Real Time Hand Gesture Detection

A Report of Major Project

Submitted towards the partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology in Civil Engineering with Specialization in Computer Science

Submitted by

Kapil Srivastava (1700644)

Kartikeya (2002051)

Under the supervision of

Ms. Priyanka Chauhan

Department of Civil Engineering

FACULTY OF ENGINEERING

DAYALBAGH EDUCATIONAL INSTITUTE

DAYALBAGH, AGRA

DAYALBAGH EDUCATIONAL INSTITUTE

DAYALBAGH, AGRA – 282005

CERTIFICATE

This is to certify that the project titled, “Real Time Hand Gesture Detection” submitted by Kapil Srivastava and Kartikeya for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Civil Engineering with Specialization in Computer Science from Faculty of Engineering, Dayalbagh Educational Institute, Agra, is a record of the bonafide work carried out under my supervision. Further, that the matter in this report has not been submitted to any other University/ Institute for the award of any degree.

Ms. Priyanka Chauhan	Head of Department
Department of Civil Engineering	Department of Civil Engineering
Faculty of Engineering	Faculty of Engineering

DAYALBAGH EDUCATIONAL INSTITUTE

DAYALBAGH, AGRA – 282005

DECLARATION

I solemnly affirm that the project report titled, "Real Time Hand Gesture Detection" submitted to the Dayalbagh Educational Institute, Agra, is a record of an original work done by me/us under the guidance of Ms. Priyanka Chauhan, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Civil Engineering with Specialization in Computer Science. The results embodied in this report have not been submitted to any other Universities or Institute for the award of any degree.

Kapil Srivastava
Roll No : 1700644

Kartikeya
Roll No : 2002051

ACKNOWLEDGEMENTS

I am very grateful to my supervisor Ms. Priyanka Chauhan,

I sincerely thank Prof, Sanjay Kumar Srivastava, Head of Civil Engineering Department. I also extend my gratitude towards Prof. C. Patvardhan, Dean, Faculty of Engineering for constantly guiding me throughout the project.

I am also thankful to my parents, for providing me with the necessary means to complete this project. I am indebted to them for their constant encouragement, love and affection.

Kapil Srivastava

Kartikeya

ABSTRACT

Hand gesture recognition has emerged as a promising and intuitive interface for human-computer interaction, enabling natural communication between users and machines. This paper provides a comprehensive overview of the advancements and methodologies in hand gesture recognition using machine learning techniques.

The review begins by presenting the foundational concepts and challenges associated with hand gesture recognition, including the complexity of hand articulation, variations in lighting conditions, and occlusion issues. Subsequently, it delves into the evolution of machine learning algorithms applied to this domain, exploring classical techniques, such as rule-based systems and template matching, to the more recent and sophisticated approaches based on deep learning.

The paper highlights the requirements as well as the methodology to create a working model that can detect hand gestures in real-time and create meaningful sentences out of it. All the environment setup details are provided with the programming requirements. The review addresses real-world implementations of hand gesture recognition systems, showcasing success stories and identifying areas for future research. The integration of gesture recognition in diverse domains, such as smart homes, automotive interfaces, and industrial settings, is explored to underscore the practical implications and societal impact of this technology.

TABLE OF CONTENTS

DESCRIPTION

TABLE OF IMAGES

1. INTRODUCTION

In the realm of human-computer interaction, the quest for more natural and intuitive interfaces has led to the development of real-time hand gesture detection using machine learning (ML). This cutting-edge technology holds the promise of revolutionizing how we interact with devices, from augmented reality applications and gaming to healthcare and smart home control.

Hand gestures are a fundamental aspect of human communication, and harnessing this natural form of expression for human-computer interaction enhances user experience and accessibility. Real-time hand gesture detection using ML enables devices to interpret and respond to gestures seamlessly, bringing us closer to a future where communication with machines feels instinctive.

The implementation of real-time hand gesture detection poses several challenges, including the variability in hand shapes, lighting conditions, and the need for quick response times. Machine learning algorithms, particularly deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have proven effective in overcoming these challenges.

Real-time hand gesture detection using machine learning is at the forefront of human-computer interaction, making technology more accessible and user-friendly. As research and development in this field progresses, we can anticipate a future where our interactions with devices are not only seamless but also deeply intuitive, creating a bridge between the digital and physical worlds.

2. DISCUSSION

2.1Overview

Real time hand gesture detection is a software/program that is used to detect the hand gestures made by the user and convert it into meaning texts/sentences. These programs use the algorithms of Machine Learning (ML), Deep Learning (DL), Natural Language Processing (NLP), etc.

The future of this kind of project holds a lot of potential that can be used to transform the lives of many specially abled persons [1]. As the technologies in the fields are advancing, a future using these types of tools has become more conceivable than before.

Sign languages are languages that use the visual-manual modality to convey meaning, instead of spoken words. Sign languages are expressed through manual articulation in combination with non-manual markers. Sign languages are full-fledged natural languages with their own grammar and lexicon.

As there are many sign languages, our model will predominantly take use of two types of sign languages.

To execute a model of such functionalities it requires various libraries to work in unison. These libraries will provide the suitable environment for the execution of the program. These such libraries are as follows: -

- Python
- Jupyter Notebook
- OpenCV
- Tensorflow
- Tensorflow Object Detection API

2.2 Python

Fig. 1 Python logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Reason for increasing popularity:

- Emphasis on code readability, shorter codes, ease of writing
- Programmers can express logical concepts in fewer lines of code in comparison to languages such as C++ or Java.
- Python supports multiple programming paradigms, like object-oriented, imperative and functional programming or procedural.
- There exist inbuilt functions for almost all of the frequently used concepts.

2.3 Jupyter Notebook

Fig. 2 Jupyter notebook logo

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Project Jupyter is a project to develop open-source software, open standards, and services for interactive computing across multiple programming languages. It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger.

Jupyter Notebook is a notebook authoring application, under the Project Jupyter umbrella. Built on the power of the computational notebook format, Jupyter Notebook offers fast, interactive new ways to prototype and explain your code, explore and visualize your data, and share your ideas with others.

Notebooks extend the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.

2.4 OpenCV

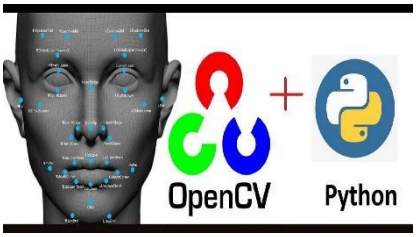


Fig. 3 Open Source Computer Vision + Python logo

OpenCV is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez. The library is cross-platform and licensed as free and open-source software under Apache License 2.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

OpenCV Functionality:

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

2.5 Tensorflow

Fig. 4 Tensorflow logo

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015.Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories:

1. Low level API:
- complete programming control
- recommended for machine learning researchers
- provides fine levels of control over the models
- TensorFlow Core is the low-level API of TensorFlow.
2. High level API:
- built on top of TensorFlow Core
- easier to learn and use than TensorFlow Core
- make repetitive tasks easier and more consistent between different users
- tf.contrib. learn is an example of a high-level API.

