**HAND GESTURE TO TEXT CONVERTER – (DIVYAANUVADAK)**

A Project Report

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

### B.Tech. in Information Technology (VI Semester)

**Submitted by**

Ayushi Rajput – 2016730

Deepali – 2016736

Shivalik – 2016821

Sneha Poddar – 2016837

**Under the supervision of**

Dr. Nisheeth Joshi

Associate Professor

Department of Computer Science

**Name of the Mentor**  **Name of Coordinator(s)**

Dr. Nisheeth Joshi Prof. Saurabh Mukherjee

Associate Professor Dr. Neelam Sharma

Department of Computer Science Dr. Pooja Asopa

Dr. Deepak Kumar

Dr. Sneha Asopa

Dr. Ashok Kumar

Dr. Rahul Vijay



**Department of Computer Science**

**Banasthali Vidyapith**

**Banasthali - 304022**

**Session: 2022-23**

**Certificate**

### Certified that Ayushi Rajput, Deepali, Shivalik, Sneha Poddar has carried out the project work titled “Gesture to Text Converter - DivyaAnuvadak” from 29-12-2022 to 13-04-2023 for the award of the B.Tech (IT) VI Semester from Department of Computer Science under my supervision. The thesis embodies result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

**Nisheeth Joshi**

**Designation**: Associate Professor

**Place**: Department of Computer Science

**Date**: 13-04-2023

**ABSTRACT**

**“APAJI’s vision to educate girls,**

**why not to extend it to specially-abled!”**

Some people affected by speech and hearing impairment cannot express their feelings and emotions, hence are not able to communicate with the people. They rely on sign language which most of the people find hard to understand. Because of this, even if they are capable, they lack opportunities.

Thus, we came up with the idea of ‘DivyaAnuvadak’ which takes hand gesture as input and translate it to equivalent text which could be understood by common people. This would further help both our community and our university build a startup to educate specially-abled girls. This would eventually boost up their self-esteem and can make them independent.

We are trying to widen Apaji’s vision to educate every girl (specially-abled) that would ultimately help these Shantas to shine too.

**ACKNOWLEDGEMENT**

We express our sincere indebtedness towards our guide and mentor **Dr. Nisheeth Joshi**, Associate Professor, Department of Computer Science for his valuable guidance, suggestions and supervision throughout the work. Without his kind patronage and guidance the project would not have taken shape. We would also like to express our gratitude and sincere regards for his kind approval of the project, time to time counseling and advices.

We would also like to thank **Prof. CK Jha,** Head of Department, Department of Computer Science for providing us the opportunity to build this project. We would also like to acknowledge with much appreciation the crucial role of the project coordinators, teachers and specially lab technicians for guiding and helping throughout the project.

Lastly, kudos to the team work, relentless efforts and dedication for successful completion of the project.

# Chapter 1 SRS

**TABLE OF CONTENTS**

1. **Introduction**
   1. [Problem Definition 4](#_TOC_250019)
   2. [Purpose 4](#_TOC_250018)
   3. [Scope 4](#_TOC_250017)
   4. [Definitions, Acronyms and Abbreviations 5](#_TOC_250016)
   5. [Overview 5](#_TOC_250015)
2. **General Description**
   1. [Product Perspective 5](#_TOC_250014)
      1. [Product Function 6](#_TOC_250013)
      2. [Hardware Interface 6](#_TOC_250012)
      3. [Software Interface 6](#_TOC_250011)
      4. [Communication Interface 7](#_TOC_250010)
   2. [User Characteristics 7](#_TOC_250009)
   3. [General Constraints 7](#_TOC_250008)
   4. [Technologies used… 7](#_TOC_250007)
3. [**Specific Requirements**](#_TOC_250006)
   1. [Functional Requirements 8](#_TOC_250005)
   2. [Non-Functional Requirements 8](#_TOC_250004)
      1. [AVAILABILITY 8](#_TOC_250003)
      2. [SECURITY 8](#_TOC_250002)
      3. [RELIABILITY… 8](#_TOC_250001)
      4. [PORTABILITY 8](#_TOC_250000)
      5. MAINTAINABILITY 8
4. INTRODUCTION

**Problem Definition**

The main problem that our project aims to solve is the communication problem between speech- impaired people and the others. As those people cannot express themselves with the words, they have many difficulties during their daily life. Since almost all of the normal people do not know sign language and cannot understand what speechless people mean by their special language, tasks such as shopping, settling affairs at a government office are so difficult that speech-impaired people cannot handle by their own.

This problem is very broad and many solutions and implementation can be raised. A solution could be teaching sign language to everyone, yet it is very obvious to be an inefficient and even non- applicable one. Since speechless people can understand other people by lip-reading (or even hearing since being speechless does not mean being deaf also) the main problem is that normal people do not understand them. Thus, a more suitable solution should be in the manner that makes speechless people’s language understandable by the other people. For the language of speechless people to be understood by others, we need the gestures performed by speech-impaired people to be recognized and turned into a form that the others can understand.

**Purpose**

The aim of this document is to specify the features, requirements of the final product and the interface of ISL recognition using Tensorflow Object Detection API.

