AI PROJECT: SIGN LANGUAGE INTERPRETER

Submitted to: Ms Navpreet

Submitted by: 102003379, 102003387,102103389

PROBLEM STATEMENT:

To interpret the ASL Sign Language(0-9).

DESCRIPTION OF PROBLEM:

Sign Language is a form of communication used primarily by people hard of hearing or deaf. This type of gesture-based language allows people to convey ideas and thoughts easily overcoming the barriers caused by difficulties from hearing issues.

A major issue with this convenient form of communication is the lack of knowledge of the language for the vast majority of the global population. Just as with any other language, learning Sign Language takes much time and effort, discouraging it from being learned by the larger population. However, an evident solution to this issue is present in the world of Machine Learning and Image Detection. Implementing predictive model technology to automatically classify Sign Language symbols can be used to create a form of real-time captioning for virtual conferences like Zoom meetings and other such things. This would greatly increase access of such services to those with hearing impairments as it would go hand-in-hand with voice-based captioning, creating a two-way communication system online for people with hearing issues**.**

WORKING OF PROJECT AND ALGORITHM:

In this sign language recognition system, a sign detector detects numbers, which can be easily extended to cover a wide range of other signs and hand signs, including the alphabet.
In this system, we are using a machine learning model known as CNN. Convolutional Neural Networks (CNNs) can learn complicated objects and patterns because they have an input layer, an output layer, numerous hidden layers, and millions of parameters.
By turning on the camera, the user can perform the hand signs, and the system will detect the sign and display it to the user. Using hand signs, the individual may send out more information in a shorter amount of time.

Steps to develop sign language project

This is divided into 3 parts:

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

This project has been made on PyCharm using Python Interpretor version 3.9.

To make the program work, we installed the following packages:

1	Opencv-python	ver 4.5.4.60
2	cvzone	ver 1.5.6
3	Mediapipe	ver 0.9.2.1
4	Keras	ver 2.12.0
5	tensorflow	ver 2.12.0

IMPORTING USED MODULES

Brief description of the imported modules:

1. cv2: OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2.

- **VideoCapture()** to get a video capture object for the camera.
- **imshow()** method is used to display an image in a window. The window automatically Its to the image size.

2. Handtracking: Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand's movement and orientation.

3. NumPy: NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays

4. Math: Python has a built-in module that you can use for mathematical tasks.

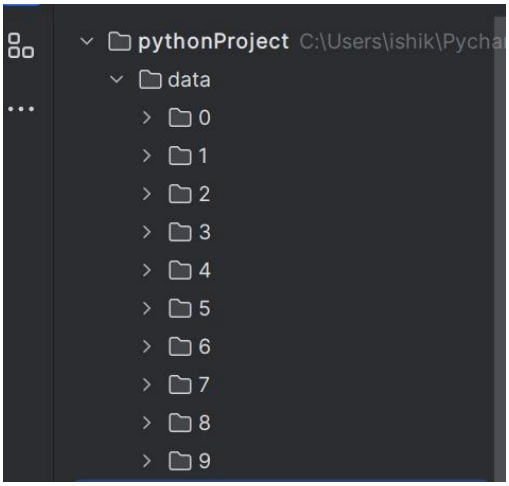
5. Time: Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc.

What is epoch?
The epoch is the point where the time starts and is platform-dependent. On Windows and most Unix systems, the epoch is January 1, 1970, 00:00:00 (UTC), and leap seconds are not counted towards the time in seconds since the epoch. To check what the epoch is on a given platform we can use time.gmtime(0).

