



**DEPARTMENT OF PHYSICS AND ASTROPHYSICS
UNIVERSITY OF DELHI**

ADVANCED ELECTRONICS LAB

PROJECT REPORT(2021-22)

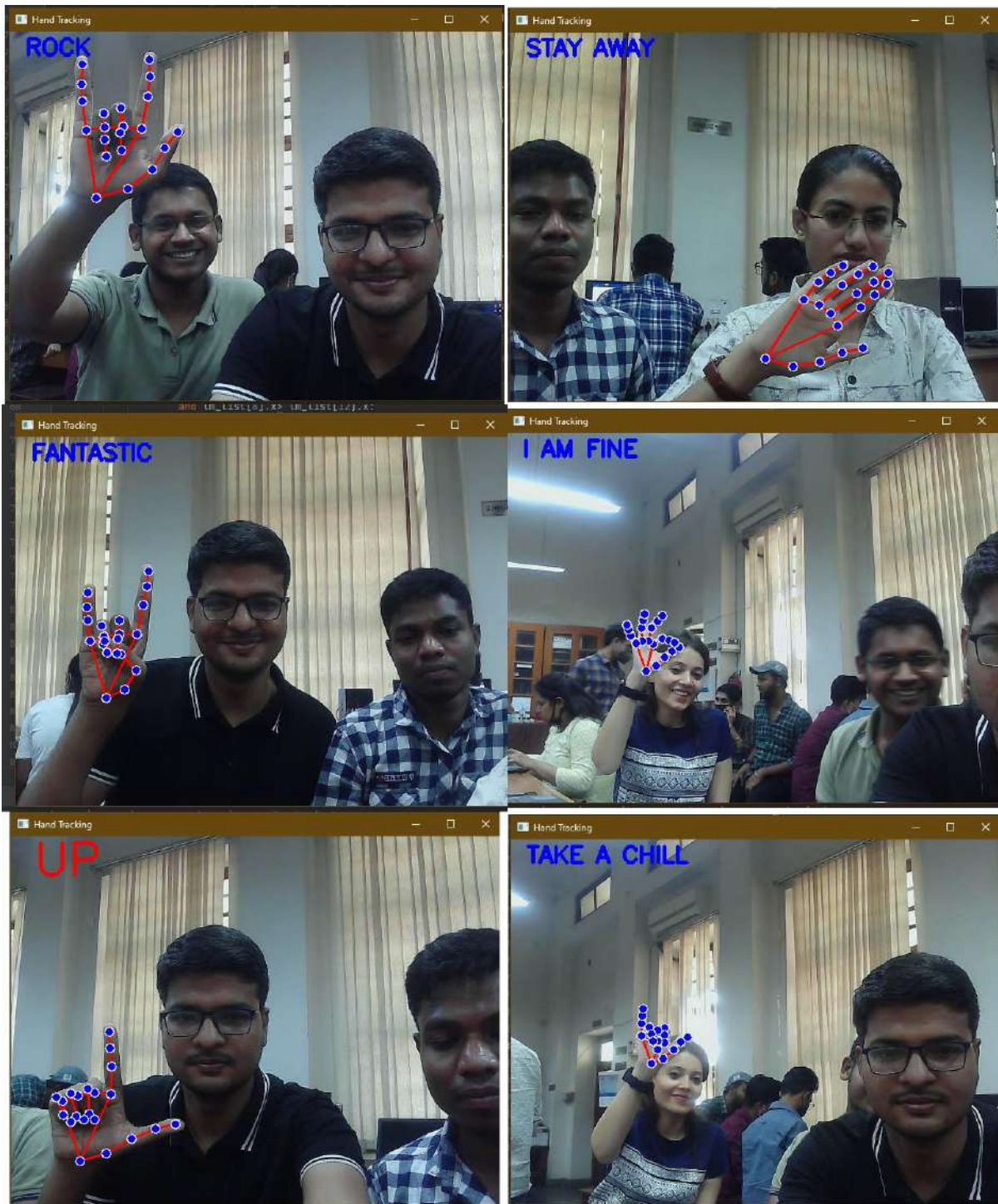
SIGN LANGUAGE RECOGNITION USING PYTHON

GROUP-2 (BARDEEN)

GROUP MEMBERS:

1. DEEPANSHU SHARMA (20025762016)
2. DIPIKA SHARMA (20025762019)
3. JAYTE PRAKASH KURIL (20025762026)
4. KANIKA KARDAM (20025762028)
5. KESHAV SHARMA (20025762031)
6. NEHA (20025762037)

A Glimpse



Acknowledgement

We would like to express our special thanks of gratitude to all the faculty members Prof.Amarjeet Kaur, Prof.Amita Chandra, Prof.Amitabh Mukherjee, Prof. G.S.Chilana, Dr.Nishant shankhwar for their spirited encouragement and inspiring discussions throughout the session January 2022 to April 2022.

We are also fortunate to have a company of all the motivated students of electronic lab (2021-2022).

Without their help and cooperation, it would have been difficult for us to enjoy the academic environment of the lab. We are also grateful to the lab staff for the constant help during the course of the work.

Contents

1. Introduction-----	01
2. Python installation-----	07
3. Libraries description-----	16
4. Approach-1 using machine learning by recognising hand landmarks-----	60
5. Flowchart-----	69
6. Different signs achieved-----	74
7. Code of the program-----	70
8. Approach-2 using tensorflow object detection API-----	77
9. Conclusion-----	106
10. Future scope-----	107
11. References-----	108

Introduction

This project focuses on the techniques employed for hand gesture recognition and introduces their merits and demerits under various circumstances. The aim of the project is to convert sign language to text and audio to be easily recognised by deaf and dumb people. We have tried here two different approaches to achieve the same.

1. Through object detection using tensorflow.
2. Using machine learning

The project is purely based on gesture recognition. In this day to day world it has become a necessity to have a computer based solution for deaf people. The goal is to make computers to understand human language and develop a user-friendly human computer interface. Gestures are non verbally exchanged information. Gesture recognition is an aspect of human computer interaction that demonstrates an academic treatise and is a vital in popularising the notion of a human-to-human connection, that must imply the correlation between the user and the machine .

Gesture analysis is a scientific field that can recognise gestures such as hand, arm, head, and even structural motions that usually entail a certain posture and/or motion. Using hand gestures, the individual may send out more information in a shorter amount of time. Our aim was to apply computer-vision ideas to the real-time processing of gesture outputs. It concentrates on gesture recognition in the open CV framework using the Python language.

Language is a huge part in communication. Languages are useless to a person with a disability. Gesture is a vital and meaningful mode of communication for the visually impaired person . So here is the computer-based method for regular people to understand what the differently abled individual is trying to say. For monitoring, there are various similar algorithms and object recognition systems.

PART – I

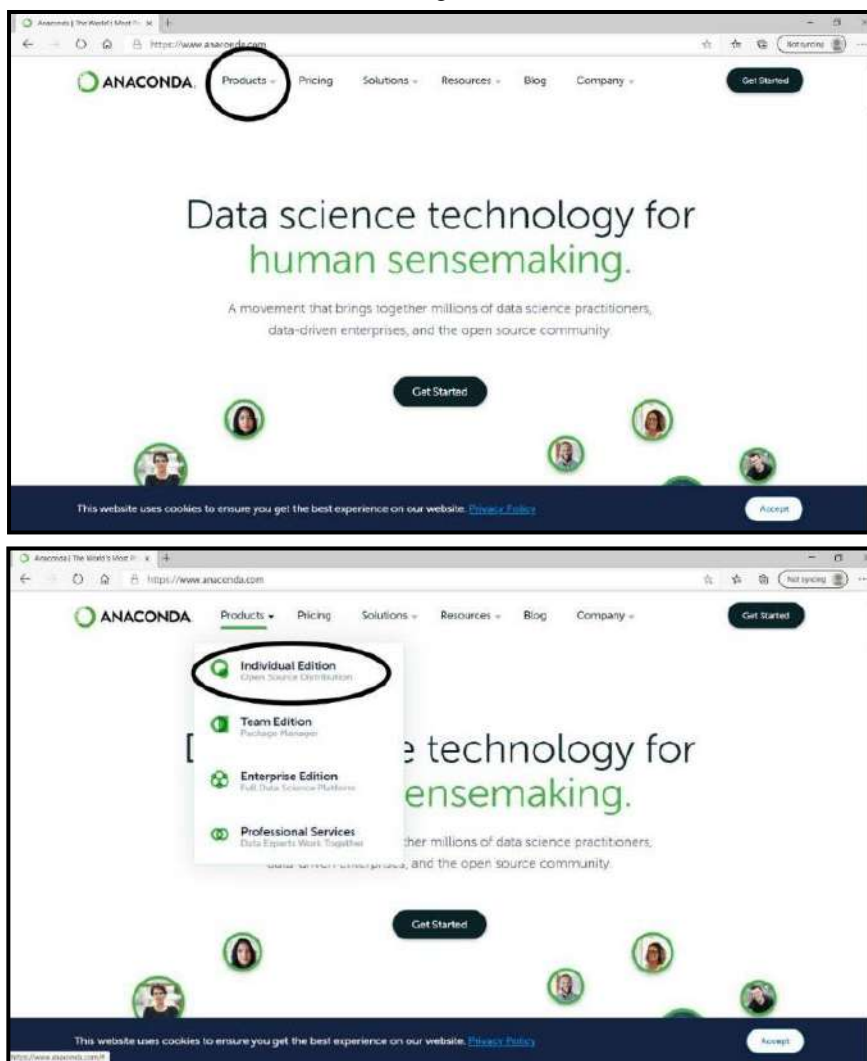
INSTALLATION OF PYTHON ON LAPTOPS/PCs

Anaconda

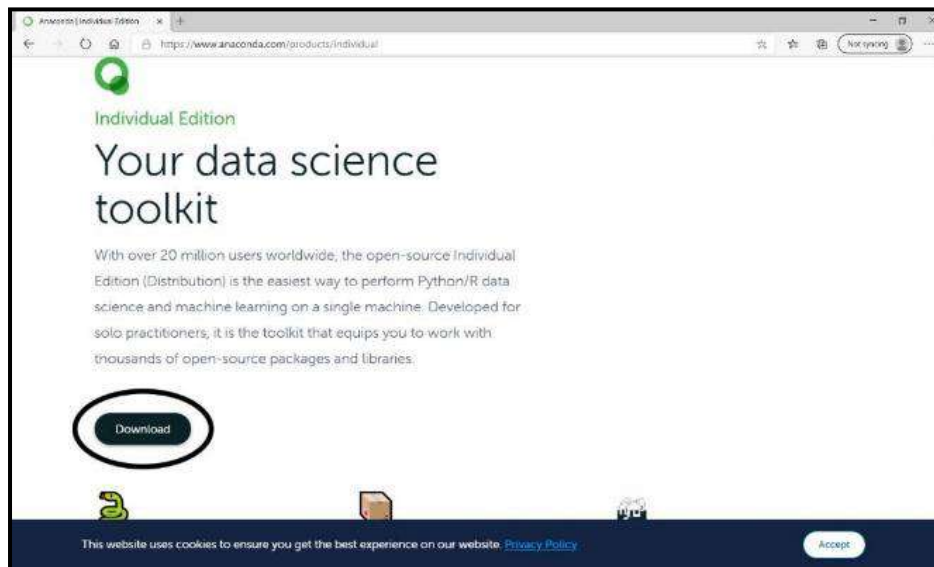
Anaconda is a package manager, an environment manager, a Python/R data science distribution, and a collection of over 7,500+ open-source packages. Anaconda is free and easy to install.

How to install “Anaconda” on your laptop

- Open “anaconda.com”, click on Products, then go to Individual Edition



- You will be directed to the following page; scroll down and click on **Download** and you will be directed to Anaconda Installers.



- Depending upon your operating system download and install the package.



PART – II

USING

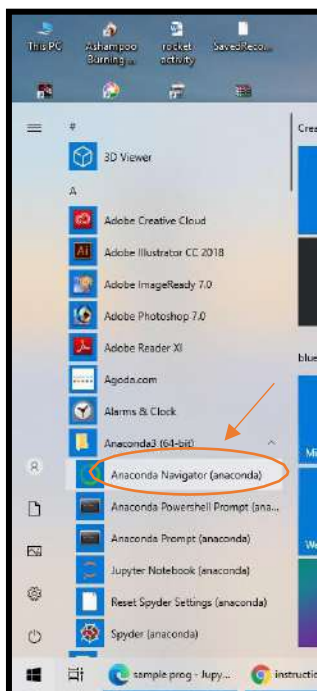
ANACONDA

- Most of the libraries are pre-installed in Anaconda.
- To write a program you can either use Spyder or Jupyter Notebook.
- To launch Spyder, search for Spyder in your windows search bar.
- Before running any program in Spyder, you need to save the program. The program will be saved in .py format automatically. In case of Jupyter Notebook, the files are saved with .ipynb extension.
- You are advised to save a copy of all your programs in a notepad document as well.
- Please save all your programs in the same folder so that you can easily access them later.
- Refer to the Anaconda documentation available on the official website for information on saving and running programs.
- Things you should know before writing and executing programs-
 - How to create and save a new folder (where you will be saving your program files and outputs)
 - How to save your program file in the desired format in that folder.
 - How to retrieve the files from the folder at a later time.
- Anything after # in a python program is considered as a comment.

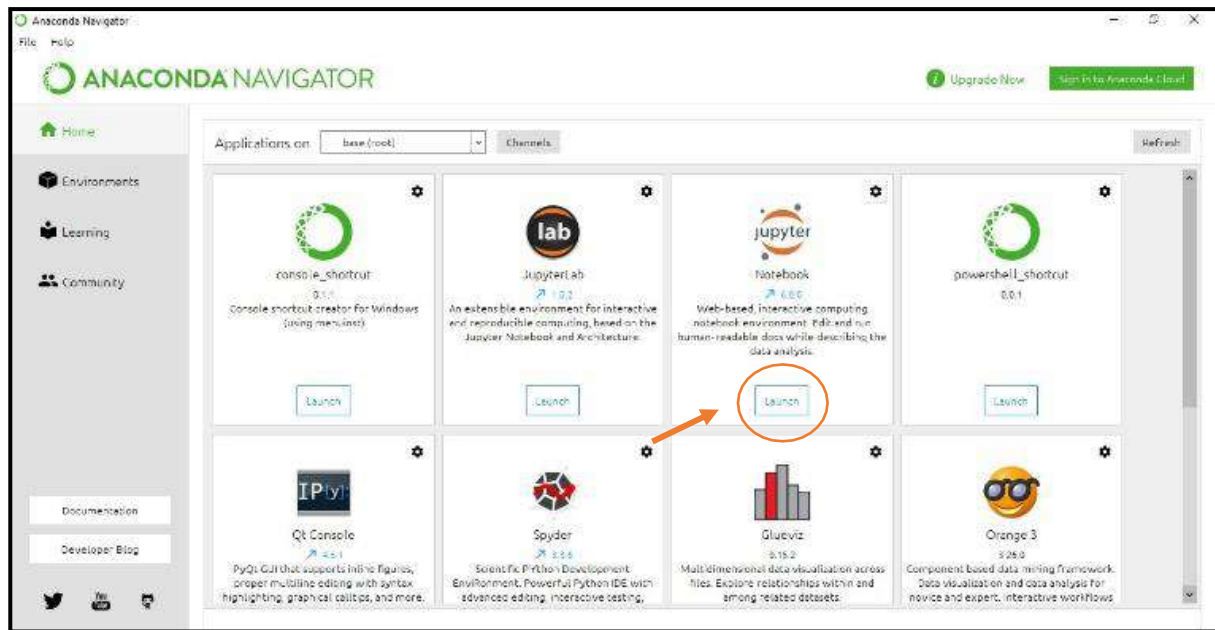
THE SCREENSHOTS IN THIS DOCUMENT ARE FROM JUPYTER NOTEBOOK

OPENING JUPYTER NOTEBOOK IN WINDOWS / macOS

1. START→ANACONDA → ANACONDA NAVIGATOR (Mac users can access the same using the Launchpad)

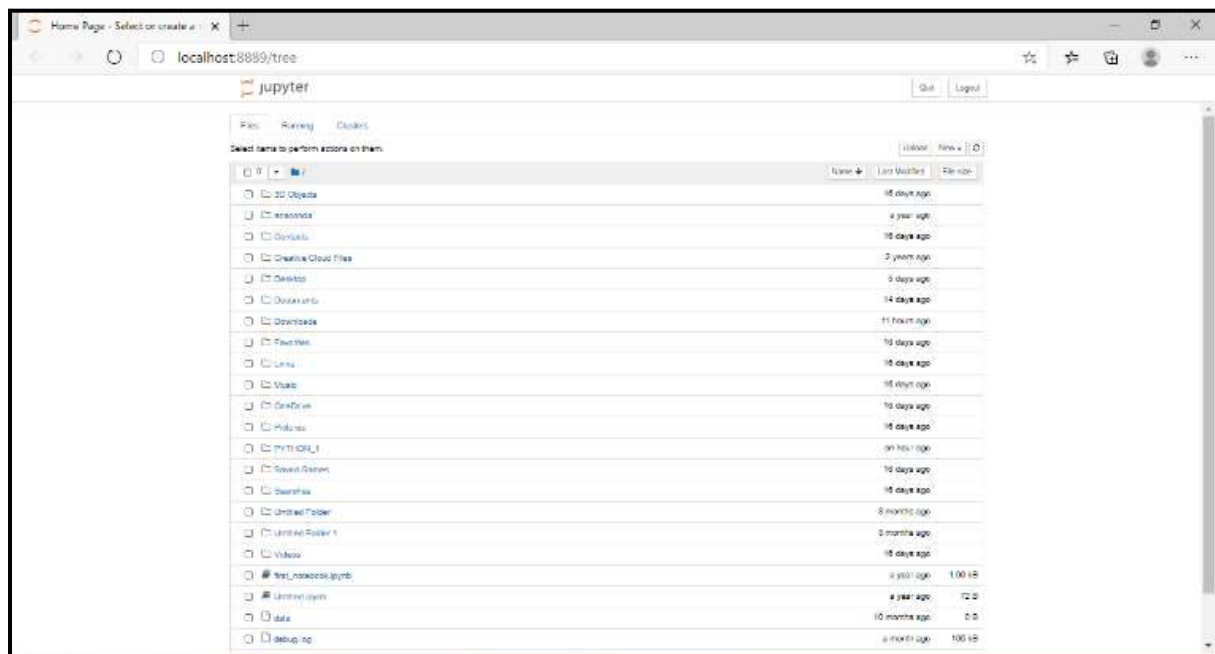


2. ANACONDA NAVIGATOR OPENS UP AS SHOWN BELOW



3. INSTALL JUPYTER NOTEBOOK & LAUNCH JUPYTER NOTEBOOK

4. ONCE YOU LAUNCH JUPYTER NOTEBOOK, A NEW WINDOW OPENS UP AS SHOWN BELOW

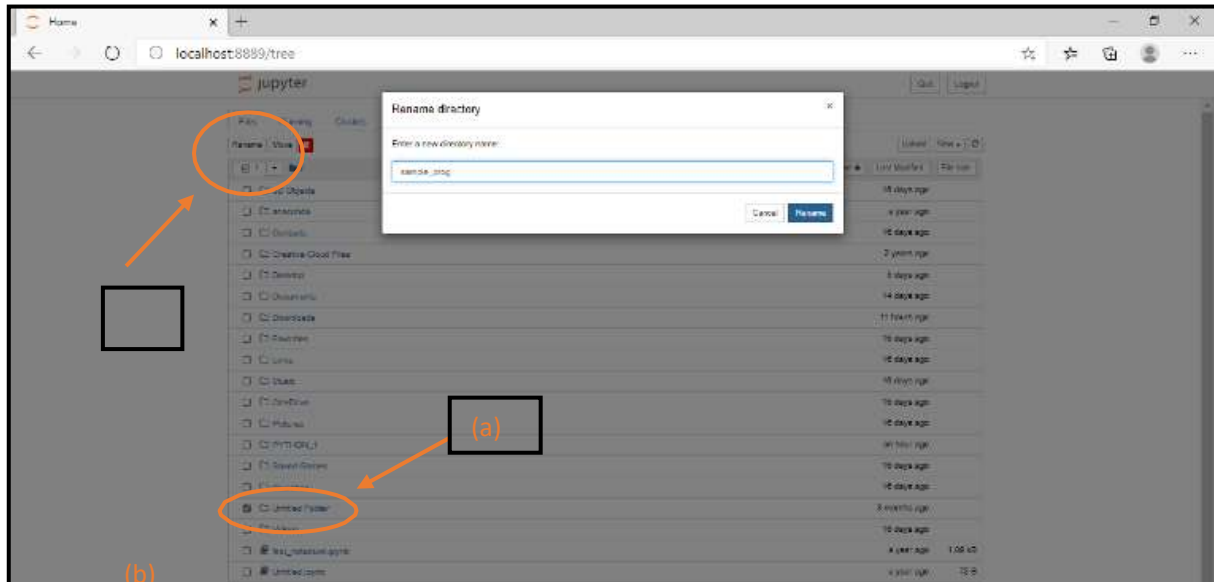


5. CREATE A NEW FOLDER. BY DEFAULT IT WILL BE CREATED WITH A NAME 'UNTITLED FOLDER'.



(a) A NEW FOLDER WILL APPEAR IN THE LIST AS 'UNTITLED FOLDER'. SELECT THE CHECKBOX IN-FRONT OF IT.

