SIGN LANGUAGE RECOGNITION

A Project Report Submitted in partial fulfillment of the requirement for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND SYSTEMS ENGINEERING

Submitted by

ARUGULA SISMAI	**318114110001**
ASRITHA DIDDI	**318114110002**
CHAVALI ANUSHA	**318114110003**
CHINTA SUNDARA SREYA	**318114110004**

Under the esteemed guidance of

Prof. B. PRAJNA

Head of the Department of Computer Science and System Engineering

DEPARTMENT OF COMPUTER SCIENCE

& SYSTEMS ENGINEERING

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN

VISAKHAPATNAM

CERTIFICATE

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN

VISAKHAPATNAM

This is to certify that this main project entitled **"SIGN LANGUAGE RECOGNITION"** is a Bonafede work carried out by **ARUGULA SISMAI (318114110001), ASRITHA DIDDI (318114110002), CHAVALI ANUSHA (318114110003), CHINTA SUNDARA SREYA (318114110004)** submitted in partial fulfillment of the requirements for the award of Degree of Bachelor Technology in Computer Science and Systems Engineering during March 2022 to May 2022.

Project Guide

Prof. B. PRAJNA
Head of the Department
Department of CS & SE,
AUCEW

Head of the department

Prof. B. PRAJNA
Head of the Department
Department of CS & SE,
AUCEW

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved guide **Prof. B. PRAJNA**, Head of the Department, Department of Computer Science and Systems Engineering, Andhra University College Of Engineering For Women for the valuable guidance and suggestions, keen interest, and encouragement extended throughout project work.

We express our deep sense of gratitude to our beloved **Prof. B. PRAJNA** Madam, Andhra University College of Engineering For Women for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved principal **Prof. M. Bhatti** Madam for permitting us to carry out this project.

We consider ourselves lucky enough to get such a project. This project would add an asset to our academic profile.

We express our thanks to all those who contributed to the successful completion of our project work.

With gratitude,

ARUGULA SISMAI (318114110001),

ASRITHA DIDDI (318114110002),

CHAVALI ANUSHA (318114110003),

CHINTA SUNDARA SREYA (318114110004).

DECLARATION

We hereby declare that project entitled "**SIGN LANGUAGE RECOGNITION"** is an authenticated record of our work carried out at **ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN, VISAKHAPATNAM** as requirements of the project for the award of Degree of **Bachelor of Technology (COMPUTER SCIENCE AND SYSTEMS ENGINEERING)**.

We also hereby declare that this project is the result of our effort and that it has not been submitted to any other university for the award of any Degree.

ARUGULA SISMAI

(318114110001),

ASRITHA DIDDI

(318114110002),

CHAVALI ANUSHA

(318114110003),

CHINTA SUNDARA SREYA

(318114110004)

INDEX

ABSTRACT

Sign language is the only tool of communication for the person who is not able to speak and hear anything. Sign language is a boon for verbally challenged people to express their thoughts and emotion. In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabet and gestures in sign language. With the help of computer vision and neural networks, we can detect the signs and give the respective text output.

The focus of this project is to develop a system to aid verbally challenged people. In this work, we present a way to classify sign gestures using a video as input and text as the output. The video is divided into many frames and the person performing the gesture is identified as a subject. After this, the background is eliminated and detection of the hand of the subject is performed in every frame of the given video. Upon identifying the hand region of each frame, the movement of the hand is tracked. This motion data is then aggregated and represented in the form of interval-valued data. A suitable interval-based classifier is generated based on this information. Testing can then be performed to obtain the efficiency of the given system. The given testing input is checked to be within the range of given interval values and then confirmed to be a certain gesture.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing-impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency. Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word's meaning. Fingerspelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry meaning in the word-level association. Despite this, fingerspelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication. A system for sign language recognition that classifies fingerspelling can solve this problem. Various machine learning algorithms are used and their accuracies are recorded and compared in this report.

SIGN LANGUAGE

It is a language that includes gestures made with the hands and other body parts, including facial expressions and postures of the body. It is used primarily by people who are deaf and dumb. There are many different sign languages such as British, Indian and American sign languages. British sign language (BSL) is not easily intelligible to users of American Sign Language (ASL) and vice versa. A functioning signing recognition system could provide a chance for inattentive communication with non-signing people without the necessity for an interpreter. It might be wont to generate speech or text making

the deaf more independent. Unfortunately, there has not been any system with these capabilities thus far. During this project, we aim to develop a system that may classify signing accurately. American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing, and is used by many hearing people as well.

SIGN LANGUAGE AND HAND GESTURE RECOGNITION

The process of converting the signs and gestures shown by the user into text is called sign language recognition. It bridges the communication gap between people who cannot speak and the general public. Image processing algorithms along with neural networks are used to map the gesture to appropriate text in the training data and hence raw images/videos are converted into respective text that can be read and understood. Dumb people are usually deprived of normal communication with other people in society. It has been observed that they find it difficult at times to interact with normal people with their gestures, as only a very few of those are recognized by most people. Since people with hearing impairment or deaf people cannot talk like normal people, they have to depend on some sort of visual communication most of the time. Sign Language is the primary means of communication in the deaf and dumb community. As with any other language it has also got grammar and vocabulary but uses visual modality for exchanging information. The problem arises when dumb or deaf people try to express themselves to other people with the help of these sign language grammar. This is because normal people are usually unaware of this grammar. As a result, it has been seen that communication of a dumb person is only limited within his/her family or the deaf community. The importance of sign language is emphasized by the growing public approval and funds for international projects. In this age of technology, the demand for a computer-based system is highly demanding for the dumb community. However, researchers have been attacking the problem for quite some time now and the results are showing some promise. Interesting technologies are being developed for speech recognition but no real commercial product for sign recognition is there in the current market. The idea is to make computers understand human language and develop user-friendly Human-Computer

Interfaces (HCI). Making a computer understand speech, facial expressions and human gestures are some steps toward it. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. The project aims to determine human gestures by creating a Human-Computer Interface (HCI). Coding these gestures into machine language demands a complex programming algorithm. In our project, we are focusing on Image Processing and Template matching for better output generation.

1.2 PROJECT DELIVERABLES

- ➢ Project Information
- ➢ Project Documentation
- ➢ Proposed System
- ➢ Requirements List
- ➢ Program

1.3 PROJECT SCOPE

Sign language has been used by people who suffer from hearing loss. These people express themselves with a specific type of language which is called “Sign Language”. Sign language is like the other languages, each region has its specific sign language like normal language. The common thing in sign language is that; each expression is stated with fingers and some arm movement.

The interest in sign language has been increasing with growing media. Today each TV series, news, and movie have been using sign language to reach deaf people or people who communicate with sign language. Thus, sign language stuff and similar applications have been being used widely nowadays.

Sign Language Recognition has been provided to convert sign language to written language faster, safely and without human beings. Since it is not technically feasible to convert all media on the internet to written language from sign language, it will be useful to use an algorithm that does the same thing.

The software that will be prepared may use the dataset that is available on the internet but also it will be needed to generate some datasets. Sign language is used by many people, so it contains small differences for every person, the software should understand the difference and ignore it by machine learning algorithms.

The objectives of the project are:

- To convert the sign language that is perceived by the camera to written language.
- To do the conversion as much as faster.

2. LITERATURE REVIEW

This chapter presents the research papers studied to carry out this project work

LITERATURE REVIEW ON HAND GESTURE DETECTION

In the past decades, gestures were usually identified and judged by wearing data gloves to obtain the angles and positions of each joint in the gesture Several papers and projects have targeted the issue of hand gesture recognition. However, it is difficult to use widely due to the cost and inconvenience of wearing the sensor. In contrast, the non-contact visual inspection methods have the advantage of low cost and comfort for the human body, which are the currently popular gesture recognition method. Chakraborty proposed the skin color models utilizing image pixel distribution in a given color space, which can significantly improve the detection accuracy in the presence of varying illumination conditions. However, it was difficult to achieve the desired results using the model-based methods because of the light sensitivity during the imaging process. The algorithm-based non-contact visual inspection methods were also used to conduct the gesture recognition, such as the hidden Markov model the particle filter, and Heer features AdaBoost learning algorithm; however, it is difficult to execute real-time due to the complicated algorithms. The above results cannot acquire gestures efficiently in real-time since only the insufficient 2D image information was used. Therefore, it is inevitable that gesture recognition by 2D images is replaced by 3D with depth information.

By using Hand gestures users can communicate more information in less period. So, to improve the interface between users and computers human computers interaction (HCI) technology has great utilization. The OpenCV itself has played a role in the growth of computer vision by enabling thousands of people to do more productive work in vision.

LITERATURE REVIEW ON MEDIAPIPE

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann, in their paper "MediaPipe: A Framework for Building Perception Pipelines", MediaPipe is a framework for building pipelines to perform inference over arbitrary sensory data. With MediaPipe, a perception pipeline can be built as a graph of modular components, including model inference, media processing algorithms and data transformations, etc. Sensory data such as audio and video streams enter the graph, and perceived descriptions such as object-localization and face landmark streams exit the graph.

MediaPipe is designed for machine learning (ML) practitioners, including researchers, students, and software developers, who implement production-ready ML applications, publish code accompanying research work, and build technology prototypes. The main use case for MediaPipe is rapid prototyping of perception pipelines with inference models and other reusable components. MediaPipe also facilitates the deployment of perception technology into demos and applications on a wide variety of different hardware platforms. MediaPipe enables incremental improvements to perception pipelines through its rich configuration language and evaluation tools. MediaPipe addresses these challenges by abstracting and connecting individual perception models into maintainable pipelines. All of the steps necessary to infer from the sensory data and get the perceived results are specified in the pipeline configuration. It is easy to reuse MediaPipe components in different pipelines across successive applications as these components share a common interface oriented around time-series data. Each pipeline can then run with the same behavior on a variety of platforms, enabling the practitioner to develop the application on workstations, and then deploy it on mobile, for example.

MediaPipe consists of three main parts:

(a) a framework for inference from sensory data,

(b) a set of tools for performance evaluation, and

(c) a collection of re-usable inference and processing components called calculators

LITERATURE REVIEW ON OpenCV

OpenCV (Open Source Computer Vision Library) is an Application Peripheral Interface (API) developed by Intel that can be used for many image processing and computer vision applications. OpenCV officially launched in 1999 and the project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls.

OpenCV library is a collection of algorithms and C/C++ functions and a few classes that implement some Image processing and computer vision algorithms. There is active development on interfaces for Python, Ruby, MATLAB and other languages. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. OpenCV is written in optimized C and can take advantage of multicore processors. If there is a need to further automatic optimization on Intel architecture that helps, we need Intel's Integrated Primitives (IPP) libraries which consist of low-level optimized routines in many different algorithmic areas. OpenCV automatically uses the appropriate IPP library at runtime if that library is installed. OpenCV contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision and robotics.

The principles behind the creation of the library are to aid commercial uses of computer vision in human-computer interface, robotics, monitoring, biometrics and security by providing a free and open infrastructure where the distributed efforts of the vision community can be consolidated and performance optimized. OpenCV support for vision is extensive, including routine support for input, display, and storage of movies and single images. OpenCV uses DirectX, which is a set of APIs developed by Microsoft for creating multimedia applications and games.

3. PROBLEM ANALYSIS

3.1 EXISTING SYSTEM:

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

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

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

3.2 PROPOSED SYSTEM:

Our proposed system is a sign language recognition system using a Long Short-Term Memory (LSTM) neural network which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image is obtained and sent for comparison to the trained model. Thus, our system is more robust in getting exact text labels of letters.

3.3 ADVANTAGES:

- ➢ The dataset can easily be extended and customized according to the need of the user and can prove to be an important step toward reducing the gap in communication for dumb and deaf people.
- ➢ Using the sign detection model, meetings held at a global level can become easy for disabled people to understand and the value of their hard work can be given.

3.4 LIMITATIONS:

- ➢ They are costly and are difficult to be used commercially.
- ➢ Limitations such as environmental factors like low light intensity and uncontrolled background cause a decrease in the accuracy of the detection.

4. SYSTEM ANALYSIS

System analysis is a problem-solving technique that decomposes a system into its component pieces to study how well those component parts work and interact to accomplish their purpose. System analysis is the process of studying a procedure to identify its goals and purposes and create systems and procedures that will efficiently achieve them. The development of a computer-based information system includes a systems analysis phase which produces or enhances the data model which itself is a precursor to creating or enhancing a database. There are several different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

4.1 SYSTEM REQUIREMENT SPECIFICATION:

System requirements specification is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specifications permit a rigorous assessment of requirements before design can begin and reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

4.1.1 Functional Requirements:

Functional requirements define what a system is supposed to do. In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). How a system implements functional requirements is detailed in the system design. In some cases, a requirements analyst generates 17 use cases after gathering and validating a set of functional requirements. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

- **Installing and importing dependencies:** The application first installs all the required libraries i.e., TensorFlow, OpenCV, MediaPipe, Sklearn, Matplotlib and imports dependencies such as cv2, NumPy, os, pyplot and time from the installed libraries.
- **Extract Key points:** Using the MediaPipe python framework we detect and extract Hand, Face, and Pose Landmarks.
- **Setup folders for data collection:** Here we create folders to export the data to be stored as NumPy arrays.
- **Preprocess data and create labels:** Here we preprocess the data, and create labels and features for the collections which would be of help to detect the sign in real-time.
- **Build & Train an LSTM deep learning model:** Using TensorFlow and Keras we build and train the model where the model summary and accuracy are defined.
- **Save model weights:** Here we save and load models.
- **Test in real-time:** Here perform real-time sign language recognition using OpenCV.