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
```

CODE AND EXPLANATION

Data Folder contains the captured image dataset of ASL 0 to 9
A directory structure can be visualised as in the following snapshot:



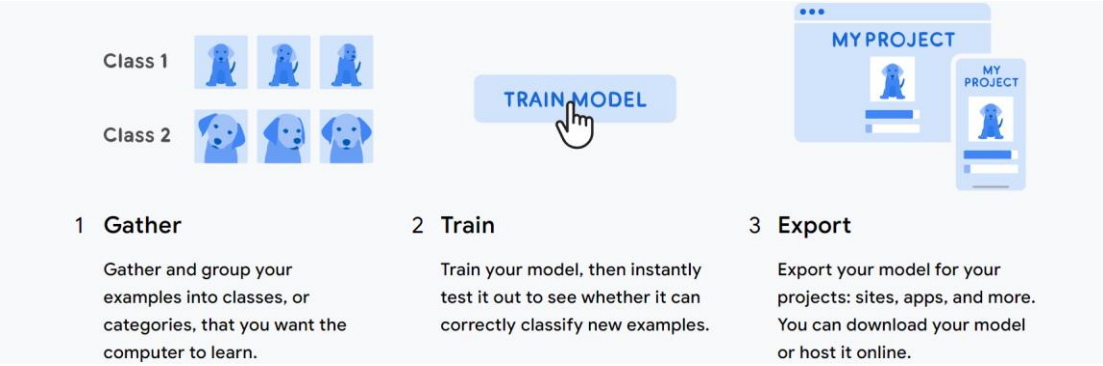
Here we start capturing the image to form a training dataset. At a particular time, only one hand can be present in the frame. The images for each of number is stored in the respective folder in data folder.

```
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)

offset = 20
imgSize = 300

folder = "Data/9"
counter = 0
```

We maintain an array labels in test.py where we store all the digits we can identify.

When we have all the images stored in data folder , we upload these images to Google Teachable Machine and train to create a tensorflow model out of the images:

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.



After training we get a keras model which is stored with an .h5 extension.

test.py

```
offset = 20
imgSize = 300

folder = "Data"
counter = 0

labels = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
```

Here we are creating the frame around the image (our hand) which is being detected by the program.

Cap.read(): This code initiates an infinite loop (to be broken later by a break statement), where we have ret and frame being defined as the cap. read(). Basically, ret is a boolean regarding whether or not there was a return at all, at the frame is each frame that is returned.

np.ones: Python numpy. ones() function returns a new array of given shape and data type, where the element's value is set to 1. This function is very similar to numpy zeros() function.

Image.copy: Image.copy() method copies the image to another image object, this method is useful when we need to copy the image but also retain the original.

```
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

        aspectRatio = h / w
```

cv2.resize: The cv2 resize () function is specifically used to resize images using different interpolation techniques.

classifier.getPrediction: This makes the predictions for each of the meshgrid points (which is then used to make a mask over the plot to show which regions are classified as what label by the classifier).

```
        if aspectRatio > 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
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, 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
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
```

cv2.rectangle: rectangle() method is used to draw a rectangle on any image. Syntax: cv2.rectangle(image, start_point, end_point, color, thickness).

Cv2.putText(): cv2.putText() method is used to draw a text string on any image.

Syntax: cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]])

Parameters:

image: It is the image on which text is to be drawn.

text: Text string to be drawn.

org: It is the coordinates of the bottom-left corner of the text string in the image. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

font: It denotes the font type. Some of font types are FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN, , etc.

fontScale: Font scale factor that is multiplied by the font-specific base size.

color: It is the color of text string to be drawn. For BGR, we pass a tuple. eg: (255, 0, 0) for blue color.

Return Value: It returns an image.