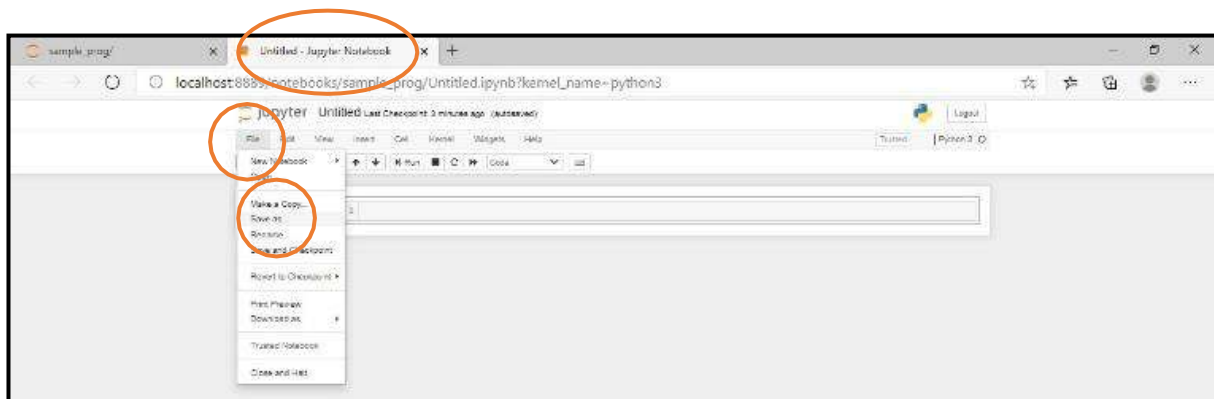
(b) RENAME IT BY SELECTING THE CHECKBOX AND THEN CLICKING ON **RENAME** THAT APPEARS AT THE TOP. (REFER TO THE CIRCLED PARTS OF THE IMAGE)
(You can give it any name. In the illustration below, it has been named as sample_prog)



6. CLICK ON THE NEW FOLDER YOU CREATED IN STEP 5 ò CLICK ON NEW ò SELECT PYTHON 3



7. A NEW UNTITLED JUPYTER NOTEBOOK TAB OPENS UP. THIS IS WHERE YOU WILL WRITE

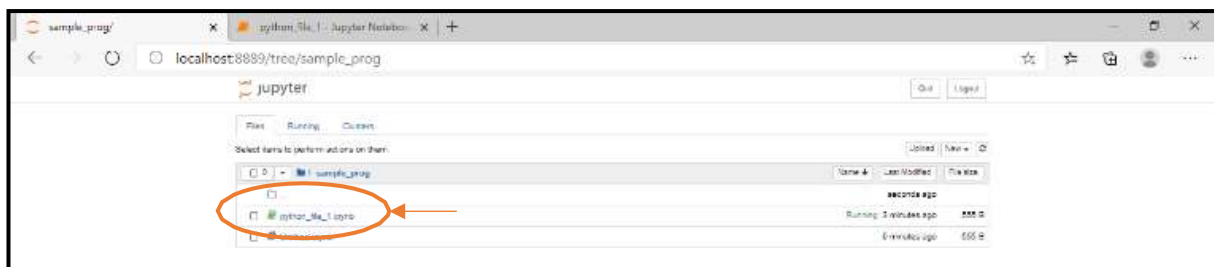
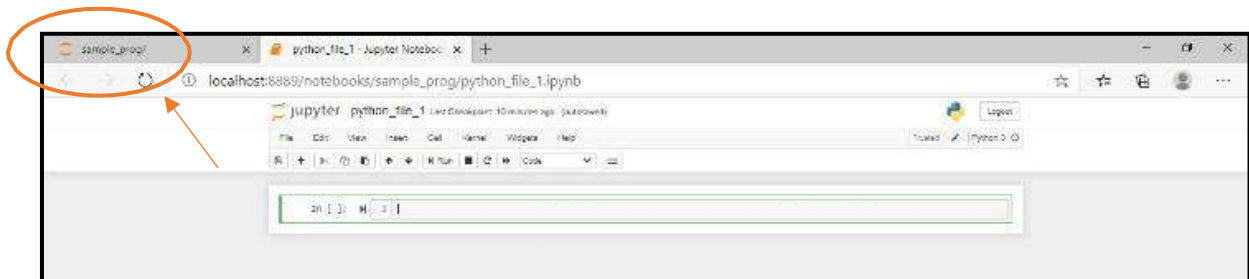


YOUR PROGRAMS AND EXECUTE THEM. BEFORE YOU DO THAT, SAVE THIS NOTEBOOK AS

IN THE ILLUSTRATION BELOW, THE FILE HAS BEEN NAMED 'python_file_1'. NOTE THAT IT IS SAVED INSIDE THE FOLDER WHICH WAS CREATED IN STEP 5.



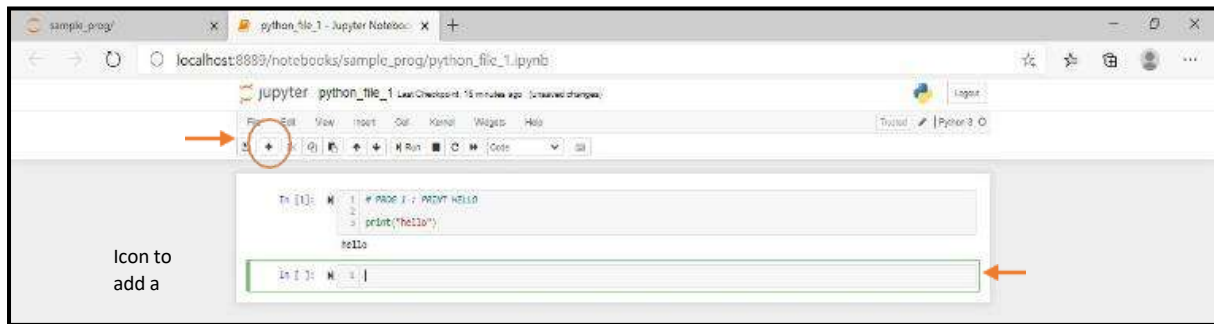
8. IF YOU GO BACK TO THE ADJACENT TAB sample_prog/, YOU SHOULD BE ABLE TO SEE THE NEW FILE YOU CREATED IN STEP 7.



YOU CAN NOW WRITE YOUR PROGRAMS IN THE FILE **python_file_1** AND EXECUTE THEM USING THE **RUN** ICON OR BY PRESSING **SHIFT+ENTER** ON YOUR KEYBOARD

NOTE:

1. YOU NEED NOT CREATE A NEW FILE FOR EVERY PROGRAM. ONCE YOU EXECUTE A PROGRAM, YOU CAN SIMPLY ADD A “**NEW CELL**” FOR A NEW PROGRAM AND EXECUTE THE NEW PROGRAM IN THE SAME FILE.



2. A new cell below the previous one can be used to generate new results and add to the outcomes of the results generated or overwrite variables in the cell above it.

The screenshot displays a Jupyter Notebook interface with the following code cells and annotations:

- Cell 1:** `# PROG 1 : PRINT HELLO`
`print("hello")`
Output: `hello`
- Cell 2:** `# PROG 2:`
`a = 5`
`b = 29`
`c = 6`
`d = 54`
`sum = a+b+c+d`
`print(sum)`
Output: `94`
- Cell 3:** `alpha = a-b`
`beta = c-d`
Output: `-24`
Annotation: An orange arrow points from the `a` and `b` variables in this cell to the previous cell's output, with the text: "The values assigned to a, b, c and d in the previous cell have been used to calculate alpha and beta."
- Cell 4:** `a = 7`
`sum = a+b+d`
`print(sum)`
Output: `90`
Annotation: An orange arrow points from the `a` variable in this cell to the previous cell's output, with the text: "Variable a has been assigned a new value and the new value is used in calculating the sum, while the values of b and d remain the same"
- Cell 5:** `1`

LIBRARIES USED:

1.OPENCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

It was developed for the common infrastructure provision for computer vision application and to escalate the use of machine learning perception in commercial products.

It has licensed of BSD(Berkeley Source Distribution) that make it easier for businesses uses and modifying the code for other purposes.

BSD licenses are a low restriction type of license for open source software that does not put requirements on redistribution. As a low restriction and requirement license type, Berkeley Source Distribution (BSD) licenses are used for the distribution of many freeware, shareware and open source software.

OpenCV claimed that it has more than 2500 optimized algorithms which includes comprehensive set of both classic and state of the art computer vision and machine learning algorithms .

These algorithms are used to detect and recognize faces , identify objects, classifying human actions in videos, tracking camera movement, tracking moving objects,extract 3-Dimensional models of objects, producing 3-D point cloud from stereocameras, stitch images together to produce a high resolution image of an entire screen, find similar images from an image database, remove red eyes from images taken due to flash light,follow eye movements, recognizing scenery and establish markers to overlay it with augmented reality and many more.....

OpenCV on its official website tells about the various uses and development made by countries in various aspects by using OpenCV. Few of them are -----

Countries	Development in field
<i>Israel</i>	<i>Detecting intrusions in surveillance videos</i>
<i>China</i>	<i>Monitoring mine equipments</i>
<i>Europe</i>	<i>Swimming pool drowning accidents detection</i>
<i>Spain</i>	<i>Running interactive arts</i>
<i>Turkey</i>	<i>Checking Runways for debris</i>
<i>Japan</i>	<i>Inspecting labels on products in factories and face detection</i>

OpenCV has C++, Python, Java and MATLAB interfaces.

It supports Windows, Linux, Android and Mac OS.

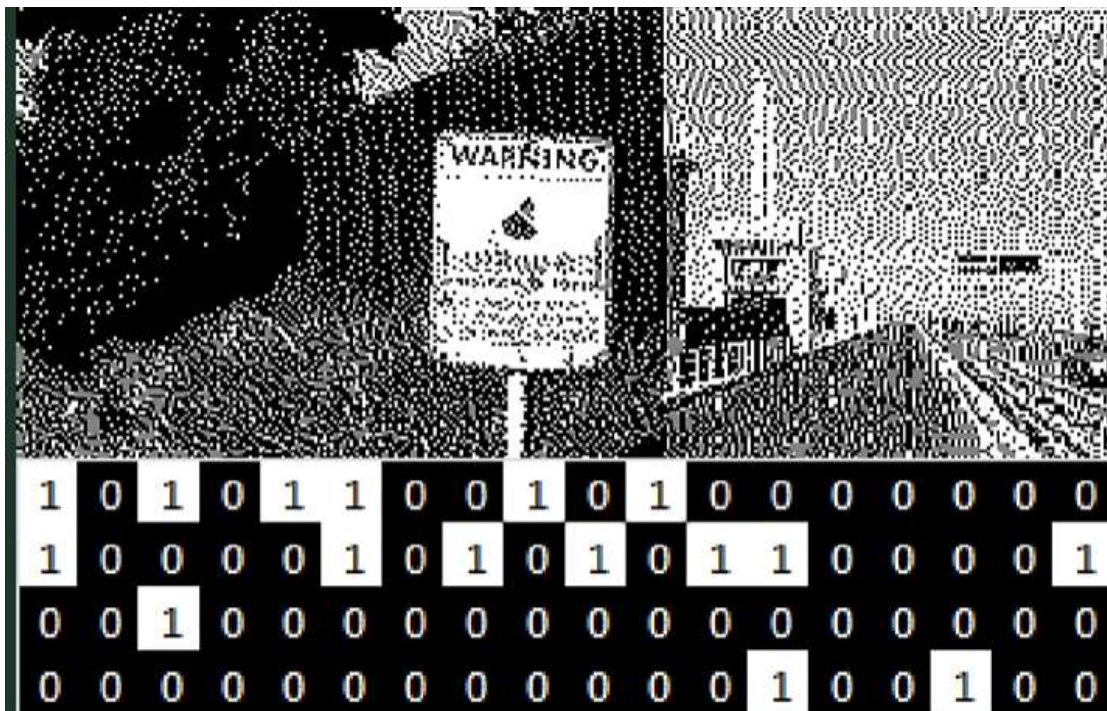
Although OpenCV has been written originally in C++ and has templated interfaces that work seamlessly with STL (Standard Template Library) containers.

We have access to read and write in image using OpenCV

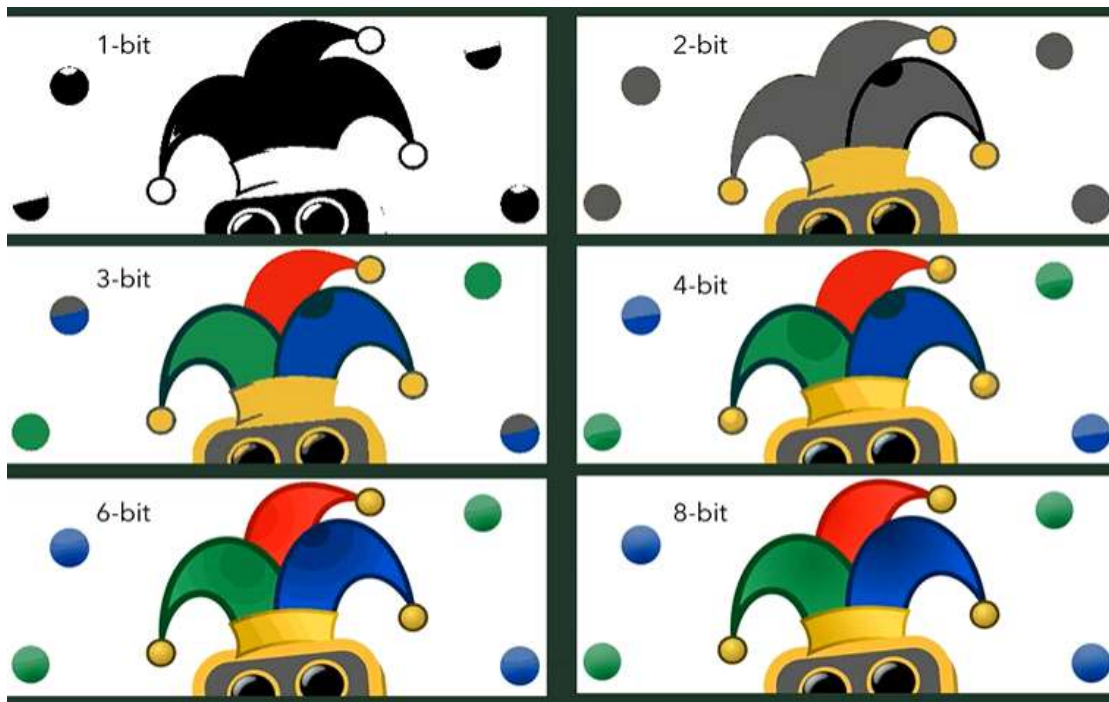
Images are represented by rows and columns and determined by pixels



binary numbers assigned to bits in images are



and different types of bit images are:-



1-bit means it has $(2^1)=2$ colours.

2-bit means it has $(2^2)=4$ colours.

.....

n-bit means it has (2^n) colours.

for example we have 8-bit image that means it will have $(2^8)=256$ colours. Various combinations of bit represents different colours

8-BIT

- 00000000 → BLACK
 - 00000001 → BLACK WITH LESS DARKNESS
 - 00000010
 - ...
 - ...
 - ...
 - 11111111 → WHITE
- OTHERS

These different colours comes from 3-channels(Red, Green and Blue).

Based on the channels contribution (RGB has taken values from 0 to 255) that helps OpenCV in computer vision.

Finally get the ultimate result of images detection.



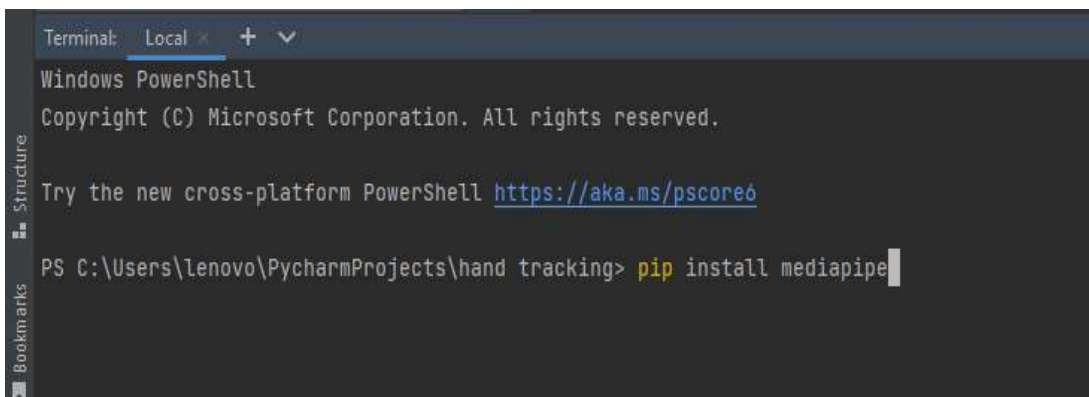
MediaPipe Library

What is MediaPipe?

- **Mediapipe** is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for media processing.
- **Machine Learning solutions** are a complete set of intellectual property, tools and software for AI development across a vast array of devices.

How to install MediaPipe Library?

- I. Open the terminal of Pycharm.
- II. Type `pip install mediapipe` and press enter.



```
Terminal: Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\lenovo\PycharmProjects\hand tracking> pip install mediapipe
```

What is possible with MediaPipe?

There are a number of AI problems that can be done by MediaPipe. Here some are mentioned:

- I. **Live Hand Tracking**
- II. **Box Tracking**
- III. **Face Mesh**
- IV. **Hair Segmentation**
- V. **Object Tracking and many more.**

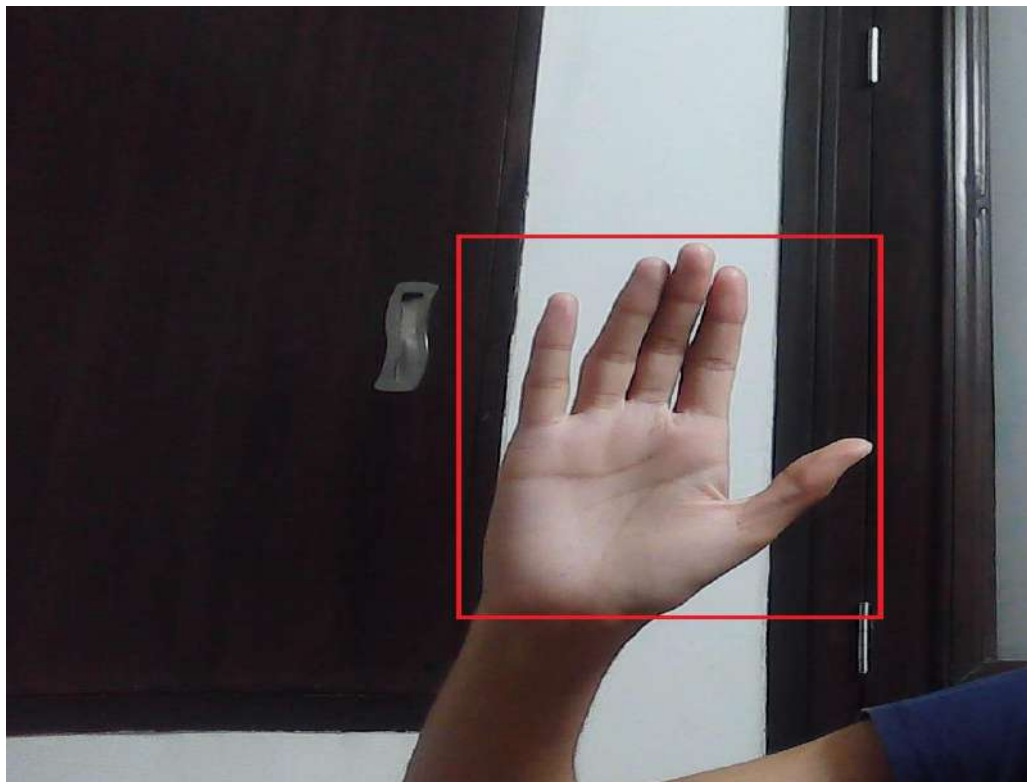
Real-Time Hand Tracking Project

- MediaPipe Hands is a high-fidelity hand and finger tracking solution.
- It employs machine learning (ML) to identify 21 3-D landmarks of a hand from just a single frame.

Hand tracking solution utilizes an ML pipeline consisting of several models working together:

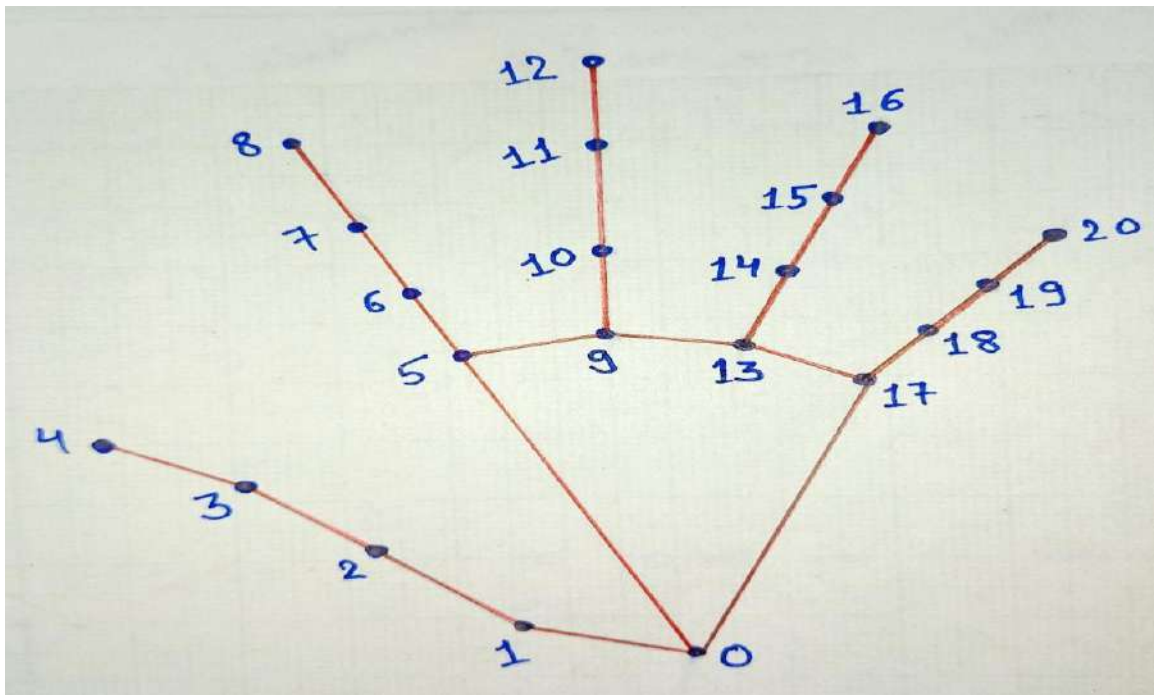
- I. **A palm detector** model operates on the full image and returns an oriented hand bounding box.

Using Palm Detection Model, we achieve an average precision of 95.7% in palm detection.



- II. **A hand landmark model** that operates on the cropped image region defined by the palm detector.

It returns high fidelity 21 3-D hand key points via regression that is direct coordinate prediction.



- III. A gesture recognizer classifies the previously computed key point configuration into a discrete set of gestures. This works on some logic that we give in programming to define that particular sign.

Playsound in Python

Playsound in python is one of the most popular Audio libraries. Most of the audio files are in MP3 and WAV file formats. WAV audio files are the simplest digital audio format with lossless high recording rates as a result WAV files are large compared to other formats. For the same reason, MP3 formats are used which are small in size and compresses files with very little difference to overall sound quality. Also, it is very easy to convert WAV to MP3 with open-source and free software which are widely available over the internet.

Play sound on Python is easy. There are several modules that can play a sound file (.wav). Some of the other modules are pudub, native player, simple audio, snack sound kit ,etc. The main difference is in the ease of use and supported file formats. All of them should work with Python 3. The audio file should be in the same directory as your python program, unless you specify a path.

Playing Audio

Using Playsound Module

The ready-to-use package we can play audio files with only a single line of code. One can play WAV or MP3 files with it. It's a single function module with no dependencies for playing sound.

Documentation of playsound library mentions that it has been tested for WAV and MP3 files, but may also work with other file formats whose testing was left up to the user. The playsound module contains only one thing – the function (also named) playsound.

How to install playsound module?

Run the following command to install the packages:

pip install playsound

- The *playsound* module contains only a single function named **playsound()**.
- It requires one argument: the path to the file with the sound we have to play. It can be a *local file*, or a *URL*.
- There's an optional second argument, **block**, which is set to *True* by default. We can set it to *False* for making the function run **asynchronously**.
- It works with both **WAV** and **MP3** files.

1. PLAYSOUND

Method 1: Using **PLAYSOUND** module

Example1: For mp3 format

Following are the code lines to play a file:

```
# import required module
from playsound import playsound

# for playing note.mp3 file
playsound('/path/note.mp3')
print('playing sound using playsound')
```

Example2: For WAV format

```
# import required module
from playsound import playsound

# for playing note.wav file
playsound('/path/note.wav')
print('playing sound using playsound')
```

We can also use other methods for playing sound. Some of them are following:

2. PYDUB

Pydub is a Python library used for manipulating audios and adding effects to it. This library is a very simple and easy but high - level interface which is based on FFmpeg and inclined by jquery. This library is used for adding id3 tags in audio, slicing it, and concatenating the audio track.

Run the following commands to install the packages:

```
sudo apt-get install ffmpeg libavcodec-extra
```

```
pip install pydub
```

Note: We can open WAV files with python. For opening mp3, we'll need [ffmpeg](#) or [libav](#).

