Project Phase IV

Report

On

SIGN LANGUAGE RECOGNITION USING MACHINE LEARNING

BACHELOR OF ENGINEERING

COMPUTER SCIENCE & ENGINEERING

Submitted to:
Er. Parwinder Kaur (Supervisor)
(e12551)

Submitted By:
Student Group
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:

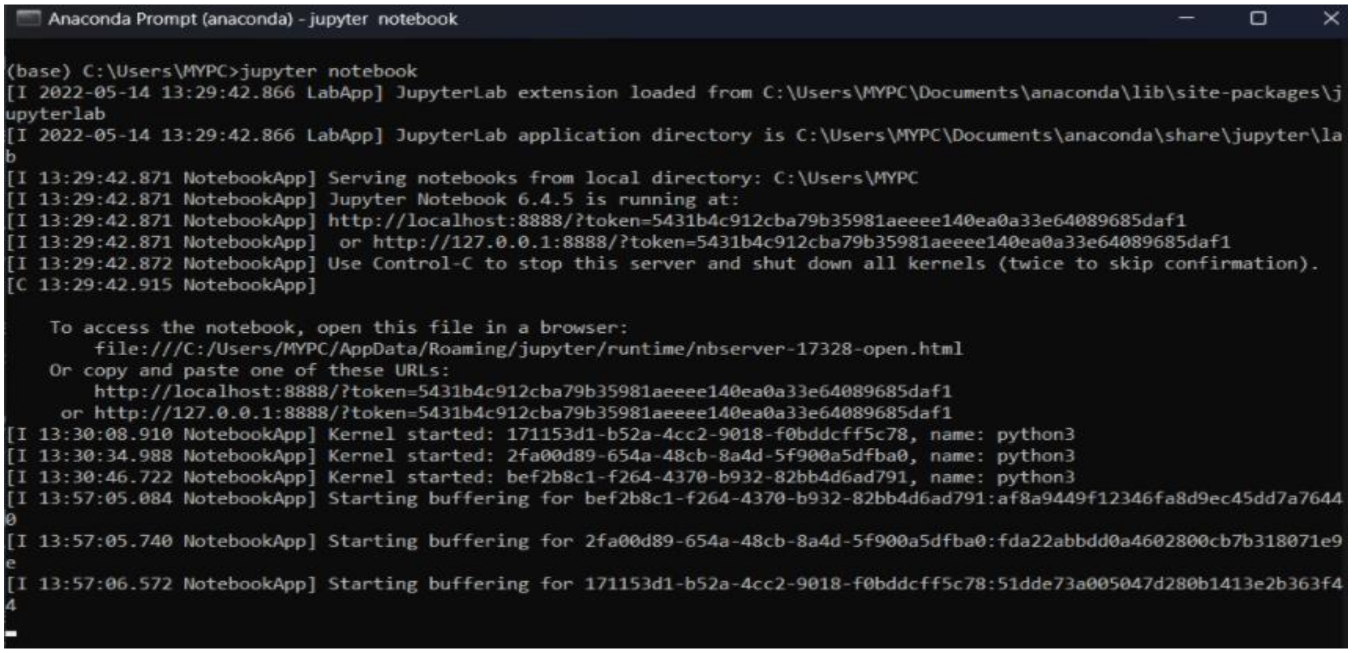
USE OF MODERN TOOLS IN DESIGN & ANALYSIS:

In this project, we have used various modern tools to edit the code of the project, for the implementation, for the live working of the project, etc. The names of the tools are as follows:

- Anaconda Prompt
- Anaconda Navigator
- Jupyter Notebook
- Github Desktop

Anaconda Prompt

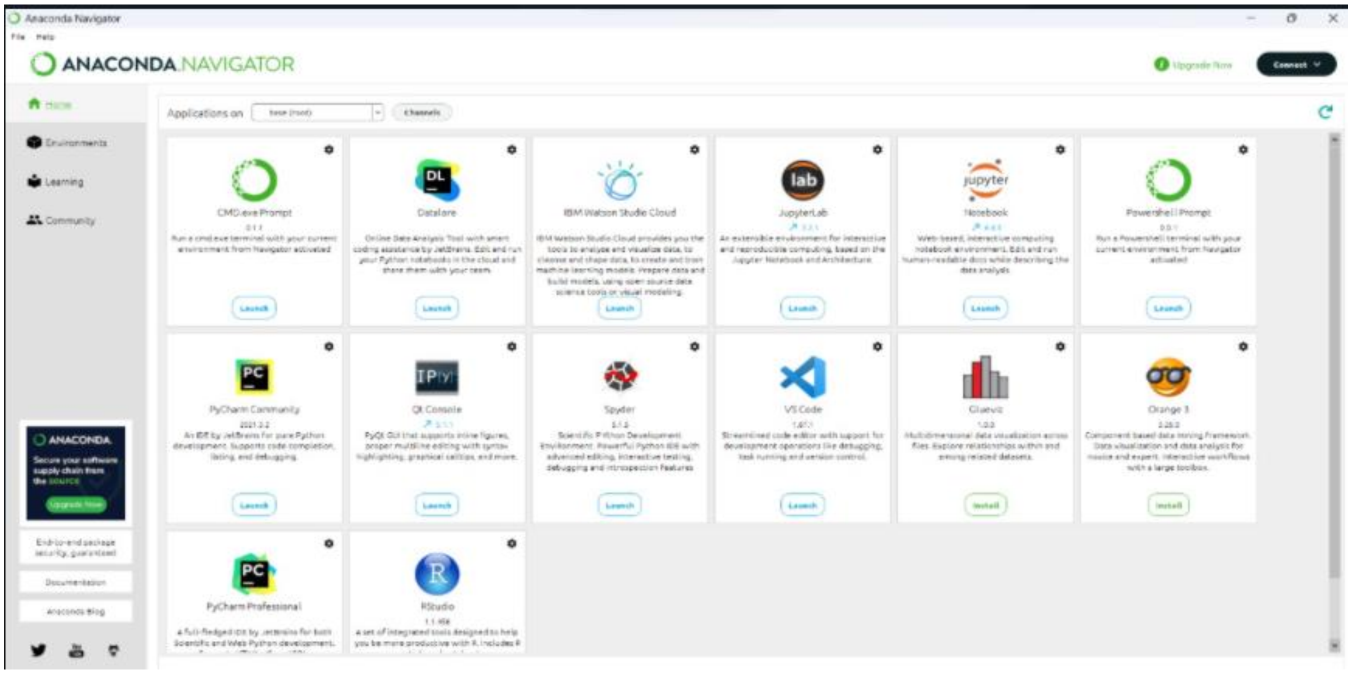
Anaconda is a Python distribution (prebuilt and preconfigured collection of packages) that is commonly used for data science. The Anaconda distribution includes the Conda package manager in addition to the preconfigured Python packages and other tools. The Conda package manager can be used from the command line to set up Python environments and install additional packages that come with the default Anaconda distribution.



```
Anaconda Prompt (anaconda) - jupyter  notebook

(base) C:\Users\MYPC>jupyter notebook
[I 2022-05-14 13:29:42.866 LabApp] JupyterLab extension loaded from C:\Users\MYPC\Documents\anaconda\lib\site-packages\j
upyterlab
[I 2022-05-14 13:29:42.866 LabApp] JupyterLab application directory is C:\Users\MYPC\Documents\anaconda\share\jupyter\la
b
[I 13:29:42.871 NotebookApp] Serving notebooks from local directory: C:\Users\MYPC
[I 13:29:42.871 NotebookApp] Jupyter Notebook 6.4.5 is running at:
[I 13:29:42.871 NotebookApp] http://localhost:8888/?token=5431b4c912cba79b35981aeeee140ea0a33e64089685daf1
[I 13:29:42.871 NotebookApp]  or http://127.0.0.1:8888/?token=5431b4c912cba79b35981aeeee140ea0a33e64089685daf1
[I 13:29:42.872 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:29:42.915 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/MYPC/AppData/Roaming/jupyter/runtime/nbserver-17328-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=5431b4c912cba79b35981aeeee140ea0a33e64089685daf1
     or http://127.0.0.1:8888/?token=5431b4c912cba79b35981aeeee140ea0a33e64089685daf1
[I 13:30:08.910 NotebookApp] Kernel started: 171153d1-b52a-4cc2-9018-f0bddcff5c78, name: python3
[I 13:30:34.988 NotebookApp] Kernel started: 2fa00d89-654a-48cb-8a4d-5f900a5dfba0, name: python3
[I 13:30:46.722 NotebookApp] Kernel started: bef2b8c1-f264-4370-b932-82bb4d6ad791, name: python3
[I 13:57:05.084 NotebookApp] Starting buffering for bef2b8c1-f264-4370-b932-82bb4d6ad791:af8a9449f12346fa8d9ec45dd7a7644
0
[I 13:57:05.740 NotebookApp] Starting buffering for 2fa00d89-654a-48cb-8a4d-5f900a5dfba0:fda22abbdd0a4602800cb7b318071e9
e
[I 13:57:06.572 NotebookApp] Starting buffering for 171153d1-b52a-4cc2-9018-f0bddcff5c78:51dde73a005047d280b1413e2b363f4
4
```

Anaconda Navigator

Anaconda Navigator is a GUI tool that is included in the Anaconda distribution and makes it easy to configure, install and launch tools such as Jupyter Notebook. Although we use the Anaconda Navigator in this book, keep in mind that you can do everything through the command line using the conda command.



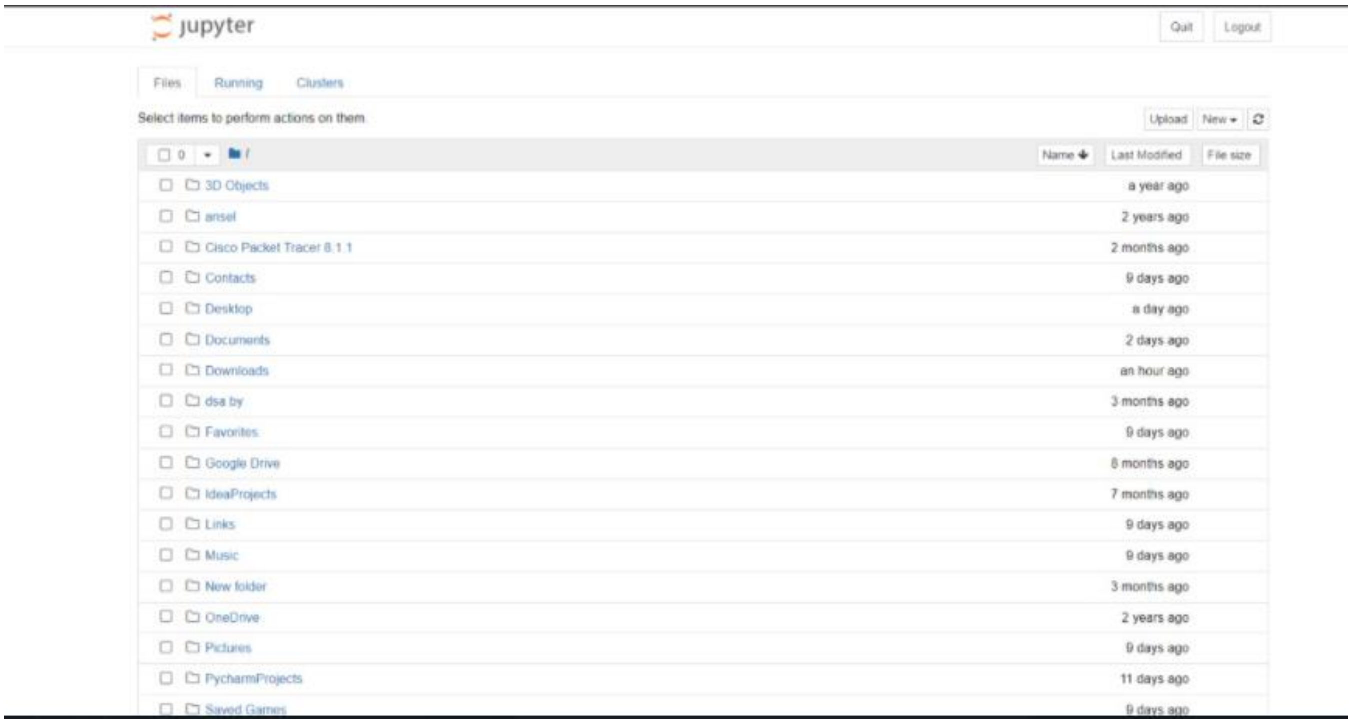
Jupyter Notebook:

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Another situation where developers often use Jupyter is to create documentation or tutorials for their team. You can explain your thought process much better in a notebook rather than using internal comments within your code.