It will explain the scenario of the desired project and necessary steps in order to succeed in the task. To do this throughout the document, overall description of the project, the definition of the problem that this project presents a solution and definitions and abbreviations that are relevant to the project will be provided.

The preparation of this SRS will help consider all of the requirements before design begins, and reduce later redesign, recoding, and retesting. If there will be any change in the functional requirements or design constraint’s part, these changes will be stated by giving reference to this SRS in the following documents.

**Scope**

The project which is going to be presented in this document is called Divya-Anuvadak. This application is planned to be used by speechless people in order to ease their life, and also government offices that should serve all of its citizens equally, private companies that want to reach and serve speechless people as well, co-operations and foundations which aims to help speech- disordered people.

**Definitions, Acronyms and Abbreviations**

SRS: Software Requirements Specification ISL: Indian Sign Language

XML: Extensible Markup Language (XML) which is a programming language with markers VS Code: Visual Studio Code

IDE: Integrated Development Environment RAM: Random Access Memory

HDD: Hard Disk Drive

GPU: Graphics Processing Unit

**Overview**

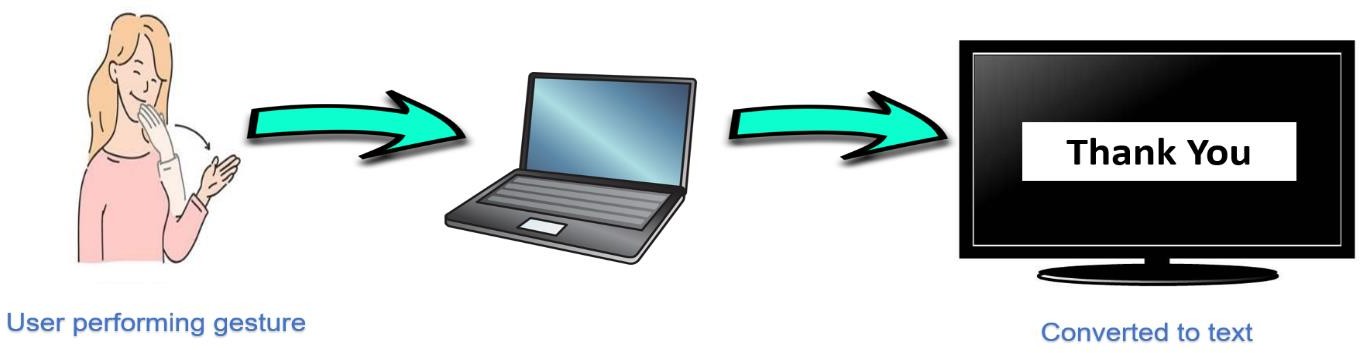
In the following section of this document, we will focus on overall description of the system. This part explains the product perspective, product functions, characteristics, constraints and technologies used in the desired application. Third part of the document explains the specific requirements of the system. Specific requirements are divided into two parts namely functional requirements and non-functional requirements of the system.

1. THE OVERALL DESCRIPTION

**Product Perspective**

Divya-Anuvadak will serve to speech-impaired people so that they won’t face difficulty in communication by providing their gestures converting to the text for people who does not know ISL. Our project is different from the existing systems because it focuses on the word recognition through gestures while the existing systems focuses on letter recognitions through hand signs which is very slow and makes having an actual conversation quite difficult.

This system will consist of one web portal. This web portal will be used for converting gestures to its corresponding text. The predefined gestures will be stored in the database and will be compared with the original gesture performed by the user.



**Product Function**

* + - * Capturing the gestures made by the sign languages user through an image sensor.
      * Tracking the gestures through OpenCV by identifying feature points.
      * Pre-Processing the captured data.
      * Feeding the data to the model.
      * Predicting the word based on processed model.
      * Selecting the word of highest possibility.
      * Converting the word into text and displaying it on the screen.
    1. **Hardware Interface**
       - **SERVER SIDE**:

**RAM :** 8GB (minimum)

**HDD :** 1 TB or more

**Processor :** Intel® i3 or faster (2–4 GHz)

**GPU :** 2 GHz or more

* + - * **CLIENT SIDE**:

**RAM :** 4 GB (minimum)

**HDD :** 512 GB or more

**Processor :** Intel® Pentium or faster (2-4 GHz)

**Camera :** 720p

* + 1. **Software Interface**
       - **SERVER SIDE**:

**Operating System :** Windows 10 or 11

**Web Server :** Apache Tomcat (v10)

**IDE :** Jupyter Notebook (v6.4.8), VS Code (v1.74.3)

**Programming Language :** Python

* + - * **CLIENT SIDE**:

**Operating System :** Windows 7 or higher versions

**Web Browser :** Any Browser

**Library :** Streamlit

**Communication Interface**

HTTP and TCP/IP Protocol is used as a Communication Interface.

