HAND GESTURE RECOGNITION IN REAL TIME

(MAJOR PROJECT)

2022-2023

Submitted By:

Mr. Harshit Agarwal

Roll. No.:1018493

SECTION : B

Course: B.Tech (CSE)

Submitted To:

Mr. Amit Gupta

Table Of Content

ACKNOWLEDGEMENT

I would like to express my gratitude to my parents for their continuous support and encouragement and providing me with the opportunity to reach this far in my studies.

I’d like to express my deepest thanks to my teachers for their patience, support and encouragement throughout the completion of this project and having faith in me.

At last I’d also like to acknowledge the help of all other people who directly or indirectly helped me during this work.

Harshit Agarwal

(Roll No. 1018493)

ABSTRACT

Sign Language, it’s a visible way to communicate via hand signs, expressions(facial) , body lang and some sort of gestures. It is most important for a community of disabled like deaf people or people with autism or with down syndrome , therefore, one can say it is a primary form of communication between them. But a person without disabilities often find it difficult to understand the sign language as a result there is a communication gap between people. So in order to reduce this gap various techniques has been introduced for the detection of sign gestures. The detection technique involves capturing of images through web cam and predict the captured images through some algorithms like CNN.

LIST OF FIGURE

1. About Project

1.1 Introduction

Sign Language: a way(visible) to communicate via hand signs or gestures , expressions(facial), and body language.

- It is a visual language. It mainly consists of 3 major components:

1. Finger spelling: Spelling out of words letter by letter, and it also includes gestures that delivers the meaning of words.
2. World-level sign vocabulary: In this the entire word is recognized through video classification.
3. Non-manual features: Some other non-Manual features includes Facial expressions, tongue movement, mouth, body positions.

1.2 Objective

Objective: The motive of the project is to recognize hand gestures in real time. In this model, classification machine learning algorithms are trained using a set of image data for different hand gestures.

1.3 Motivation

The Indian census cites roughly 1.3 million people with “hearing impairment”.In contrast to that numbers from India’s National Association of the Deaf estimates that 18 million people –roughly 1 per cent of Indian population are deaf. If there is a common interface that converts the gestures or sign language to text then it can be easily understood by the other people. So research has been made for a vision based interface system where deaf people can enjoy communication without really knowing each other's language.

The aim is to develop a user friendly human computer interfaces (HCI)

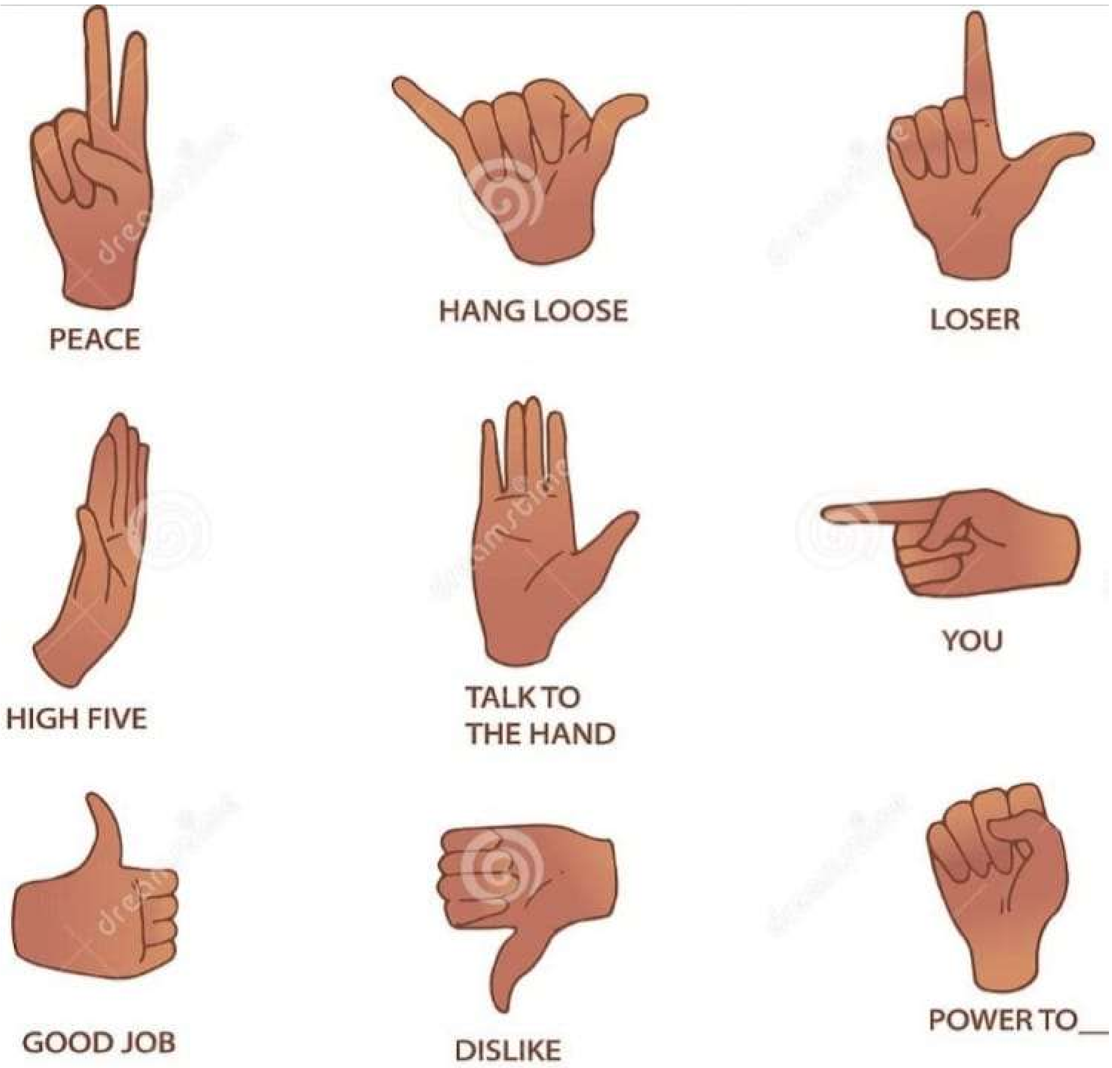
where the computer understands the human sign language. There are various sign languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language(ISL), Japanese Sign Language and work has been done on other languages all around the world. Our project hence is aimed at converting the hand gestures pf different words into text that is readable for normal people.