```
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 - 26), cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255),
cv2.rectangle(imgOutput, (x-offset, y-offset),
              (x + w+offset, y + h+offset), (255, 0, 255), 4)
```

Cv2.imshow(): cv2.imshow() method is used to display an image in a window. The window automatically fits the image size.

Syntax: cv2.imshow(window_name, image)

Parameters:

window_name: A string representing the name of the window in which image to be displayed.

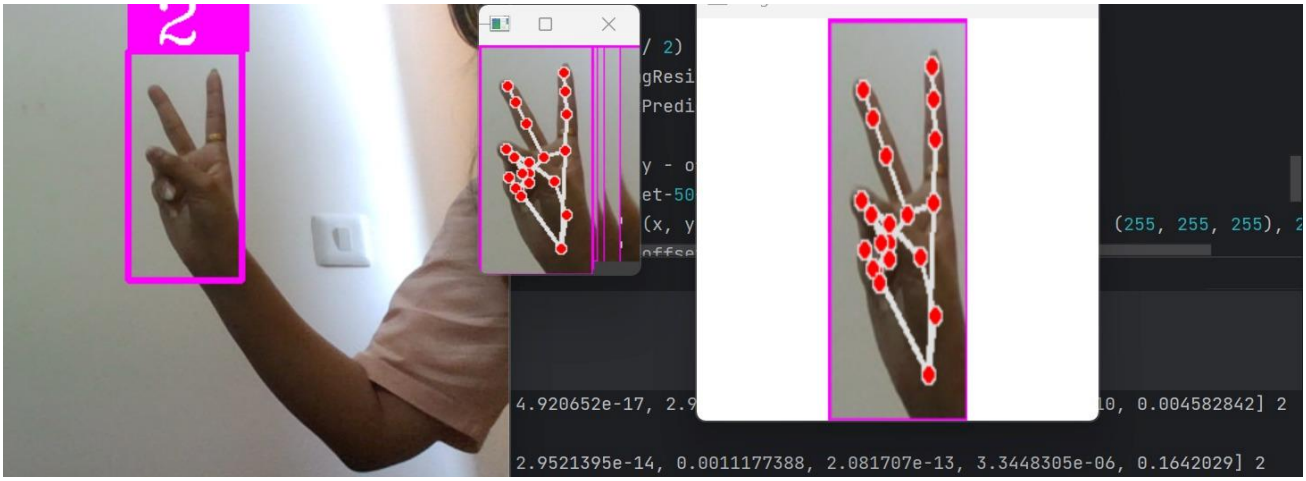
image: It is the image that is to be displayed.

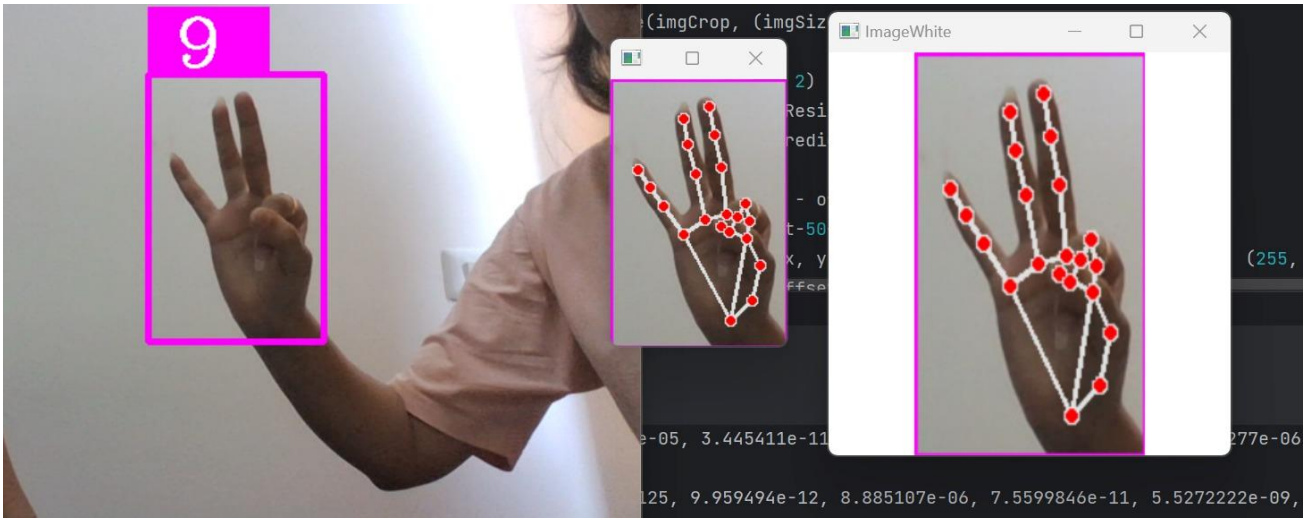
Return Value: It doesn't returns anything.

