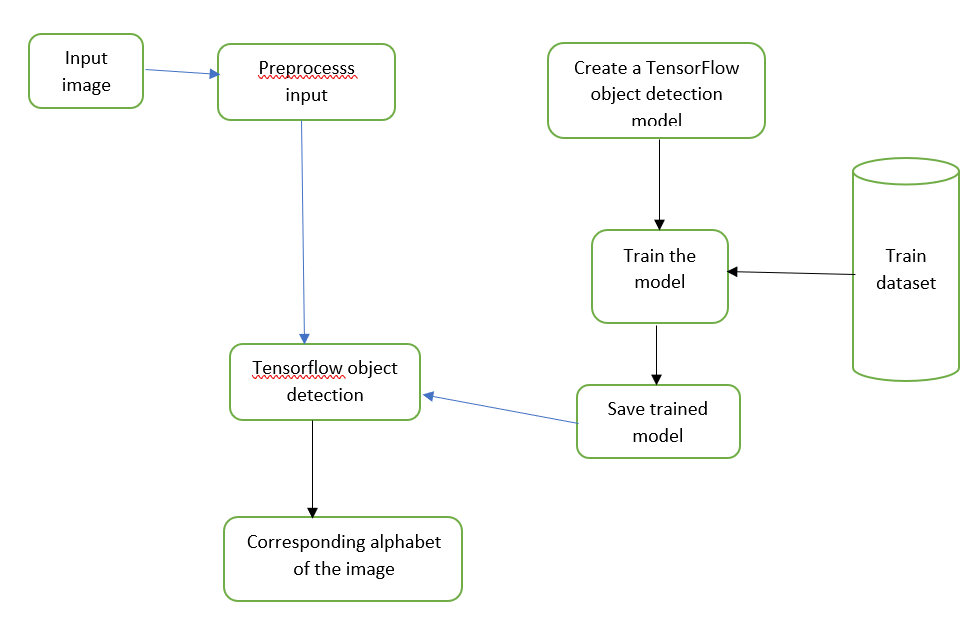
Open command prompt run python labellimg.py

Then we got



We got this page

In this page, we do the labelling to the data images

Flow chart of the Project 

My python program

WORKSPACE\_PATH = 'Tensorflow/workspace'

SCRIPTS\_PATH = 'Tensorflow/scripts'

APIMODEL\_PATH = 'Tensorflow/models'

ANNOTATION\_PATH = WORKSPACE\_PATH+'/annotations'

IMAGE\_PATH = WORKSPACE\_PATH+'/images'

MODEL\_PATH = WORKSPACE\_PATH+'/models'

PRETRAINED\_MODEL\_PATH = WORKSPACE\_PATH+'/pre-trained-models'

CONFIG\_PATH = MODEL\_PATH+'/my\_ssd\_mobnet/pipeline.config'

CHECKPOINT\_PATH = MODEL\_PATH+'/my\_ssd\_mobnet/'

labels = [{'name':'hello', 'id':2}, {'name':'yes', 'id':1},{'name': 'no','id':2},{'name':'thanks','id':5},{'name':'I\_love \_you','id':4}]

with open(ANNOTATION\_PATH + '\label\_map.txt', 'w') as f:

for label in labels:

f.write('item { \n')

f.write('\tname:\'{}\'\n'.format(label['name']))

f.write('\tid:{}\n'.format(label['id']))

f.write('}\n')

!python {SCRIPTS\_PATH + '/generate\_tfrecord.py'} -x {IMAGE\_PATH + '/train'} -l {ANNOTATION\_PATH + '/label\_map.pbtxt'} -o {ANNOTATION\_PATH + '/train.record'}

!python {SCRIPTS\_PATH + '/generate\_tfrecord.py'} -x{IMAGE\_PATH + '/test'} -l {ANNOTATION\_PATH + '/label\_map.pbtxt'} -o {ANNOTATION\_PATH + '/test.record'}

!cd Tensorflow && git clone <https://github.com/tensorflow/models>

CUSTOM\_MODEL\_NAME = 'my\_ssd\_mobnet'