1.4 Literature Survey

AUTHOR	PURPOSE	DATASET	TECHNIQUE USED	ACCURACY
R Rumana , Reddygari Sandhya Rani , Mrs. R. Prema	Sign Language Recognition for the deaf and dumb	Self made Dataset	CNN	92% Accuracy

Rekha	Hand Gesture Recognition	dataset of 23 ISL static alphabet signs	fYCbCr skin model	92% Accuracy
M. Geetha and U. C. Manjusha	Hand Gesture Recognition	50 specimens of every alphabets and digits	B-Spine algorithm	90% Accuracy
Pigou	Hand Sign Language Classification	CLAP14	CNN Model	91.70% Accuracy
J Huang	Gesture Recognition	own dataset	3D CNN	94.2% Accuracy
Siming	Sign language recognition	40 common words and 10,000 sign language images	R-CNN with an embedded RPN	99% Accuracy
Dardina Tasmere	Real Time Hand Gesture Recognition in Depth Image	Own Dataset	CNN	94.61% Accuracy
J.Carriera	sign gesture recognition	ImageNet and Kinetic Dataset	RGB model	80.69% Accuracy
Hsien-I Lin† , Ming-Hsiang Hsu	human hand gesture recognition system	Own Dataset	CNN model	95.96% Accuracy
Norah Alnaim , Abdullrahman Albar Maysam Abbod	Hand Gesture Recognition Using CNN for Post-Stroke People	Self Madec Dataset	CNN model	99.89% Accuracy

1.5 Example for Gestures



2.Requirements of Project

2.1 Hardware Requirement

- Processor: 32-bit/ 64-bit, eight-core
- RAM: 4-8GB
- Hard Disk: 2GB or more

2.2 Software Requirement

- Operating System:Windows
- Language : Python
- Libraries:
  - NumPy: library for numerical calculations
  - Pandas: library for data manipulation and analysis
  - Tensorflow: library for large numerical computations without keeping deep learning in mind
  - Keras: neural network library
  - Matplotlib: for creating static, animated, and interactive visualizations
- IDE for workspace : Anaconda , Jupyter Notebook