This module uses the **from_wav()** method for playing wav file and **from_mp3()** method for playing an mp3 file. The **play()** method is used to play the wav and mp3 file:

Example 1: For WAV format

```
# import required modules
from pydub import AudioSegment
from pydub.playback import play

# for playing wav file
song = AudioSegment.from_wav("note.wav")
print('playing sound using pydub')
play(song)
```

Example 2 : For mp3 format

```
# import required module
from pydub import AudioSegment
from pydub.playback import play
```

```
# for playing mp3 file
song = AudioSegment.from_mp3("note.mp3")
print('playing sound using pydub')
play(song)
```

3. SIMPLE AUDIO

Simpleaudio is a Python library which is a cross - platform. This library is also used for playing back WAV files without any dependencies. simpleaudio library waits for the file to finish the playing audio in WAV format before termination of the script. This is mainly designed to play **WAV** files and **NumPy arrays**.

4. NATIVE PLAYER

In this method, we play sounds **natively** on our system. This method plays the audio file with an **external player** installed on your terminal.

OS Module

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on.

Getting Started

Python's OS module comes packaged within python when installed. This means you do not need to separately install it using PIP. In order to access its various methods/functions, you just need to import the module.

Just write this in your python editor

```
1  # OS MODULE INSTALL
2  import os
```

Now you've imported the module, you can start using its various functions.

We can use this OS module for different tasks.

1. Create a directory
2. Delete a directory
3. Change the directory
4. Rename the folder
5. Create the file txt
6. Can check the elements in that folder etc.

Getting current working directory

The currently working directory is the folder in which the python script is saved and being run from.

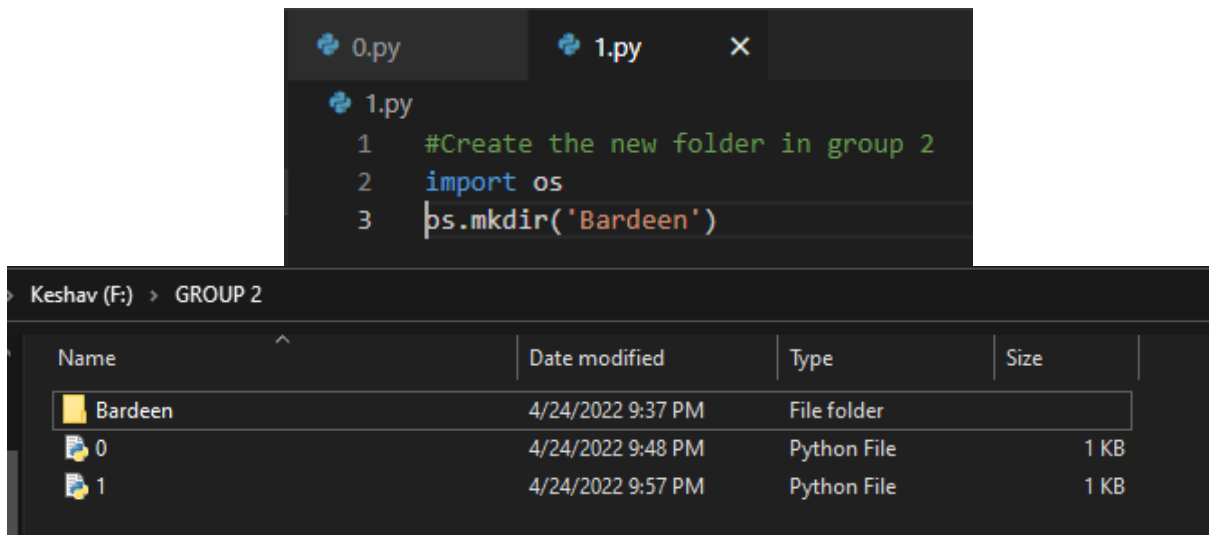
```
2
3  #Check the current working directory
4  import os
5  print(os.getcwd())
6
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

PS F:\GROUP 2> & "C:/Users/Keshav Sharma/AppData/Local/F:\GROUP 2

Creating a directory

Suppose I want to make another folder in F drive Group 2 folder.



The image shows a code editor window with a file named '1.py' open. The code inside the file is as follows:

```
1 #Create the new folder in group 2
2 import os
3 os.mkdir('Bardeen')
```

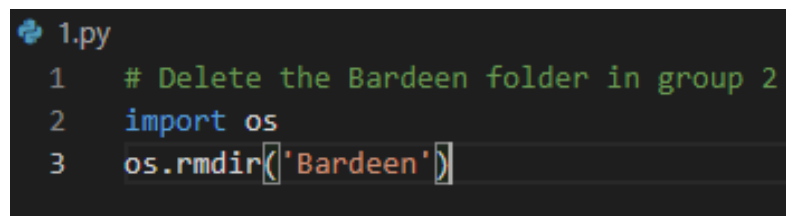
Below the code editor, a file explorer window is open, showing the contents of the 'GROUP 2' folder in the 'Keshav (F:)' drive. The folder 'Bardeen' has been successfully created, along with two Python files, '0' and '1'.

Name	Date modified	Type	Size
Bardeen	4/24/2022 9:37 PM	File folder	
0	4/24/2022 9:48 PM	Python File	1 KB
1	4/24/2022 9:57 PM	Python File	1 KB

Deleting a directory

Now that you know how to create a folder, let us learn about how you can delete one.

In order to delete a directory, we will be using the `rmdir()` function, it stands for remove directory

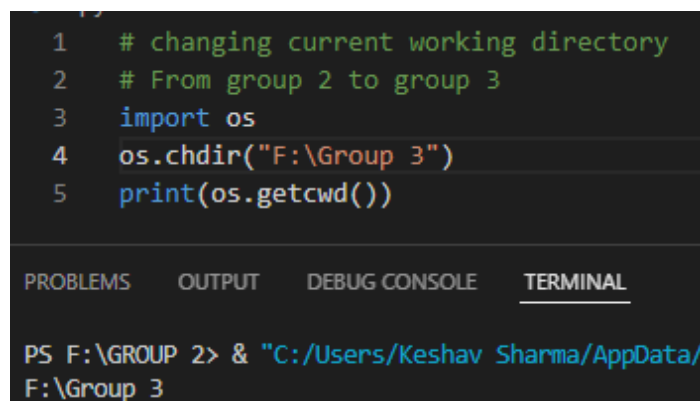


The image shows a code editor window with a file named '1.py' open. The code inside the file is as follows:

```
1 # Delete the Bardeen folder in group 2
2 import os
3 os.rmdir('Bardeen')
```

Output : This will delete the Bardeen folder in Group 2.

Changing the Director



The image shows a code editor window with a file named '1.py' open. The code inside the file is as follows:

```
1 # changing current working directory
2 # From group 2 to group 3
3 import os
4 os.chdir("F:\Group 3")
5 print(os.getcwd())
```

Below the code editor, a terminal window is open, showing the output of the code. The terminal displays the current working directory as 'F:\Group 3'.

```
PS F:\GROUP 2> & "C:/Users/Keshav Sharma/AppData/Local/Programs/Python/Python39-64/Scripts/python" -i -x -c "os.chdir('F:\Group 3'); print(os.getcwd())"
F:\Group 3
```


Renaming a directory

In order to rename a folder, we use the rename function present in the os module.

```
1  # Change the folder name Bardeen to Cooper
2  import os
3  os.rename("Bardeen", "Cooper")
```

The above line of code renames Bardeen to Cooper.

Basic file manipulation

Now that you know how to work around with folders, let us look into file manipulation.

Creating a file

```
import os
open('file.txt', 'a').close()
```

A file named file.txt is created in the current working directory.

Check the elements in that folder

The listdir() function returns the list of all files and directories in the specified directory.

```
2.py
1  # Checking the elements
2  # In that folder
3  import os
4  print(os.listdir("F:\\GROUP 2"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS F:\GROUP 2> & "C:/Users/Keshav Sharma/AppData
['0.py', '1.py', '2.py', 'file.txt']
```

OBJECT DETECTION API

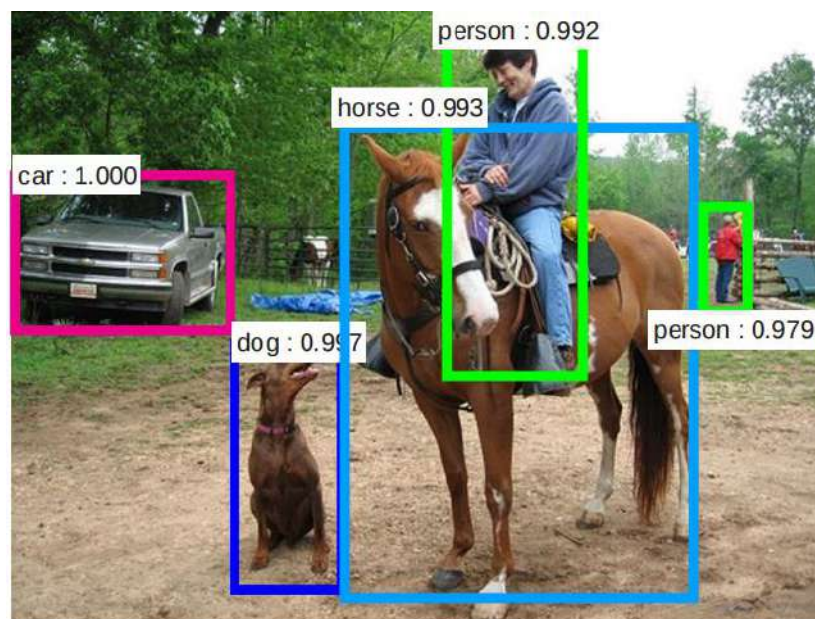
The use of computer vision has revolutionized our society today. It is possible to communicate with machines and make robots do particular tasks to differentiate between different objects. Deep learning and artificial intelligence have come to aid computer vision with similar capabilities to the human eye in identifying items.

Whereas there is a similarity in image recognition and object detection, it is good to note the difference. In image recognition, the computer identifies the main object and gives it a label.

On the other hand, in object detection, computer vision makes a bounding box in each item's image and provides a brand. Due to the ever-growing demand for object detection, there is thus a need for object detection APIs.

Importance

- Offer real-time detection of objects.
- Accurately identifies multiple objects in an image.
- Provides data for fast analysis.
- Detect text and capture movements
- It can be used to identify unsafe equipment such as weapons.





Tensorflow :-

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Tensorflow Example, Google users can experience a faster and more refined search experience with AI. If the user types a keyword in the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning.

1. Researchers
2. Data Scientists
3. Programmers

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java

Why use Tensorflow?,

It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

Main Use Cases of TensorFlow:-

One of the most well-known uses of TensorFlow are Sound based applications. With the proper data feed, neural networks are capable of understanding audio signals. These can be:

- Voice recognition – mostly used in IoT, Automotive, Security and UX/UI
- Voice search – mostly used in Telecoms, Handset Manufacturers
- Sentiment Analysis – mostly used in CRM
- Flaw Detection (engine noise) – mostly used in Automotive and Aviation

Regarding common use cases, we are all familiar with voice-search and voice-activated assistants with the new wide spreading smartphones such as Apple's Siri, Google Now for Android and Microsoft Cortana for Windows Phone.

Sound based applications also can be used in CRM. A use case scenario might be: TensorFlow algorithms standing in for customer service agents, and route customers to the relevant information they need, and faster than the agents.

2.Text Based Applications

Further popular uses of TensorFlow are, text based applications such as sentimental analysis (CRM, Social Media), Threat Detection

(Social Media, Government) and Fraud Detection (Insurance, Finance)

Language Detection is one of the most popular uses of text based applications.

- We all know **Google Translate**, which supports over 100 languages translating from one to another. The evolved versions can be used for many cases like translating jargon legalese in contracts into plain language.
- **Text Summarization**

Google also found out that for shorter texts, summarization can be learned with a technique called sequence-to-sequence learning. This can be used to produce headlines for news articles. Below, you can see [an example](#) where the model reads the article text and writes a suitable headline.

3. Image Recognition

Mostly used by Social Media, Telecom and Handset Manufacturers; Face Recognition, Image Search, Motion Detection, Machine Vision and Photo Clustering can be used also in Automotive, Aviation and Healthcare Industries. Image Recognition aims to recognize and identify people and objects in images as well as understanding the content and context

4. Time Series

TensorFlow Time Series algorithms are used for analyzing time series data in order to extract meaningful statistics. They allow forecasting non-specific time periods in addition to generate alternative versions of the time series.

The most common use case for Time Series is **Recommendation**. You've probably heard of this use from Amazon, Google, Facebook and Netflix where they analyze customer activity and compare it to the millions of other users to determine what the customer might like to purchase or watch. These recommendations are getting even smarter, for example, they offer

you certain things as gifts (not for yourself) or TV shows that your family members might like.

5. Video Detection

TensorFlow neural networks also work on video data. This is mainly used in Motion Detection, Real-Time Thread Detection in Gaming, Security, Airports and UX/UI fields. Recently, Universities are working on Large scale Video Classification datasets like YouTube-8M aiming to accelerate research on large-scale video understanding, representation learning, noisy data modeling, transfer learning, and domain adaptation approaches for video.

Installation:

Install Tensorflow (Linux , Mac OS and Windows)

1. Download Anaconda
2. Create an environment with all must-have libraries.
3. `$ conda create n tensorflow python=3.8`
4. `$ activate tensorflow`
5. `$ conda install pandas matplotlib jupyter notebook`
6. `$ pip install tensorflow`

Install Anaconda Python 3.8

WindowsLinux

- Go to <https://www.anaconda.com/products/individual> and click the “Download” button
- Download the [Python 3.8 64-Bit Graphical Installer](#) or the [32-Bit Graphical Installer](#) system requirements.
- Run the downloaded executable (`.exe`) file to begin the installation.
- (Optional) In the next step, check the box “Add Anaconda3 to my PATH environment variable”. This will make Anaconda your default Python distribution, which should ensure that you have the same default Python distribution across all editors.

Create a new Anaconda virtual environment

- Open a new *Terminal* window
- Type the following command:

```
conda create -n tensorflow pip python=3.9
```
- The above will create a new virtual environment with name `tensorflow`

Activate the Anaconda virtual environment

- Activating the newly created virtual environment is achieved by running the following in the *Terminal* window:

```
conda activate tensorflow
```
- Once you have activated your virtual environment, the name of the environment should be displayed within brackets at the beginning of your cmd path specifier, e.g.:

```
(tensorflow) C:\Users\Dell>
```

TensorFlow Installation:-

Getting setup with an installation of TensorFlow can be done in 3 simple steps.

Install the TensorFlow PIP package

- Run the following command in a *Terminal* window:

```
pip install --ignore-installed --upgrade tensorflow==2.5.0
```

Verify your Installation

- Run the following command in a *Terminal* window:

```
python -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```
- Once the above is run, you should see a print-out similar to the one below:

```
2022-02-22 19:20:32.614181: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2022-02-22 19:20:32.620571: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore
```


above cudart dLError **if** you do **not** have a GPU set up on your machine.

- 2022-02-22 19:20:35.027232: I
tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
- 2022-02-22 19:20:35.060549: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1561]
Found device 0 **with** properties:
- pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti
computeCapability: 6.1
- coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth: 238.66GiB/s
- 2022-04-22 19:20:35.074967: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'cudart64_101.dll'; dLError: cudart64_101.dll **not** found
- 2022-04-22 19:20:35.084458: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'cublas64_10.dll'; dLError: cublas64_10.dll **not** found
- 2022-02-22 19:20:35.094112: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'cufft64_10.dll'; dLError: cufft64_10.dll **not** found
- 2022-02-22 19:20:35.103571: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'curand64_10.dll'; dLError: curand64_10.dll **not** found
- 2022-02-22 19:20:35.113102: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'cusolver64_10.dll'; dLError: cusolver64_10.dll **not** found
- 2022-02-22 19:20:35.123242: W
tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could **not** load dynamic library 'cusparse64_10.dll'; dLError: cusparse64_10.dll **not** found

- 2022-02-22 19:20:35.140987: I
tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
- 2022-02-22 19:20:35.146285: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:1598] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly **if** you would like to use GPU. Follow the guide at <https://www.tensorflow.org/install/gpu> **for** how to download **and** setup the required libraries **for** your platform.
- Skipping registering GPU devices...
- 2022-02-22 19:20:35.162173: I
tensorflow/core/platform/cpu_feature_guard.cc:143] Your CPU supports instructions that this TensorFlow binary was **not** compiled to use: AVX2
- 2022-02-22 19:20:35.178588: I
tensorflow/compiler/xla/service/service.cc:168] XLA service 0x15140db6390 initialized **for** platform Host (this does **not** guarantee that XLA will be used). Devices:
- 2022-02-22 19:20:35.185082: I
tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
- 2022-02-22 19:20:35.191117: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor **with** strength 1 edge matrix:
- 2022-02-22 19:20:35.196815: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]
- tf.Tensor(1620.5817, shape=(), dtype=float32)

GPU Support (Optional)

Although using a GPU to run TensorFlow is not necessary, the computational gains are substantial. Therefore, if your machine is equipped with a compatible CUDA-enabled GPU, it is recommended that you follow the steps listed below to install the relevant libraries necessary to enable TensorFlow to make use of your GPU.

By default, when TensorFlow is run it will attempt to register compatible GPU devices. If this fails, TensorFlow will resort to running on the platform's CPU. This can also be observed in the printout shown in the previous section, under the "Verify the install" bullet-point, where there are a number of messages which report missing library files

(e.g. `Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found`).

In order for TensorFlow to run on your GPU, the following requirements must be met:

Prerequisites

Nvidia GPU (GTX 650 or newer)
CUDA Toolkit v11.2
CuDNN 8.1.0

Install CUDA Toolkit

WindowsLinux

- Follow this [link](#) to download and install CUDA Toolkit 11.2
- Installation instructions can be found [here](#)

Install CUDNN

WindowsLinux

- Go to <https://developer.nvidia.com/rdp/cudnn-download>
- Create a user profile if needed and log in
- Select [Download cuDNN v8.1.0 \(January 26th, 2021\), for CUDA 11.0,11.1 and 11.2](#)
- Download [cuDNN Library for Windows \(x86\)](#)
- Extract the contents of the zip file (i.e. the folder named `cuda`) inside `<INSTALL_PATH>\NVIDIA GPU Computing Tool`

`kit\CUDA\v11.2\`, where `<INSTALL_PATH>` points to the installation directory specified during the installation of the CUDA Toolkit. By

default `<INSTALL_PATH>` = `C:\Program Files`.

Environment Setup:-

WindowsLinux

- Go to *Start* and Search “environment variables”
- Click “Edit the system environment variables”. This should open the “System Properties” window
- In the opened window, click the “Environment Variables...” button to open the “Environment Variables” window.
- Under “System variables”, search for and click on the `Path` system variable, then click “Edit...”
- Add the following paths, then click “OK” to save the changes:

- `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v11.2\bin`
- `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v11.2\libnvvp`
- `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v11.2\include`
- `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v11.2\extras\CUPTI\lib64`
- `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v11.2\cuda\bin`

Update your GPU drivers (Optional):-

If during the installation of the CUDA Toolkit (see [Install CUDA Toolkit](#)) you selected the *Express Installation* option, then your GPU drivers will have been overwritten by those that come bundled with the CUDA toolkit. These drivers are typically NOT the latest drivers and, thus, you may wish to update your drivers.

- Go to <http://www.nvidia.com/Download/index.aspx>
- Select your GPU version to download
- Install the driver for your chosen OS

Verify the installation

- Run the following command in a **NEW Terminal** window:

- ```
python -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```
- Once the above is run, you should see a print-out similar to the one below:

- 2022-03-08 18:28:38.452128: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library cudart64\_110.dll
- 2022-03-08 18:28:40.948968: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library nvcuda.dll
- 2022-03-08 18:28:40.973992: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1733] Found device 0 with properties:
- pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
- coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth: 238.66GiB/s
- 2022-03-08 18:28:40.974115: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library cudart64\_110.dll
- 2022-03-08 18:28:40.982483: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library cublas64\_11.dll
- 2022-03-08 18:28:40.982588: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library cublasLt64\_11.dll
- 2022-03-08 18:28:40.986795: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:53] Successfully opened dynamic library cufft64\_10.dll

- 2022-03-08 18:28:40.988451: I  
tensorflow/stream\_executor/platform/default/dso\_loader.cc:53]  
Successfully opened dynamic library curand64\_10.dll
- 2022-03-08 18:28:40.994115: I  
tensorflow/stream\_executor/platform/default/dso\_loader.cc:53]  
Successfully opened dynamic library cusolver64\_11.dll
- 2022-03-08 18:28:40.998408: I  
tensorflow/stream\_executor/platform/default/dso\_loader.cc:53]  
Successfully opened dynamic library cusparse64\_11.dll
- 2022-03-08 18:28:41.000573: I  
tensorflow/stream\_executor/platform/default/dso\_loader.cc:53]  
Successfully opened dynamic library cudnn64\_8.dll
- 2022-03-08 18:28:41.001094: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1871]  
Adding visible gpu devices: 0
- 2022-03-08 18:28:41.001651: I  
tensorflow/core/platform/cpu\_feature\_guard.cc:142] This  
TensorFlow binary **is** optimized **with** oneAPI Deep Neural  
Network Library (oneDNN) to use the following CPU  
instructions **in** performance-critical operations: AVX AVX2  
• To enable them **in** other operations, rebuild TensorFlow **with**  
the appropriate compiler flags.
- 2022-03-08 18:28:41.003095: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1733]  
Found device 0 **with** properties:
- pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti  
computeCapability: 6.1
- coreClock: 1.683GHz coreCount: 19 deviceMemorySize:  
8.00GiB deviceMemoryBandwidth: 238.66GiB/s
- 2022-03-08 18:28:41.003244: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1871]  
Adding visible gpu devices: 0
- 2022-03-08 18:28:42.072538: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1258]  
Device interconnect StreamExecutor **with** strength 1 edge  
matrix:

- 2022-03-08 18:28:42.072630: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1264]  
0
- 2022-03-08 18:28:42.072886: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1277]  
0: N
- 2022-03-08 18:28:42.075566: I  
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1418]  
Created TensorFlow device  
(/job:localhost/replica:0/task:0/device:GPU:0 with 6613 MB  
memory) -> physical GPU (device: 0, name: GeForce GTX  
1070 Ti, pci bus id: 0000:02:00.0, compute capability: 6.1)
- tf.Tensor(641.5694, shape=(), dtype=float32)
- Notice from the lines highlighted above that the library files  
are now **Successfully opened** and a debugging message is  
presented to confirm that TensorFlow has  
successfully **Created TensorFlow device**.

### TensorFlow Object Detection API Installation

Now that you have installed TensorFlow, it is time to install the TensorFlow Object Detection API.

### Downloading the TensorFlow Model Garden

- Create a new folder under a path of your choice and name  
it **TensorFlow**.  
(e.g. **C:\Users\Dell\RealTimeObjectDectection\TensorFlow**).
- From your *Terminal* **cd** into the **TensorFlow** directory.
- To download the models you can either use [Git](#) to clone  
the [TensorFlow Models repository](#) inside  
the **TensorFlow** folder, or you can simply download it as  
a [ZIP](#) and extract its contents inside the **TensorFlow** folder.  
To keep things consistent, in the latter case you will have to  
rename the extracted folder **models-master** to **models**.



- You should now have a single folder named `models` under your `TensorFlow` folder, which contains another 4 folders as such:

```
TensorFlow/
├── models/
│ ├── community/
│ ├── official/
│ ├── orbit/
│ └── research/
└── ...
```

### Protobuf Installation/Compilation

The Tensorflow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be downloaded and compiled.

This should be done as follows:

- Head to the [protoc releases page](#)
- Download the latest `protoc-*.zip` release (e.g. `protoc-3.12.3-win64.zip` for 64-bit Windows)
- Extract the contents of the downloaded `protoc-*.zip` in a directory `<PATH_TO_PB>` of your choice (e.g. `C:\Program Files\Google Protobuf`)
- Add `<PATH_TO_PB>\bin` to your `Path` environment variable (see [Environment Setup](#))
- In a new *Terminal* `1`, `cd` into `TensorFlow/models/research/` directory and run the following command:

```
From within TensorFlow/models/research/
protoc object_detection/protos/*.proto --python_out=.
```

NOTE: You MUST open a new *Terminal* for the changes in the environment variables to take effect.



## COCO API installation

As of TensorFlow 2.x, the `pycocotools` package is listed as [a dependency of the Object Detection API](#). Ideally, this package should get installed when installing the Object Detection API as documented in the [Install the Object Detection API](#) section below, however the installation can fail for various reasons and therefore it is simpler to just install the package beforehand, in which case later installation will be skipped.

## WindowsLinux

Run the following command to install `pycocotools` with Windows support:

```
pip install cython
pip install
git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
```

## Install the Object Detection API

Installation of the Object Detection API is achieved by installing the `object_detection` package. This is done by running the following commands from within `Tensorflow\models\research`:

```
From within TensorFlow/models/research/
cp object_detection/packages/tf2/setup.py .
python -m pip install --use-feature=2020-resolver .
```

## Note

During the above installation, you may observe the following error:

ERROR: Command errored out **with** exit status 1:

```
command: 'C:\Users\sglvladi\Anaconda3\envs\tf2\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '''C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-install-yn46ceci\\pycocotools\\setup.py'''';
```

```

__file__ = '''C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-install-
yn46ecei\\pycocotools\\setup.py''';f=getattr(tokenize, '''open''',
open)(__file__);code=f.read().replace(''''\\r\\n''',
'''\\n''');f.close();exec(compile(code, __file__, '''exec'''))' install
--record 'C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-record-
wpn7b6qo\\install-record.txt' --single-version-externally-managed --
compile --install-headers
'C:\\Users\\sglvladi\\Anaconda3\\envs\\tf2\\Include\\pycocotools'
 cwd: C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-install-
yn46ecei\\pycocotools\\
 Complete output (14 lines):
 running install
 running build
 running build_py
 creating build
 creating build\\lib.win-amd64-3.8
 creating build\\lib.win-amd64-3.8\\pycocotools
 copying pycocotools\\coco.py -> build\\lib.win-amd64-
3.8\\pycocotools
 copying pycocotools\\cocoeval.py -> build\\lib.win-amd64-
3.8\\pycocotools
 copying pycocotools\\mask.py -> build\\lib.win-amd64-
3.8\\pycocotools
 copying pycocotools__init__.py -> build\\lib.win-amd64-
3.8\\pycocotools
 running build_ext
 skipping 'pycocotools_mask.c' Cython extension (up-to-date)
 building 'pycocotools._mask' extension
 error: Microsoft Visual C++ 14.0 is required. Get it with "Build
Tools for Visual Studio":
https://visualstudio.microsoft.com/downloads/

ERROR: Command errored out with exit status 1:
'C:\\Users\\sglvladi\\Anaconda3\\envs\\tf2\\python.exe' -u -c 'import sys,
setuptools, tokenize; sys.argv[0] =
'''C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-install-
yn46ecei\\pycocotools\\setup.py''';

```

```
__file__='\"C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-install-yn46ceci\\pycocotools\\setup.py\"';f=getattr(tokenize, '\"open\"', open)(__file__);code=f.read().replace('\"\\r\\n\"', '\"\\n\"');f.close();exec(compile(code, __file__, '\"exec\"'))' install --record 'C:\\Users\\sglvladi\\AppData\\Local\\Temp\\pip-record-wpn7b6qo\\install-record.txt' --single-version-externally-managed --compile --install-headers 'C:\\Users\\sglvladi\\Anaconda3\\envs\\tf2\\Include\\pycocotools' Check the logs for full command output.
```

This is caused because installation of the `pycocotools` package has failed. To fix this have a look at the [COCO API installation](#) section and rerun the above commands.

Test your Installation

To test the installation, run the following command from within `Tensorflow\models\\research`:

```
From within TensorFlow/models/research/
```

```
python object_detection/builders/model_builder_tf2_test.py
```

Once the above is run, allow some time for the test to complete and once done you should observe a printout similar to the one below:

```
...
[OK]
ModelBuilderTF2Test.test_create_ssd_models_from_config
[RUN]
ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
I0608 18:49:13.183754 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
[OK]
ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
[RUN]
ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
```

```
I0608 18:49:13.186750 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
[OK]
ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
[RUN] ModelBuilderTF2Test.test_invalid_model_config_proto
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
I0608 18:49:13.188250 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
[OK] ModelBuilderTF2Test.test_invalid_model_config_proto
[RUN]
ModelBuilderTF2Test.test_invalid_second_stage_batch_size
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
I0608 18:49:13.190746 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
[OK]
ModelBuilderTF2Test.test_invalid_second_stage_batch_size
[RUN] ModelBuilderTF2Test.test_session
[SKIPPED] ModelBuilderTF2Test.test_session
[RUN]
ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0608 18:49:13.193742 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[OK]
ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[RUN] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
```

```
I0608 18:49:13.195241 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_unknown_meta_architectu
re): 0.0s
[OK] ModelBuilderTF2Test.test_unknown_meta_architecture
[RUN]
ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknow
n_ssd_feature_extractor): 0.0s
I0608 18:49:13.197239 29296 test_util.py:2102]
time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_ext
ractor): 0.0s
[OK]
ModelBuilderTF2Test.test_unknown_ssd_feature_extractor

Ran 24 tests in 29.980s

OK (skipped=1)
```

If the previous step completed successfully it means you have successfully installed .

## Training Custom Object Detector

So, up to now you should have done the following:

- Installed TensorFlow
- Installed TensorFlow Object Detection API

Now that we have done all the above, we can start doing some cool stuff. Here we will see how you can train your own object detector, and since it is not as simple as it sounds, we will have a look at:

1. How to organise your workspace/training files
2. How to prepare/annotate image datasets
3. How to generate tf records from such datasets
4. How to configure a simple training pipeline
5. How to train a model and monitor it's progress
6. How to export the resulting model and use it to detect objects.

## Preparing the Workspace

1. you should by now have a folder `Tensorflow`, placed under `<PATH_TO_TF>` (e.g. `C:/Users/Dell/RealTimeObject Detection`), with the following directory tree:

```
2. TensorFlow/
3. |— addons/ (Optional)
4. | |— labelImg/
5. |— models/
6. | |— community/
7. | |— official/
8. | |— orbit/
9. | |— research/
10. | |— ...
```

11. Now create a new folder under `TensorFlow` and call it `workspace`. It is within the `workspace` that we will store all our training set-ups. Now let's go under `workspace` and create another folder named `training_demo`. Now our directory structure should be as so:

```
12. TensorFlow/
13. |— addons/ (Optional)
14. | |— labelImg/
15. |— models/
16. | |— community/
17. | |— official/
18. | |— orbit/
19. | |— research/
20. | |— ...
21. |— workspace/
22. | |— training_demo/
```

23. The `training_demo` folder shall be our *training folder*, which will contain all files related to our model training. It is advisable to create a separate training folder each time we wish to train on a different dataset. The typical structure for training folders is shown below.

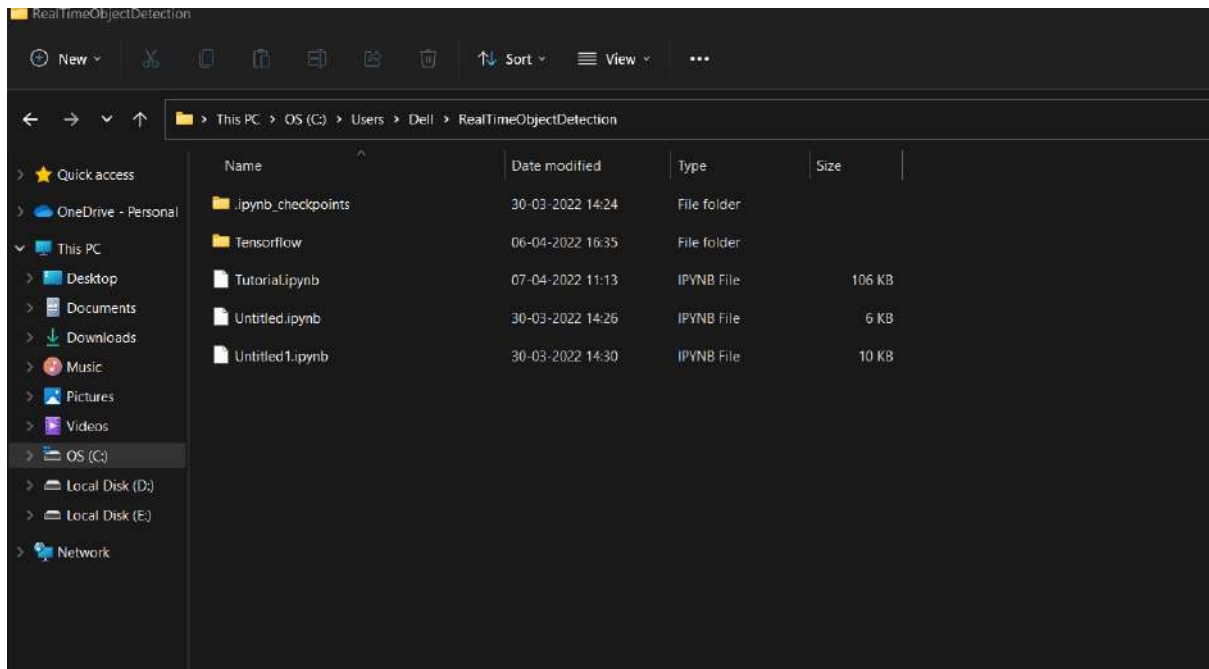
```
24. training_demo/
25. |— annotations/
26. |— exported-models/
27. |— images/
28. | |— test/
29. | |— train/
30. |— models/
31. |— pre-trained-models/
32. |— README.md
```

Here's an explanation for each of the folders/filer shown in the above tree:

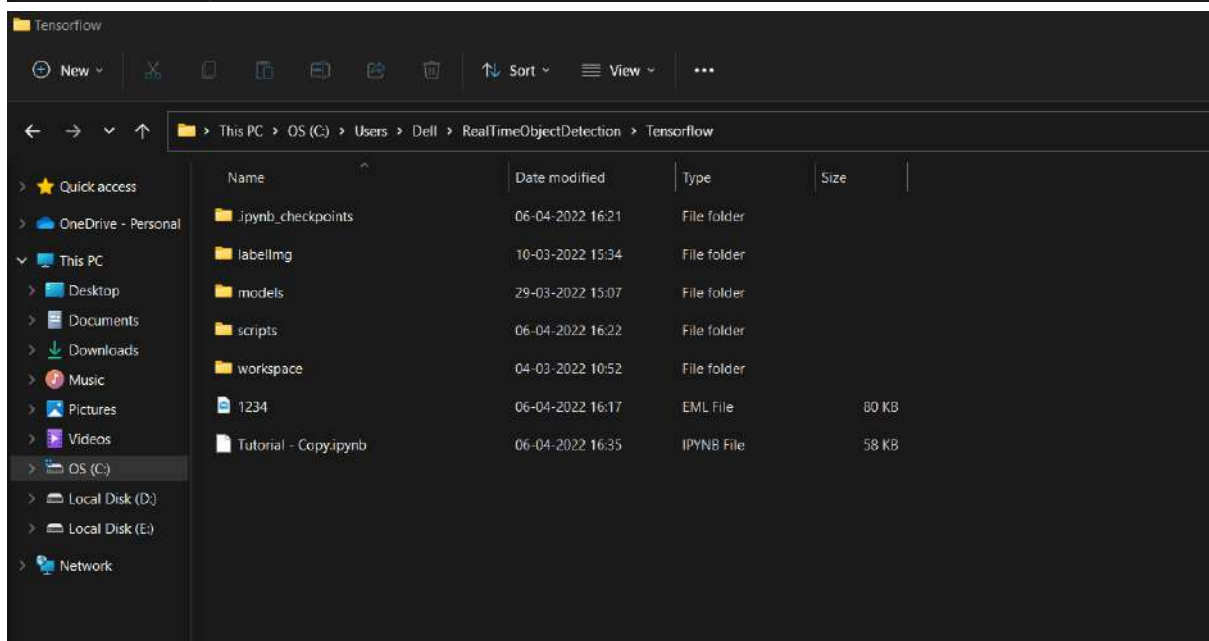
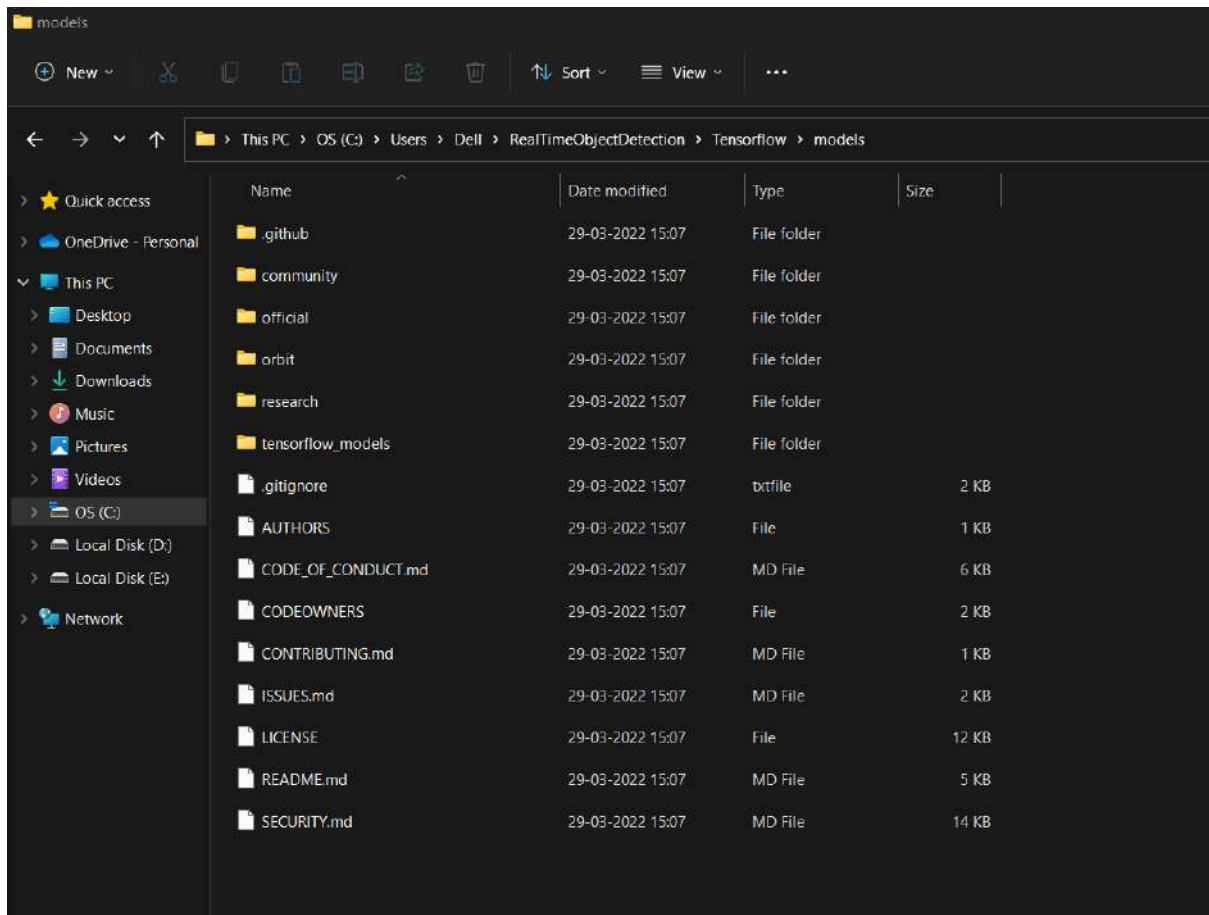
- **annotations**: This folder will be used to store all **\*.csv** files and the respective TensorFlow **\*.record** files, which contain the list of annotations for our dataset images.
- **exported-models**: This folder will be used to store exported versions of our trained model(s).
- **images**: This folder contains a copy of all the images in our dataset, as well as the respective **\*.xml** files produced for each one, once **labelImg** is used to annotate objects.
  - **images/train**: This folder contains a copy of all images, and the respective **\*.xml** files, which will be used to train our model.
  - **images/test**: This folder contains a copy of all images, and the respective **\*.xml** files, which will be used to test our model.
- **models**: This folder will contain a sub-folder for each of training job. Each subfolder will contain the training pipeline configuration file **\*.config**, as well as all files generated during the training and evaluation of our model.
- **pre-trained-models**: This folder will contain the downloaded pre-trained models, which shall be used as a starting checkpoint for our training jobs.

- **README.md**: This is an optional file which provides some general information regarding the training conditions of our model. It is not used by TensorFlow in any way, but it generally helps when you have a few training folders and/or you are revisiting a trained model after some time.

If you do not understand most of the things mentioned above, no need to worry, as we'll see how all the files are generated further down.







## **LabelImg**

### 1. What is Data labeling?

One of the most important part in the data science process is data pre-processing which makes data scientists put lots of time and effort into. Particularly, in the context of machine learning, data pre-processing requires a step of labeling. This is the step to detect and label data sample into multi-classifications so that the labeled data can be used in further machine learning process. Two typical labeling data type in ML projects are text and image annotation.

Labeling process is normally handled by manual, but it can be assisted by some applications to reduce the working-time.

### 2. What is LabelImg?

**LabelImg** is a graphical image annotation tool which provides images with bounding boxes after labeling. It is written in Python and uses Qt (one of the most common GUI for python) for its graphical interface. The output labeled data are saved as XML files in PASCAL VOC format, the format used by ImageNet.

### 3. Why do we use LabelImg?

- It is open-source then it is free to use.

- It has both online and offline versions. The offline interface is user-friendly and allows the user to make fast annotations.
- The output annotation images are saved separately as XML files along with Pascal VOC format or YOLO format. These are common formats supported by many ML packages.
- After installation, it is easy to use with simple instructions.
- It can process a large number of images when compared to other free apps such as **Cloud Vision API** (the free-offline version is limited up to only 2000 image files) or **Labelbox** (the free version only allows maximum of 5000 images to be processed).
- It can be run in multiple operating systems such as Linux, Mac or Windows.
- It is written in Python then it can interact with other python packages and is easy to be used in python applications for further development.

However, this tool still has some limitations:

- Bounding box (rectangle shape) is the only region shape to be used. Therefore, it cannot give an exactly detection for complicated objects with curved lines...
- The installation requires some tricks we need to know.

### 3. Installation.

#### 3.1. Prerequisites.

The installation below is used for **Windows OS** with **Anaconda**. Therefore, we need to have **Anaconda** to have an **Anaconda terminal**. This step we have already done.

#### 3.2. Installing LabelImg in Windows with Anaconda.

- Install **Anaconda** as described in the above session.
- Create a new environment in **Anaconda**.

We should create a new environment instead of using the base environment in **Anaconda** to avoid conflict of versions while installing the packages. Because the current **LabelImg** packages requires the older version of python ( $< 3.8.x$ ). Therefore, creating a new separate environment can avoid the error conflict when there are two versions of python in a same environment.

Do the following commands:

```
conda create -n env_name
conda activate env_name
```

E.g:

```
conda create -n labeling
conda activate labeling
```

### 3.3. Install python/tensorflow.

Please note that we only install the python version which is older than 3.8.x because the latest tensorflow version is not compatible with python version > 3.8.0.

Do the following commands:

- Install python (version < 3.8.x):

```
conda install python=3.7.0
```

- Install tensorflow:

```
conda install tensorflow
this command will install the latest version of tensorflow
```

### 3.4. Install pyqt and lxml.

These are prerequisite packages needed to be imported before running labelImg.

- Install pyqt 5:

```
conda install pyqt=5
```

- Install lxml:

```
conda install -c anaconda lxml
```

### 3.5. Install labelImg package.

- Download package github labelImg from <https://github.com/tzutalin/labelImg> and extract into folder.

- Change the current directory to the folder of **labelImg**.  
`cd path_to_labelImg_folder`

e.g:

```
cd C:\Users\sang.huynh\Documents\SANG\labelImg
```

- Do the following command:  
`pyrcc5 -o resources.py resources.qrc`

Also, to avoid manually changing the location of resources.py file, we can use the command below instead (as suggested by Jaggu):

```
pyrcc5 -o libs/resources.py resources.qrc
```

This command will create a **resource.py** file in folder **labelImg**. This file will be moved to the libs folder in the next step.

- Change the location of **resources.qrc** and **resource files** to **libs** folder (libs folder is in labelImg folder). Please note that this step is very important to have a successful installation of **labelImg**.
- Finally, do the below command:  
`python labelImg.py`

Then it opens a window of **labelImg** app. In View menu, tick Auto Save Mode — this will make the output files are automatically saved in Dir Folder. Output files are saved in XML format as default.

#### 4. Open labelImg after the first installation.

- Open anaconda terminal;

- Change the current directory to the **labellmg** folder (please remember to change the current directory path in anaconda to the **labellmg** folder when opening this tool):

```
cd path_to_labellmg_folder
```

- Do the follow command:

```
python labellmg.py
```

## **OUTLINE OF 1<sup>st</sup> Approach**

We will have to take given steps for detection of sign language.

- 1 - Import Libraries
- 2 - Opening Webcam
- 3 - Detecting Handlandmarks
- 4 - Highlighting HandLandmarks
- 5 - Coordinate Defining
- 6 - Logic Defining
- 7 - Playing Sound



# STEP 1. IMPORT LIBRARIES

How to install opencv ?

Here in this approach we have used V S Code .

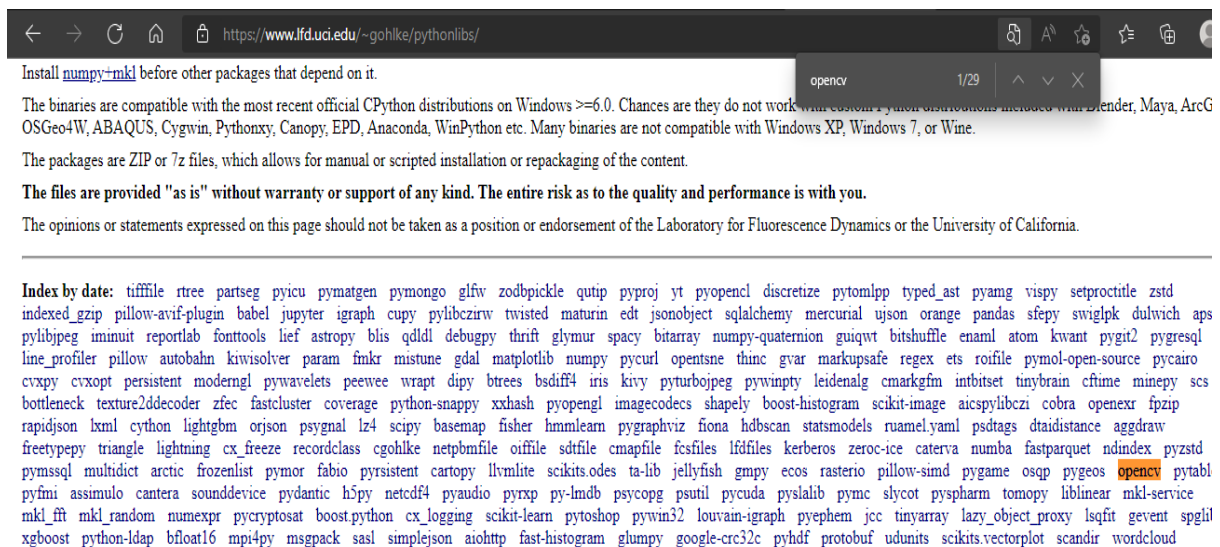
We were unable to import opencv library using pip install

In Visual studio like pycharm so we have followed this procedure

First go to <https://www.lfd.uci.edu/~gohlke/pythonlibs/>

Then click CTRL+F

Search OpenCV



← → ↻ 🏠 🔒 <https://www.lfd.uci.edu/~gohlke/pythonlibs/#opencv>

**OpenCV:** a real time computer vision library.

[opencv\\_python\\_headless-4.5.4+dummy-py3-none-any.whl](#)  
[opencv\\_python-4.5.5-pp38-pypy38\\_pp73-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5-cp310-cp310-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5-cp310-cp310-win32.whl](#)  
[opencv\\_python-4.5.5-cp39-cp39-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5-cp39-cp39-win32.whl](#)  
[opencv\\_python-4.5.5-cp38-cp38-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5-cp38-cp38-win32.whl](#)  
[opencv\\_python-4.5.5-cp37-cp37m-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5-cp37-cp37m-win32.whl](#)  
[opencv\\_python-4.5.5+mkl-cp310-cp310-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5+mkl-cp39-cp39-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5+mkl-cp38-cp38-win\\_amd64.whl](#)  
[opencv\\_python-4.5.5+mkl-cp37-cp37m-win\\_amd64.whl](#)  
[opencv\\_python-4.4.0-cp36-cp36m-win\\_amd64.whl](#)  
[opencv\\_python-4.4.0-cp36-cp36m-win32.whl](#)

Then give the location of file using `cd`

Then write the name of file or press the first word `open cv` and press `TAB`.

