SIGN LANGUAGE RECOGNITION

Project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

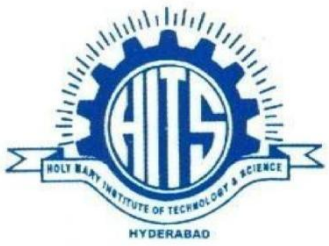
BY

KATLA VARSHITHA (18C91A0546)
GUNDU VAMSHI KRISHNA (18C91A0529)
K.KARTHIK REDDY (18C91A0535)

Under the Esteemed guidance of

Mrs.SreeLakshmi M.Tech

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

(COLLEGE OF ENGINEERING) 11

(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)

Bogaram (V), Keesara (M), Medchal District -501 301.

2021- 2022

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

(COLLEGE OF ENGINEERING)

(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)

Bogaram (V), Keesara (M), Medchal Dist-501301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled "SIGN LANGUAGE RECOGNITION" is being submitted by KATLA VARSHITHA (18C91A0546), GUNDU VAMSHI KRISHNA (18C91A0529), K.KARTHIK REDDY (18C91A0535) in Partial fulfillment of the academic requirements for the award of the degree of Bachelor of Technology in "COMPUTER SCIENCE AND ENGINEERING" HOLY MARY INSTITUTE OFTECHNOLOGY & SCIENCE, JNTU Hyderabad during the year 2021- 2022.

INTERNAL GUIDE

Mrs. SREE LAKSHMI M.Tech
Assistant Professor
Dept. of Computer Science & Engineering.

HEAD OF THE DEPARTMENT

DR .B.NARSIMHA M.Tech, Ph.D.
Professor & HoD
Dept. of Computer Science & Engineering

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, who's constant guidance and encouragement crowns all effort with success.

I take this opportunity to express my profound gratitude and deep regards to My Guide **Mrs. Sree Lakshmi, Assistant Professor**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science for his / her exemplary guidance, monitoring and constant encouragement throughout the project work.

My special thanks to **Dr. B. Narsimha, Head of the Department**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science who has given an immense support throughout the course of the project.

I also thank to **Dr. P. Bhaskara Reddy,** the **honorable Director** of my college Holy Mary Institute of Technology & Science for providing me the opportunity to carry out this work.

At the outset, I express my deep sense of gratitude to the beloved **Chairman A. Siddartha Reddy** of **Holy Mary Institute of Technology & Science**, for giving me the opportunity to complete my course of work

I am obliged to **staff members** of Holy Mary Institute of Technology & Science for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Last but not the least I thank **ALMIGHTY** and My **Parents**, and **Friends** for their constant encouragement without which this assignment would not be possible.

KATLA VARSHITHA **(18C91A0546)**

GUNDU VAMSHI KRISHNA **(18C91A0529)**

K.KARTHIK REDDY **(18C91A0535)**

DECLARATION

This is to certify that the work reported in the present project titled **"SIGN LANGUAGE RECOGNITION"** is a record of work done by me in the Department of Computer Science & Engineering, Holy Mary Institute of Technology and Science.

No part of the thesis is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text the reported are based on the project work done entirely by me not copied from any other source.

KATLA VARSHITHA **(18C91A0546)**

GUNDU VAMSHI KRISHNA **(18C91A0529)**

K.KARTHIK REDDY **(18C91A0535)**

ABSTRACT

This project aims to develop a sign detector, which detects hand gestures of sign language, which is used by deaf and dumb people to exchange information between their own community and with other people. Knowing sign language is not something that is common to all, this project can be very helpful for the deaf and dumb people in communicating with others ,this can also be extended to automatic editors. Three dimensional spaces and the hand movements are used to identify meanings. The goal of this project is to identify and review the sign language recognition with an efficient output.Our project provides 95% accuracy for 10 numbers(dataset).

CONTENTS

LIST OF TABLES I

LIST OF FIGURES II

1.INTRODUCTION

1.1.Problem statement:

To develop an ML - based application which can translate hand gestures to text. It is a camera based sign language recognition system that would be useful for the deaf and dumb for converting sign language gestures to text. The data acquisition, datapreprocessing and transformation, feature extraction, classification and results obtained are examined. We use small dataset for pre-training. The project flow goes by collecting images using OpenCV ,Python and labelling them using label image package and then building a sign language detector using TensorFlow Object Detection API to detect the sign language in real time.

1.2.Objectives:

The project aims at building a machine Learning model that will be able to classify the various hand gestures used for fingerspelling in sign language, we will develop a sign detector, which detects the signs and hand gestures. Sign gestures can be classified as static and dynamic.

In our project we basically focus on producing a model which can recognise Fingerspelling based hand gestures in order to form a complete word by combining each gesture.

1.3.Motivation:

The reason behind the thought of development of a sign language translator , as it bridges the communication gap between normal people and deaf and dumb people.

Sign language recognition (SLR) system takes an input expression from the hearing impaired person gives output to the normal person in the form text.

It is challenging for a normal person to understand the exact context of symbolic expressions of deaf and dumb people without an interpreter.

1.4.Existing system:

There are few similar applications, which are mainly used to translate text to sign language ,

Wecapable.com: This tool easily converts English text into sign language symbols. It isuseful for both teaching and learning American sign language.

Hand Talk: Hand Talk app is a pocket translator that automatically translates orallanguages

both in text and audio to sign languages, such as English to ASL .

1.5.Proposed system:

The project aims at building an ML - based application which can translate hand gestures to text.It is a camera based sign language recognition system that would be useful for the deaf and dumb for converting sign language gestures to text.
The data acquisition, data preprocessing and transformation, feature extraction, classification and results obtained are examined.
We use small dataset for pre-training.
The project flow goes by collecting images using OpenCV ,Python and labelling them using label image package and then building a sign language detector to detect the sign language in real time.

2.LITERATURE SURVEY

2.1.Existing System:

Literature review of our proposed system shows that there have been many explorations done to tackle the sign recognition in videos and images using several methods and algorithms.
There are few similar applications, which are mainly used to translate text to sign language,

Wecapable.com: This tool easily converts English text into sign language symbols. It is useful for both teaching and learning American sign language.
Hand Talk: Hand Talk app is a pocket translator that automatically translates oral languages both in text and audio to sign languages, such as English to ASL .
EnableTalk: Sensor Enabled Gloves The EnableTalk Gloves, developed by a group of students in Ukraine, are fitted with a variety of sensors to detect the position of the fingers and palm in space. This data is transmitted through the controller fitted on the back of the glove, via Bluetooth to a mobile device. The mobile device contains the Microsoft Speech API and Bing API in order to convert the sign language to text and speech. A major disadvantage of this system is its cost. It also requires a variety of hardware components to bring about the translation. C. Sign Language Rings Sign Language Ring is designed by a group of students from the Asian University and is the recipient of the Red Dot Design Award. The Buddhist prayer beads are an inspiration for this device. Though conceptual, it shows a great future potential. This device comprises of 3 rings on each hand and a bracelet. The rings have motion sensors which are responsible to track the movement of the fingers of the person wearing them and for translating the sign language depicted to spoken language. The word in the spoken language is then displayed on the LED Screen present on the bracelet and is also played by the speaker and amplified by the microphone. Both, the speaker and the microphone are a part of the bracelet.

Sign Mobiles: Android Application The sign mobile application makes use of three technologies- Video Relay Service (VRS), Outfit 7 and Mimix, in order to convert the sign language into textual and audible output. It makes use of mobile gesture recognition and also supports remote text-to-sign conversion and vice versa. However it is an extremely complex application from the developer's point of view and is also platform dependent.

Siming proposed a system having a dataset of 40 common words and 10,000 sign language images. To locate the hand regions in the video frame, Faster R-CNN with an embedded RPN module is used. It improves performance in terms of accuracy. Detection and template classification can be done at a higher speed as compared to single stage target detection algorithm such as YOLO. The detection accuracy of Faster R-CNN in the paper increases from 89.0% to 91.7% as compared to Fast-RCNN.

A 3D CNN is used for feature extraction and a sign-language recognition framework consisting of long and short time memory (LSTM) coding and decoding network are built for the language image sequences.On the problem of RGB sign language image or videorecognition in practical problems , the paper merges the hand locating network, 3D CNN feature extraction network and LSTM encoding and decoding to construct the algorithm for extraction. This paper has achieved a recognition of 99% in common vocabulary dataset.

The paper by M. Geetha and U. C. Manjusha, make use of 50 specimens of every alphabets and digits in a vision based recognition of Indian Sign Language characters and numerals using B-Spine approximations. The region of interest of the sign gesture is analysed and the boundary is removed . The boundary obtained is further transformed to a B-spline curve by using the MaximumCurvature Points(MCPs) as the Control points. The B-spline curve undergoes a series of smoothening process so features can be extracted. Support vector machine is used to classify the images and the accuracy is 90.00%.

The research done, which made use of YCbCr skin model to detect and fragment the skin region of the hand gestures. Using Principal Curvature based Region Detector, the image features are extracted and classified with Multi class SVM, DTW and non-linear KNN. A dataset of 23 Indian Sign Language static alphabet signs were used for training and 25 videos for testing. The experimental result obtained were 94.4%for static and 86.4% for dynamic.

A low cost approach has been used for image processing . The capture of images was done with a green background so that during processing, the green colour can be easily subtracted from the RGB colourspace and the image gets converted to black and white. The sign gestures were in Sinhala language. The method that thy have proposed in the study is to map the signs using centroid method. It can map the input gesture with a database irrespective of the hands size and position. The prototype has

correctly recognised 92% of the sign gestures.

Another research paper on Action recognition topic by the author J.Carriera [11] shares some similarities to sign gesture recognition .He used a transfer learning method for his research As his pre-trained dataset, he used both ImageNet[12] and Kinetic Dataset [9] . After training the pertained models using another two datasets namely UCF-101 [13] and HMDB-51 [14], he then merged the RGB model, flow model, pre-trained Kinetic and pre-trained ImageNet.The accuracy he got on UCF-101 dataset is 98.0% and on HMDB-51 is 80.9%

Limitations: These applications lack the option of translating a sign language gesture to a text. So these applications cannot be used by deaf and dumb community to interact with people who are not familiar with sign language.

Some applications are extremely complex from the developer's point of view and is also platform dependent.

A major disadvantage of some systems is its cost. It also requires a variety of hardware components to bring about the translation.

2.2.Proposed System:

With the help of literature survey done we realized the basic steps in hand gesture recognition are :-

- Data acquisition
- Data preprocessing
- Feature extraction
- Gesture classification

1.Dataset creation : We have created our own dataset by taking live feed from the video cam to detect the hand. These gestures will be saved in a directory. There will be two folders in this directory namely test and train ,each containing 10 folders containing captured images.

2.Training a CNN on the captured dataset : Now on the created data set we train a CNN a convolutional neural network (CNN, or ConvNet) is a class of deep neural network most commonly applied to analyze visual imagery,They are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs,We fit the model and save it to be used in the last module ,the optimization algorithm used is SGD (stochastic gradient descent, that means the weights are updated at every training instance).After compiling the model we fit the model on train batches for 10 epochs.