!mkdir {'Tensorflow\workspace\models\\'+CUSTOM\_MODEL\_NAME}

!cp {PRETRAINED\_MODEL\_PATH+'/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/pipeline.config'} {MODEL\_PATH+'/'+CUSTOM\_MODEL\_NAME}

import tensorflow as tf

from object\_detection.utils import config\_util

from object\_detection.protos import pipeline\_pb2

from google.protobuf import text\_format

CONFIG\_PATH = MODEL\_PATH+'/'+CUSTOM\_MODEL\_NAME+'/pipeline.config'

config = config\_util.get\_configs\_from\_pipeline\_file(CONFIG\_PATH)

config

pipeline\_config = pipeline\_pb2.TrainEvalPipelineConfig()

with tf.io.gfile.GFile(CONFIG\_PATH, "r") as f:

proto\_str = f.read()

text\_format.Merge(proto\_str, pipeline\_config)

pipeline\_config.model.ssd.num\_classes = 2

pipeline\_config.train\_config.batch\_size = 4

pipeline\_config.train\_config.fine\_tune\_checkpoint = PRETRAINED\_MODEL\_PATH+'/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/checkpoint/ckpt-0'

pipeline\_config.train\_config.fine\_tune\_checkpoint\_type = "detection"

pipeline\_config.train\_input\_reader.label\_map\_path= ANNOTATION\_PATH + '/label\_map.pbtxt'

pipeline\_config.train\_input\_reader.tf\_record\_input\_reader.input\_path[:] = [ANNOTATION\_PATH + '/train.record']

pipeline\_config.eval\_input\_reader[0].label\_map\_path = ANNOTATION\_PATH + '/label\_map.pbtxt'

pipeline\_config.eval\_input\_reader[0].tf\_record\_input\_reader.input\_path[:] = [ANNOTATION\_PATH + '/test.record']

config\_text = text\_format.MessageToString(pipeline\_config)

with tf.io.gfile.GFile(CONFIG\_PATH, "wb") as f:

f.write(config\_text)

AUTO = tf.data.experimental.AUTOTUNE # used in tf.data.Dataset API

option\_no\_order = tf.data.Options()

option\_no\_order.experimental\_deterministic = False

option\_no\_order.experimental\_optimization.noop\_elimination = True

option\_no\_order.experimental\_optimization.apply\_default\_optimizations = True

print("""python {}/research/object\_detection/model\_main\_tf2.py --model\_dir={}/{} --pipeline\_config\_path={}/{}/pipeline.config --num\_train\_steps=10000""".format(APIMODEL\_PATH, MODEL\_PATH,CUSTOM\_MODEL\_NAME,MODEL\_PATH,CUSTOM\_MODEL\_NAME))

import os

from object\_detection.utils import label\_map\_util

from object\_detection.utils import visualization\_utils as viz\_utils

from object\_detection.builders import model\_builder

# Load pipeline config and build a detection model

configs = config\_util.get\_configs\_from\_pipeline\_file(CONFIG\_PATH)

detection\_model = model\_builder.build(model\_config=configs['model'], is\_training=False)

# Restore checkpoint

ckpt = tf.compat.v2.train.Checkpoint(model=detection\_model)

ckpt.restore(os.path.join(CHECKPOINT\_PATH, 'ckpt-6')).expect\_partial()

@tf.function

def detect\_fn(image):

image, shapes = detection\_model.preprocess(image)

prediction\_dict = detection\_model.predict(image, shapes)

detections = detection\_model.postprocess(prediction\_dict, shapes)

return detections

import cv2

import numpy as np

category\_index = label\_map\_util.create\_category\_index\_from\_labelmap(ANNOTATION\_PATH+'/label\_map.pbtxt')

# Setup capture

cap = cv2.VideoCapture(0)

width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

while True:

ret, frame = cap.read()

image\_np = np.array(frame)

input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)

detections = detect\_fn(input\_tensor)

num\_detections = int(detections.pop('num\_detections'))

detections = {key: value[0, :num\_detections].numpy()

for key, value in detections.items()}

detections['num\_detections'] = num\_detections

# detection\_classes should be ints.

detections['detection\_classes'] = detections['detection\_classes'].astype(np.int64)

label\_id\_offset = 1

image\_np\_with\_detections = image\_np.copy()

viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

image\_np\_with\_detections,

detections['detection\_boxes'],

detections['detection\_classes']+label\_id\_offset,

detections['detection\_scores'],

category\_index,

use\_normalized\_coordinates=True,

max\_boxes\_to\_draw=5,

min\_score\_thresh=.5,

agnostic\_mode=False)