Jupyter notebooks have three particularly strong benefits:

- They're great for showcasing your work. You can see both the code and the results. ...
- It's easy to use other people's work as a starting point.
- Very easy to host server-side, which is useful for security purposes.

Due to Jupyter Notebook's multi-programming support, huge feature availability and rapidly growing popularity among the community, it has become a standard for all sorts of analysis, visualizations, rapid prototyping, ML and various code practices.

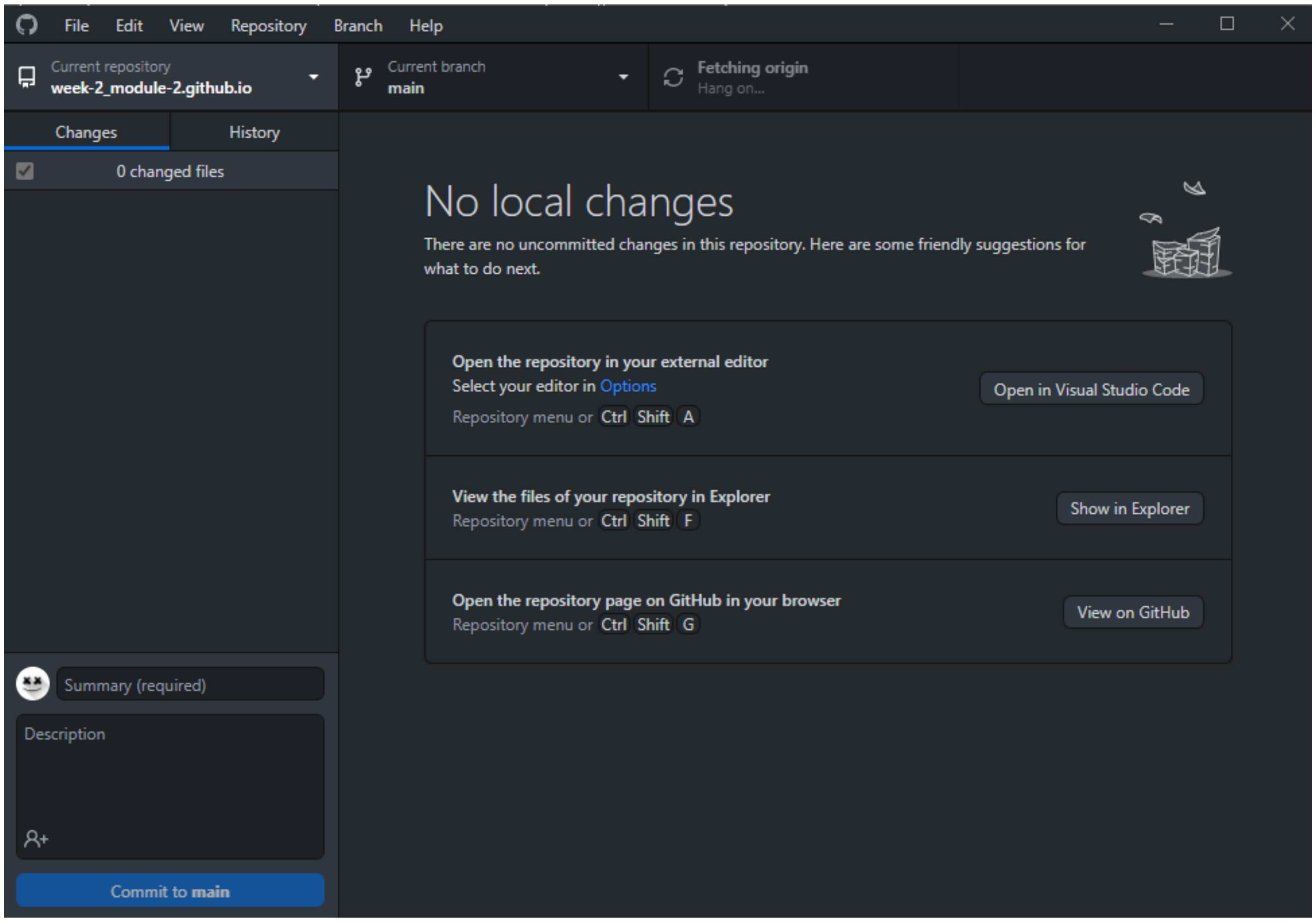


Select items to perform actions on them

Name	Last Modified	File size
3D Objects	a year ago	
ansel	2 years ago	
Cisco Packet Tracer 8.1.1	2 months ago	
Contacts	9 days ago	
Desktop	a day ago	
Documents	2 days ago	
Downloads	an hour ago	
dsa by	3 months ago	
Favorites	9 days ago	
Google Drive	8 months ago	
IdeaProjects	7 months ago	
Links	9 days ago	
Music	9 days ago	
New folder	3 months ago	
OneDrive	2 years ago	
Pictures	9 days ago	
PycharmProjects	11 days ago	
Saved Games	9 days ago	

Github Desktop:

GitHub Desktop is an open-source project. You can see the roadmap, contribute to the project, or open an issue to provide feedback or feature requests. For more information, see the ____________ repository.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G

We use GitHub Desktop to create issues or pull requests to collaborate on projects with other people. Issues help us keep track of ideas and discuss possible changes to projects. Pull requests let us share our proposed changes with others, receive feedback, and merge changes into a project.

We can view our own or our collaborator's pull requests in GitHub Desktop. Viewing a pull request in GitHub Desktop lets us see any proposed changes and make additional changes by opening the project's files and repositories in your default text editor.

DISCUSSION AND REPORT ANALYSIS:

EXISTING SYSTEM

In Literature survey we have gone through other similar works that are implemented in the domain of sign language recognition.The summaries of each of theproject works are mentioned below

A Survey of Hand Gesture Recognition Methods in Sign LanguageRecognition

Sign Language Recognition (SLR) system, which is required to recognize sign languages, has been widely studied for years.The studies are based on various input sensors, gesture segmentation, extraction of features and classifcation methods.This paper aims to analyze and compare the methods employed in the SLR systems, classi cations methods that have been used, and suggests the most promising method for future research. Due to recent advancement in classifcationmethods, many of the recent proposed works mainly contribute on the classifcation methods, such as hybrid method and Deep Learning. This paper focuses on the classifcation methodsused in prior Sign Language Recognition system. Based on our review, HMM- based approaches have been explored extensively in prior research, including its modifcations.

This study is based on various input sensors, gesture segmentation, extraction of features and classification methods. This paper aims to analyze and compare the methods employed in the SLR systems, classifications methods that have been used, and suggests the most reliable method for future research. Due to recent advancement in classification methods, many of the recently proposed works mainly contribute to the classification methods, such as hybrid method and Deep Learning. Based on our review, HMM-based approaches have been explored extensively in prior research, including its modifications. Hybrid CNN-HMM and fully Deep Learning approaches have shown promising results and offer opportunities for further exploration.

Communication between Deaf-Dumb People and Normal People

Chat applications have become a powerful mediathat assist people to communicate in different languages witheach other. There are lots of chat applications that are useddifferent people in different languages but there are not such achat application that has facilitate to communicate with signlanguages. The developed system isbased on Sinhala Sign language. The system has included fourmain components as text messages are converted to sign messages, voice messages are converted to sign messages, signmessages are converted to text messages and sign messages areconverted to voice messages. Google voice recognition API hasused to develop speech character recognition for voice messages.The system has been trained for the speech and text patterns by usingsome text parameters and signs of Sinhala Sign language isdisplayed by emoji. Those emoji and signs that are included inthis system will bring the normal people more close to the disabled people. This is a 2 way communication system but it uses pattern of gesture recognition which is not very realiable in getting appropriate output.

A System for Recognition of Indian Sign Language for Deaf People using Otsu's Algorithm

In this paper we proposed some methods,through which the recognition of the signs becomes easy forpeoples while communication. And the result of thosesymbols signs will be converted into the text. In this project,we are capturing hand gestures through webcam andconvert this image into gray scale image. The segmentationof gray scale image of a hand gesture is performed usingOtsu thresholdingalgorithm.. Total image level is dividedinto two classes one is hand and other is background. Theoptimal threshold value is determined by computing the ratio between-class variance and total class variance. Tofind the boundary of hand gesture in image Canny edgedetection technique is used.In Canny edge detection we used edge based segmentation and threshold based segmentation. Then Otsu's algorithm is used because of its simple calculation and stability. This algorithm fails, when the global distribution of the target and background vary widely.

Hand Gesture Recognition based on Digital Image Processing using MATLAB

This research work presents a prototype system that helps to recognize hand gesture to normal people in order to communicate more effectively with the special people. Aforesaid research work focuses on the problem of gesture recognition in real time that sign language used by the community of deaf people. The problem addressed is based on Digital Image Processing using Color Segmentation, Skin Detection, Image Segmentation, Image Filtering, and Template Matching techniques. This system recognizes gestures of ASL (American Sign Language) including the alphabet and a subset of its words.

Gesture Recognition System:

Communication plays a crucial part in human life. It encourages a man to pass on his sentiments, feelings and messages by talking, composing or by utilizing some other medium. Gesture based communication is the main method for communication for the discourse and hearing weakened individuals. Communication via gestures is a dialect that utilizations outwardly transmitted motions that consolidates hand signs anddevelopment of the hands, arms, lip designs, body developments and outward appearances, rather than utilizing discourse or content, to express the individual's musings. Gestures are the expressive and important body developments that speaks to some message or data. Gestures are the requirement for hearing and discourse hindered, they pass on their message to others just with the assistance of motions. Gesture Recognition System is the capacity of the computer interface to catch, track and perceive the motions and deliver the yield in light of the caught signals. It enables the clients to interface with machines (HMI) without the any need of mechanical gadgets. There are two sorts of sign recognition methods: image-based and sensor- based strategies. Image based approach is utilized as a part of this project that managescommunication via gestures motions to distinguish and track the signs and change over them into the relating discourse and content.

PROPOSED SYSTEM

Our proposed system is sign language recognition system using convolution neural networks which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus, our system is more robustin getting exact text labels of letters.