3.Prediction : In this, we create a bounding box for ROI detection , Region of interest pooling (also known as RoI pooling) is an operation widely used in object detection tasks using convolutional neural networks. For example, to detect multiple cars and pedestrians in a single image.As we did in creating the dataset.

This is done for identifying any foreground object .We load the previously saved model consisting of the hand as an input to the model for prediction.

2.3.Applications:

- **Deaf communities and Deaf culture**

When Deaf people constitute a relatively small proportion of the general population, Deaf communities often develop that are distinct from the surrounding hearing community. These Deaf communities are very widespread in the world, associated especially with sign languages used in urban areas and throughout a nation, and the cultures they have developed are very rich.

- **Teaching country's sign languages in schools**

Due to much exposure to sign language-interpreted announcements on national television, more schools and universities are expressing interest in incorporating sign language.

- **Bridges the communication gap**

The reason behind the thought of development of a sign language translator , as it bridges the communication gap between normal people and deaf and dumb people.

Sign language recognition (SLR) system takes an input expression from the hearing impaired person gives output to the normal person in the form text.

It is challenging for a normal person to understand the exact context of symbolic expressions of deaf and dumb people without an interpreter.

2.4.Summary:

Hand gesture recognition provides an intelligent, natural, and convenient way of human–computer interaction. Sign language recognition (SLR) and gesture-based control are two major applications for hand gesture recognition technologies . SLR aims to interpret sign languages automatically by a computer in order to help the deaf communicate with hearing society conveniently. Since sign language is a kind of highly structured and largely symbolic human gesture set, SLR also serves as a good basic for the development of general gesture-based HCI.Dumb people use hand signs to communicate, hence normal people face problem in recognizing their language by signs made. Hence there is a need of the systems which recognizes the different signs and conveys the information to the normal people. No one form of sign language is universal as it varies from region to region and country to country and a single gesture can carry a different meaning in a different part of the world.

Sign language is a means of communication among the deaf and mute community. It emerges and evolves naturally within hearing impaired community. The sign language differs from people to people and from region to region. There is no particular standardized sign language they follow. As the sign language becomes a communicating language to the deaf and mute people it can be made easier for them to communicate with normal people too. Gesture recognition is an efficient way through which the movements of hand can be recognized easily. According to a survey made by some organizations there are many people who are deaf and mute and find it difficult to communicate. Gestures are made by them as an easy way of communication.

3.SOFTWARE REQUIREMENT SPECIFICATIONS

3.1.Software Requirements:

- Python (3.7.4)
- IDE (Jupyter) -
- NumPy (1.16.5)
- cv2 (OpenCV) (3.4.2)
- Keras (2.3.1)
- TensorFlow (2.0.0)

3.2.Hardware Requirements:

- RAM 4 GB
- Free Space 1 GB
- Web Cam (5 MP preferable)
- HardDisk 500GB(Min.)
- Processor I3

4.SYSTEM DESIGN

4.1.System Architecture:

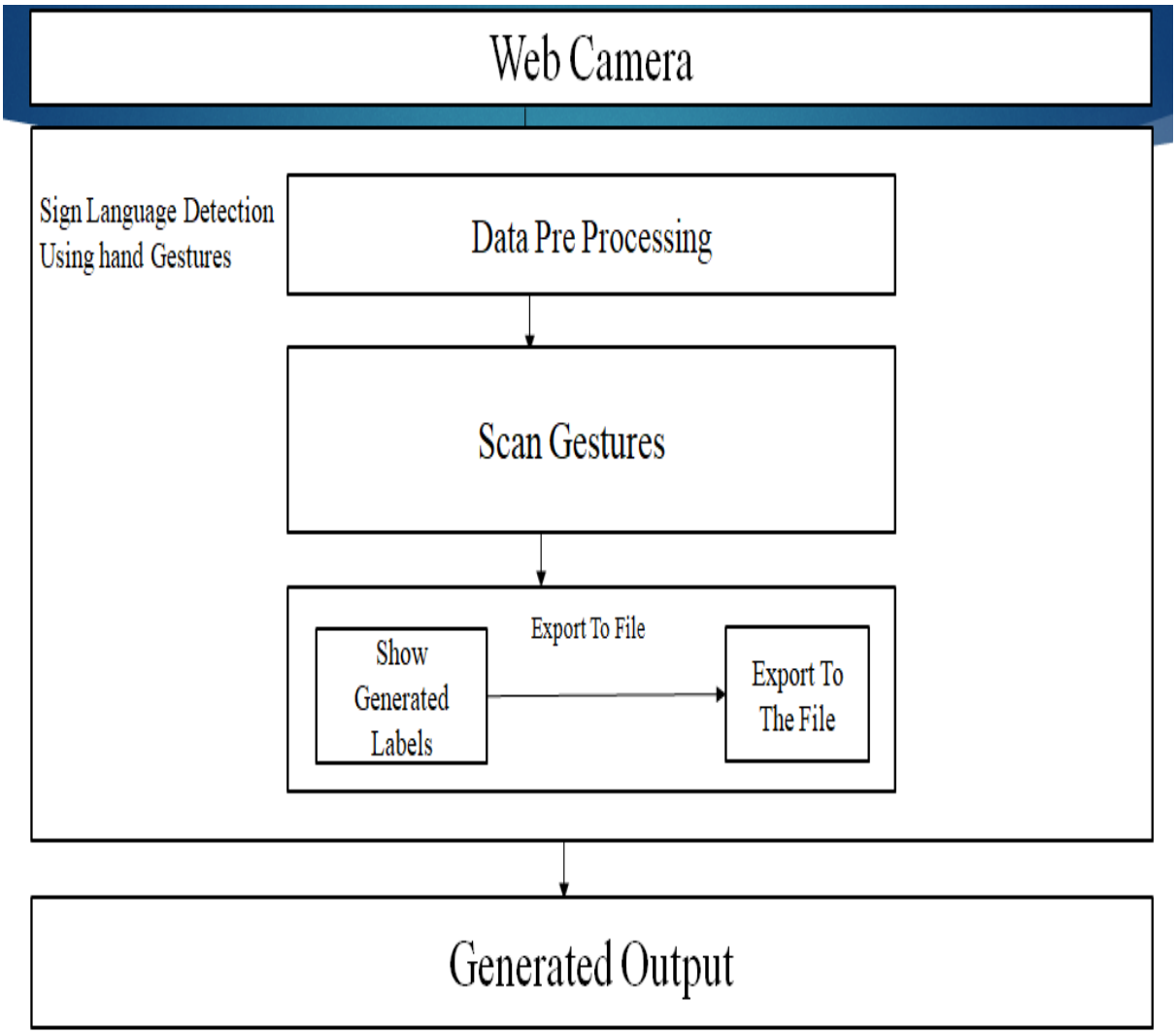


Fig 4.1 :Architectural Diagram

4.2.USECASE DIAGRAM:

In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role (known in the UML as an actor) and a system to achieve a goal. The actor can be a human or other external system.

Actor: EndUser

Usecases:providegestures,geneate new gestures,capture image,adjust lightning,associate ASL Label with gesture,export contents to a file

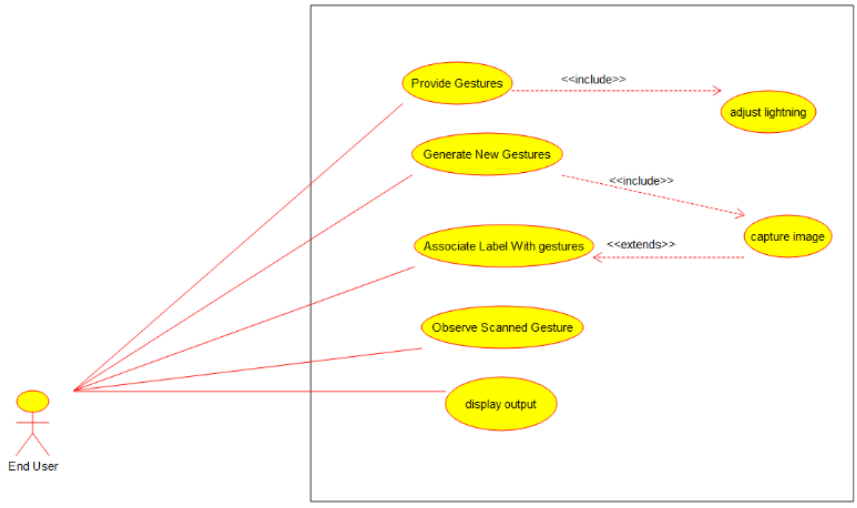


Fig 4.2 :Usecase Diagram

4.3.ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc.,

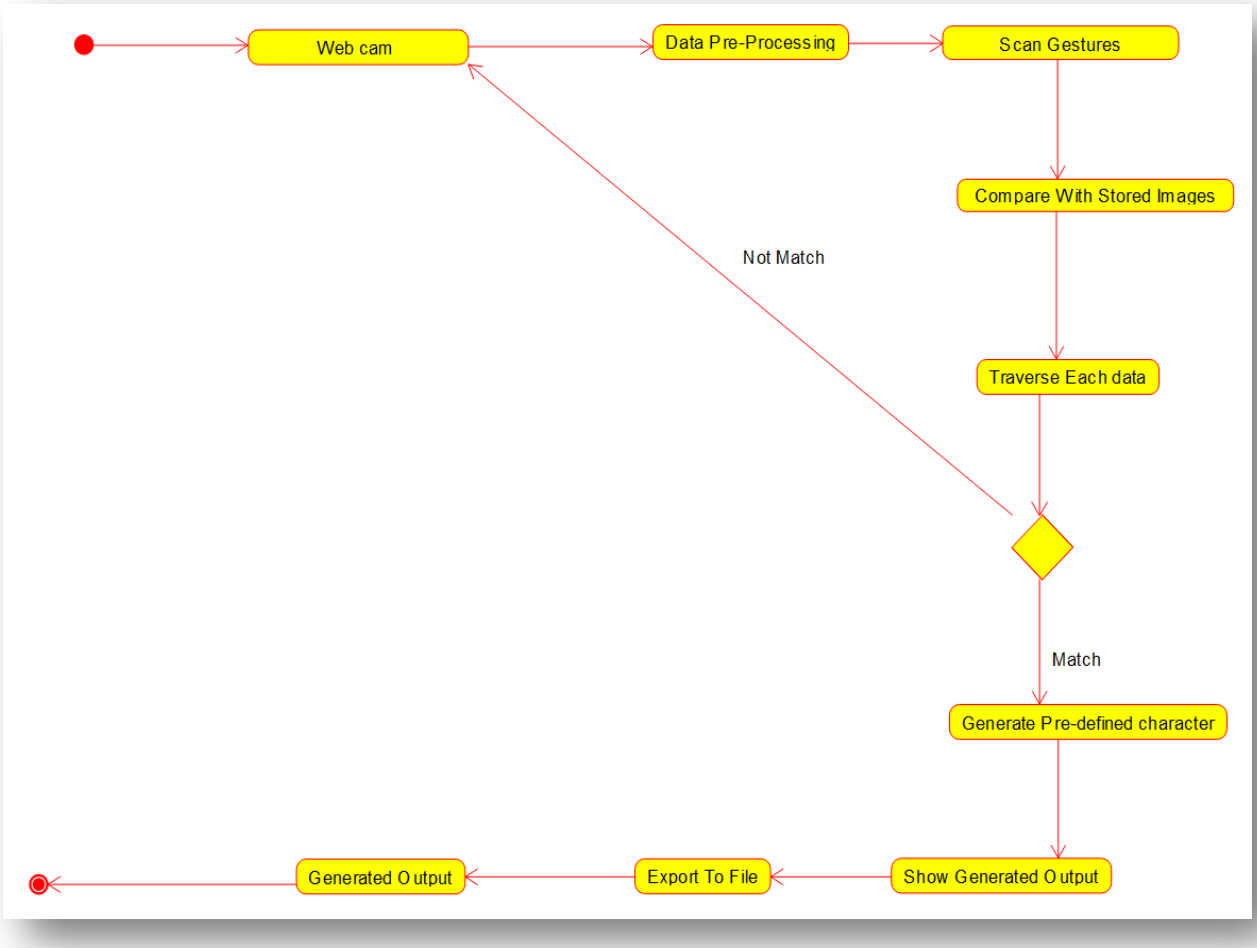


Fig 4.3. :Activity Diagram

4.4.SEQUENCE DIAGRAM:

Sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

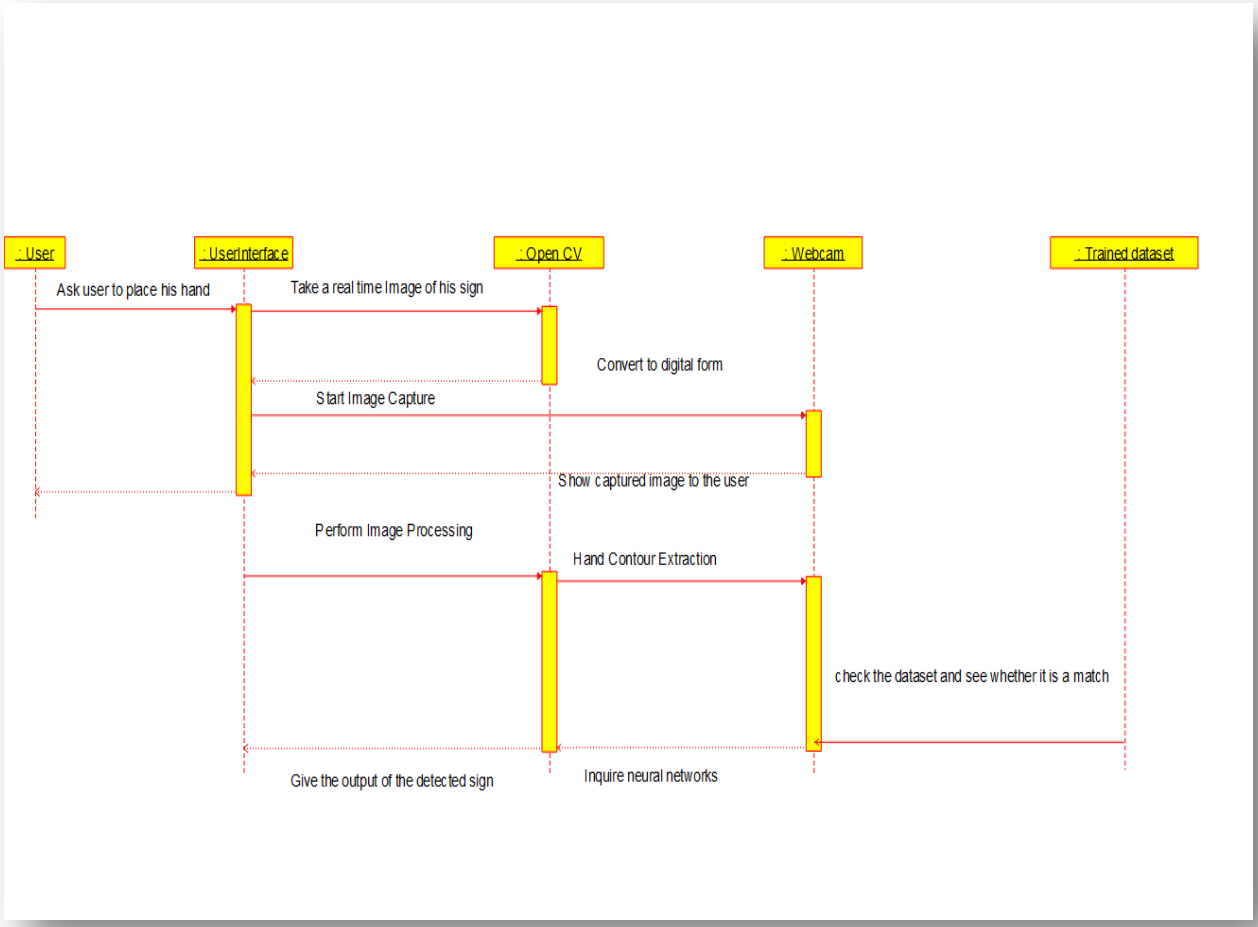


Fig 4.4. :Sequence Diagram

5.IMPLEMENTATION

5.1.Environmental setup:

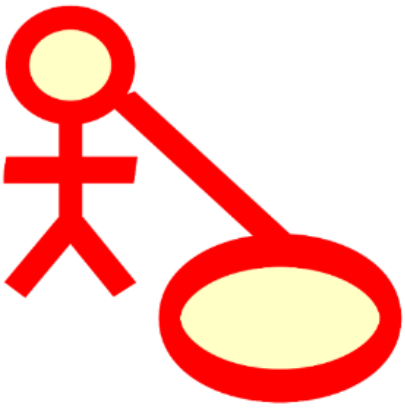
Tools:

The following tools are used mainly to develop the ML application:

1.Jupyter

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code,equations,visualizations and narrative text.Uses include:data cleaning and transformation,numerical simulation,statistical modeling,data visualization,machine learning and much more.

2.Umbrello



Umbrello is a Unified Modelling Language(UML) modeling tool and code generator.It can create industry-standard UML format,and can also generate code from UML diagrams in a variety of programming languages.

Features:

- Supported formats:XMI
- Several type of diagrams supported:use case,class,sequence,communication,state,activity,component,deployment,entity relationship.

Technologies:

1.Python

Python is an interpreted,high-level,general-purpose programming language.Python's design philosophy emphasizes code redability.Its language constructs and object-oriented approach aim to help programmers write clear,logical code for small and large scale projects.Python is dynamically typed and garbage collected.It supports multiple programming paradigms,includingstructured(particularly,procedural),object-oriented and functional programming.

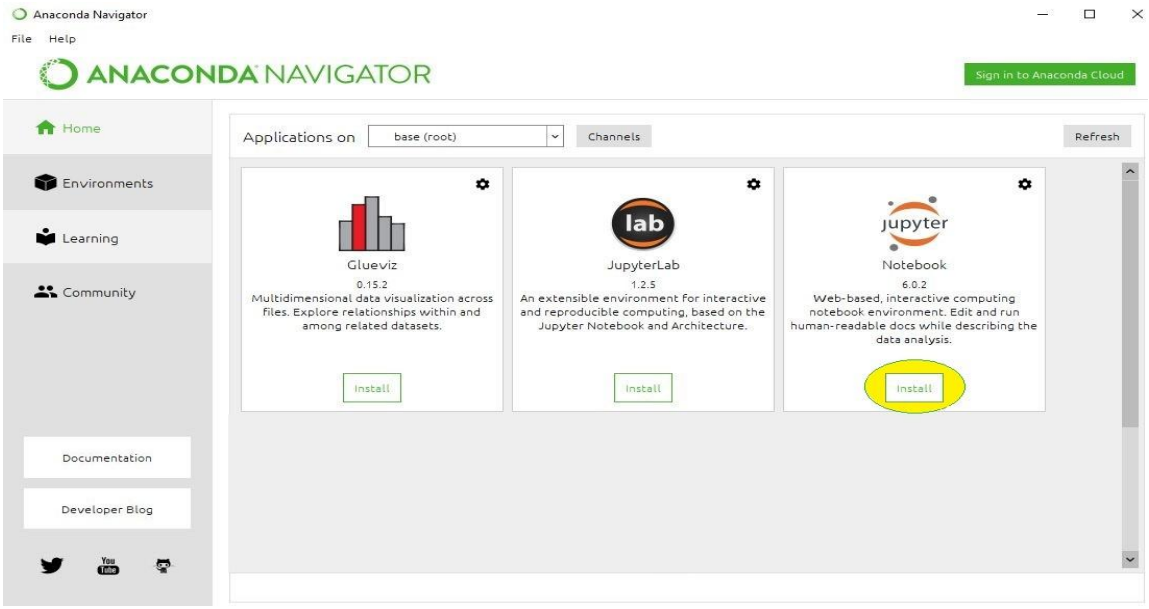
Software Installation Procedure:

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

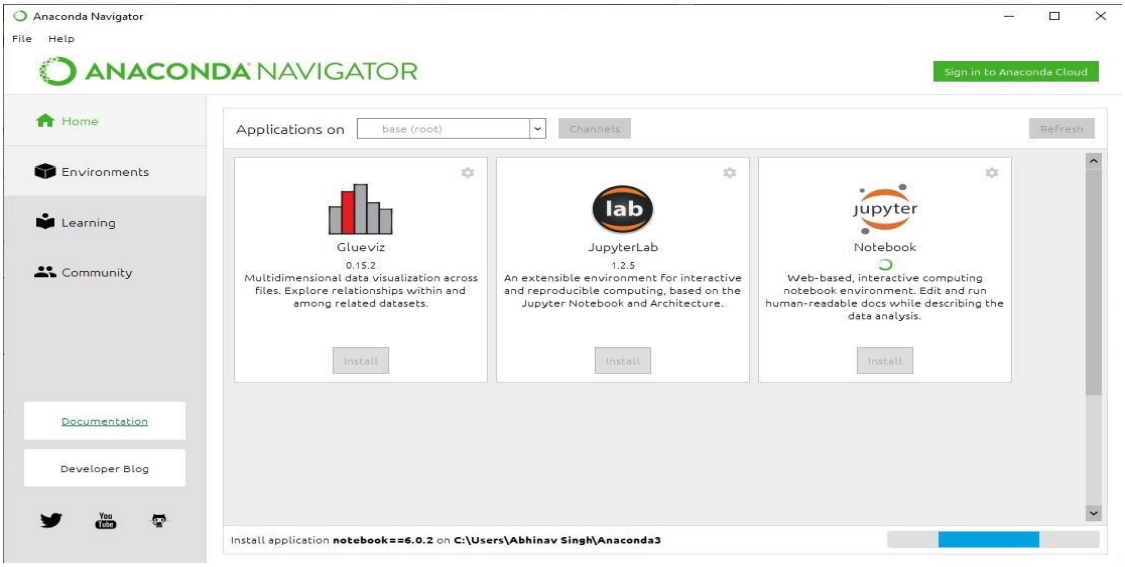
1.Installing Jupyter Notebook using Anaconda:

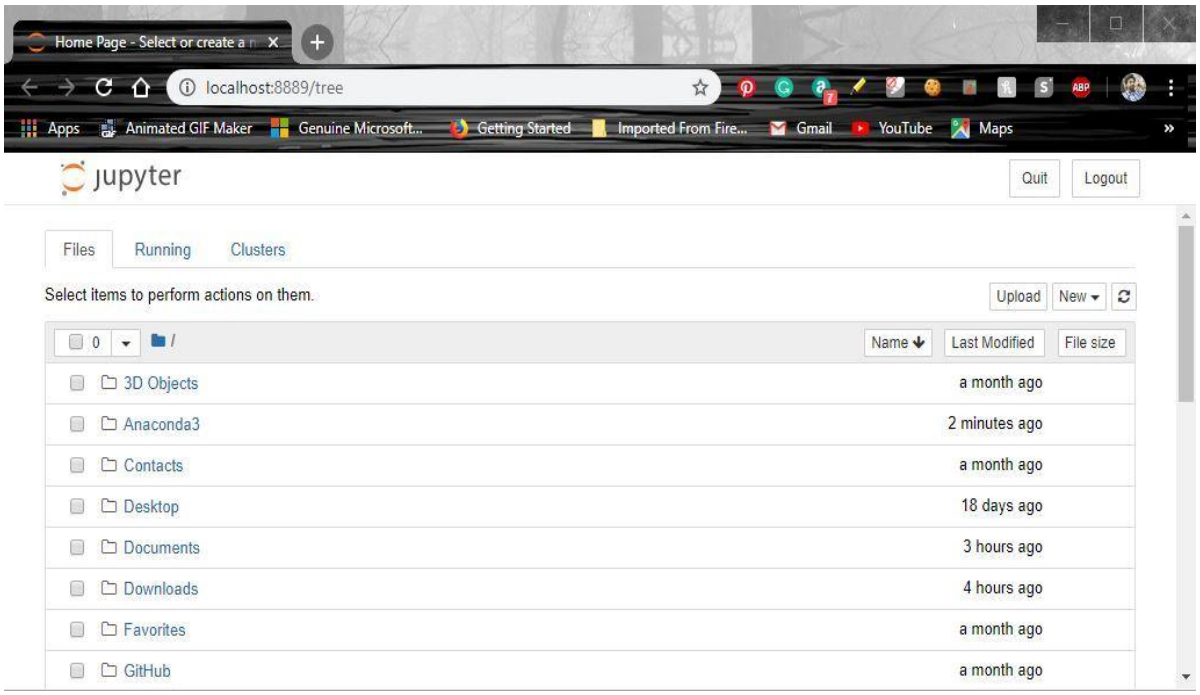
1)Launch Anaconda Navigator:

2)**Click on the Install Jupyter Notebook Button:**



3)Begining the Installation and launching jupyter:





2.Python Installation Procedure:

Step 1: Download Python Executable Installer.
Step 2:Run Executable Installer
Step 3: Verify Python Was Installed On Windows.
Step 4: Verify Pip Was Installed
Step 5: Add Python Path to Environment Variables (Optional)

3.NumPy (version 1.16.5)

Installation Command: pip install numpy

4.cv2 (OpenCV) (version 3.4.2)

Installation Command: pip install opencv-python

5.TensorFlow

Installation Command: pip install tensorflow

5.2.Module Description

User Modules:

The implementation is divided into 3 Modules:

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

All of which are created as three separate .py files. The file structure is given below:

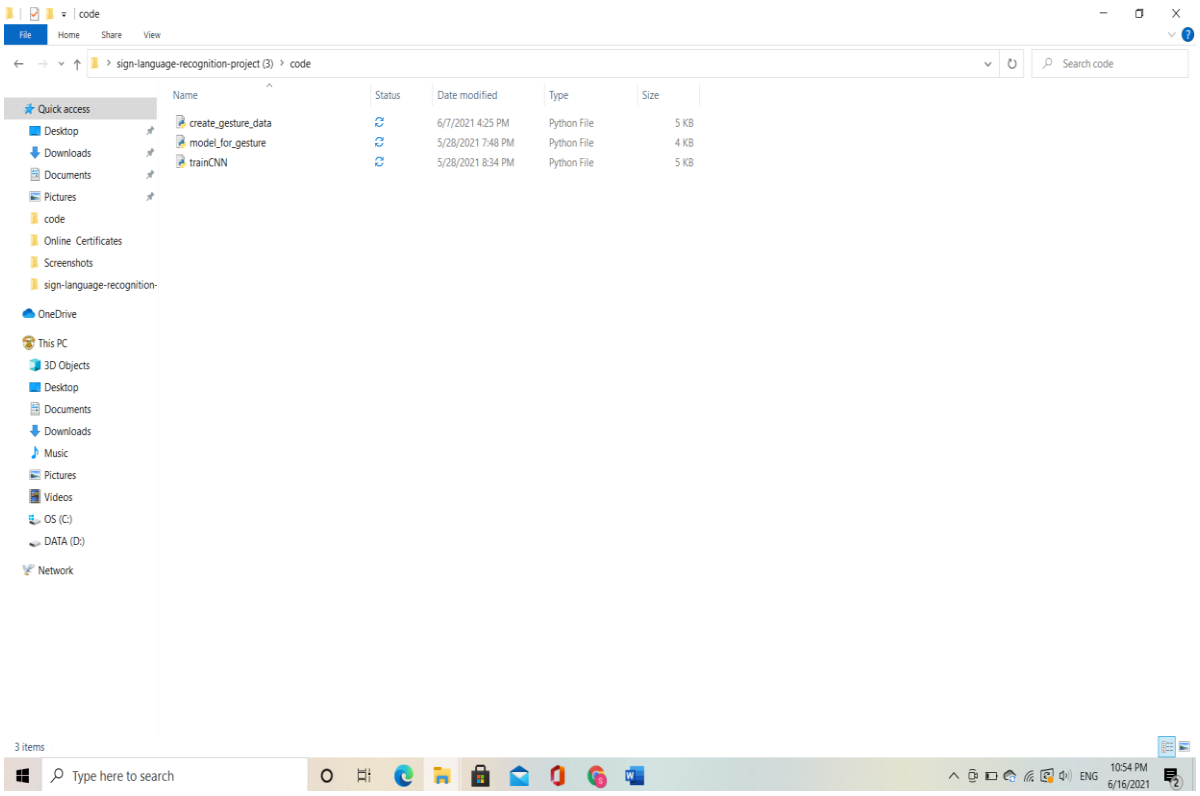


Fig 5.2.1 :File Structure for .py file

5.2.1. Creating the dataset for sign language detection:

Dataset is created with the help of create_gestures_data.py

create_gestures_data.py takes live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory (here gesture directory) that contains two folders train and test, each containing 10 folders containing images captured.

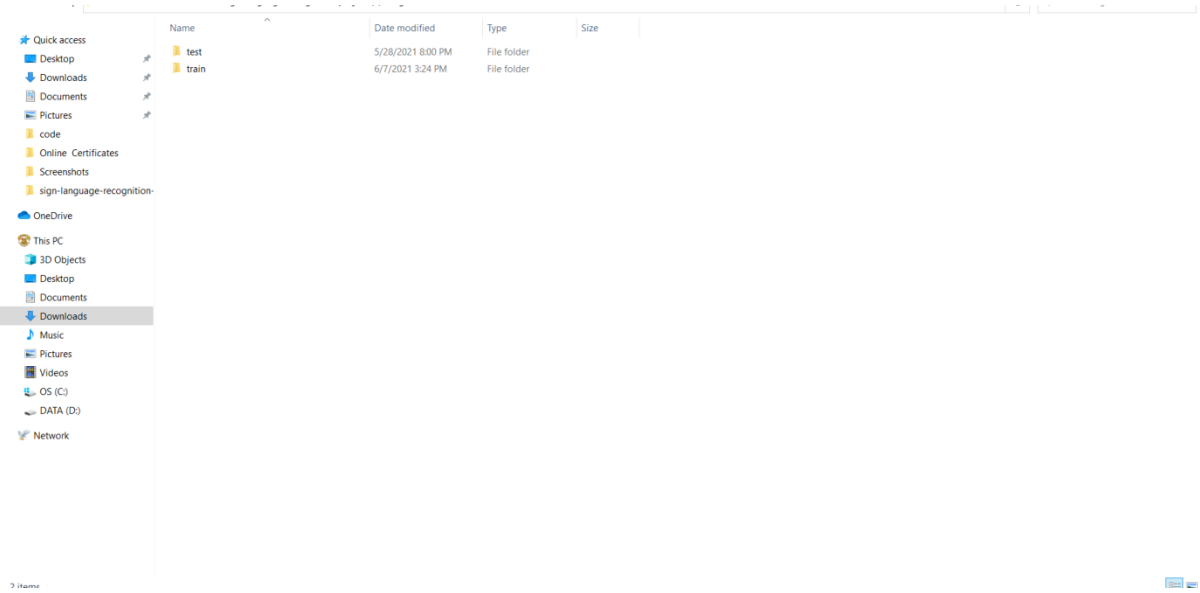


Fig 5.2.1.1:Dataset Directory Structure

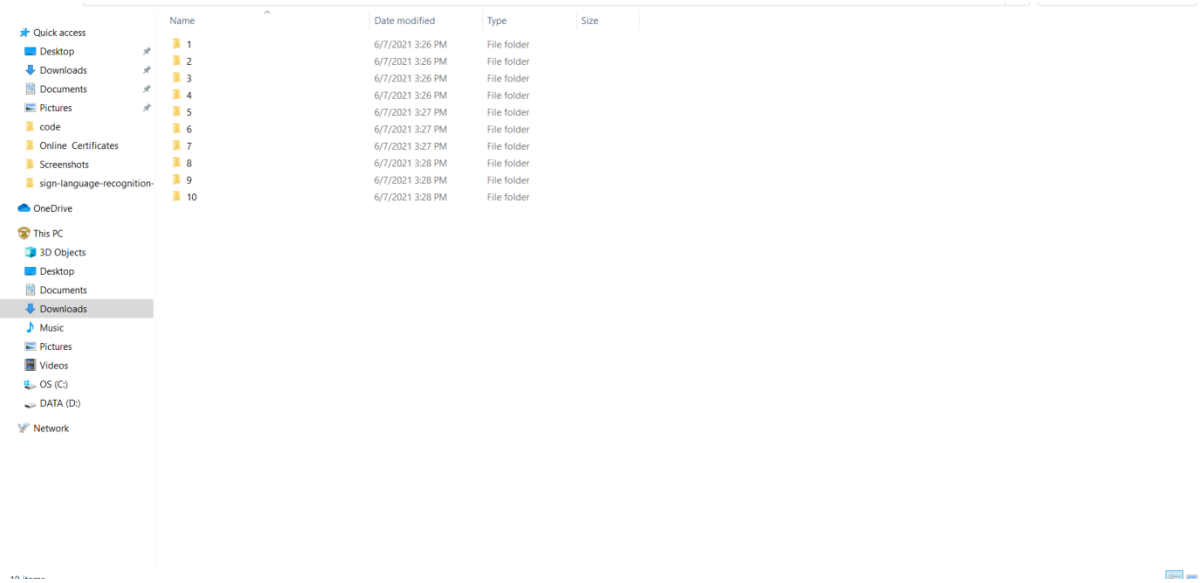


Fig 5.2.1.2:Inside Test(Train folder is similar to this)

For creating the dataset we get the live cam feed using OpenCV and create an ROI that is nothing but the part of the frame where we want to detect the hand in for the gestures.

