A

Project Seminar Report

on

# ACTION DETECTION FOR SIGN LANGUAGE

Submitted for partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**T.HARITHA VANI (2451-19-733-313)**
**G.RAJITHA(2451-19-733-316)**
**S.KAMALA(2451-19-733-318)**

Under the guidance of

**K.V.SRILAKSHMI ASHARANI**
Assistant Professor
Department of CSE


**MATURI VENKATA SUBBA RAO (MVSR) ENGINEERING COLLEGE**
Department of Computer Science and Engineering
(Affiliated to Osmania University & Recognized by AICTE)
Nadergul, Balapur Mandal, Hyderabad – 501 510
Academic Year: 2022-23

## CERTIFICATE

*This is to certify that this Project Seminar Report entitled "Action Detection For Sign Language" is a bonafide work carried out by* ***Ms.T.Haritha vani (2451-19-733-313), Ms.G.Rajitha (2451-19-733-316), Ms.S.Kamala (2451-19-733-318)*** *in partial fulfillment of the requirements for the award of degree of* ***Bachelor of Engineering*** *in* ***Computer Science And Engineering*** *from* ***Maturi Venkata Subba Rao (MVSR) Engineering College,*** *affiliated to OSMANIA UNIVERSITY, Hyderabad, during the Academic Year 2022-23 under our guidance and supervision.*

*The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.*

K.V. Srilakshmi

**Internal Guide**
Mrs. K.V.Srilakshmi asharani
Assistant Professor
Department of CSE
MVSREC.

**Head of the Department**
Prof J.Prasanna Kumar
Professor & Head of
Department of CSE
MVSREC.

## DECLARATION

This is to certify that the work reported in the present project entitled "**Action Detection For Sign Language**" is a record of bonafide work done by us in the Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University during the Academic Year 2022-23. The reports are based on theproject work done entirely by us and not copied from any other source. The results embodied in this project report have not been submitted to any other University or Institute for the awardof any degree or diploma.

| | | |
|---|---|---|
| T.Haritha vani | G.Rajitha | S.Kamala |
| 2451-19-733-313 | 2451-19-733-316 | 2451-19-733-318 |

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude and indebtedness to my project coordinator / guide **K.V.Srilakshimi Asharani** for her valuable suggestions and interest throughout the course of this project.

We are also thankful to our principal **Dr. G. Kanaka Durga** and **Prof J. Prasanna Kumar**, Head, Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Hyderabad for providing supporting infrastructure, environment and ambiance for completing this project successfully as a part of our B.E.(CSE) Degree. We would like to thank our project coordinator **Sathya Prakash Racharla** for theirconstant monitoring, guidance, and support.

We convey our heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, We would like to take this opportunity to thank my family for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

T.Haritha vani (2451-19-733-313)
G.Rajitha(2451-19-733-316)
S.Kamala(2451-19-733-318)

**VISION**

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societal problems.

**MISSION**

- To make learning process exciting, stimulating and interesting.
- To impart adequate fundamental knowledge and soft skills to students.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity in students.
- To develop economically feasible and socially acceptable software.

**PEOs:**

**PEO-1:** Achieve recognition through demonstration of technical competence for successful execution of software projects to meet customer business objectives..
**PEO-2:** Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.
**PEO-3:** Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

## PROGRAM OUTCOMES (POs)

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialisation for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions**.**

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong learning:** Recognise the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

## PROGRAM SPECIFIC OUTCOMES (PSOs)

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinary domains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs through a holistic professional development leading to pioneering careers or entrepreneurship.

## COURSE OBJECTIVES AND OUTCOMES

**COURSE: B.E (CSE) SEMESTER: VIII Acad. Year: 2022-23**
**COURSE NAME: Project Work-II**
**COURSE CODE: PW 861 CS**

**Course Objectives:**

- To enhance practical and professional skills.
- To familiarize tools and techniques of systematic literature survey and documentation
- To expose the students to industry practices and team work.
- To encourage students to work with innovative and entrepreneurial ideas

**Course Outcomes:**

| **Course code PW 861 CS** | **Statement** Student will be able to |
|---|---|
| CO1 | Demonstrate the ability to synthesize and apply the knowledge and skills acquired in the academic program to real-world problems. |
| CO2 | Evaluate different solutions based on economic and technical feasibility |
| CO3 | Effectively plan a project and confidently perform all aspects of project management |
| CO4 | Demonstrate effective written and oral communication skills |
| CO5 | Understand tools and techniques ,apply for problem statements and prepare reports |

# ABSTRACT

Inability to speak is frequently regarded as a disability. Persons with this impairment can utilise a variety of techniques of communication to interact with others. Sign language is one of the methods available to people with disabilities for communication. It is imperative to create certain sign language applications for the deaf and dumb so that they can easily communicate with individuals who cannot understand sign language. The major goal of our project is to start closing the communication gap between hearing individuals and sign language users who are deaf and dumb. The main goal of our research is to create a vision-based system that can recognise sign language gestures in action or video sequences. The strategy for the sign language gestures was employed as: Video sequences contain temporal and spatial properties. We have trained both temporal and spatial features using models. The LSTM model in the recurrent neural network was used to train the model on spatial data from the video series. Frames from video clips were used to teach it. With temporal data, we trained the model using LSTM (which can process not only single data points i.e images but also entire sequences of data i.e speech or video). Each video's single frame predictions were made using a trained model, which produced a series of forecasts

T.Haritha vani (2451-19-733-313)
G.Rajitha (2451-19-733-316)
S.Kamala (2451-19-733-318)

# Index

## Table of Figures

# CHAPTER-1

# 1.INTRODUCTION

A process by which any kind of information is being exchanged, between two individuals or a set of people, know as communication. Communication plays a vital role in expressing one's emotions, attitude or ideas. It is very important and advantageous. But communication presents itself as an obstacle when it comes to people with speaking and hearing disabilities. To overcome the obstacle, Sign Language is used.

Sign Language is nothing but a means of communicating bodily, especially using hands, arms or head to express an idea or a piece of information. The sign symbols or sign gestures used in the sign language are already organised in a particular way linguistically. It's simply a non- verbal language which provides its advantage to people with speaking and hearing disabilities for communication. Different countries use different sign languages in the world. They are not the same. So, sign language cannot be declared as a universal language.

When dived into sign language further, we observe that sign language is a visual language and it consists of three main components. They are:

1) Finger spelling.
2) Word-level sign vocabulary.
3) Non-manual

## 1.1 Problem Statement

Here, our goal is to create a language via the use of various gestures and signs. Numerous studies have used various SLR techniques, however they wrongly refer to the issue as gesture recognition (GR).Three essential components make up the sign: Hand motions that are manual features; non-manual features include body posture or facial expressions.

It can be used as a component of a sign, and fingerprinting, in which words are gestured out in the native tongue.There are many different types of signs in each Sign Language, and each one differs from the next only slightly in terms of hand form, motion, position, non-manual elements, or context.We will construct a Sign Language Recognition Multi-Model Pipeline using these various sign kinds.Functional prerequisites.A service is a functional requirement

## 1.2 Objective

Recognition of sign language makes it easier for hearing-impaired people to communicate with the rest of society.Researchers have created a number of sign language recognition(SLR) systems, but they are only able to recognize single sign motions.Here, we introduced a Multi- Model Pipeline model that detects a series of interconnected gestures for continuous sequences of gestures or continuous SLR. By breaking down continuous signs into smaller components and modelling them with neural networks, it is done in this case.Different types of indicators have been used to test the suggested system. For persons who are dumb and deaf to communicate with the rest of the world, these various signs can be combined to form phrases.

## 1.3 Motivation

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So they depend on vision based communication for interaction. If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people. So research has been made for a vision based interface system where D&M 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, Japanese Sign Language and work has been done on other languages all around the world.

## 1.4 Existing System

Convolutional Neural Networks (CNN), are deep neural networks used to process data thathave a grid-like topology, e.g images that can be represented as a 2-D array of pixels. A CNN model consists of four main operations: Convolution, Non-Linearity (Relu), Poolingand Classification.

The concept of Transfer learning is used here, where the model is first pre-trained on a dataset that is different from the original. This way the model gains knowledge that can betransferred to other neural networks. The knowledge gained by the model, in the form of “weights” is saved and can be loaded into some other model.

## 1.5 Proposed System

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare.

## 1.6 Scope

Our major aim is to develop sign language recognition using Long short –term memory(LSTM) model to reduce communication barriers. It is based on splitting of continuous signs into sub-units and modeling them with neural networks.

One of the solutions to communicate with the deaf-mute people is by using the services of sign language interpreter. But the usage of sign language interpreters could be 3 expensive Cost-effective solution is required so that the deaf-mute and normal people can communicate normally and easily.

Our strategy involves implementing such an application which detects pre-defined American sign language (ASL) through hand gestures. For the detection of movement of gesture, we would use basic level of hardware component like camera and interfacing is required. Our application would be a comprehensive User-friendly Based system built on PyQt5 module. Instead of using technology like gloves or kinect, we are trying to solve this problem using state of the art computer vision and machine learning algorithms.

This application will comprise of two core module one is that simply detects the gesture and displays appropriate alphabet. The second is after a certain amount of interval period the scanned frame would be stored into buffer so that a string of character could be generated forming a meaningful word.

Additionally, an-addon facility for the user would be available where a user can build their own custom-based gesture for a special character like period (.) or any delimiter so that a user could form a whole bunch of sentences enhancing this into paragraph and likewise. Whatever the predicted outcome was, it would be stored into a .txt file.

## 1.7 Software & Hardware Requirements

### 1.7.1 Software Requirements

Operatin System:Windows8/Windows 10/Windows 11/Unix

Coding Language:Python

Tool:Jupyter notebook

Dataset:Own Dataset

Libraries: NumPy,OpenCV,Matplotlib,Mediapipe,TensorFlow

### 1.7.2 Hardware Requirements

Processor: Any processor over i3 and above

RAM:4GB and Higher

Hard Disk:100GB HDD or SSD

System Type:64-bit OS,x64-based processor

# CHAPTER-2

# LITERATURE SURVEY

## 2.1 Survey of Major Area relevant to Project

Sign language detection involves the recognition and interpretation of sign language gestures or movements to translate them into a form that can be understood by non-signing individuals. This field draws on knowledge and techniques from several areas of study. Here are some major areas relevant to sign language detection:

1. Computer Vision: Computer vision is the area of artificial intelligence and computer science that focuses on enabling machines to interpret and understand visual information from the world around them. In sign language detection, computer vision techniques are used to identify and track the movements of the signer's hands, face, and body.

2. Machine Learning: Machine learning is a subset of artificial intelligence that involves developing algorithms that enable computers to learn from data and make predictions or decisions based on that learning. Machine learning techniques are often used in sign language detection to train models that can recognize different signs and gestures.

3. Linguistics: Linguistics is the scientific study of language and its structure. Understanding the linguistic features of sign languages, such as the grammar, syntax, and semantics, is important for developing accurate sign language detection systems.

4. Human-Computer Interaction: Human-computer interaction (HCI) is the study of how people interact with computers and other digital technologies. In sign language detection, HCI principles are important for designing user-friendly interfaces and systems that can effectively communicate sign language to non-signing individuals.

5. Signal Processing: Signal processing is the science of analyzing, modifying, and synthesizing signals such as sound, images, and video. In sign language detection, signal processing techniques are used to preprocess and extract features from video recordings of sign language gestures.

6. Neuroscience: Neuroscience is the scientific study of the nervous system and the brain. Understanding how the brain processes and recognizes sign language can provide insights into how sign language detection systems can be improved.

7. Assistive Technology: Assistive technology refers to devices and software that are designed to assist people with disabilities in performing tasks that might otherwise be difficult or impossible. Sign language detection technology is an example of assistive technology that can enable non-signing individuals to communicate with signers.

## 2.1.1 Related Work

Sign language facilitates communication between hearing impaired peoples and the rest of the society. A number of sign language recognition (SLR) systems have been developed by researchers, but they are limited to isolated sign gestures only. In this paper, we propose a modified long short-term memory (LSTM) model for continuous sequences of gestures or continuous SLR that recognizes a sequence of connected gestures. It is based on splitting of continuous signs into sub-units and modeling them with neural networks. Thus, the consideration of a different combination of sub-units is not required during training. The proposed system has been tested with 942 signed sentences of Indian Sign Language (ISL). These sign sentences are recognized using 35 different sign words. The average accuracy of 72.3% and 89.5% has been recorded on signed sentences and isolated sign words, respectively.

## 2.1.2 Methodology

**Image Acquisition**

It is the action of extracting an image from a source, typically a hardware-based source, for process of image processing. WebCamera is the hardware-based source in our project. It is the first step in the workflow sequence because no processing can be done without an image.

**Segmentation**

The method of separating objects or signs from the context of a captured image is known as segmentation. Context subtracting, skin-color detection, and edge detection are all used in the segmentation process. The motion and location of the hand must be detected and segmented in order to recognise gestures.

**Features Extraction**

Predefined features such as form, contour, geometrical feature (position, angle, distance, etc.), colour feature, histogram, and others are extracted from the preprocessed images and used later for sign classification or recognition. Feature extraction is a step in the dimensionality reduction process that divides and organises a large collection of raw data. Reduced to smaller, easier-to-manage classes as a result, processing would be simpler. The fact that these massive data sets have a large number of variables is the most important feature. To process these variables, a large amount of computational power is needed. As a result, function extraction aids in the extraction of the best feature from large data sets by selecting and combining variables into functions.