* 1. **User Characteristics**
* User is required to have basic knowledge of computing and internet surfing.
* User should have good knowledge of ISL and English.
  1. **General Constraints**
* Hardware limitation on mobile devices as mobile devices have very limited hardware power.
* Full-fledged translation is not possible because the English language has more than 1,000,000 words.
* Only one-way communication is possible through this project.
* Fast-paced communications are not possible as the data captured requires some time to process and predict the words and the hardware is not possible to process that fast.
  1. **Technologies Used**
* Python
* Jupyter Notebook
* Cv2(OpenCV)
* Keras
* VS Code
* TensorFlow (as Keras uses TensorFlow in backend and for image pre-processing)
* Streamlit

1. SPECIFIC REQUIREMENTS

**Functional Requirements**

We describe the functional requirements by giving various Use Cases:

* **USE CASE 1:** Webcam started by user.
* **USE CASE 2:**Text is displayed on the screen.

Start Webcam

Display Results

User

**Use Case Diagram**

**Non-Functional Requirements**

**Availability**

This web application will be available free on the internet. The users do not require to login so it is user friendly.

**Security**

This web application does not ask for any personal information from the user. Hence it is a secure application.

**Reliability**

The application will be 99% reliable and will produce correct speech for the corresponding gestures.

**Portability**

This project is a web application so it will be available online and can be accessed from any device having internet connectivity, e.g.: laptops, computers, mobile devices etc.

**Chapter 2 SDS**

**TABLE OF CONTENTS**

1. **Introduction**

* Purpose 11
* Scope 11
* Terms, Definitions, Acronyms and Abbreviations 11
* Overview 11

1. **System Architectural Design**

* High Level Design 12
* Detailed Description of Components
* Structure Chart 13
* Data Flow Diagram 13-14
* Flow Chart 15-16
* Use Case Diagram 17-18
* Sequence Diagram 19
* Activity Diagram 20

1. **Data Design**

* Dataset Description 21

1. **User Interface Design**

* Description of user interface 22
* Screen image 23
* Objects and actions 23

1. **Types of Tests** 24
2. **References** 25
3. **Introduction**
   1. **Purpose**

This Software Design Document provides the design details of

**“DivyaAnuvadak”**.

This would help speech and hearing impaired people to communicate with the world.

* 1. **Scope**

**“DivyaAnuvadak.com”** is a web-based service that would allow speech and hearing impaired people to convert their hand gestures to equivalent text.

This would not merely help them in communication but also in their education which would further boost their self-esteem.

This would ultimately aid Banasthali to take a step ahead and inaugurate a new department for such specially-abled people.

* 1. **Terms,Definitions, Acronyms, and abbreviations**

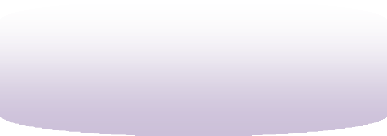
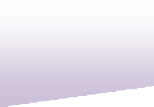
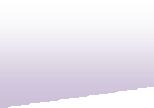
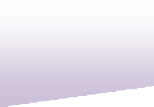
Admin – Administrator DFD – Data Flow Diagram

* 1. **Overview**

The rest of the SDS consists of the following parts –

* + - System Architectural Design
      * High Level Design
      * Structure Chart
      * DFD
      * Usecase Diagram
      * Sequence Diagram
      * Activity Diagram
    - Data Design includes Dataset Description
    - Detailed description of components
    - User Interface Designs
    - Testing Strategies

1. **System Architectural Design**



* 1. **High – Level Design**

Image Pre-Processing

Vision

Data

Feature-Extraction

System Testing Vision

System Training

Finish

System Database

Retrieving

Updating

**Detailed description of components**

* + 1. **Structure Chart**

**Main**

Gesture Window

Text Window

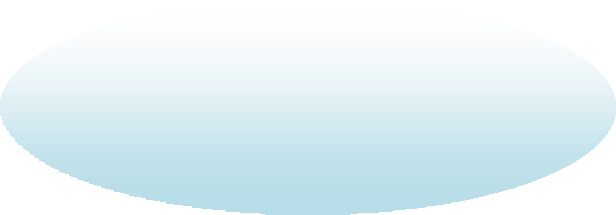
Display



Verified

Display text

**a 0-Level DFD**



Information

Train

View Home Page

Update dataset

DivyaAnuvadak.com

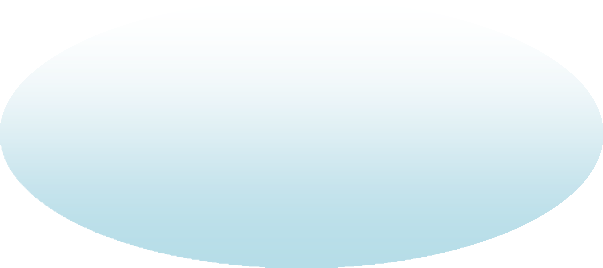
Admin

User

Database

**b 1-Level DFD**

text



Output

Input gestures

Invalid gesture

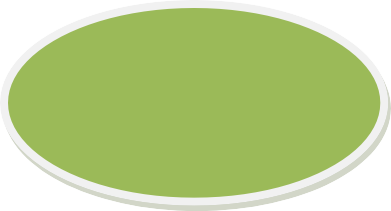
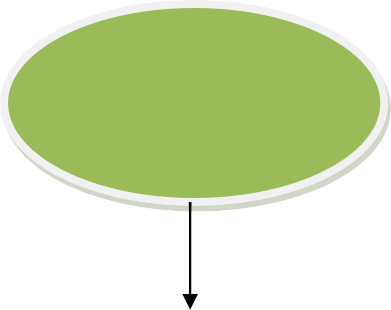
DivyaAnuvadak.com