4.1.2 Non-Functional Requirements:

In Systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behavior. This should be contrasted with functional requirements that define specific behavior or functions. In general, functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements," and "non-behavioural requirements." Qualities, that is, non-functional requirements, can be divided into two main categories: Execution qualities (which are observable at run time), Evolution qualities (which are embodied in the static structure of the software system).

Functionality: The application is designed to translate sign language into human language.

Reliability: The application will work properly on computers.

Security: Users are not allowed to reach the database directly, so data will be saved in this way. User information will not share with third-party applications.

Safety: Safety will be covered by using TCP/IP on the network side.

Performance: Translation should take less than 5 seconds.

Flexibility: Responsive web design (RWD) will be used in GUI design. RWD is an approach that makes web pages render well on a variety of devices and window or screen sizes.

Scalability: The application is designed to cover the needs of all users.

Portability: The system works on all operating systems.

Availability: As long as there is access to the internet, there will be access to the application without any interrupts.

Usability: GUI will be designed to make users comfortable and familiar with the application. Every type of human can easily use applications. GUI will be mapped according to that.

Maintainability: The application will be updated according to coming feedback from users and improving the word database of the application. The database will be backed up monthly in case of data loss.

4.2 FEASIBILITY STUDY:

The feasibility study is an evaluation and analysis of the potential of a proposed project. It is based on extensive investigation and research to support the process of decision-making. Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained

A well-designed feasibility study should provide a historical background of a project, a description of a service, and details of the operations. Generally, feasibility studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility:

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Scalability: Ability to process huge amounts of data.

Reliability: It is an efficient way of processing data without loss.

Benefits of Conducting a Feasibility Study:

Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of your idea. Below are the key benefits of conducting a feasibility study:

Gives project teams more focus and provides an alternative outline:

- ➢ Narrow the business alternatives.
- ➢ Identifies a valid reason to undertake the project.
- ➢ Enhances the success rate by evaluating multiple parameters.
- ➢ Aids decision-making on the project

Object-Oriented Analysis:

Object-Oriented Analysis is a popular technical approach for analyzing, and designing an application, system or business by applying the object-oriented paradigm and visual modeling throughout the development lifecycle to faster, better, stakeholder communication and product quality. In the case of object-oriented analysis, the process varies. But these two are identical in use case analysis.

The steps involved in the analysis phase are:

- ➢ Identify the actors.
- ➢ Classification-develops a static UML class diagram.
- ➢ Develop use cases.
- ➢ Identify classes, relationships, attributes, methods

4.3 SYSTEM REQUIREMENTS:

System requirements specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the system is to do, without making any commitment as to how the system is to do it.

A system requirements specification is normally produced in response to user requirements specifications or another expression of requirements and is then used as the basis for system design. The system requirements specification typically differs from the expression of requirements in both scope and precision the latter may cover both the envisaged system and the environment in which it will operate but may leave many broad concepts unrefined.

4.3.1 Software Requirements:

Software Requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Software Requirements for the present project:

- ➢ Operating System: Windows 7 and Above
- ➢ Language used: Python
- ➢ Libraries: OpenCV, TensorFlow, Keras, NumPy, MediaPipe

4.3.2 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL lists tested are compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

Hardware Requirements for the present project:

- ➢ Camera: Good quality,3MP
- ➢ Ram: Minimum 8GB or higher
- ➢ GPU: 4GB dedicated
- ➢ Processor: Intel Pentium 4 or higher
- ➢ HDD: 10GB or higher
- ➢ Monitor: 15” or 17” color monitor
- ➢ Mouse: Scroll or Optical
- ➢ Mouse or Touchpad
- ➢ Keyboard: Standard 110 keys keyboard

5. SYSTEM DESIGN

5.1 INTRODUCTION:

System design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements one could see it as the application of systems theory to product development. There is some overlap and synergy between the disciplines of systems analysis, systems architecture and systems engineering.

Systems design mainly concentrates on defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it considers all related variables of the system that needs to be created—from the architecture to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system.

Systems design then overlaps with systems analysis, systems engineering and systems architecture. The systems design approach first appeared right before World War II, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science.

The system design and implementation service provide the following capabilities:

- ➢ Design of technical architectures for new IT services.
- ➢ Performance and capacity analysis of planned and existing systems.
- ➢ System development.

A system approach to design asks:

- ➢ For this situation, what is the system?
- ➢ What is the environment?
- ➢ What goal does the system have about its environment?
- ➢ What is the feedback loop by which the system corrects its actions?
- ➢ How does the system measure whether it has achieved its goal?
- ➢ Who defines the system, environment, goal etc. and monitors it?
- ➢ What resources does the system have for maintaining the relationship it desires?
- ➢ Are its resources sufficient to meet its purpose?

5.2 UNIFIED MODELING LANGUAGE:

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic-semantic and pragmatic rules. A UML system is represented using five different views that describe the system from a distinctly different perspective. Each view is defined by a set of a diagram, which is as follows:

a) *USER MODEL VIEW:* This view represents the system from the user's perspective. The analysis representation describes a usage scenario from the end-user's perspective.

b) *STRUCTURAL MODEL VIEW:* In this model data and functionality are arrived from inside the system. This model view models the static structures.

c) *BEHAVIOURAL MODEL VIEW:* It represents the dynamic behaviour as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

d) *IMPLEMENTATION MODEL VIEW:* In this, the structural and behavioural parts of the system are represented as they are to be built.

e) *ENVIRONMENTAL MODEL VIEW:* In this, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

As the name implies, is a modeling language. It may be used to visualize, specify, construct, and document the artifacts of a software system. It provides a set of notations to create a visual model of the system. UML has been designed for a broad range of applications. Hence, it provides constructs for a broad range of systems and activities (e.g., distributed systems, analysis, system design, and deployment).

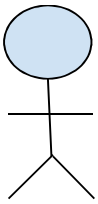
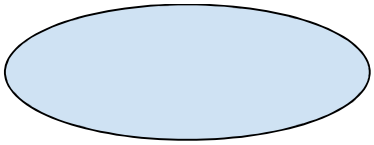
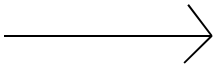
System development focuses on three different models of the system:

- The functional model, represented in UML with use case diagrams, describes the functionality of the system from the user's point of view.
- The object model, represented in 25 UML with class diagrams, describes the structure of the system in terms of objects, attributes, associations, and operations.
- The dynamic model, represented in UML with interaction diagrams, state machine diagrams, and activity diagrams, describes the internal behaviour of the system. Interaction diagrams describe behaviour as a sequence of messages exchanged among a set of objects, whereas state machine diagrams describe behaviour in terms of the states of an individual object and the possible transitions between states. Activity diagrams describe behaviour in terms of control and data flows.

5.2.1 USE CASE DIAGRAM:

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions that some systems should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Graphical Representation of Use Case Diagram

Actor	The actor represents users of a system, including human users and other systems.	
Use Case	A use case represents functionality or services provided by a system to users.	
Association	Associations between actors and use cases are indicated by solid lines. An association exists whenever an actor is involved with an interaction described by a use case	

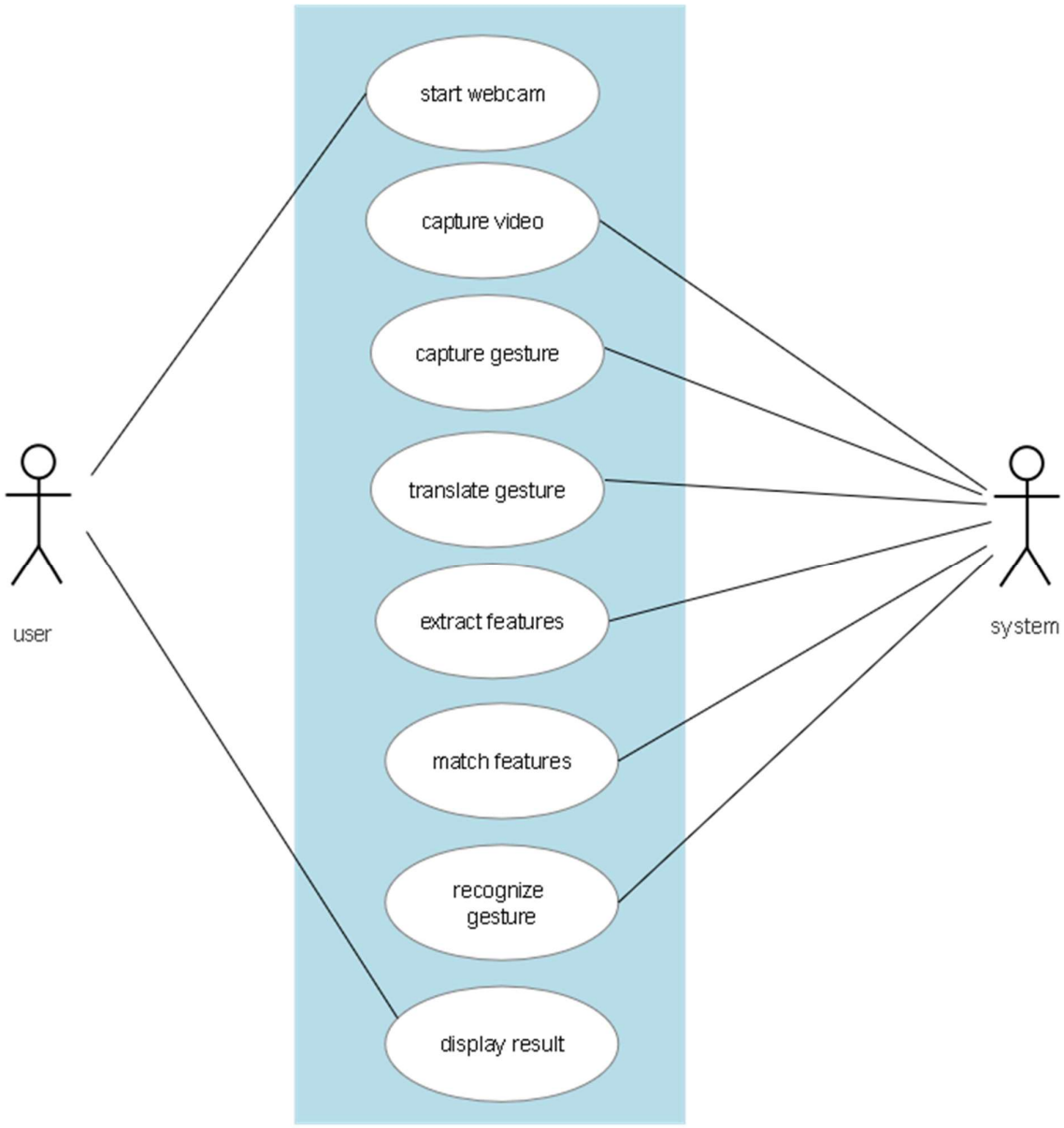
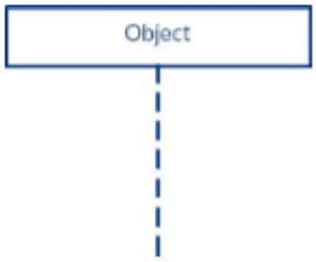
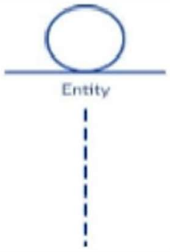
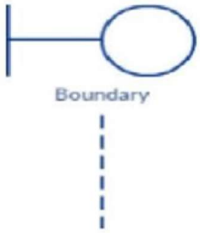


Figure 1: Use Case Diagram

5.2.2 SEQUENCE DIAGRAM:

A sequence diagram shows object interactions arranged in a time sequence. It depicts - the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

Object	Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name	
Actor	Actor can also communicated with objects so they too can be listed as a column.An Actor is modeled using the stick figure.	
LifeLine	The Lifeline identifies the existene of the object over time. The notation for a life time is a vertical dotted line etending from an object	
Activation	Activation modeled as rectangular boxes on the lifeline indicate when the object is performing an action	

Message	Messages modeled as horizontal arrows between Activations indicate the communication between the objects.	
Entity	A lifeline with an entity element represents system data.	
Boundary	A lifeline with a boundary element indicates a system boundary/ software element in a system	
Control	lifeline with a control element indicates a controlling entity or manager. It organizes and schedules the interactions between the boundaries and entities and serves as the mediator between them.	