**PreProcessing**

Each picture frame is pre-processed to eliminate noise using a variety of filters including erosion, dilation, and Gaussian smoothing, among others. The size of an image is reduced when a colour image is transformed to grayscale. A common method for reducing the amount of data to be processed is to convert an image to grey scale. The phases of pre-processing are as follows:

1.Morphological Transform

2.Blurring

3.Thresholding

4.Recognition

5.Text Output

## 2.1.3 Vision-based

Computer cameras are used as the input device for hand or finger information in vision-based approaches. The vision-based approaches just need a camera, enabling seamless contactbetween people and computers without the need for any additional hardware. These systems often describe hardware- and/or software-bas ed artificial vision systems that supplementbiological vision. This is a difficult issue since for these systems to execute in real time, they must be background invariant,

lighting insensitive, person and camera independent. Additionally, these systems need to be adjusted to satisfy the demands, which include accuracy and resilience.

The vision-based hand gesture recognition system is shown in fig

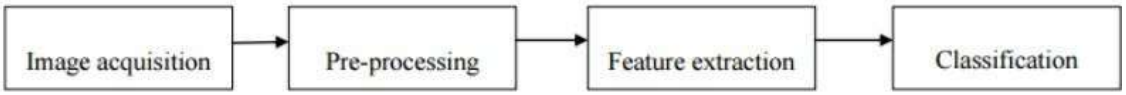



**Fig.2.1 Block Diagram of vision based recognition system**

The foundation of vision-based analysis is how people interpret information. However, it is arguably the hardest to put into action in an acceptable manner. Different approaches have been tested so far. Building a three-dimensional model of the human hand is one option. One or more cameras match the model to photos of the hand, and parameters related to joint angles and palm orientation are computed. The classification of gestures is then done using these factors.

The second step is to take the image using a camera, then extract some features, and then use those features as input into a classification system.

### 2.1.4 Automatic Sign Language Recognition for Continuous Video Sequence

The proposed system comprises of four major modules: Data Acquisition, Re-processing, Feature Extraction and Classification. Re-processing stage involves Skin Filtering and histogram matching after which Exigent-vector based Feature Extraction, and Exigent value weighted Euclidean distance-based Classification Technique were used. 24 different alphabets were considered in this paper where a 96% recognition rate was obtained

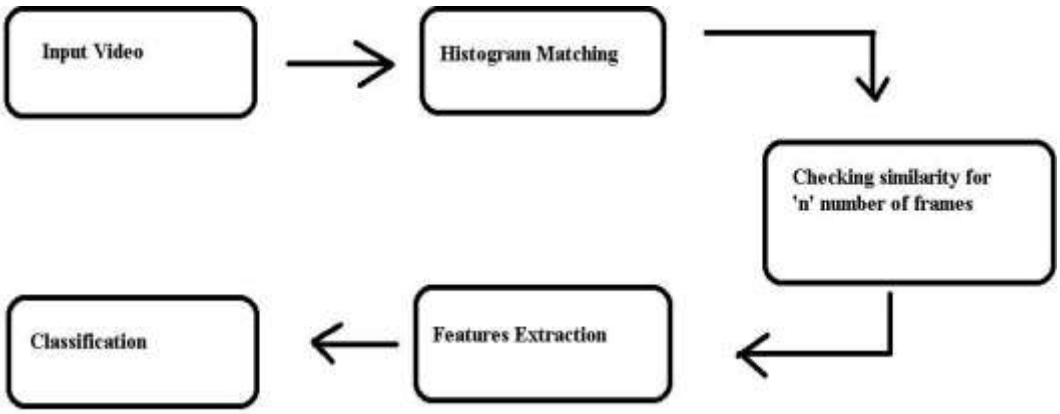



**Fig 2.2 System Overview**

### 2.1.5. Continuous Sign Language Gesture Recognition and Sentence Formation:

It is a very difficult research problem to distinguish continuous motions from sign language movements. These key-frames extraction method based on gradients was used by the researchers to tackle this issue. These key frames were useful for eliminating unnecessary frames and breaking up continuous sign language motions into sequences of signs. Each gesture has been treated separately following the division of gestures. Then, using the Orientation Histogram (OH) and Principal Component Analysis (PCA) to reduce the dimension of the features generated after the OH, features of the re-processed gestures were extracted. The Robotics and Artificial Intelligence Laboratory (IIIT-A) used their own continuous ISL dataset that was produced using a Canon EOS camera for the experiments. Numerous classifiers, including the 12 Euclidean distance, correlation, and Manhattan distance, were used to test the Sensor.

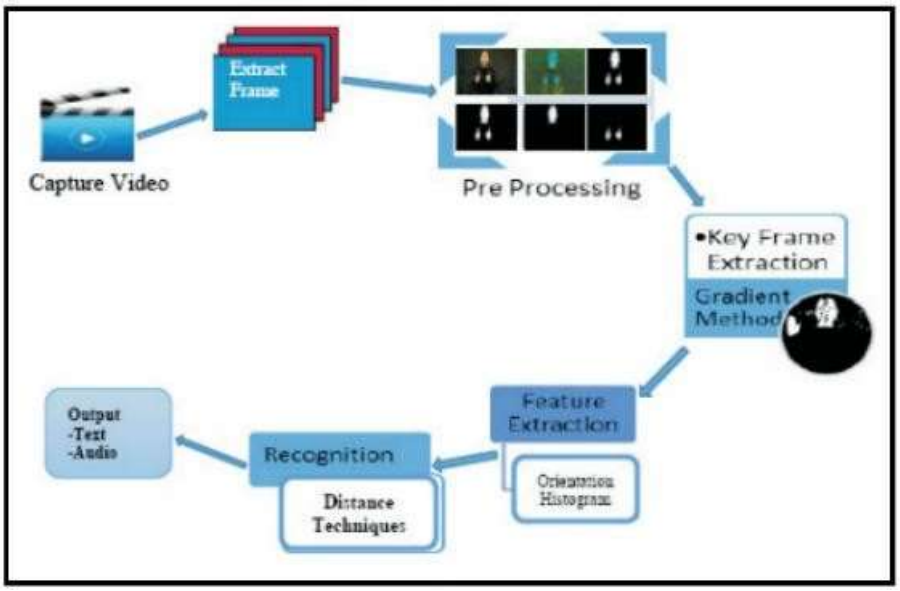



**Fig 2.3 Flow of work**

### 2.1.6 Recognition of isolated Indian Sign Language Gestures in Real Time:

This work illustrates statistical methods for simultaneous recognition of ISL movements including both hands. A database of videos was various indicators. The characteristic used for categorization owing to the direction histogram. Due to its illumination and orientation in variance appeal. Two distinct Euclidean distance and K-nearest neighboring methods were used for the recognition of neighbor metrics.

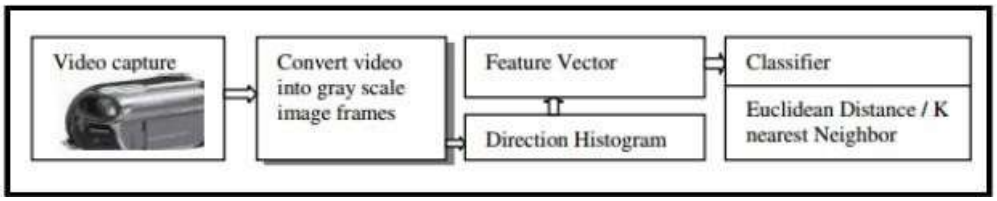



**Fig 2.4 Real-time classification of the process**

## 2.2 Techniques and algorithms applicable

Sign language facilitates communication between hearing impaired peoples and the rest ofthe society. A number of sign language recognition (SLR) systems have been developed byresearchers, but they are limited to isolated sign gestures only. In this paper, we propose amodified long short-term memory (LSTM) model for continuous sequences of gestures orcontinuous SLR that recognizes a sequence of connected gestures. It is based on splitting of continuous signs into sub-units and modeling them with neural networks. Thus, the consideration of a different combination of sub-units is not required during training. The proposed system has been tested with 942 signed sentences of Indian Sign Language (ISL).These sign sentences are recognized using 35 different sign words. The average accuracy of 72.3% and 89.5% has been recorded on signed sentences and isolated sign words, respectively.

### 2.2.1 ALGORITHM

Long Short-term Memory(LSTM)

Deep learning uses the artificial recurrent neural network (RNN) model known as long shortterm memory (LSTM). The neural network is a typical feed-forward neural network with feedback connections. It can handle both individual data points (like photos) and complete data sequences (such as speech and video). Applications of LSTM include un-segmented, linked handwriting recognition and speech recognition, for instance. An input gate, an output gate, and a forget gate are all parts of a standard LSTM machine. These three gates control how information enters and leaves the cell, allowing the cell to retain values across arbitrary time periods.

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) architecture that is designed to overcome the vanishing gradient problem in traditional RNNs.

Traditional RNNs suffer from the vanishing gradient problem, where the gradients used to update the weights in the network become very small as they propagate through many time steps. This can result in the network being unable to learn long-term dependencies.

LSTM networks use a series of memory cells and gates to selectively remember or forget information over time. The three main components of an LSTM cell are the input gate, the forget gate, and the output gate. The input gate controls whether new input is added to the memory cell, the forget gate controls whether information is removed from the memory cell, and the output gate controls what information is output from the memory cell.

LSTM networks have been widely used in various applications, including natural language processing, speech recognition, and image captioning, among others. They have shown to be effective in capturing long-term dependencies and are capable of learning and remembering complex patterns in data.

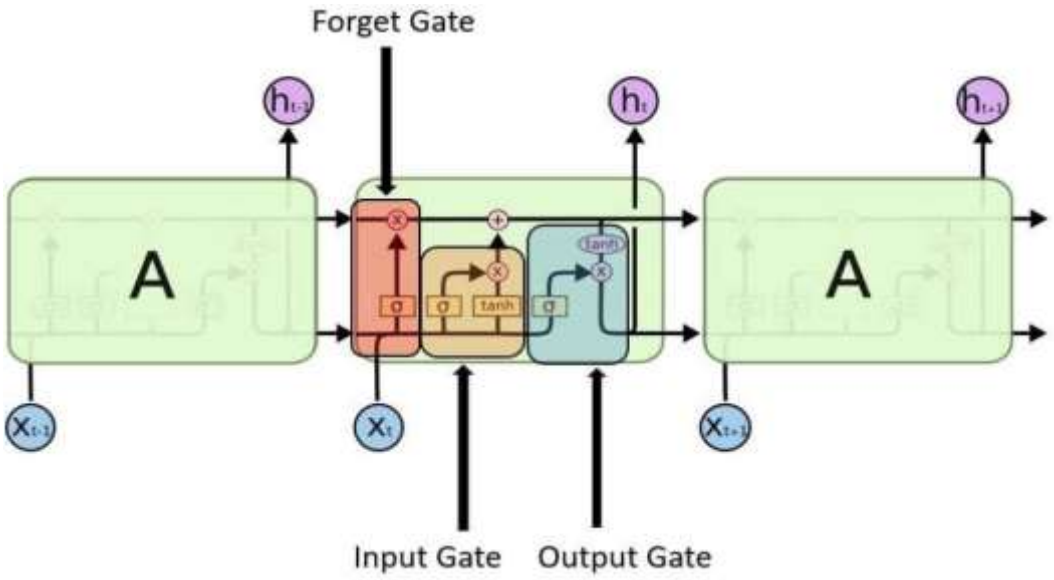



**Fig 2.5 LSTM process of flow**

## 2.2.2 LSTM Layers

The basic building block of an LSTM network is the LSTM cell, which contains several layers that work together to selectively remember or forget information over time.

Here are the different layers in an LSTM cell:

1. Input gate: This layer decides whether to let new input into the LSTM cell or not. It takes the current input and the previous hidden state as inputs and applies a sigmoid function, which generates a number between 0 and 1 that represents the relevance of the new input.
2. Forget gate: This layer decides whether to forget the previous memory or not. It takes the previous hidden state and the current input as inputs and applies a sigmoid function, which generates a number between 0 and 1 that represents the degree to which each memory should be retained.
3. Memory cell: This layer stores the long-term memory of the LSTM cell. It updates the previous memory based on the previous memory, the previous hidden state, and the current input.

4. Output gate: This layer decides what information to output from the LSTM cell. It takes the current input and the previous hidden state as inputs and applies a sigmoid function, which generates a number between 0 and 1 that represents the relevance of the output. It then applies a tanh function, which generates a number between -1 and 1 that represents the value of the output.
5. Hidden state: This layer represents the short-term memory of the LSTM cell. It is calculated based on the current input, the previous hidden state, the input gate, and the output gate.

Overall, the LSTM layers work together to selectively retain and discard information from the input, the memory, and the hidden state, allowing the network to capture long-term dependencies in sequential data. The number of LSTM cells and layers used in a network can be adjusted based on the complexity of the problem being solved.

The LSTM layers should be built using a function that can manage the size and number of layers dynamically. The service will accept a list of LSTM sizes, and depending on the list's length, can determine how many LSTM layers are present (e.g., our example will use a list of length 2, containing the sizes 128 and 64, indicating a two-layered LSTM network where the first layer size 128 and the second layer has hidden layer size 64).

A TensorFlow MultiRNN cell is then given the list of dropout wrapped LSTMs in order to stack the layers.

Loss function, optimizer and accuracy

Finally, we develop functions to specify our model's accuracy, optimizer, and loss function. Even if the accuracy and loss are simply determined based on the findings, everything in TensorFlow is a component of a computation graph.

### 2.2.3 Building the graph and training

To build the network, we first call each of the defined functions. Next, we call a TensorFlow session to train the model using mini-batches over a predetermined number of epochs. We will print the loss, training accuracy, and validation accuracy at the conclusion of each epoch to keep track of the outcomes as we train the model.

### 2.2.4 Testing

In order to design an algorithm that compares the learned data to the new dataset, we first create a new dataset. and deliver the function's precision.

