***Project Phase III***

***Report***

***On­­­***

**SIGN LANGUAGE RECOGNITION USING PYTHON**

BACHELOR OF ENGINEERING

**COMPUTER SCIENCE & ENGINEERING**

****

**Submitted to: Submitted By:**

**Er. Parwinder Kaur (Supervisor) Student Group**

**(e12551) 2 Students**

1. **SAHIL KUMAR (20BCS9238)**
2. **SHRUTI SHREYA(20BCS9229)**

**Co-Supervisor Signature**

**Dr. Dheresh Saini**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**CHANDIGARH UNIVERSITY, GHARUAN**

**June 2022**

# **ABSTRACT**

There have been several advancements in technology and a lot of research has been done to help the people who are deaf and dumb. Aiding the cause, Deep learning, and computer vision can be used too to make an impact on this cause.

This can be very helpful for the deaf and dumb people in communicating with others as knowing sign language is not something that is common to all, moreover, this can be extended to creating automatic editors, where the person can easily write by just their hand gestures.

In this sign language recognition project, we create a sign detector, which detects numbers from 1 to 10 that can very easily be extended to cover a vast multitude of other signs and hand gestures including the alphabets.

We have developed this project using OpenCV and Keras modules of python.

**TABLE OF CONTENT:**

|  |  |  |
| --- | --- | --- |
| Sr no. | Topic | Page No. |
| 1 | Feature/characteristics identification | 4-12 |
| 2 | Constraints Identification | 13 |
| 3 | Analysis of features and finalization subject to constraints | 14 |
| 4 | Design selection | 15-20 |

# **FEATURES IDENTIFICATION:**

This is divided into 3 parts:

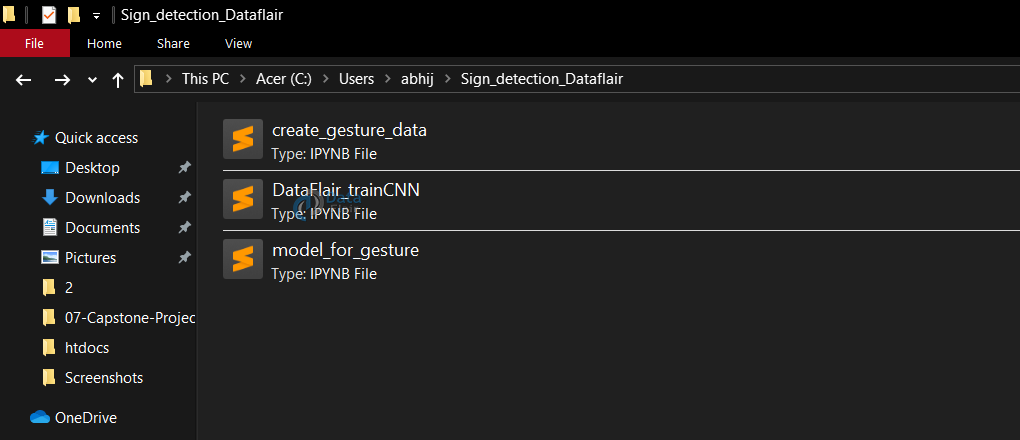
Creating the dataset

Training a CNN on the captured dataset

Predicting the data

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

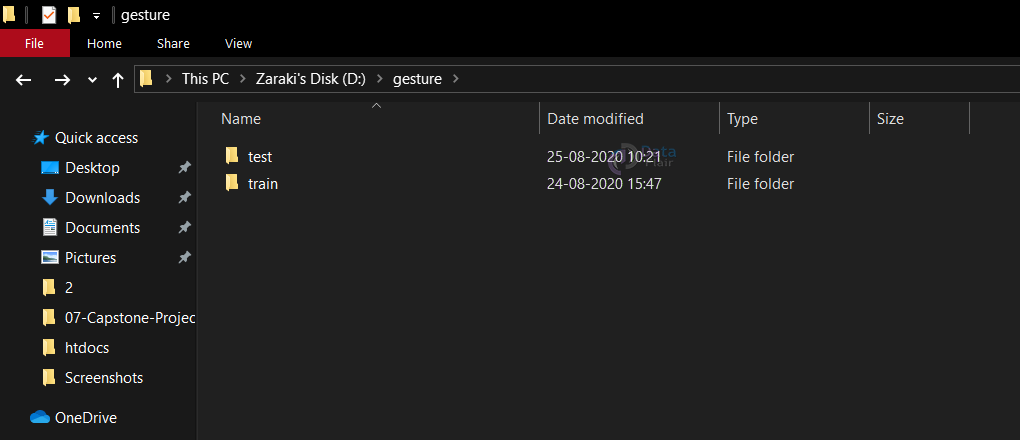
## CREATING THE DATASET FOR SIGN LANGUAGE:



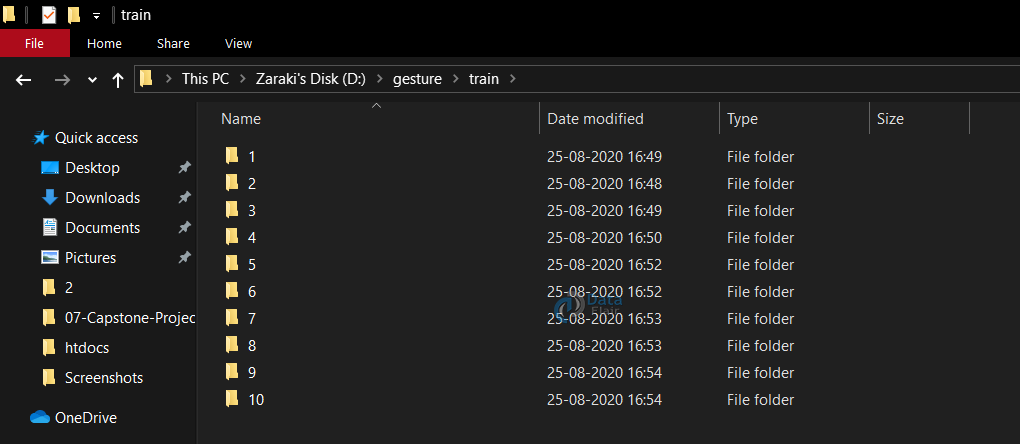
It is fairly possible to get the dataset we need on the internet but in this project, we will be creating the dataset on our own.

We will be having a 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 using the create\_gesture\_data.py

#### DIRECTORY STRUCTURE:

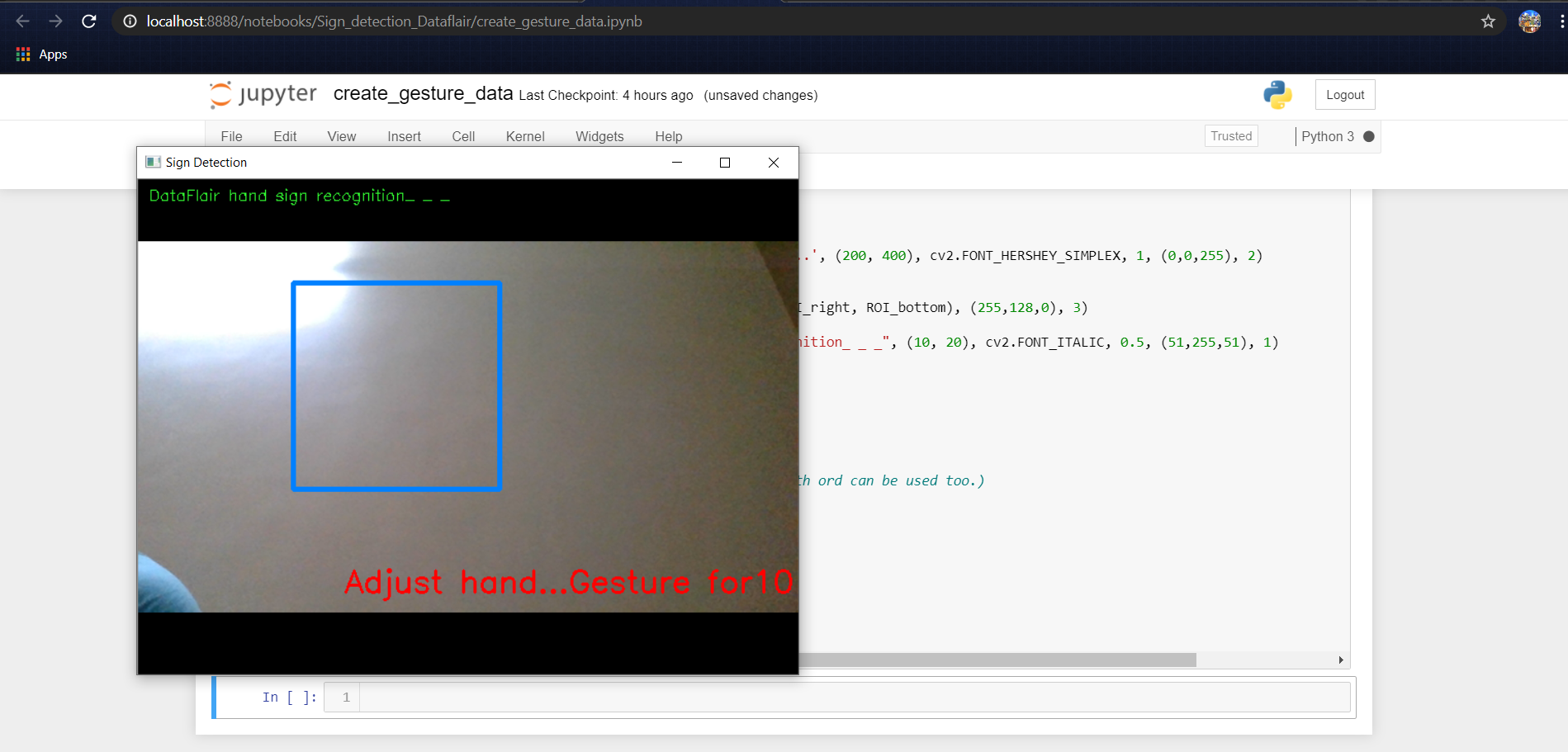


**Inside of train (test has the same structure inside)**



Now 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.