Graphical Notations for Sequence Diagram.

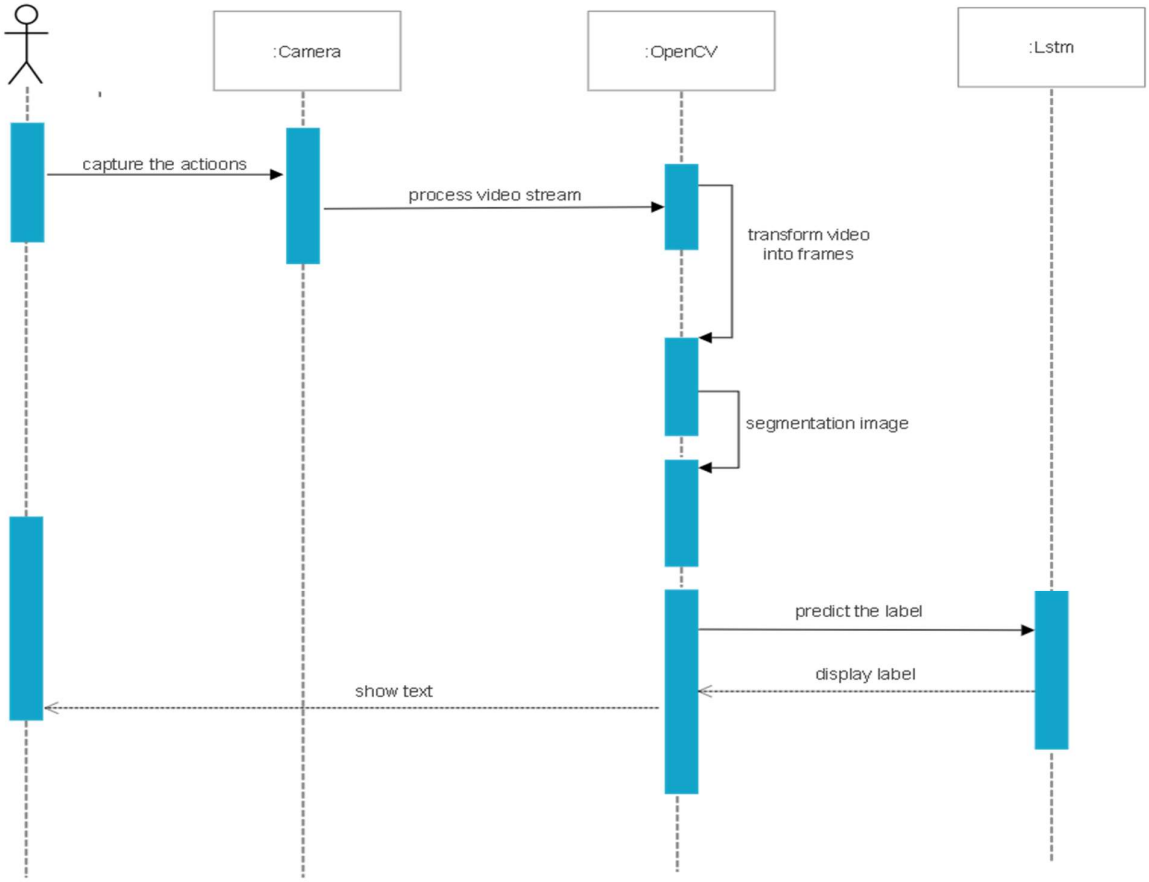


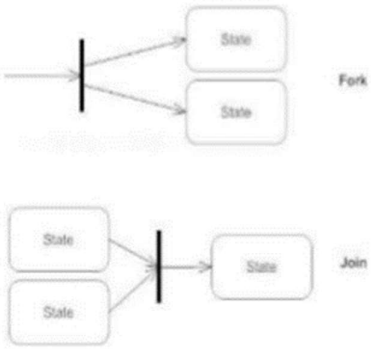
Figure 2: Sequence Diagram

5.2.3 STATE CHART DIAGRAM:

A State chart diagram is used to describe the states of different objects in their life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement accurately.

State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

A State chart diagram describes a state machine. The state machine can be defined as a machine that defines different states of an object and these states are controlled by external or internal events. A state diagram is used to describe the behavior of systems. This behavior is analyzed and represented in a series of events that could occur in one or more possible states. State diagrams require that the system described is composed of a finite number of states. Sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

Initial State	The initial state represents the source of all objects. A filled circle followed by an arrow represents the object's initial state.	
State	State represents situations during the life of an object. Rectangular boxes with curved edges represent a state	
Transition	A transition represents the change from one state to another. A solid arrow represents the path between different states of an object	
Final State	The final state represents the end of an object's existence. A final state is not a real estate, because objects in this state don't exist anymore. A filled circle represents the final state	
Synchronization and Splitting	A short heavy bar with two transitions entering it represents a synchronization of control. The first bar is called a fork where a single transition splits into concurrent multiple transitions. The second bar is called a join, where the concurrent transitions reduce back to one	

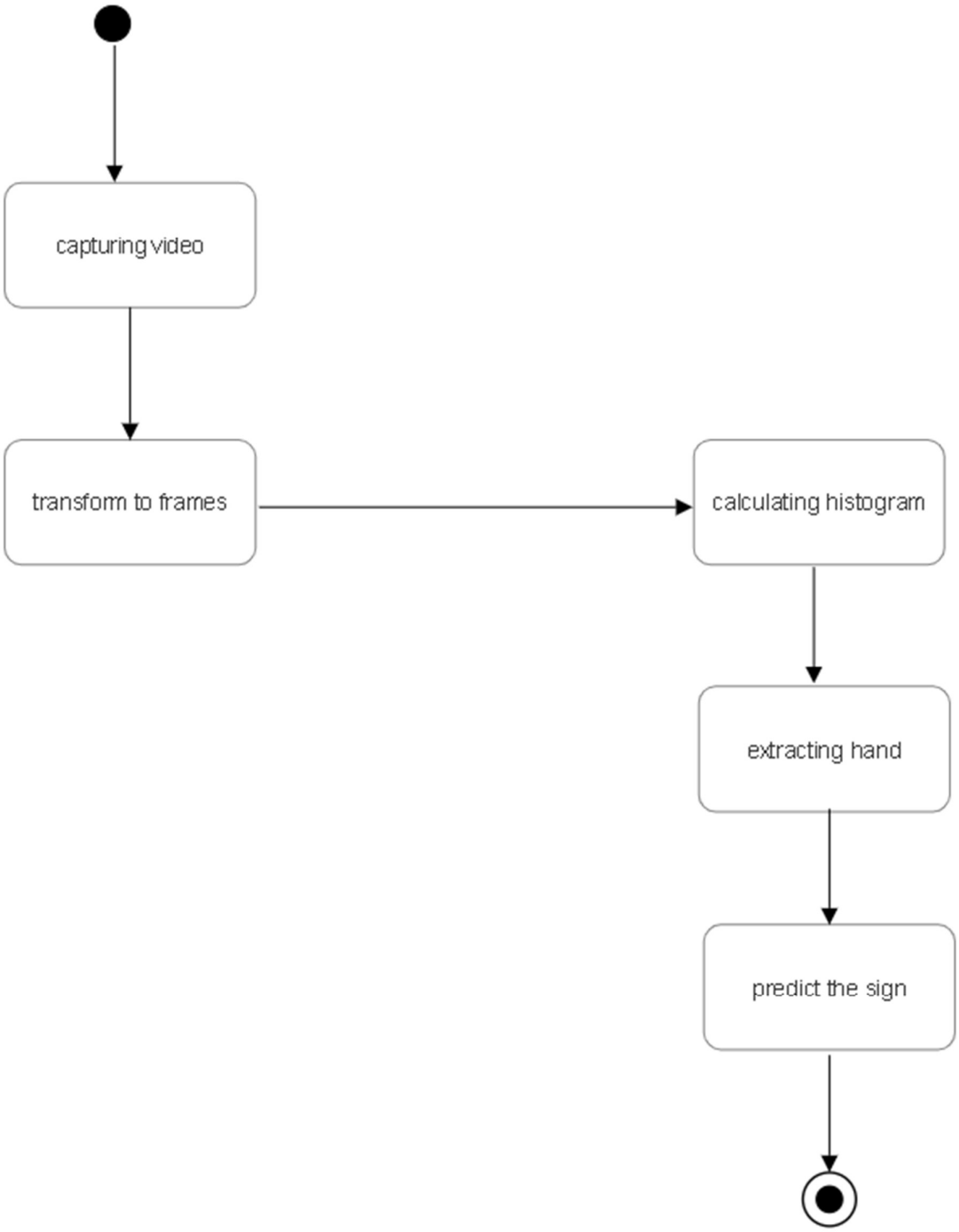


Figure 3: State Chart Diagram

5.2.4 ACTIVITY DIAGRAM:

Activity diagrams are mainly used as a flowchart consisting of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities.

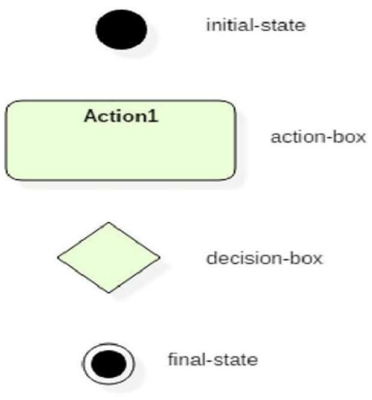
This UML activity diagram shows the Login activity, where the admin will be able to log in using a username and password.

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent.

Activity Diagram Notations:

Activity diagrams symbol can be generated by using the following notations:

- **Initial states:** The starting stage before an activity takes place is depicted as the initial state.
- **Final states**: The state that the system reaches when a specific process ends is known as a Final State.
- **State or Action box:** An activity represents the execution of action on objects or by objects.
- **Decision box:** It is a diamond shape box that represents a decision with alternate paths. It represents the flow of control**.**



The flow of Control of Activity Diagram

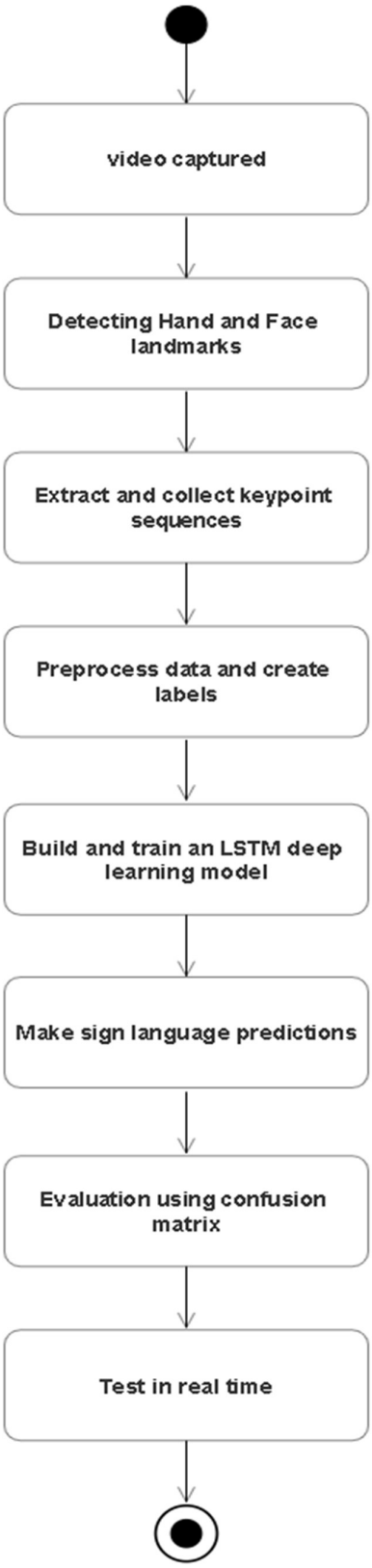


Figure 4: Activity Diagram

5.3 SYSTEM ARCHITECTURE:

The system architecture is the conceptual model that defines the structure, behavior, and views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