### 2.2.5 MediaPipe

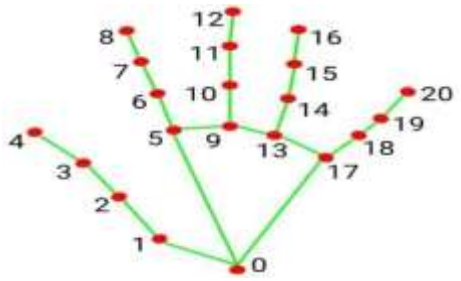
The individual models for the stance, face, and hand components—each of which is optimised for a specific platform—are combined via the MediaPipe Holistic pipeline. But because they each have different areas of expertise, some input is inappropriate for other components. The pose estimation model treats their input as a video frame with a lower, fixed resolution (256x256). However, the image resolution would be too poor to make an accurate remark if one were to trim the hand and facial portions from that image to send to their corresponding models. As a result, MediaPipe Holistic was created as a multi-stage pipeline that addresses the various regions according to the relevant region for picture resolution.There are three landmarks there.

1. Hand Landmarks: There are 21 landmarks on each hand.
2. Pose Landmarks: There are 33 of these.
3. Face Landmarks: There are 468 landmarks in total.

## 2.2.6 Hand Landmarks

Understanding the shape and movement of hands can play a key role in enhancing user experience across a variety of technology platforms and domains. It can serve as the foundation for hand gesture control and sign language comprehension, as well as for the overlay of digital information and content on top of the real environment. Although it comes effortlessly to individuals, strong real-time hand perception is an extremely difficult computer vision problem because hands frequently overlap (for example, in handshakes and finger/palm inclusions) and lack high contrast patterns.

A high-precision hand and finger tracking solution is MediaPipe Hands. From a single shot, it uses machine learning (ML) to extract 21 3D landmarks of a hand. Our method enables realtime performance of several hands scale, while current state-of-the-art approaches generally rely on powerful desktop environments for inference. It gathers the landmarks of the palm and stores them in the function before detecting the palm. It then finds the hand landmarks.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

**Fig 2.6 Hand Landmarks**

## 2.2.7 Pose Landmarks

In many applications, such as measuring physical activity, understanding sign language, and controlling full-body gestures, human position estimation from video is essential. It is possible to create the based on fitness, yoga, and dance applications. Additionally, it can make it possible to display digital information and material on top of the real environment.

A machine learning (ML) solution for high-precision body position tracking, MediaPipe Pose uses BlazePose to extract 33 3D landmarks and a backdrop segmentation mask from RGB video frames. Our solution achieves real-time performance on the majority of smart phones, desktops, and laptops with the aid of Python, unlike current state-of-the-art systems that mostly rely on strong desktop environments for inference.

## 2.2.8 Pose Estimation Quality

To compare our models' quality to that of other effective models. We employ three distinct validation datasets that reflect three distinct industry verticals: yoga, dance, and HIIT. Only one subject is seen in each image taken by the camera. We only take into account landmarks for 17 keypoints in the COCO topology. 33 pose landmarks are predicted by the landmark model in MediaPipe Pose.

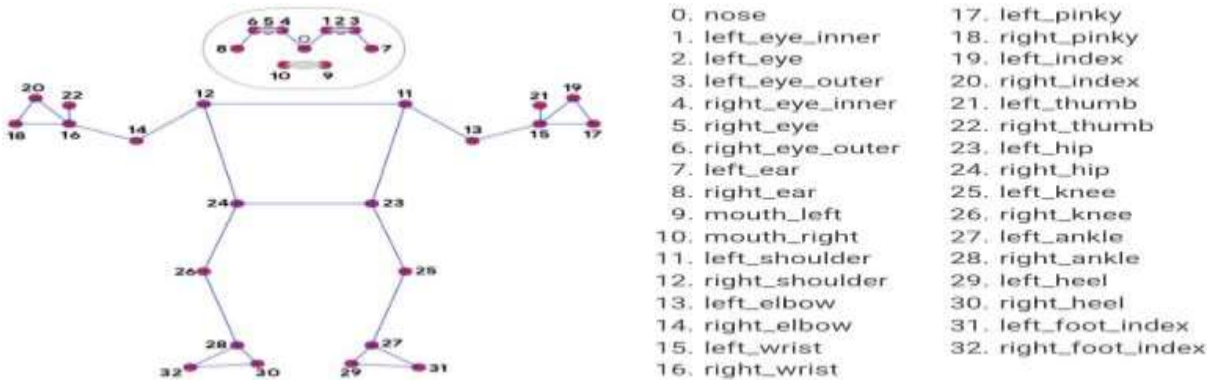



**Fig 2.7 Pose Landmarks**

### 2.2.9 Face Mesh Landmarks

A technology called MediaPipe Facial Mesh evaluates 468 3D face landmarks in the real world. Machine learning (ML) is used to create the 3D facial surface. It doesn't require a specialised depth sensor and simply needs one camera input. uses GPU acceleration and lightweight model structures throughout the pipeline. The Facial Transform module bridges the gap between accurate real-time augmented reality (AR) applications and face landmark estimation. In order to estimate a face transform within that space, it establishes a 3D space metric and employs the face landmark and screen position. The face transform data is made up of typical 3D primitives such a triangular face mesh and a face position transformation matrix.

### 2.2.10 Face Landmark Model

Fetch learning was used to train a network for 3D face landmarks, which concurrently predicts 3D landmark coordinates on rendered synthetic data and 2D semantic contours on annotated real-world data. We can make reasonable 3D landmark predictions using the network's output on both artificially rendered and real-world data. Without further depth information, a cropped video frame is provided to the 3D landmark network as input. The model returns the 3D point locations as well as the likelihood that a face is present and adequately aligned in the input data. By iteratively bootstrapping and 16 adjusting predictions, we can increase the precision and stability of our model. By doing so, we may expand our dataset to include more difficult scenarios, such grimaces, oblique angles, and occlusions.

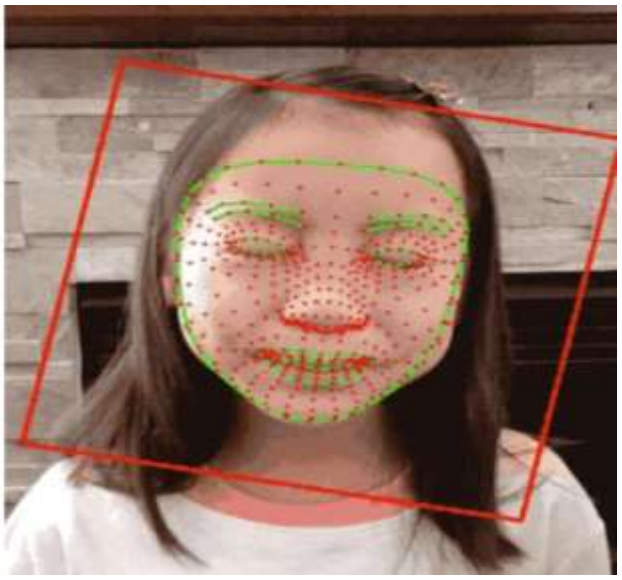


**Fig 2.8 Capturing landmarks**

## 2.3 Applications

There are various ways of communication or expression, but the predominant mode of human communication is speech; when it is hindered, people must use a tactile- kinesthetic mode of communication instead.In India, the overall percentage of persons with this disability in the population was 2.2 percent.

1. Improving accessibility: Sign language detection can be used to create tools that help deaf and hard of hearing people communicate with those who do not understand sign language. For example, a sign language detection app could translate sign language into spoken or written language, allowing for more seamless communication.

2. Education: Sign language detection can be used in educational settings to help teach sign language. For example, a sign language detection system could provide real-time feedback to a student who is learning sign language, helping them improve their skills.

3. Interpreting: Sign language detection can be used to improve the accuracy of sign language interpreting. For example, a sign language detection system could be used to monitor the accuracy of an interpreter and provide real-time feedback to help them improve.

4. Healthcare: Sign language detection can be used in healthcare settings to improve communication between doctors and patients who use sign language. For example, a sign language detection system could translate sign language into spoken or written language for a doctor who does not understand sign language.

5. Assistive technology: Sign language detection can be used in assistive technology to improve the accessibility of various devices. For example, a sign language detection system could be used to control a smart home device or a computer using sign language, making it easier for deaf and hard of hearing people to use these technologies.

## 2.4 Summary

The topic of human computer interaction has several potential applications for hand gestures, which are an effective method for communication. The technique for recognisinghand gestures using vision has several demonstrated benefits. Videos contain both temporaland spatial characteristics, making them challenging to analyse. To categorise based on thespatial and temporal data, we have employed models. On both characteristics, LSTM was employed for classification. Sign language recognition is a hard problem if we consider allthe 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 toone of them. The system didn't perform too well but it was demonstrated that it can be builta first-person sign language translation system using only cameras and convolutional neuralnetworks. It was observed that the model tends to confuse several signs with each other, such as U and W. But thinking a bit about it, maybe it doesn't need to have a perfect performance since using an orthography corrector or a word predictor would increase the translation accuracy. The next step is to analyse the solution and study ways to improve thesystem. Some improvements could be carrying by collecting more quality data, trying moreconvolutional neural network architectures, or redesigning the vision system.

# CHAPTER-3

# SYSTEM DESIGN

## 3.1 System Architecture / Block Diagram

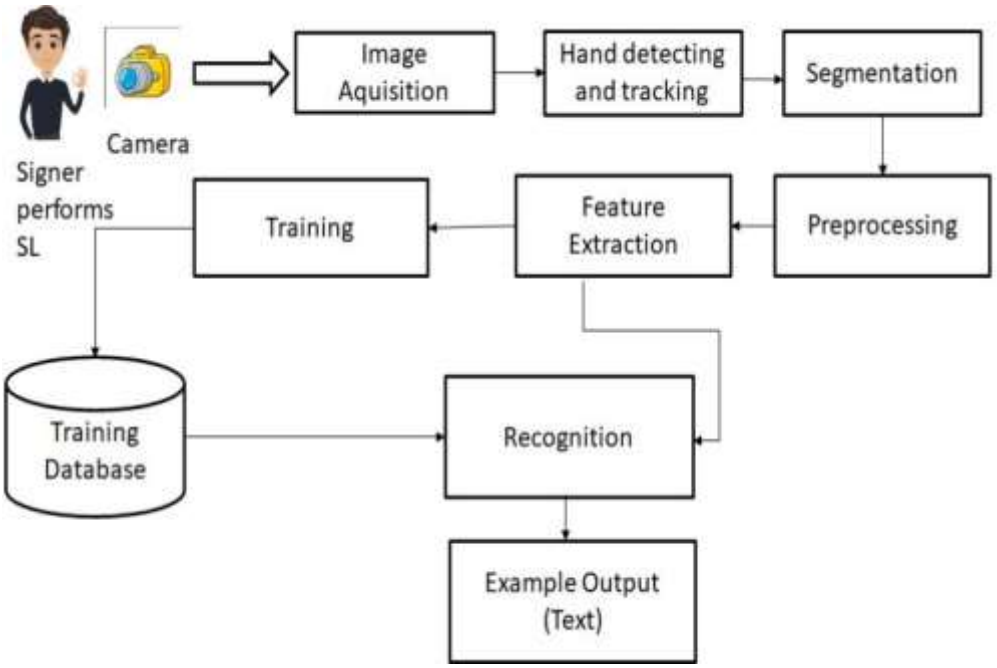



**Fig 3.1 System Architecture**

Web Camera is the hardware-based source in our project. It is the first step in the workflowsequence because no processing can be done without an image. The picture that is obtainedhas not been processed in any way.

The method of separating objects or signs from the context of a captured image is known as segmentation. Context subtracting, skin-color detection, and edge detection are all usedin the segmentation process. The motion and location of the hand must be detected and segmented to recognise gestures.

Predefined features such as form, contour, geometrical feature (position, angle, distance, etc.), colour feature, histogram, and others are extracted from the pre-processed images andused later for sign classification or recognition. Feature extraction is a step in the dimensionality reduction process that divides and organises a large collection of raw data.Reduced to smaller, easier-to-manage classes as a result, processing would be simpler. Thefact that these massive data sets have many variables is the most important feature.