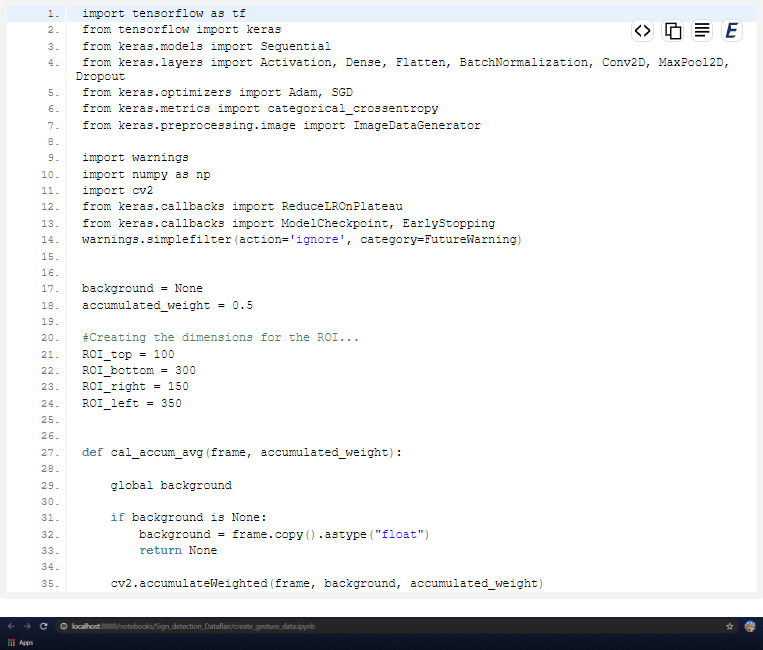
The red box is the ROI and this window is for getting the live cam feed from the webcam.