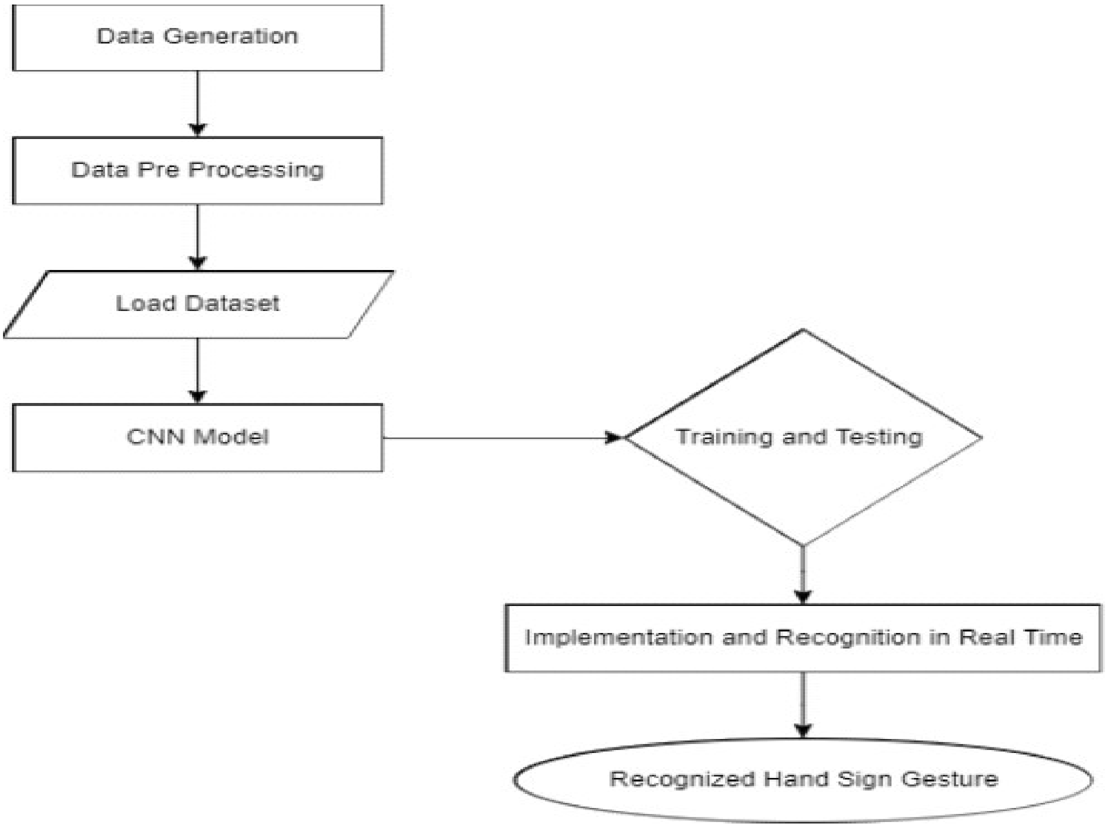
2.3 Algorithm Used

- Convolutional Neural Networks (CNN)

3.METHODOLOGY

The following diagram represents the flow of the working of the project.

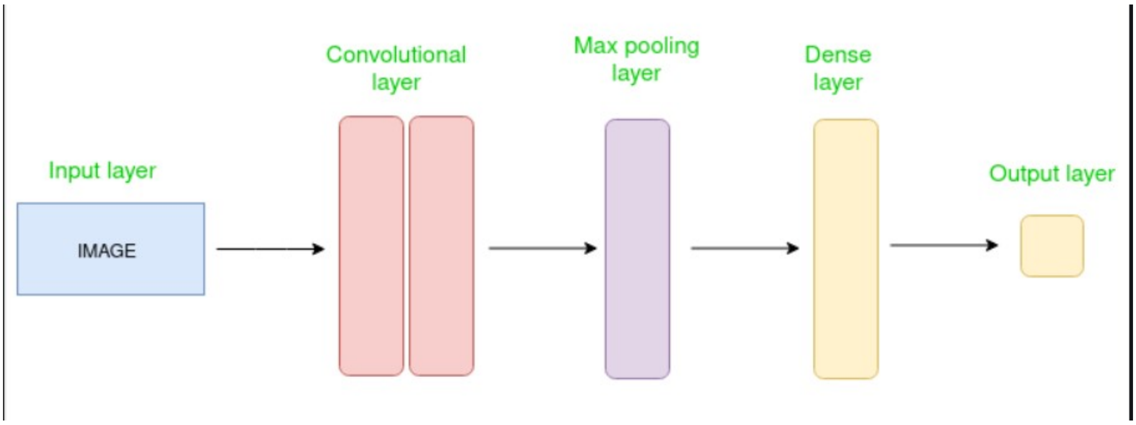
- Data Generation
- Data Processing
- Building the model
- Training and Testing the model
- Recognition and Prediction in real time



3.1 CONVOLUTIONAL NEURAL NETWORK:

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

In a Regular Neural Network there are three types of layers:
1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.



3.1.1 Convolution Layer : In convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color

3.1.2 Pooling Layer : We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type of pooling :

a) Max Pooling : In max pooling we take a window size [for example

window of size 2*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.

b) Average Pooling : In average pooling we take average of all values in a window.

3.1.3. Flattening: The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

3.1.4. Fully Connected Layer : In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

3.1.5. Output Layer : After getting values from fully connected layer, well connect them to final layer of neurons[having count equal to total number of classes], that will predict the probability of each image to be in different classes.

3.2 TENSORFLOW:

TensorFlow is a free and open-source software library for data flow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Features: TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs, and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB,R, Scala, Rust, OCaml, and Crystal."New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as Deep Dream.

3.3 OPENCV:

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition

- Automated inspection and surveillance

● number of people – count (foot traffic in a mall, etc)

● Vehicle counting on highways along with their speeds

● Interactive art installations

● Street view image stitching

● Video/image search and retrieval

● Robot and driver-less car navigation and control

● object recognition

● Medical image analysis

3.4 KERAS:

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made userfriendly, extensible, and modular for facilitating faster experimentation with 15deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

3.5 NUMPY:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

4. Dataset

Self made Dataset for hand gestures of different types containing 10 classes with 2000-2100 images per class.

5. PROJECT DESIGN

In the recent years there has been tremendous research done on the handgesture recognition. With the help of literature survey done we realized thebasic steps in hand gesture recognition are :-

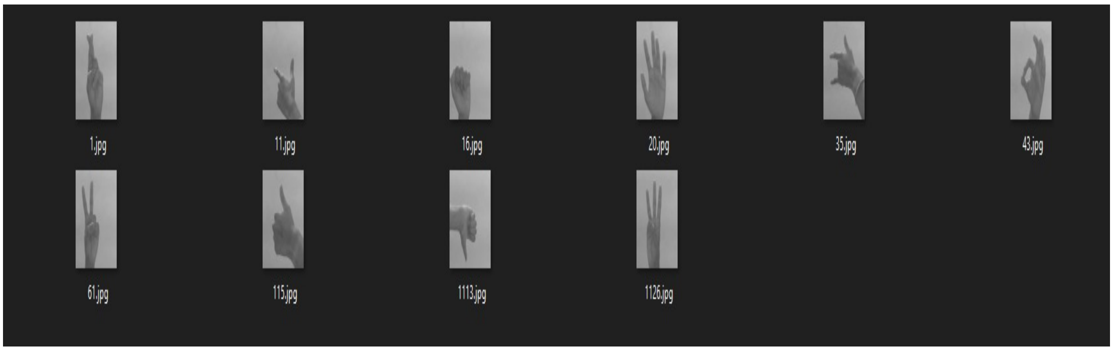
- Data generation
- Data preprocessing
- Building the model
- Training and Testing
- Gesture Recognition in real time

5.1 DATA GENERATION

In vision based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware.