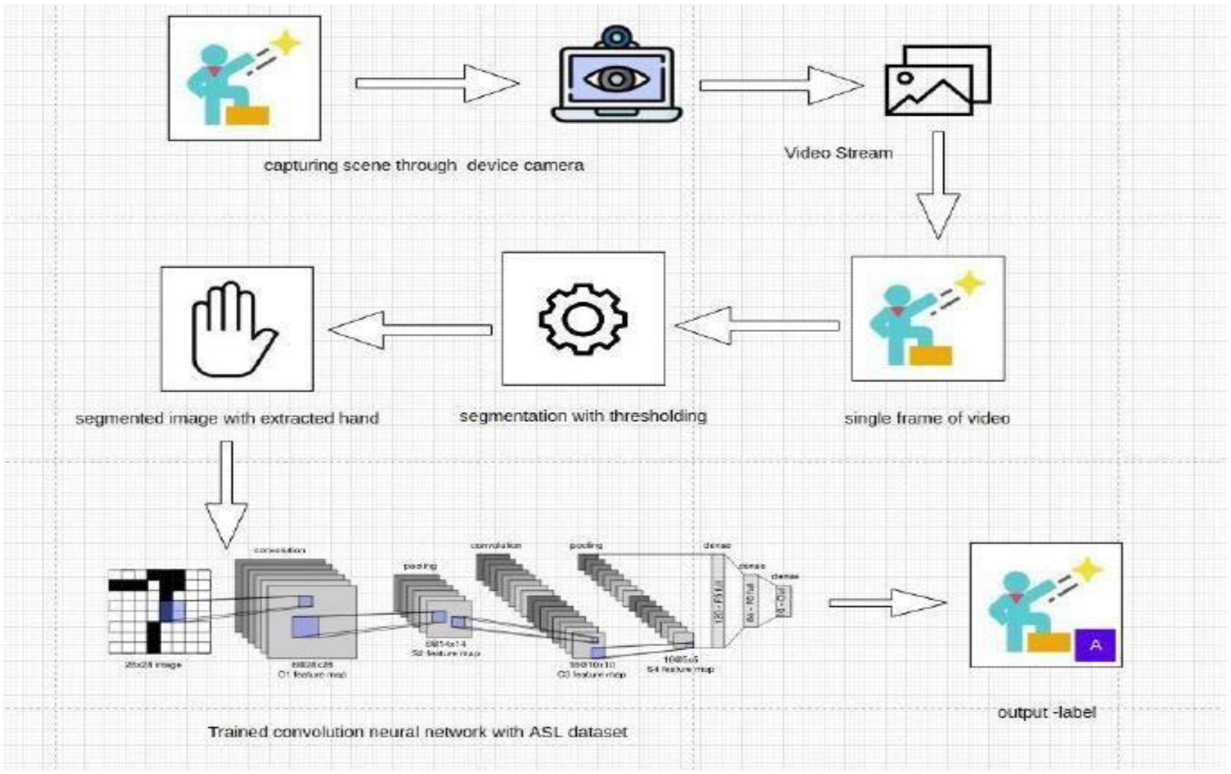
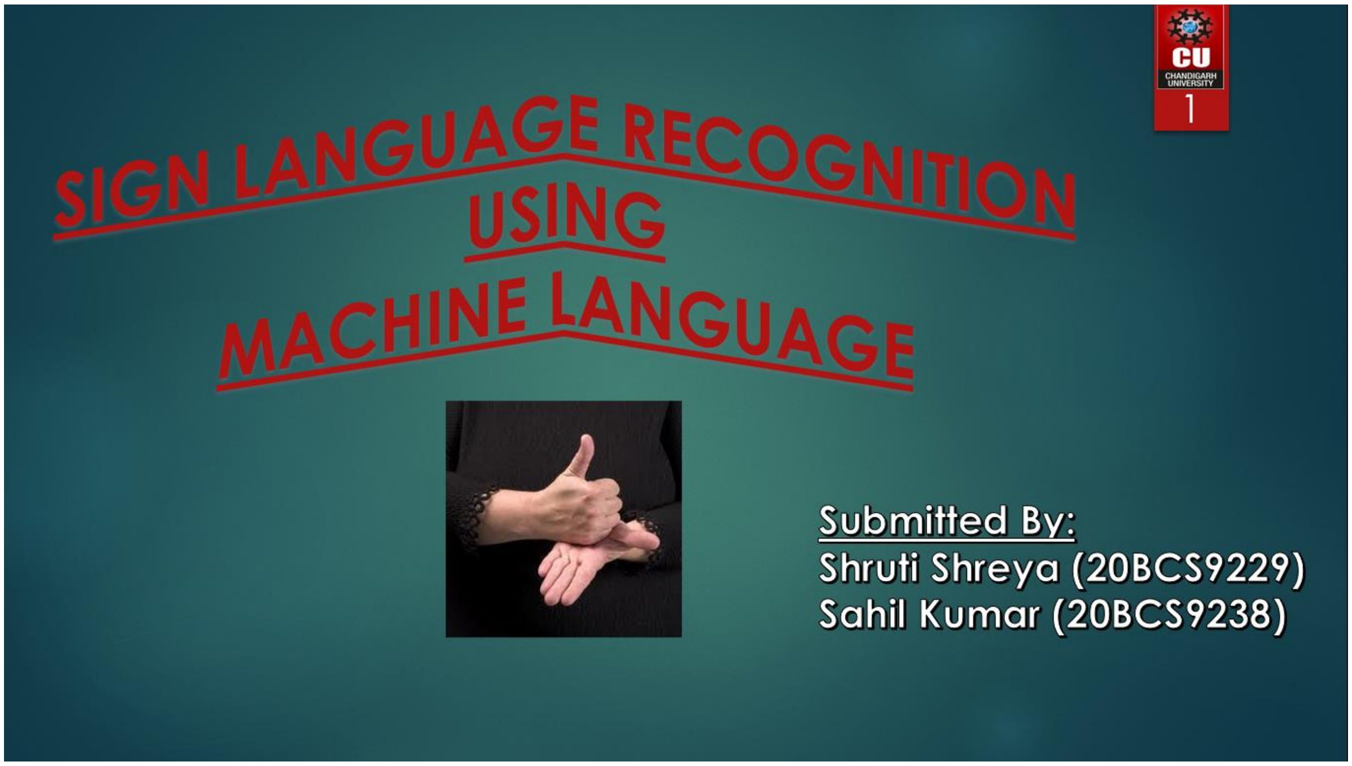

capturing scene through device camera
Video Stream
segmented image with extracted hand
segmentation with thresholding
single frame of video
Trained convolution neural network with ASL dataset
output -label
A

Fig: Architecture of Sign Language recognition System

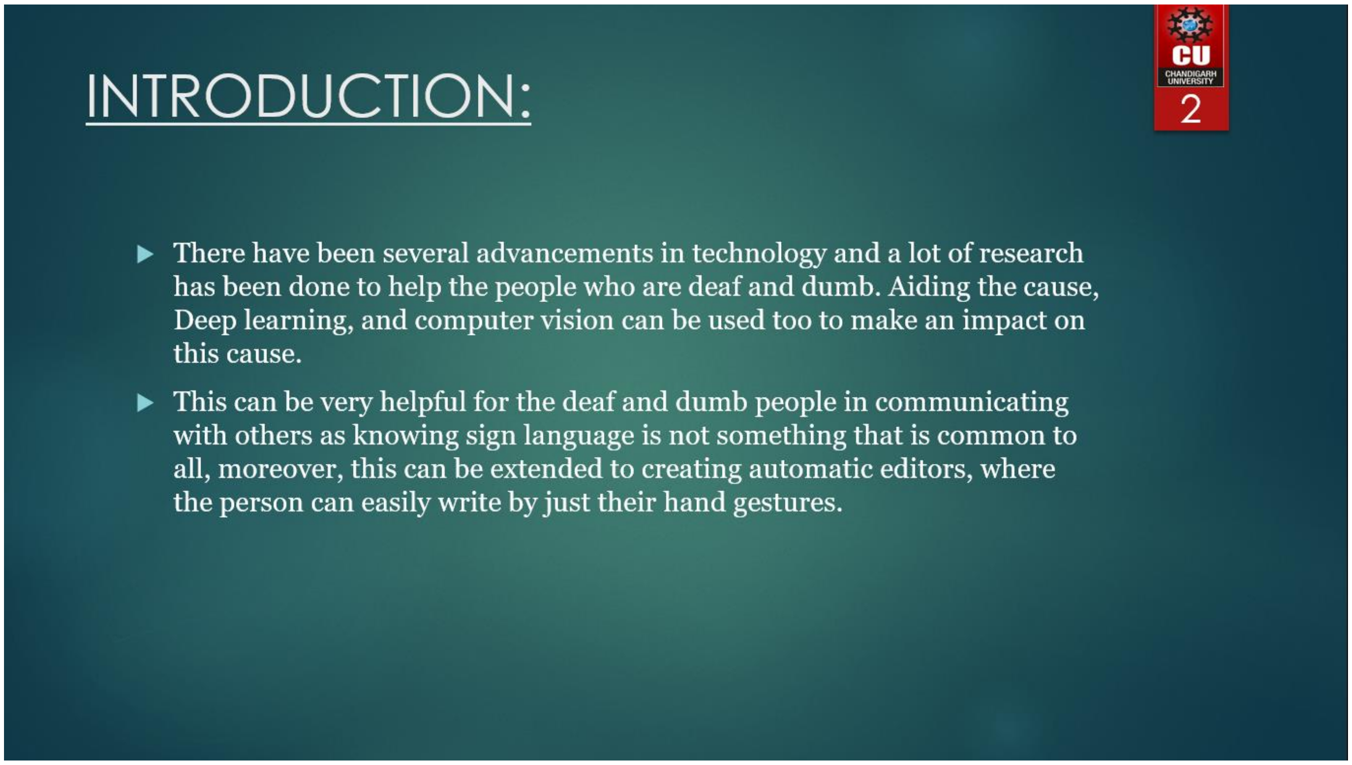
PROJECT MANAGEMENT AND PROFESSIONAL COMMUNICATION (PPT):



SIGN LANGUAGE RECOGNITION USING MACHINE LANGUAGE

Submitted By:
Shruti Shreya (20BCS9229)
Sahil Kumar (20BCS9238)

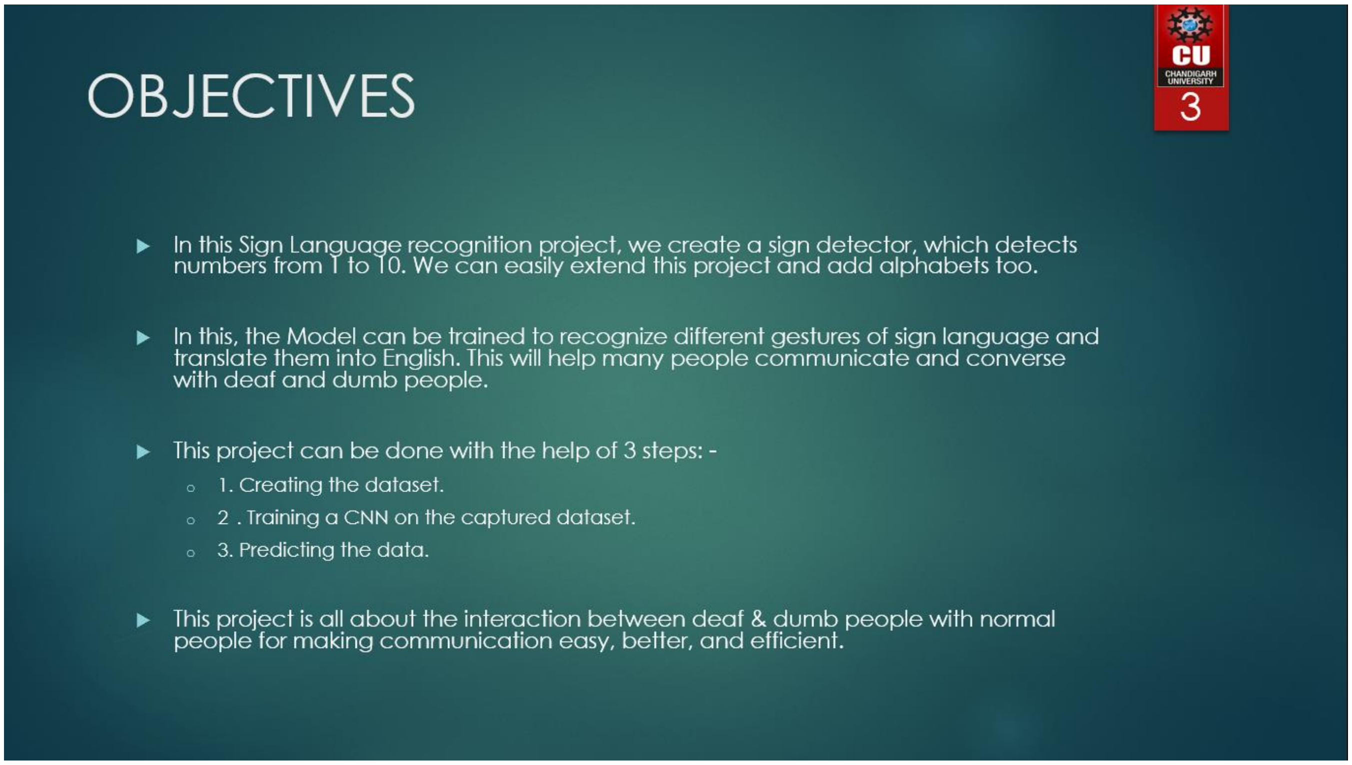
1



INTRODUCTION:

- 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.