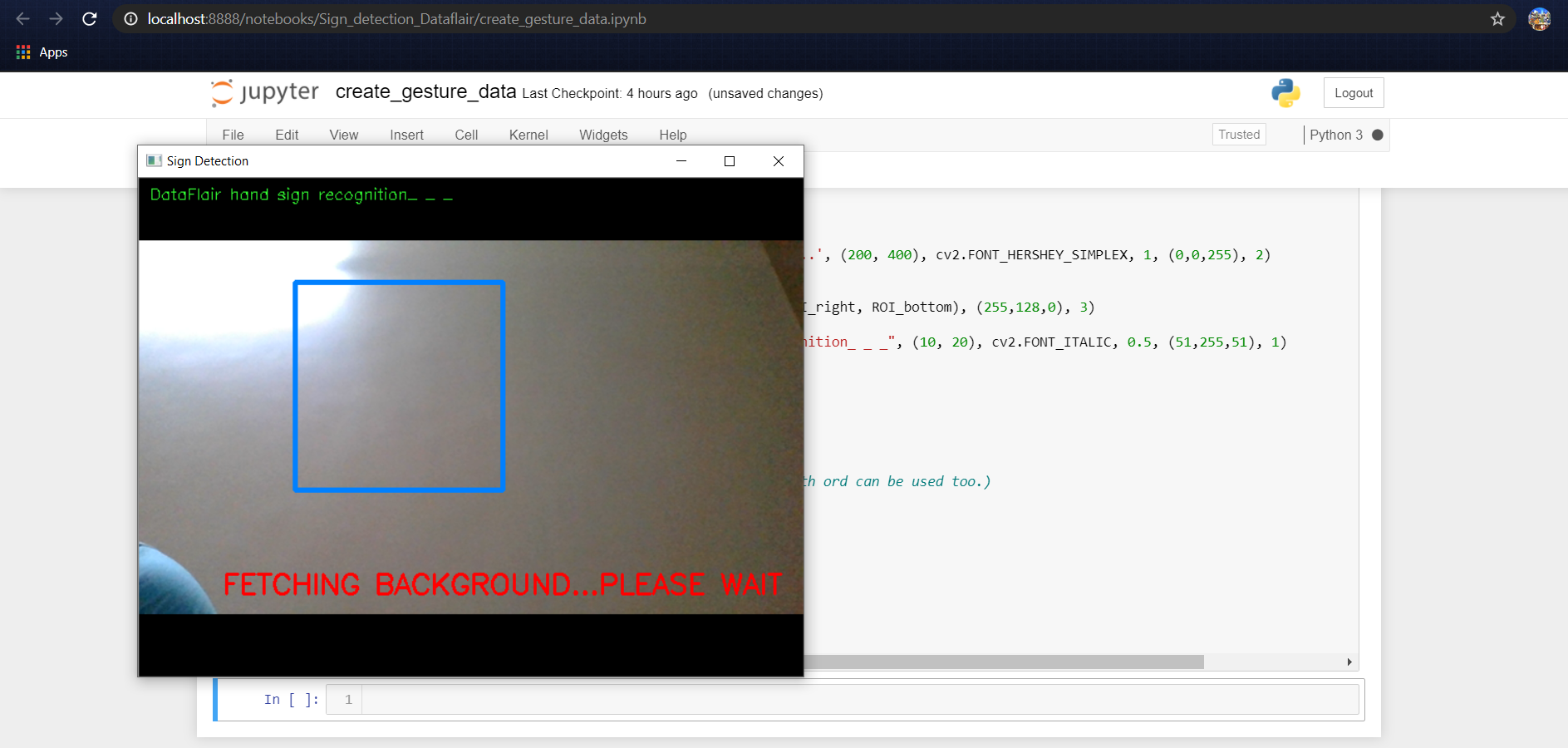


For differentiating between the background we calculate the accumulated weighted avg for the background and then subtract this from the frames that contain some object in front of the background that can be distinguished as foreground.

This is done by calculating the accumulated\_weight for some frames (here for 60 frames) we calculate the accumulated\_avg for the background.

After we have the accumulated avg for the background, we subtract it from every frame that we read after 60 frames to find any object that covers the background.

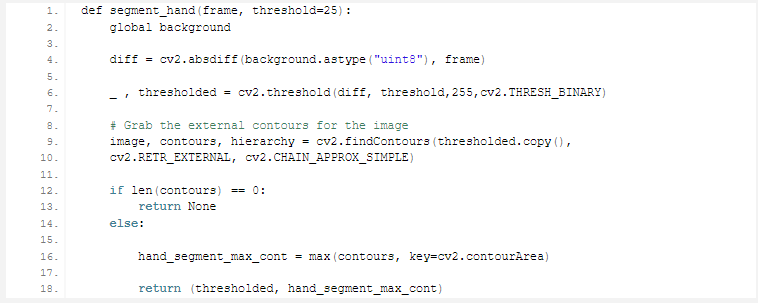
****



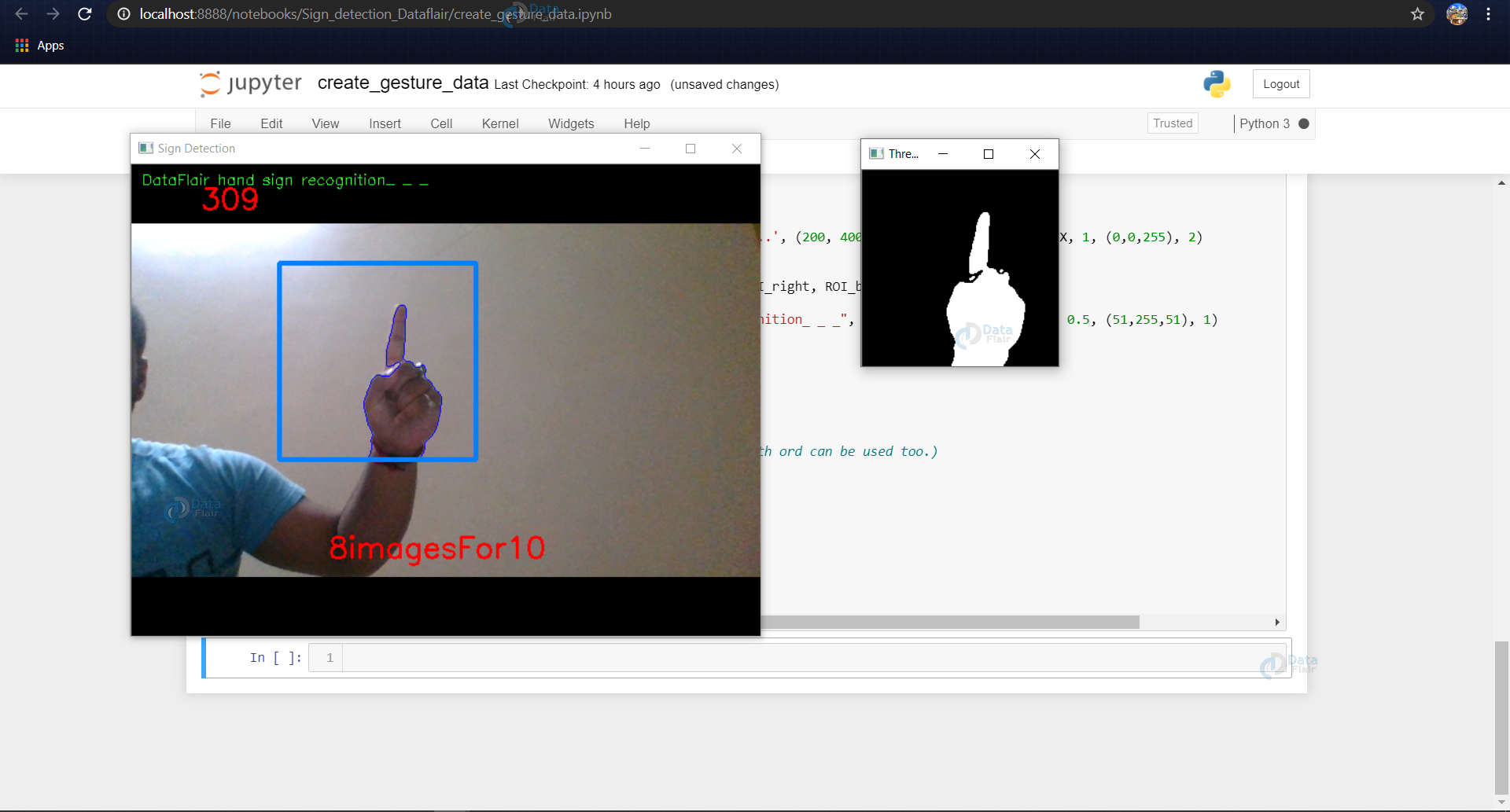
We put up a text using cv2.putText to display to wait and not put any object or hand in the ROI while detecting the background.