To process these variables, a large amount of computational power is needed. As a result, function extraction aids in the extraction of the best feature from large data sets by selectingand combining variables into functions. Reducing the size of the data These features are simple to use while still accurately and uniquely describing the actual data collection

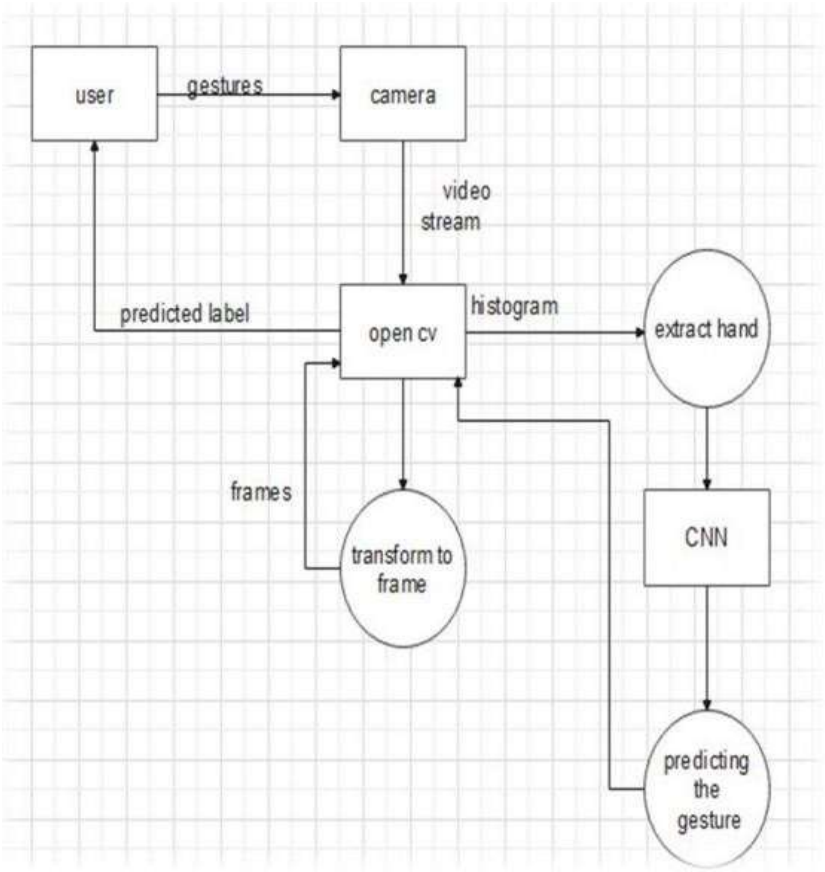



**Fig 3.2 Data Flow**

The DFD is also known as bubble chart. It is a simple graphical Formalism that can be usedto 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 wouldbe hard to explain in words and they work for both technical and non- technical. There arefour components in DFD:

External Entity

Process,Data Flow,data Store

## 3.2 Diagrams applicable

### Class Diagram

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

Analysis and design of the static view of an application.

Describe responsibilities of a system.

Base for component and deployment diagrams.

Forward and reverse engineering.

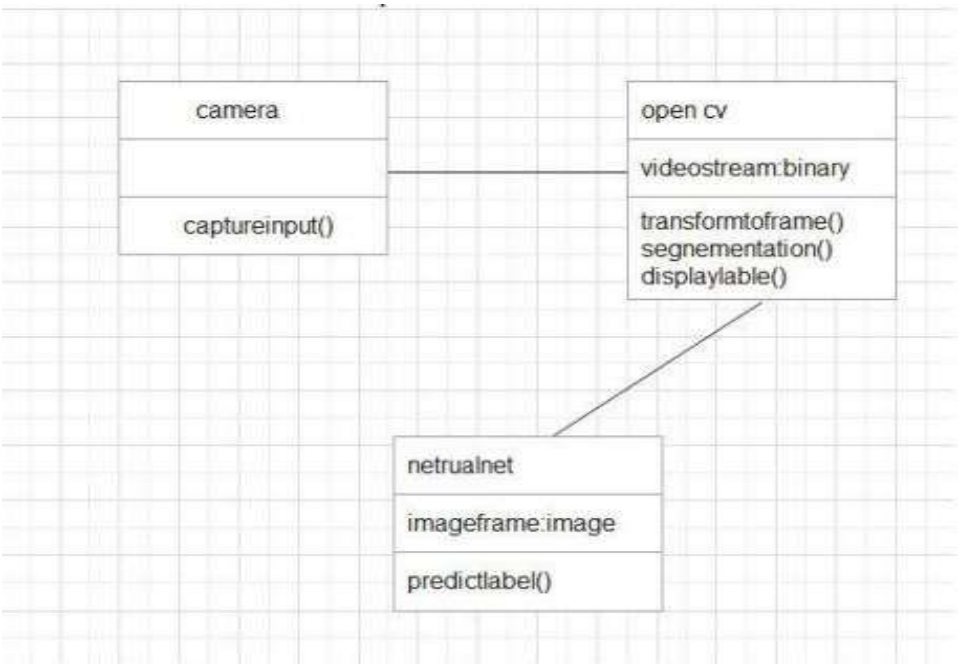



**Fig 3.3 Class Diagram**

## Use Case Diagram

The main purpose of use case diagrams is to capture the dynamic aspects of the system.Use case diagram models the dynamic behaviour of a system. It describes the functionality and users of the system and it models the complete behaviour of a system.So, when a system is analysed to gather its functionalities use cases are prepared and actors are identified.

Use-Case used to gather the requirements of a system.

Use-Case used to get an outside view of a system.

Identify the external and internal factors influencing the system.

Show the interaction among the requirements are actors.

Representation of the Use cases:

A **use case** is represented by an **Ellipse**.

**System boundary** is represented by a **Rectangle**.

**Users** are represented by **Stick person icons (Actors)**.

**Communication relationship between actor and use ca**se is represented by a line.

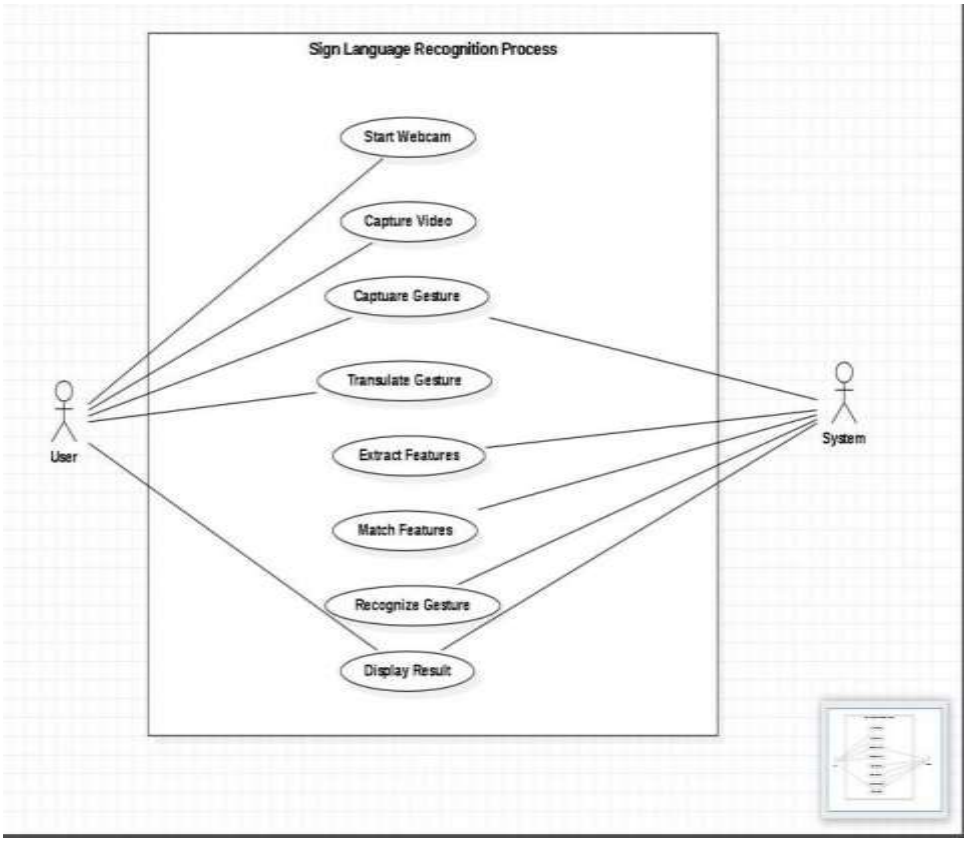



**Fig 3.4 Use Case Diagram**

## Sequence Diagram

Sequence diagram is an interaction diagram that models the dynamic aspects of a system.
Mainly, interaction diagrams are used to describe some type of interactions among different elements in the model. Sequence diagram emphasizes on time ordering of messages. These are used to visualize the interactive behaviour of a system. An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.
Sequence diagrams are used to
Capture the dynamic behaviour system of a system.
Describe interaction among objects.
Describe the message flow in the system. Sequence diagram contains the following elements:
Lifelines: It represents the existence of an object over a period of time.
Activations: It represents the time duration of an object performing anoperation.
Messages: It represents communication between objects.

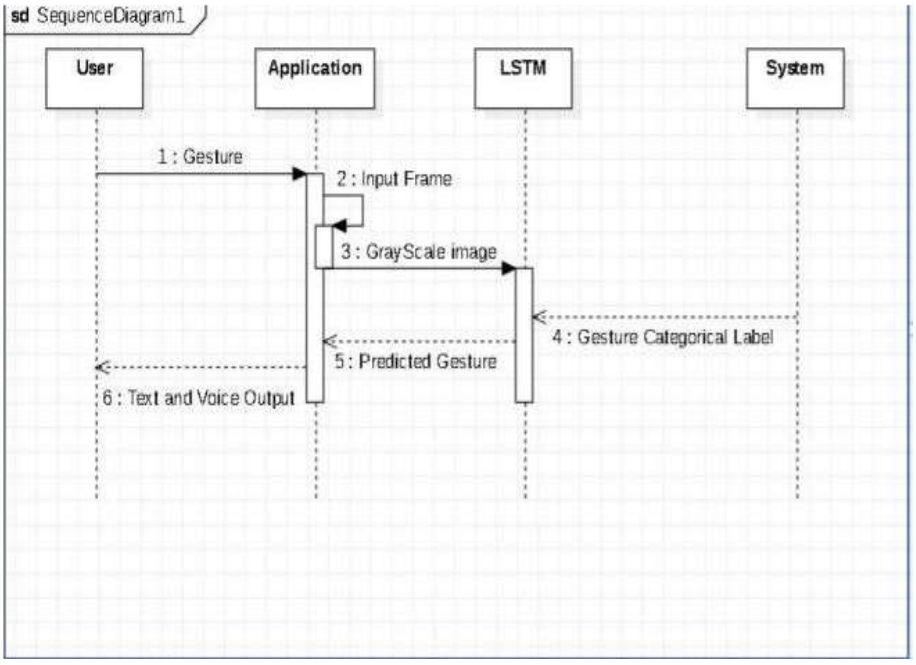



**Fig 3.5 Sequence Diagram**

## Activity Diagram

An activity diagram is a special kind of a state chart diagram that illustrates the flow from one activity to another activity. It is the behavioural diagram i.e.; it depicts the behaviour of a system. The activity is explained 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 type of flow by using elementslike activity, fork, join etc. It portrays the control flow from a start point to finish point showing the various decision paths that exist while the activity is being executed.

Elements of Activity diagram include:

Fork: A fork represents the splitting of a single node of control into two or moreconcurrent flow of control.

Join: A join represents the synchronization of two or more concurrent flows ofcontrol.

Branch: A branch specifies alternate paths takes based on some Booleanexpressions.

The purpose of an activity diagram is to model the dynamic behaviour of a system.

It illustrates dynamic process view of a system.

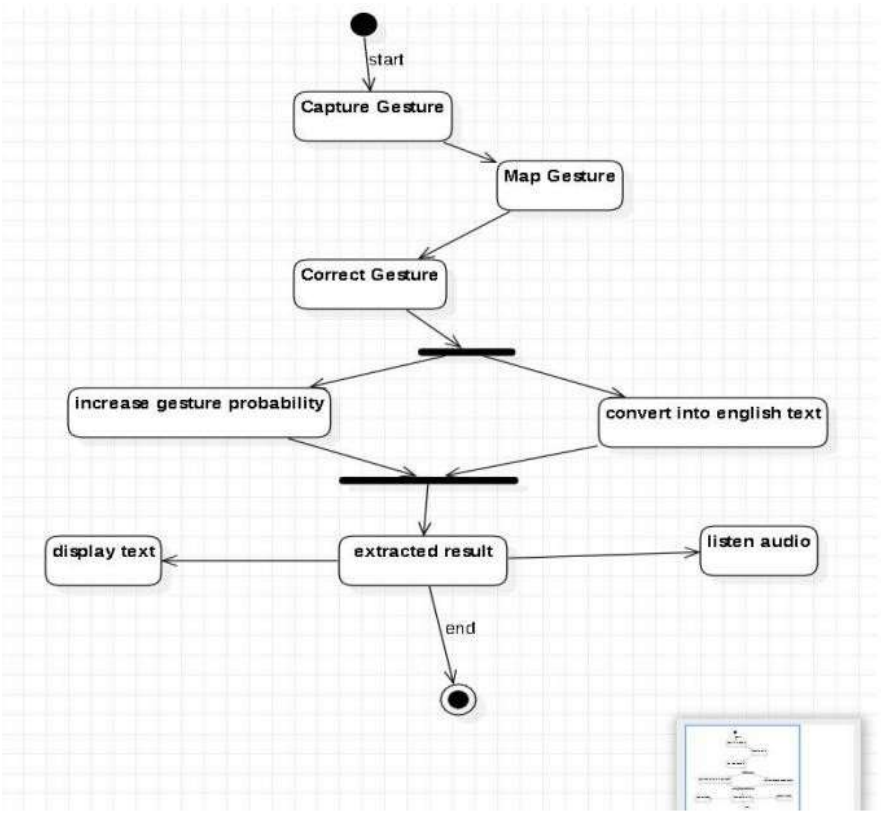



**Fig 3.6 Activity Diagram**

# CHAPTER 4

# Implementation

## 4.1 Environmental Setup

### 4.1.1 Language Used

Coding Language-Python Python facilitates developers to increase the confidence and productivity about their developing software from development to deployment and maintenance. The benefits of making Python the perfect solution for machine learning and AI-driven projects include simplicity and consistency, flexibility, access to powerful AI and machine learning (ML) libraries and frameworks, platform independence, and large communities. These things increase the popularity of the language.

Python is easy to use, learn, and it is versatile too. It means that Python, which is used to develop machine learning, can run on all platforms, including Windows, Linux, Unix, macOS, and 21 others. To shift the process from one platform to another, developers implement some minor changes and modify a few lines of code to create executable code for the selected platform. Developers can use software packages such as PyInstaller to prepare code to run on different platforms. That saves time and money on testing across other platforms and makes the process easier and more convenient. These are the packages used while implementing:

- NumPy
- OpenCV
- Matplotlib
- Mediapipe
- TensorFlow

### 4.1.2 Methods Used

We used LSTM to design a model using machine learning in python language. These are the following methods used during project implementation:

- Training and Testing methods
- Read () methods to fetch the data.