2.6 Tensorflow Object Detection API

Fig. 5 Tensorflow object detection API logo

The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train, and deploy object detection models. There are already pre-trained models in their framework which they refer to as Model Zoo.

The API provides a set of Python libraries and tools for training and deploying object detection models. It also includes a set of pre-trained models that can be used for a variety of object detection tasks.

To use the TensorFlow Object Detection API, you will need to install the TensorFlow library. Once you have installed TensorFlow, you can clone the TensorFlow Models repository and install the Object Detection API.

Once you have installed the Object Detection API, you can start training your own object detection model. The API provides a number of tutorials and examples that can help you get started.

3. WORK FLOW OF MODEL

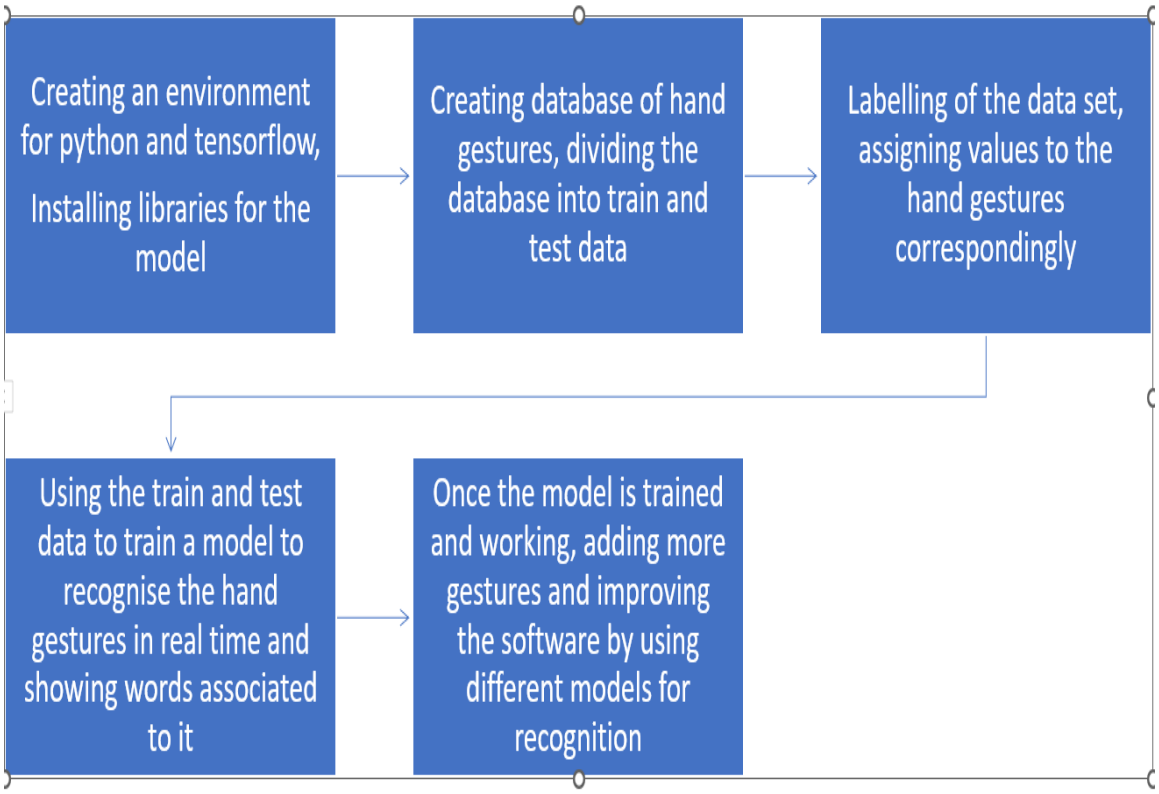


Fig. 6 Flow chart of the model

3.1 Creating databases

Jupyter notebook and python program is shown in the picture that creates a database of the images to be used further for training the model [2].

We imported various libraries of python, and specified a folder for the collection of datasets.

For starters, we are using ten hand gestures to create the database. These ten hand gestures, our model is able to recognise and produce a sentence pertaining to it.

```
import cv2

import os

import time

import uuid
```

```
IMAGES_PATH = 'Tensorflow\workspace\images\CollectedImages'

labels = ['Hello', 'ThankYou', 'Yes', 'No', 'ILoveYou', 'Good', 'Excellent', 'Enjoy',
'BestOfLuck', 'HowAreYou']

number_imgs = 40

for label in labels:

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

    cap = cv2.VideoCapture(0)

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

    time.sleep(5)

    for imgnum in range(number_imgs):

        ret, frame = cap.read()

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

        cv2.imwrite(imgname, frame)

        cv2.imshow('frame', frame)

        time.sleep(2)



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

#             break

    cap.release()
```

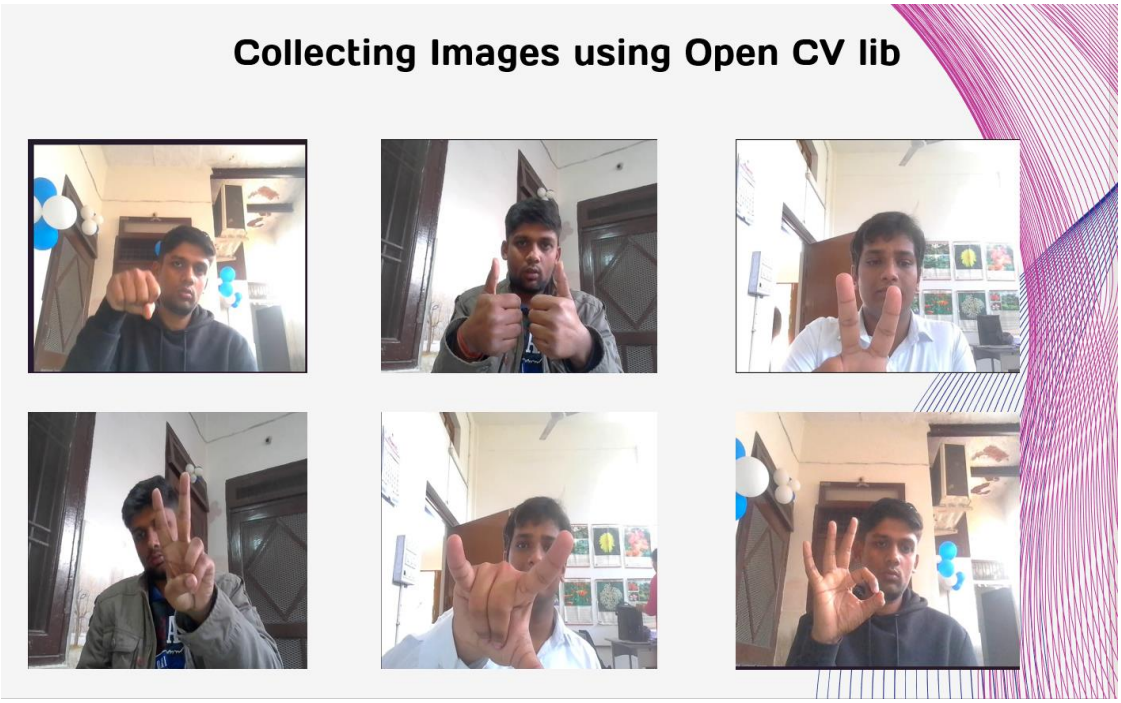


Fig. 7 Hand gestures used to train model

3.2 Labeling of images

The collected images then are labeled by using the labeling package of python.