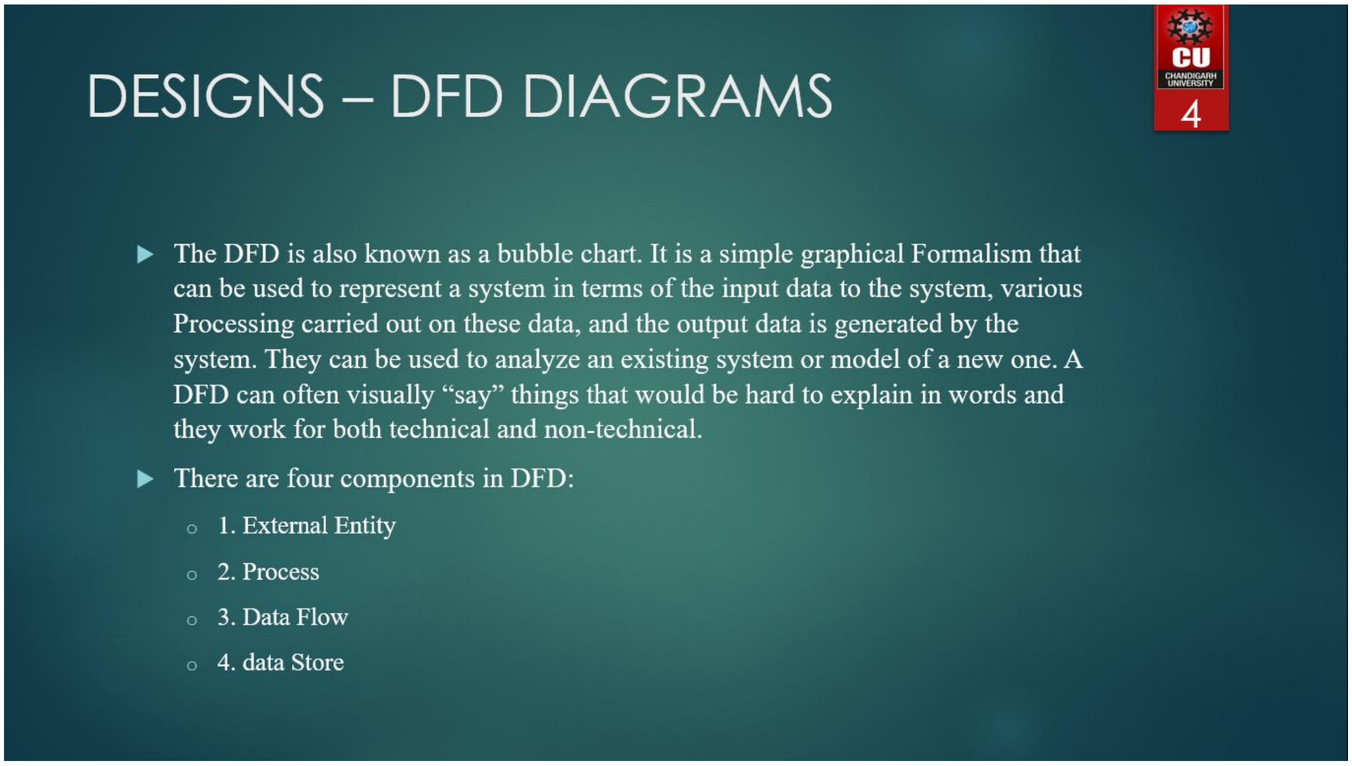
2



OBJECTIVES

3

- In this Sign Language recognition project, we create a sign detector, which detects numbers from 1 to 10. We can easily extend this project and add alphabets too.
- In this, the Model can be trained to recognize different gestures of sign language and translate them into English. This will help many people communicate and converse with deaf and dumb people.
- This project can be done with the help of 3 steps: -
  - 1. Creating the dataset.
  - 2 . Training a CNN on the captured dataset.
  - 3. Predicting the data.
- This project is all about the interaction between deaf & dumb people with normal people for making communication easy, better, and efficient.

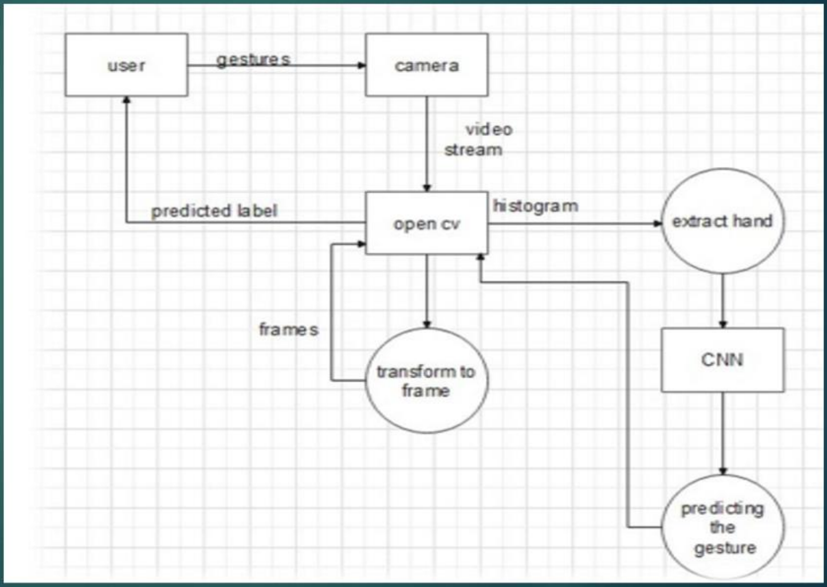


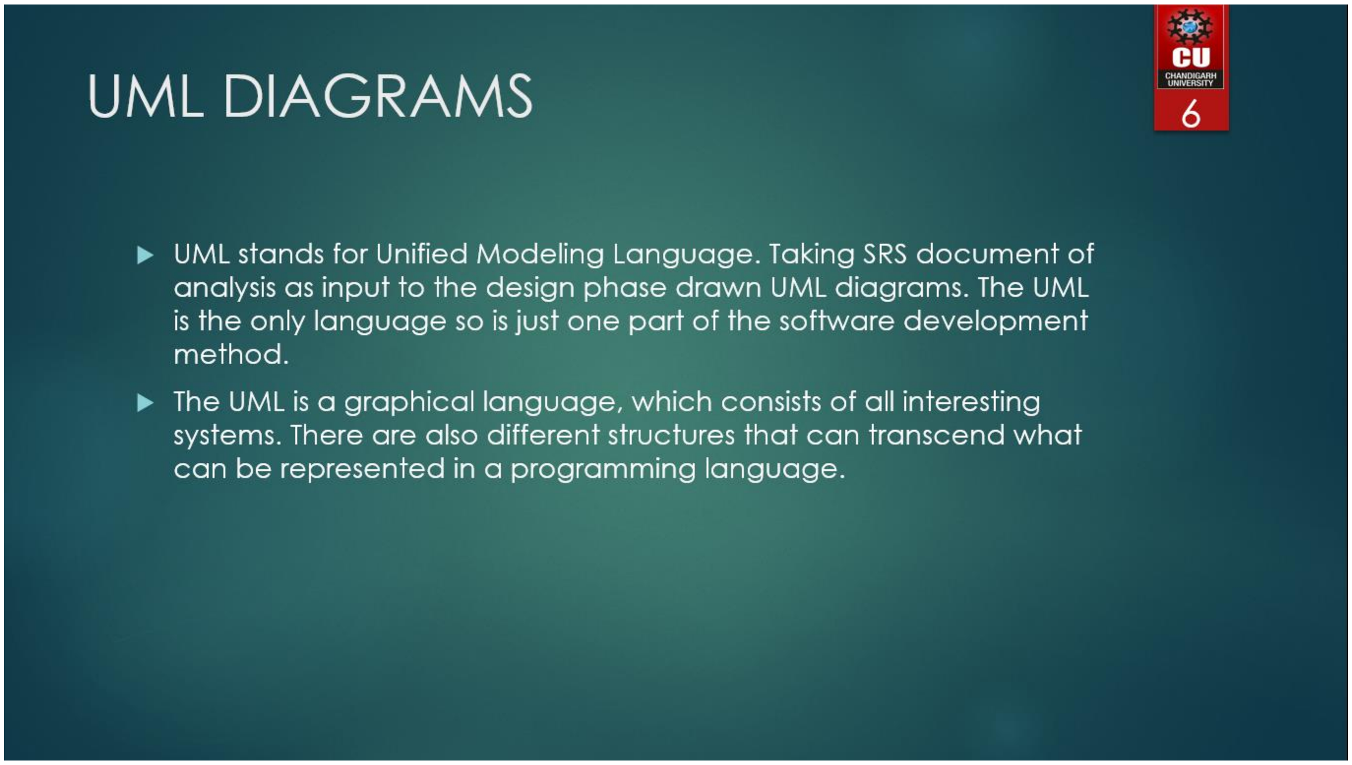
DESIGNS – DFD DIAGRAMS

4

- The DFD is also known as a 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. They can be used to analyze 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

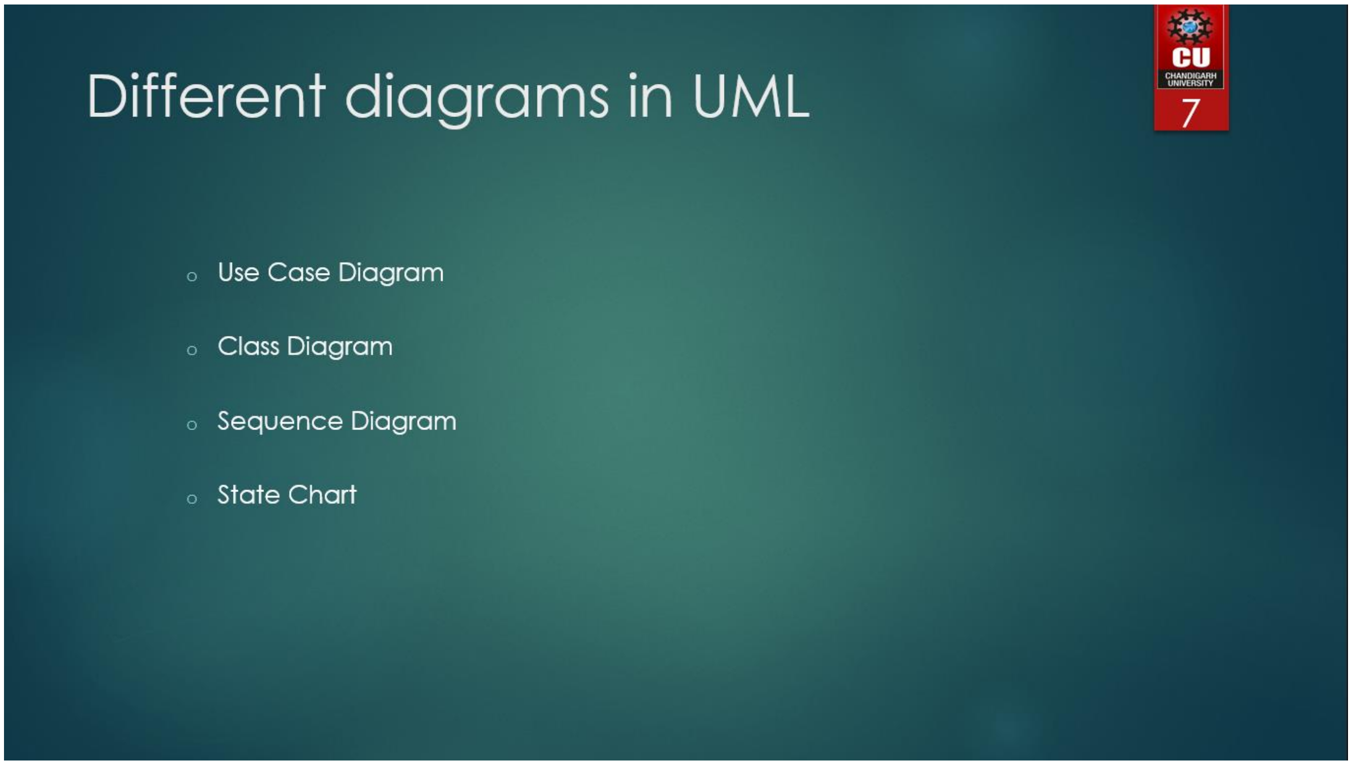
Dataflow Diagram for Sign Language Recognition





UML DIAGRAMS

- UML stands for Unified Modeling Language. Taking SRS document of analysis as input to the design phase drawn UML diagrams. The UML is the only language so is just one part of the software development method.
- The UML is a graphical language, which consists of all interesting systems. There are also different structures that can transcend what can be represented in a programming language.