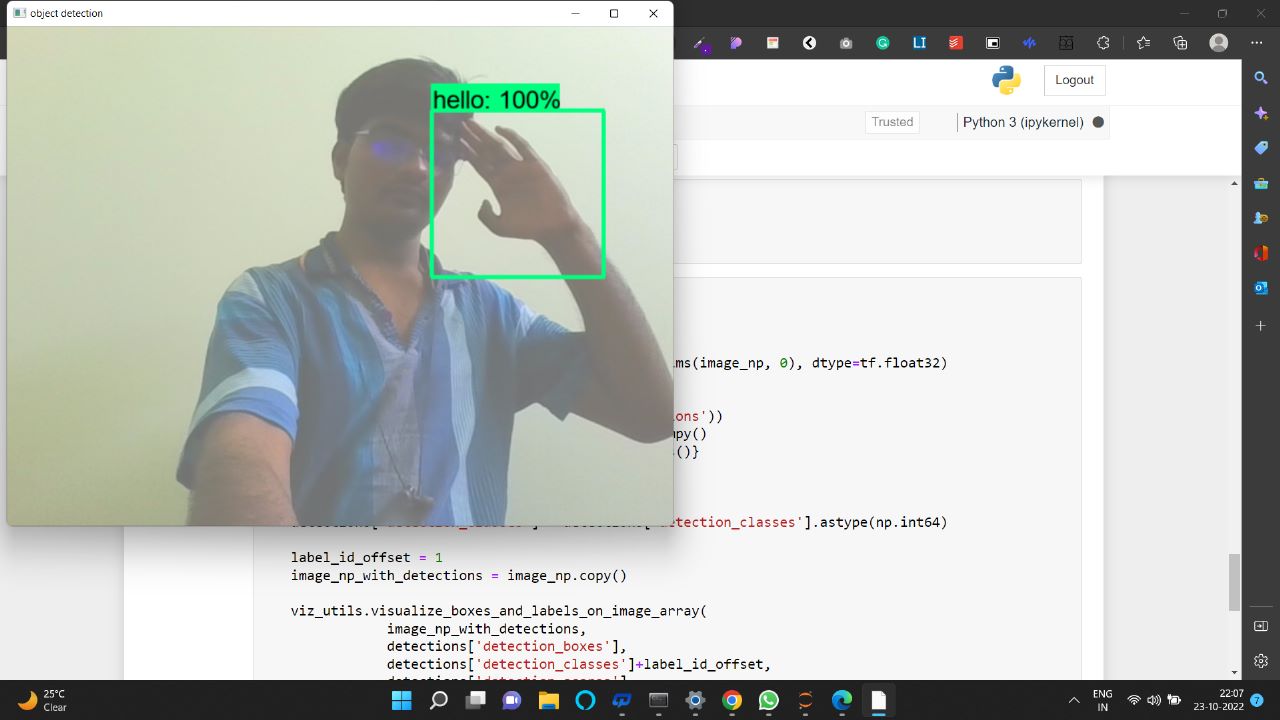
cv2.imshow('object detection', cv2.resize(image\_np\_with\_detections, (800, 600)))

if cv2.waitKey(1) & 0xFF == ord('q'):

cap.release()

break

Result:-



I am also using another approach i.e, using a Techable machine.

In this approach, we got the directly train our code.

The python code

import cv2

from cvzone.HandTrackingModule import HandDetector

import numpy as np

import math

import time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)

offset = 20

imgSize = 300

folder = "data/Hello"

counter = 0

while True:

    success, img = cap.read()

    hands, img = detector.findHands(img)

    if hands:

        hand = hands[0]

        x,y,w,h = hand['bbox']

        imgWhite = np.ones((imgSize,imgSize,3),np.uint8)\*255

        imgCrop = img[y-offset : y+ h+offset,x-offset : x+w+offset]

        imgCropShape = imgCrop.shape

        Ratio  = h/w

        if Ratio >1:

            k = imgSize/h

            wCal  = math.ceil(k\*w)

            imgResize = cv2.resize(imgCrop,(wCal, imgSize))

            imgResizeShape = imgResize.shape

            Wgap = math.ceil((imgSize-wCal)/2)

            imgWhite[:,Wgap:wCal+Wgap] = imgResize

        else:

            k = imgSize/w

            hCal  = math.ceil(k\*h)

            imgResize = cv2.resize(imgCrop,(imgSize, hCal))

            imgResizeShape = imgResize.shape

            hgap = math.ceil((imgSize-hCal)/2)

            imgWhite[hgap:hCal+hgap,:] = imgResize

        #cv2.imshow('ImageCrop', imgCrop)

        cv2.imshow('ImageWhite',imgWhite)

    cv2.imshow("Image", img)

    key = cv2.waitKey(1)

    if key == ord('s'):

        counter += 1

        cv2.imwrite(f'{folder}/Image\_{time.time()}.png',imgWhite)