In labeling, each image is provided with a value, while recognition of this value is displayed. This is done by using a ‘LabelIng’ library of python.

These files are converted into their xml files, which contains the values of all the attributes required for the detection part. Hence the next set of code involves the conversion of the XML file to an intermediate CSV output and then the final conversion of the CSV output to TF-Records.

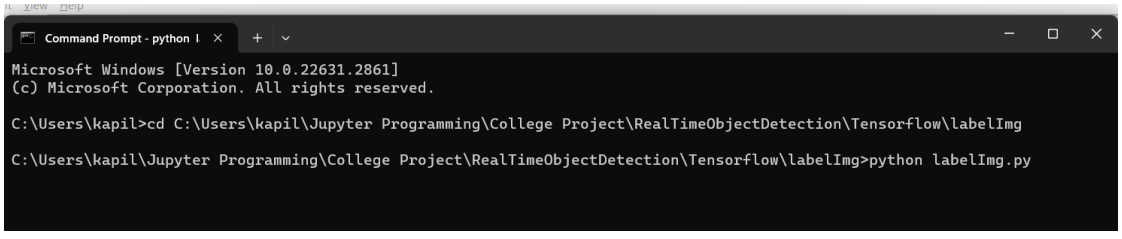


Fig. 8 Open labelimg.py using CMD

<annotation>
<folder>CollectedImages</folder>
<filename>BestOfLuck.0aa9bae9-b852-11ee-869a-fb7472eb6576.jpg</filename>
<path>C:\Users\kapil\Jupyter Programming\College Project\RealTimeObjectDetection\Tensorflow\workspace\images\CollectedImages\BestOfLuck.0aa9bae9-b852-11ee-869a-fb7472eb6576.jpg</path>
<source>
<database>Unknown</database>
</source>
<size>
<width>640</width>
<height>480</height>
<depth>3</depth>
</size>
<segmented>0</segmented>
<object>
<name>BestOfLuck</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>220</xmin>

<ymin>190</ymin>

<xmax>347</xmax>

<ymax>403</ymax>

</bndbox>

</object>

</annotation>

3.3 Training Set

Training set usually consists of the 80% of the data set that is used to train the model. These are the general guidelines but can be changed according to the nature and size of the dataset.

The photo shows the training dataset along with the corresponding xml files, both of these will be used to train the model.

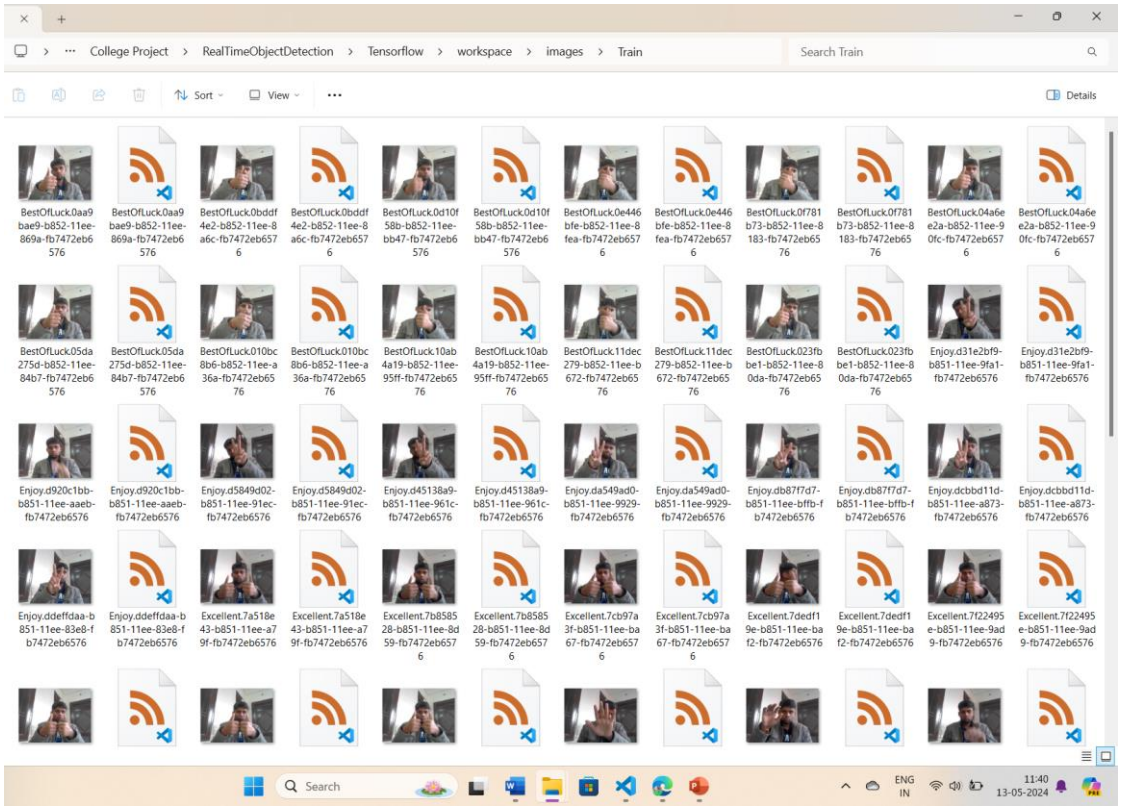


Fig. 9 Training folder contains training dataset

3.4 Testing Set

The testing dataset usually contains the 20% of the dataset, it comes in use to test the efficiency of the model, as mentioned earlier the size of this dataset can also vary.

As shown in the figure, the testing dataset also contains pictures along with their xml files.