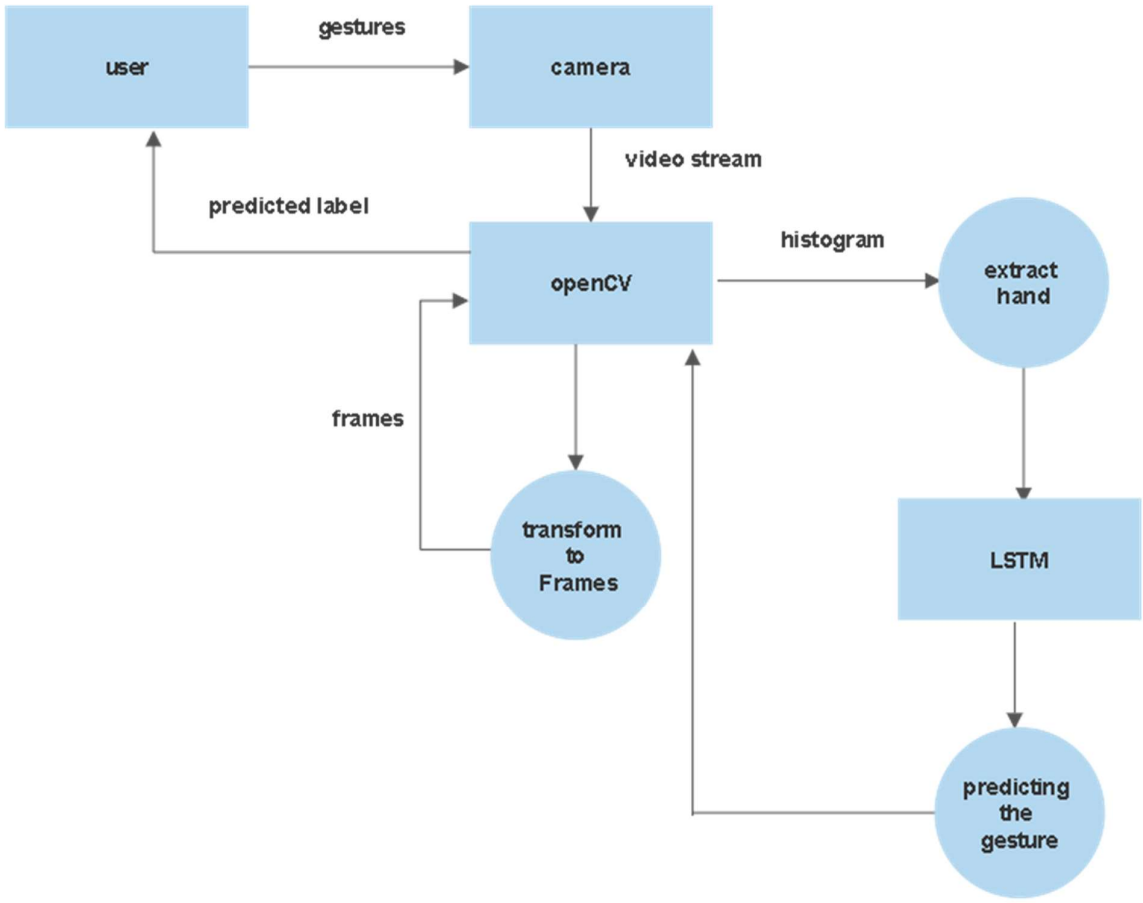


Figure 5: System Architecture for Sign Language Recognition

5.3.1 Algorithm Specification:

Description:

- ***Installing and importing dependencies:***

  Here we do install all the required tools i.e., TensorFlow, OpenCV, MediaPipe, Scikit-learn, matplotlib and import dependencies of NumPy, OS, time and pyplot from matplotlib.

- ***Key points using MediaPipe holistic:***

  Using the MediaPipe python framework we detect Hand, Face, and Pose Landmarks.

- ***Extract Key Points:***

  We now extract all the key points detected using MediaPipe holistic which are in return used for training and building our LSTM Deep Learning model.

- ***Setup folders for data collection:***

  Here we create folders to export the data to be stored as NumPy arrays.

- ***Collect Key point sequences:***

  We collect the sequences for training and testing the model.

- ***Preprocess data and create labels:***

  Here we preprocess the data and create labels and features for the collections which would be of help to detect the sign in real-time.

- ***Build & Train an LSTM deep learning model:***

  Using TensorFlow and Keras we build and train the model where the model summary, and accuracy are defined.

- ***Make sign language predictions:***

  Predictions are made using models from Keras on the tested & trained model.

- ***Save model weights***

- ***Evaluation:***

  We import a couple of metrics i.e. confusion matrix and accuracy from Scikit-learn and evaluate the performance of the model.

- ***Test in real time***

6. SYSTEM IMPLEMENTATION

6.1 TECHNOLOGY DESCRIPTION:

Windows:

Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers and PCs.

Features:

1. Speed:

Even aside from incompatibilities and other issues that many people had with Vista, one of the most straightforward was speed – it just felt too sluggish compared to XP, even on pumped-up hardware. Windows 10 brings a more responsive and sprightlier feel and Microsoft has spent a lot of time and effort getting the Start Menu response just right. Microsoft has also recognized the need for improved desktop responsiveness, which gives the impression that the computer is responding to the user and that they are in control – something that was often lacking with Vista. You can also expect faster boot times. And the boot sequence is now not only prettier than it was with Vista, but it's speedier too.

2. Compatibility:

In simple terms, compatibility on Windows 10 will be far better than it was with Vista. Many programs that individuals and companies used on Windows XP did not work immediately and required updates, but with Windows 10 almost all applications that work on Vista should still run.

3. Lower Hardware Requirements

Vista gained a reputation for making even the beefiest hardware look rather ordinary. Windows 10, however, will run well on lower-end hardware, making the transition from Windows XP less painful.

4. Search and Organization

One of the best things about Windows 10 is the improved search tool, which now rivals Mac OS X‘s Spotlight to be able to find what you need quickly and easily. For example, typing a mouse will bring up the mouse option within the control panel or typing a word will display it and split it up neatly into files, folders and applications.

5. Safety and Security

New security features in Windows include two new authentication methods tailored toward touch screens (PINs and picture passwords), the addition of antivirus capabilities to Windows Defender (bringing it in parity with Microsoft Security Essentials) Smart Screen filtering integrated into Windows, and support for the "Secure Boot" functionality on UEFI systems to protect against malware infecting the 48 boot process. Family Safety offers Parental controls, which allows parents to monitor and manage their children's activities on a device with activity reports and safety controls.

Anaconda (Python Distribution):

Anaconda is a free and open-source distribution of the python and R programming language for scientific computing such as Data Science, Machine Learning applications, large-scale data processing, predictive analytics, etc.

Anaconda Navigator:

Anaconda Navigator is a desktop Graphical User Interface included in Anaconda Distribution that allows users to launch applications and manage Anaconda packages, environments and challenges without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them.

The following Applications are available by default in the navigator:

- ➢ Jupyter Lab
- ➢ Jupyter Notebook
- ➢ Qt Console
- ➢ Spyder
- ➢ Glue

- Orange
- RStudio
- Visual Studio Code

Benefits of Using Python Anaconda:

- It is free and open-source.
- It has more than 1500 Python packages.
- It has tools to easily collect data from sources using machine learning and AI.
- Anaconda simplifies package management and deployment.
- Anaconda is the industry standard for developing, testing and training on a single machine.
- It creates an environment that is easily manageable for deploying any project.
- It has good community support- you can ask your questions there.

What you get:

- Download more than 1500 Python/R data science packages.
- Manage libraries, dependencies, and environments with Anaconda.
- Use Dask, NumPy, Pandas and Numba to analyze scalable data and fast.
- Build and train ML and deep learning models with scikit-learn, Tensor Flow and Theano.
- Perform visualization with Matplotlib, Bokeh, Data shader, and Holo views.

Jupyter Notebook:

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. 49 Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running with Jupyter Notebook:

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This will focus on just two of them to install the Jupyter Notebook. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using Python 3.

Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level Programming language. Python is designed to be highly readable. It uses English keywords frequently whereas in other languages. Python is an easy to learn yet powerful and versatile scripting language, which makes it for Application development.

- **Python is interpreted:**

  Python is processed at runtime by the interpreter. You don‘t need to compile your programs before executing them. This is similar to PHP.

- **Python is Interactive:**

  You can sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented:**

  Python supports Object-oriented style programming that encapsulates code within objects.

- **Python is Beginner’s Language:**

  Python is a great language for the beginner level and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Applications of Python:

- **Web Applications:**

  It Provides Libraries to handle internet protocols such as HTML and XML, JSON, Email Processing, request, beautiful Soup, Feed Parser, etc. It also provides Frameworks such as Django, Pyramid, Flask, etc. to design and develop web-based applications. Some important developments are Python Wiki Engines, Poco, and Python Blog Software.

- **Desktop GUI Applications:**

  Python provides Tk GUI Library to develop user interfaces in python-based applications. Some other toolkits such as wxWidgets, Kivy, and pyqt are useable on several platforms. The kivy is popular for writing multitouch applications.

- **Software Development:**

  Python is popular and widely used in scientific and numeric computing. Some useful libraries and packages are SciPy, Pandas, Ipython, etc. SciPy is a group of packages for engineering, science and mathematics.

- **Business Applications:**

  Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high-level application platform.

- **Console Based Applications:**

  We can use Python to develop console-based applications.
  eg: IPYTHON

- **Audio Video-Based Applications:**

  Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of the real applications are Time Player, cplay etc.

- **3D CAD Applications:**

  To create CAD applications Fandango is a real application that provides full features of CAD.

➢ **Enterprise Applications:**

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications: OpenErp, Tryton, Picalo etc.

Features of Python Programming:

1. **Easy to code:**

   Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like c, c#, JavaScript etc. It is very easy to code in python language and anybody can learn python basic in a few hours or days. It is also a developer-friendly age.

2. **Free and Open Source:**

   Python language is freely available on the official website. Since, it is open-source, this means that source code is also available to the public. So, you can download it as, use it as well as share it.

3. **Object-Oriented Language:**

   One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation etc.

4. **GUI Programming Support:**

   Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

5. **High-Level Language:**

   Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. **Extensible feature:**

   Python is an Extensible language. We can write our python code in C or C++ language and also, and we can compile that code in C/C++ language.

7. **Python is a Portable language:**

   Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, UNIX and Mac then we do not need to change it, we can run this code on any platform.

8. **Python is an integrated language:**

   Python is also an integrated language because we can easily integrate Python with other languages like C, C++ etc.

9. **Interpreted Language:**

   Python is an Interpreted Language because python code is executed line by line at a time. Unlike other languages C, C++, Java etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called Byte code.

10. **Large Standard Library:**

    Python has a large standard library which provides a rich set of modules and functions so you do not have to write your code for every single thing. There are many libraries present in python such as regular expressions, unit-testing, web browsers etc.

11. **Dynamically Typed Language:**

    Python is a dynamically-typed language. That means the type (for example- int, double, long etc.) for a variable is decided at run time not in advance because of this feature we don‘t need to specify the type of variable.

Libraries used:

- **TensorFlow:**

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

- **OpenCV:**

  OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage and then Itseez (which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source BSD license.

- **NumPy:**

  NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, and provides tools for integrating C, C++ etc., The NumPy arrays take significantly less amount of memory as compared to python lists. It also provides a mechanism for specifying the data types of the contents, which allows further optimization of the code.

- **Matplotlib:**

  Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits. Matplotlib, pyplot: Plotting library for 2D graphics plotting.

- **MediaPipe:**

  MediaPipe is an open-source framework to “build world-class machine learning solutions” by Google — currently in the alpha stage. It has been open-sourced for a year now but has likely been under development for far longer.

➢ **Scikit-learn:**

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

➢ **Keras:**

Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer are François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

6.2 SYSTEM MODULES

1. **Install and import dependencies:**

   Here we do install all the required tools i.e. TensorFlow, OpenCV, MediaPipe, Scikit-learn, matplotlib and import dependencies of NumPy, os, time and pyplot from matplotlib.

2. **Collecting key points from MediaPipe holistic:**

   We detect Hand, Face, and Pose Landmarks extract all the key points detected using MediaPipe.

3. **Collecting and preprocessing data:**

   We create folders to export the data to be stored as NumPy arrays and create labels.

4. **Training and Testing:**

   Using TensorFlow and Keras we build and train the model using an LSTM deep learning neural network where the model summary and accuracy are defined, and test in real time.