Valid gesture

Database

User

* + 1. **a Backend Flowchart**



**Start**

**End**

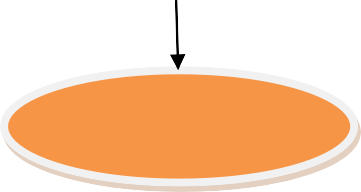
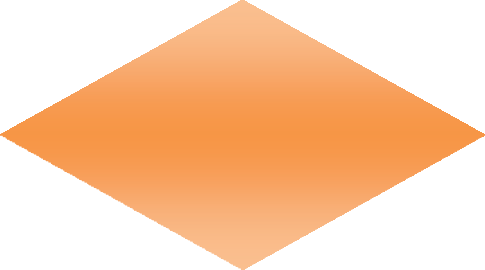
Test system

Train system

Hand region segmentation

Dataset collection

**b Frontend Flowchart**



**Start**

Hand Gestures

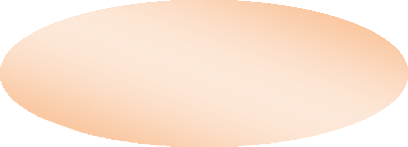
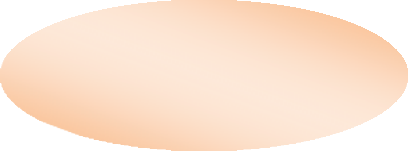
Recognition

**End**

Text Output

Hand image input from web camera

**Usecase Diagram**



Start webcam

Text output

User

**Table : Usecase Scenario**

|  |  |
| --- | --- |
| **Usecase Name** | **Sign Language Recognition** |
| Participating actors | User |
| Flow of events | Start the system(u)  Capturing video(s) Capture gestures(s) Translate gestures(s) Extract features(s) Match features(s) Recognizing gestures(s)  Display result |
| Entry condition | Run the code |
| Exit condition | Displaying the text label |
| Quality requirements | Cam pixels clarity, good  light condition |

**Sequence Diagram**



Images()

Feature extraction()

Database

System

Result()

Webcam

User

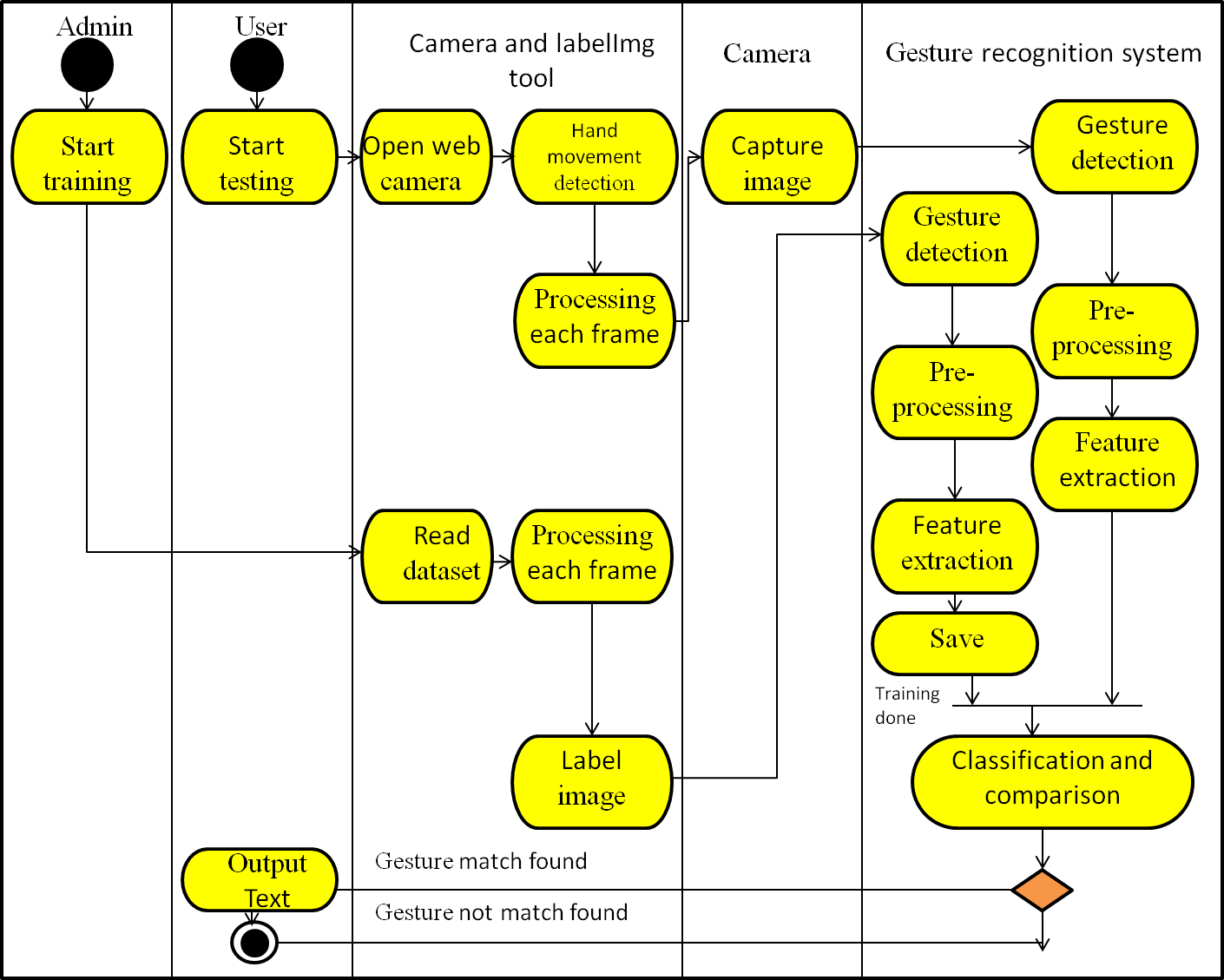
Matching result()