For data generation 10 different classes were defined

- labels=['ThumbsUp' , 'Hello' , 'ThumbsDown' ,'ILoveYou' , 'GoodLuck' ,'Ok','Yes' , 'Peace' , 'Water' , 'Smile']
- Around 2000 images per label were clicked using opencv



5.2 DATA PREPROCESSING:

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

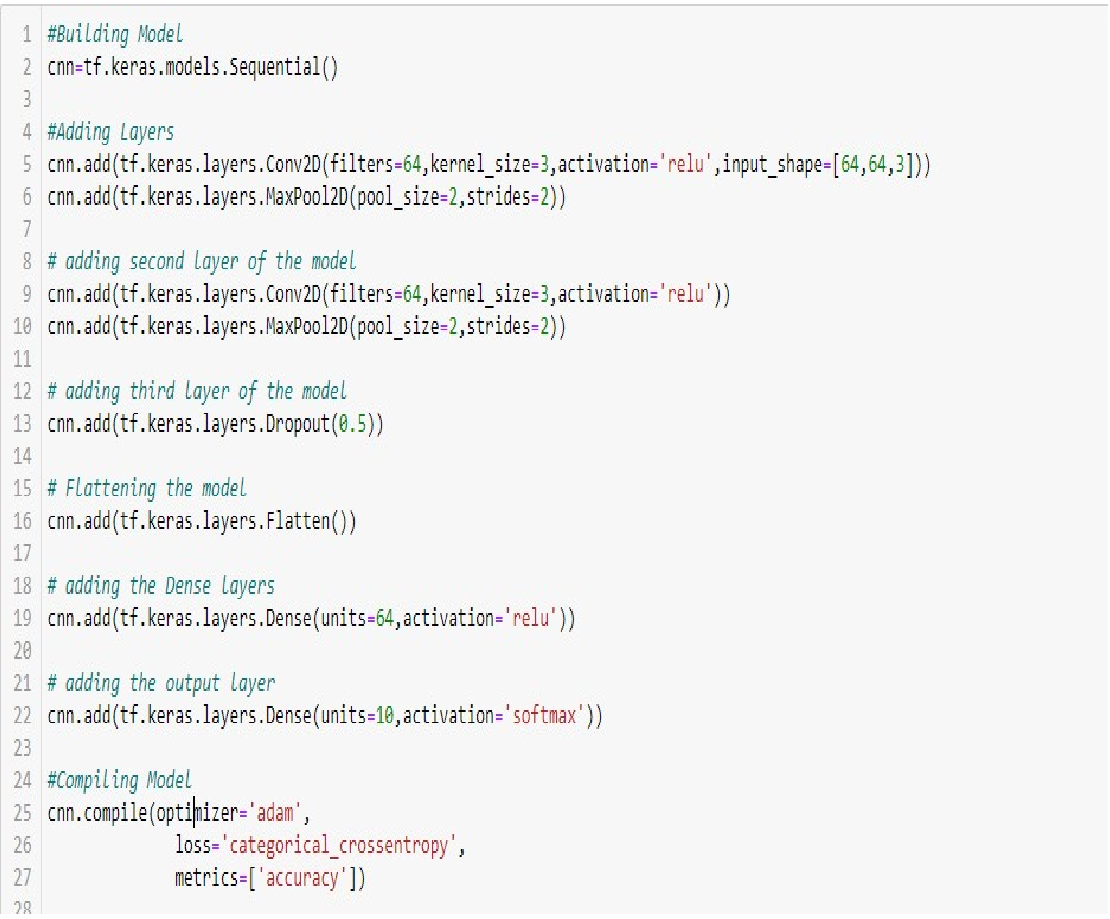
In this project the image clicked in the previous step is modified and processed accordingly . The steps followed are :

1. Rescaling
2. Height and Width shift
3. Resize Image
4. Zooming Images

5.3 BUILDING MODEL

A CNN Model is defined for the classification of different hand sign gestures .

Defined Model:



```
#Building Model
cnn=tf.keras.models.Sequential()

#Adding Layers
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu',input_shape=[64,64,3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

# adding second layer of the model
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

# adding third layer of the model
cnn.add(tf.keras.layers.Dropout(0.5))

# Flattening the model
cnn.add(tf.keras.layers.Flatten())

# adding the Dense layers
cnn.add(tf.keras.layers.Dense(units=64,activation='relu'))

# adding the output layer
cnn.add(tf.keras.layers.Dense(units=10,activation='softmax'))

#Compiling Model
cnn.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])
```

5.4 TRAINING:

The defined CNN- Model is trained on the self generated dataset.