#### Calculate threshold value

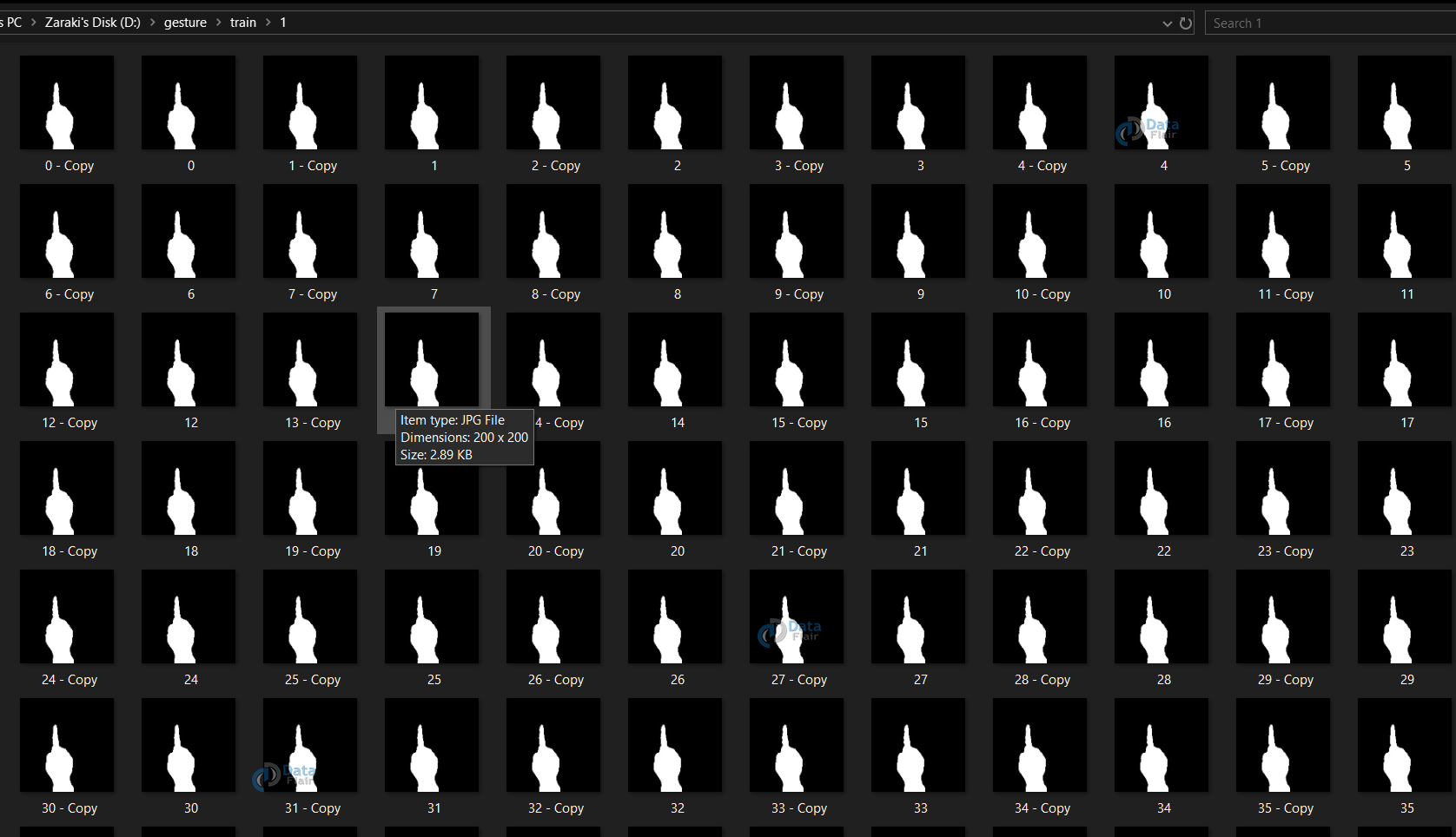
Now we calculate the threshold value for every frame and determine the contours using cv2.findContours and return the max contours (the most outermost contours for the object) using the function segment. Using the contours we are able to determine if there is any foreground object being detected in the ROI, in other words, if there is a hand in the ROI.