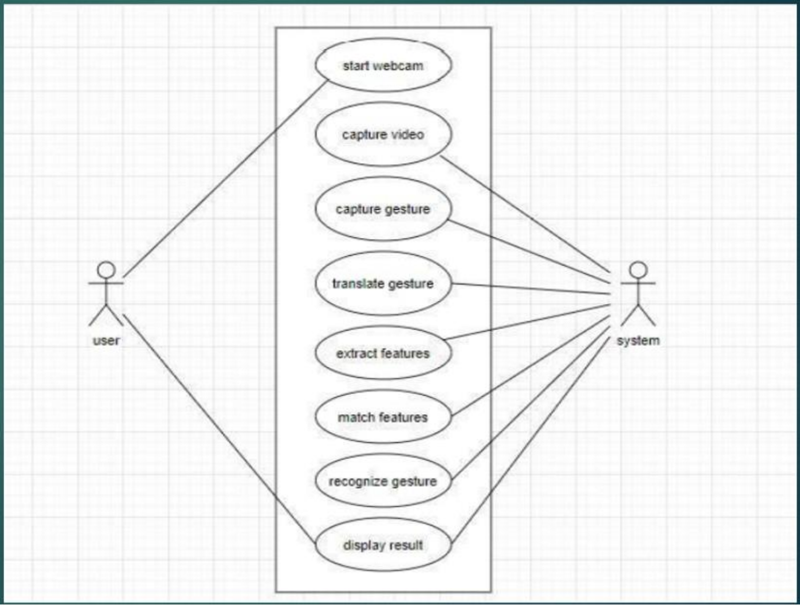
Different diagrams in UML

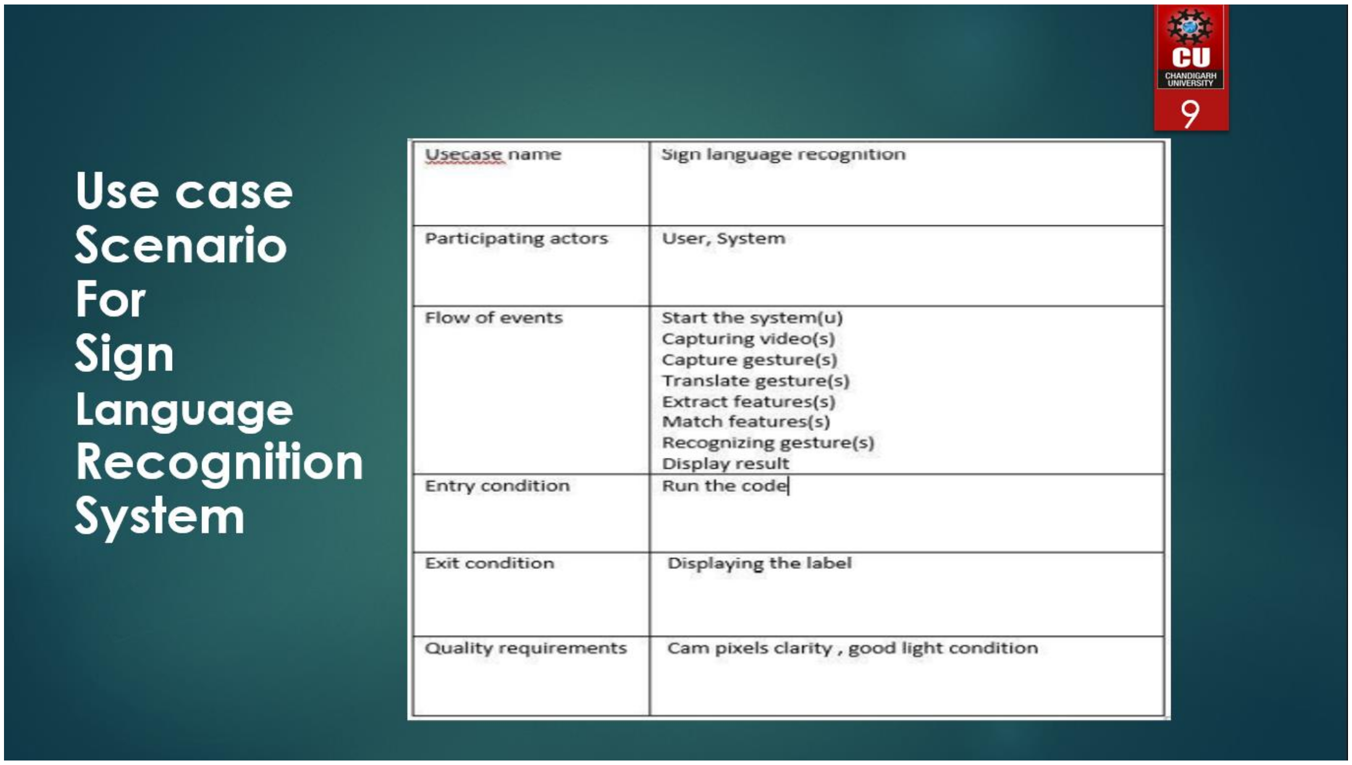
7

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- State Chart

8

Use case Diagram For Sign Language Recognition System





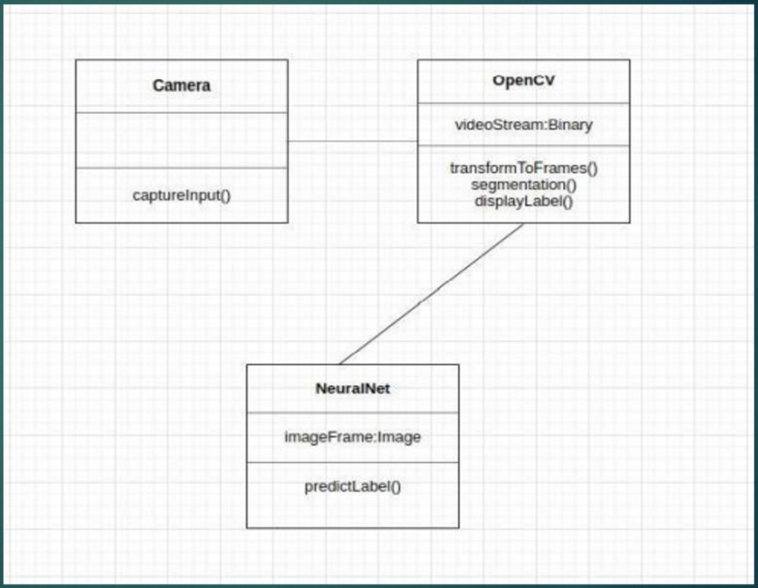
9

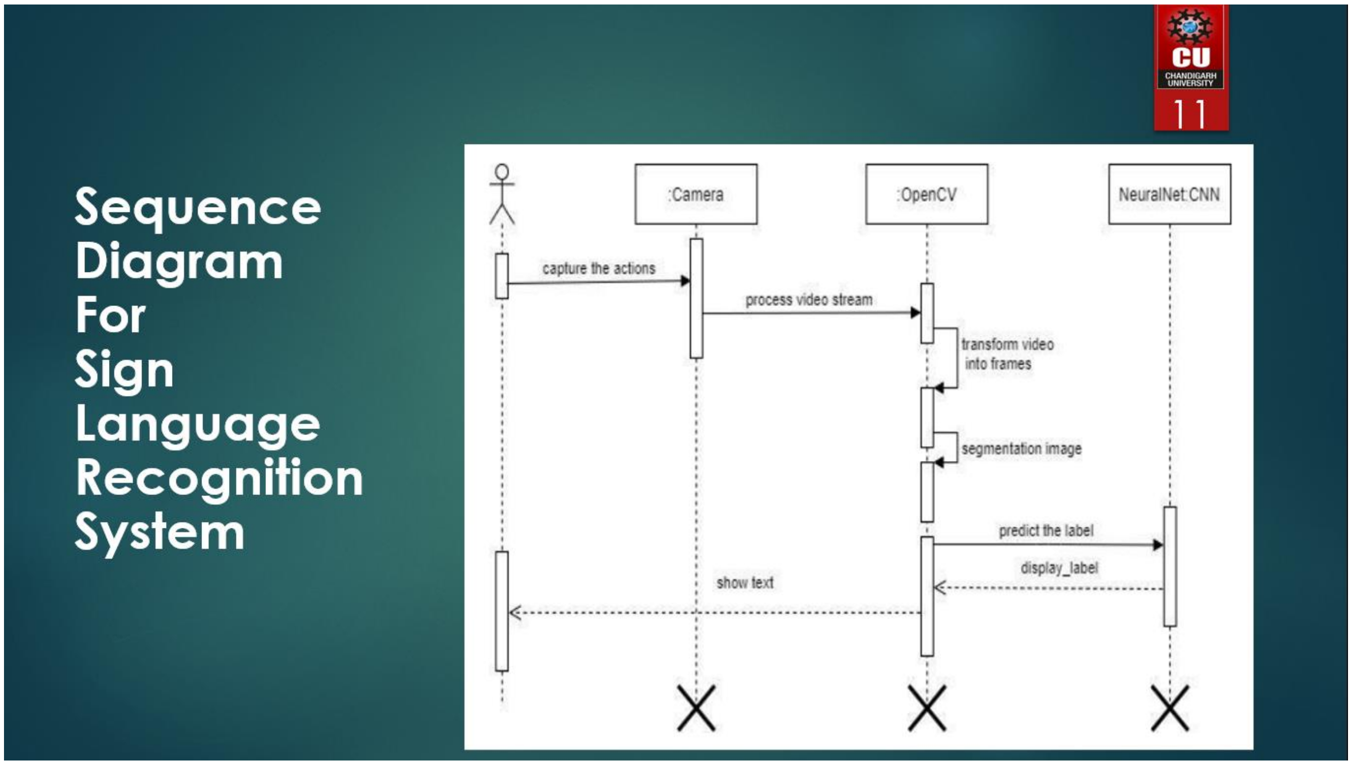
Use case Scenario For Sign Language Recognition System

Usecase name	Sign language recognition
Participating actors	User, System
Flow of events	Start the system(u) Capturing video(s) Capture gesture(s) Translate gesture(s) Extract features(s) Match features(s) Recognizing gesture(s) Display result
Entry condition	Run the code
Exit condition	Displaying the label
Quality requirements	Cam pixels clarity , good light condition