        print(counter)

this code is used for data collection

then we train the data images in the teachable machine. After the copy, the Keras file

locates in the model folder

And then we test this code

import cv2

from cvzone.HandTrackingModule import HandDetector

from cvzone.ClassificationModule import Classifier

import numpy as np

import math

import time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)

classifier = Classifier("Model/keras\_model.h5","Model/labels.txt")

offset = 20

imgSize = 300

folder = "data/C"

counter = 0

labels = ['1','2','3','4','5','6','7','8','9','0','Hello','Yes','No']

while True:

    success, img = cap.read()

    imgOutput = img.copy()

    hands, img = detector.findHands(img)

    if hands:

        hand = hands[0]

        x,y,w,h = hand['bbox']

        imgWhite = np.ones((imgSize,imgSize,3),np.uint8)\*255

        imgCrop = img[y-offset : y+ h+offset,x-offset : x+w+offset]

        imgCropShape = imgCrop.shape

        Ratio  = h/w

        if Ratio >1:

            k = imgSize/h

            wCal  = math.ceil(k\*w)

            imgResize = cv2.resize(imgCrop,(wCal, imgSize))

            imgResizeShape = imgResize.shape

            Wgap = math.ceil((imgSize-wCal)/2)

            imgWhite[:,Wgap:wCal+Wgap] = imgResize

            perdiction, index = classifier.getPrediction(imgWhite,draw=False)

            print(perdiction,index)

        else:

            k = imgSize/w

            hCal  = math.ceil(k\*h)

            imgResize = cv2.resize(imgCrop,(imgSize, hCal))

            imgResizeShape = imgResize.shape

            hgap = math.ceil((imgSize-hCal)/2)

            imgWhite[hgap:hCal+hgap,:] = imgResize

            perdiction, index = classifier.getPrediction(imgWhite,draw=False)

        cv2.rectangle(imgOutput,(x-offset,y-offset-50),(x-offset+90,y-offset-50+50),(255,0,255),cv2.FILLED)

        cv2.putText(imgOutput,labels[index],(x,y-20),cv2.FONT\_HERSHEY\_COMPLEX,2,(255,255,255),2)

        cv2.rectangle(imgOutput,(x-offset,y-offset),(x+w+offset,y+h+offset),(255,0,255),2)

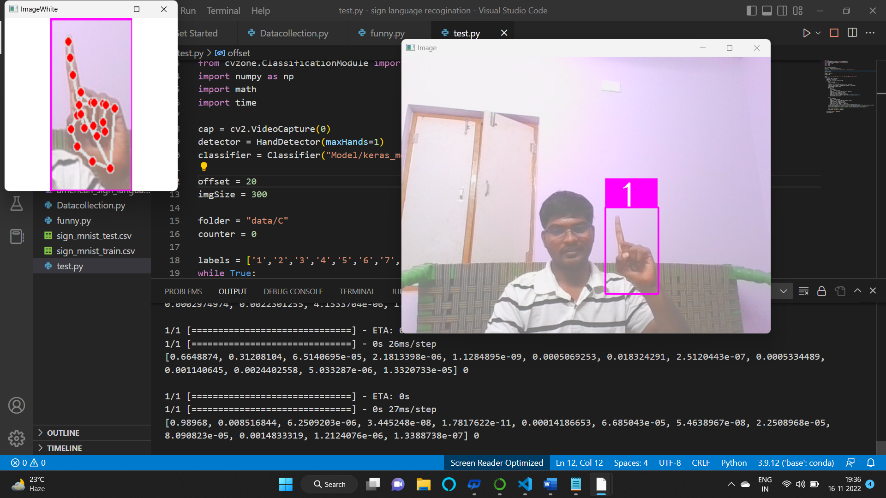
        #cv2.imshow('ImageCrop', imgCrop)

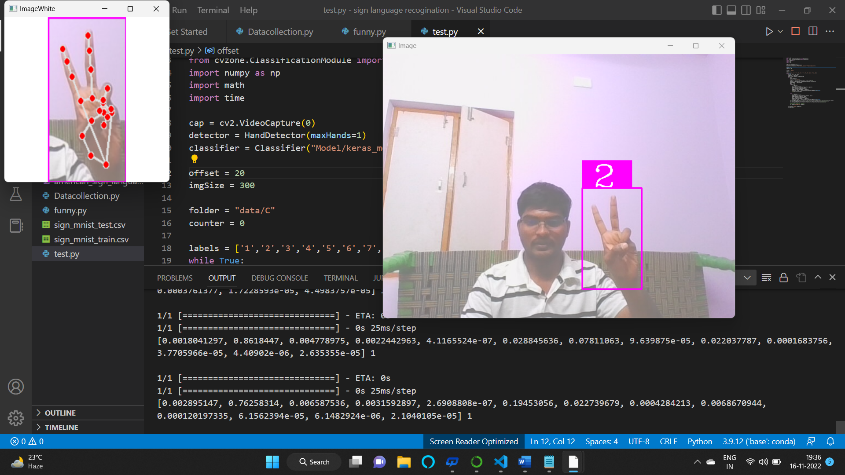
        cv2.imshow('ImageWhite',imgWhite)