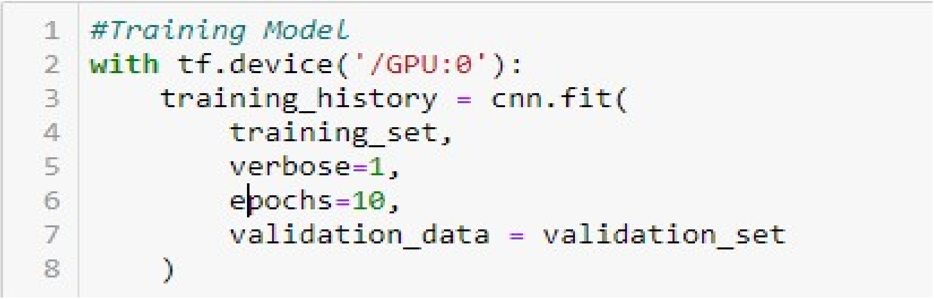
The dataset contains around 2500 images per class.

Total Classes = 10 classes

Total Images = 24618 images

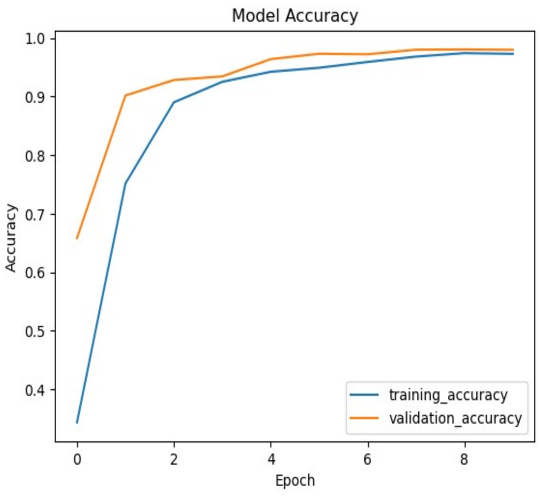
Train Set = 20274 images

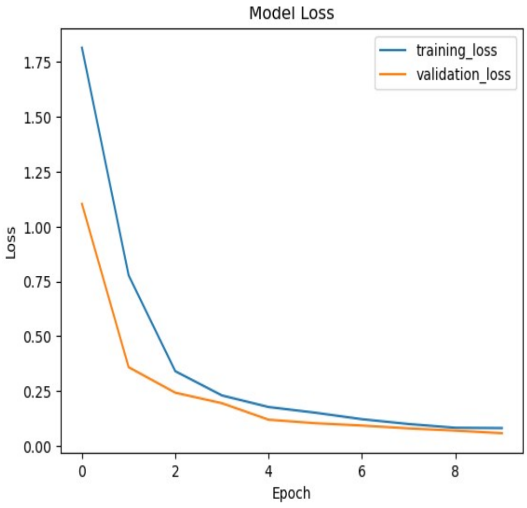
Validation Set = 4344 images



```
#Training Model
with tf.device('/GPU:0'):
    training_history = cnn.fit(
        training_set,
        verbose=1,
        epochs=10,
        validation_data = validation_set
    )
```

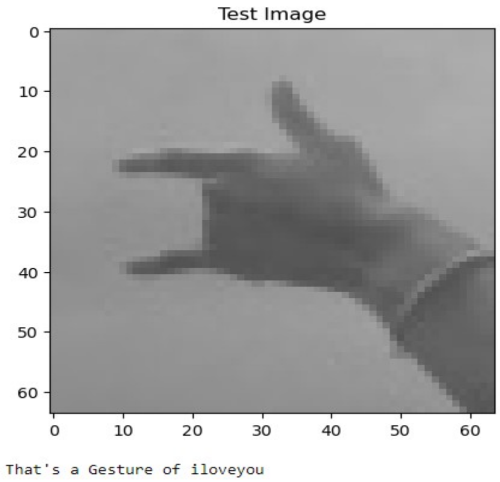
5.5 ACCURACY - LOSS GRAPH



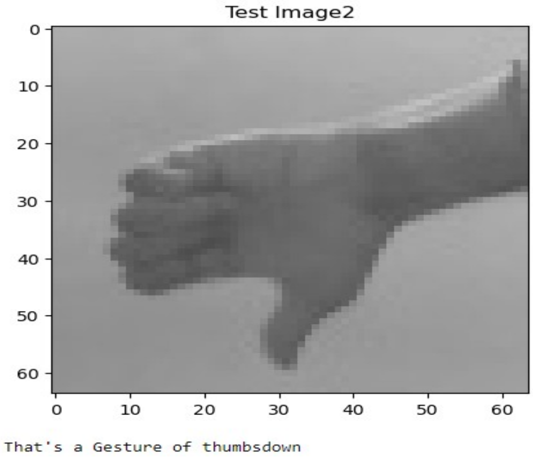


5.6 TESTING

The trained model is successfully able to predict different hand gestures.



```
That's a Gesture of iloveyou
```