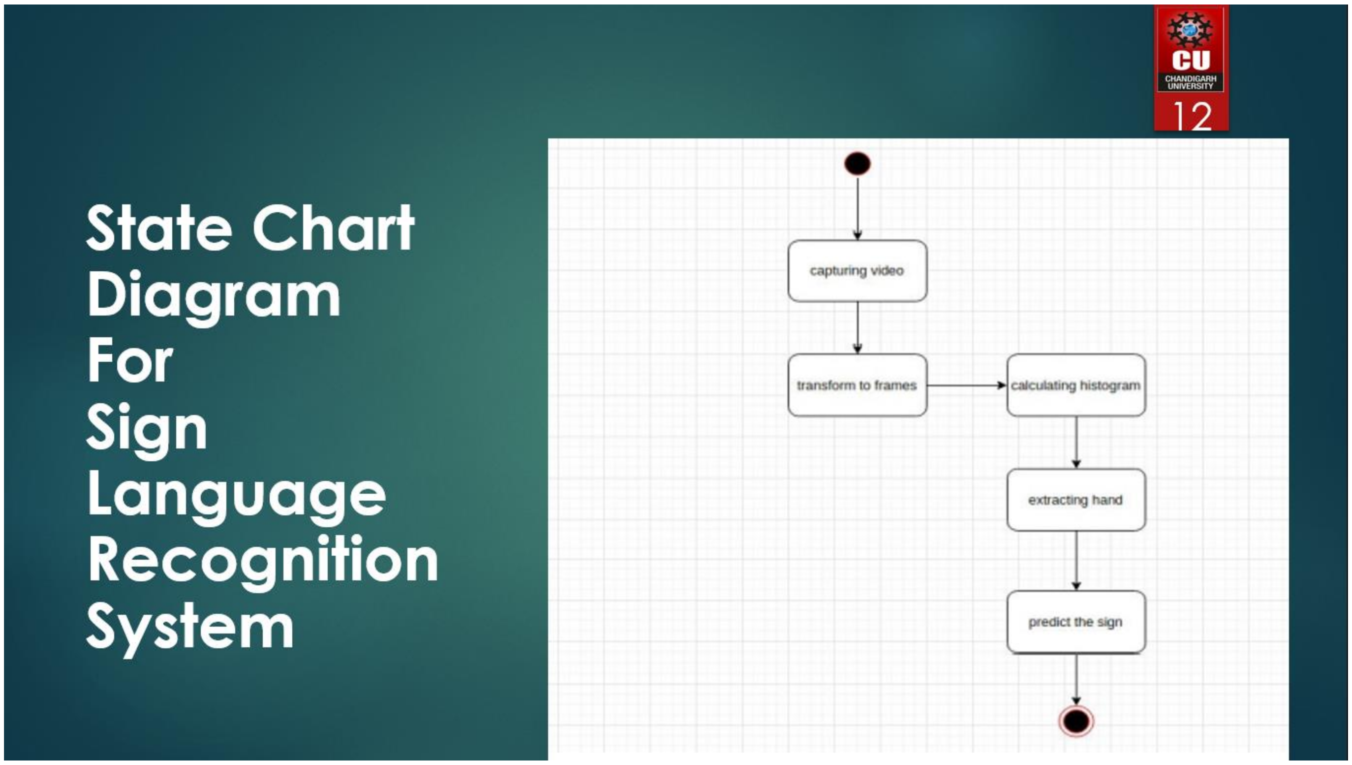
10

Class Diagram For Sign Language Recognition System

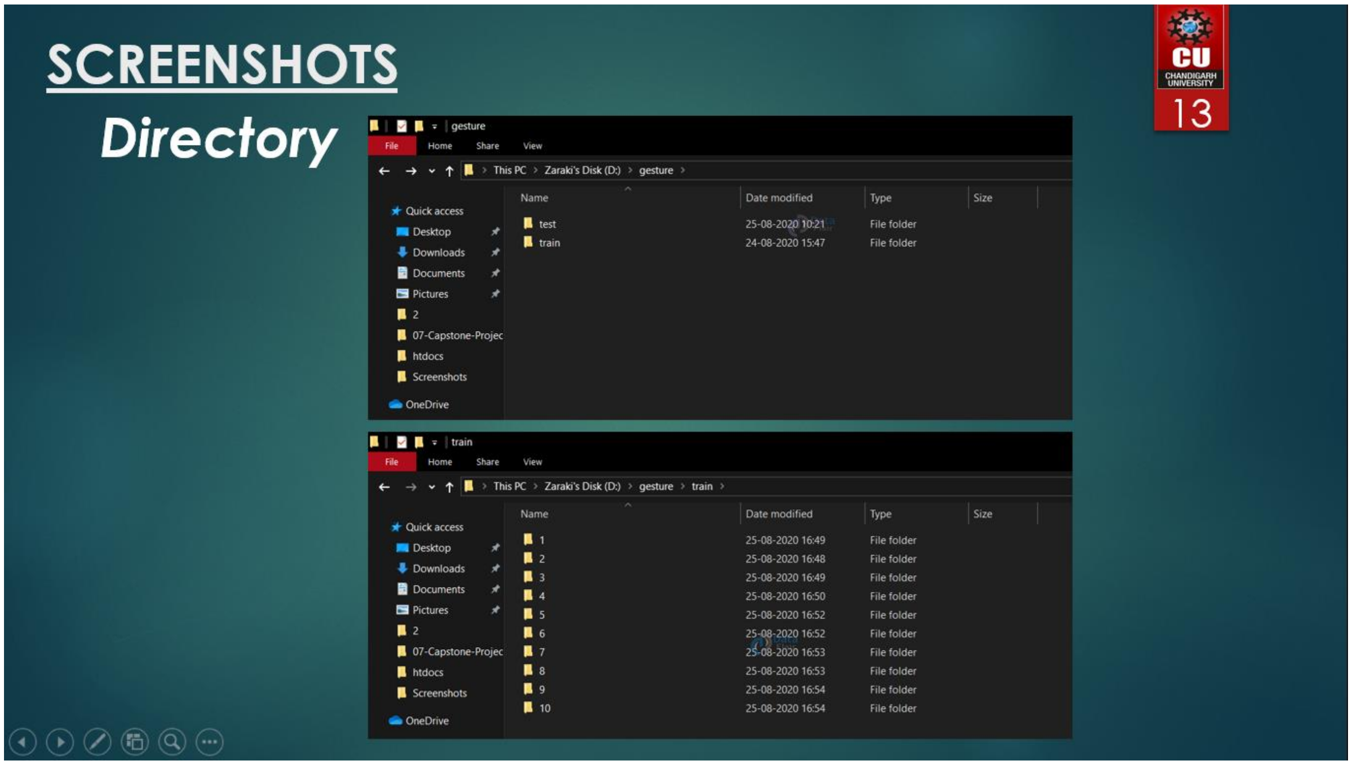




Sequence Diagram For Sign Language Recognition System



State Chart Diagram For Sign Language Recognition System



SCREENSHOTS

Directory

gesture

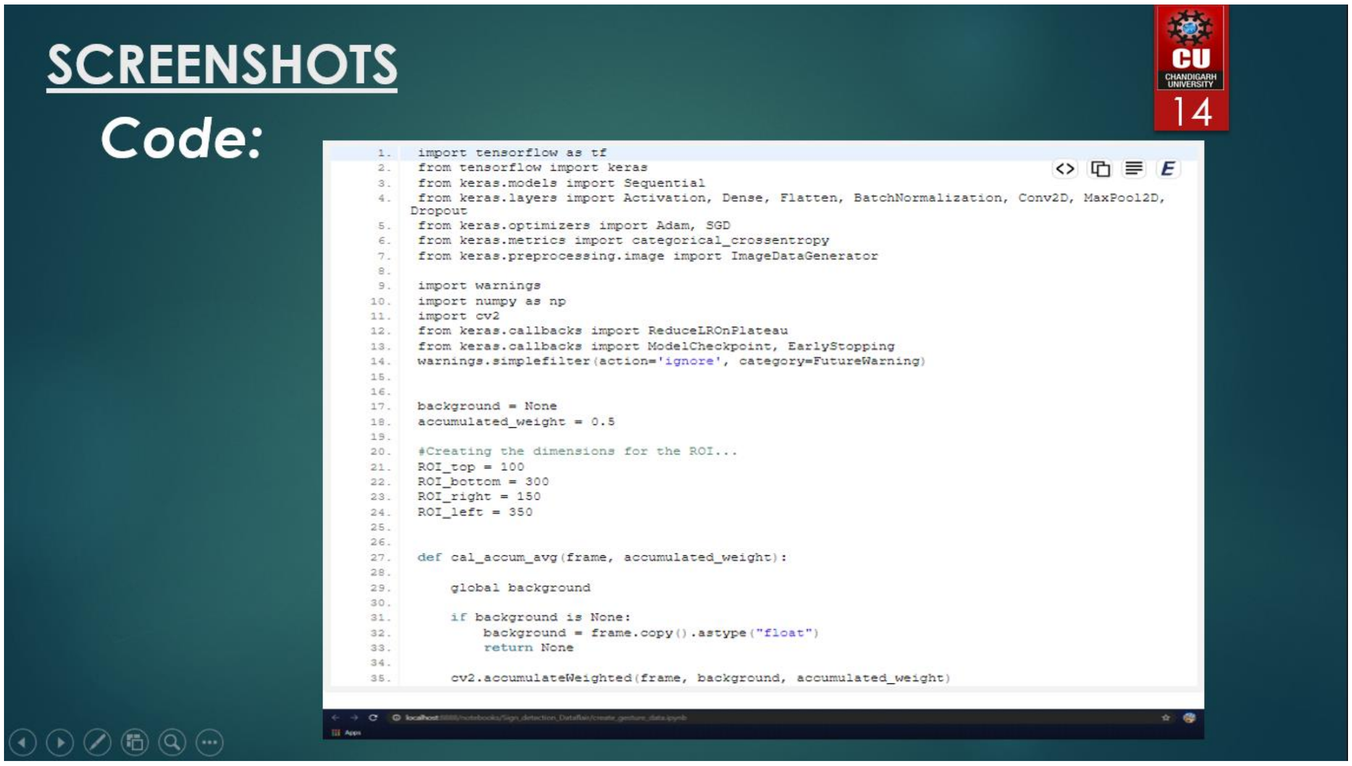
This PC > Zaraki's Disk (D:) > gesture >