System take the input of data through action sequence of the actions performed in video. The actions performed in video sequences are internal converted in no. of frames specified by value.By using open CV, we extracted key points from the video sequence. We took three sample inputs from the signs which recognizes as hello, thanks and iloveyou.

### 4.1.3 Methodology

We begin by gathering data from important mediapipe holistic points, including our hands, our bodies, and our faces, and then we record the data as numpy arrays. Depending on our needs, we can change the number of sequences, but each series will include 30 frames. We next create an LSTM model and train it using the data we've already collected, which enables us to recognise action over a range of frames. We choose the number of epochs for the model; while doing so improves accuracy, doing so also lengthens the time it takes to run the model and raises the risk of over-fitting the model for gesture recognition. We can use this once training is finished.

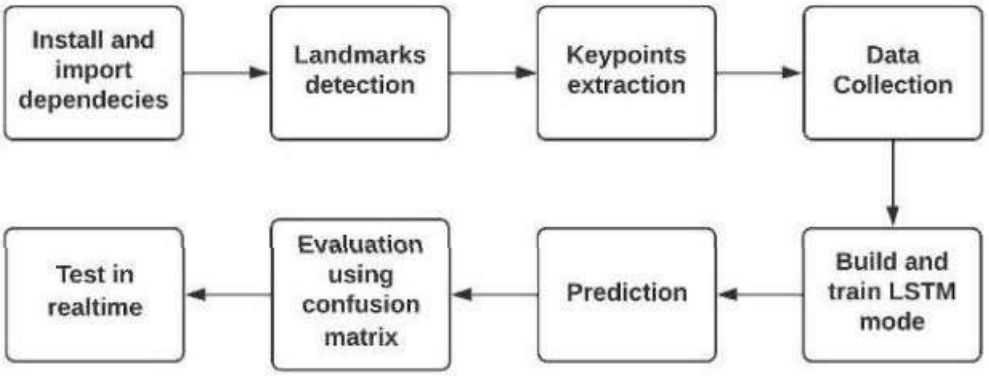



**Fig 4.1 Methodology**

Model is trained by LSTM which not only collect the single data points but also sequences ofdata points. Model summary describes total parameters follows under LSTM criteria.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 64)            442112

 lstm_1 (LSTM)               (None, 30, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 12)                396