```
Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Keshav Sharma>cd Downloads
C:\Users\Keshav Sharma\Downloads>pip install "opencv_python-4.5.5+mkl-cp310-cp310-win_amd64.whl"
```

Then just type `import cv2` in VS code.

Now we have imported OpenCV library .

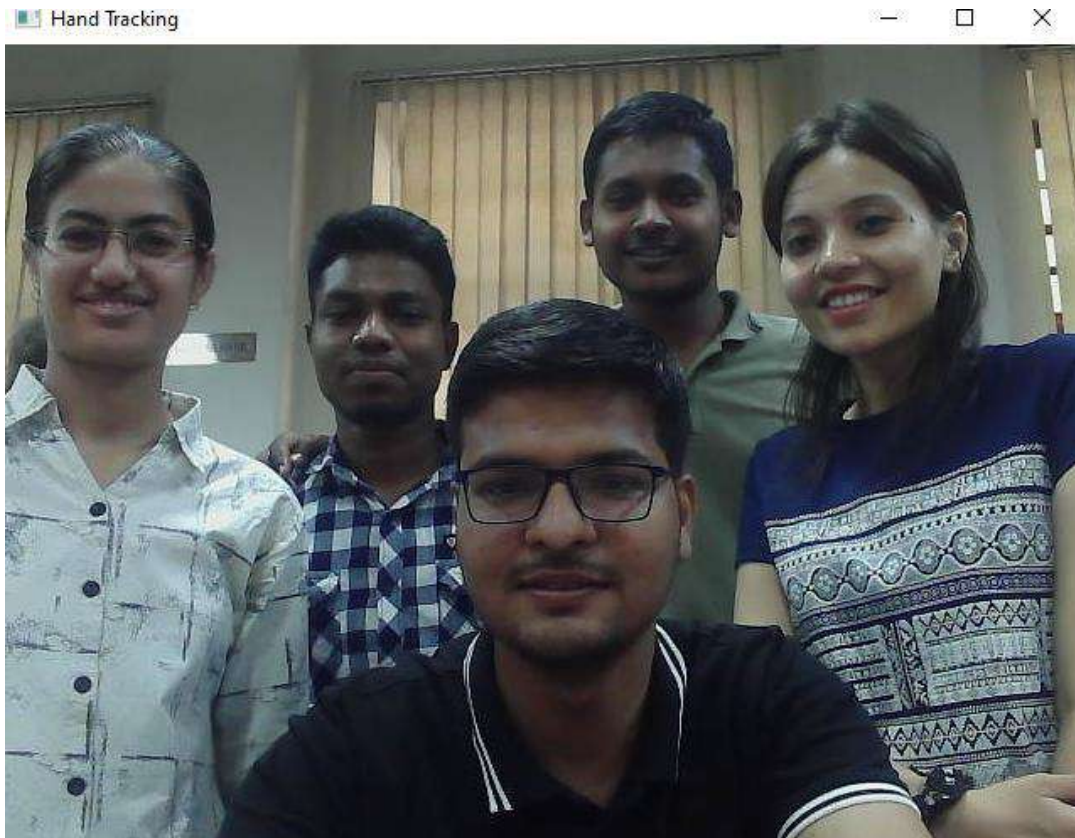
## How to install MediaPipe Library?

- I. Open the terminal of Pycharm.
- II. Type `pip install mediapipe` and press enter.

## Step 2 : Open Webcam

This enables to get live frame in which we are recognizing various signs. Only After coding for opening of webcam, we are able to further image processing to identify signs.

```
import cv2 # Import Open CV
cap = cv2.VideoCapture(0) # 0 for primary camera
while True:
 ret,img = cap.read() # For reading all the captured frames
 cv2.imshow("Hand Tracking",img) # For showing the images
 cv2.waitKey(1) # Wait for 1 second
```




## Step 3: Detecting Hand Land Marks:

In this step we are marking 21 3-D hand landmarks. This step enables to identify hand from rest of frame.

```
import cv2 # Import Open CV
import mediapipe as mp #Import Mediapipe Library

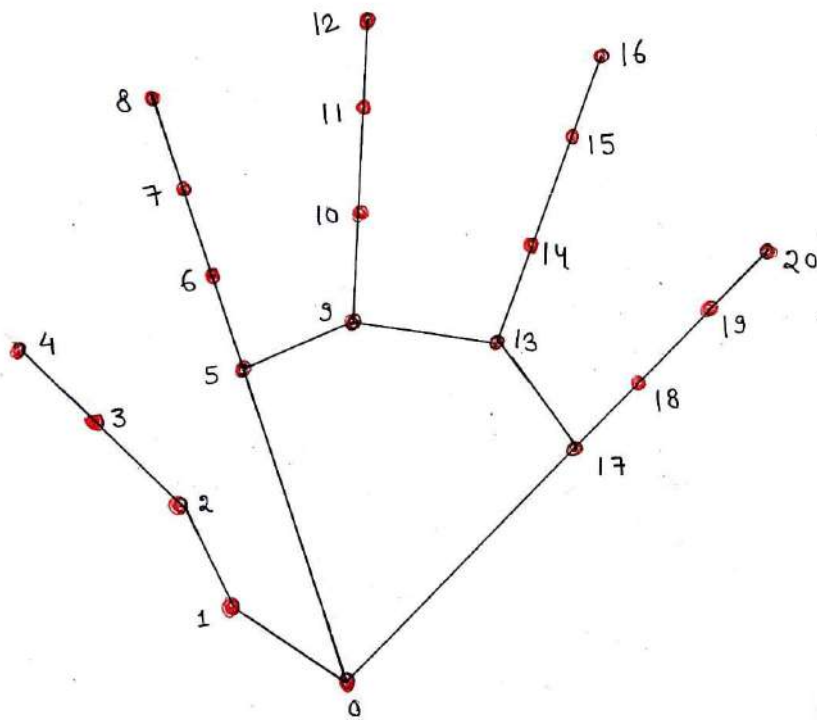
mp_hands = mp.solutions.hands
hands = mp_hands.Hands()
mp_draw = mp.solutions.drawing_utils
cap = cv2.VideoCapture(0) # 0 for primary camera
while True:
 ret,img = cap.read() # For reading all the captured frames
 results = hands.process(img)
 print(results.multi_hand_landmarks) #For printing Hand landmarks
 if results.multi_hand_landmarks:
 for hand_landmarks in results.multi_hand_landmarks:
 mp_draw.draw_landmarks(img,hand_landmarks)

 cv2.imshow("Hand Tracking",img) # For showing the images
 cv2.waitKey(1) # Wait for 1 second
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

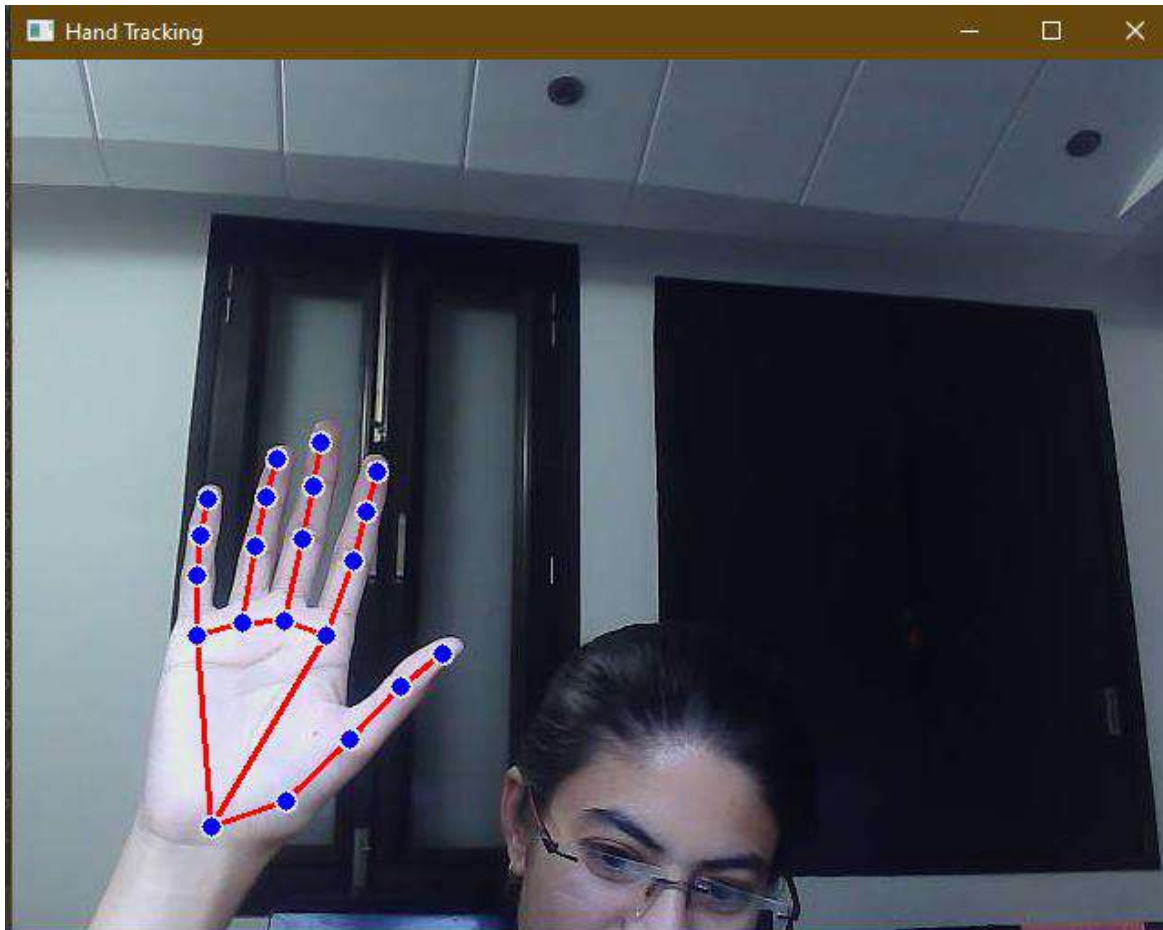
```
y: 0.8005363941192627
z: -0.07322151213884354
}
landmark {
 x: 0.6057062149047852
 y: 0.743550181388855
 z: -0.08883782476186752
}
landmark {
 x: 0.7552133798599243
 y: 0.6691367030143738
 z: -0.02432514727115631
}
landmark {
 x: 0.7080128788948059
 y: 0.5606077313423157
 z: -0.04697339981794357
}
landmark {
 x: 0.6813290119171143
 y: 0.4951696991920471
 z: -0.06743648648262024
}
landmark {
 x: 0.6618152260780334
 y: 0.43604692816734314
 z: -0.08407513052225113
}
landmark {
```



- |                      |                      |
|----------------------|----------------------|
| 0 WRIST              | 11 MIDDLE FINGER DIP |
| 1 THUMB CMC          | 12 MIDDLE FINGER TIP |
| 2 THUMB MCP          | 13 RING FINGER MCP   |
| 3 THUMB IP           | 14 RING FINGER PIP   |
| 4 THUMB TIP          | 15 RING FINGER DIP   |
| 5 INDEX FINGER MCP   | 16 RING FINGER TIP   |
| 6 INDEX FINGER PIP   | 17 PINKY MCP         |
| 7 INDEX FINGER DIP   | 18 PINKY PIP         |
| 8 INDEX FINGER TIP   | 19 PINKY DIP         |
| 9 MIDDLE FINGER MCP  | 20 PINKY TIP         |
| 10 MIDDLE FINGER PIP |                      |

## Step 4: Highlighting Hand Landmarks:

Here we highlight each landmark with a circle of defined radius and all landmarks are connected with straight lines, looking like a bone like structure of hand.



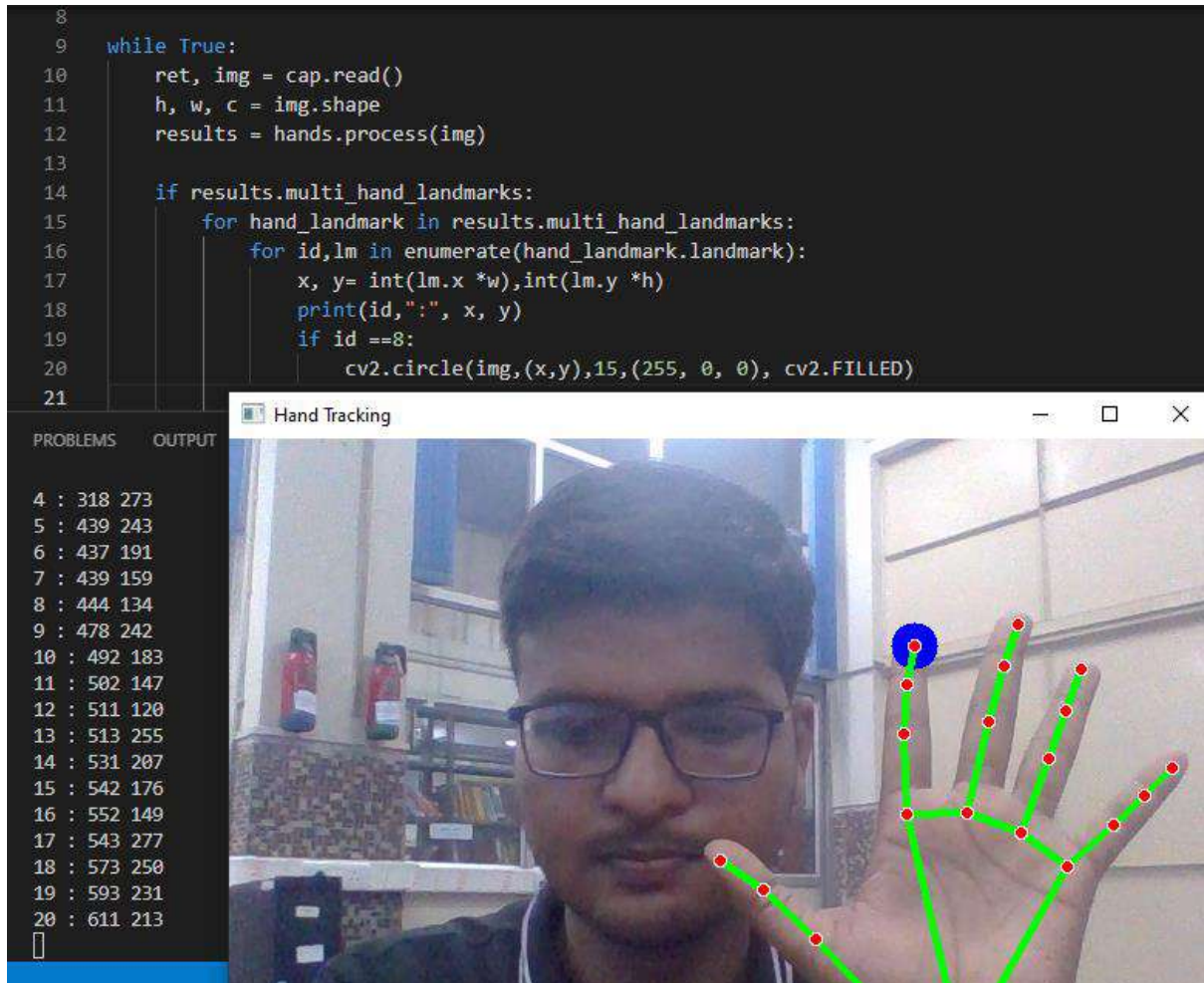
Just change the last third line from this to this.

```
mp_draw.draw_landmarks(img,hand_landmarks,mp_hands.HAND_CONNECTIONS,mp_draw.DrawingSpec((255,0,0),5,2),
mp_draw.DrawingSpec((0,0,255),5,2)) # Just for color , radius and thickness
```

## Step 5: Coordinates defining:



In this step we define coordinate of each landmark by multiplying with width and height. This step enables us to apply some logic on landmarks to define a sign.



```

8
9 while True:
10 ret, img = cap.read()
11 h, w, c = img.shape
12 results = hands.process(img)
13
14 if results.multi_hand_landmarks:
15 for hand_landmark in results.multi_hand_landmarks:
16 for id, lm in enumerate(hand_landmark.landmark):
17 x, y = int(lm.x * w), int(lm.y * h)
18 print(id, ":", x, y)
19 if id == 8:
20 cv2.circle(img, (x, y), 15, (255, 0, 0), cv2.FILLED)
21

```

PROBLEMS OUTPUT

```

4 : 318 273
5 : 439 243
6 : 437 191
7 : 439 159
8 : 444 134
9 : 478 242
10 : 492 183
11 : 502 147
12 : 511 120
13 : 513 255
14 : 531 207
15 : 542 176
16 : 552 149
17 : 543 277
18 : 573 250
19 : 593 231
20 : 611 213

```

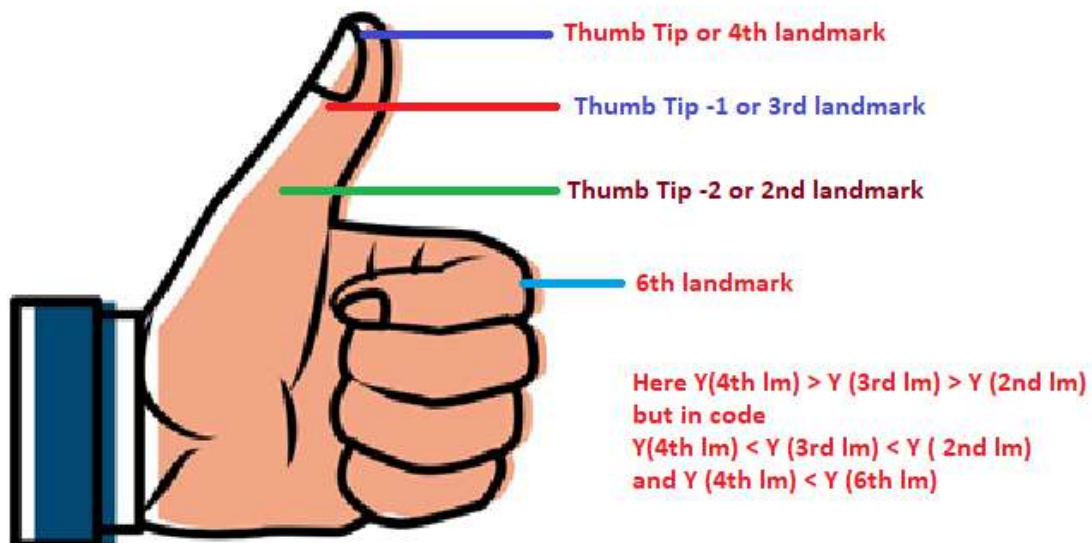
**Step 6: Logic defining:** here we give certain conditions to landmarks so that in real time frame, satisfying the conditions on landmarks, it displays that signs on frame itself.

## FOR LIKE SIGN

For like sign I can see here that y co-ordinate of thumb TIP (4<sup>th</sup> landmark) Should be greater then Thumb IP (3<sup>rd</sup> landmark) And Thumb IP(3<sup>rd</sup> landmark) will be greater then Thumb MCP (2<sup>nd</sup> landmark).

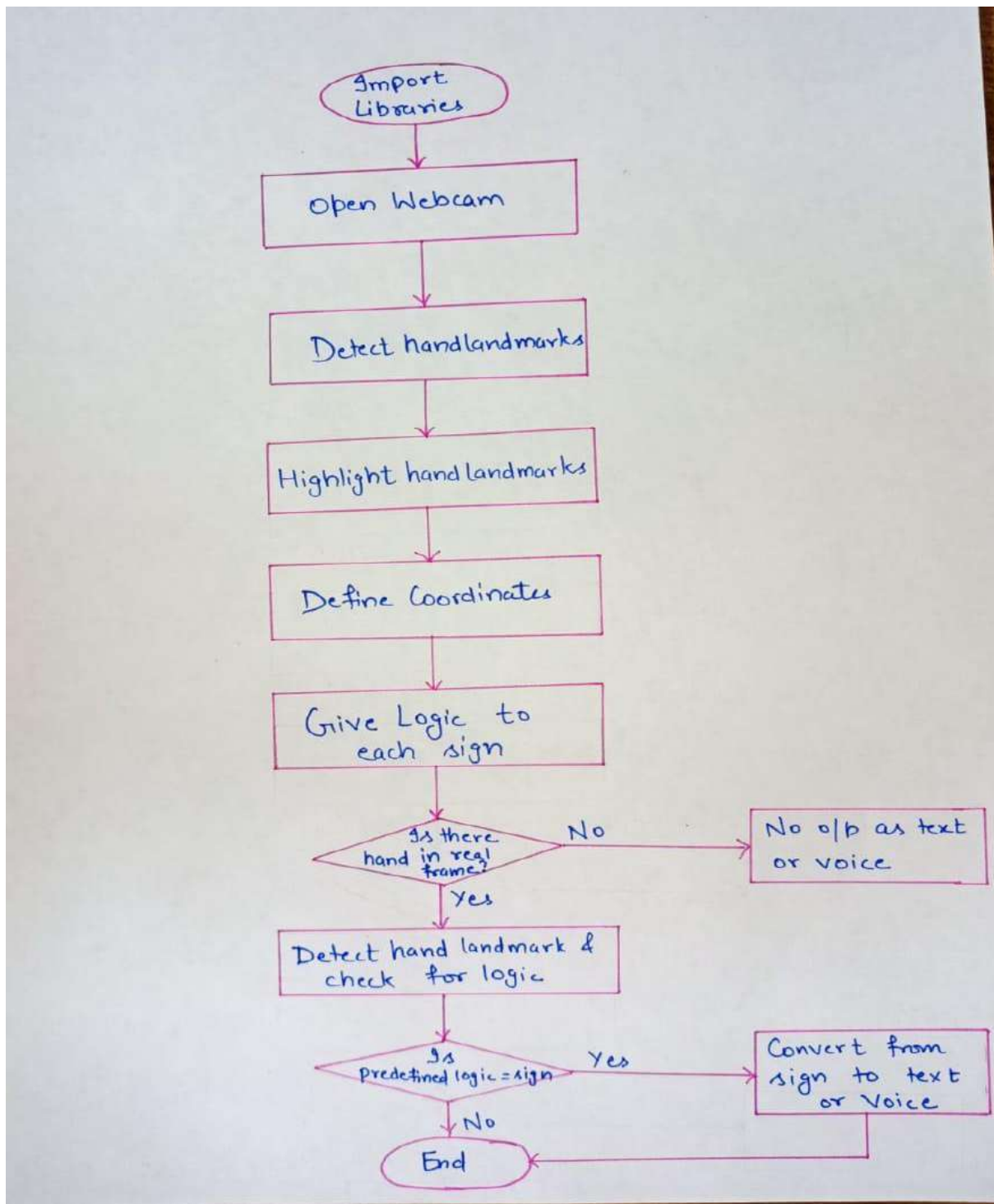
But In our programme we have used flip function in 22 line which is rotating images by 180 degrees. So Here we will have to use Y(Thumb tip) or (4<sup>th</sup> point) < Y (Thumb tip -1) or (3<sup>rd</sup> point) < Y (thumb tip -2) or (2<sup>nd</sup> point) or we can also

see in figure that In figure that  $Y$  (Thumb tip ) or  $Y(4^{th} \text{ landmark}) < Y$  (Index finger PIP) or  $Y(6^{th} \text{ landmark})$ .