6.3 SAMPLE SOURCE CODE

INSTALL AND IMPORT DEPENDENCIES

```
!pip install tensorflow tensorflow-gpu opencv-python mediapipe sklearn matplotlib
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
```

KEYPOINTS USING MP HOLISTIC

```
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities
mp_holistic??
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
def draw_landmarks(image, results):
     mp_drawing.draw_landmarks(image,results.face_landmarks,
mp_holistic.FACEMESH_CONTOURS)
    mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS)
```

```
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS)
def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACEMESH_CONTOURS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1,
circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1,
circle_radius=1)
                              )
    # Draw pose connections
    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)
                              )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2,
circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=2,
circle_radius=2)
                              )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2,
circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2,
circle_radius=2)
```

```
                    )
cap = cv2.VideoCapture(0)
# Set 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)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

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

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
draw_landmarks(frame, results)
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

EXTRACT KEYPOINT VALUES

```
results.face_landmarks.landmark
results.face_landmarks.landmark[0]
results.face_landmarks.landmark[0].x
```

```
results.face_landmarks.landmark[0].y
results.face_landmarks.landmark[0].z
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(132)
face = np.array([[res.x, res.y, res.z] for res in
results.face_landmarks.landmark]).flatten() if results.face_landmarks else
np.zeros(1404)
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)
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])
result_test = extract_keypoints(results)
result_test
np.save('0', result_test)
np.load('0.npy')
```

SETUP FOLDERS FOR COLLECTION

```
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('MP_Data')

# Actions that we try to detect
actions = np.array(['hello', 'thanks', 'iloveyou'])

# Thirty videos worth of data
no_sequences = 30

# Videos are going to be 30 frames in length
sequence_length = 30

# Folder start
start_folder = 30
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

COLLECT KEYPOINT VALUES FOR TRAINING AND TESTING

```
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:

    # NEW LOOP
    # Loop through actions
    for action in actions:
```

```
    # Loop through sequences aka videos
    for sequence in range(no_sequences):
        # Loop through video length aka sequence length
        for frame_num in range(sequence_length):

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

            # Make detections
            image, results = mediapipe_detection(frame, holistic)
            print(results)
            # Draw landmarks
            draw_styled_landmarks(image, results)

            # NEW Apply wait logic
            if frame_num == 0:
                cv2.putText(image, 'STARTING COLLECTION', (120,200),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                cv2.putText(image, 'Collecting frames for {} Video Number
{}'.format(action, sequence), (15,12),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1,
cv2.LINE_AA)
                # Show to screen
                cv2.imshow('OpenCV Feed', image)
                cv2.waitKey(500)
            else:
                cv2.putText(image, 'Collecting frames for {} Video Number
{}'.format(action, sequence), (15,12),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1,
cv2.LINE_AA)
                # Show to screen
                cv2.imshow('OpenCV Feed', image)

            # NEW Export keypoints
```

```
            keypoints = extract_keypoints(results)
            npy_path = os.path.join(DATA_PATH, action, str(sequence),
str(frame_num))
            np.save(npy_path, keypoints)

            # Break gracefully
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

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

PREPROCESS DATA AND CREATE LABELS AND FEATURES

```
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 = [], []
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])
np.array(sequences).shape
np.array(labels).shape
X = np.array(sequences)
X.shape
y = to_categorical(labels).astype(int)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05)
y_test.shape
X_test.shape
X_train.shape
```

BUILD AND TRAIN LSTM NEURAL NETWORK

```
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()
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'))
model.compile(optimizer='Adam', loss='categorical_crossentropy',
metrics=['categorical_accuracy'])
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
model.summary()
```

MAKE PREDICTIONS

```
res = model.predict(X_test)
actions[np.argmax(res[4])]
actions[np.argmax(y_test[4])]
```

SAVE WEIGHTS

```
model.save('action.h5')
model.load_weights('action.h5')
```

EVALUATION USING CONFUSION MATRIX AND ACCURACY

```
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)
```

TEST IN REALTIME

```
from scipy import stats
# 1. New detection variables
sequence = []
sentence = []
predictions = []
threshold = 0.5

cap = cv2.VideoCapture(0)
# Set 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)
print(results)

# Draw landmarks
draw_styled_landmarks(image, results)

# 2. Prediction logic
keypoints = extract_keypoints(results)
sequence.append(keypoints)
sequence = sequence[-30:]

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. Viz logic
    if np.unique(predictions[-10:])[0]==np.argmax(res):
        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 probabilities
    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)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
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), 2, cv2.LINE_AA)

    return output_frame
plt.figure(figsize=(18,18))
plt.imshow(prob_viz(res, actions, image, colors))
```

7. TESTING & MAINTENANCE

7.1 INTRODUCTION:

The purpose of testing is to discover errors. Testing is a process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivating forces for well-planned testing.

Testing Objectives:

Several rules can serve as testing objectives: Testing is a process of executing a program with the intent of finding an error. A good test case has a high probability of finding an undiscovered error.

Types of Testing:

To make sure that the system does not have errors, the different levels of testing strategies that are applied at different phases of software development are:

7.1.1 Unit Testing

Unit testing is done on individual models as they are completed and become executable. It is confined only to the designer's requirements. Unit testing is different from and should be preceded by other techniques, including:

- Inform Debugging
- Code Inspection

Black Box testing

In this strategy, some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find errors in the following categories:

- Incorrect or missing functions Interface errors
- Errors in data structures are external database access
- Performance error
- Initialization and termination of errors
- In this testing only the output is checked for correctness
- The logical flow of data is not checked

White Box testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been used to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structures to ensure their validity.

7.1.2 Integration Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated. It is typically performed by developers, especially at the lower, module to the module level. Testers become involved in higher levels

7.1.3 System Testing

Involves in-house testing of the entire system before delivery to the user. The aim is to satisfy the user the system meets all requirements of the client's specifications. It is conducted by the testing organization if a company has one. Test data may range from generation to production. Requires test schedule to plan and organize:

- Inclusion of changes/fixes.
- Test data to use

One common approach is graduated testing: as system testing progresses and (hopefully) fewer and fewer defects are found, the code is frozen for testing for increasingly longer periods.

7.1.4 Acceptance Testing

It is pre-delivery testing in which the entire system is tested at the client's site on real-world data to find errors.

7.2 MAINTENANCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, this system has been developed to satisfy the needs to the largest possible extent. And as part of maintenance, most of the features and expected outcomes of end-users are considered while developing this system. With the development in cluster forming technology, it may be possible to add many more features based on the requirements in the future. The coding and designing are simple and easy to understand which will make maintenance easier.

7.3 TEST CASES

Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objectives. Planning and testing of a programming system involve formulating a set of test cases, which are similar to the real data that the system is intended to manipulate.

S.No	**Test Case**	**Input Description**	**Expected Output**	**Result (Pass/Fail)**
1.	Run the code in Jupyter Notebook without having a dataset	Delete the downloaded dataset on the testing machine &Run the Jupyter notebook cells	Error is thrown	Pass
2.	Loading model	Initializing the trained model and loading it into ON	Loaded model without errors	Pass
3.	Converting video to frames	Capturing video and converting it into frames	Image frames of the captured video stream	Pass
4.	Recognize hand gesture	Image frame that contains hand object	label	Pass

8. OUTPUT SCREENS

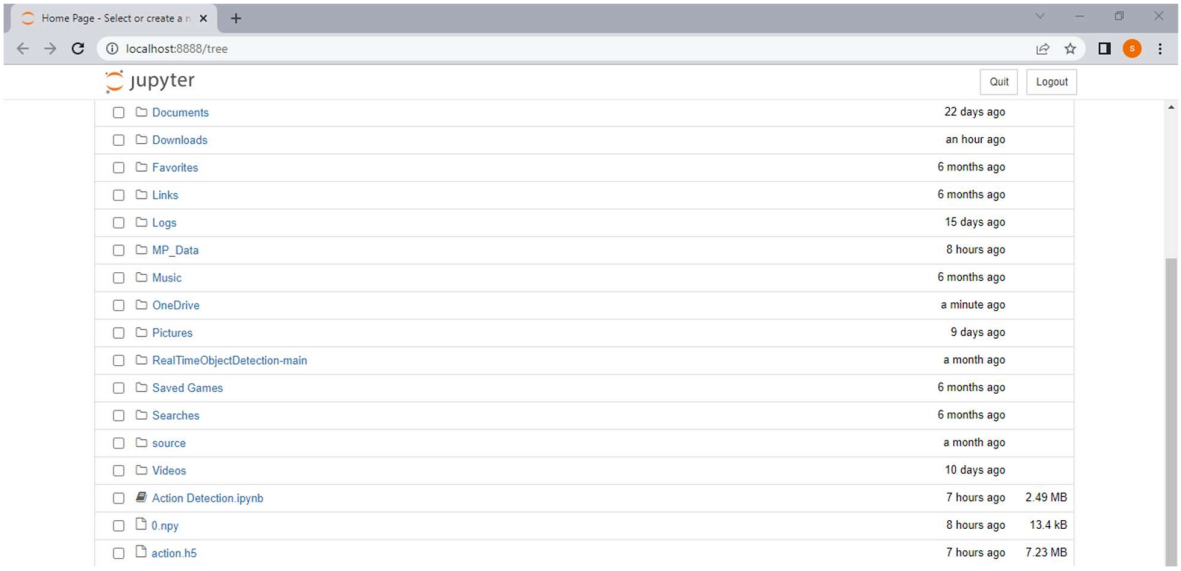


Figure 6: Opening Jupyter Notebook

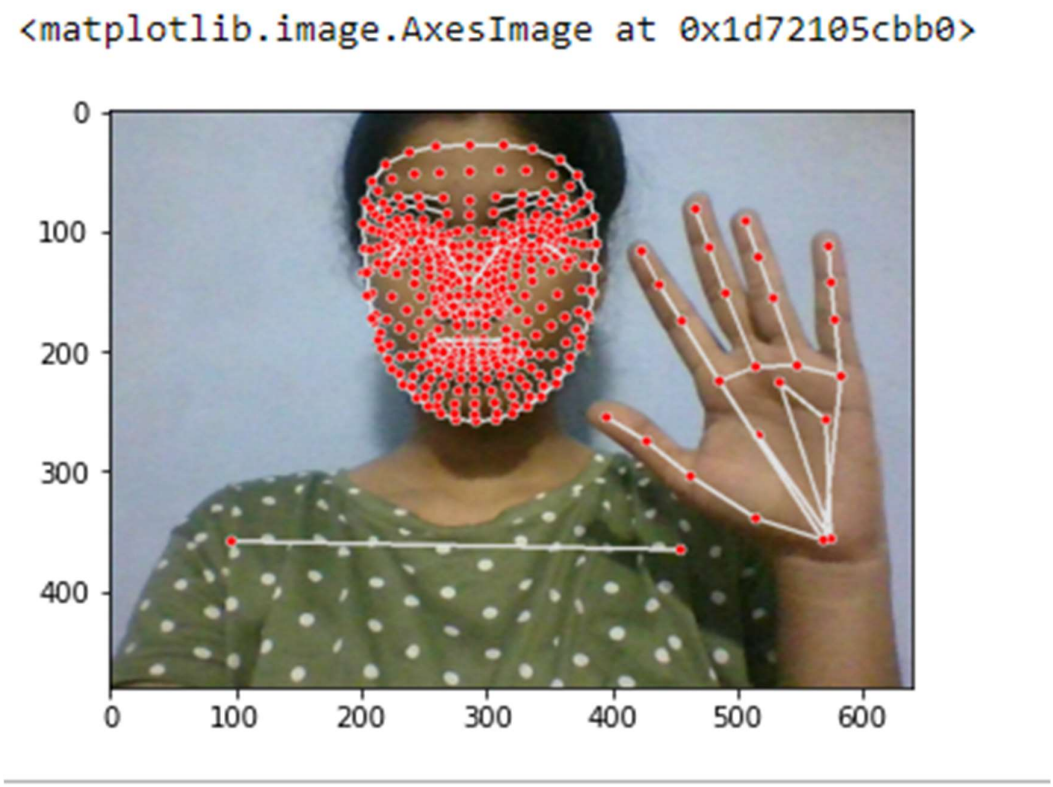


Figure 7: Detecting Key Points

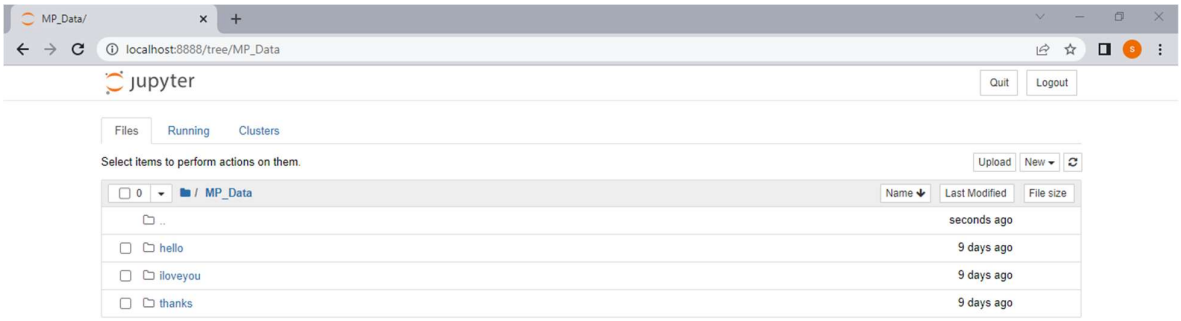


Figure 8: Dataset used to Train and Test

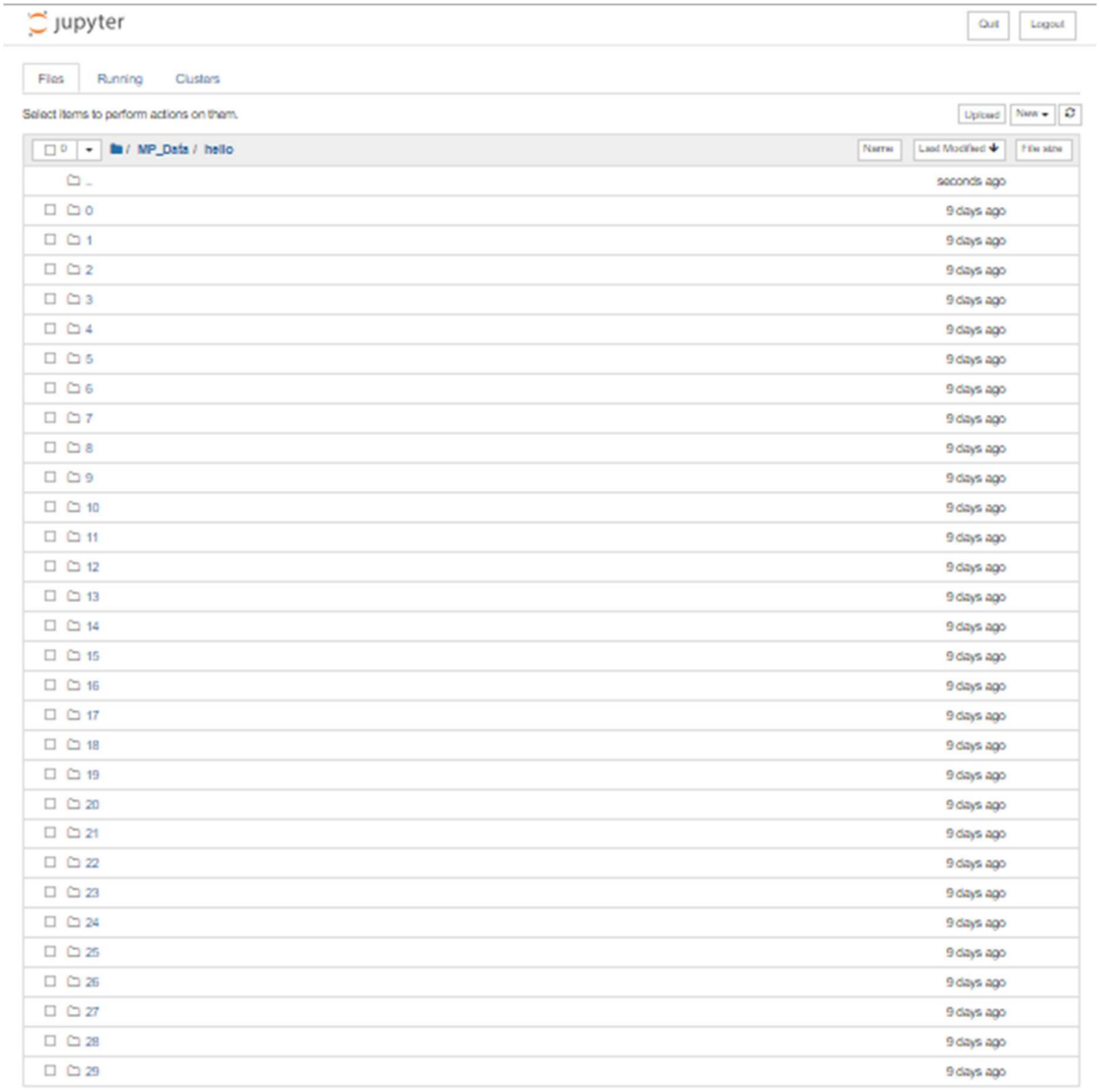


Figure 8: Dataset used to Train and Test

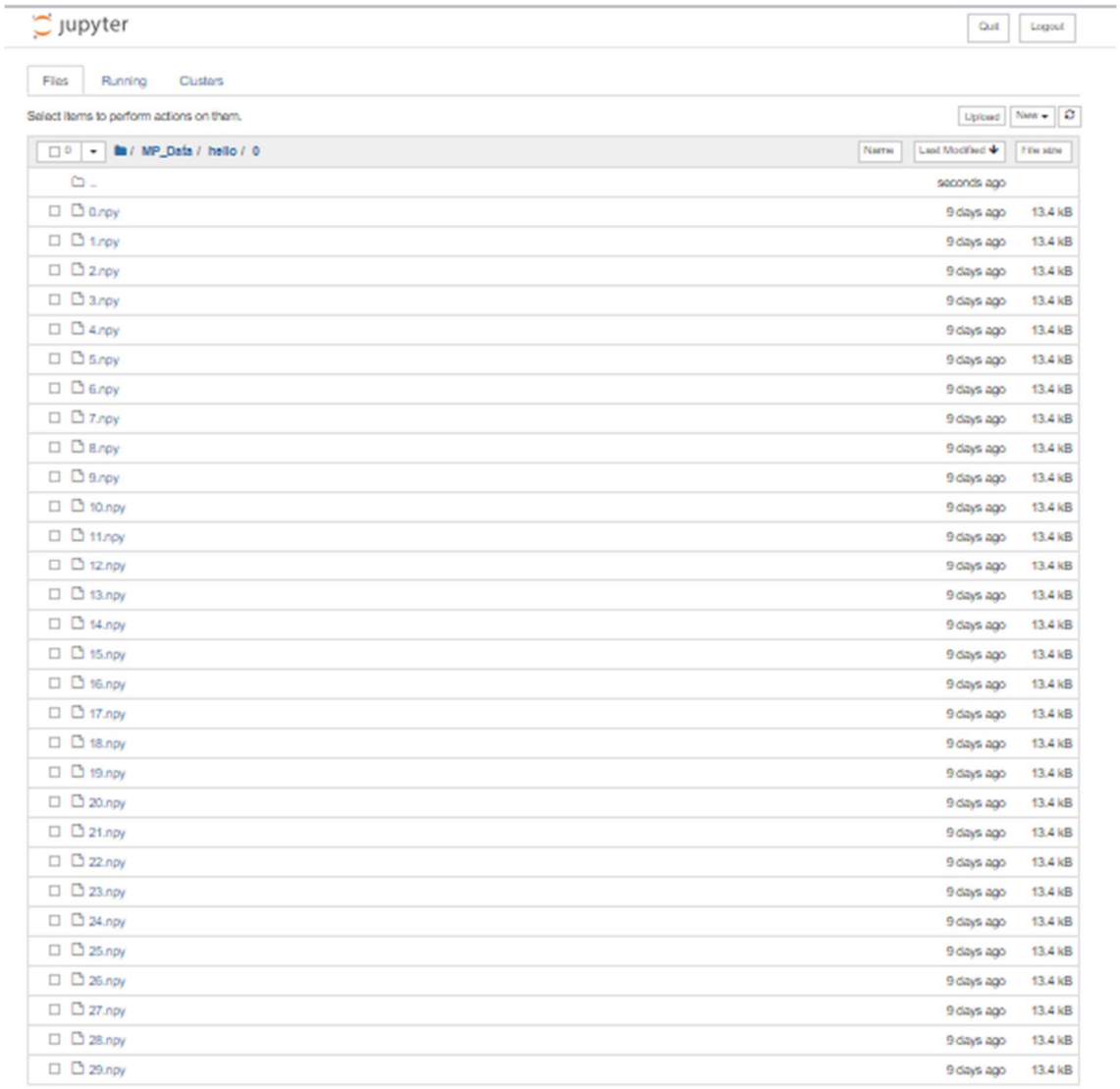


Figure 8: Dataset used to Train and Test