****

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



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/example.jpg

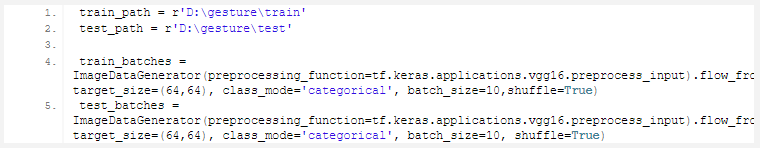


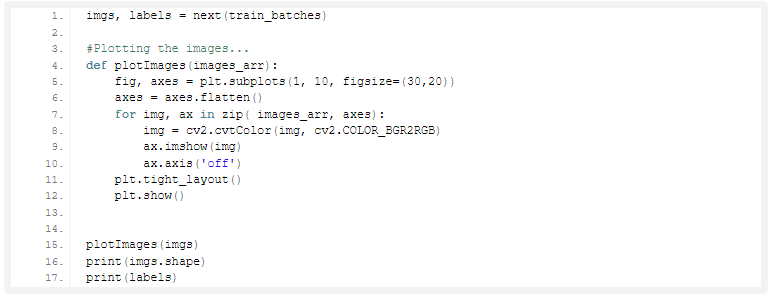
For the train dataset, we save 701 images for each number to be detected, and for the test dataset, we do the same and create 40 images for each number.

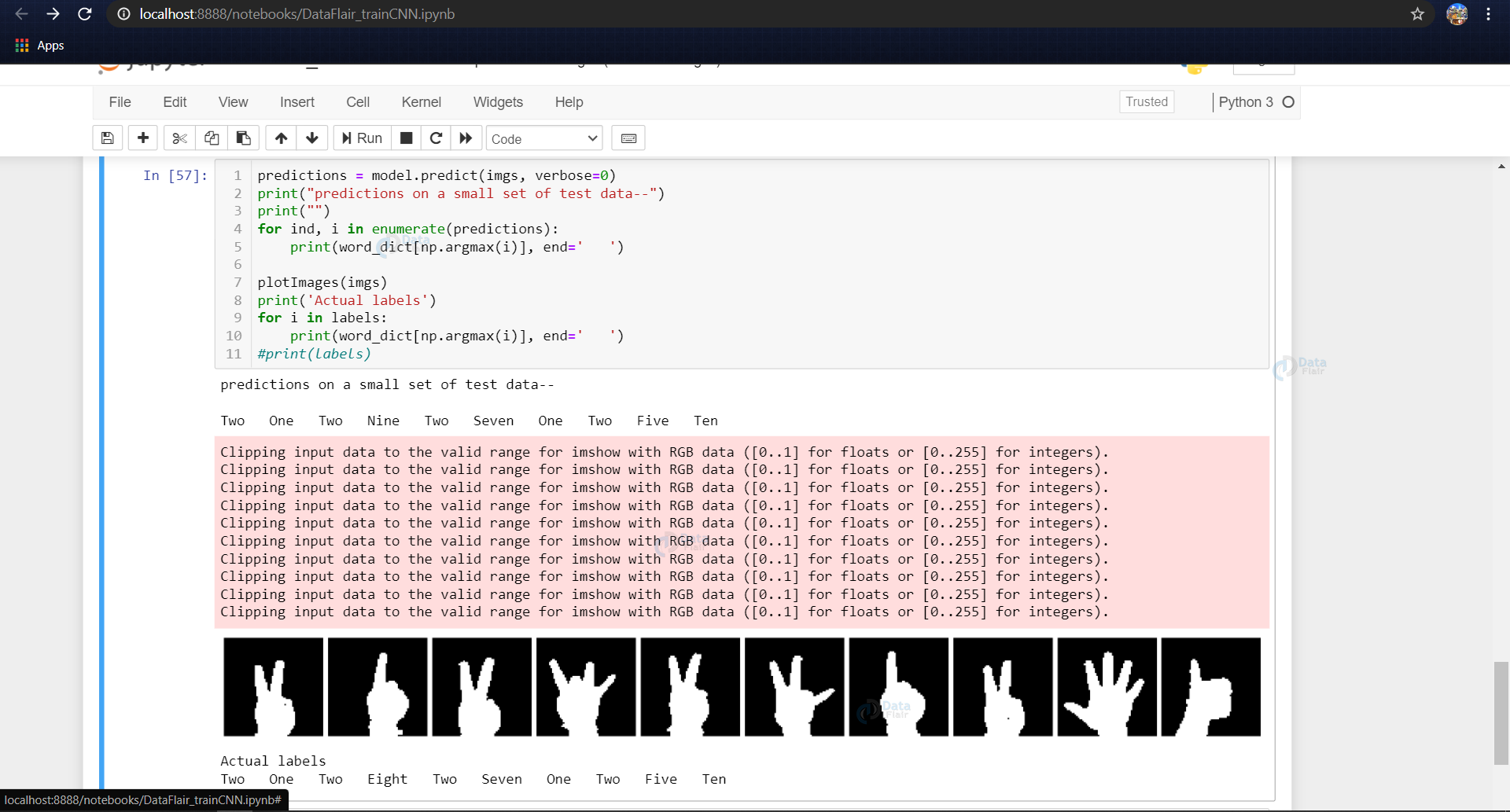
## 2. TRAINING CNN

Now on the created data set we train a CNN.