The red box is the ROI and this window is for getting the live cam feed from the webcam.

Fig 5.2.1.3: Region of Interest

For differentiating between the background ,accumulated weighted avg is calculated for the background and then subtract it from the frames that contain some object in front of the background that can be distinguished as foreground.

This is obtained by calculating the accumulated_weight for some frames (here for 60 frames) as the accumulated_avg for the background is calculated.

After calculating the accumulated avg for the background, It is subtract from every frame that is read after 60 frames to find any object that covers the background.

Code for creating region of interest and fetching background:

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D,
Dropout
from keras.optimizers import Adam, SGD
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
import warnings
import numpy as np
import cv2
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import ModelCheckpoint, EarlyStopping
warnings.simplefilter(action='ignore', category=FutureWarning)
background = None
accumulated_weight = 0.5
#Creating the dimensions for the ROI...
ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350
def cal_accum_avg(frame, accumulated_weight):
global background
if background is None:
background = frame.copy().astype("float")
return None
cv2.accumulateWeighted(frame, background, accumulated_weight)
```

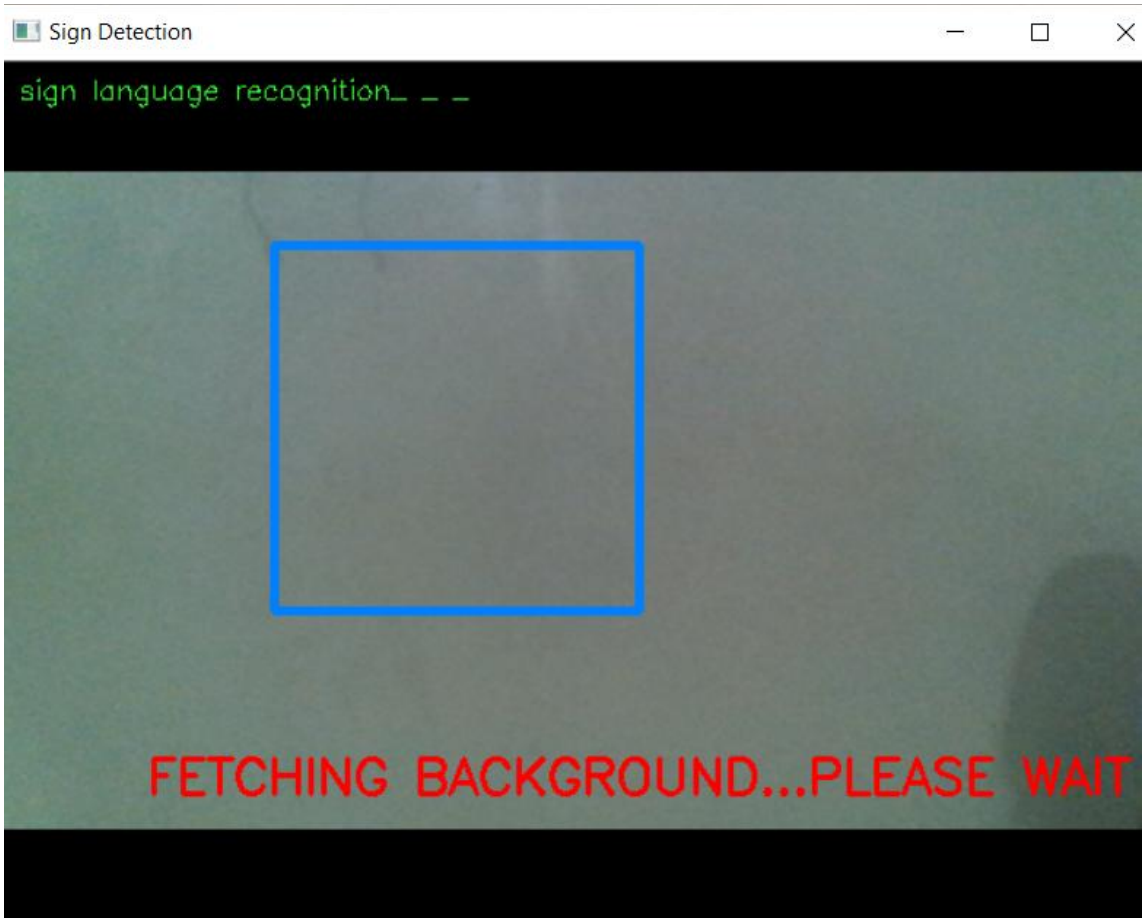


Fig 5.2.1.4:Fetching background before the gestures are detected

Calculate threshold value

Now the threshold value for every frame is calculated and determine the contours using cv2.findContours and return the max contours (the most outermost contours for the object) using the function segment. Contours are used to determine if there is ay foreground object being detected in in the ROI(To check if there is a hand in the ROI)

Function for calculating the threshold value:

```
def segment_hand(frame, threshold=25):
global background
diff = cv2.absdiff(background.astype("uint8"), frame)
_ , thresholded = cv2.threshold(diff, threshold,255,cv2.THRESH_BINARY)
# Grab the external contours for the image
image, contours, hierarchy = cv2.findContours(thresholded.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if len(contours) == 0:
return None
else:
hand_segment_max_cont = max(contours, key=cv2.contourArea)
return (thresholded, hand_segment_max_cont)
```

When contours are detected (or hand is present in the ROI), It start's to save the image of the ROI in the train and test set respectively for the letter or number that it is detecting for.

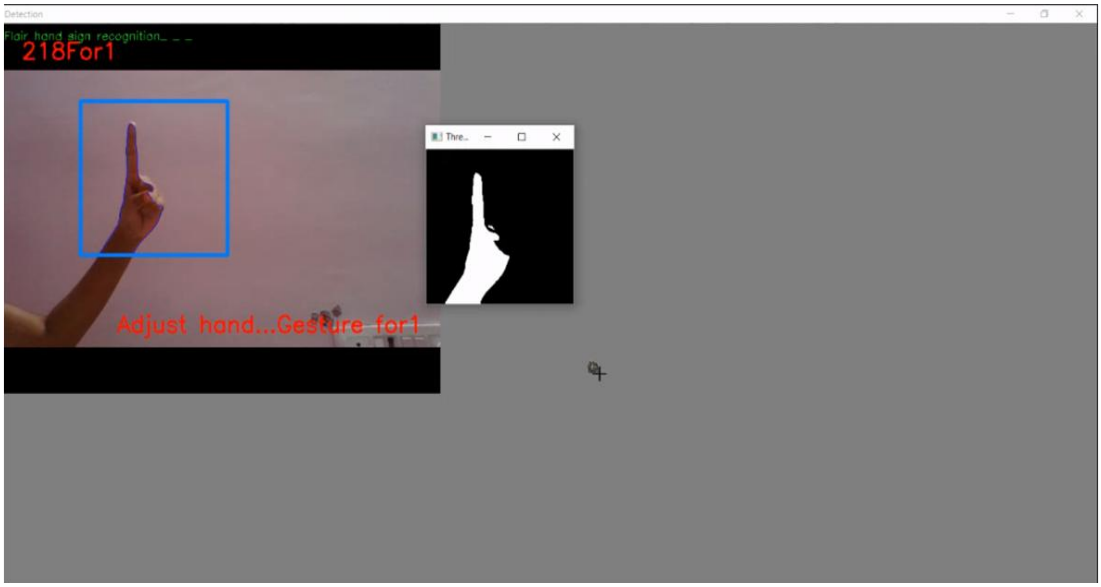


Fig 5.2.1.1.1:Create_gestures_data.py capturing the data

In the above example, the dataset for 1 is being created and the thresholded image of the ROI is being shown in the next window and this frame of ROI is being saved in ..train/1

Fig 5.2.1.1.2 :Saved images for number one

For the train dataset, 701 images for each number to be detected are saved, and for the test dataset, 40 images are saved for each number.

5.2.2. Training CNN

In this step training of model is done.

First, the data is loaded using ImageDataGenerator of keras through which the flow_from_directory function is used to load the train and test set data, and each of the names of the number folders will be the class names for the images loaded.

Code for loading train and test data:

```
train_path = r'C:\Sign Language\gesture\train'
test_path = r'C:\Sign Language\gesture\test'
train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from
_directory(directory=train_path, target_size=(64,64), class_mode='categorical',
batch_size=10,shuffle=True)
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from
_directory(directory=test_path, target_size=(64,64), class_mode='categorical', batch_size=10,
shuffle=True)
```

Code for plotImages function:

```
imgs, labels = next(train_batches)
#Plotting the images...
def plotImages(images_arr):
fig, axes = plt.subplots(1, 10, figsize=(30,20))
axes = axes.flatten()
for img, ax in zip( images_arr, axes):
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
ax.imshow(img)
ax.axis('off')
plt.tight_layout()
plt.show()
plotImages(imgs)
print(imgs.shape)
print(labels)
```

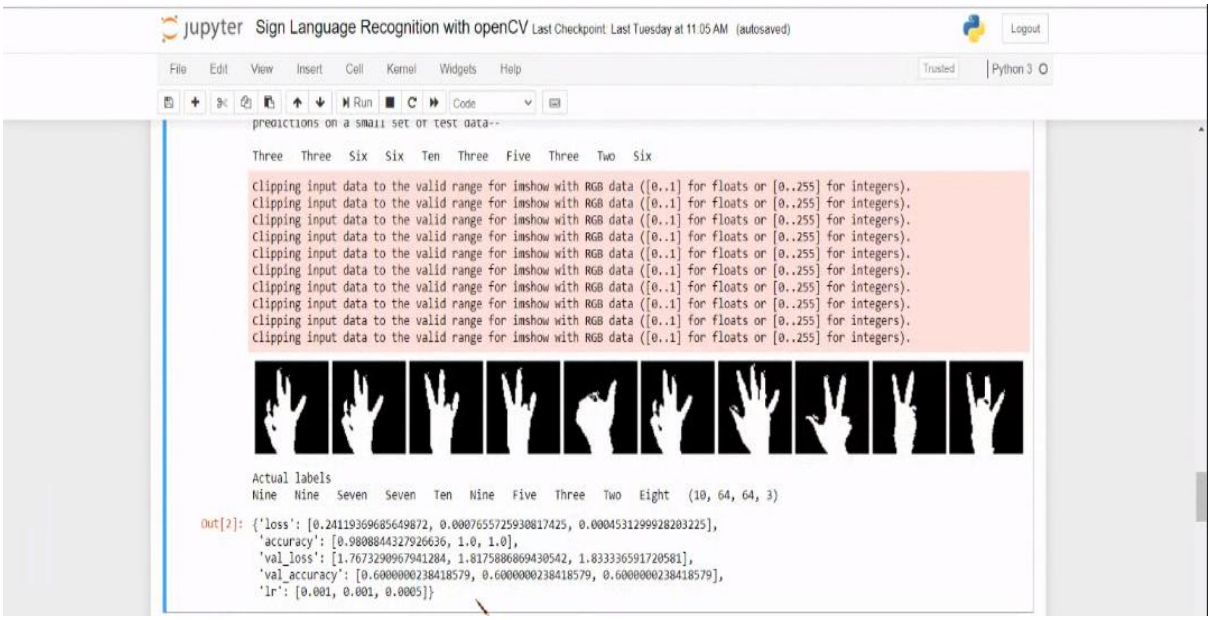


Fig 5.2.2 : Plotting images with the dataset

Code for Designing CNN:

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Flatten())
model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.2))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.3))
model.add(Dense(10,activation ="softmax"))
```

The model is fitted and to be used in the last module (model_for_gesture.py)

In training callbacks of Reduce LR on plateau and earlystopping is used, and both of them are dependent on the validation dataset loss.