Feature matching()

Video Stream()

Hand detection()

**Activity diagram**



1. **Data Design**
   1. **Dataset description**

* Hello
* Thank You
* Yes
* No
* I love you

**CODE**

**Capture Images**

import cv2

import os

import time

import uuid

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

labels=['hello','iloveyou','no','thankyou','yes']

number\_imgs=30

for label in labels:

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

    cap=cv2.VideoCapture(0)

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

    time.sleep(10)

    for imgnum in range(number\_imgs):

        ret, frame=cap.read()

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

        cv2.imwrite(imgname, frame)

        cv2.imshow('frame', frame)

        time.sleep(2)

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

            break

    cap.release()

**0. Setup Paths**

WORKSPACE\_PATH = 'Tensorflow/workspace'

SCRIPTS\_PATH = 'Tensorflow/scripts'

APIMODEL\_PATH = 'Tensorflow/workspace/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/'

# 1. Create Label Map

labels = [

    {'name':'hello', 'id':1},

    {'name':'iloveyou', 'id':2},

    {'name':'no', 'id':3},

    {'name':'thank you', 'id':4},

    {'name':'yes', 'id':5}

]

print(ANNOTATION\_PATH)

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

    for label in labels:

        f.write('item { \n')

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

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

        f.write('}\n')

import pandas as pd

import tensorflow as tf   #.compat.v1

# 2. Create TF records

!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'}

# 3. Download TF Models Pretrained Models from Tensorflow Model Zoo

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

# 4. Copy Model Config to Training Folder

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

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

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

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

# 5. Update Config For Transfer Learning

import tensorflow as tf

from object\_detection.utils import config\_util

from object\_detection.protos import pipeline\_pb2

from google.protobuf import text\_format

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

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

pipeline\_config = pipeline\_pb2.TrainEvalPipelineConfig()

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

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

pipeline\_config.model.ssd.num\_classes = 5

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

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

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

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

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

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

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

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

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

    f.write(config\_text)

# 6. Train the model

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

# 7. Load Train Model From Checkpoint

import os

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

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

from object\_detection.builders import model\_builder

# Load pipeline config and build a detection model

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

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

# Restore checkpoint

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

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

@tf.function

def detect\_fn(image):

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

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

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

    return detections

# 8. Detect in Real-Time

import cv2

import numpy as np

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

# Setup capture

cap = cv2.VideoCapture(0)

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

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

while True:

    ret, frame = cap.read()

    image\_np = np.array(frame)

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

    detections = detect\_fn(input\_tensor)

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

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

                  for key, value in detections.items()}

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

    # detection\_classes should be ints.

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

    label\_id\_offset = 1

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

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

                image\_np\_with\_detections,

                detections['detection\_boxes'],

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

                detections['detection\_scores'],

                category\_index,

                use\_normalized\_coordinates=True,

                max\_boxes\_to\_draw=1,

                min\_score\_thresh=.5,

                agnostic\_mode=False)