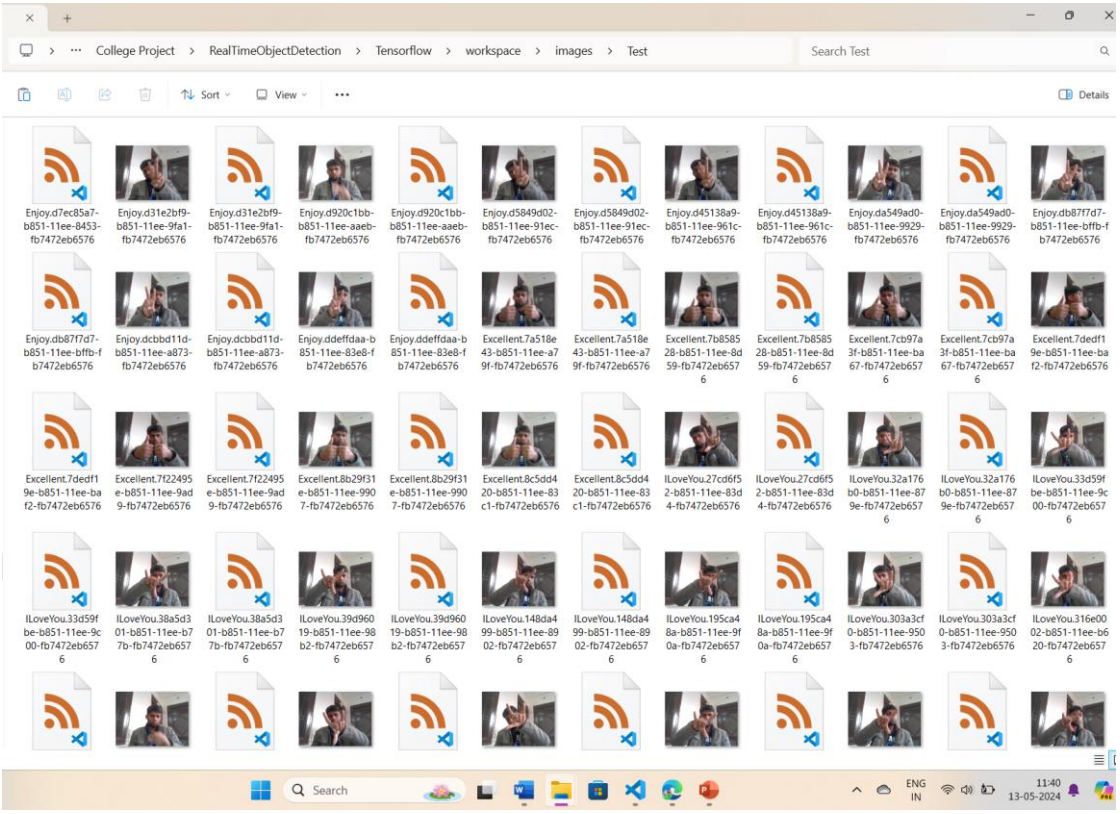


Fig. 10 Testing folder contains test dataset

3.5 Generating TF_Records

TF_Record is a custom TensorFlow format for storing a sequence of binary records. TF_Records are highly optimized for TensorFlow, which lead to them having the following advantages:

- Efficient form of data storage
- Faster read speed compared to other types of formats

One of the most important use cases of TF_Records is when we train a model using TPU. TPUs are super powerful but require the data they interact with to be stored remotely (usually, we use Google Cloud Storage), and that's where TF_Records come in. We store the datasets remotely in TF_Record format when training a model on TPU since it makes saving the data efficiently and loading the data easier.

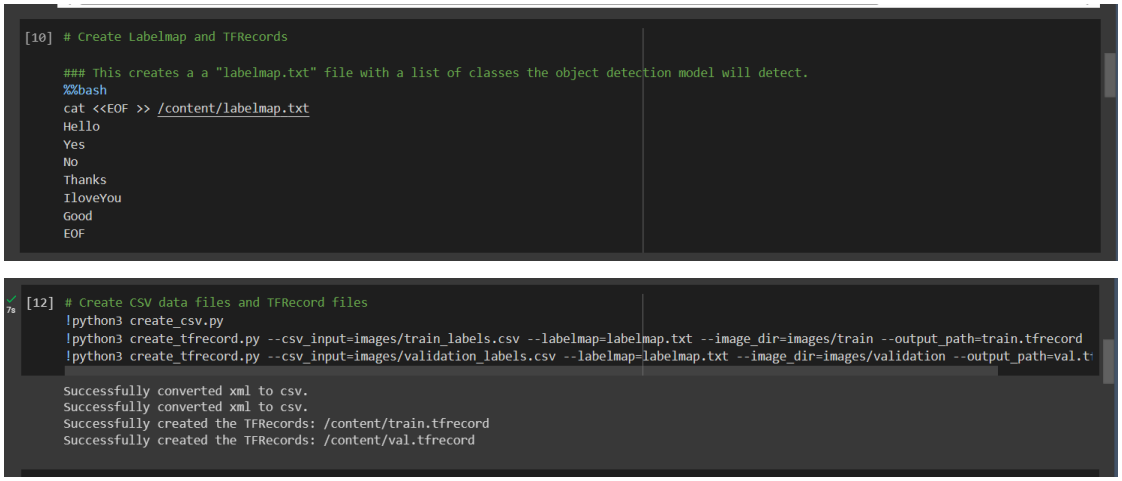
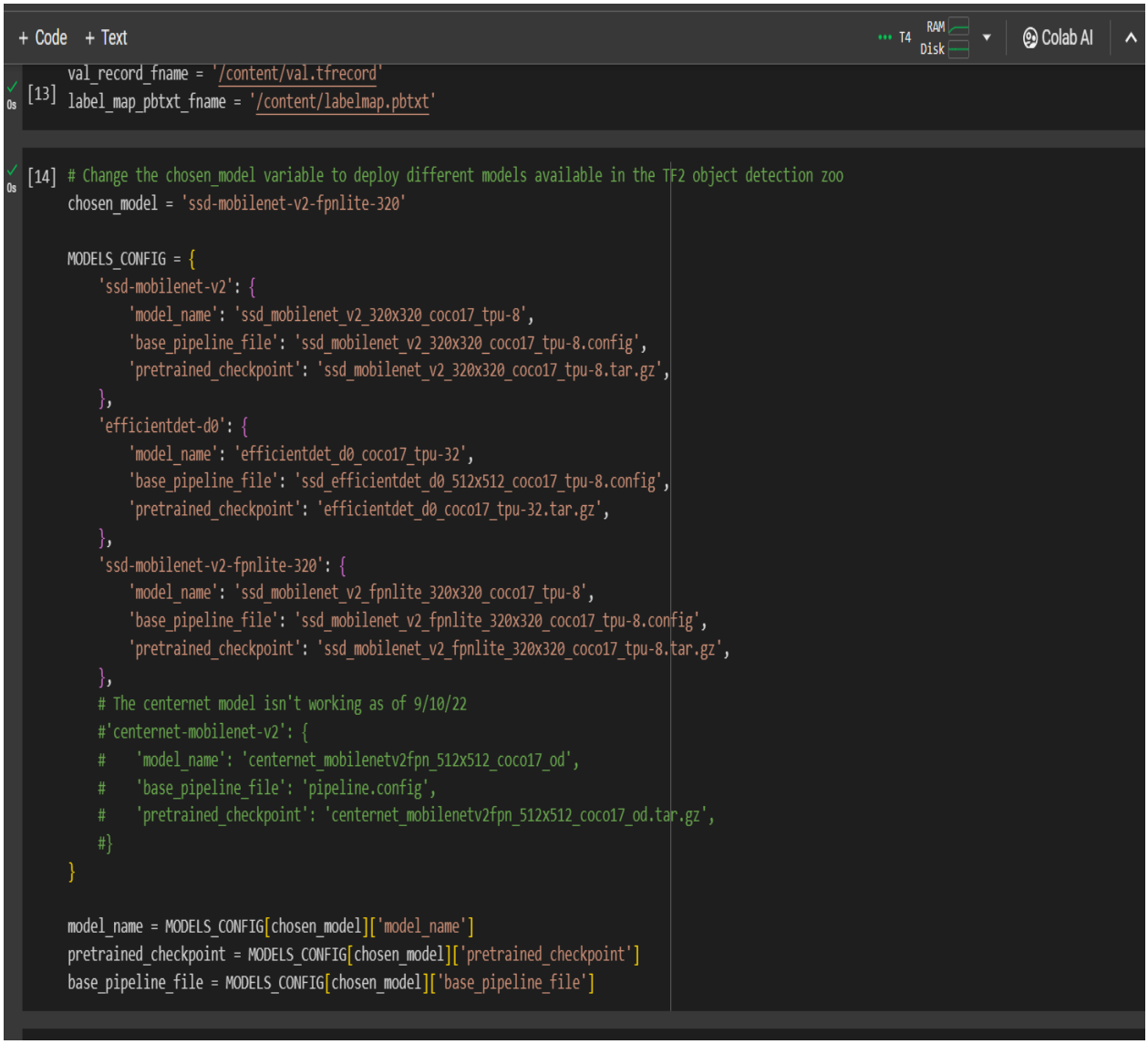