# Flowchart for approach-1



```

1 # SIGN LANGUAGE RECOGNIZATION USING PYTHON AND OPENCV (MACHINE LEARNING)
2 # GROUP 2 (BARDEEN)
3 # ADVANCED ELECTRONICS LAB 2
4 # -----
5
6 from locale import LC_MONETARY
7 import cv2 # IMPORT OPENCV
8 import mediapipe as mp # IMPORTING MEDIAPIPE LIBRARY
9 from playsound import playsound
10 import os # IMPORTING OPERATING SYSTEM
11
12 mp_hands = mp.solutions.hands
13 hands = mp_hands.Hands()
14 mp_draw = mp.solutions.drawing_utils
15 cap = cv2.VideoCapture(0)
16
17 finger_tips = [8 ,12 ,16, 20]
18 thumb_tip =4
19
20 while True:
21 ret, img = cap.read()
22 img = cv2.flip(img, 1)
23 h, w, c = img.shape
24 results = hands.process(img)
25
26 if results.multi_hand_landmarks:
27 for hand_landmark in results.multi_hand_landmarks:
28 lm_list=[]
29 for id,lm in enumerate(hand_landmark.landmark):
30 lm_list.append(lm)
31 finger_fold_status=[]
32 for tip in finger_tips:
33 x, y= int(lm_list[tip].x *w),int(lm_list[tip].y *h)
34 #print(id,":", x, y)
35 cv2.circle(img,(x,y),15,(255, 0, 0), cv2.FILLED)
36
37 if lm_list[tip].x < lm_list[tip -3].x:
38 cv2.circle(img,(x,y),15,(0, 255, 0), cv2.FILLED)
39 finger_fold_status.append(True)
40 else:
41 finger_fold_status.append(False)
42
43 #print(finger_fold_status)
44
45 if all(finger_fold_status):
46
47 # LIKE
48
49 if lm_list[thumb_tip].y < lm_list[thumb_tip - 1].y <
lm_list[thumb_tip - 2].y \
50 and lm_list[4].y < lm_list[6].y:
51 # print("LIKE")
52 cv2.putText(img, "LIKE", (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 3)
53
54
55 # DISLIKE
56

```

```

57 if lm_list[thumb_tip].y > lm_list[thumb_tip - 1].y >
lm_list[thumb_tip - 2].y \
58 and lm_list[4].y > lm_list[6].y:
59 cv2.putText(img, "DISLIKE", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, (0, 0, 255), 3)
60 #print("DISLIKE")
61
62
63 # HELLO
64
65 if lm_list[4].y < lm_list[2].y and lm_list[8].y < lm_list[6].y and
lm_list[12].y < lm_list[10].y and \
66 lm_list[16].y < lm_list[14].y and lm_list[20].y < lm_list[18].y \
67 and lm_list[4].x > lm_list[8].x > lm_list[12].x > lm_list[16].x >
lm_list[20].x:
68 cv2.putText(img, "HELLO", (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 3)
69 #playsound
70
71 # THANKYOU
72
73 if lm_list[20].y > lm_list[16].y > lm_list[12].y > lm_list[8].y and
lm_list[4].y < lm_list[2].y \
74 and lm_list[4].x < lm_list[20].x \
75 and lm_list[12].x > lm_list[10].x and lm_list[8].x > lm_list[6].x
and lm_list[14].x < lm_list[16].x \
76 and lm_list[20].x > lm_list[18].x:
77 cv2.putText(img, "THANKYOU", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 0, 0), 3)
78
79 # DOWN
80
81 if lm_list[3].x > lm_list[4].x and lm_list[3].y < lm_list[4].y and
lm_list[6].y < lm_list[8].y and \
82 lm_list[16].y < lm_list[14].y and lm_list[20].y < lm_list[18].y:
83 #and lm_list[17].x > lm_list[13].x > lm_list[9].x > lm_list[5].x:
84 cv2.putText(img, "DOWN", (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 3)
85
86
87 # VICTORY
88
89 if lm_list[3].x > lm_list[4].x and lm_list[8].y < lm_list[6].y and
lm_list[12].y < lm_list[10].y and \
90 lm_list[16].y > lm_list[14].y and lm_list[20].y > lm_list[18].y \
91 and lm_list[8].x > lm_list[12].x:
92 cv2.putText(img, "VICTORY", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 0, 0), 3)
93
94
95 # FANTASTIC
96
97 if lm_list[3].x > lm_list[4].x and lm_list[8].y < lm_list[6].y and
lm_list[12].y > lm_list[10].y and \
98 lm_list[16].y > lm_list[14].y and lm_list[20].y < lm_list[18].y:
99 cv2.putText(img, "FANTASTIC ", (20, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 3)
100
101 # ROCK
102

```

```

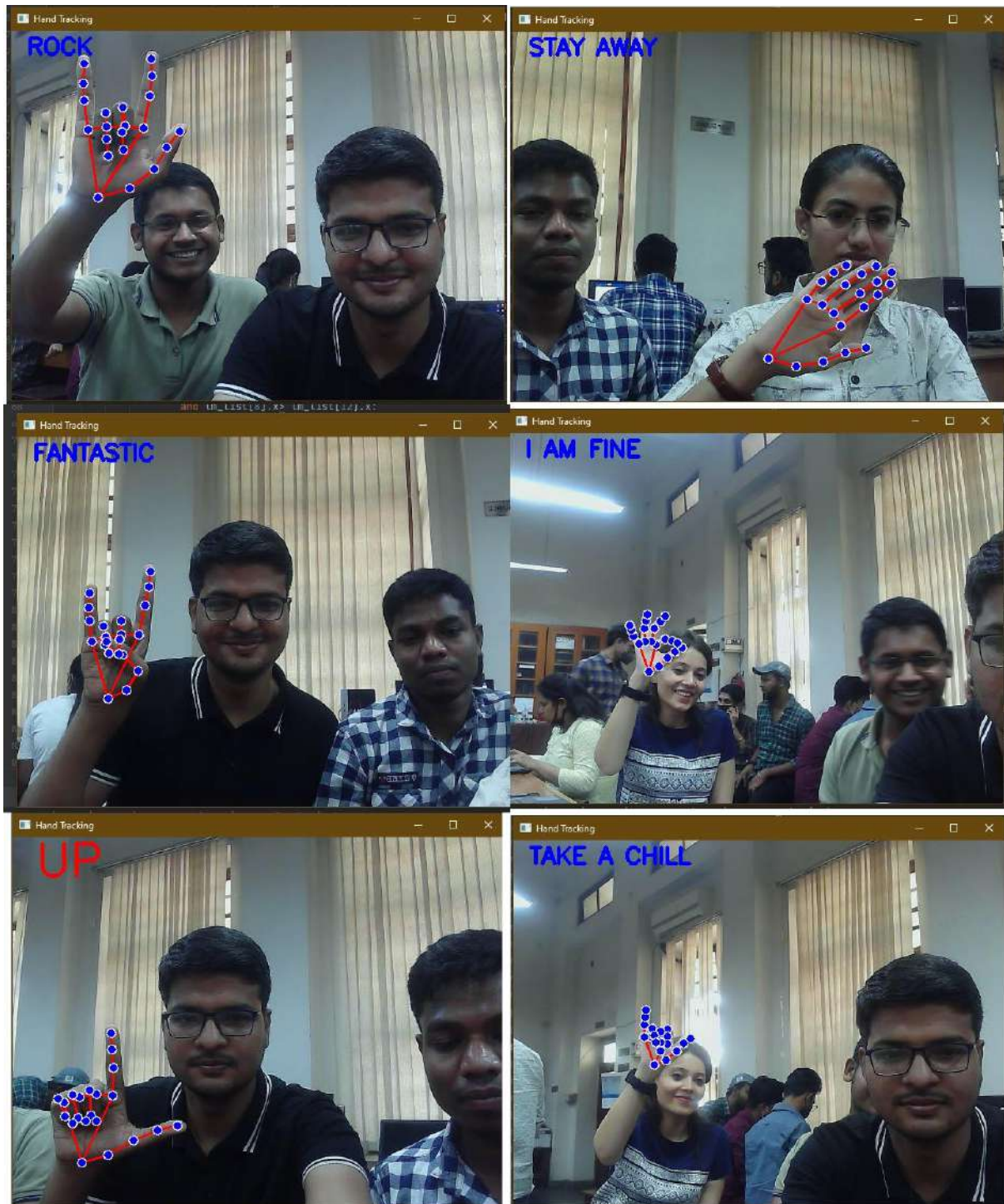
103 if lm_list[3].x < lm_list[4].x and lm_list[8].y<lm_list[6].y and
lm_list[12].y>lm_list[10].y and \
104 lm_list[16].y > lm_list[14].y and lm_list[20].y<lm_list[18].y:
105 cv2.putText(img, "ROCK", (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 3)
106
107
108 # I AM FINE
109
110 if lm_list[12].x > lm_list[16].x > lm_list[20].x and lm_list[8].y<
lm_list[4].y \
111 and lm_list[12].y<lm_list[10].y and lm_list[16].y< lm_list[14].y\
112 and lm_list[20].y< lm_list[18].y and lm_list[6].y< lm_list[8].y \
113 and lm_list[4].y< lm_list[2].y:
114 cv2.putText(img, "I AM FINE", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 0, 0), 3)
115 #playsound('C:/Users/lenovo/Downloads/Sound/IAMFINE.mp4')
116
117 # TAKE A CHILL
118
119 if lm_list[8].y>lm_list[6].y and lm_list[12].y>lm_list[10].y and
lm_list[4].y<lm_list[2].y \
120 and lm_list[4].x>lm_list[20].x and lm_list[16].y>lm_list[14].y
and lm_list[20].y<lm_list[18].y:
121 cv2.putText(img, "TAKE A CHILL", (20, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 3)
122
123 # HOW
124
125 if lm_list[2].x > lm_list[4].x and lm_list[8].y < lm_list[5].y and
lm_list[12].y > lm_list[10].y and \
126 lm_list[16].y > lm_list[14].y and lm_list[20].y >
lm_list[18].y\
127 and lm_list[4].x<lm_list[5].x<lm_list[9].x<lm_list[17].x:
128 cv2.putText(img, "HOW ARE YOU", (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 3)
129
130 # UP
131
132 if lm_list[0].x < lm_list[4].x and lm_list[8].y < lm_list[5].y and
lm_list[12].y > lm_list[10].y and \
133 lm_list[16].y > lm_list[14].y and lm_list[20].y >
lm_list[18].y and lm_list[4].x> lm_list[2].x:
134 cv2.putText(img, "UP", (30, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,
0, 255), 3)
135
136 # RIGHT
137
138 if lm_list[4].x > lm_list[0].x and lm_list[8].y > lm_list[5].y and
lm_list[12].y > lm_list[9].y and \
139 lm_list[16].y > lm_list[13].y and lm_list[20].y >
lm_list[17].y\
140 and lm_list[6].y<lm_list[4].y and
lm_list[5].x>lm_list[13].x>lm_list[17].x:
141 cv2.putText(img, "TAKE RIGHT", (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 4)
142
143 # LEFT
144

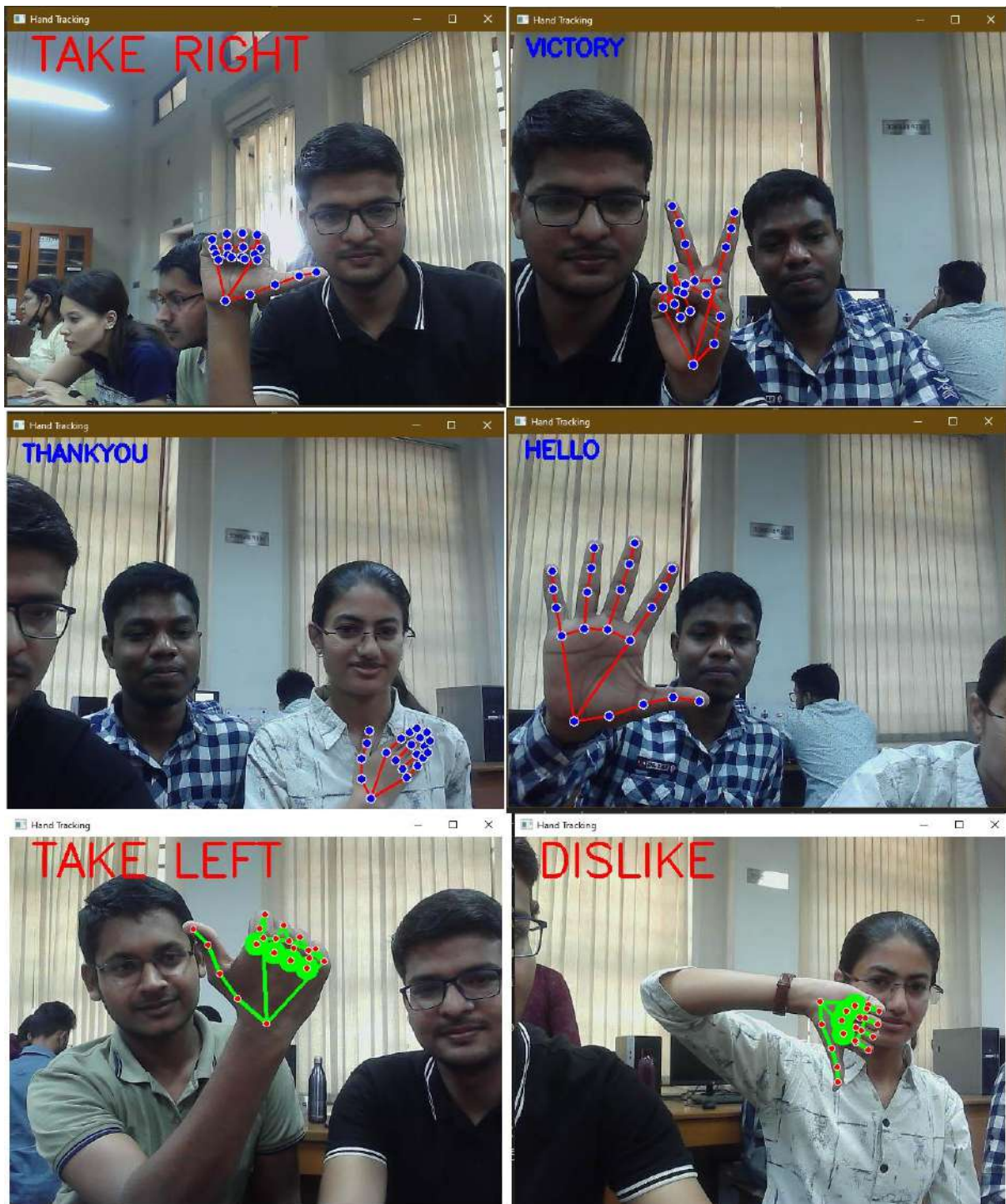
```

```
145 if lm_list[4].x < lm_list[0].x and lm_list[8].y > lm_list[5].y and
lm_list[12].y > lm_list[9].y and \
146 lm_list[16].y > lm_list[13].y and lm_list[20].y >
lm_list[17].y:
147 cv2.putText(img, "TAKE LEFT", (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 4)
148
149 # STAY AWAY
150
151 if lm_list[4].y > lm_list[8].y > lm_list[12].y >
lm_list[16].y > lm_list[20].y \
152 and lm_list[4].x > lm_list[20].x:
153 cv2.putText(img, "STAY AWAY", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 0, 0), 3)
154
155
156 mp_draw.draw_landmarks(img, hand_landmark,
157 mp_hands.HAND_CONNECTIONS,
158 mp_draw.DrawingSpec((0,0,255),2,2),
159 mp_draw.DrawingSpec((0,255,0),4,2)
160)
161
162 cv2.imshow("Hand Tracking",img)
163 cv2.waitKey(1)
164
165
166
```

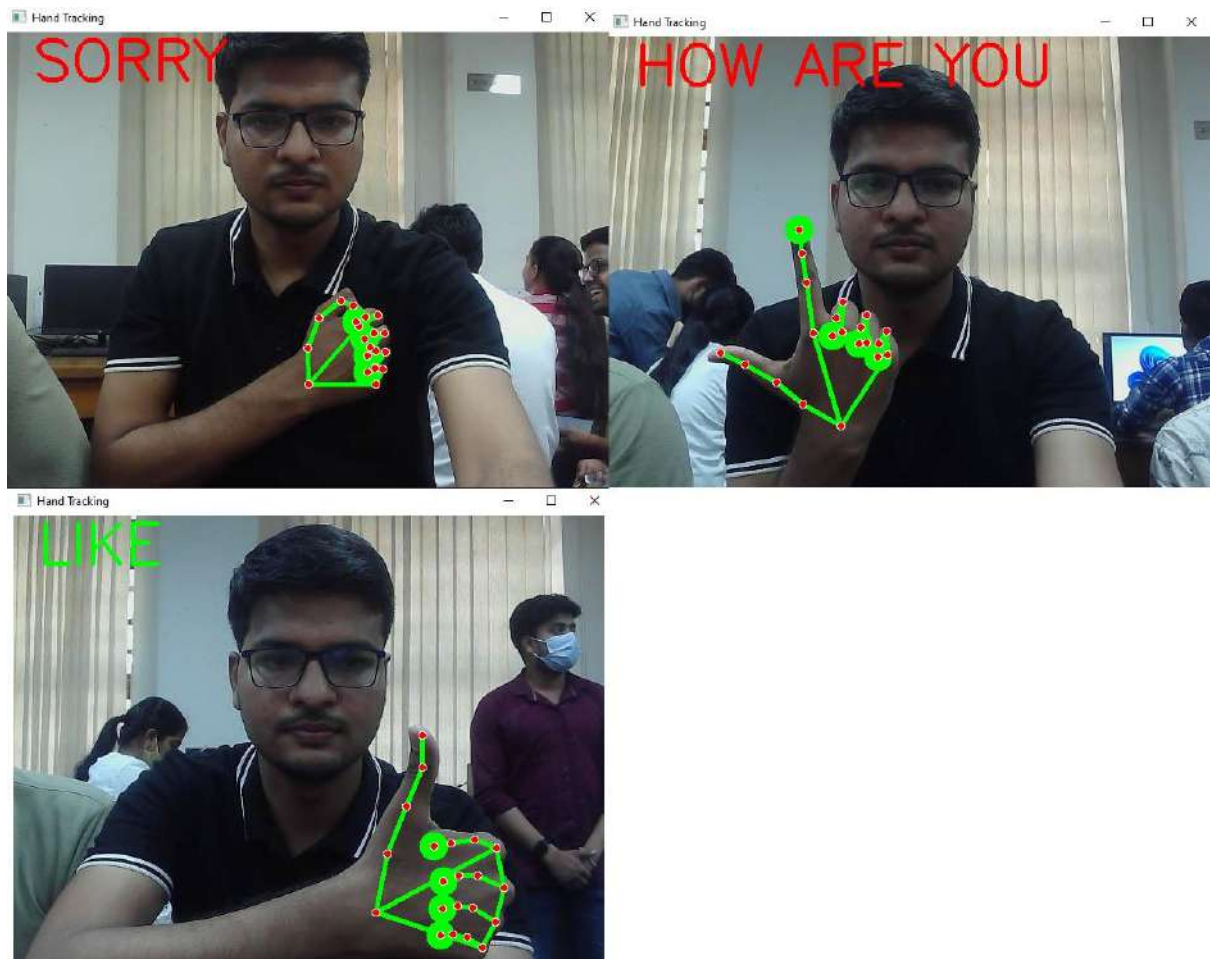


# DETECTION OF DIFFERENT SIGNS











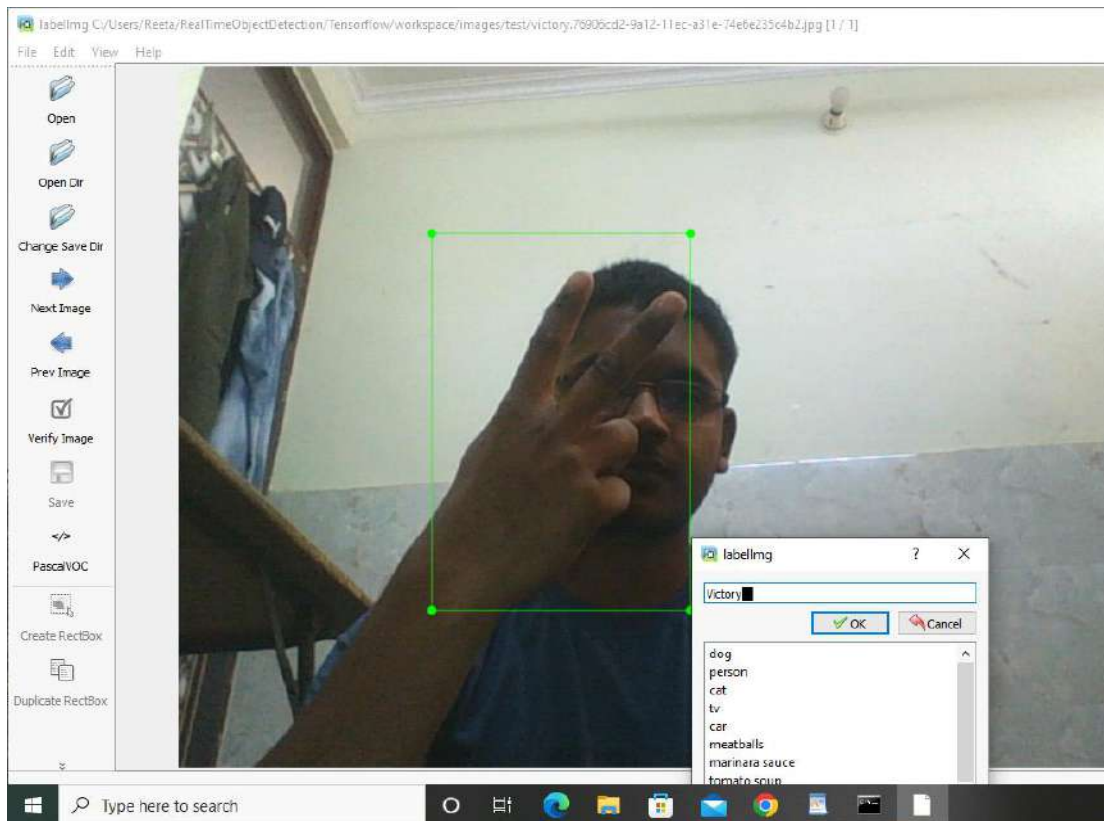
## APPROACH 2

In our approach 2, *Sign Language Detection By Using Tensorflow Object Detection API*, we are collecting images for our sign using *python* and *opencv* and label them using *labellmg* package and then using them to *train* the *tensorflow object detection api* for our sign language and then detecting sign language in *Real time*.