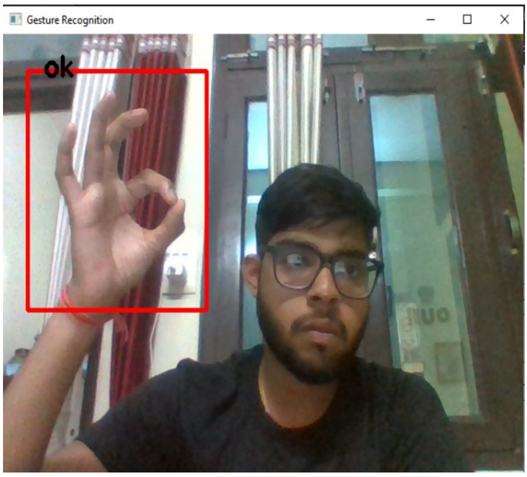


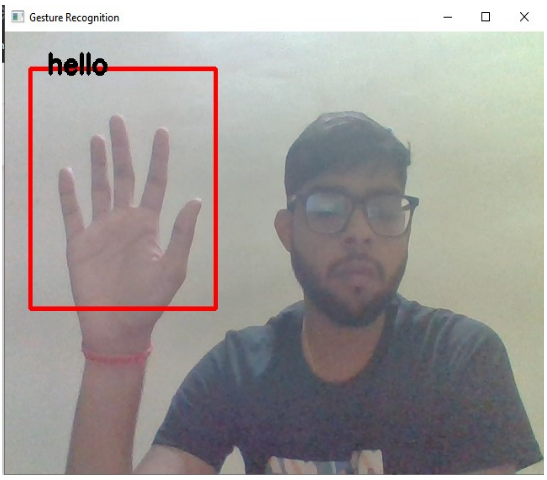
```
That's a Gesture of thumbsdown
```

5.7 RECOGNITION IN REAL TIME

OpenCV is used for recognition of different hand sign gestures in real time.

The code is used to print the identified hand sign gesture in real time.





6.Applications

- Sign Language Recognition:
  As the name suggests , it can predict the sign language , gestures etc.
- Virtual Controllers:
  Gesture can be used as an alternative control mechanism where act of acquiring a physical controller could takes too much time . for example: Controlling cars.
- Remote Control:
  Through the use of gesture recognition , remote control with the wave of a hand of various devices is possible.
- Socially assistive robotics:
  By using proper tools robot can be made for assistance for people with verbal disability. That can convert sign language to text and speech

7. Future Work

- The model can be further trained with variation in dataset.
- Text to speech feature can be added for converting the recognized hand signs to speech in real time .
- Multiple classes can be added later.