Fig. 11 Creating TF_records

3.6 Model Training

One of the most common hurdles in an object detection problem is choosing a model that will do really well on a custom dataset. But how do you decide which model to use when there are so many models out there to choose from? Research and Development in Deep Learning has led to the availability of several models which makes selecting a model somewhat difficult and time-consuming.

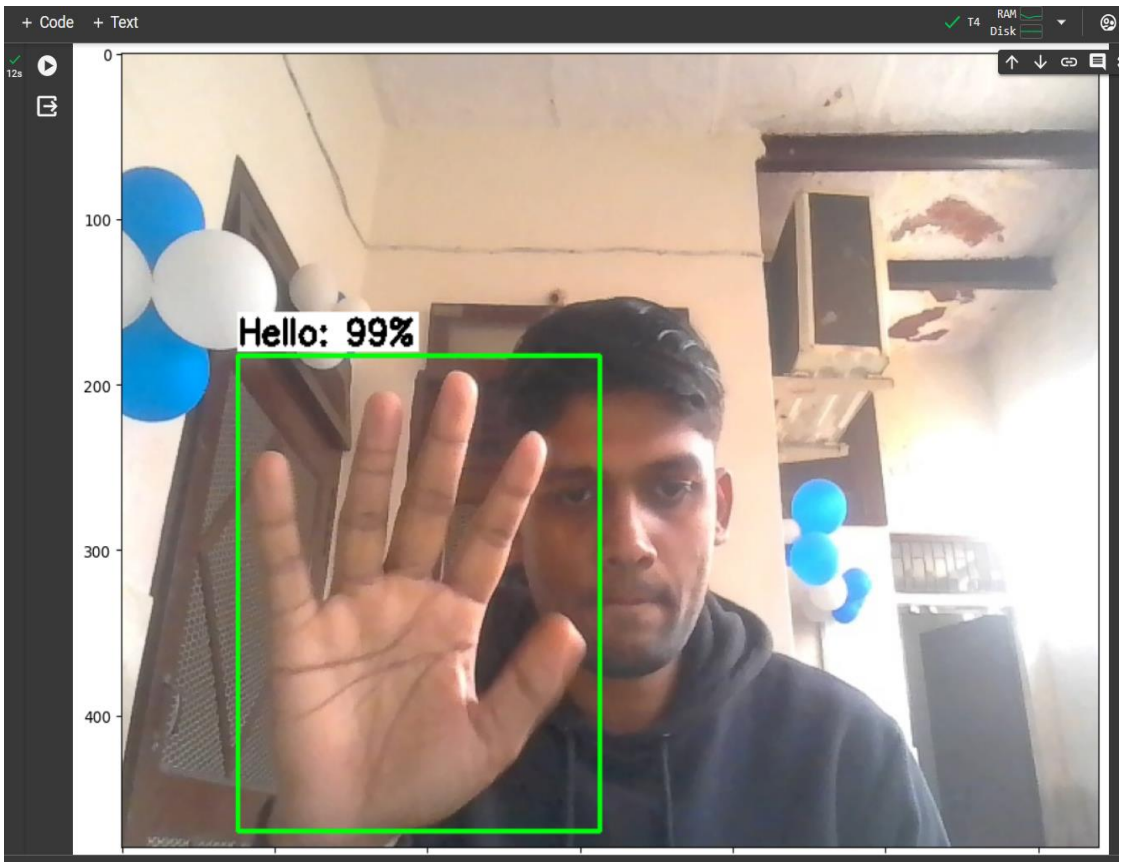


```
+ Code + Text                                                    T4 RAM Disk   Colab AI

[13] val_record_fname = '/content/val.tfrecord'
     label_map_pbtxt_fname = '/content/labelmap.pbtxt'

[14] # Change the chosen_model variable to deploy different models available in the TF2 object detection zoo
     chosen_model = 'ssd-mobilenet-v2-fpnlite-320'

     MODELS_CONFIG = {
         'ssd-mobilenet-v2': {
             'model_name': 'ssd_mobilenet_v2_320x320_coco17_tpu-8',
             'base_pipeline_file': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.config',
             'pretrained_checkpoint': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz',
         },
         'efficientdet-d0': {
             'model_name': 'efficientdet_d0_coco17_tpu-32',
             'base_pipeline_file': 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config',
             'pretrained_checkpoint': 'efficientdet_d0_coco17_tpu-32.tar.gz',
         },
         'ssd-mobilenet-v2-fpnlite-320': {
             'model_name': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8',
             'base_pipeline_file': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config',
             'pretrained_checkpoint': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz',
         },
         # The centernet model isn't working as of 9/10/22
         #'centernet-mobilenet-v2': {
         #    'model_name': 'centernet_mobilenetv2fpn_512x512_coco17_od',
         #    'base_pipeline_file': 'pipeline.config',
         #    'pretrained_checkpoint': 'centernet_mobilenetv2fpn_512x512_coco17_od.tar.gz',
         #}
     }

     model_name = MODELS_CONFIG[chosen_model]['model_name']
     pretrained_checkpoint = MODELS_CONFIG[chosen_model]['pretrained_checkpoint']
     base_pipeline_file = MODELS_CONFIG[chosen_model]['base_pipeline_file']
```

Fig. 12 Code for model selection and training

SSD Mobilenet V2 is a one-stage object detection model which has gained popularity for its lean network and novel depth wise separable convolutions. It is a model commonly deployed on low compute devices such as mobile (hence the name Mobilenet) with high accuracy performance [3].