Firstly, we command our web camera to collect the images for sign language by using *opencv* and *python*.



These images will be used for training purposes by passing them through label image and drawing detection boxes against each sign language poses.



Next using transfer learning against at tensorflow object detection api.

Finally Real Time detection using webcam and opencv.

## Steps followed

1.) **Cloning of Repository** : cloning of RealTimeOBjectDetection repository will provide us leverage to all that transfer learning and pre-processing and training code inside it.

open your command prompt and type the link (<https://github.com/nicknochnack/RealTimeObjectDetection>) with git clone command .

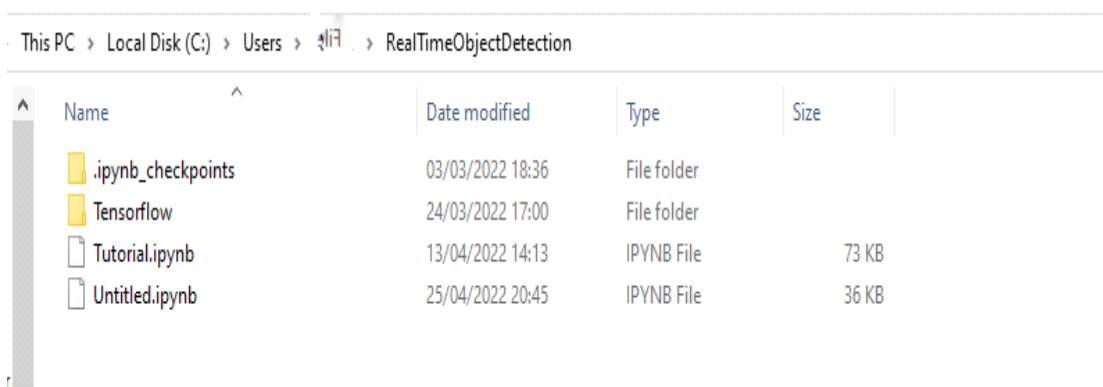
```
Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\RON\>git clone https://github/nicknochnack/RealTimeObjectDetection
fatal: destination path 'RealTimeObjectDetection' already exists and is not an empty directory.

C:\Users\git>git clone https://github/nicknochnack/RealTimeObjectDetection
Cloning into 'RealTimeObjectDetection'...
```

It will clone the repository in C:\Users\git>  
and the following files will clone into your PC  
as path prescribed .

*C:\Users\git\RealTimeObjectDetection*

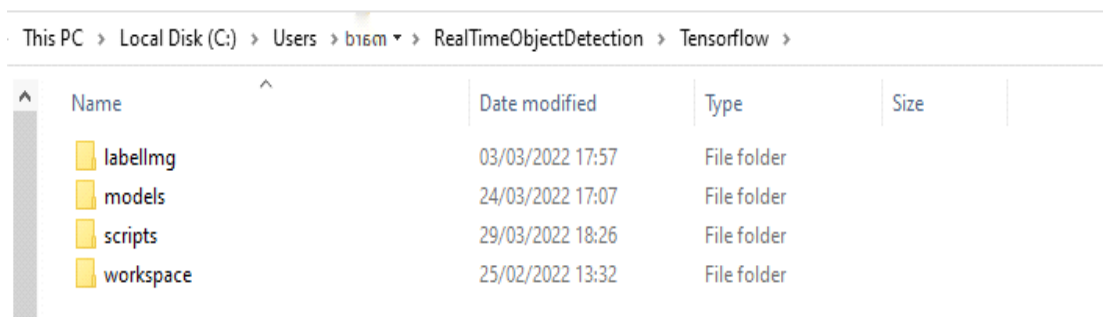


This PC > Local Disk (C:) > Users > git > RealTimeObjectDetection

| Name               | Date modified    | Type        | Size  |
|--------------------|------------------|-------------|-------|
| .ipynb_checkpoints | 03/03/2022 18:36 | File folder |       |
| Tensorflow         | 24/03/2022 17:00 | File folder |       |
| Tutorial.ipynb     | 13/04/2022 14:13 | IPYNB File  | 73 KB |
| Untitled.ipynb     | 25/04/2022 20:45 | IPYNB File  | 36 KB |

In Tensorflow folder

*C:\Users\git\RealTimeObjectDetection\Tensorflow*



This PC > Local Disk (C:) > Users > b16m > RealTimeObjectDetection > Tensorflow

| Name      | Date modified    | Type        | Size |
|-----------|------------------|-------------|------|
| labellmg  | 03/03/2022 17:57 | File folder |      |
| models    | 24/03/2022 17:07 | File folder |      |
| scripts   | 29/03/2022 18:26 | File folder |      |
| workspace | 25/02/2022 13:32 | File folder |      |

In workspace folder

*C:\Users\git\RealTimeObjectDetection\Tensorflow\workspace*

| This PC > Local Disk (C:) > Users > ame > RealTimeObjectDetection > Tensorflow > workspace |                  |             |      |  |
|--------------------------------------------------------------------------------------------|------------------|-------------|------|--|
| Name                                                                                       | Date modified    | Type        | Size |  |
| annotations                                                                                | 29/03/2022 17:49 | File folder |      |  |
| images                                                                                     | 01/03/2022 23:03 | File folder |      |  |
| models                                                                                     | 24/03/2022 17:11 | File folder |      |  |
| pre-trained-models                                                                         | 25/02/2022 13:32 | File folder |      |  |

In pre-trained-models folder

C:\Users\git\RealTimeObjectDetection\Tensorflow\workspace\pre-trained-models

| This PC > Local Disk (C:) > Users > Nam > RealTimeObjectDetection > Tensorflow > workspace > pre-trained-models > |                  |                |           |  |
|-------------------------------------------------------------------------------------------------------------------|------------------|----------------|-----------|--|
| Name                                                                                                              | Date modified    | Type           | Size      |  |
| ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8                                                                     | 25/02/2022 13:32 | File folder    |           |  |
| ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar                                                                 | 25/02/2022 13:32 | WinRAR archive | 20,035 KB |  |

In ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8

C:\Users\Reeta\RealTimeObjectDetection\Tensorflow\workspace\pre-trained-models\ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8

| Local Disk (C:) > Users > aoard > RealTimeObjectDetection > Tensorflow > workspace > pre-trained-models > ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8 > |                  |             |      |  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|-------------|------|--|
| Name                                                                                                                                                      | Date modified    | Type        | Size |  |
| checkpoint                                                                                                                                                | 25/02/2022 13:32 | File folder |      |  |
| saved_model                                                                                                                                               | 25/02/2022 13:32 | File folder |      |  |
| pipeline                                                                                                                                                  | 25/02/2022 13:32 | CONFIG File | 5 KB |  |

and the pipeline file contains the code

```
model {
 ssd {
 num_classes: 2
 image_resizer {
```

```

fixed_shape_resizer {
 height: 320
 width: 320
}
}
feature_extractor {
 type: "ssd_mobilenet_v2_fpn_keras"
 depth_multiplier: 1.0
 min_depth: 16
 conv_hyperparams {
 regularizer {
 l2_regularizer {
 weight: 3.9999998989515007e-05
 }
 }
 }
 initializer {
 random_normal_initializer {
 mean: 0.0
 stddev: 0.009999999776482582
 }
 }
 activation: RELU_6
 batch_norm {
 decay: 0.996999979019165
 }
}

```

```

 scale: true
 epsilon: 0.0010000000474974513
 }
}
use_depthwise: true
override_base_feature_extractor_hyperparams: true
fpn {
 min_level: 3
 max_level: 7
 additional_layer_depth: 128
}
}
box_coder {
 faster_rcnn_box_coder {
 y_scale: 10.0
 x_scale: 10.0
 height_scale: 5.0
 width_scale: 5.0
 }
}
matcher {
 argmax_matcher {
 matched_threshold: 0.5
 unmatched_threshold: 0.5
 }
}

```

```

ignore_thresholds: false
negatives_lower_than_unmatched: true
force_match_for_each_row: true
use_matmul_gather: true
}
}
similarity_calculator {
 iou_similarity {
 }
}
box_predictor {
 weight_shared_convolutional_box_predictor {
 conv_hyperparams {
 regularizer {
 l2_regularizer {
 weight: 3.9999998989515007e-05
 }
 }
 initializer {
 random_normal_initializer {
 mean: 0.0
 stddev: 0.009999999776482582
 }
 }
 }
 }
}

```

```

activation: RELU_6
batch_norm {
 decay: 0.996999979019165
 scale: true
 epsilon: 0.0010000000474974513
}
}
depth: 128
num_layers_before_predictor: 4
kernel_size: 3
class_prediction_bias_init: -4.599999904632568
share_prediction_tower: true
use_depthwise: true
}
}
anchor_generator {
 multiscale_anchor_generator {
 min_level: 3
 max_level: 7
 anchor_scale: 4.0
 aspect_ratios: 1.0
 aspect_ratios: 2.0
 aspect_ratios: 0.5
 scales_per_octave: 2
 }
}

```



```

 }
 }
 post_processing {
 batch_non_max_suppression {
 score_threshold: 9.99999993922529e-09
 iou_threshold: 0.6000000238418579
 max_detections_per_class: 100
 max_total_detections: 100
 use_static_shapes: false
 }
 score_converter: SIGMOID
 }
 normalize_loss_by_num_matches: true
 loss {
 localization_loss {
 weighted_smooth_l1 {
 }
 }
 }
 classification_loss {
 weighted_sigmoid_focal {
 gamma: 2.0
 alpha: 0.25
 }
 }
}

```

```

 classification_weight: 1.0
 localization_weight: 1.0
 }
 encode_background_as_zeros: true
 normalize_loc_loss_by_codesize: true
 inplace_batchnorm_update: true
 freeze_batchnorm: false
}
}
train_config {
 batch_size: 128
 data_augmentation_options {
 random_horizontal_flip {
 }
 }
 data_augmentation_options {
 random_crop_image {
 min_object_covered: 0.0
 min_aspect_ratio: 0.75
 max_aspect_ratio: 3.0
 min_area: 0.75
 max_area: 1.0
 overlap_thresh: 0.0
 }
 }
}

```

```

}
sync_replicas: true
optimizer {
 momentum_optimizer {
 learning_rate {
 cosine_decay_learning_rate {
 learning_rate_base: 0.07999999821186066
 total_steps: 50000
 warmup_learning_rate: 0.026666000485420227
 warmup_steps: 1000
 }
 }
 momentum_optimizer_value: 0.8999999761581421
 }
 use_moving_average: false
}
fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
num_steps: 50000
startup_delay_steps: 0.0
replicas_to_aggregate: 8
max_number_of_boxes: 100
unpad_groundtruth_tensors: false
fine_tune_checkpoint_type: "classification"
fine_tune_checkpoint_version: V2

```

```

}
train_input_reader {
 label_map_path: "PATH_TO_BE_CONFIGURED"
 tf_record_input_reader {
 input_path: "PATH_TO_BE_CONFIGURED"
 }
}
eval_config {
 metrics_set: "coco_detection_metrics"
 use_moving_averages: false
}
eval_input_reader {
 label_map_path: "PATH_TO_BE_CONFIGURED"
 shuffle: false
 num_epochs: 1
 tf_record_input_reader {
 input_path: "PATH_TO_BE_CONFIGURED"
 }
}

```

## **2.)Collecting images**

we will collect images by using webcamra with the help of opencv and python. and we will write the following code in jupyter notebook.

```
Collecting opencv-python
 Downloading opencv_python-4.5.5.64-cp36-abi3-win_amd64.whl (35.4 MB)
Requirement already satisfied: numpy>=1.19.3 in c:\users\... \anaconda3\envs\tensorflow_env\lib\site-packages (from opencv-python) (1.21.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.5.64
```

```
In [6]: IMAGES_PATH = 'Tensorflow/workspace/images/collectedimages'
```

In [ ]:

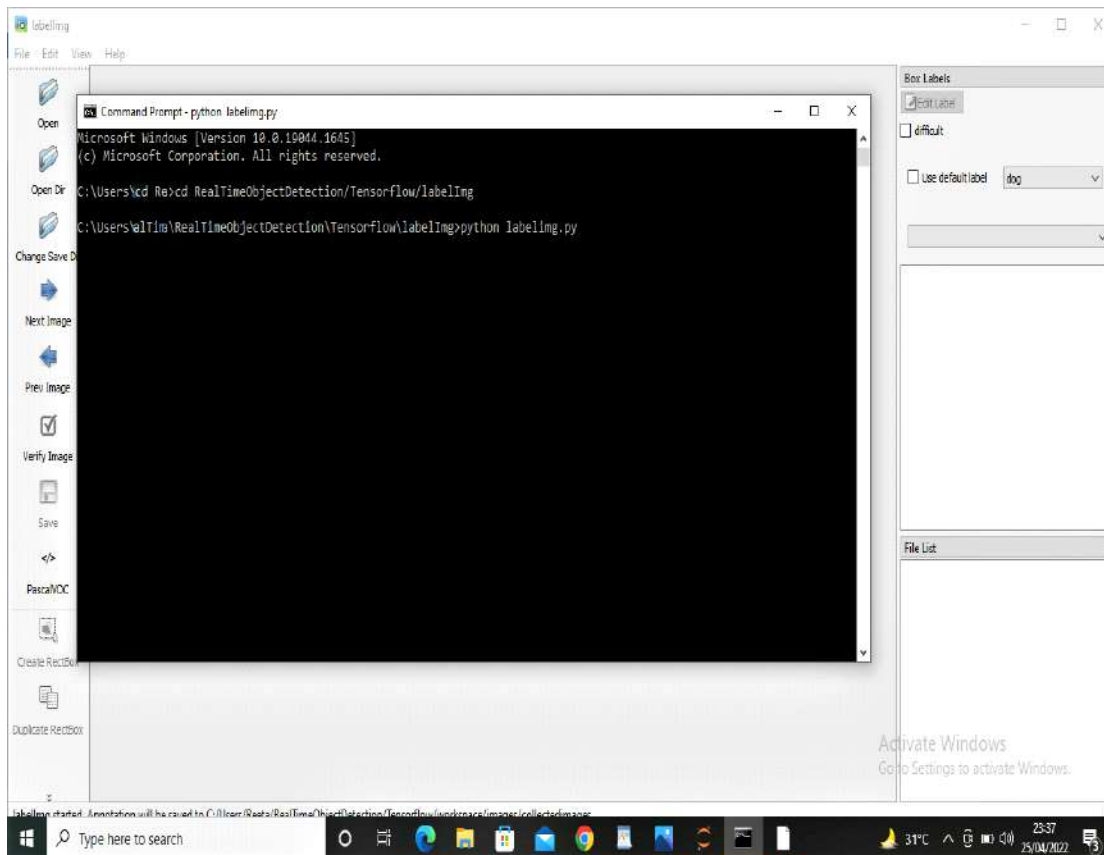
```
A subdirectory or file Tensorflow\workspace\images\collectedImages\namaste already exists.
Collecting images for namaste

A subdirectory or file Tensorflow\workspace\images\collectedImages\victory already exists.
Collecting images for victory
```

```
Requirement already satisfied: pyqt5 in c:\users\w_em\anaconda3\envs\tensorflow_env\lib\site-packages (5.15.6)
Requirement already satisfied: PyQt5-sip<13, >=12.8 in c:\users\w_em\anaconda3\envs\tensorflow_env\lib\site-packages (from pyqt5) (12.10.1)
Requirement already satisfied: PyQt5-Qt5==5.15.2 in c:\users\w_em\anaconda3\envs\tensorflow_env\lib\site-packages (from pyqt5) (5.15.2)
```

```
Requirement already satisfied: tensorflow in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (2.6.0)
Requirement already satisfied: keras-preprocessing>=1.1.2 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: keras>=2.4.0 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: absl-py==0.10 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (0.10.0)
Requirement already satisfied: typing-extensions>=3.7.4 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (3.10.0.2)
Requirement already satisfied: tensorflow-estimator==2.6 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (2.6.0)
Requirement already satisfied: h5py>=3.1.0 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: flatbuffers==1.12 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (1.12)
Requirement already satisfied: opt-einsum==3.3.0 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: wrapt>=1.11.2 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow) (1.13.3)
Requirement already satisfied: gast==0.4.0 in c:\users\7.4a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow)
```



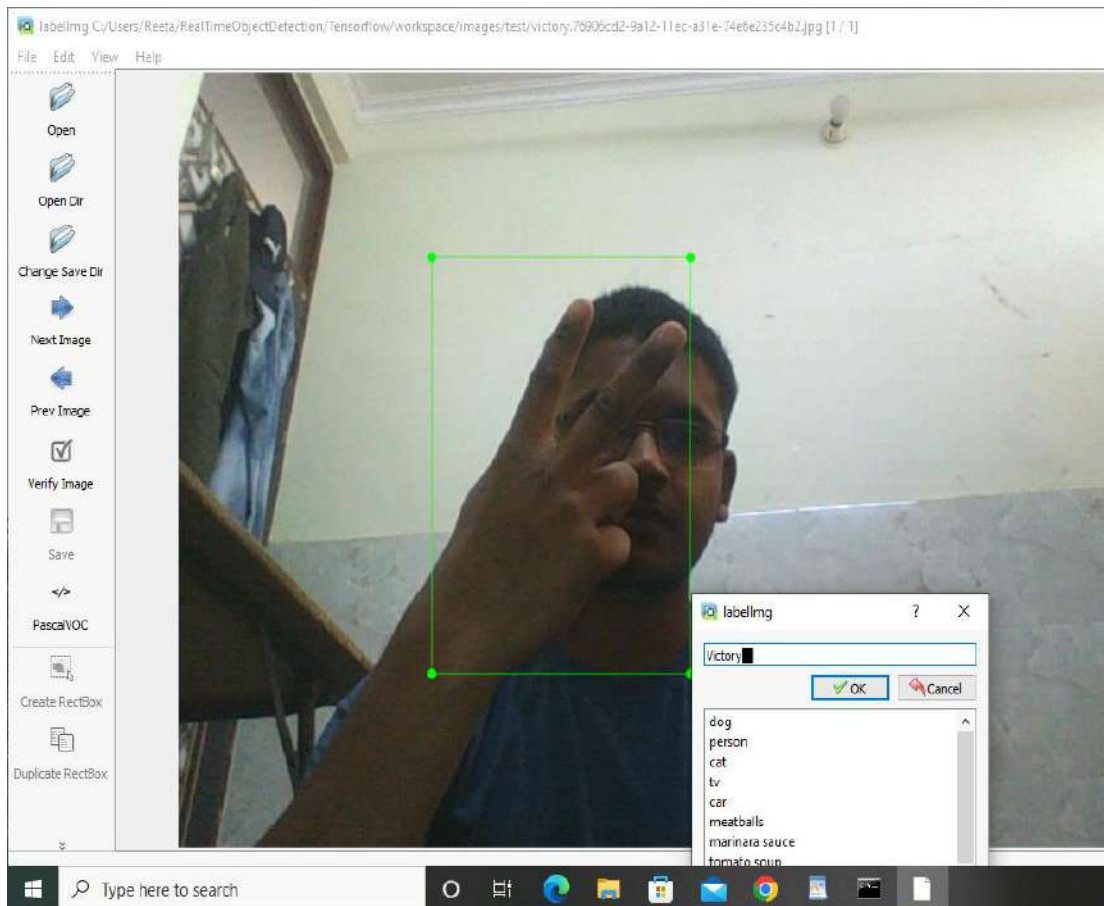


then it will pop up the labeling window as shown in back screen of above photo.

Click on open directory and chose the path where your collected images exist and then change the directories

Click W on scree that will provide you height and width measurement cursor.

select the sign with appropriate box and assign name to that sign.



This will create sample file for your sign in **.xml** format and it will contain all info about sign dimension.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<annotation>
 <folder>collectedimages</folder>
 <filename>victory.75556f55-9a12-11ec-9877-74e6e235c4b2.jpg</filename>
 <path>C:\Users\Reeta\RealTimeObjectDetection\Tensorflow\workspace\images\collectedimages\victory.75556f55-9a12-11ec-9877-74e6e235c4b2.jpg</path>
 <source>
 <database>Unknown</database>
 </source>
 <size>
 <width>640</width>
 <height>480</height>
 <depth>3</depth>
 </size>
 <segmented>0</segmented>
 <object>
 <name>victory</name>
 <pose>Unspecified</pose>
 <truncated>0</truncated>
 <difficult>0</difficult>
 <bndbox>
 <xmin>123</xmin>
 <ymin>99</ymin>
 <xmax>293</xmax>
 <ymax>378</ymax>
 </bndbox>
 </object>
</annotation>
```

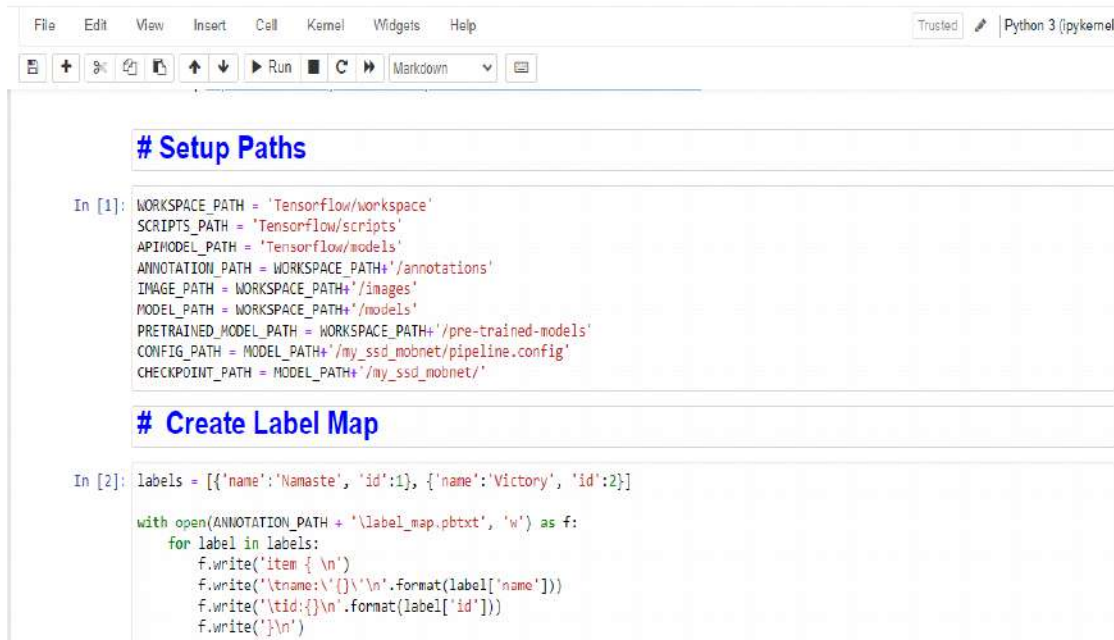
we choose our 8 pictures for every sign and copied it into **train folder** and 2 for our **text folder**.



## 4.)Updating Labelmap

Now open tutorial file in jupyter notebook that come along repository cloning.

Now defining paths and assigning label to different sign as follows