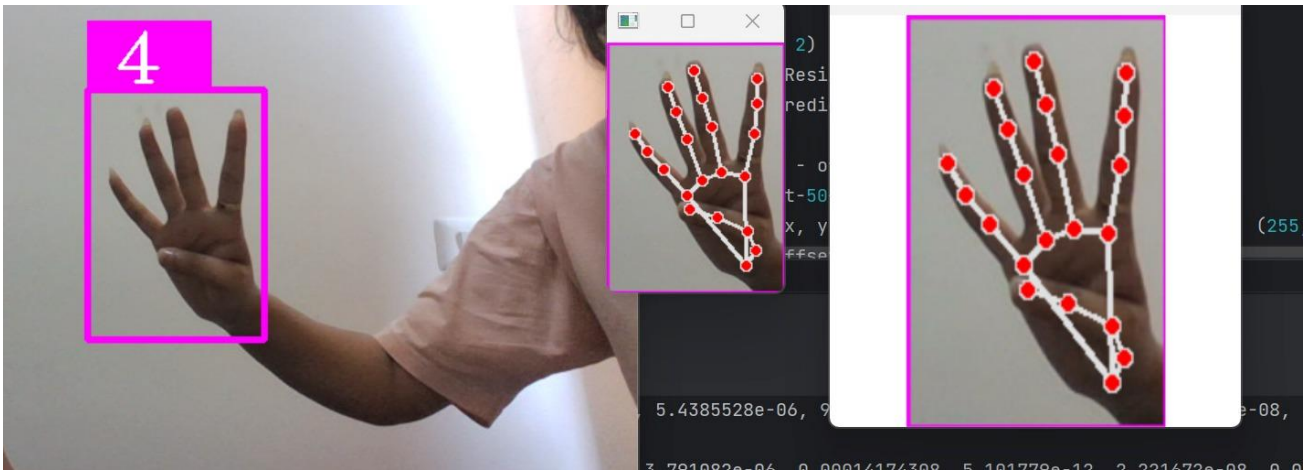
Cv2.waitkey(): waitkey() function of Python OpenCV allows users to display a window for given milliseconds or until any key is pressed. It takes time in milliseconds as a parameter and waits for the given time to destroy the window, if 0 is passed in the argument it waits till any key is pressed.

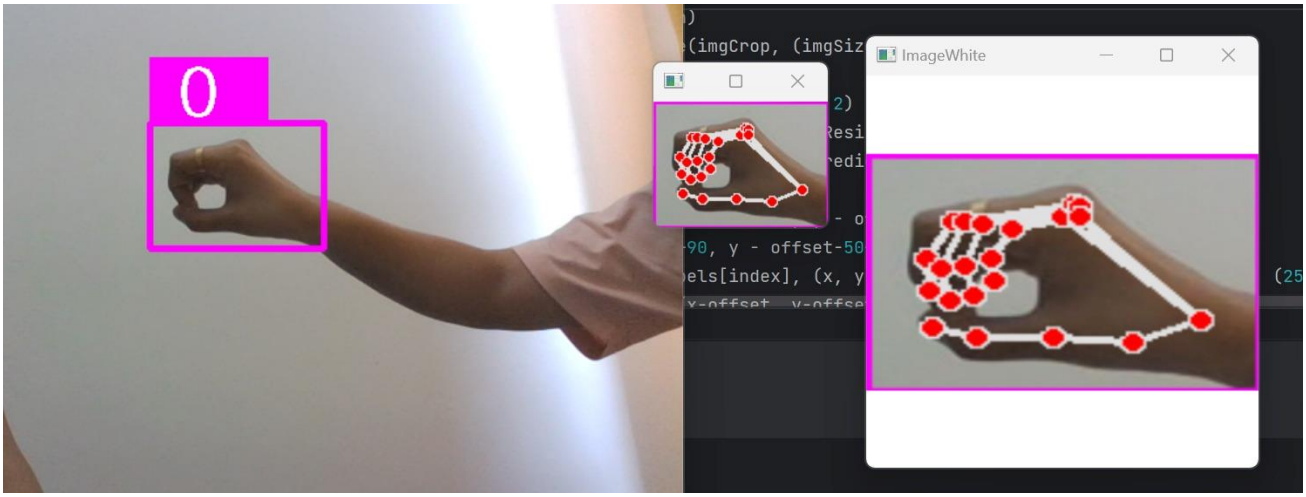
```
    cv2.imshow("ImageCrop", imgCrop)
    cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", imgOutput)
cv2.waitKey(1)
```

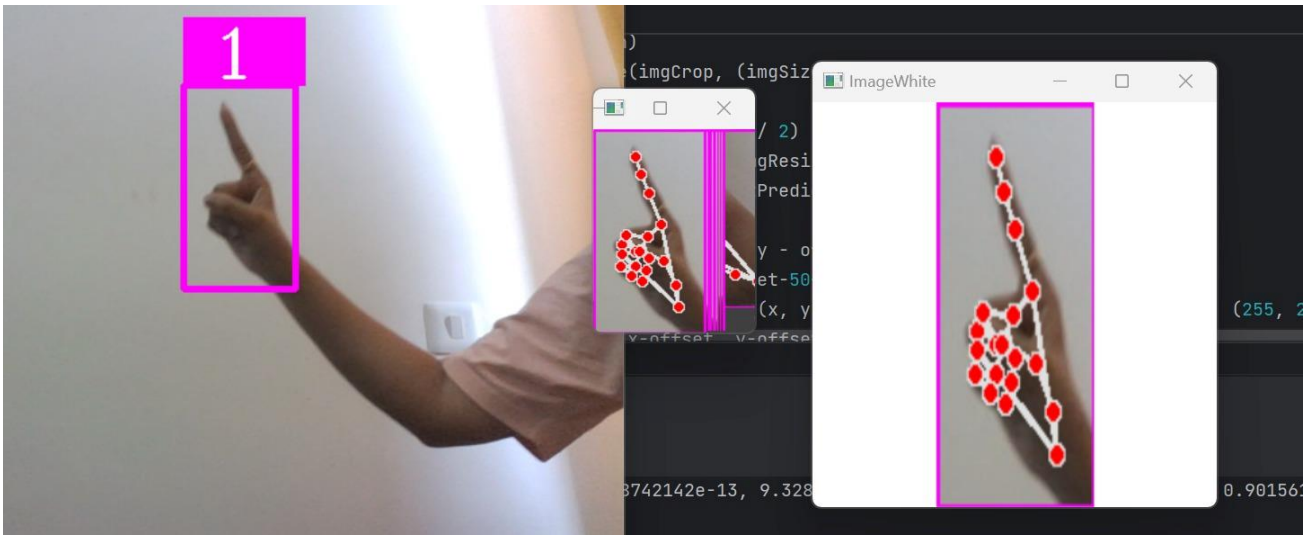
Some of the images detected are as shown:


2
(255, 255, 255),
4.920652e-17, 2.9
10, 0.004582842] 2
2.9521395e-14, 0.0011177388, 2.081707e-13, 3.3448305e-06, 0.1642029] 2


9
ImageWhite
(imgCrop, (imgSiz
(255,
e-05, 3.445411e-11
277e-06
125, 9.959494e-12, 8.885107e-06, 7.5599846e-11, 5.5272222e-09,


4
(255
5.4385528e-06, 9
e-08,





CONCLUSION:

This is a solution to sign language translation. Hard-of-hearing individuals who need to communicate with someone who does not understand sign language can have a personalized, virtual interpreter at anytime, anywhere. We have successfully developed sign language detection project. This can be further extended for detecting the English alphabets. Some of the disadvantages of the project is:

- There should be sufficient light around while detecting.
- Only pre-defined signs will be recognized.

#