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

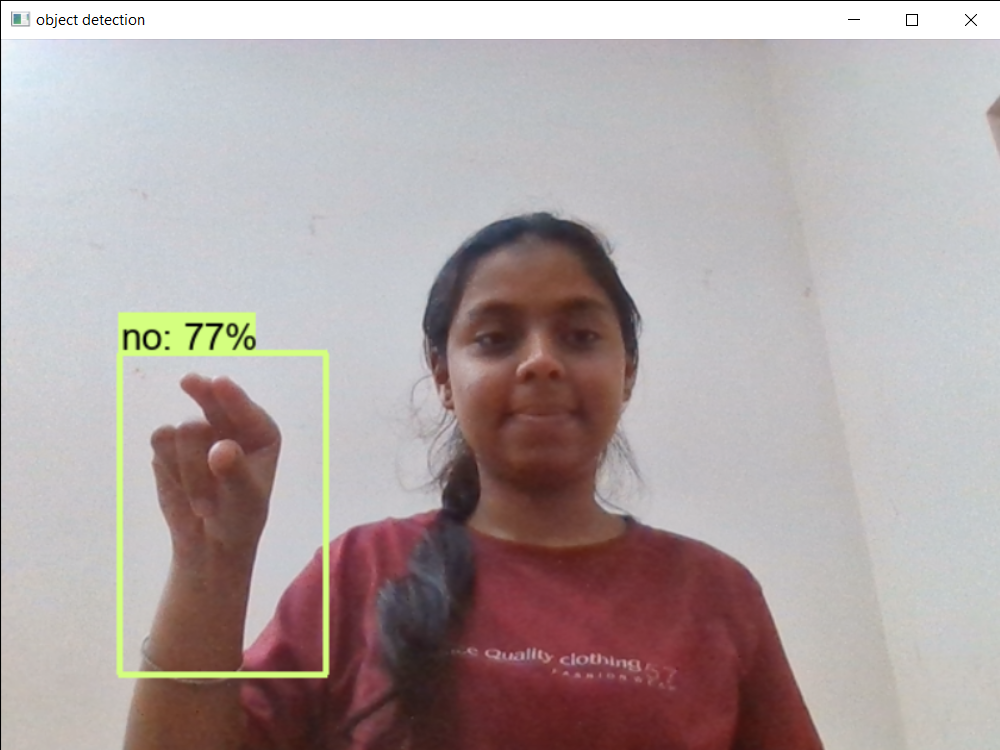
    if cv2.waitKey(1) & 0xFF == ord('q'):

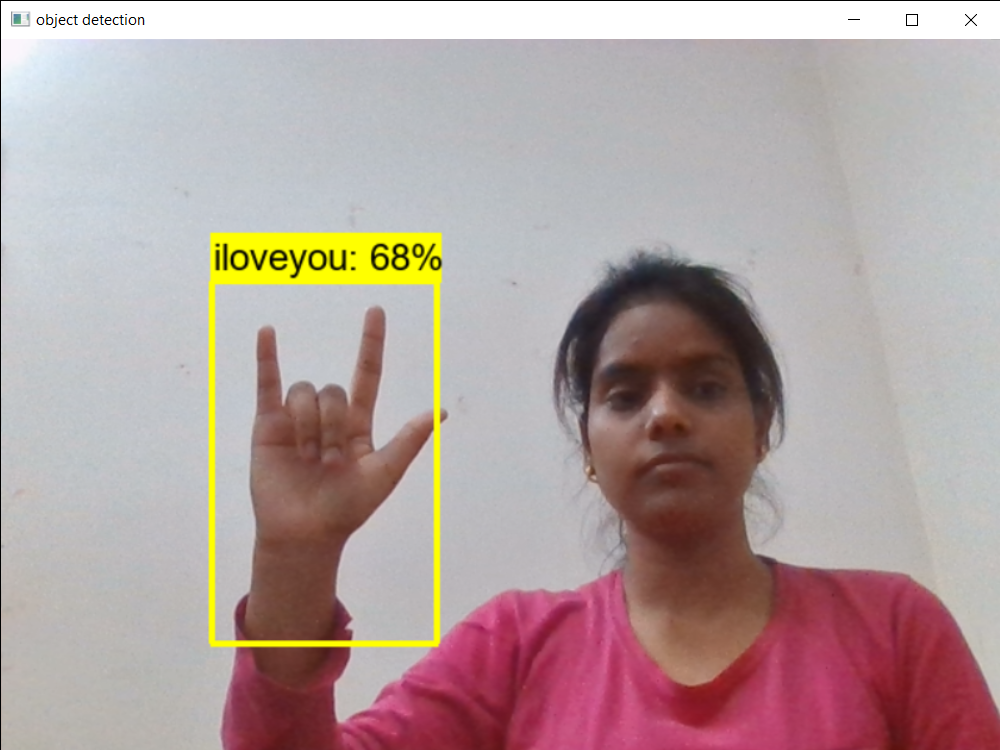
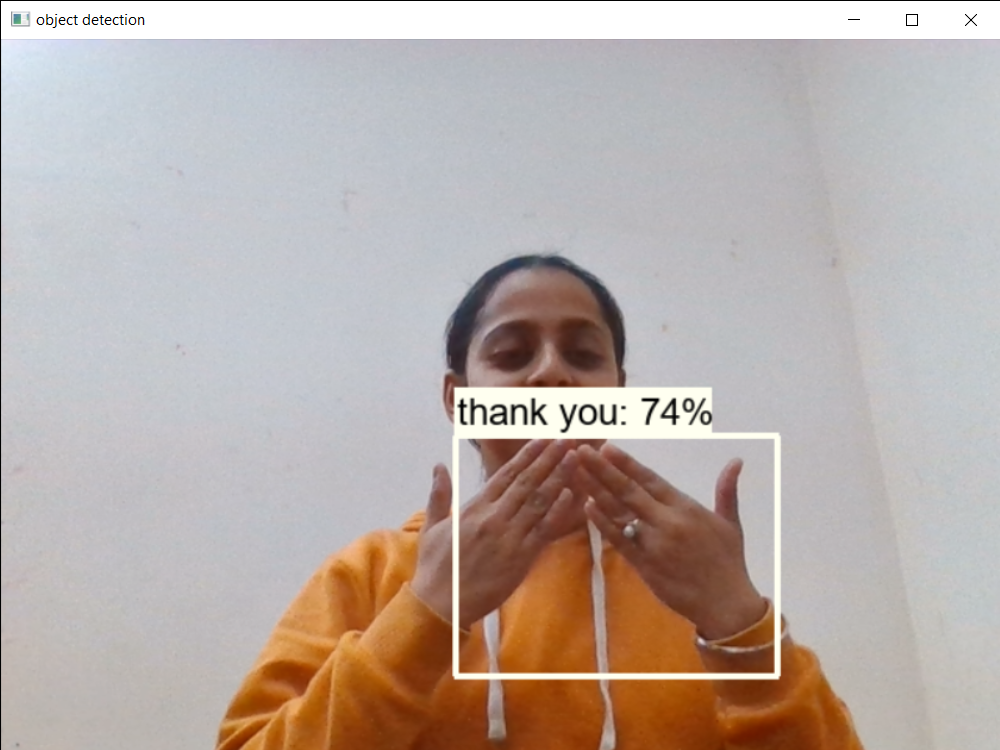
        break