Name	Date modified	Type	Size
test	25-08-2020 10:21	File folder	
train	24-08-2020 15:47	File folder	

train

This PC > Zaraki's Disk (D:) > gesture > train >

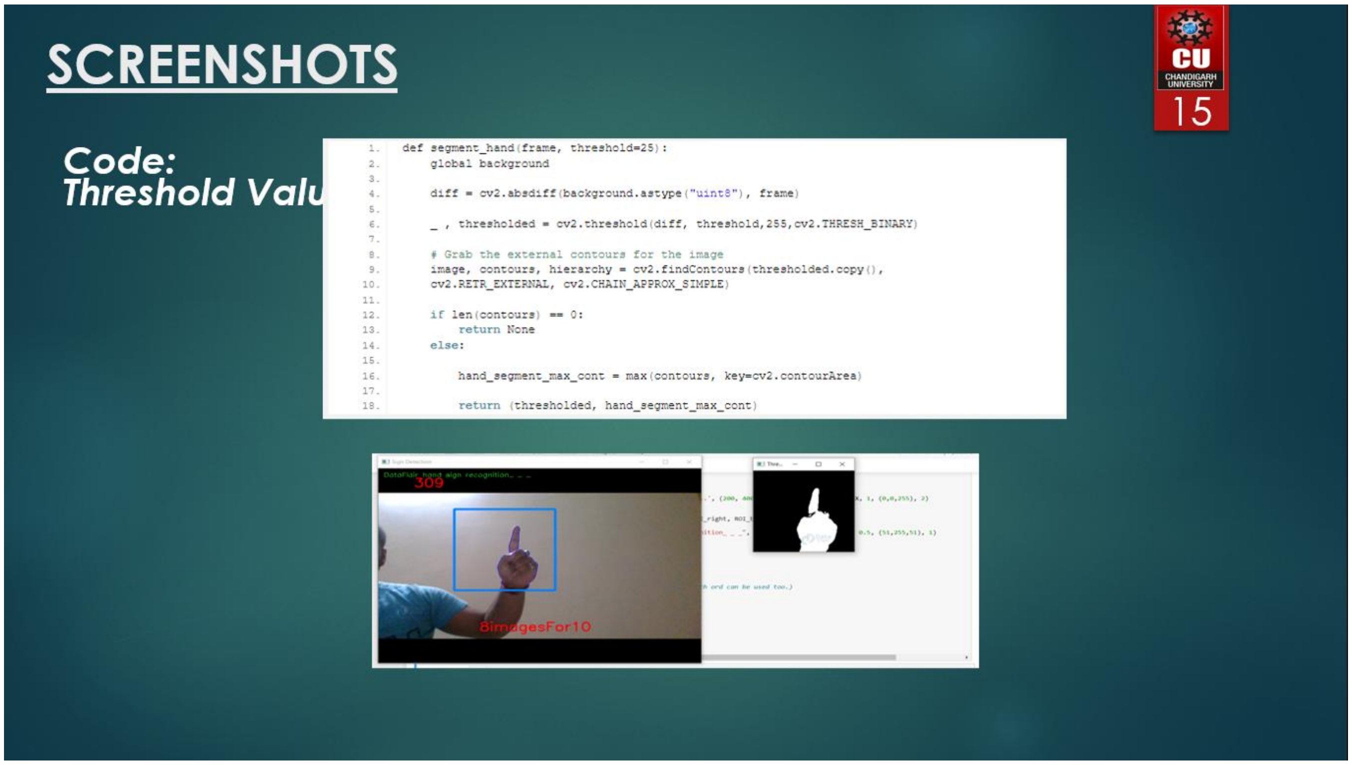
Name	Date modified	Type	Size
1	25-08-2020 16:49	File folder	
2	25-08-2020 16:48	File folder	
3	25-08-2020 16:49	File folder	
4	25-08-2020 16:50	File folder	
5	25-08-2020 16:52	File folder	
6	25-08-2020 16:52	File folder	
7	25-08-2020 16:53	File folder	
8	25-08-2020 16:53	File folder	
9	25-08-2020 16:54	File folder	
10	25-08-2020 16:54	File folder	



SCREENSHOTS

Code:

```
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)
```

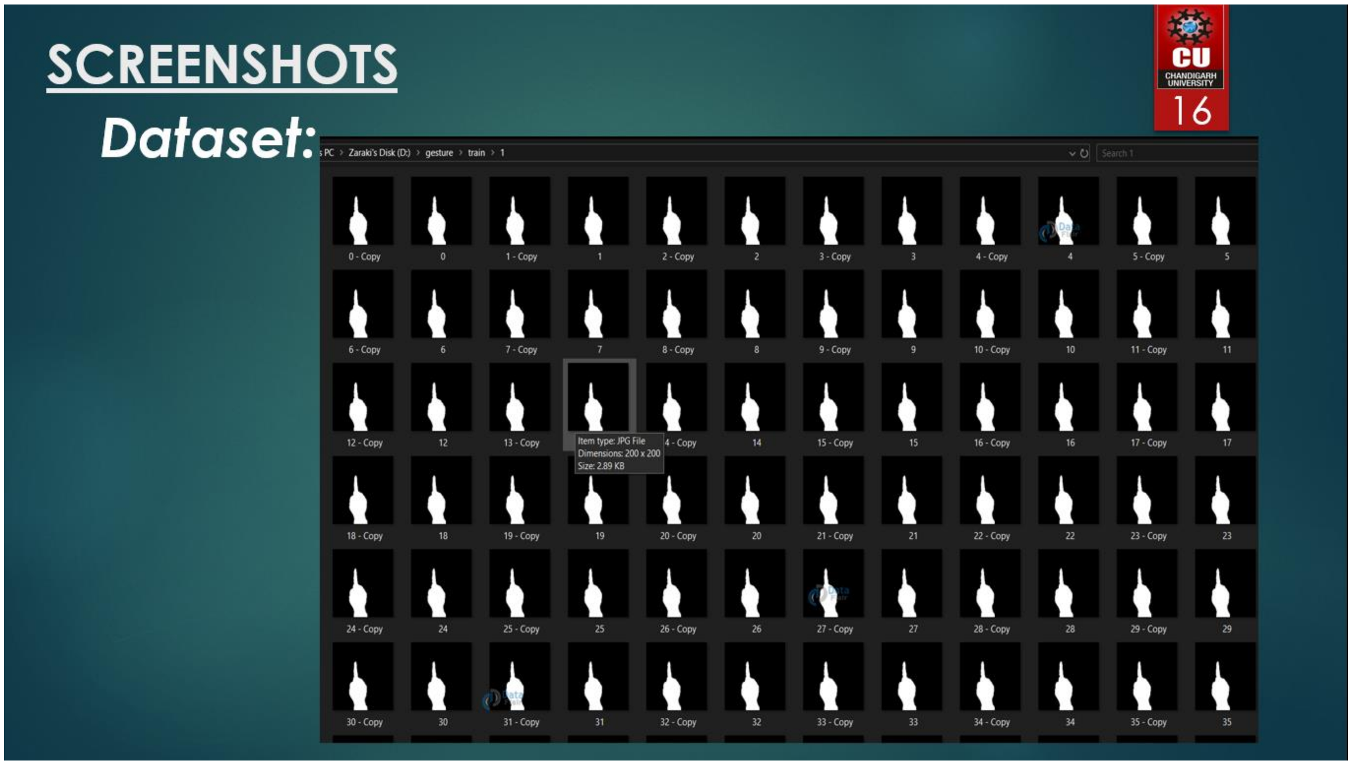


SCREENSHOTS

Code:
Threshold Valu

```
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)
```

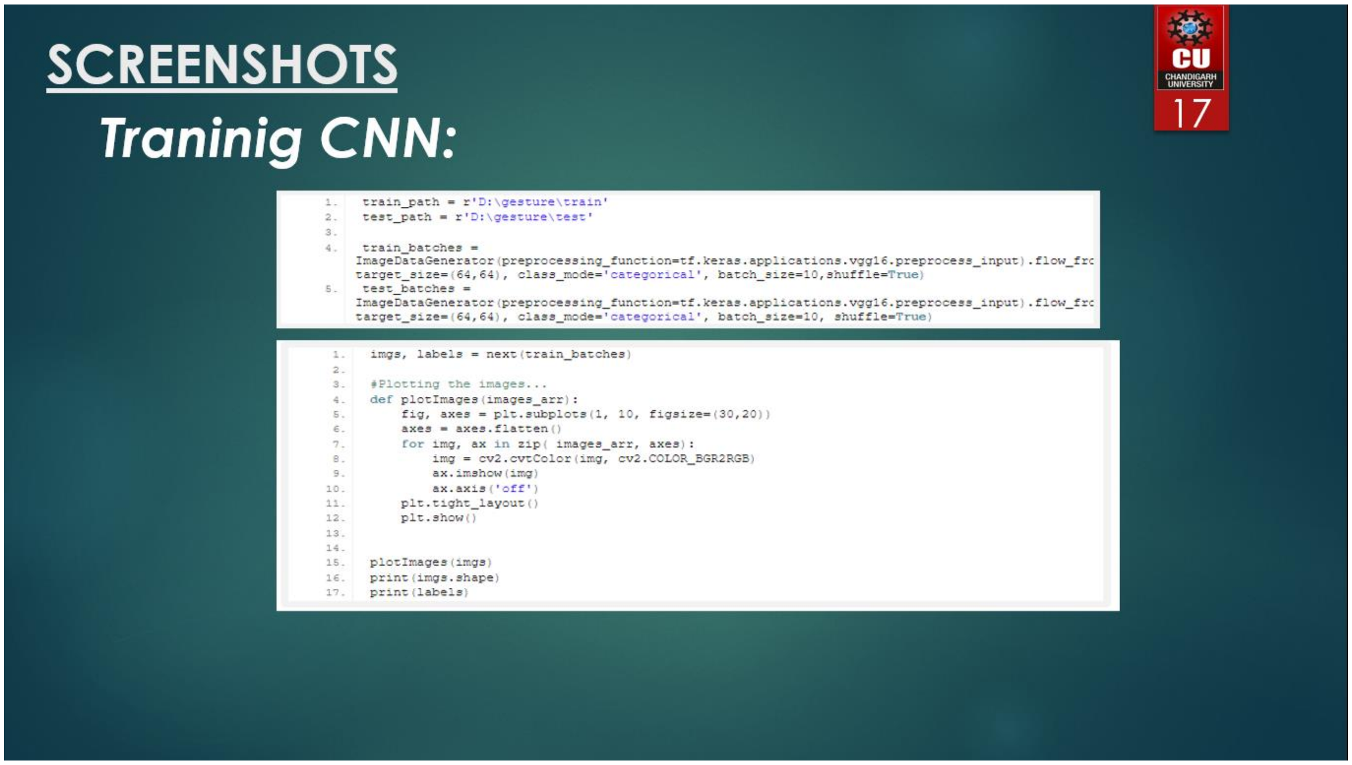
DataFlair hand sign recognition_ _ _
309
8imagesFor10



SCREENSHOTS

Dataset:

PC > Zaraki's Disk (D:) > gesture > train > 1
Search 1
0 - Copy, 0, 1 - Copy, 1, 2 - Copy, 2, 3 - Copy, 3, 4 - Copy, 4, 5 - Copy, 5
6 - Copy, 6, 7 - Copy, 7, 8 - Copy, 8, 9 - Copy, 9, 10 - Copy, 10, 11 - Copy, 11
12 - Copy, 12, 13 - Copy, 14 - Copy, 14, 15 - Copy, 15, 16 - Copy, 16, 17 - Copy, 17
Item type: JPG File
Dimensions: 200 x 200
Size: 2.89 KB
18 - Copy, 18, 19 - Copy, 19, 20 - Copy, 20, 21 - Copy, 21, 22 - Copy, 22, 23 - Copy, 23
24 - Copy, 24, 25 - Copy, 25, 26 - Copy, 26, 27 - Copy, 27, 28 - Copy, 28, 29 - Copy, 29
30 - Copy, 30, 31 - Copy, 31, 32 - Copy, 32, 33 - Copy, 33, 34 - Copy, 34, 35 - Copy, 35