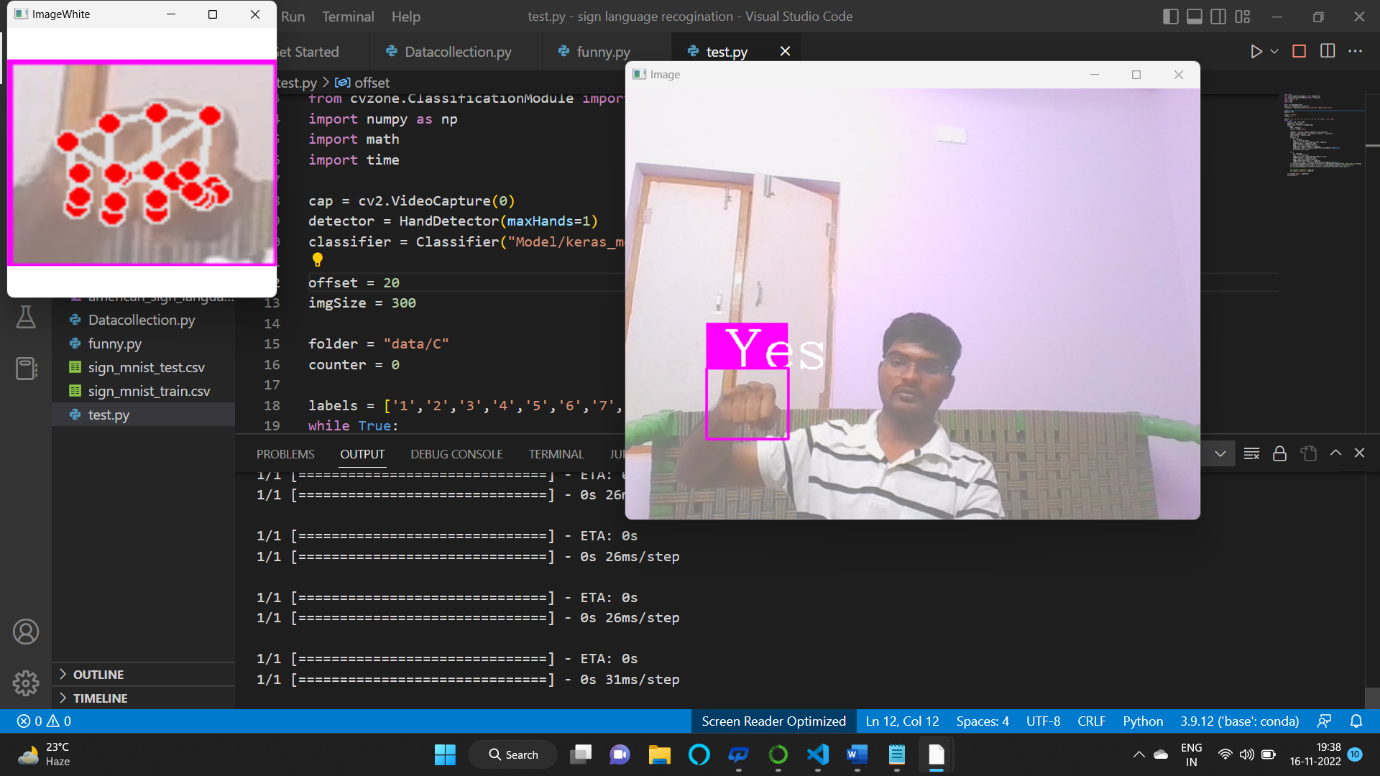
    cv2.imshow("Image", imgOutput)

    cv2.waitKey(1)

github link:- [Madan1314114/AI-sign (github.com)](https://github.com/Madan1314114/AI-sign)

Result:-





Conclusion:-

In this project, we proposed an idea for feasible communication between the hearing impaired and normal people with the help of deep learning an

machine learning approach

Thank you