First, we load the data using ImageDataGenerator of keras through which we can use the flow\_from\_directory function to load the train and test set data, and each of the names of the number folders will be the class names for the imgs loaded.

****

****



Now we design the CNN (or depending upon some trial and error other hyperparameters can be used)

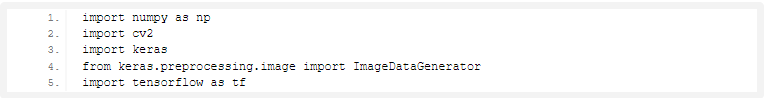
## 3. PREDICT THE GESTURE:

In this, we create a bounding box for detecting the ROI and calculate the accumulated\_avg as we did in creating the dataset. This is done for identifying any foreground object.

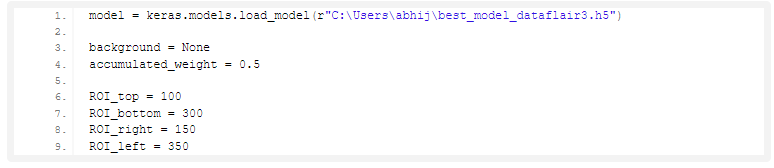
Now we find the max contour and if contour is detected that means a hand is detected so the threshold of the ROI is treated as a test image.

We load the previously saved model 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.

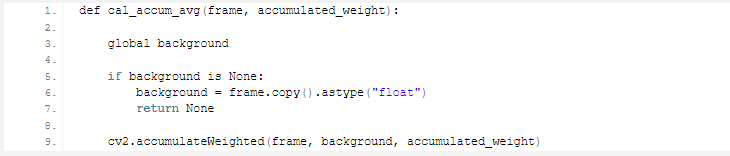
Getting the necessary imports for model\_for\_gesture.py



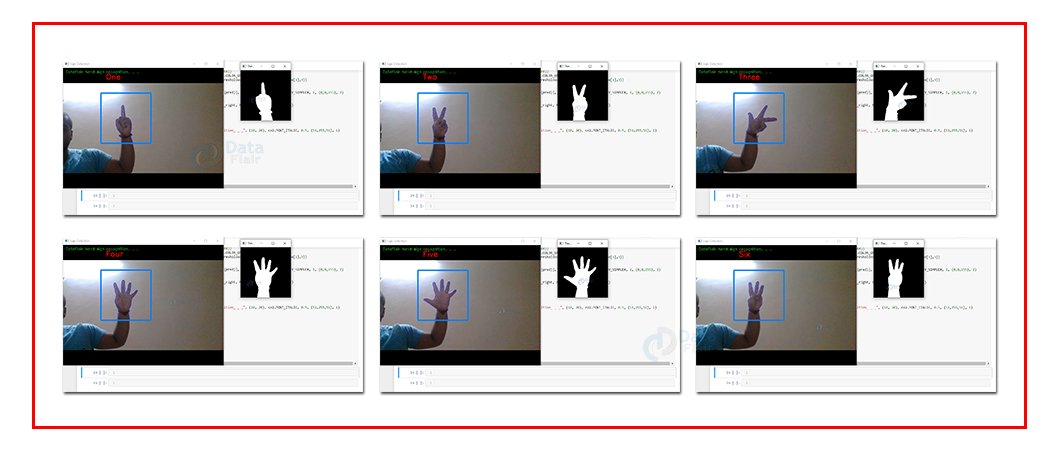
Now we load the model that we had created earlier and set some of the variables that we need, i.e, initializing the background variable, and setting the dimensions of the ROI.



Function to calculate the background accumulated weighted average (like we did while creating the dataset…)



# **Sign Language Recognition Output**



# **CONSTRAINT IDENTIFICATION:**

1. **Cost constraint:** The cost of the project, often dubbed the project’s budget, comprises all of the financial resources needed to complete the project on time, in its predetermined scope. Keep in mind that cost does not just mean money for materials. It encompasses costs for labor, vendors, quality control, and other factors, as well.
2. **Time constraint:** The time constraint refers to the project’s schedule for completion, including the deadlines for each phase of the project, as well as the date for rollout of the final deliverable.
3. **Scope constraint:** The scope of a project defines its specific goals, deliverables, features, and functions, in addition to the tasks required to complete the project.
4. **Monitoring:** This step occurs once the project is underway and requires the project team to analyze how the past stages of the project performed, noting trends and impacts on future plans, and communicating these findings to all relevant stakeholders.
5. **Control:** In the control step, the team must, upon communicating the results of each phase of the project, move forward accordingly. That means if things are running smoothly, the team must analyze the factors contributing to that positive outcome so that it can be continued and replicated. If there has been a derailment, the team must know how and why the derailment occurred and correct it for future actions.

# **ANALYSIS OF FEATURES AND FINALIZATION SUBJECT TO CONSTRAINTS:**

Vision is a key factor in sign language, and every sign language is intended to be understood by one person located in front of the other, from this perspective, a gesture can be completely observable. Viewing a gesture from another perspective makes it difficult or almost impossible to be understood since every finger position and movement will not be observable.