After every epoch, the accuracy and loss are calculated using the validation dataset and if the validation loss is not decreasing, the LR of the model is reduced using the Reduce LR to prevent the model from overshooting the minima of loss and also we are using the earlystopping algorithm so that if the validation accuracy keeps on decreasing for some epochs then the training is stopped.

The optimization algorithms used here is SGD (stochastic gradient descent, that means the weights are updated at every training instance

The training is found 100% training accuracy and validation accuracy of about 81%

After compiling the model the model is fitted on the train batches for 10 epochs (may vary according to the choice of parameters of the user)

For evaluating the model on the test set and printing the accuracy and loss scores,the next batch of images from the test data are taken.

At the end, visualizing and making a small test on model is done to check if everything is working while detecting on the live cam feed.

5.1.3. Predict the gesture

In this, a bounding box is created for detecting the ROI and calculate the accumulated_avg as like in the creating the dataset. This is done for identifying any foreground object.

Now the max contour is found and if contour is detected that means a hand is detected so the threshold of the ROI is treated as a test image.

The previously saved model is loaded using keras.models.load_model and feed the threshold image of the ROI consisting of the hand as an input to the model for prediction.

Code for importing modules for model_for_gesture.py:

```
import numpy as np
import cv2
import keras
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
```

Now the model is loaded ,set some of the variables that is needed, i.e, initializing the background variable, and setting the dimensions of the ROI.

Code for loading model and setting ROI:

```
model = keras.models.load_model(r"C:\Users\ best_model.h5")
background = None
accumulated_weight = 0.5
ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350
```

Function to calculate the background accumulated weighted average:

```
def cal_accum_avg(frame, accumulated_weight):
global background
if background is None:
background = frame.copy().astype("float")
return None
cv2.accumulateWeighted(frame, background, accumulated_weight)
```

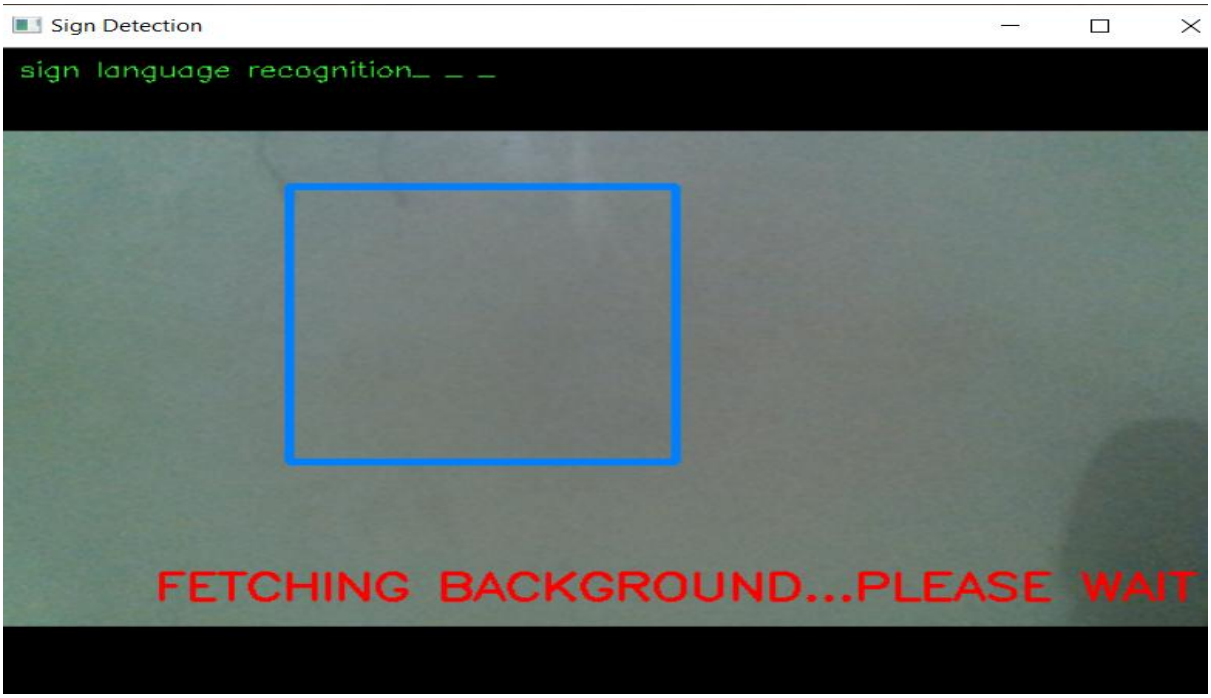
Segmenting the hand, i.e, getting the max contours and the thresholded image of the hand detected.The hand is detected on live cam feed.

5.3.Software description:

This following steps describe the procedure to use the software

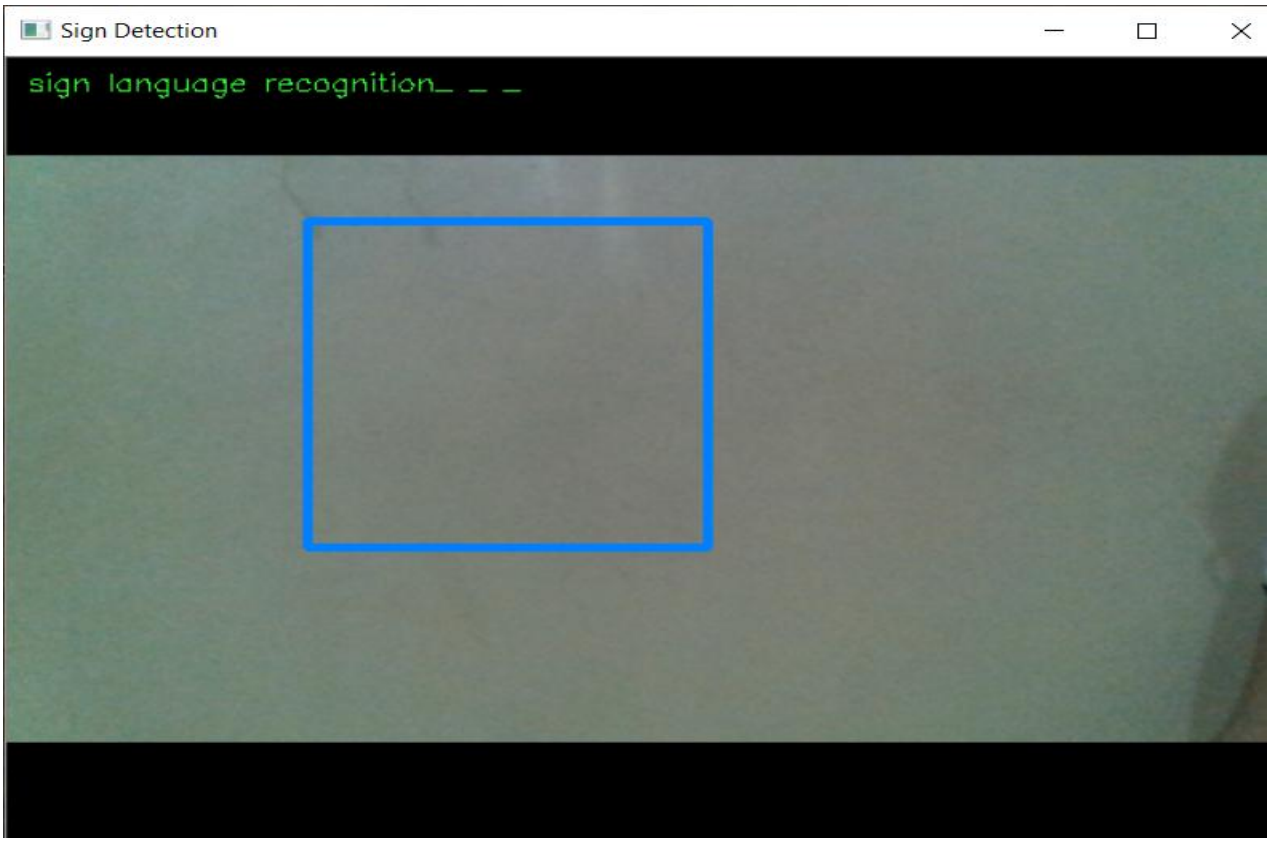
STEP-1:

When the application starts running it takes few seconds to adjust the background (the background should be plain for easy detection).



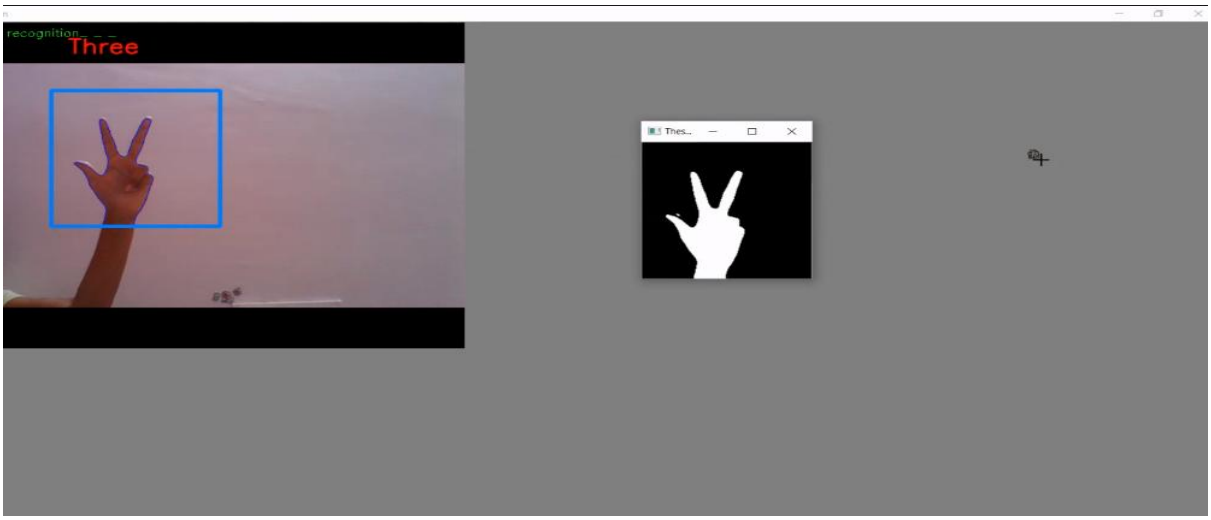
STEP-2:

After the background is adjusted .It asks to show the gesture in the region of interest,which is a box shown.



STEP-3:

The detected sign will be shown on the screen as shown



STEP-4:

To stop the application user need to press ESC key.