8.APPENDIX

8.1 Code for DataGeneration.py

```
import os
import cv2
def makedir(directory):
    if not os.path.exists(directory):
        os.makedirs(directory)
        return None
    else:
        pass
cap=cv2.VideoCapture(0)
i=0
image_count=0
while i<31:
    ret,frame=cap.read()
    frame=cv2.flip(frame,1)
    roi=frame[100:400,320:620]
    cv2.imshow('roi',roi)
    roi=cv2.cvtColor(roi,cv2.COLOR_BGR2GRAY)
    roi=cv2.resize(roi,(64,64),interpolation=cv2.INTER_AREA)
    cv2.imshow('roi scaled and gray',roi)
    copy=frame.copy()
    cv2.rectangle(copy,(320,100),(620,400),(255,0,0),5)
    if i==0:
        image_count=0
        cv2.putText(copy,"Hit Enter to record goodluck
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
    if i==1:
        image_count+=1
        cv2.putText(copy,"Recording 1st Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_one='./handgestures/train/goodluck/'
        makedir(gesture_one)
        cv2.imwrite(gesture_one + str(image_count+1100)+".jpg",roi)
    if i==2:
        image_count+=1
        cv2.putText(copy,"Recording 1st Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_one='./handgestures/validation/goodluck/'
        makedir(gesture_one)
        cv2.imwrite(gesture_one + str(image_count+1100)+".jpg",roi)
    if i==3:
        cv2.putText(copy,"Hit Enter to record hello
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
    if i==4:
        image_count+=1
        cv2.putText(copy,"Recording 2nd Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_two='./handgestures/train/hello/'
        makedir(gesture_two)
        cv2.imwrite(gesture_two + str(image_count+1100)+".jpg",roi)
    if i==5:
        image_count+=1
        cv2.putText(copy,"Recording 2nd Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_two='./handgestures/validation/hello/'

        makedir(gesture_two)

        cv2.imwrite(gesture_two + str(image_count+1100)+".jpg",roi)

    if i==6:

        cv2.putText(copy,"Hit Enter to record iloveyou
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

    if i==7:

        image_count+=1

        cv2.putText(copy,"Recording 3nd Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_three='./handgestures/train/iloveyou/'

        makedir(gesture_three)

        cv2.imwrite(gesture_three + str(image_count+1100)+".jpg",roi)

    if i==8:

        image_count+=1

        cv2.putText(copy,"Recording 3nd Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_three='./handgestures/validation/iloveyou/'

        makedir(gesture_three)

        cv2.imwrite(gesture_three + str(image_count+1100)+".jpg",roi)

    if i==9:

        cv2.putText(copy,"Hit Enter to record ok
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
```

```
    if i==10:
        image_count+=1
        cv2.putText(copy,"Recording 4th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_four='./handgestures/train/ok/'
        makedir(gesture_four)
        cv2.imwrite(gesture_four + str(image_count+1100)+".jpg",roi)
    if i==11:
        image_count+=1
        cv2.putText(copy,"Recording 4th Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_four='./handgestures/validation/ok/'
        makedir(gesture_four)
        cv2.imwrite(gesture_four  +str(image_count+1100)+".jpg",roi)
    if i==12:
        cv2.putText(copy,"Hit Enter to record peace
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
    if i==13:
        image_count+=1
        cv2.putText(copy,"Recording 5th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_five='./handgestures/train/peace/'
        makedir(gesture_five)
        cv2.imwrite(gesture_five + str(image_count+1100)+".jpg",roi)
    if i==14:
        image_count+=1

        cv2.putText(copy,"Recording 5th Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_five='./handgestures/validation/peace/'

        makedir(gesture_five)

        cv2.imwrite(gesture_five + str(image_count+1100)+".jpg",roi)

    if i==15:

        cv2.putText(copy,"Hit Enter to record smile
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

    if i==16:

        image_count+=1

        cv2.putText(copy,"Recording 6th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_six='./handgestures/train/smile/'

        makedir(gesture_six)

        cv2.imwrite(gesture_six + str(image_count+1100)+".jpg",roi)

    if i==17:

        image_count+=1

        cv2.putText(copy,"Recording 6th Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_six='./handgestures/validation/smile/'

        makedir(gesture_six)

        cv2.imwrite(gesture_six + str(image_count+1100)+".jpg",roi)

    if i==18:
        cv2.putText(copy,"Hit Enter to record thumbsdown
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

    if i==19:

        image_count+=1

        cv2.putText(copy,"Recording 7th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_seven='./handgestures/train/thumbsdown/'

        makedir(gesture_seven)

        cv2.imwrite(gesture_seven + str(image_count+1100)+".jpg",roi)

    if i==20:

        image_count+=1

        cv2.putText(copy,"Recording 7st Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_seven='./handgestures/validation/thumbsdown/'

        makedir(gesture_seven)

        cv2.imwrite(gesture_seven + str(image_count+1100)+".jpg",roi)

    if i==21:

        cv2.putText(copy,"Hit Enter to record thumbsup
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

    if i==22:

        image_count+=1

        cv2.putText(copy,"Recording 8th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_eight='./handgestures/train/thumbsup/'

        makedir(gesture_eight)
        cv2.imwrite(gesture_eight + str(image_count+1100)+".jpg",roi)

    if i==23:

        image_count+=1

        cv2.putText(copy,"Recording 1st Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_eight='./handgestures/validation/thumbsup/'

        makedir(gesture_eight)

        cv2.imwrite(gesture_eight + str(image_count+1100)+".jpg",roi)

    if i==24:

        cv2.putText(copy,"Hit Enter to record water
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

    if i==25:

        image_count+=1

        cv2.putText(copy,"Recording 9th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_nine='./handgestures/train/water/'

        makedir(gesture_nine)

        cv2.imwrite(gesture_nine + str(image_count+1100)+".jpg",roi)

    if i==26:

        image_count+=1

        cv2.putText(copy,"Recording 9th Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)


cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)

        gesture_nine='./handgestures/validation/water/'

        makedir(gesture_nine)

        cv2.imwrite(gesture_nine + str(image_count+1100)+".jpg",roi)
    if i==27:
        cv2.putText(copy,"Hit Enter to record yes
gesture",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
    if i==28:
        image_count+=1
        cv2.putText(copy,"Recording 10th Gesture -
Train",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_ten='./handgestures/train/yes/'
        makedir(gesture_ten)
        cv2.imwrite(gesture_ten + str(image_count+1100)+".jpg",roi)
    if i==29:
        image_count+=1
        cv2.putText(copy,"Recording 1st Gesture -
Test",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)

cv2.putText(copy,str(image_count),(400,400),cv2.FONT_HERSHEY_COMPLEX,1,(
0,255,0),1)
        gesture_ten='./handgestures/validation/yes/'
        makedir(gesture_ten)
        cv2.imwrite(gesture_ten + str(image_count+1100)+".jpg",roi)
    if i==30:
        cv2.putText(copy,"Hit Enter to
Exit",(100,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),1)
    cv2.imshow('frame',copy)


    if cv2.waitKey(1) ==13:
        image_count=0
        i+=1
cap.release()
cv2.destroyAllWindows()
```

8.2 Code for HandGestureRecognitionModel.py

```
#IMPORTING LIBRARIES

try:

    import numpy as np

    import tensorflow as tf

    import os

    import matplotlib.pyplot as plt

    print("----Libraries Loaded----")

except:

    print("----Libraries Not Loaded----")

    #CHANGING DIRECTORY


    os.chdir(r'C:\Users\HP\OneDrive\Desktop\Projects\HandGestureInRealTime')

    print("----Folder Loaded----")

    os.listdir()

    #Data Preprocessing And Dataset Loading


    from tensorflow.keras.preprocessing.image import ImageDataGenerator

    training_path=
'C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/Dataset/train'

    validation_path=
'C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/Dataset/validation
'

    train_datagen =ImageDataGenerator(

            rescale=1.0 / 255.0,

            rotation_range=0,

            zoom_range = 0.15,

            width_shift_range=0.10,

            height_shift_range=0.10,

            horizontal_flip=False,

            vertical_flip=False
    )
    val_datagen = ImageDataGenerator(rescale=1.0 / 255.0,
            rotation_range=0,
            zoom_range = 0.15,
            width_shift_range=0.10,
            height_shift_range=0.10,
            horizontal_flip=False,
            vertical_flip=False)
    training_set = train_datagen.flow_from_directory(training_path,
        class_mode='categorical',
        batch_size=32,
        target_size=(64,64)
    )
    validation_set = val_datagen.flow_from_directory(validation_path,
        class_mode='categorical',
        batch_size=32,
        target_size=(64,64)
    )
    print("Dataset Loaded")
    #Building Model
    cnn=tf.keras.models.Sequential()

    #Adding Layers
    cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu',input_sh
ape=[64,64,3]))
    cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

    # adding second layer of the model
    cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
    cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