Trying to understand sign language from a first-vision perspective has the same limitations, some gestures will end up looking the same way. But, this ambiguity can be solved by locating more cameras in different positions. In this way, what a camera can’t see, can be perfectly observable by another camera.

**DESIGN SELECTION:**

## DFD DIAGRAM:

The DFD is also known as bubble chart. It is a simple graphical Formalism that can be used to represent a system in terms of the input data to the system, various Processing carried out on these data, and the output data is generated by the system. It maps out the flow of information for any process or system, how data is processed in terms of inputs and outputs. It uses defined symbols like rectangles, circles and arrows to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyse an existing system or model of a new one. A DFD can often visually “say” things that would be hard to explain in words and they work for both technical and non- technical. There are four components in DFD:

1. External Entity

2. Process

3. Data Flow

4. data Store

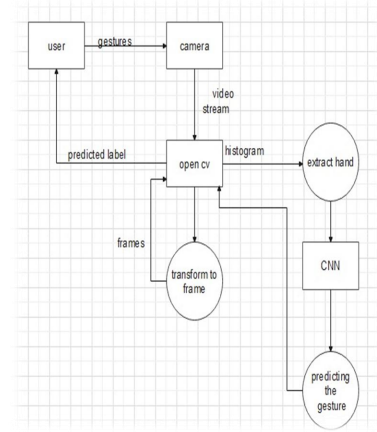


Fig-Dataflow Diagram for Sign Language Recognition

## USECASE DIAGRAM

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the task s accomplished by the system and the tasks accomplished by its environment. The actors are on the outside of the system's border, whilst the use cases are on the inside. The behaviour of the system as viewed through the eyes of the actor is described in a use case. It explains the system's role as a series of events that result in a visible consequence for the actor. Use Case Diagrams: What Are They Good For? The objective of a use case diagram is to capture a system's dynamic nature.. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams:

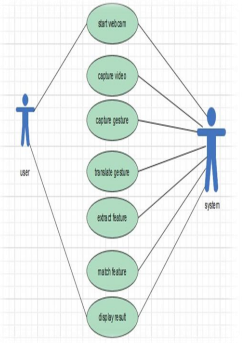


Fig: Usecase diagram of sign language recognition System

Table-1: Usecase Scenario for sign language recognition system



## CLASS DIAGRAM

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagram describe the different perspective when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes:

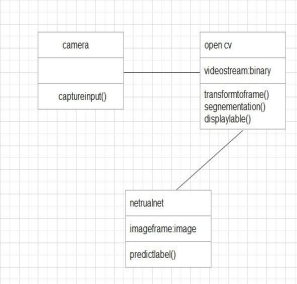


Fig: Class diagram of sign language recognition system

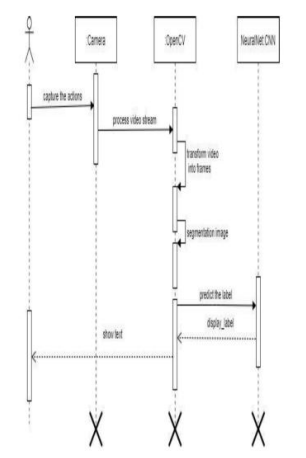
***SEQUENCE DIAGRAM***Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects). Objects: An object can be thought of as an entity that exists at a specified time and has a definite value, as well as a holder of identity. A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them in order to carry out the scenario's functionality. In the Logical View of the system under development, sequence diagrams are often related with use case realisations. Event diagrams and event scenarios are other names for sequence diagrams. A sequence diagram depicts multiple processes or things that exist simultaneously as parallel vertical lines (lifelines), and the messages passed between them as horizontal arrows, in the order in which they occur. This enables for the graphical specification of simple runtime scenarios.

Fig: Sequence diagram of sign language recognition system

***STATE CHART***

A state chart diagram describes a state machine which shows the behavior of classes. It shows the actual changes in state not processes or commands that create those changes and is the dynamic behavior of objects over time by model1ing the life cycle of objects of each class. It describes how an object is changing from one state to another state. There are mainly two states in State Chart Diagram: 1. Initial State 2. Final-State. Some of the components of State Chart Diagram are: State: It is a condition or situation in life cycle of an object during which it’s satisfies same condition or performs some activity or waits for some event. Transition: It is a relationship between two states indicating that object in first state performs some actions and enters into the next state or event. Event: An event is specification of significant occurrence that has a location in time and space

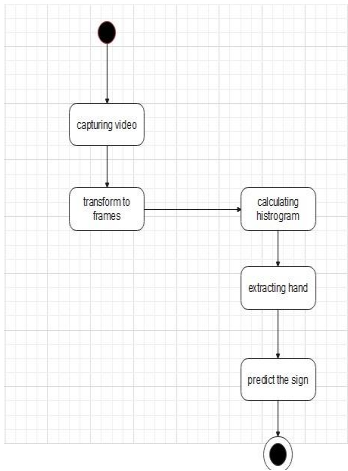


Fig:State Chart diagram of sign language recognition system