6.SYSTEM TESTING

6.1.Tests:

Test Cases	Test Case Objective	Input Data	Expected Output	Actual Output	Test Result
TC-1	Recognising the number one	Index Finger is shown to the webcam	One	One	Pass
TC-2	Recognising the number ten	Thumb open with other fingers closed	Ten	Ten	Pass
TC-3	Recognising the number four	Thumb closed and other finger open	Four	Six	Fail
TC-4	Recognising the number two	Index finger and middle finger shown	Two	Two	Pass

TC-5	Recognising the number Nine	Thumb and index finger closed as circle	Nine	Nine	Pass
TC-6	Recognising the number Three	Thumb,Index and middle finger open	Three	Three	Pass

Table-1:Testcases

The project is tested upon various test cases whose details and corresponding results are described here under:

Testcase-1) Index finger is shown to webcam

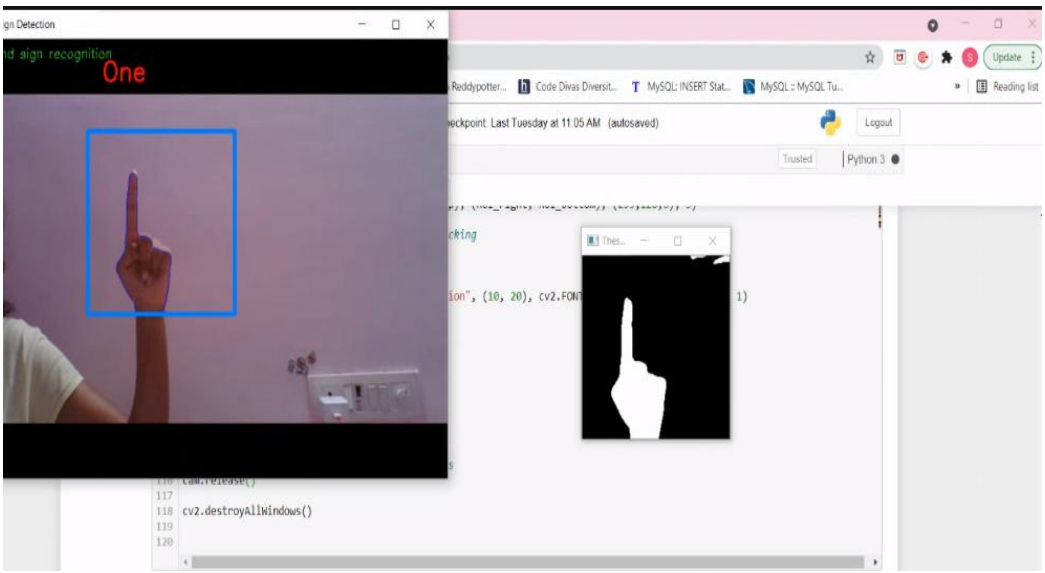


Fig 6.1:IndexFinger is shown to webcam

Testcase-2) Thumb with other fingers closed

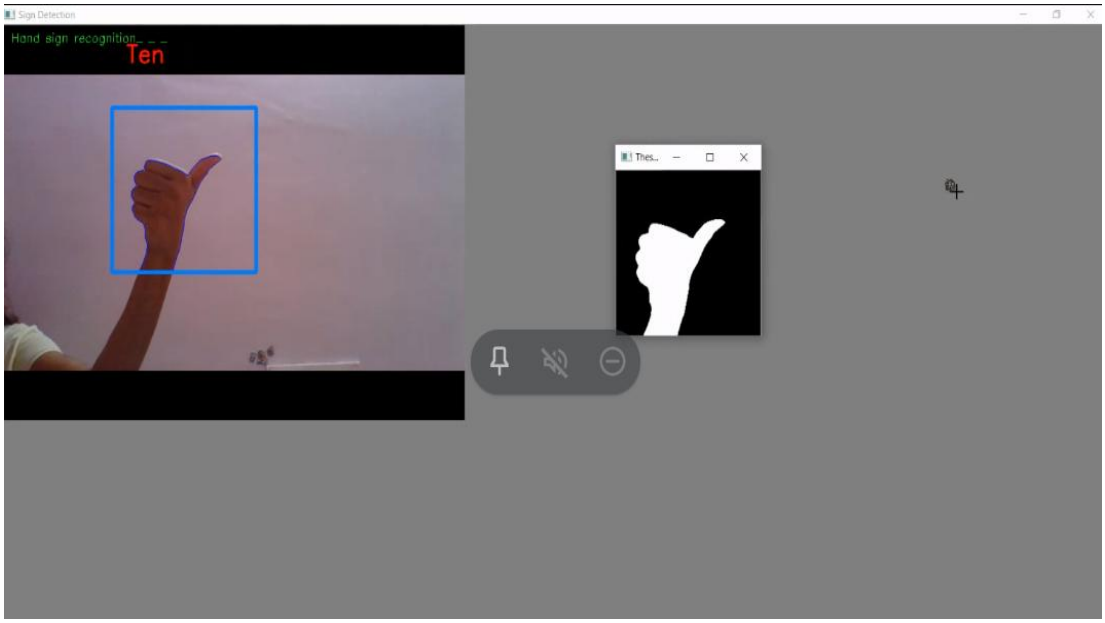


Fig 6.2:ThumbFinger is shown to webcam

Testcase-3)

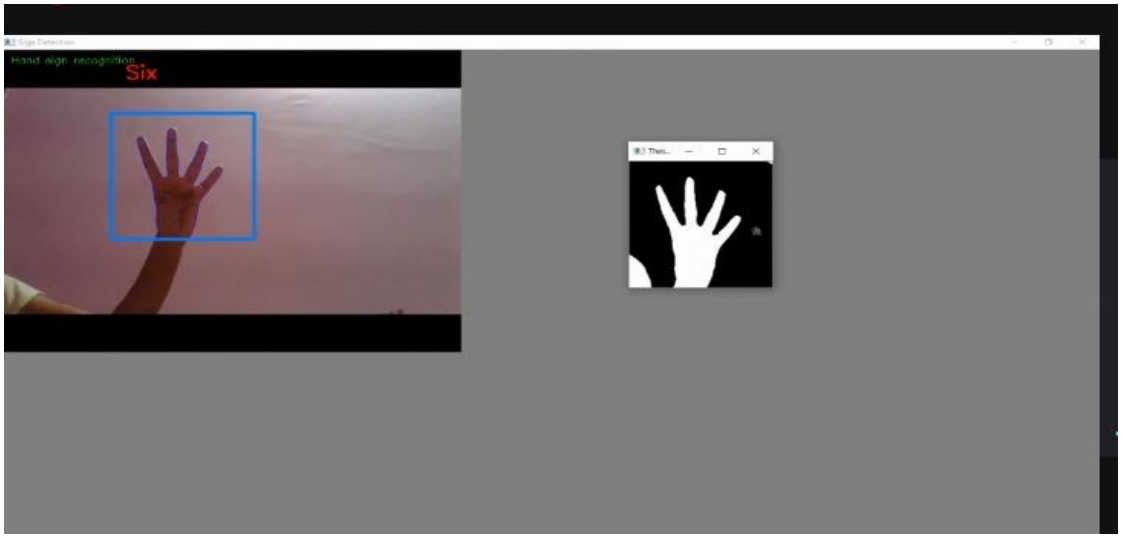


Fig 6.3:Not Recognising the gesture

Testcase-4)

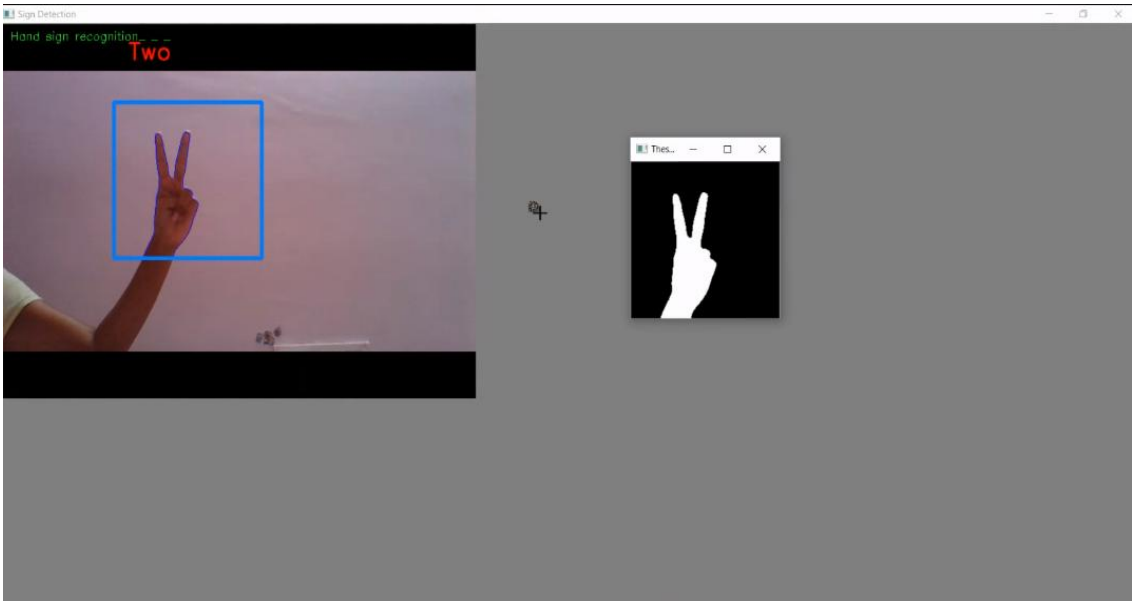


Fig 6.4 :Gesture indicating TWO

Testcase -5)

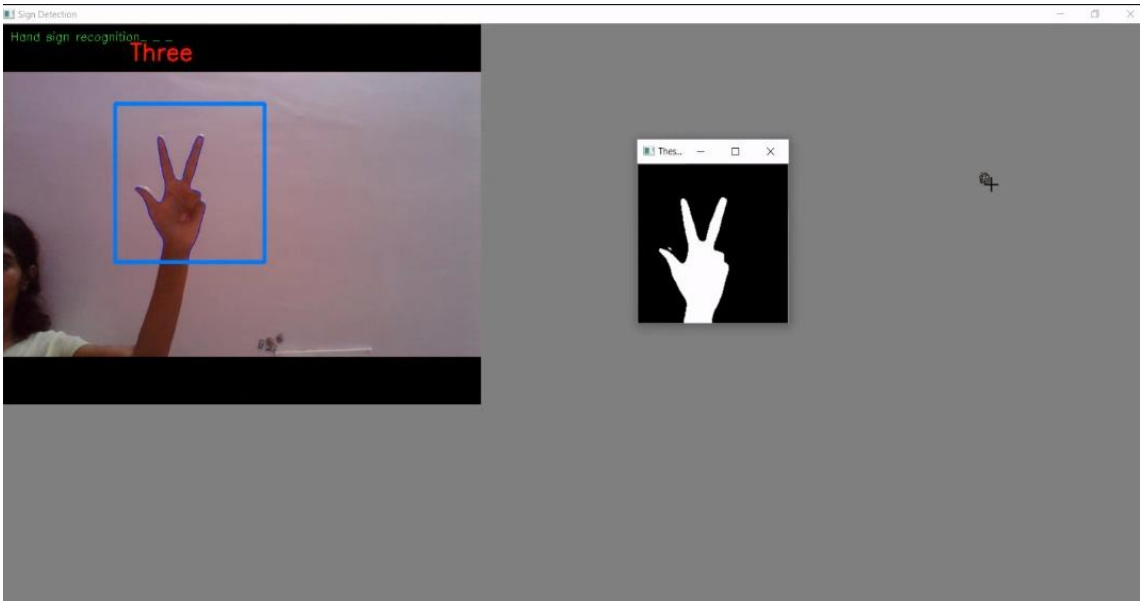


Fig 6.5 :Gesture indicating THREE

Testcase-6)