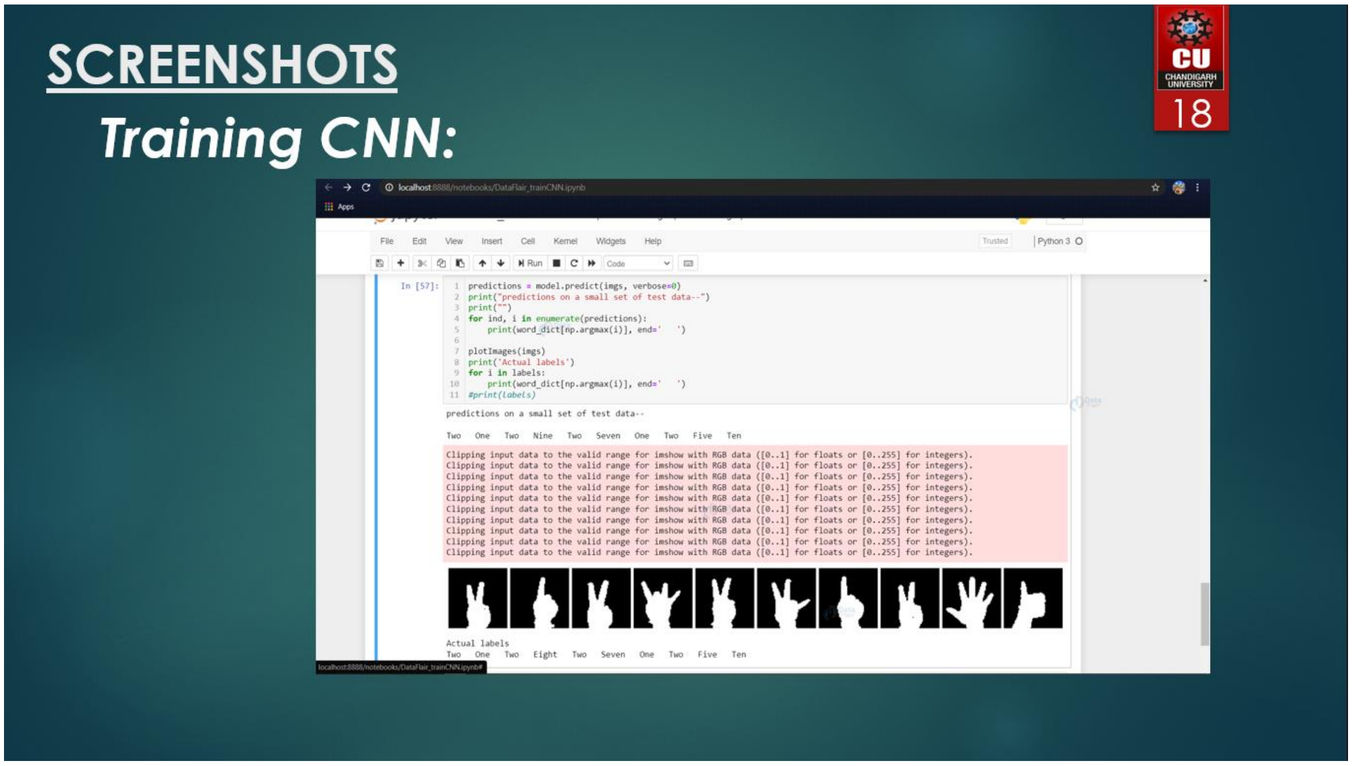
SCREENSHOTS

Traninig CNN:

17

```
train_path = r'D:\gesture\train'
test_path = r'D:\gesture\test'

train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_frc
target_size=(64,64), class_mode='categorical', batch_size=10,shuffle=True)
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_frc
target_size=(64,64), class_mode='categorical', batch_size=10, shuffle=True)
```

```
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)
```



SCREENSHOTS

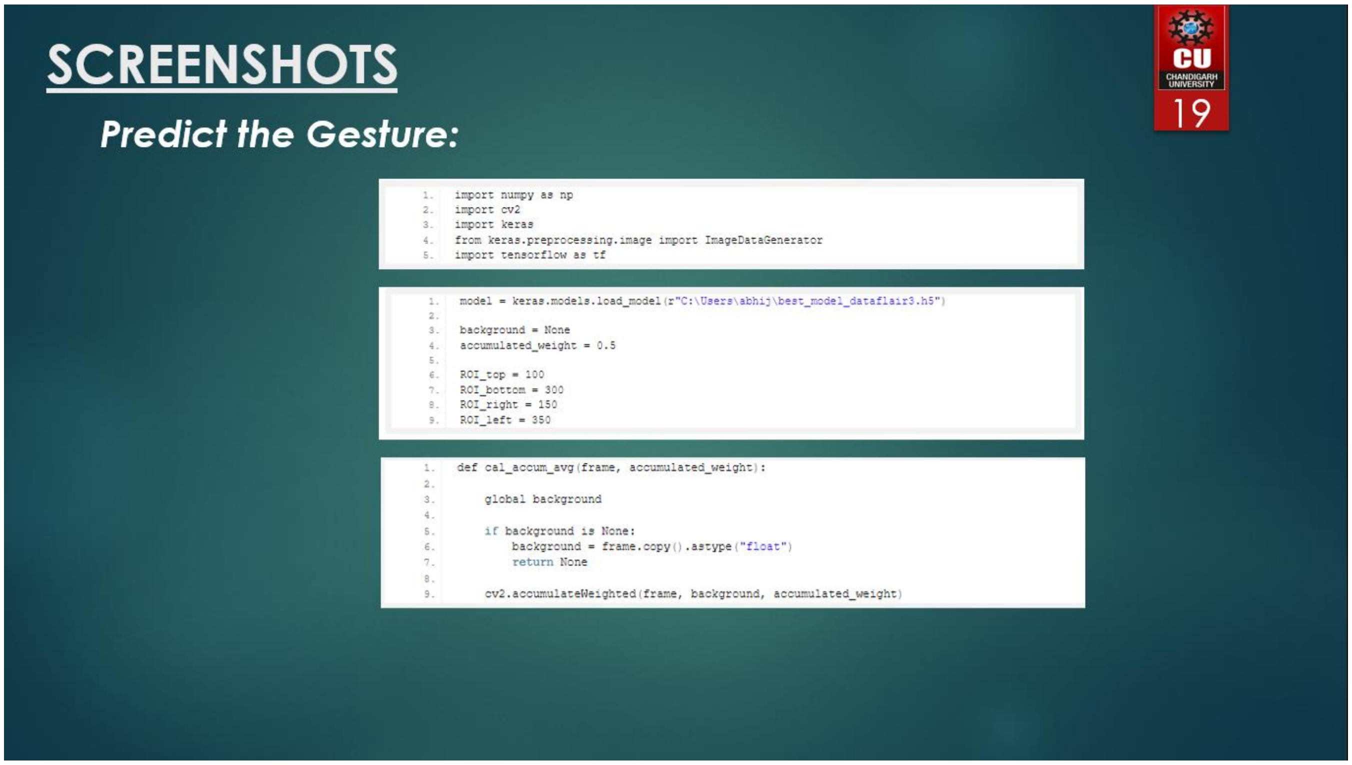
Training CNN:

18

```
In [57]:
predictions = model.predict(imgs, verbose=0)
print("predictions on a small set of test data--")
print("")
for ind, i in enumerate(predictions):
    print(word_dict[np.argmax(i)], end='   ')

plotImages(imgs)
print('Actual labels')
for i in labels:
    print(word_dict[np.argmax(i)], end='   ')
#print(labels)
```

```
predictions on a small set of test data--

Two   One   Two   Nine   Two   Seven   One   Two   Five   Ten
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
Actual labels
Two   One   Two   Eight   Two   Seven   One   Two   Five   Ten
```



SCREENSHOTS

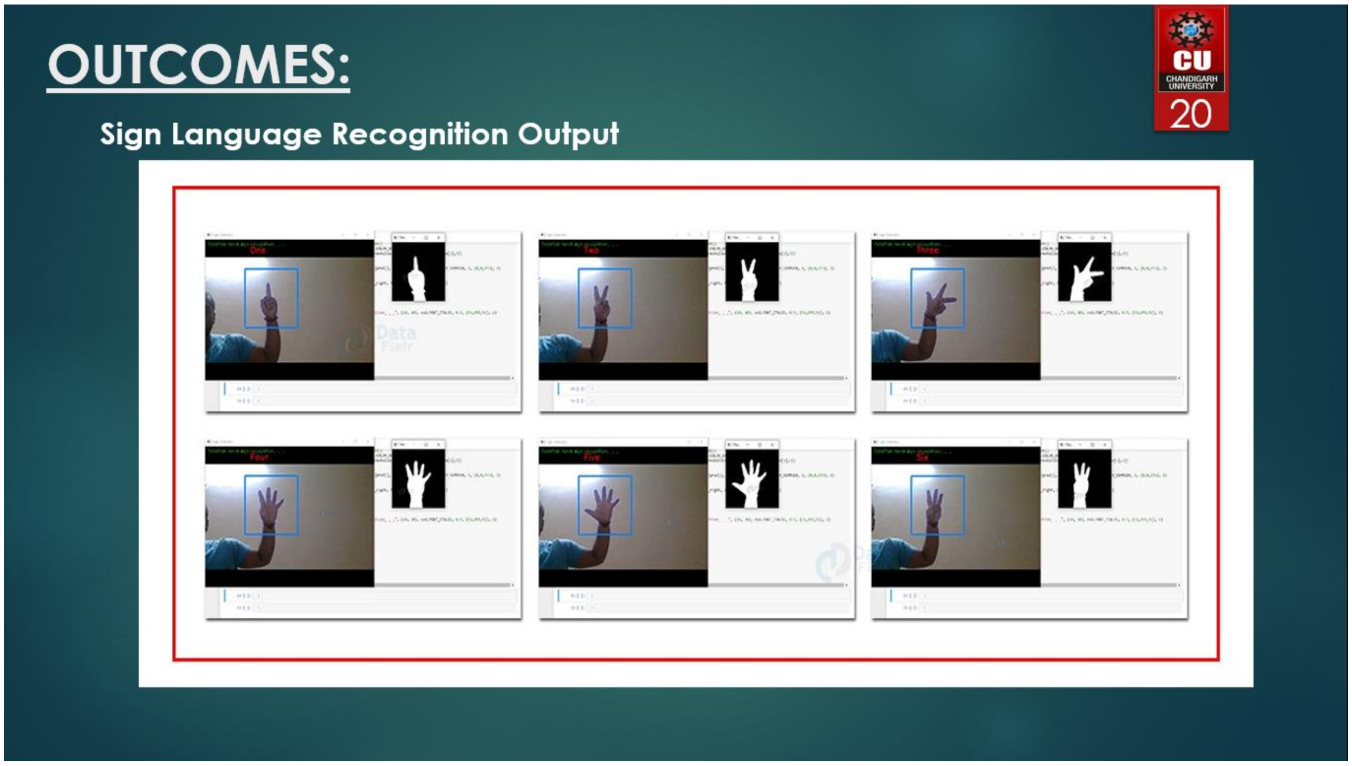
Predict the Gesture:

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

```
model = keras.models.load_model(r"C:\Users\abhij\best_model_dataflair3.h5")

background = None
accumulated_weight = 0.5

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)
```

19



OUTCOMES:

Sign Language Recognition Output

20

ATTAINMENT OF STATED OUTCOMES:

Hand gestures are a powerful way for human communication, with lots of potential applications in the area of human computer interaction. Vision-based hand gesture recognition techniques have many proven advantages compared with traditional devices. However, hand gesture recognition is a difficult problem and the current work is only a small contribution towards achieving the results needed in the field of sign language recognition. This paper presented a vision-based system able to interpret static hand gestures from the Portuguese Sign Language. Experiments with two different datasets were carried out in order to find the best hand features, among two different types, in terms of Portuguese Sign Language gesture classification. The extracted features were tested with the help of the RapidMiner tool for machine learning and data mining. That way, it was possible to identify the best SVM parameters for learning and classification. The obtained parameters were able to achieve very good results in terms of real-time gesture classification with a minimal difference (0,2%) between the two hand features. The proposed solution was tested in real time situations, were it was possible to prove that the obtained classification models were able to recognize all the trained gestures being at the same time user independent, important requirements for this type of systems. The selected hand features, in conjunction with machine learning algorithms, proved to be very efficient, allowing their application in any real-time sign language recognition systems. As future work it is intended to keep improving the system and make experiments with complete language datasets. It is also intended to test systems able to interpret dynamic sign language gestures where there is a need for reliable solutions for the problem of start and end gesture identification. As a final conclusion one can say that although there is still much to do in the area, the proposed solution is a solid foundation for the development of any vision-based sign language recognition user interface system. The sign language grammar can be easily changed and the system is configured to train the new language gestures.