```
In [45]: model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
Epoch 1/2000
3/3 [==============================] - 1s 101ms/step - loss: 1.3544e-05 - categorical_accuracy: 1.0000
Epoch 2/2000
3/3 [==============================] - 0s 113ms/step - loss: 1.3529e-05 - categorical_accuracy: 1.0000
Epoch 3/2000
3/3 [==============================] - 0s 113ms/step - loss: 1.3508e-05 - categorical_accuracy: 1.0000
Epoch 4/2000
3/3 [==============================] - 0s 109ms/step - loss: 1.3501e-05 - categorical_accuracy: 1.0000
Epoch 5/2000
3/3 [==============================] - 0s 114ms/step - loss: 1.3470e-05 - categorical_accuracy: 1.0000
Epoch 6/2000
3/3 [==============================] - 0s 111ms/step - loss: 1.3436e-05 - categorical_accuracy: 1.0000
Epoch 7/2000
3/3 [==============================] - 0s 126ms/step - loss: 1.3414e-05 - categorical_accuracy: 1.0000
Epoch 8/2000
3/3 [==============================] - 0s 115ms/step - loss: 1.3420e-05 - categorical_accuracy: 1.0000
Epoch 9/2000
3/3 [==============================] - 0s 124ms/step - loss: 1.3396e-05 - categorical_accuracy: 1.0000
Epoch 10/2000
```

Figure 9: Training model for 2000 epochs

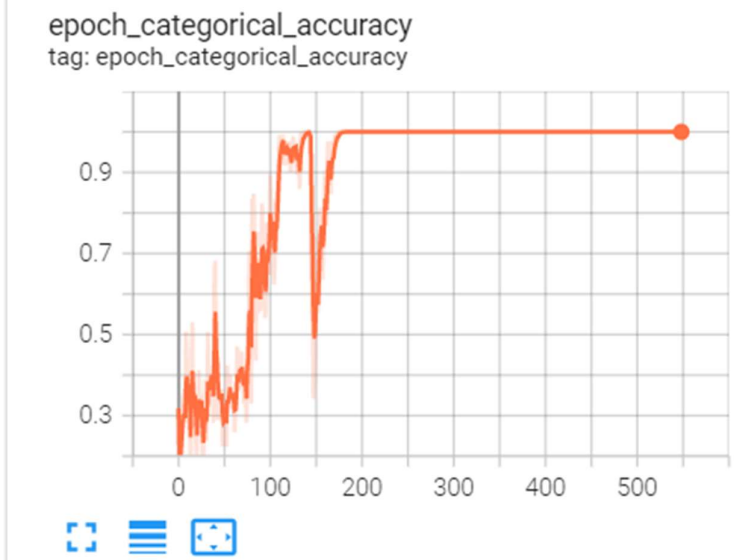


Figure 10: Accuracy

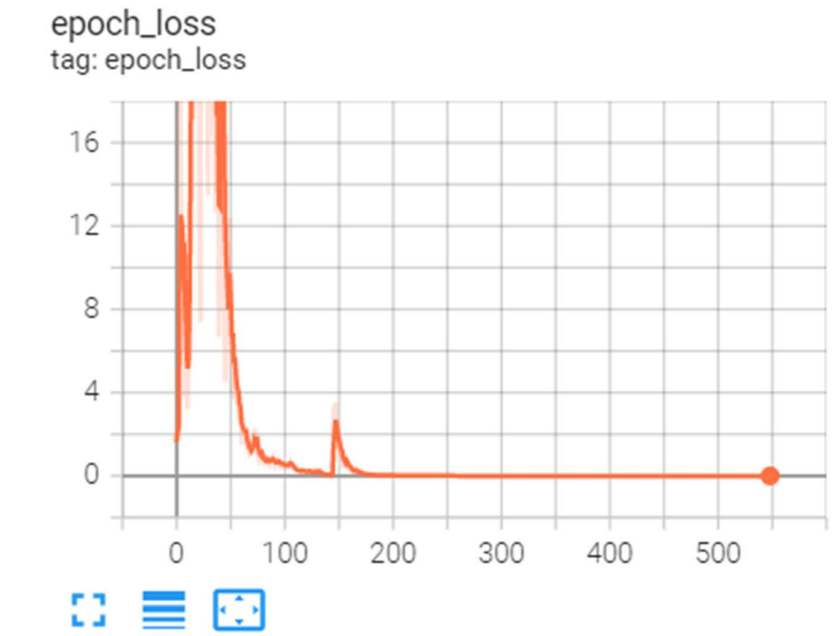


Figure 11: Loss

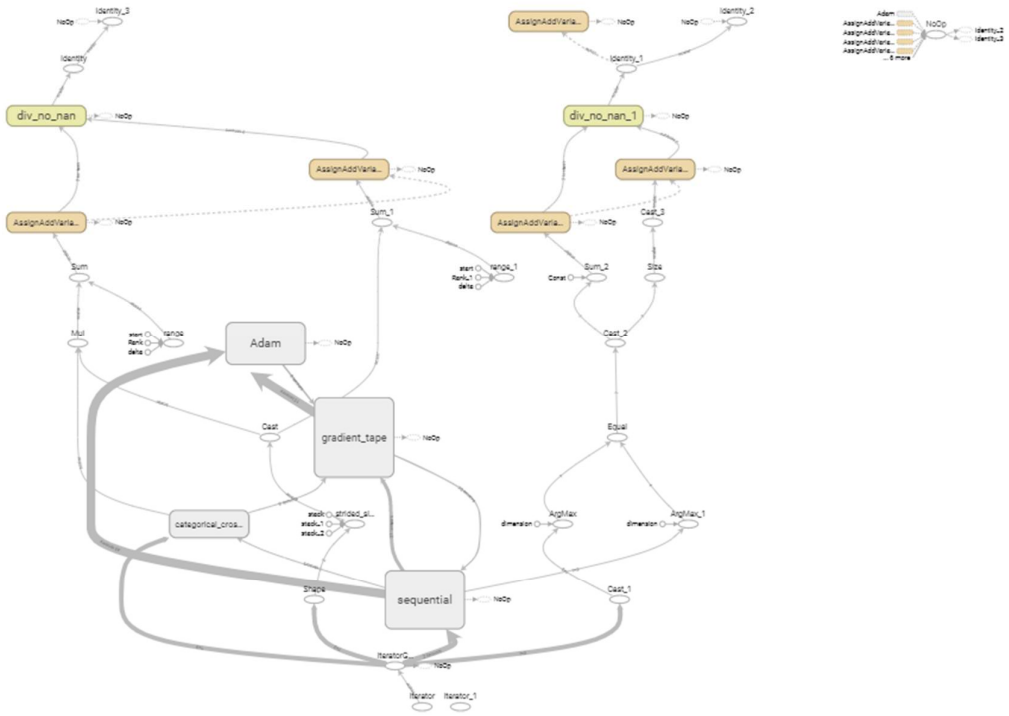


Figure 12: TensorFlow Network Graphs

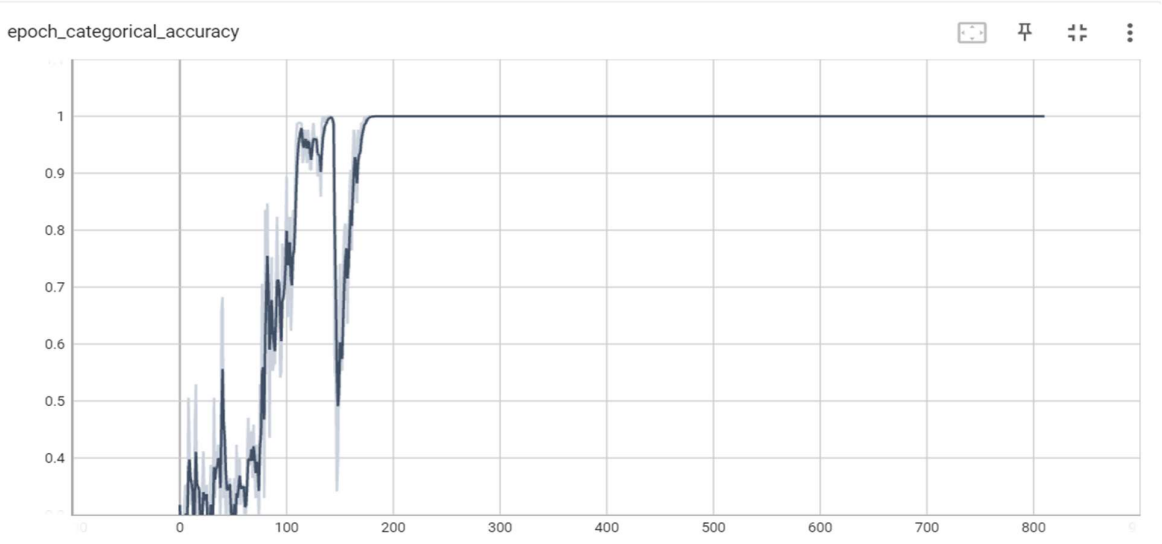


Figure 13: Evaluating performance by the accuracy

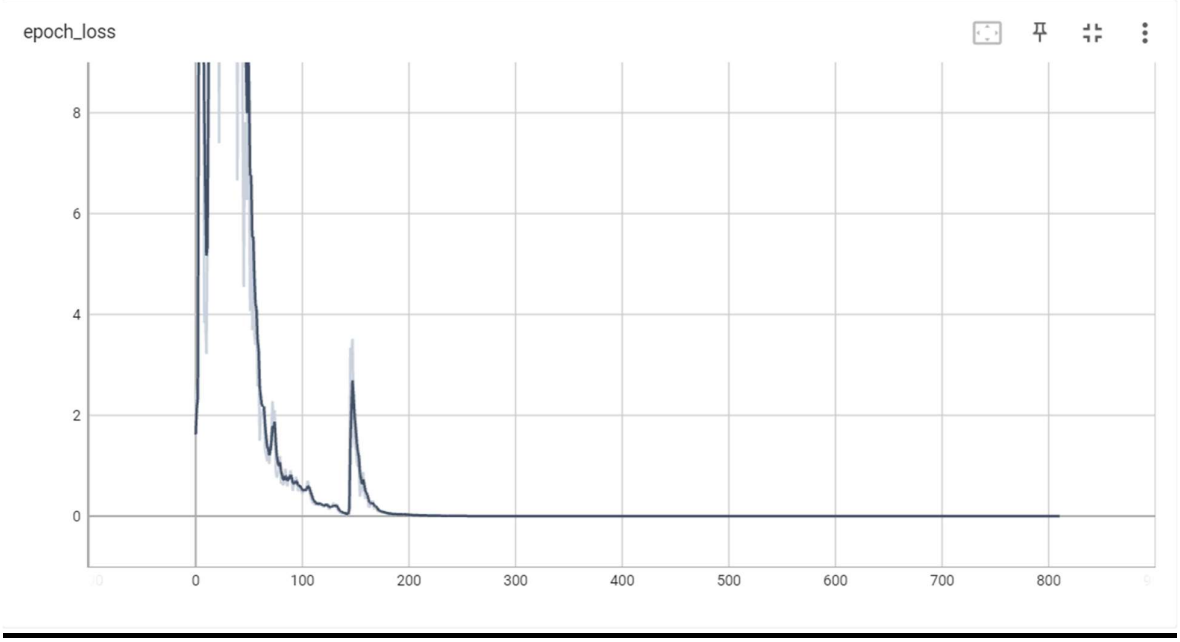


Figure 14: Minimizing Loss using Adam optimizer

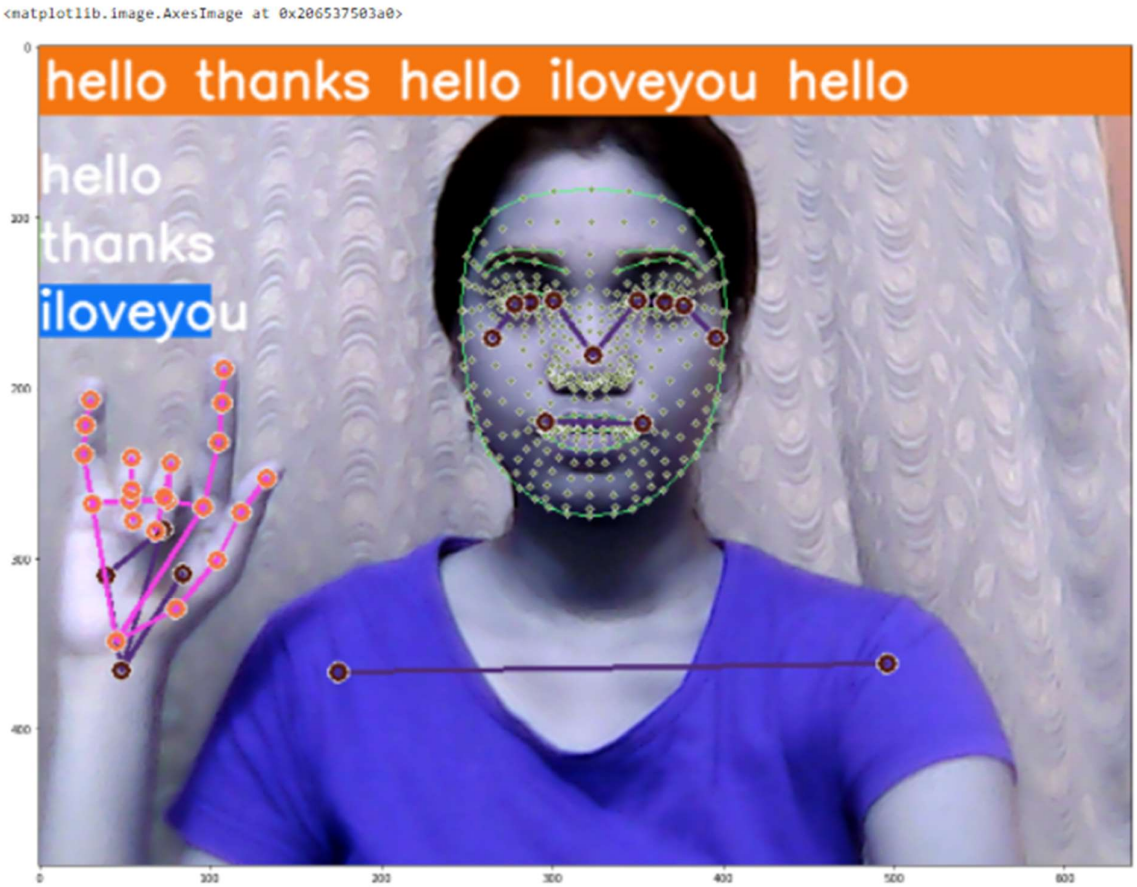


Figure 15: Real-time recognition of sign language

9. CONCLUSION

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

10. FUTURE ENHANCEMENT

The future scope contains yet is not limited to:

- This developed model can be introduced to other sign languages such as Indian Sign Language, at present it is finite to American Sign Language.

- The model can be further trained with a dataset such that it automatically segments the gesture from the captured frame by automatically subtracting the background.

- Tuning and Augmenting the model to identify usual words and expressions.

- Additionally, training the neural network model to be well organized and identify symbols require two hands.

- Incorporate active hand gestures in augmented to the contemporaneous static fingerspelling.

- Integration of the optimized model to existing AI systems like Amazon Alexa for advancements in visual recognition.

11. REFERENCES

[1] Google AI Blog: On-Device, Real-Time Hand Tracking with Media Pipe (googleblog.com)

[2] http://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html

[3] https://mediapipe.dev/

[4] https://ieeexplore.ieee.org/document/8537248

[5] A Practical Introduction to Computer Vision with OpenCV-WILEY

[6] https://opencv.org/

[7] https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c

[8]https://www.researchgate.net/publication/342331104_Sign_Language_Recognition_Using_Deep_Learning_and_Computer_Vision

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

SIGN LANGUAGE RECOGNITION

Arugula Sismai[*1], Asritha Diddi[*2], Chavali Anusha[*3],
Chinta Sundara Sreya[*4], Prof. B. Prajna[*5]

[*1,2,3,4,5]Department Of Computer Science And Systems Engineering, Andhra University College Of Engineering For Women, Visakhapatnam, Andhra Pradesh, India.

ABSTRACT

Sign language is the only tool of communication for the person who is not able to speak and hear anything. Sign language is a boon for verbally challenged people to express their thoughts and emotion. Using this Sign Language Recognition system, the communication gap between people with hearing impairments and the general public can be cleared. In this work, a scheme of sign language recognition has been proposed for identifying the gestures in sign language. With the help of computer vision and neural networks, we can detect the signs and give the respective text as output. The major aim of this work is to build a neural network using a Long Short-Term Memory (LSTM) deep learning model using the video frames which offer the translation of gestures into text. The model is trained with the dataset that is collected using holistic key points from the video of the person which detects the pose, face and hand landmarks.

Keywords: Sign Language Recognition; Application Of Sign Language; Data Input; Deaf; Mute.

I. INTRODUCTION

Sign Language is a language that includes gestures with bodily movements made with the hands including facial expressions and postures. It is mainly used by people who are deaf and dumb. There are many different sign languages such as British, Indian and American sign languages. People with disabilities like deaf and dumb use sign language as a tool to express their emotions and thoughts to common people around them. Yet the general public finds it hard to understand the sign and therefore such a trained system like sign language recognition is required during medical and legal appointments, educational and training sessions and for the global meetings being held. A few years ago, there has been an increase in demand for such systems which are formed as video remote human interpreters using high-speed internet connectivity which provided an easy way to translate the sign language that has been used and benefited from yet had a various number of limitations.

To overcome this, we use a Long Short-Term Memory (LSTM) model to detect the actions in sign language. A neural network of six layers is constructed using LSTM deep learning model in which three are LSTM layers and the other three are Dense layers. The dataset we use contains the actions as a specific number of sequences stored as frames which are captured using OpenCV with an interval of time.