3.7 Model Testing and Deployment



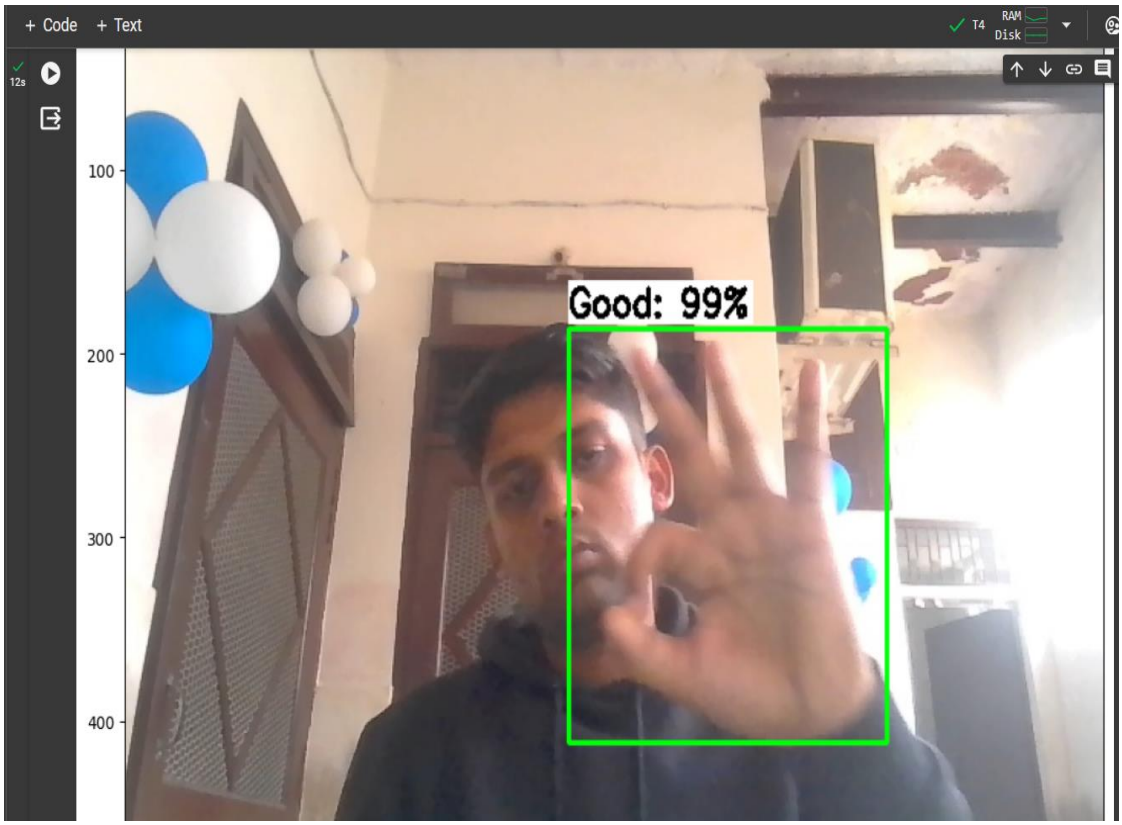
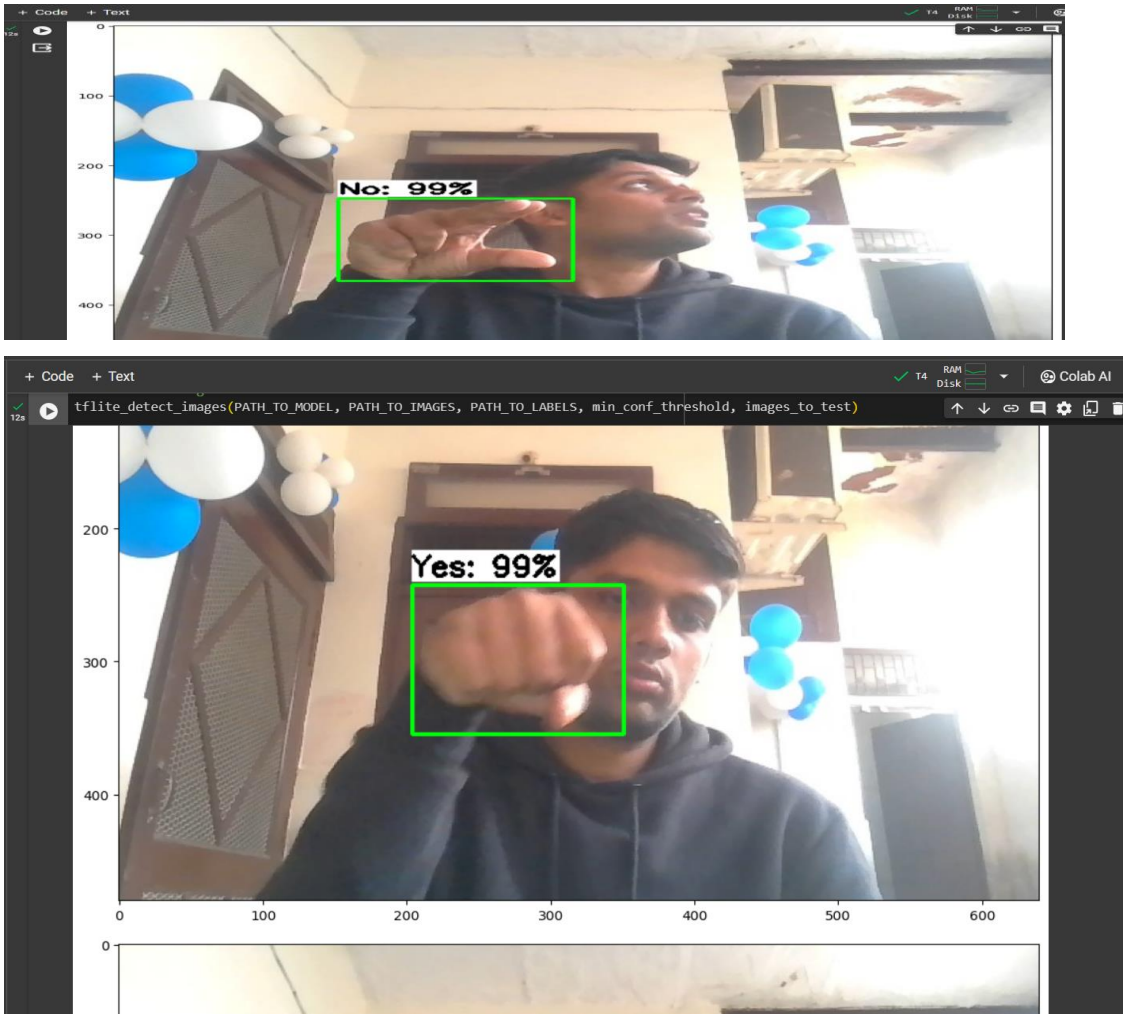