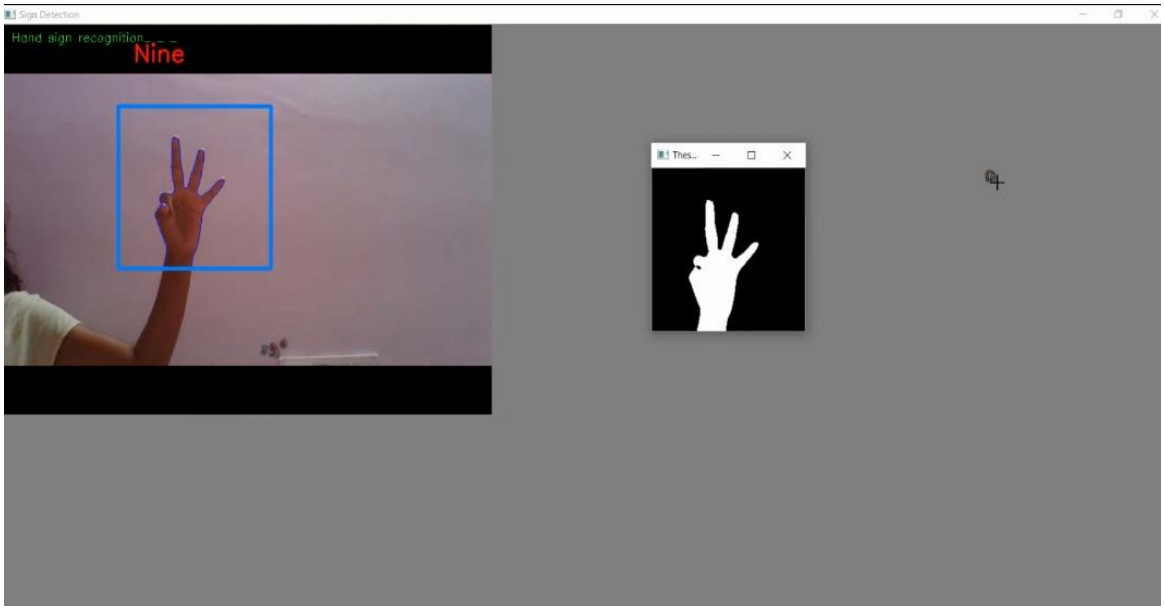


Fig 6.6 :Gesture Indicating NINE

7.RESULT SCREENSHOTS

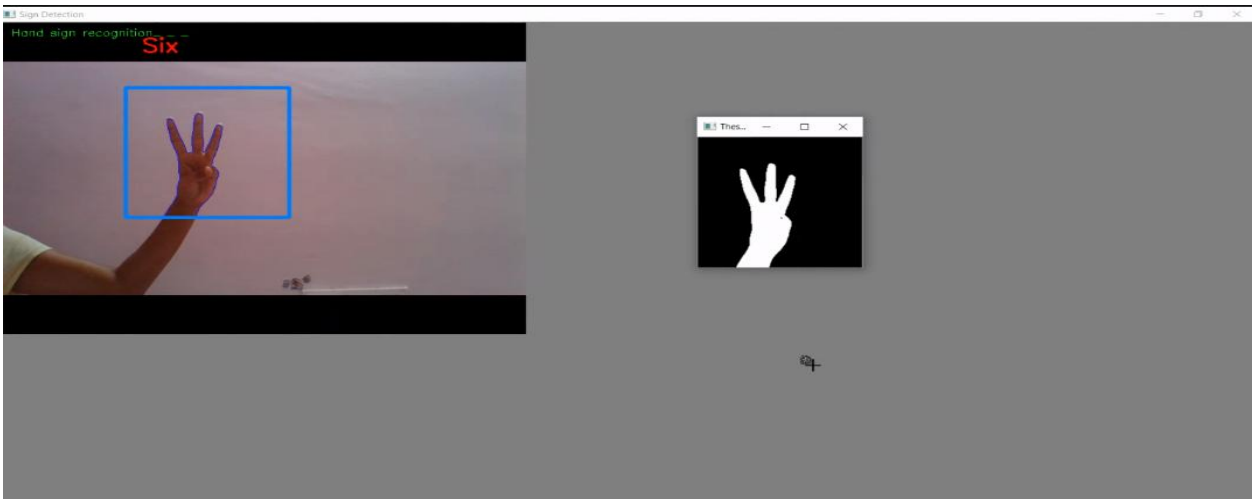


Fig 7.1:Result indicating SIX

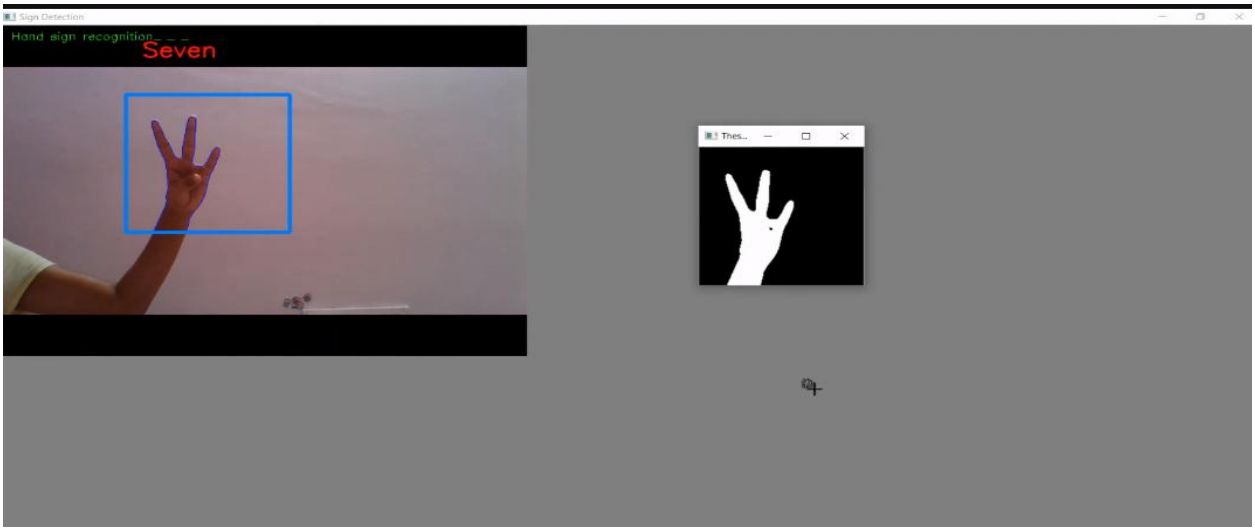


Fig 7.2:Result indicating SEVEN

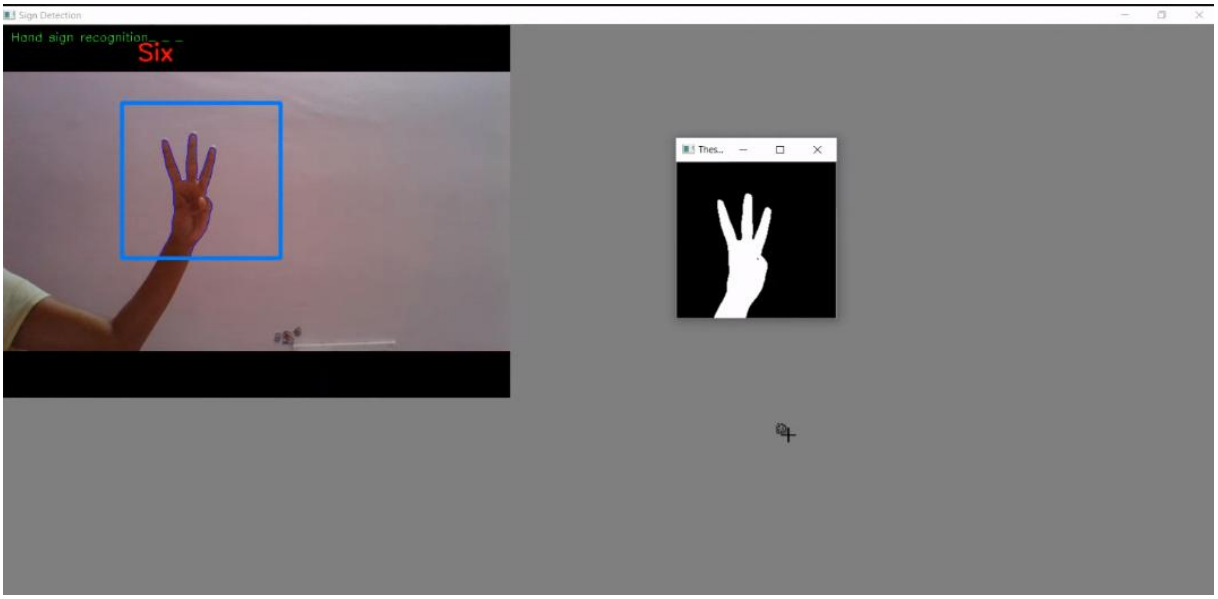


Fig 7.3 :Output for hand gesture number Six

8.CONCLUSION

We have successfully developed sign language recognition project. This is an interesting machine learning python project to gain expertise. The system does not require the background to be perfectly black. It works on any plain background. Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them.

The system didn’t perform too well but it was demonstrated that it can be built a first-person sign language translation system using only cameras and convolutional neural networks.Our project aims to make communication simpler between deaf and dumb people by introducing Computer in communication path so that sign language can be automatically captured, recognized, translated to text and displayed it on LCD.We used CNN to classify the sign gestures , we got an accuracy of 81%.The Project is successfully recognizing hand gestures and giving respective number in text.

9.BIBLIOGRAPHY

[1]T. Yang, Y. Xu, and “A. , Hidden Markov Model for Gesture Recognition”, CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ.,Pittsburgh,PA, May 1994.

[2]Pujan Ziaie, Thomas M¨uller , Mary Ellen Foster , and Alois Knoll“A Na¨ive Bayes Munich,Dept. of Informatics VI, Robotics and Embedded Systems,Boltzmannstr. 3, DE-85748 Garching, Germany

[3]https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_b
ilateral_filter/gausian_median_blur_bilateral_filter.html

[4]Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.

[5]aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/

[6] https://link.springer.com/chapter/10.1007/978-981-15-1420-3_80

[7]https://ieeexplore.ieee.org/document/7916786

[8] https://link.springer.com/chapter/10.1007/978-3-319-16178-5_40