=================================================================
Total params: 596,972
Trainable params: 596,972
Non-trainable params: 0
_________________________________________________________________
```

**Fig 4.2 Model Summary**


**Fig 4.3 Gesture Signs**

## 4.2 Implementaion each Module

### 4.2.1 Image Acquisition

It is the action of extracting an image from a source, typically a hardware-based source, for process of image processing. WebCamera is the hardware-based source in our project. It is the first step in the workflow sequence because no processing can be done without an image.

### 4.2.2 PreProcessing

Each picture frame is pre-processed to eliminate noise using a variety of filters including erosion, dilation, and Gaussian smoothing, among others. The size of an image is reduced when a colour image is transformed to grayscale. A common method for reducing the amount of data to be processed is to convert an image to grey scale. The phases of pre-processing are as follows:

1.Morphological Transform

2.Blurring

3.Thresholding

4.Recognition

5.Text Output

### 4.2.3 Segmentation

The method of separating objects or signs from the context of a captured image is known as segmentation. Context subtracting, skin-color detection, and edge detection are all used in the segmentation process. The motion and location of the hand must be detected and segmented in order to recognise gestures.

### 4.2.4 Features Extraction

Predefined features such as form, contour, geometrical feature (position, angle, distance, etc.), colour feature, histogram, and others are extracted from the preprocessed images and used later for sign classification or recognition. Feature extraction is a step in the dimensionality reduction process that divides and organises a large collection of raw data. Reduced to smaller, easier-to-manage classes as a result, processing would be simpler. The fact that these massive data sets have a large number of variables is the most important feature. To process these variables, a large amount of computational power is needed. As a result, function extraction aids in the extraction of the best feature from large data sets by selecting and combining variables into functions.

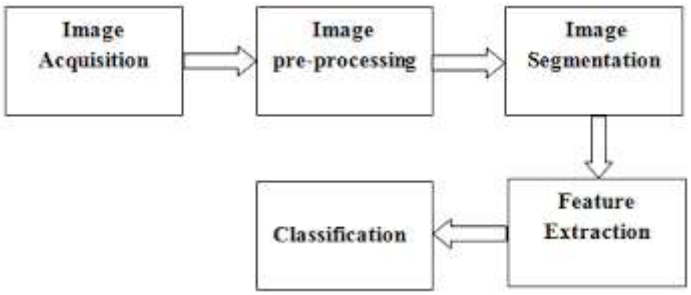



**Fig 4.4 Modules**

## 4.3 Integration & Deployment

Integrating and deploying a sign language detection model using LSTM involves several steps. Here is a high-level overview of the process:

1. Data preparation: Collect sign language videos and preprocess them to extract features such as hand gestures, facial expressions, and body language.

2. Model development: Train an LSTM-based model on the preprocessed data. The model should take in a sequence of features and output a prediction for the corresponding sign.

3. Model evaluation: Evaluate the performance of the model on a test set of sign language videos. Use metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.

4. Model optimization: Fine-tune the model parameters and architecture to improve performance. Use techniques such as regularization, dropout, and hyperparameter tuning to achieve the best possible performance.

5. Integration: Integrate the trained model into a larger system, such as a mobile app or a web application. The integration process may involve converting the model into a format that can be easily loaded into the application.

6. Deployment: Deploy the integrated system to a production environment. This may involve setting up servers, configuring networking, and ensuring that the system is secure and scalable.

7. Monitoring and maintenance: Monitor the deployed system for errors and performance issues. Update the system as needed to fix bugs, add new features, and improve performance.

Overall, integrating and deploying a sign language detection model using LSTM requires a combination of data science, software engineering, and DevOps skills. It is a complex process that requires careful planning, testing, and iteration to achieve the best possible results.

# CHAPTER 5

# Testing & Results

## 5.1 Software Testing

In machine learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from the both training and validation sets, but it should follow the same probability distribution as the training set.

We performed following tests on the model:

• Unit tests. The program is broken down into blocks, and each element (unit) is tested separately.

• Regression tests. They cover already tested software to see if it doesn't suddenly break.

• Integration tests. This type of testing observes how multiple components of the program work together.

**Test Cases**

We usually write two different classes of tests for Machine Learning systems.

• Pre-train tests

• Post-train tests

**Pre-train tests**: The intention is to write such tests which can be run without trained parameters so that we can catch implementation errors early on. This helps in avoiding the extra time and effort spent in a wasted training job. we can test the following in the pre-train test:

• The model predicted output shape is proper or not

• Test dataset leakage i.e., checking whether the data in training and testing datasets have no duplication • Temporal data leakage which involves checking whether the dependencies between training and test data do not lead to unrealistic situations in the time domain like training on a future data point and testing on a past data point

• Check for the output ranges. In the cases where we are predicting outputs in a certain range (for example when predicting probabilities), we need to ensure the final prediction is not outside the expected range of values.

• Ensuring a gradient step training on a batch of data leads to a decrease in the loss

• Data profiling assertions

**Post-train tests:** post-train tests are aimed at testing the model's behaviour. We want to test the learned logic and it could be tested on the following points and more:

• Invariance tests which involve testing the model by tweaking only one feature in a data point and checking for consistency in model predictions. For example, if we are working with a loan prediction dataset then change in sex should not affect an individual's eligibility for the loan given all other features are the same or in the case of titanic survivor probability prediction data, change in the passenger's name should not affect their chances of survival.

• Directional expectations wherein we test for a direct relation between feature values and predictions. For example, in the case of a loan prediction problem, having a higher credit score should definitely increase a person's eligibility for a loan.

• Apart from this, you can also write tests for any other failure modes identified for your model

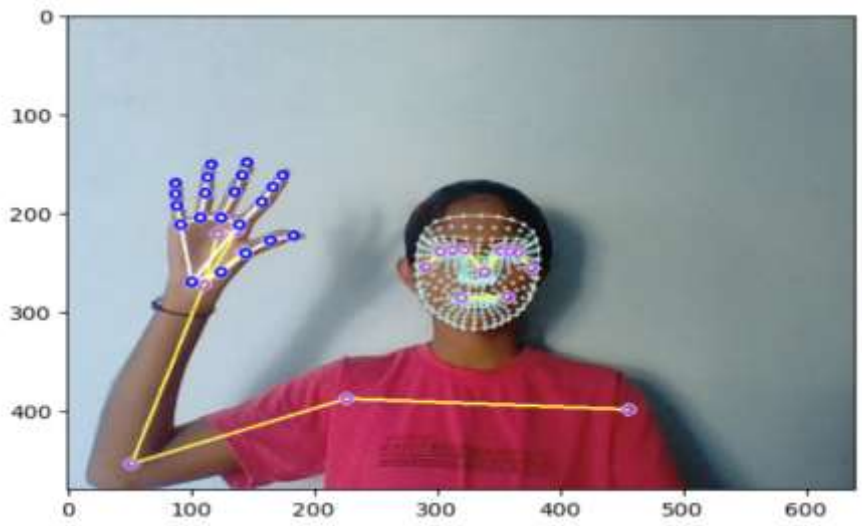



**Fig 5.1 Hand and face landmarks detection**

## 5.2 DataSets

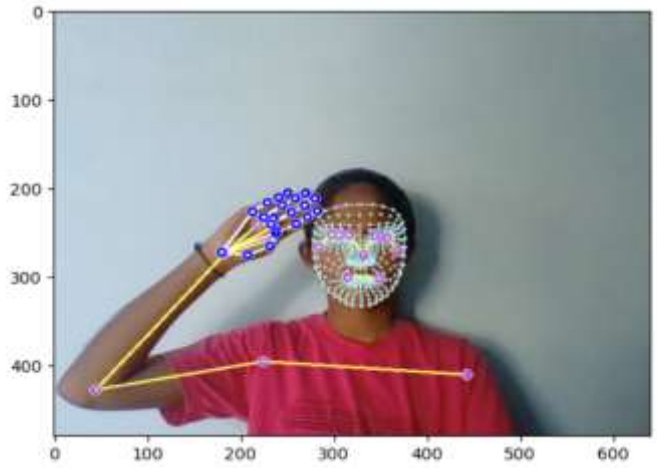



**Fig 5.2 Gestures sign for hello**

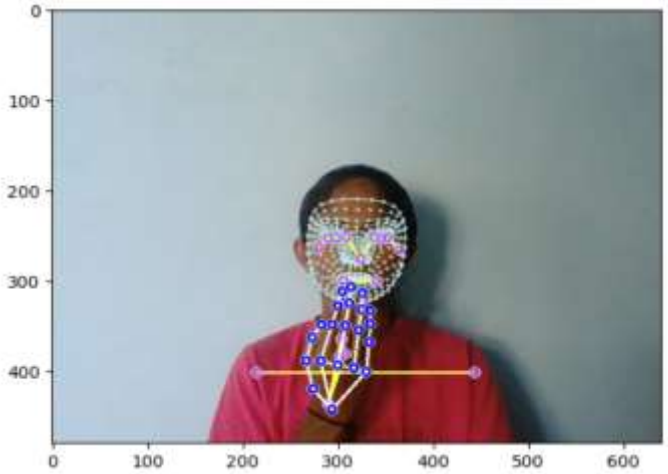



**Fig 5.3 Gestures sign for Thankyou**

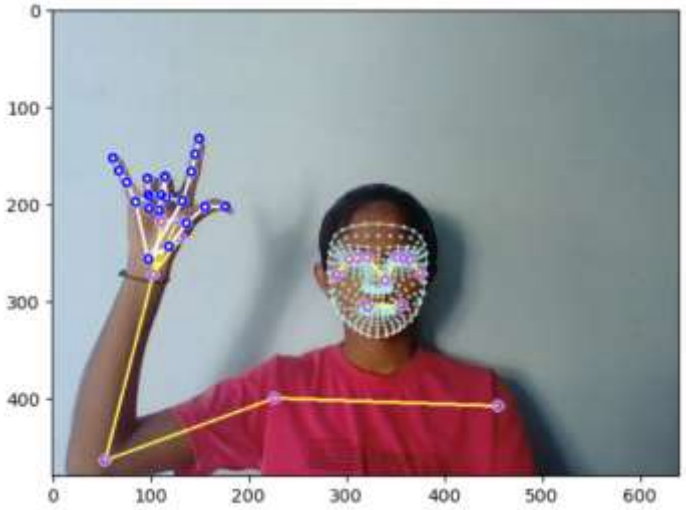



**Fig 5.4 Gestures sign for I Love You**

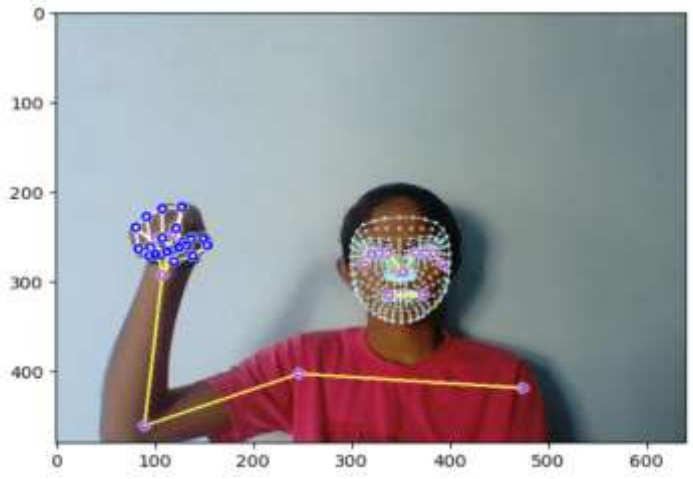



**Fig 5.5 Gestures sign for Yes**

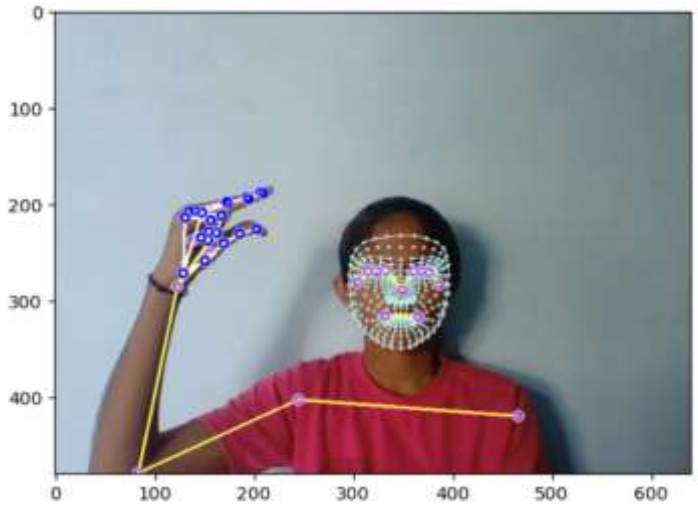



**Fig 5.6 Gestures sign for No**

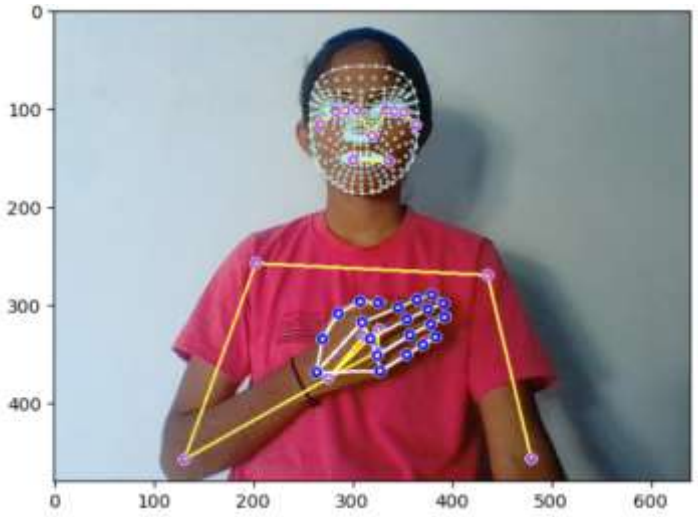



**Fig 5.7 Gestures sign for Please**

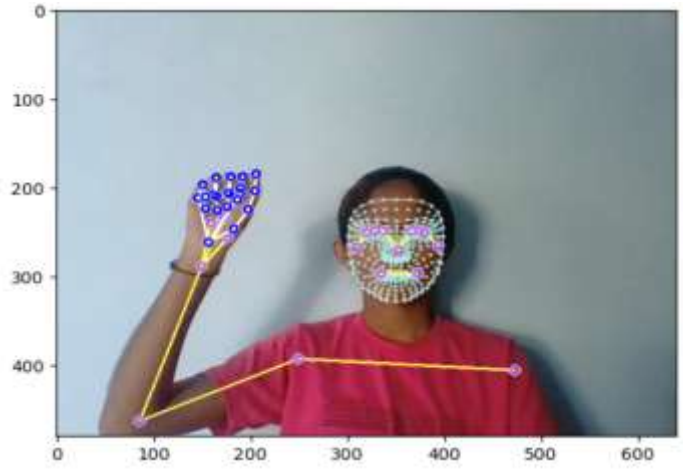



**Fig 5.8 Gestures sign for Good Bye**

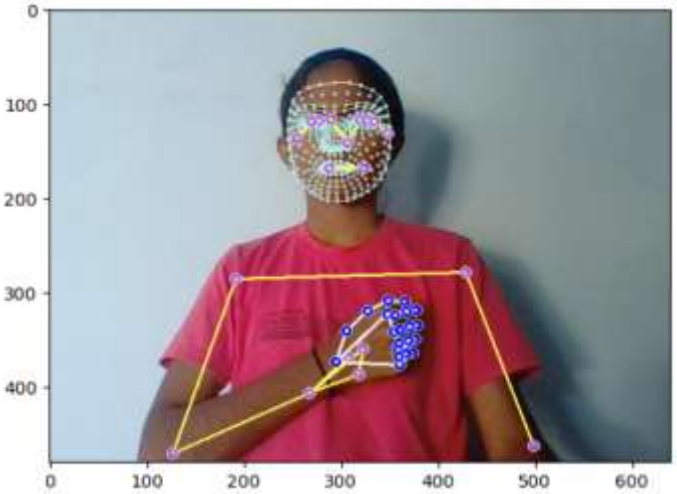



**Fig 5.9 Gestures sign for Sorry**

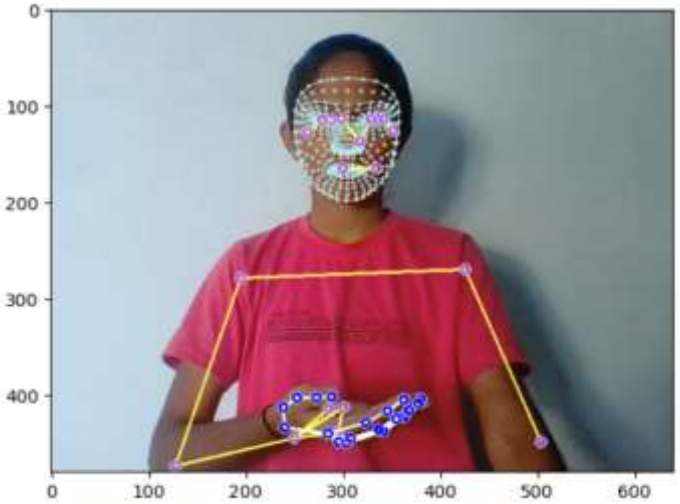



**Fig 5.10 Gesture sign for You are Welcome**

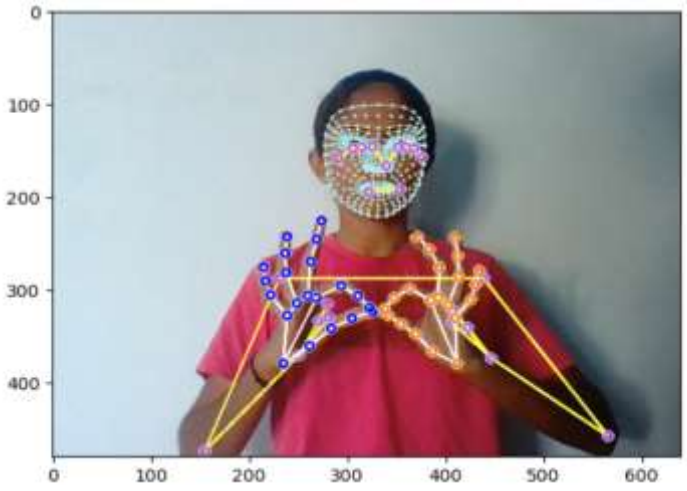



**Fig 5.12 Gestures sign for Family**

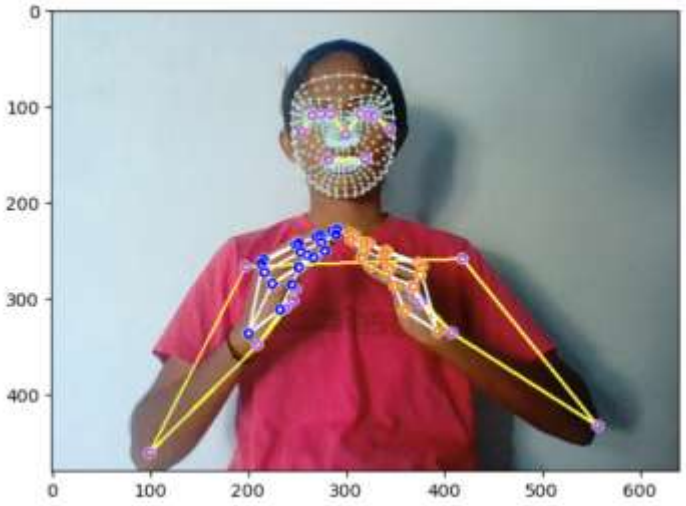



**Fig 5.13 Gestures sign for Home**

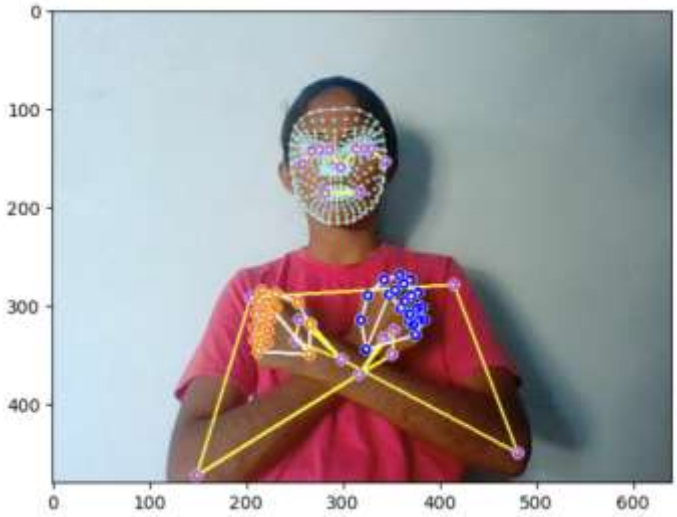



**Fig 5.14 Gestures sign for Love**

## 5.4 Results


**Fig 5.15 Output based on action performed for 3 signs**

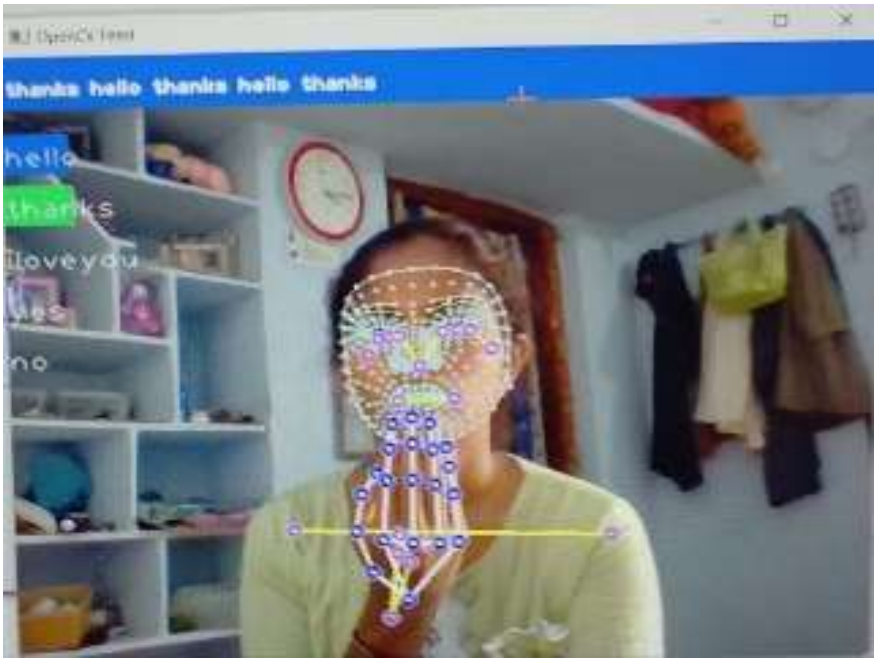



**Fig 5.16 Output based on action performed for 5 signs**


**Fig 4.14 Output based on action performed for 12 signs**

# CHAPTER-6

# CONCLUSION & FUTURE ENANCEMENT

## 6.1 Conclusion

The topic of human computer interaction has several potential applications for hand gestures, which are an effective method for communication. The technique for recognisinghand gestures using vision has several demonstrated benefits. Videos contain both temporaland spatial characteristics, making them challenging to analyse. To categorise based on thespatial and temporal data, we have employed models. On both characteristics, LSTM was employed for classification. Sign language recognition is a hard problem if we consider allthe 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 toone of them. The system didn't perform too well but it was demonstrated that it can be builta first-person sign language translation system using only cameras and convolutional neuralnetworks. It was observed that the model tends to confuse several signs with each other, such as U and W. But thinking a bit about it, maybe it doesn't need to have a perfect performance since using an orthography corrector or a word predictor would increase the translation accuracy. The next step is to analyse the solution and study ways to improve thesystem. Some improvements could be carrying by collecting more quality data, trying moreconvolutional neural network architectures, or redesigning the vision system.

## 6.2 Future Enhancements

We can develop a model for sing language word and sentence level recognition. This will require a system that can detect changes with respect to the temporal space. We can develop a complete product that will help the speech and hearing-impaired people, and thereby reduce the communication gap.

Image Processing part should be improved so that System would be able to communicate in both directions i.e.it should be capable of converting normal language to sign language and vice versa. We will try to recognize signs which include motion. Moreover, we will focus on converting the sequence of gestures into text i.e., word and sentences and then converting it into the speech which can be heard.

Sign Language Recognition System has been developed from classifying only static signs and alphabets, to the system that can successfully recognize dynamic movements that comes in continuous sequences of images. Researcher nowadays are paying more attention to make a large vocabulary for sign language recognition systems. Many researchers are developing their Sign Language Recognition System by using small vocabulary and selfmade database. Large database build for Sign Language Recognition System is still not available for some of the country that involved in developing Sign Language Recognition System. Especially the Kinect-based data, which provide the colour stream and depth stream video. The classification method of identifying the sign language is also varied from researchers. Using their own ideas and limitations for the Sign Language Recognition System, the comparison of one method to another method is still subjective. Fair and direct comparison between approaches is limited because of the variation of sign language in different countries and the difference in limitation set by each researcher. Variation of sign language in most of the country is based on their grammar and their way to present each word, such as presenting the language by word or by sentence

# REFERENCES

[1].P. K. Athira, C. J. Sruthi, A. Lijiya, "A Signer Independent Sign Videos:IndianScenario",2019.https://www.sciencedirect.com/science/article/pii/S1319157818 31228 X

[2].M. Jaiswal, V. Sharma, A. Sharma, S. Saini, R. Tomar, "An Efficient Binarized NeuralNetwork for Recognizing Two Hands Indian Sign Language Gestures in Real-time Environment",2020 https://ieeexplore.ieee.org/abstract/document/9342454

[3].C. J. Sruthi, A. Lijiya, "Signet: A Deep Learning based Indian Sign LanguageRecognition System",

[4].https://ieeexplore.ieee.org/abstract/document/8698006

[5].https://ijrpr.com/uploads/V2ISSUE9/IJRPR1329.pdf

[6].https://www.researchgate.net/figure/Sign-Language-Recognition-Prototype-diagram_fig1_262698351

[7].https://www.itmconferences.org/articles/itmconf/pdf/2021/05/itmconf_icacc2021_030 04.pdf

[8]. https://en.wikipedia.org/wiki/Long_short-term_memory

[9]. https://www.sciencedirect.com/topics/computer-science/sign-language-recognition

[10]. Joseph A. De Vito, Susan O'Rourke and Linda O'Neill, Human communication, New York:Longman, 2000.

[11]. Pradyumna Narayana, Ross Beveridge and Bruce A. Draper, "Gesture recognition: Focus on the hands", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5235-5244, 2018.

[12]. M. Al-Qurishi, T. Khalid and R. Souissi, "Deep Learning for Sign Language Recognition: Current Techniques Benchmarks and Open Issues", *IEEE Access*, vol. 9, pp. 126917-126951, 2021.

# APPENDIX

## Source Code:

### Data Collection :

In [1]:
```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
```

In [2]:
```
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
```

In [3]:
```
def mediapipe_detection(image,model):
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
    return image,results
```

In [4]:
```
def draw_styled_landmarks(image,results):
    mp_drawing.draw_landmarks(image,results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )

    mp_drawing.draw_landmarks(image,results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )

    mp_drawing.draw_landmarks(image,results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=1, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=1, circle_radius=2)
                              )

    mp_drawing.draw_landmarks(image,results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=1, circle_radius=1)
                              )
```

In [5]:
```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() i
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() i
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if re
    return np.concatenate([pose,face,lh,rh])
```

In [6]:
```
DATA_PATH = os.path.join('MP_Data')
#Actions
actions = np.array(['hello','thanks','iloveyou'])
#30 videos worth of data
no_sequences = 30
#30 frames
sequence_length = 30
```

In [7]:
```
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH,action,str(sequence)))
        except:
            pass
```

***Collect Key-point Values for Train & Test***

In [13]:
```
cap = cv2.VideoCapture(0) #to access webcam device

# set mediapipe model
with mp_holistic.Holistic(min_detection_confidence = 0.5, min_tracking_confidence = 0.5) as holistic:
    #NEW LOOP
    for action in actions: # Loop through actions
        for sequence in range(no_sequences): # Loop through sequences/videos
            for frame_no in range(sequence_length): # Loop through video length

                # read frames
                ret, frame = cap.read()

                # make detections
                image, results = mediapipe_detection(frame, holistic)

                # draw landmarks
                draw_styled_landmarks(image, results)

                # NEW collection wait logic
                if frame_no == 0:
                    cv2.putText(image, 'COLLECTING NOW...', (120, 200),
                                cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 3, cv2.LINE_AA)
                    cv2.putText(image, 'Collecting frames for Action: {} & Video: {}'.format(action, sequence), (15, 12),
                                cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1, cv2.LINE_AA)
                    cv2.imshow('OpenCV Feed', image) # show image on screen
                    cv2.waitKey(1500) # 1.5 seconds break
                else:
                    cv2.putText(image, 'Collecting frames for Action: {} & Video: {}'.format(action, sequence), (15, 12),
                                cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1, cv2.LINE_AA)
                    cv2.imshow('OpenCV Feed', image) # show image on screen

                # NEW extract key-points
                keypoints = extract_keypoints(results)
                npy_path = os.path.join(data_path, action, str(sequence), str(frame_no))
                np.save(npy_path, keypoints)

                # break gracefully if hit 'q' on keyboard
                if cv2.waitKey(10) & 0xFF == ord('q'):
                    break
    cap.release()           # release webcam
    cv2.destroyAllWindows() # close down all frames
```

## Model Training :

In [1]:
```
import os
os.getcwd()
os.chdir('C:\\Users\\Anitha\\Desktop\\SLD-Folder')
```

In [2]:
```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import mediapipe as mp
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
```

In [3]:
```
import pyttsx3
engine=pyttsx3.init()
```

In [4]:
```
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils

def mediapipe_detection(image,model):
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
    return image,results
```

In [5]:
```
def draw_styled_landmarks(image,results):
    mp_drawing.draw_landmarks(image,results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )

    mp_drawing.draw_landmarks(image,results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )

    mp_drawing.draw_landmarks(image,results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=1, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=1, circle_radius=2)
                              )

    mp_drawing.draw_landmarks(image,results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=1, circle_radius=1)
                              )
```

In [6]:
```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() i
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() i
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if r
    return np.concatenate([pose,face,lh,rh])
```

In [7]:
```
colors = [(245,117,16),(117,245,16),(16,117,245)]
def prob_viz(res,actions,input_frame,colors):
    output_frame = input_frame.copy()
    for num,prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40),colors[num], -1)
        cv2.putText(output_frame,actions[num],(0,85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255),
    return output_frame
```

In [8]:
```
DATA_PATH = os.path.join('MP_Data')
#Actions
actions = np.array(['hello','thanks','iloveyou'])
#30 videos worth of data
no_sequences = 30
#30 frames
sequence_length = 30
```

In [9]:
```
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH,action,str(sequence)))
        except:
            pass
```

In [10]:
```
label_map = {label:num for num, label in enumerate(actions)}
label_map
```

Out[10]:
```
{'hello': 0, 'thanks': 1, 'iloveyou': 2}
```

In [11]:
```
#label_map = {label:num for num, label in enumerate(actions)}
sequences, labels = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH,action, str(sequence),"{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

In [12]:
```
X = np.array(sequences)
y = to_categorical(labels).astype(int)

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.05)

log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir = log_dir)

model = Sequential()
model.add(LSTM(64,return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128,return_sequences=True, activation = 'relu'))
model.add(LSTM(64, return_sequences = False,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation = 'relu'))
model.add(Dense(actions.shape[0],activation='softmax'))

res = [.2,0.7,.01]
print(actions[np.argmax(res)])

model.compile(optimizer = 'Adam',loss='categorical_crossentropy',metrics=['categorical_accuracy'])
```

```
thanks
```

In [13]:
```
model.fit(x_train,y_train,epochs = 1000, callbacks=[tb_callback])
model.summary()
```

```
3/3 [==============================] - 0s 103ms/step - loss: 1.2962 - categorical_accuracy: 0.3294
Epoch 4/1000
3/3 [==============================] - 0s 109ms/step - loss: 1.2062 - categorical_accuracy: 0.3176
Epoch 5/1000
3/3 [==============================] - 0s 106ms/step - loss: 1.1337 - categorical_accuracy: 0.3059
Epoch 6/1000
3/3 [==============================] - 0s 102ms/step - loss: 1.1246 - categorical_accuracy: 0.2941
Epoch 7/1000
3/3 [==============================] - 0s 109ms/step - loss: 1.1055 - categorical_accuracy: 0.3176
Epoch 8/1000
3/3 [==============================] - 0s 99ms/step - loss: 1.0855 - categorical_accuracy: 0.3765
Epoch 9/1000
3/3 [==============================] - 0s 102ms/step - loss: 1.0695 - categorical_accuracy: 0.3765
Epoch 10/1000
3/3 [==============================] - 0s 98ms/step - loss: 1.0506 - categorical_accuracy: 0.4941
Epoch 11/1000
3/3 [==============================] - 0s 98ms/step - loss: 1.0149 - categorical_accuracy: 0.6000
Epoch 12/1000
3/3 [==============================] - 0s 98ms/step - loss: 0.9640 - categorical_accuracy: 0.6471
```

In [14]:
```
#actions[np.argmax(res)]
#model.compile(optimizer = 'Adam',loss='categorical_crossentropy',metrics=['categorical_accuracy'])
#actions[np.argmax(res[1])]

model.load_weights('action.h5')

#New Detection Variables
sequence = []
sentence = []
threshold = .4
```

In [19]:

```
cap = cv2.VideoCapture(0)
#Mediapipe Model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        #Read Feed
        ret, frame = cap.read()

        #Make detections
        image,results = mediapipe_detection(frame,holistic)

        #Draw Styled Landmarks
        draw_styled_landmarks(image,results)

        #Prediction Logic
        keypoints = extract_keypoints(results)
        sequence.insert(0,keypoints)
        sequence = sequence[:30]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence,axis=0))[0]
            print(actions[np.argmax(res)])

        #Visualization
        if res[np.argmax(res)] > threshold:
            if len(sentence) > 0:
                if actions[np.argmax(res)] != sentence[-1]:
                    sentence.append(actions[np.argmax(res)])
            else:
                sentence.append(actions[np.argmax(res)])


        if len(sentence)>5:
            sentence = sentence[-5:]


        #Viz probability
        image = prob_viz(res,actions,image,colors)


        cv2.rectangle(image,(0,0),(640,40),(245,117,16),-1)
        cv2.putText(image, ' '.join(sentence),(3,30),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255),2,cv2.LINE_AA)

        #Show to Screen
        cv2.imshow('OpenCV Feed', image)

        #Breaking the Feed
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 53ms/step
thanks
1/1 [==============================] - 0s 25ms/step
thanks
1/1 [==============================] - 0s 27ms/step
thanks
1/1 [==============================] - 0s 25ms/step
thanks
1/1 [==============================] - 0s 25ms/step
thanks
1/1 [==============================] - 0s 33ms/step
thanks
1/1 [==============================] - 0s 36ms/step
thanks
1/1 [==============================] - 0s 29ms/step
hello
1/1 [==============================] - 0s 32ms/step
hello
1/1 [==============================] - 0s 28ms/step
```

## Source code:

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
mp_holistic = mp.solutions.holistic      # holistic model-downloading the model and leveraging it to make detections
mp_drawing = mp.solutions.drawing_utils # drawing utilities-It makes easier to actually draw the key-points on our face.
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # convert color space from BGR to RGB
    image.flags.writeable = False                  # image is no longer writeable
    results = model.process(image)                 # predictions
    image.flags.writeable = True                   # image is writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # convert color space from RGB to BGR
    return image, results


def draw_styled_landmarks(image, results):
    # face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS,
                              mp_drawing.DrawingSpec(color=(255, 255, 86), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(255, 255, 255), thickness=1, circle_radius=1))

    # pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(255, 86, 170), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(86, 255, 255), thickness=2, circle_radius=2))

    # left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(0, 127, 255), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(255, 255, 255), thickness=2, circle_radius=2))

    # right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(255, 255, 255), thickness=2, circle_radius=2))


cap = cv2.VideoCapture(0)  # read the feed from webcam device

# set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():  # double-check for webcam access & loop through all frames

        # read feed/frames from webcam
        ret, frame = cap.read()

        # make detections
        image, results = mediapipe_detection(frame, holistic)

        # draw formatted landmarks
```

```
        draw_styled_landmarks(image, results)

        # show image to screen
        cv2.imshow('OpenCV Feed', image)

        # break gracefully if hit 'q' on keyboard
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()  # release webcam
    cv2.destroyAllWindows()  # close down all frames
draw_styled_landmarks(frame, results) # apply landmarks
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)) # color conversion
results.pose_landmarks


def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]).flatten()
\
        if results.pose_landmarks else np.zeros(33 * 4)

    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() \
        if results.face_landmarks else np.zeros(468 * 3)

    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() \
        if results.left_hand_landmarks else np.zeros(21 * 3)

    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() \
        if results.right_hand_landmarks else np.zeros(21 * 3)

    return np.concatenate([pose, face, lh, rh])
extract_keypoints(results)
extract_keypoints(results).shape
data_path = os.path.join('MP_Data') #path for exported data

actions = np.array(['HELLO', 'THANKS', 'I LOVE U','YES','NO','PLEASE','GOOD
BYE','SORRY','YOU ARE WELCOME','FAMILY,HOUSE','LOVE']) #actions to be detected

no_sequences = 30 #no. of videos collected for each action

sequence_length = 30 #frame length of each video
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(data_path, action, str(sequence)))
        except:
            pass
cap = cv2.VideoCapture(0)  # to access webcam device

# set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    # NEW LOOP
    for action in actions:  # loop through actions
        for sequence in range(no_sequences):  # loop through sequences/videos
            for frame_no in range(sequence_length):  # loop through video length

                # read frames
                ret, frame = cap.read()

                # make detections
```

```
            image, results = mediapipe_detection(frame, holistic)