```
# adding third layer of the model
cnn.add(tf.keras.layers.Dropout(0.5))

# Flattening the model
cnn.add(tf.keras.layers.Flatten())

# adding the Dense layers
cnn.add(tf.keras.layers.Dense(units=64,activation='relu'))

# adding the output layer
cnn.add(tf.keras.layers.Dense(units=10,activation='softmax'))

#Compiling Model
cnn.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])

#Printing Summary Of Model
cnn.summary()
#Training Model
with tf.device('/GPU:0'):
    training_history = cnn.fit(
        training_set,
        verbose=1,
        epochs=10,
        validation_data = validation_set
    )
#Saving Model
cnn.save("HandGestureRecognitionModel.h5")
```

```
print('----Model Saved----')

#Saving History Of Model

import json

with open ('training_hist.json','w') as f:

  json.dump(training_history.history,f)

#Printing Model Accuracy

print('Validation set Accuracy:
{}%'.format(training_history.history['val_accuracy'][-1]*100))

# Accuracy Visualization

plt.plot(training_history.history['accuracy'][0:220])

plt.plot(training_history.history['val_accuracy'][0:220])

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['training_accuracy', 'validation_accuracy'])

plt.show()

#Loss Visualization

plt.plot(training_history.history['loss'][0:220])

plt.plot(training_history.history['val_loss'][0:220])

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['training_loss', 'validation_loss'])

plt.show()

#Importing Libraries


import cv2

import matplotlib.pyplot as plt

import tensorflow as tf

import numpy as np
```

```
# Model Loading


model = tf.keras.models.load_model('C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/HandGestureRecognitionModel.h5')

print('----Model Loaded----')

# Printing Test image for prediction


image_path='C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/Dataset/test/35.jpg'

image_path2='C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/Dataset/test/164.jpg'

img = cv2.imread(image_path)

img2= cv2.imread(image_path2)

plt.imshow(img)

plt.title("Test Image")

plt.show()

#preprocessing of testing Image

image=tf.keras.preprocessing.image.load_img(image_path,target_size=(64,64))

input_arr =tf.keras.preprocessing.image.img_to_array(image)

input_arr=np.array([input_arr])


#Predicting Image on Trained Model

predictions=model.predict(input_arr)

print(predictions)

#Printing Test Image2

plt.imshow(img2)

plt.title("Test Image2")

plt.show()

#preprocessing of testing Image

image=tf.keras.preprocessing.image.load_img(image_path2,target_size=(64,64))
```

```
input_arr =tf.keras.preprocessing.image.img_to_array(image)
input_arr=np.array([input_arr])

#Predicting Image on Trained Model
predictions=model.predict(input_arr)
print(predictions)
#Storing Index Of Gesture
result_index= np.where(predictions[0]==max(predictions[0]))

#Printing Test Image For Prediction
plt.imshow(img2)
plt.title("Test Image2")
plt.show()
print("That's a Gesture of {}".format(gesture_classes[result_index[0][0]]))
```

8.3 Code for HandGestureRecognitionMain.py:

```
try:
    import cv2
    import numpy as np
    import tensorflow as tf
    import time
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.models import load_model
    print('-----libraries loaded-----')
except:
    print('-----libraries not loaded-----')
cnn =
load_model('C:/Users/HP/OneDrive/Desktop/Projects/HandGestureInRealTime/Hand
GestureRecognitionModel.h5')
print('----Model Loaded----')
```

```
    cap = cv2.VideoCapture(0)

    while cap.isOpened():
        ret, frame = cap.read()
        cv2.flip(frame,1)
        # Defining cordinates for roi
        x1=30
        y1=40
        x2=250
        y2=300

        # Crop the frame to roi
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0,0,255) ,3)
        roi = frame[y1:y2, x1:x2]

        #Pre_processing roi image
        roi=cv2.resize(roi,(64,64))
        img = image.img_to_array(roi)
        img = np.expand_dims(img, axis = 0)
        img /= 255

        # Prediction using trained model
        prediction = cnn.predict(img)

        # Defining Gesture Classes
        gesture_classes= {0: 'goodluck', 1: 'hello', 2: 'iloveyou', 3: 'ok', 4: 'peace', 5:
'smile', 6: 'thumbsdown', 7: 'thumbsUp', 8: 'water', 9: 'yes'}

        # Printing Prediction
        print(gesture_classes[np.argmax(prediction)])
        # Getting Gesture Name
        gesture=gesture_classes[np.argmax(prediction)]
        cv2.putText(frame, gesture, (50, 45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 3)
        cv2.imshow("Gesture Recognition", frame)
        if cv2.waitKey(2) & 0xFF==ord('q'):
            break

    # Release resources
    cap.release()
    cv2.destroyAllWindows()
```

9.REFERENCES

1. https://www.youtube.com/watch?v=pDXdlXlaCco
2. https://www.youtube.com/watch?v=YjnGou4skGU&t=560s
3. https://www.youtube.com/watch?v=vQZ4IvB07ec&t=487s
4. https://www.youtube.com/watch?v=6Bn0PY_ouBY&t=1s
5. https://www.kaggle.com/grassknoted/asl-alphabet
6. https://www.youtube.com/watch?v=WQeoO7MI0Bs
7. https://www.youtube.com/watch?v=doDUihpj6ro