Fig. 13 Model accuracy training



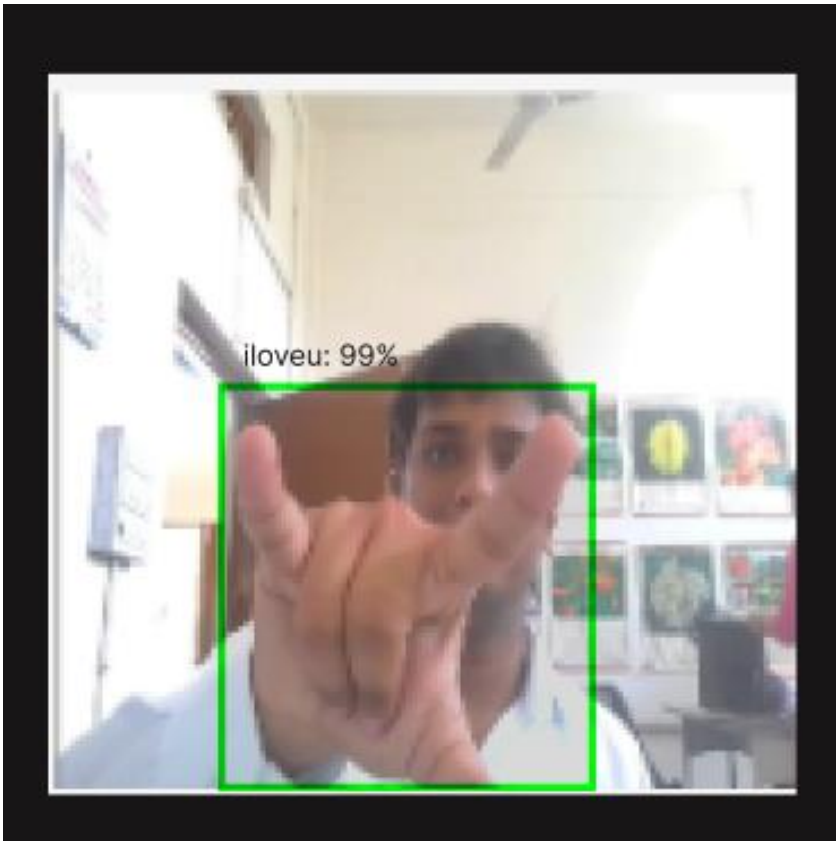


Fig. 14 Model accuracy testing

Once your machine learning model is built, i.e. trained, you need unseen data to test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Model after training is run on the testing dataset. It correctly recognises the fed hand gestures with good accuracy [4].

For model deployment, A Virtual environment is set up downloading various dependencies like python=3.9, python scripts, Tensorflow, anaconda, etc.

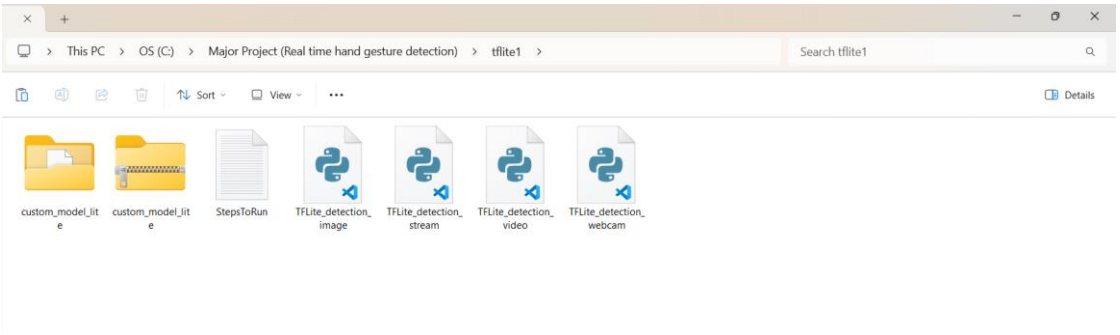


Fig. 15 Model deployment

Trained model is downloaded on the local computer after creating a virtual environment to run the python commands. Various python scripts are used to open and start webcam feed.

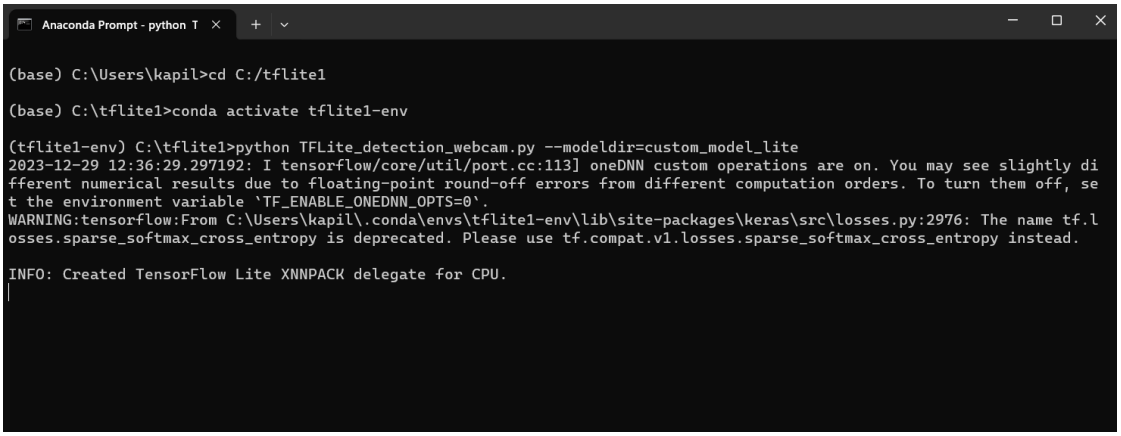


Fig. 16 Activating Conda environment

Activating the Conda environment through anaconda prompt and typing the run commands will run the model on the local computer.