            # draw landmarks
            draw_styled_landmarks(image, results)

            # NEW collection wait logic
            if frame_no == 0:
               cv2.putText(image, 'COLLECTING NOW...', (120, 200),
                        cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 3, cv2.LINE_AA)
               cv2.putText(image, 'Collecting frames for Action: {} & Video: {}'.format(action, sequence),
                        (15, 12),
                        cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1, cv2.LINE_AA)
               cv2.imshow('OpenCV Feed', image)  # show image on screen
               cv2.waitKey(1500)  # 1.5 seconds break
            else:
               cv2.putText(image, 'Collecting frames for Action: {} & Video: {}'.format(action, sequence),
                        (15, 12),
                        cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1, cv2.LINE_AA)
               cv2.imshow('OpenCV Feed', image)  # show image on screen

            # NEW extract key-points
            keypoints = extract_keypoints(results)
            npy_path = os.path.join(data_path, action, str(sequence), str(frame_no))
            np.save(npy_path, keypoints)

            # break gracefully if hit 'q' on keyboard
            if cv2.waitKey(10) & 0xFF == ord('q'):
               break
   cap.release()  # release webcam
   cv2.destroyAllWindows()  # close down all frames
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
label_map = {label:num for num, label in enumerate(actions)}
label_map
sequences, labels = [], [] # blank arrays
for action in actions:
   for sequence in range(no_sequences):
      window = [] # blank array
      for frame_num in range(sequence_length):
         res = np.load(os.path.join(data_path, action, str(sequence), "{}.npy".format(frame_num)))
         window.append(res)
      sequences.append(window)
      labels.append(label_map[action])
np.array(sequences).shape
np.array(labels).shape
X = np.array(sequences)
y = to_categorical(labels).astype(int) # one-hot-encoding
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
model = Sequential()

# add 3 set of LSTM layers
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30, 1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
```

```
# add 3 Dense layers
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax')) # actions layer
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
model.fit(X_train, y_train, epochs=1000, callbacks=[tb_callback])
model.summary()
res = model.predict(X_test)#Now, we will make the predictions on the test data based on this model.
actions[np.argmax(res[2])]#We can check the prediction for a random ac
actions[np.argmax(y_test[2])]
model.save('fin_mod.h5')
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
yhat = model.predict(X_test)
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
multilabel_confusion_matrix(ytrue, yhat)
accuracy_score(ytrue, yhat)
yhat_t = model.predict(X_train)
ytrue_t = np.argmax(y_train, axis=1).tolist()
yhat_t = np.argmax(yhat_t, axis=1).tolist()
multilabel_confusion_matrix(ytrue_t, yhat_t)
accuracy_score(ytrue_t, yhat_t)
# render probabilites
colors = [(245, 117, 16), (117, 245, 16), (16, 117, 245), (255,0,0), (255,153,0),(255,255,0),(128,0,128),(0,255,255),(128,0,0),(127,255,212),(75,0,130),(250,0,0)]
def prob_viz(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0, 60+num*35), (int(prob*100), 90+num*40), colors[num], -1) # drawing a dynamic rectangle
        cv2.putText(output_frame, actions[num], (0, 85+num*35), cv2.FONT_HERSHEY_PLAIN, 1, (255, 255, 255), 1, cv2.LINE_AA)
    return output_frame


# 1. new detection variables
sequence = []
sentence = []
predictions = []
threshold = 0.7

cap = cv2.VideoCapture(0)  # to access webcam device

# set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():  # loop through all frames

        # read frames
        ret, frame = cap.read()

        # make detections
        image, results = mediapipe_detection(frame, holistic)

        # draw formatted landmarks
        draw_styled_landmarks(image, results)
        # ----------------------------------------------------------------------
        # 2. prediction logic
        keypoints = extract_keypoints(results)  # extracting the keypoints
```

```
        sequence.append(keypoints)
        sequence = sequence[-30:]  # grab last 30 frames

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            #           print(actions[np.argmax(res)])
            predictions.append(np.argmax(res))
            # ------------------------------------------------------------------------
            # 3. visualisation logic
            if np.unique(predictions[-10:])[0] == np.argmax(res):

                if res[np.argmax(res)] > threshold:

                    if len(sentence) > 0:  # check for words in sentence
                        if actions[np.argmax(res)] != sentence[-1]:
                            sentence.append(actions[np.argmax(res)])
                    else:
                        sentence.append(actions[np.argmax(res)])

            if len(sentence) > 8:
                sentence = sentence[-8:]
            # ---------------------q-------------------------------------------------
            # 4. visualization probabilites
            image = prob_viz(res, actions, image, colors)
        # ------------------------------------------------------------------------

        cv2.rectangle(image, (0, 0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3, 30), cv2.FONT_HERSHEY_PLAIN, 1, (255, 255, 255), 2,
cv2.LINE_AA)

        # show image on screen
        cv2.imshow('OpenCV Feed', cv2.resize(image, (600, 500)))

        # break gracefully if hit 'q' on keyboard
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()  # release webcam
    cv2.destroyAllWindows()  # close down all frames
```

**Rubrics for Project**

| Focus Areas | CNo | Criterion [c] | Exemplary 4 | Satisfactory 3 | Developing 2 | Unsatisfactory 1 |
|---|---|---|---|---|---|---|
| **Problem Formulation (PO1,PO2, PO6, PO7,PSO1)** | I | **Identify/Define Problem** Ability to identify a suitable problem and define the project objectives. | Demonstrates a skillful ability to identify / articulate a problem and the objectives are well defined and prioritized. | Demonstrates ability to Identify / articulate a problem and All major objectives are identified. | Demonstrates some ability to identify / articulate a problem that is partially connected to the issues and most major objectives are identified but one or two minor ones are missing or priorities are not established. | Demonstrates minimal or no ability to identify / articulate a problem and many major objectives are not identified. |
| | II | **Collection of Background Information:** Ability to gather background Information (existing knowledge, research, and/or indications of the problem) | Collects sufficient relevant back ground information from appropriate sources, and is able to identify pertinant/critical information; | Collects sufficient relevant background information from appropriate sources; | Collects some relevant backgrou nd informati on from appropriat e Sources. | Minimal or no ability to collect relevant background inf ormation |
| | III | **Define scope of the problem** Ability to identify | Demonstrates a skillful ability to define the scope of | Demonstrates ability to define problem scope | Demonstrates some ability to define problem | Demonstrates minimal or no ability to define problem scope and fails to |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | problem scope suitable to the degree considering the impact on society and environment | problem accurately mentioning the relevant fields of engineering precisely. Considers, explains and evaluates the impact of engineering interventions on society and environment. | mentioning the relevant fields of engineering broadly. Considers and explains the impact of engineering interventions on society and environment | scope mentioning some of the relevant fields . Some considerati on of the impact of engineerin g interventio ns on society and environme nt. | mention relevant fields of engineering. Minimal or no consideration of the impact of engineering interventions on society and environment |
| **Project Design (PO3,PS O1)** | IV | **Understandi ng the Design Process and Problem Solving:** Ability to explain the design process including the importance of needs, specifications , concept generation and to develop an approach to solve a problem. | Demonstrates a comprehensiv e ability to understand and explain a design process. Considers multiple approaches to solving a problem, and can articulate reason for choosing solution | Demonstrates an ability to understand and explain a design process. Considers multiple appr oaches to solving a problem, which is justified and considers consequences . | Demonstra tes some ability to understand and explain a design process. Considers a few approaches to solving a problem; doesn't always consider consequen ces. | Demonstrates minimal or no ability to understand and explain a design process. Considers a single approach to solving a problem. Does not consider consequences. |
| **Build (PO4,PO 5, PSO1)** | V | **Implementin g Design Strategy and Evaluating Final Design:** Ability to execute a solution taking into consideration design requirements using appropriate | Demonstrates a skillful ability to execute a solution taking into consideration all design requirements using the most relevant tool. | Demonstrates an ability to execute a solution taking into consideration design requirements using relevant tool. | Demonstra tes some ability to execute a solution but not using most relevant tool. | Demonstrates minimal or no ability to execute a solution. Solution does not directly attend to the problem. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | tool (software/hardware); | | | | |
| **Test & Deploy (PO4, PO5, PSO1)** | VI | To evaluate/confirm the functioning of the final design. To deploy the project on the target environment | Demonstrates a skillful ability to evaluate/confirm the functioning of the final design skillfully, with deliberation for further Improvement after deployment. | Demonstrates an ability to evaluate/confirm the functioning of the final design. The evaluation is complete and has sufficient depth. | Ability to evaluate/confirm the functioning of the final design, but the evaluation lacks depth and/or is incomplete. | Demonstrates minimal or no ability to evaluate/confirm the functioning of the final design. |
| **Ethical responsibility (PO8)** | VII | **Proper Use of Others' Work:** Ability to recognize, understand and apply proper ethical use of intellectual property, copyrighted materials, and research. | Always recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research. | Recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research. | Some recognition and application of proper ethical use of intellectual property, copyrighted materials, and others' research. | Minimal or no recognition and/or application of proper ethical use of intellectual property, Copyrighted materials, or others' research. |
| **Team Skills (PO9)** | VIII | **Individual Work Contributions and Time Management:** Ability to carry out individual Responsibilities and manage time (estimate, prioritize, establish deadlines/ milestones, follow timeline, plan | Designated jobs are accomplished by deadline; completed work is carefully and meticulously prepared and meets all requirements. | Designated jobs are accomplished by deadline; completed work meets requirements. | Designated jobs are accomplished by deadline; completed work meets most requirements. | Some Designated jobs are accomplished by deadline; completed work meets some requirements. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | for contingencies, adapt to change). | | | | |
| | IX | **Leadership Skills**: Ability to lead a team. (i) Mentors and accepts mentoring from others. (ii) Demonstrates capacity for initiative while respecting others' roles. (iii) Facilitates others' involvment. (iv) Evaluates team Effectiveness and plans for improvements | Exemplifies leadership skills. | Demonstrates leadership skills. | Demonstrates some leadership skills at times. | Demonstrates minimal or no leadership skills. |
| | X | **Working with Others:** Ability to listen to, collaborate with, and champion the efforts of others. | Skillfully listens to, collaborates with, and champions the efforts of others. | Listens to, collaborates with, and champions the efforts of others. | Sometimes listens to, collaborates with, and champions others' efforts. | Rarely listens to, collaborates with, or champions others' efforts. |
| **Project Presentation (P10)** | XI | **Technical Writing Skills** Ability to communicate the main idea with clarity. Ability to use illustrations properly to support ideas (citations, position on page etc) | Main idea is clearly and precisely stated. Materials are seamlessly arranged in a logical sequence Illustrations are skillfully used to support ideas | Main idea is understandable. Material moves logically forward, Illustrations are properly used to support ideas | Main idea is somewhat understandable. Matericals has some logical order and is somewhat coherent or easy to | Main idea is difficult to understand. Material has little logical order, and is often unclear, inconherent. Illustrations are used, but minimally support ideas. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | follow. Illustrations are for the most part properly used to support ideas | (not properly cited etc) |
| | XII | **Communication Skills for Oral Reports** Ability to present strong key ideas and supporting details with clarity and concision. Maintain contact with audience, and ability to complete in the allotted time | Presentation logically and skillfully structured. Key ideas are compelling, and articulated with exceptional clarity and concision. Introduction, supporting details and summary are clearly evident and memorable, and ascertain the credibility of the speaker Presentation fits perfectly within time constraint. | Presentation has clear structure and is easy to follow. Key ideas are clearly and concisely articulated, and are interesting. There is sufficient detail to ascertain speaker's authority, and presentation includes an introduction and summary. Presentation fits within time constraint, though presenter might have to subtly rush or slow down. | Presentation has some structure. Key ideas generally identifiable, although not very remarkable. Introduction, supporting details and/or summary may be too broad, too detailed or missing. Credibility of the speaker may be questionable at times. Presentation does not quite fit within time constraint; presenter has to rush or slow down at end | Presentation rambles. Not organized; key ideas are difficult to identify, and are unremarkable. No clear introduction, supporting details and summary. Speaker has no credibility. Presentation is unsuitablably short or unreasonably long. |
| **Project management (PO11,PSO2)** | XIII | Use of software project management principles and tools | Employ all the appropriate tools or engineering techniques. | Employ the appropriate tools or engineering techniques | Employ some management tools | Does not make use of any tool |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | (versioning, time schedules etc ) | Clearly demonstrates mastery of several areas of the curriculum. | | | |
| **Lifelong Learning (PO12,PSO2)** | XIV | **Extend Scope of Work:** Ability to extend the project through implementation in other study areas | Demonstrates a skillful ability to explore a subject/topic thoroughly, discusses the road map to extend the project in other areas. | Demonstrates an ability to explore a subject/topic, and shows possible areas in which project can be extended | Demonstrates some ability to explore a subject/topic, providing some knowledge of areas in which project can be extended | Demonstrates minimal or no ability to explore a subject/topic, and does not discuss future work clearly mentioning other areas |

**Table 3 : Rubrics for Project**

## 111. Rubrics Evaluation:

| | **PO/ PSO** | PO1,PO2, PO6, PO7, PSO1 | | | PO 3, PS O1 | PO4, PO5, PSO1 | PO4, PO5, PSO1 | P O8 | PO9 | | | PO10 | | PO 11, PS O2 | PO 12 ,PS O2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Batch** | **Rubrics** | R1 | | | R2 | R3 | R4 | R5 | R6 | | | R7 | | R8 | R9 |
| | **Roll No** | C I | C II | CI II | CI V | CV | CVI | C VI I | CV III | CI X | C X | C X I | C XI I | CX III | CX IV |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

**Table 4 : Rubrics Evaluation**