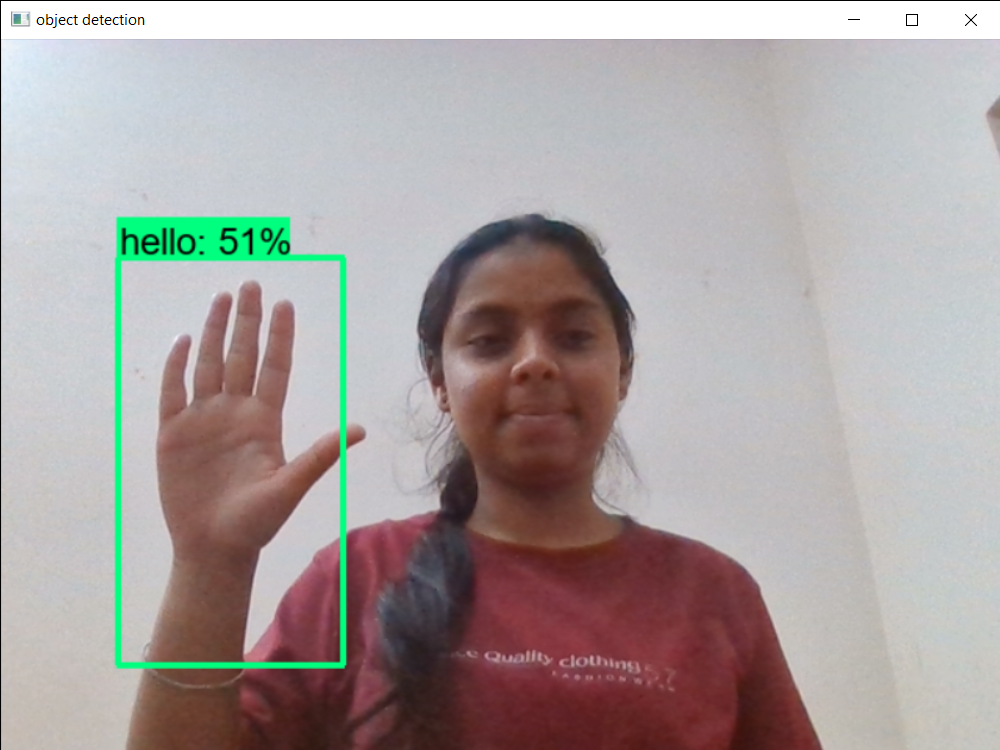
cap.release()

cv2.destroyAllWindows()

**Test Cases**

****





1. **User Interface Design**
   1. **Description of the user interface**

**Input –** Hand Gesture

**Output –** Corresponding Text.

**Processing Details**

* The hand gestures given by the user through web-camera will be processed.
* Then it will be checked whether the input is valid or not.
* Eventually, the system would generate the corresponding output text in the text window.