II. METHODOLOGY

2.1 REQUIREMENTS

- Software Requirements:

1. Operating system: Windows 7 and above.
2. Language: Python
3. Libraries:

a. OpenCV: An open-source distribution that is primarily focused on real-time applications that provide computational efficiency for managing massive volumes of data. It processes photos and videos to recognize signs and gestures.

b. TensorFlow: An open-source artificial intelligence package that is used to build models using data-flow graphs and to build large-scale neural networks using several layers.

c. Keras: This is used along with TensorFlow to build a neural network that leverages LSTM layers to handle the sequence of key points.

d. MediaPipe: A Framework for building machine learning pipelines for processing time-series data like video, audio, etc. Here we'll be using mediapipe holistic which is one of the pipelines which contains optimized face, hands, and pose landmarks that allow collecting key points, thus enabling the model to simultaneously detect hand and body poses along with face landmarks.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

e. Sklearn: This is used to evaluate the performance of the system using the built-in metrics and confusion matrix which gives us the accuracy.

- Hardware Requirements:

1. Camera: Good quality, 3MP
2. RAM: Minimum 8GB and higher.
3. GPU: 4GB dedicated.
4. Processor: Intel Pentium 4 or higher.
5. HDD: 10GB or higher.
6. Monitor: 15 inches or 17 inches color monitor.
7. Mouse: Scroll or Optical. / Touchpad.

2.2 PROPOSED SYSTEM

Initially, the video of the person is captured using OpenCV which is taken as an input and then the data is collected using MediaPipe holistic which detects the face, pose and hand landmarks as key points. The dataset to be trained is stored as several sequences put in frames of video format where the key points are pushed into a NumPy array.

Hereafter, the system is trained and built using Long Short-Term Memory (LSTM) deep learning model which is constructed using three LSTM layers and three Dense layers. This model was trained for 2000 epochs on a batch size of 128 using the dataset extracted. The model was trained using the dataset to minimize the loss by categorical cross-entropy using the Adam optimizer.

Finally, after building the neural network, real-time sign language recognition is performed using OpenCV where the gestures are recognized and displayed as text within the highlighted section.

The working of the proposed system is shown as a network architecture below:

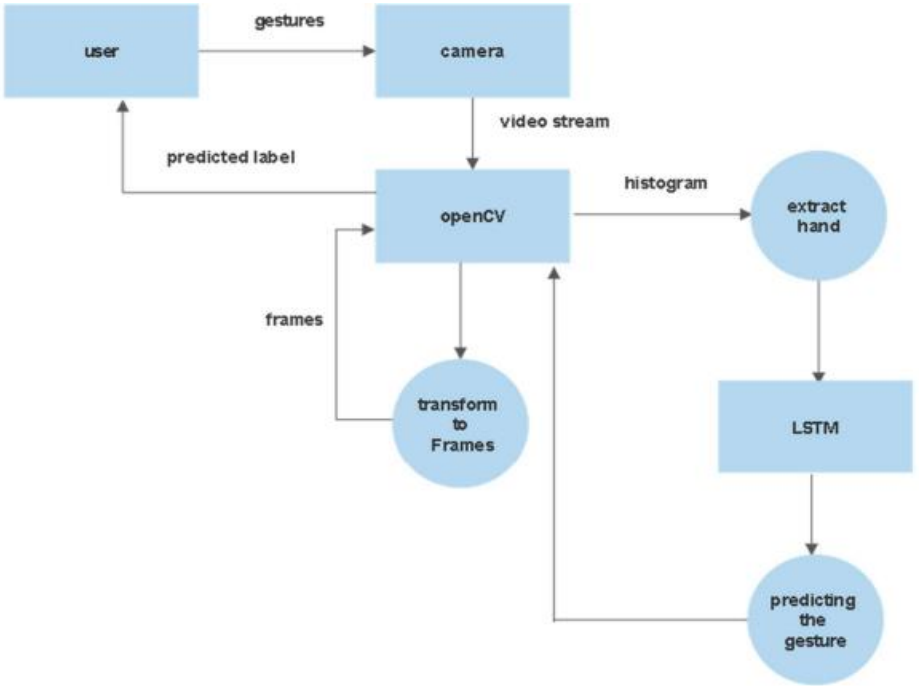


Figure 1: Network architecture of the proposed system.

2.3 ALGORITHM SPECIFICATION

The step by step procedure to construct this system:

1. Installing and importing dependencies.
2. Key points using MediaPipe holistic.
3. Extract Key Points.
4. Setup folders for data collection.
5. Collect Keypoint sequences.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

6. Preprocess data and create labels.
7. Build and Train an LSTM deep learning model.
8. Make sign language predictions.
9. Save model weights.
10. Evaluation using confusion matrix and accuracy score.
11. Test in real-time.

2.4 SYSTEM MODULES

1. Install and import dependencies:

Here we do install all the required tools i.e. TensorFlow, OpenCV, MediaPipe, Sklearn, matplotlib and import dependencies of NumPy, os, time and pyplot from matplotlib.

2. Collecting key points from MediaPipe holistic:

We detect Hand, Face, and Pose Landmarks and extract all the key points detected using MediaPipe holistic.

3. Collecting and pre-processing data:

We create folders to export the data to be stored as NumPy arrays and create labels.

4. Training and Testing:

Using TensorFlow and Keras we build and train the model using an LSTM deep learning neural network where the model summary and accuracy are defined and tested in real-time.

III. TESTING

Software testing is an examination that provides us with information about the quality of the system being evaluated. Product testing can also provide an objective, unbiased view of the software, allowing them to appreciate the usage and understand the limitations associated with its implementation. One example of a test technique is the practice of executing a program or application to find software problems such as errors and many more. Program testing can provide users with objective, unbiased information regarding software quality and the risk of failure.

Table 1: Test Cases Representation

S.No	Test Case	Input Description	Expected Output	Result (Pass/Fail)
1.	Run the code in Jupyter Notebook without havinga dataset	Delete the downloaded dataset onthe testing machine & Run the Jupyter notebook cells	Error is thrown	Pass
2.	Loading model	Initializing the trained model and loading it into ON	Loaded modelwithout errors	Pass
3.	Converting video to frames	Capturing video and converting it intoframes	Image frames ofthe captured video stream	Pass
4.	Recognize hand gesture	Image frame that contains hand object	label	Pass

As soon as there is an executable program, software testing can be started even if it is completed partially. The whole strategy for software testing or development usually determines when and how testing is done, as well as the results. The majority of testing, for example, occurs after system requirements have been created and then implemented in testable code in a planned process.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

IV. RESULTS

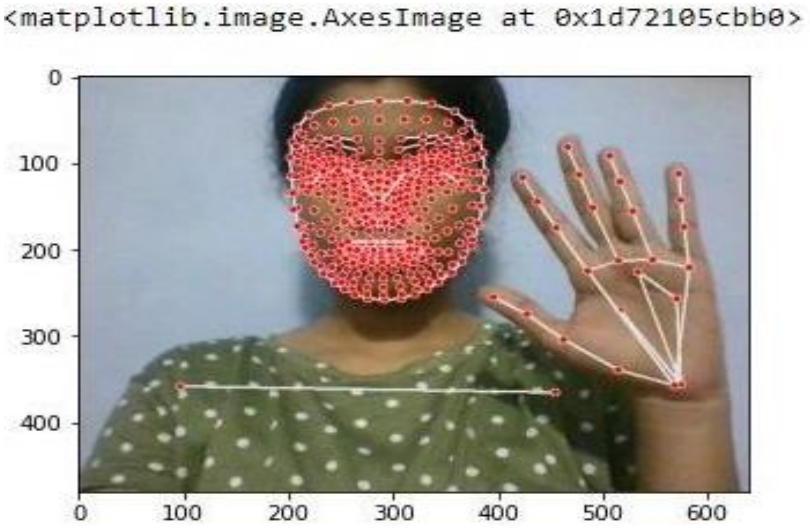


Figure 2: Extracting key points using MediaPipe holistic.