3.8 Text-to-Speech

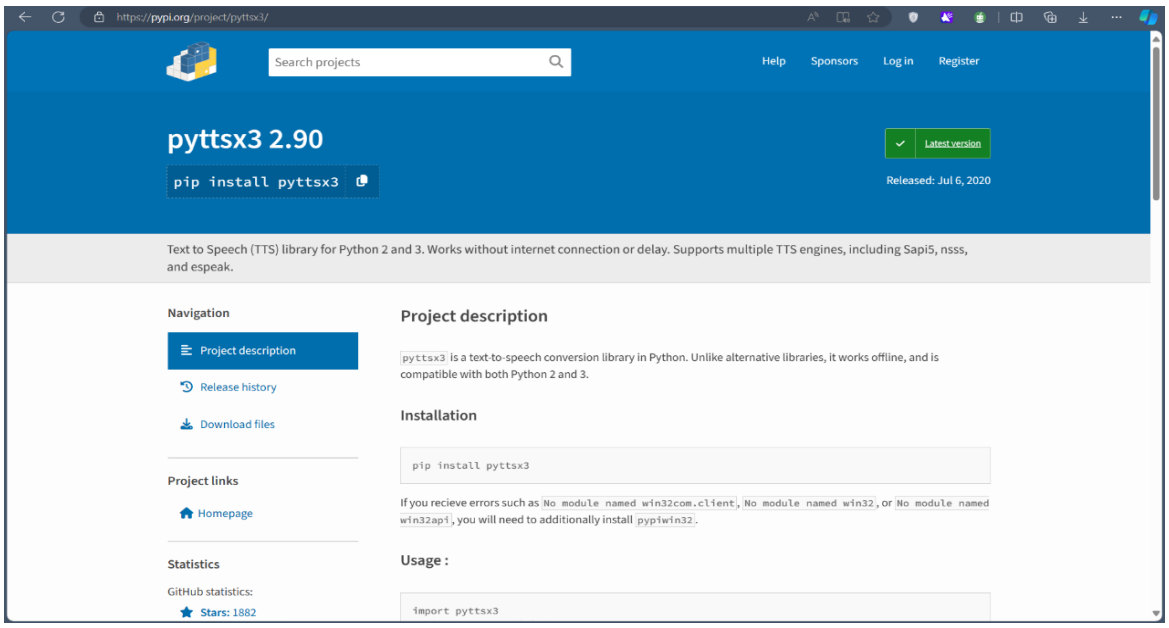


Fig. 17 pyttxs3 library page

Pyttsx3 is a Python library that facilitates text-to-speech conversion, allowing developers to integrate speech synthesis capabilities into their applications effortlessly. With pyttsx3, users can convert text strings or files into audible speech, providing accessibility features or enhancing user experiences in various applications. This library is built on top of the Speech Synthesis Markup Language (SSML) standard, which enables finer control over speech parameters such as pitch, rate, and volume. Its simplicity and versatility make it a popular choice for developers seeking to incorporate speech synthesis functionalities into their projects.

Pyttsx3 supports multiple platforms, including Windows, macOS, and Linux, making it suitable for a wide range of applications across different operating systems. Furthermore, its straightforward API and extensive documentation streamline the integration process, empowering developers to create dynamic and engaging applications with ease.

KeyWord=["Hello","Yes","No","ThankYou","ILoveYou","Good","BestOfLuck","Enjoy","Excellent","HowAreYou"]

```
index=random.randint(0,9)

KeyWord_Chosen=KeyWord[index]

print(KeyWord_Chosen)

r2=index

r1 = random.randint(0, 5)

l=p[r2][r1]

print(l)

# This will import all the widgets

# and modules which are available in

# tkinter and ttk module

# creates a Tk() object

master = Tk()

# sets the geometry of main

# root window

master.geometry("300x300")

# function to open a new window

# on a button click

def openNewWindow():

# Toplevel object which will

 # be treated as a new window

    newWindow = Toplevel(master)
```

```
    # sets the title of the
    # Toplevel widget
    newWindow.title("New Window")
    # sets the geometry of toplevel
    newWindow.geometry("200x200")
    # A Label widget to show in toplevel
    Label(newWindow,
        text ="This is a new window").pack()
label = Label(master,
            text =l)
label.pack(pady = 10)
# a button widget which will open a
# new window on button click
# mainloop, runs infinitely
mainloop()
engine = pyttsx3.init()
newVoiceRate = 160
engine.setProperty('rate',newVoiceRate)
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
# efficiencies
```

```
eff=random.randint(6,9)

efficiency=eff*10

engine.say("The maximum efficiency is")

cv2.waitKey(1000)

engine.say(efficiency)

engine.say("The keyword having maximum efficiency is")

engine.say(KeyWord_Chosen)

#cv2.waitKey(1000)

sentence=" And the sentence formed is "

engine.say(sentence)

cv2.waitKey(1000)

engine.say(l)

engine.runAndWait()
```

This program uses the above-mentioned library, the purpose of this code is to convert the chosen keyword into a meaningful sentence and at the same time display and speak it. However, we are able to perform these tasks on the ten selected hand gestures, but it holds a potential and maybe extended to perform this same task on variety of hand gestures efficiently.

4. ERRORS ENCOUNTERED

4.1 Environment issues

- There are many dependency issues related to python.
- Python uses multiple dependencies for image detection and model preparation.
- Some of the libraries and dependencies are not compatible with the system and with the model.

4.2 Model incompatibility issues

- A compatible model should be chosen very carefully based on the amount and type of dataset.
- As the number of classes and images will increase a suitable model will be used instead of the model used now.

4.3 Future development limitations

- Dataset Unavailability: For Hand Gesture Detection, there is very little data available. To overcome this problem, a universal dataset needs to be created.
- Multiple Overlapping Classes: As observed there can be multiple hand gestures that are very similar to each other, Hence, detection with accuracy becomes impossible.
- Integration with NLP: Integration of this algorithm with the algo's of NLP is quite a difficult task.
- Increasing various functionalities like sentence formation, model accuracy, text-to-speech conversion will require in-depth knowledge of the used concepts.

4.4 Supposed solution for the issues

- Analyzing the environment of Python that is resolving dependency issues.
- Reinstalling various packages of Python.
- Updating the most compatible and updated packages.
- TensorFlow object detection API error analyzing.
- Selection of a suitable model for object detection.
- Training the model and increasing the range of gestures.

5. SCOPE OF WORK

1. This project can prove to be a great asset in the coming era of technology, as the tech will advance, the underlying algorithms associated with this project will also improve [5].

2. It can be used in construction sites, where people need to interact with hand gestures.

Fig. 18 Surveying instruments

3. In the future, a glove can be made which converts hand gestures into speech.

Fig. 19 Hand glove with sensors

4. It can be a very handy tool for specially abled people, but the main problem is

scalability issues and the efficiency of these recognition techniques.

Fig. 20 Interaction with specially abled person

References

[1] S. B., " Gesture Recognition Using Tensor flow, Opencv and Python.," *Amity Journal of Computational Sciences.,* 2023 Jan 1;7(1)..

[2] G. A. M. R. S. S. Srivastava S, " Sign language recognition system using TensorFlow object detection API.," *InInternational conference on advanced network technologies and intelligent computing 2021 Dec 17 (pp. 634-646). Cham: Springer Internation.*

[3] M. S. S. P. R. A. Kumari S, " Hand gesture-based recognition for interactive human computer using tenser-flow.," *International Journal of Advanced Science and Technology. 2020;29(7):14186-97..*

[4] K. K. M. M. D. M. A. H. Abhishek B, "Hand gesture recognition using machine learning algorithms.," *Computer Science and Information Technologies. ,* 2020 Nov 1;1(3):116-20..

[5] B. N. K. R. D. G. S. K. Aggarwal A, ". Real-time hand gesture recognition using multiple deep learning architectures.," *Signal, Image and Video Processing.,* pp. :3963-71., 2023 Nov;17(8).