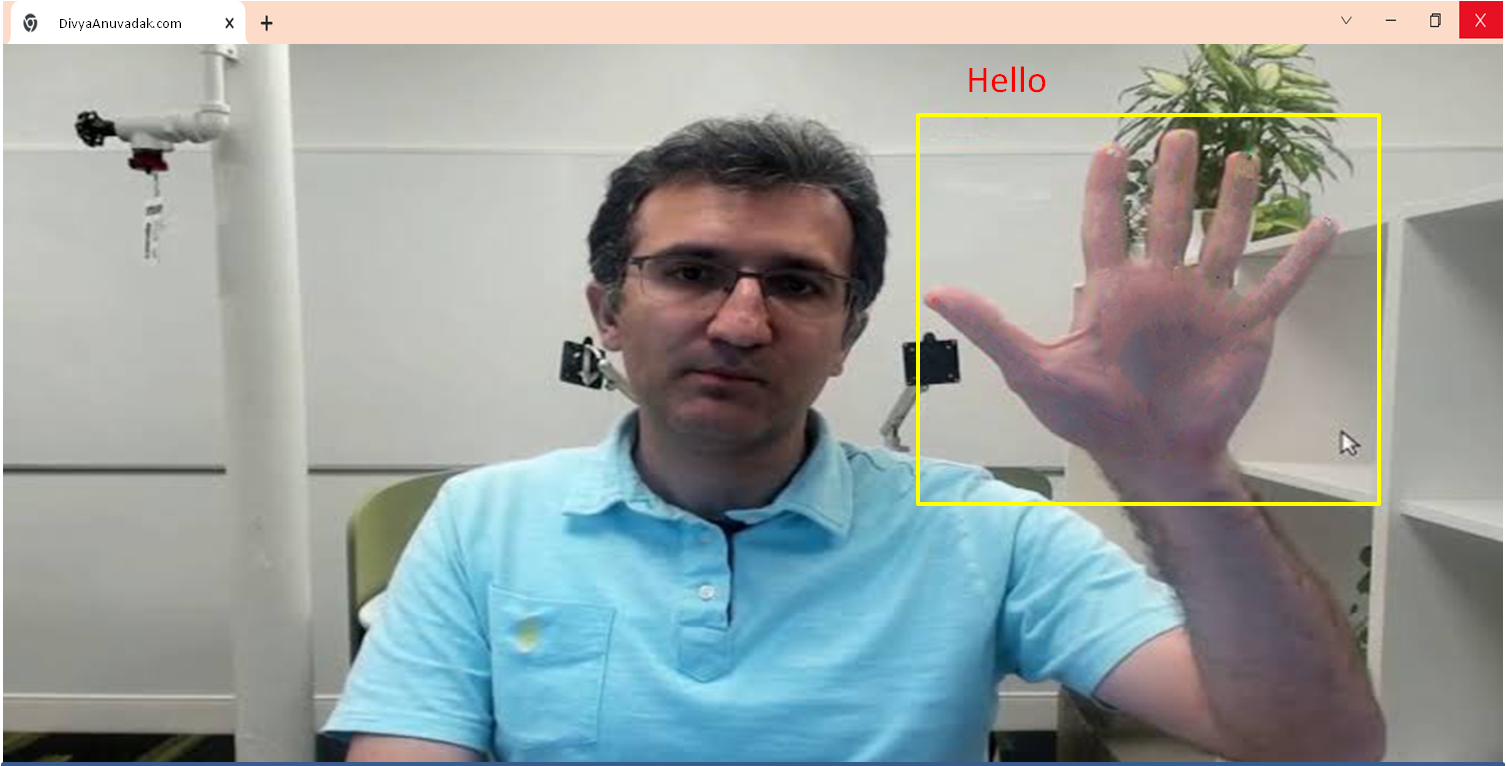
**Description of Component Gesture Window**

In this window, user would provide hand gesture as input which would be further verified and processed by the system.

**Text Window**

In this window, corresponding text will be displayed with respect to the valid hand gestures.

**Screen Image**

****

**Objects and actions**

Object – Textbox

Purpose – Hand Gesture to text

Action – Gesture

Response – Corresponding text output of gesture

1. **Types of Tests**
   1. **Unit Tests**

This test is applied on each of the module to find whether or not each module is properly working or not.

* 1. **System Tests**

Black box testing or system testing involves the external working of the software opposite to white box testing which checks for the internal working or code of a software application.

* 1. **Integration or Regression Tests**

After unit test integration testing is done to test if the entire system works together correctly, as a whole. It tries to detect integration related defects by ensuring that the different modules of the software work together correctly and provides the expected results when they are used with each other.

While regression test which is used as synonym of integration test involves retesting the application using the previously executed test data to check for any defects. In short, regression testing tries to ensure that a newly added feature or modified code does not break any functionality already working in the existing system. It is done on each build after it passed all unit tests.

* 1. **Stress Tests**

Here we provide our software with some unfavorable conditions to check how they perform in such conditions. It is particularly used to determine error handling under extremely heavy load conditions to ensure that the system would not crash under crunch situations.

* 1. **Acceptance Tests and Staging**

This test will ensure that the quality of the product is not compromised and also helps in determining to what degree this software meets end users’ approval.

Its implementation can be done by engaging end users’ in the testing process an gather their feedback.

## References

1. Pressman Roger S., Software Engineering “A Practitioner’s Approach” Fifth Edition, McGraw-Hill Publication, 2000.
2. IEEE STD 830-1998, IEEE Recommended Practice for Software Requirement Specifications.
3. "A Concise Introduction to Software Engineering" by Pankaj Jalota
4. https://ieeexplore.ieee.org/document/720574