```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Setup Paths

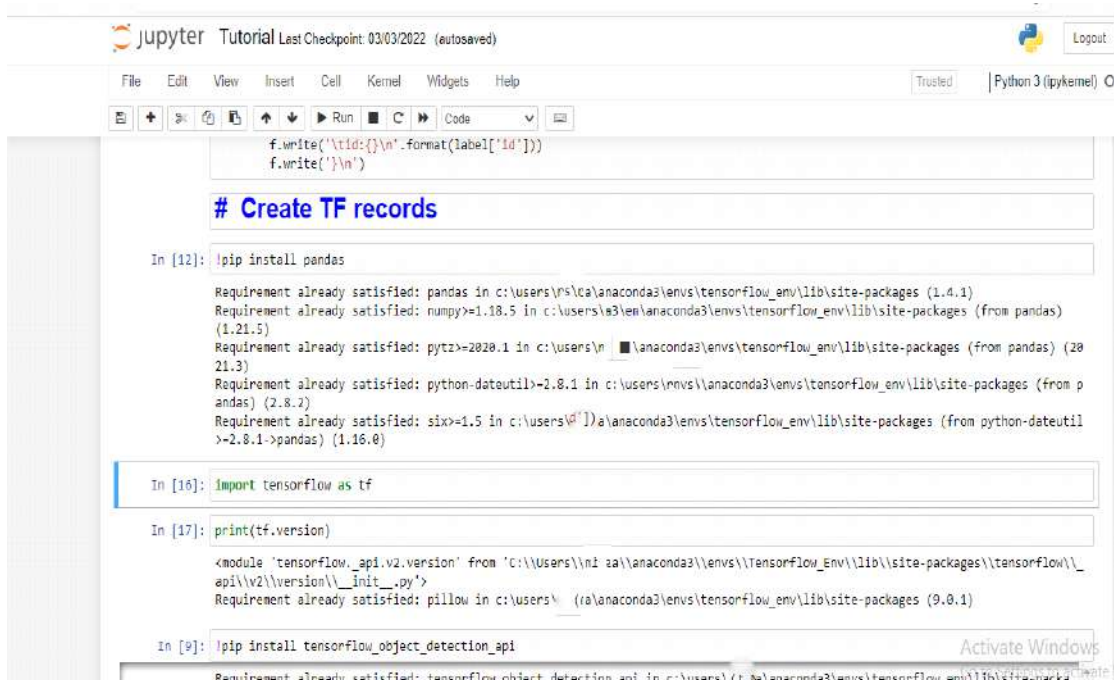
In [1]: WORKSPACE_PATH = 'Tensorflow/workspace'
SCRIPTS_PATH = 'Tensorflow/scripts'
APIMODEL_PATH = 'Tensorflow/models'
ANNOTATION_PATH = WORKSPACE_PATH+'/annotations'
IMAGE_PATH = WORKSPACE_PATH+'/images'
MODEL_PATH = WORKSPACE_PATH+'/models'
PRETRAINED_MODEL_PATH = WORKSPACE_PATH+'/pre-trained-models'
CONFIG_PATH = MODEL_PATH+'/my_ssd_mobnet/pipeline.config'
CHECKPOINT_PATH = MODEL_PATH+'/my_ssd_mobnet/'

Create Label Map

In [2]: labels = [{'name': 'Namaste', 'id':1}, {'name': 'Victory', 'id':2}]

with open(ANNOTATION_PATH + '\\label_map.pbtxt', 'w') as f:
 for label in labels:
 f.write('item {\n')
 f.write(' name: '\\\\n'.format(label['name']))
 f.write(' id: {}\\n'.format(label['id']))
 f.write('}\\n')
```

While working with Object Detection api , it allows to work with special type of file format that why we need to generate tf records



```
Jupyter Tutorial Last Checkpoint: 03/03/2022 (autosaved) Python 3 (ipykernel)

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

Create TF records

In [12]: !pip install pandas

Requirement already satisfied: pandas in c:\users\ns\ca\anaconda3\envs\tensorflow_env\lib\site-packages (1.4.1)
Requirement already satisfied: numpy>=1.18.5 in c:\users\ea3\anaconda3\envs\tensorflow_env\lib\site-packages (from pandas) (1.21.5)
Requirement already satisfied: pytz>=2020.1 in c:\users\n\anaconda3\envs\tensorflow_env\lib\site-packages (from pandas) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\nvs\anaconda3\envs\tensorflow_env\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\af\anaconda3\envs\tensorflow_env\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)

In [16]: import tensorflow as tf

In [17]: print(tf.version)

<module 'tensorflow_api.v2.version' from 'c:\users\ni\aa\anaconda3\envs\tensorflow_env\lib\site-packages\tensorflow\api\v2\version__init__.py'>
Requirement already satisfied: pillow in c:\users\n\anaconda3\envs\tensorflow_env\lib\site-packages (9.0.1)

In [9]: !pip install tensorflow_object_detection_api

Requirement already satisfied: tensorflow object detection api in c:\users\l\l\anaconda3\envs\tensorflow_env\lib\site-packa
```

```

In [10]: !pip install tensorflow_io

Requirement already satisfied: tensorflow_io in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (0.24.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem==0.24.0 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_io) (0.24.0)

In [11]: !pip install tensorflow_gpu

Requirement already satisfied: tensorflow_gpu in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (2.8.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (3.10.0.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (1.13.3)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\reeta\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (1.1.0)
Requirement already satisfied: flatbuffers>=1.12 in c:\users\lib\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (1.12)
Requirement already satisfied: absl-py>=0.4.0 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (0.15.0)
Requirement already satisfied: gast>=0.2.1 in c:\users\reeta\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (0.4.0)
Requirement already satisfied: protobuf>=3.9.2 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (3.14.0)
Requirement already satisfied: grpcio<2.0, >=1.24.3 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (1.42.0)
Requirement already satisfied: numpy>=1.20 in c:\users\flow_a\anaconda3\envs\tensorflow_env\lib\site-packages (from tensorflow_gpu) (1.21.5)

```

!pip list command result in:-

Package	Version
-----	
absl-py	0.15.0
aiohttp	3.8.1
aiosignal	1.2.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
astor	0.8.1
astunparse	1.6.3
async-timeout	4.0.1
attrs	21.4.0
backcall	0.2.0
bleach	4.1.0
blinker	1.4

brotlipy	0.7.0
cachetools	4.2.2
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.4
clang	5.0
click	8.0.4
colorama	0.4.4
contextlib2	21.6.0
cryptography	3.4.8
cycler	0.11.0
Cython	0.29.28
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
docutils	0.18.1
entrypoints	0.3
flatbuffers	1.12
fonttools	4.31.2
frozenset	1.2.0
gast	0.4.0
google-auth	1.35.0
google-auth-oauthlib	0.4.1
google-pasta	0.2.0

grpcio	1.42.0
h5py	3.6.0
idna	3.3
importlib-metadata	4.8.2
ipykernel	6.4.1
ipython	7.31.1
ipython-genutils	0.2.0
ipywidgets	7.7.0
jedi	0.18.1
Jinja2	3.0.2
jsonschema	3.2.0
jupyter	1.0.0
jupyter-client	7.1.2
jupyter-console	6.4.3
jupyter-core	4.9.1
jupyterlab-pygments	0.1.2
jupyterlab-widgets	1.1.0
keras	2.8.0
Keras-Preprocessing	1.1.2
keyring	23.5.0
kiwisolver	1.4.0
libclang	13.0.0
lxml	4.8.0
Markdown	3.3.4

MarkupSafe	2.0.1
matplotlib	3.5.1
matplotlib-inline	0.1.2
mistune	0.8.4
mk1-fft	1.3.1
mk1-random	1.2.2
mk1-service	2.4.0
multidict	5.1.0
nbclient	0.5.11
nbconvert	6.1.0
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.8
numpy	1.21.5
oauthlib	3.2.0
opt-einsum	3.3.0
packaging	21.3
pandas	1.4.1
pandocfilters	1.5.0
parso	0.8.3
pickleshare	0.7.5
Pillow	9.0.1
pip	21.2.4
pkginfo	1.8.2

prometheus-client	0.13.1
prompt-toolkit	3.0.20
protobuf	3.14.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycparser	2.21
Pygments	2.11.2
PyJWT	2.1.0
pyOpenSSL	21.0.0
pyparsing	3.0.4
pyreadline	2.1
pyrsistent	0.18.0
PySocks	1.7.1
python-dateutil	2.8.2
pytz	2021.3
pywin32	302
pywin32-ctypes	0.2.0
pywinpty	2.0.2
pyzmq	22.3.0
qtconsole	5.2.2
QtPy	2.0.1
readme-renderer	34.0
requests	2.27.1
requests-oauthlib	1.3.0

requests-toolbelt	0.9.1
rfc3986	2.0.0
rsa	4.7.2
scipy	1.7.3
Send2Trash	1.8.0
setuptools	58.0.4
six	1.16.0
tensorboard	2.8.0
tensorboard-data-server	0.6.0
tensorboard-plugin-wit	1.6.0
tensorflow	2.6.0
tensorflow-estimator	2.6.0
tensorflow-gpu	2.8.0
tensorflow-io	0.24.0
tensorflow-io-gcs-filesystem	0.24.0
tensorflow-object-detection-api	0.1.1
termcolor	1.1.0
terminado	0.13.1
testpath	0.5.0
tf-estimator-nightly	2.8.0.dev2021122109
tornado	6.1
tqdm	4.63.1
traitlets	5.1.1
twine	3.8.0

typing-extensions	3.10.0.2
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
Werkzeug	2.0.3
wheel	0.35.1
widgetsnbextension	3.6.0
win-inet-pton	1.1.0
wincertstore	0.2
wrapt	1.13.3
yaml	1.6.3
zipp	3.7.0

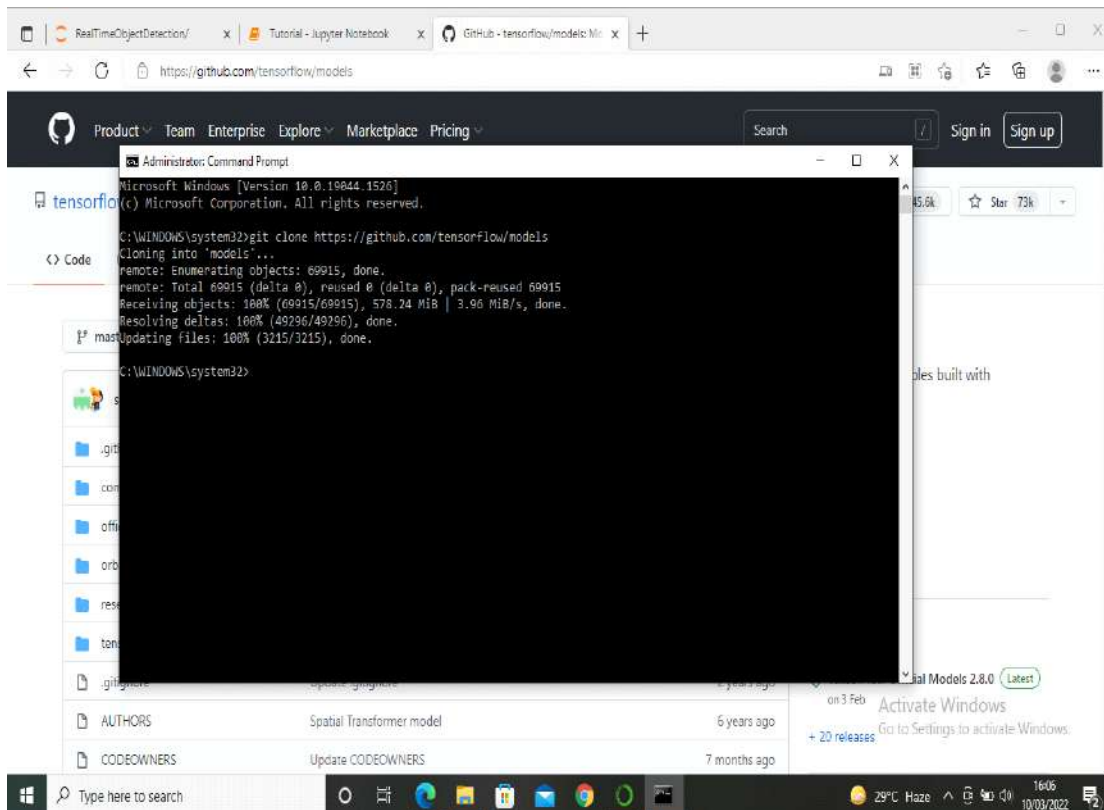
***To generate tf records we need to write code as***

```
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x
{IMAGE_PATH + '/train'} -l {ANNOTATION_PATH +
'/label_map.pbtxt'} -o {ANNOTATION_PATH + '/train.record'}

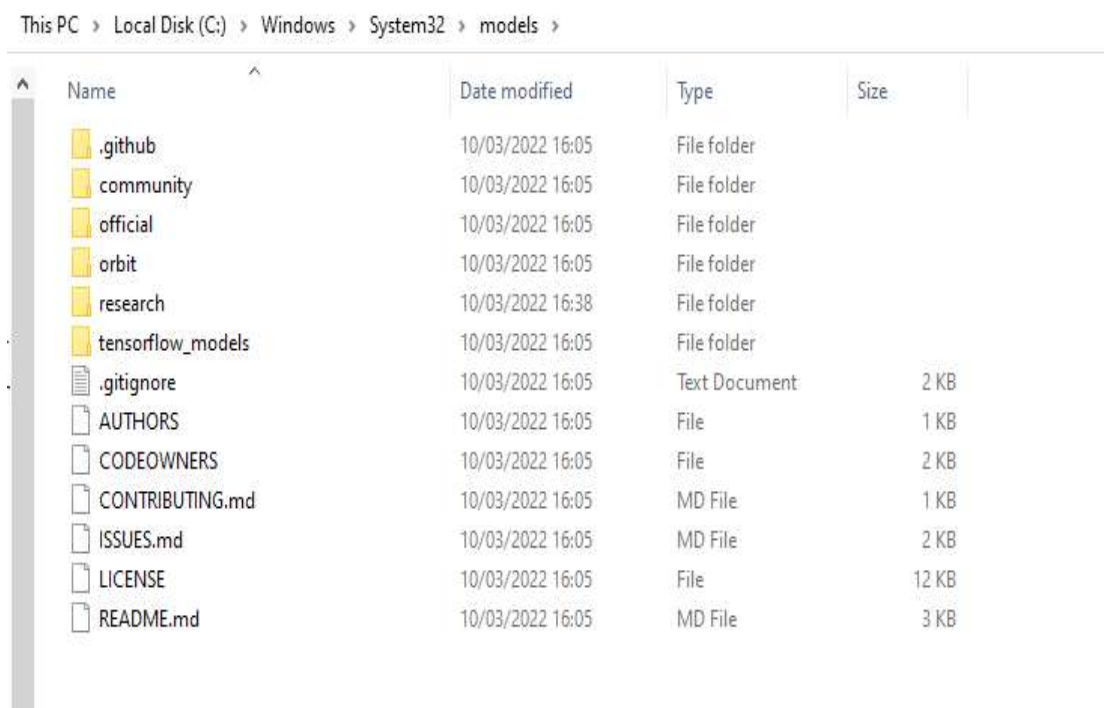
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -
x{IMAGE_PATH + '/test'} -l {ANNOTATION_PATH +
'/label_map.pbtxt'} -o {ANNOTATION_PATH + '/test.record'}
```

Next is to command in command prompt of git clone <https://github.com/tensorflow/models>. This is have the all repositories required and all used the script that is needed to work with tensorflow.





and this will create file path as >> C:\Windows\System32\models



```
In [4]: !cd Tensorflow && git clone https://github.com/tensorflow/models
fatal: destination path 'models' already exists and is not an empty directory.

In [5]: #wget.download('http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz')
#mv ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz {PRETRAINED_MODEL_PATH}
#cd {PRETRAINED_MODEL_PATH} && tar -zxvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

#### 4. Copy Model Config to Training Folder

```
In [6]: CUSTOM_MODEL_NAME = 'my_ssd_mobnet'

In [7]: !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}
A subdirectory or file TensorFlow\workspace\models\my_ssd_mobnet already exists.
```

#### 5. Update Config For Transfer Learning

```
In [8]: import os
os.add_dll_directory("C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.1/bin")

Out[8]: <AddedDllDirectory('C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.1/bin')>
```

## Importing dependencies

```
In [9]: import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

```
In [10]: CONFIG_PATH = MODEL_PATH+'/' +CUSTOM_MODEL_NAME+'pipeline.config'
```

```
In [11]: !pip install pycocotools
```

```
Collecting pycocotools
 Using cached pycocotools-2.0.4.tar.gz (106 kB)
 Installing build dependencies: started
 Installing build dependencies: finished with status 'done'
 Getting requirements to build wheel: started
 Getting requirements to build wheel: finished with status 'done'
 Preparing wheel metadata: started
 Preparing wheel metadata: finished with status 'done'
Requirement already satisfied: numpy in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from pycocotools) (1.21.0)
Requirement already satisfied: matplotlib>=2.1.0 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from pycocotools) (3.5.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (9.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (1.4.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (3.0.4)
Requirement already satisfied: cycler>=0.10 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\user\anaconda3\envs\tensorflow_env\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (21.3)
```

```
In [12]: config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
```

```
In [13]: config
```

```
Out[13]: {'model': 'ssd',
 'num_classes': 90,
 'image_resizer':
 {'fixed_shape_resizer':
 {'height': 320,
 'width': 320}
 },
 'feature_extractor':
 {'type': 'ssd_mobilenet_v2_fpn_keras',
 'depth_multiplier': 1.0,
 'min_depth': 16,
 'conv_hyperparams':
 {'regularizer':
 {'l2_regularizer':
 {'weight': 4e-05}
 }
 },
 'initializer':
```

Activate Windows

```
In [14]: 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)
```

```
In [15]: 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'
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']
```

```
In [16]: config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(CONFIG_PATH, "wb") as f:
 f.write(config_text)
```

## 6. Train the model

```
In [17]: print("""python {}research/object_detection/model_main_tf2.py --model_dir={} --pipeline_config_path={} --pipeline.config --num_train_steps=10000""")

python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000

In [25]: !pip install -U cython

Requirement already satisfied: cython in c:\users\jta\anaconda3\envs\tensorflow_env\lib\site-packages (0.29.28)

In [27]: !pip install git+https://github.com/philferriere/cocoapi.git#egg=pycocotools^&subdirectory=PythonAPI

Collecting pycocotools
 Running command git clone -q https://github.com/philferriere/cocoapi.git 'C:\Users\jta\AppData\Local\Temp\pip-install-earm8liq\pycocotools_97408baf04904f83a4fc6dbb6d9d4c99'
 Cloning https://github.com/philferriere/cocoapi.git to c:\users\jta\AppData\Local\Temp\pip-install-earm8liq\pycocotools_97408baf04904f83a4fc6dbb6d9d4c99
 Resolved https://github.com/philferriere/cocoapi.git to commit 2929bd2ef6b451054755dfd7ceb09278f935f7ad
 Building wheels for collected packages: pycocotools
 Building wheel for pycocotools (setup.py): started
 Building wheel for pycocotools (setup.py): finished with status 'done'
 Created wheel for pycocotools: filename=pycocotools-2.0-cp39-cp39-win_amd64.whl size=82051 sha256=f035b0d430f7a20d7ea7a118374
```

Copied the link

(pythonTensorflow/models/research/object\_detection/model\_main\_tf2.py --model\_dir=Tensorflow/workspace/models/my\_ssd\_mobnet --pipeline\_config\_path=Tensorflow/workspace/models/my\_ssd\_mobnet/pipeline.config --num\_train\_steps=10000)

and run it in command prompt as shown

```

Command Prompt
C:\Users\ vna>activate Tensorflow_Env

(Tensorflow_Env) C:\Users\ C:\>cd RealTimeObjectDetection

(Tensorflow_Env) C:\Users\ C:\RealTimeObjectDetection>python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000
2022-04-06 17:06:35.928148: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cuda64_110.dll'; dlerror: cuda64_110.dll not found
2022-04-06 17:06:35.928386: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Traceback (most recent call last):
 File "C:\Users\d_mob\RealTimeObjectDetection\Tensorflow\models\research\object_detection\model_main_tf2.py", line 32, in <module>
 from object_detection import model_lib_v2
 File "C:\Users\te-pa\anaconda3\envs\Tensorflow_Env\lib\site-packages\object_detection\model_lib_v2.py", line 29, in <module>
 from object_detection import eval_util
 File "C:\Users\R_Env\anaconda3\envs\Tensorflow_Env\lib\site-packages\object_detection\eval_util.py", line 35, in <module>
 slim = tf.compat.v1.estimator.slim
 File "C:\Users\te-pa\anaconda3\envs\Tensorflow_Env\lib\site-packages\tensorflow\python\util\lazy_loader.py", line 59, in __getattr__
 return getattr(module, item)
 File "C:\Users\conda\anaconda3\envs\Tensorflow_Env\lib\site-packages\tensorflow\python\util\module_wrapper.py", line 232, in __getattr__
 attr = getattr(self, tfmw_wrapped_module, name)
AttributeError: module 'tensorflow_estimator.python.estimator.api.v1.estimator' has no attribute 'slim'

(Tensorflow_Env) C:\Users\von> \RealTimeObjectDetection>
fb3b671f116e4203747e7341cd3a3035e435b

```

This error of attribute 'slim' was non-resolvable . if it resolves then it leads to other attributeerror 'contrib'.

```

ModuleNotFoundError: No module named 'pycocotools'

(Tensorflow_Env) C:\Users\flowa\RealTimeObjectDetection>python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000
2022-04-06 16:47:45.413981: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cuda64_110.dll'; dlerror: cuda64_110.dll not found
2022-04-06 16:47:45.447728: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Matplotlib is building the font cache; this may take a moment.
Traceback (most recent call last):
 File "C:\Users\flowa\RealTimeObjectDetection\Tensorflow\models\research\object_detection\model_main_tf2.py", line 32, in <module>
 from object_detection import model_lib_v2
 File "C:\Users\det\anaconda3\envs\Tensorflow_Env\lib\site-packages\object_detection\model_lib_v2.py", line 29, in <module>
 from object_detection import eval_util
 File "C:\Users\reet\anaconda3\envs\Tensorflow_Env\lib\site-packages\object_detection\eval_util.py", line 35, in <module>
 slim = tf.contrib.slim
AttributeError: module 'tensorflow' has no attribute 'contrib'

Activate Windows
Go to Settings to activate Windows

```

## Conclusions:

We have been successfully able to recognise 16 signs by machine learning using OPENCV (to access webcam),mediapipe (to recognise hand landmarks) and playsound (to convert text to audio).

The project initially aimed at converting sign language to text but here we were able to go a step ahead by converting it to sound also.

For the approach 2 we have successfully cloned repositories, collected images using our webcam, labelled them, generated Tf records and went on to the training part where we encountered multiple errors ; some of them we solved , for others we were stuck in a looped error.

## **Future scope:**

1. This could be used by dumb and deaf people as a way of communication. Further a app could be developed that could be interfaced with others like Microsoft teams ,google meet or zoom so that communication could be at ease at these platforms too for them.
- 2.This method could be further used for object detection. It could be implemented to detect facial and gesture recognition.
3. It could be used in security checks at public places.
4. This method could also be employed for lane detection, number plate detection, face mask detection and to control multiple computer operations.
5. It could be further extended to two handed dynamic signs as well.

And many more.

## References:

1. Computer Science with python by Sumita Arora.
2. Giraffe Academy for python learning
3. [\(1451\) Real Time Sign Language Detection with Tensorflow Object Detection and Python | Deep Learning SSD - YouTube](#)
4. <https://google.github.io/mediapipe/>
5. <https://omdena.com/blog/mediapipe-python-tutorial/>
- 6.. <https://youtube.com/c/DataMagic2020>
7. <https://github.com/tensorflow/tensorflow/issues/55620>
8. <https://github.com/tensorflow/tensorflow/issues/27989>
9. [https://github.com/datitran/raccoon\\_dataset/issues/85](https://github.com/datitran/raccoon_dataset/issues/85)
10. <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html>
11. stack exchange for debugging errors.
12. <https://tutedude.com/category/python>