```
In [45]: model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])

Epoch 1/2000
3/3 [==============================] - 1s 101ms/step - loss: 1.3544e-05 - categorical_accuracy: 1.0000
Epoch 2/2000
3/3 [==============================] - 0s 113ms/step - loss: 1.3529e-05 - categorical_accuracy: 1.0000
Epoch 3/2000
3/3 [==============================] - 0s 113ms/step - loss: 1.3508e-05 - categorical_accuracy: 1.0000
Epoch 4/2000
3/3 [==============================] - 0s 109ms/step - loss: 1.3501e-05 - categorical_accuracy: 1.0000
Epoch 5/2000
3/3 [==============================] - 0s 114ms/step - loss: 1.3470e-05 - categorical_accuracy: 1.0000
Epoch 6/2000
3/3 [==============================] - 0s 111ms/step - loss: 1.3436e-05 - categorical_accuracy: 1.0000
Epoch 7/2000
3/3 [==============================] - 0s 126ms/step - loss: 1.3414e-05 - categorical_accuracy: 1.0000
Epoch 8/2000
3/3 [==============================] - 0s 115ms/step - loss: 1.3420e-05 - categorical_accuracy: 1.0000
Epoch 9/2000
3/3 [==============================] - 0s 124ms/step - loss: 1.3396e-05 - categorical_accuracy: 1.0000
Epoch 10/2000
```

Figure 3: Training the LSTM model for 2000 epochs.

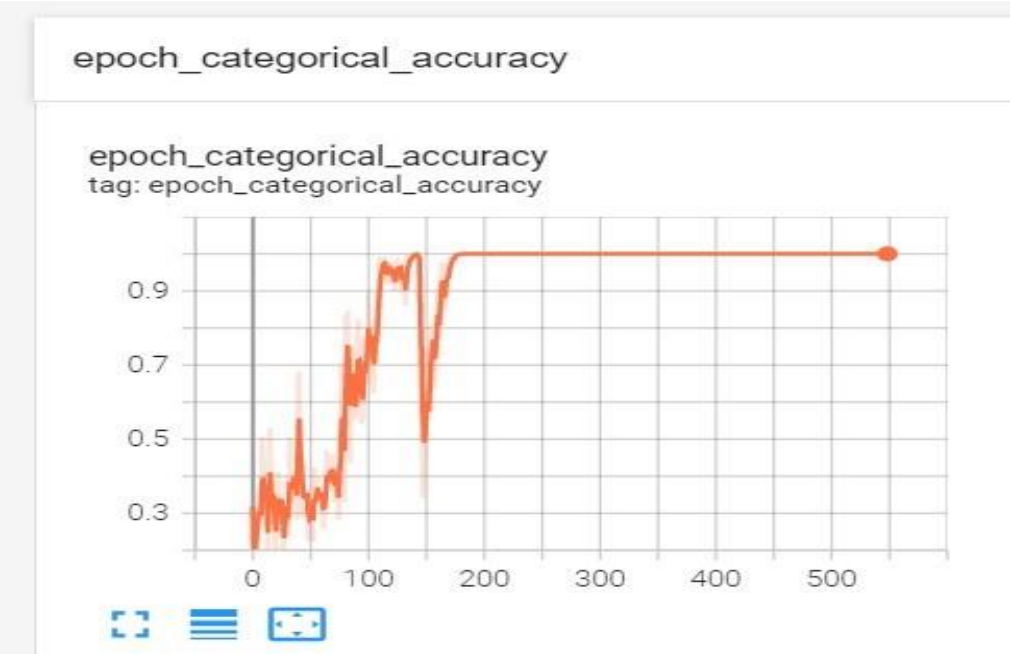


Figure 4: Evaluation using accuracy score of 1.000

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

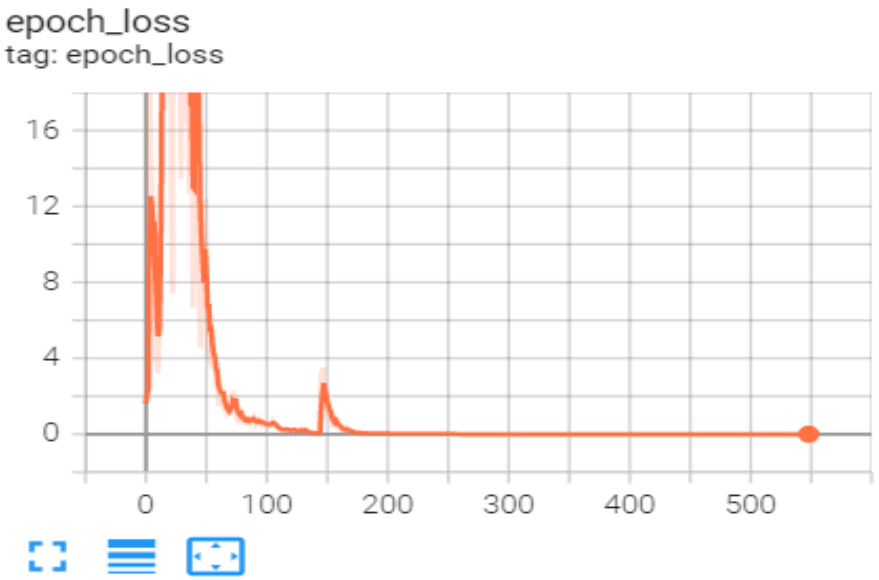


Figure 5: The model is trained to minimize the loss using Adam Optimizer.

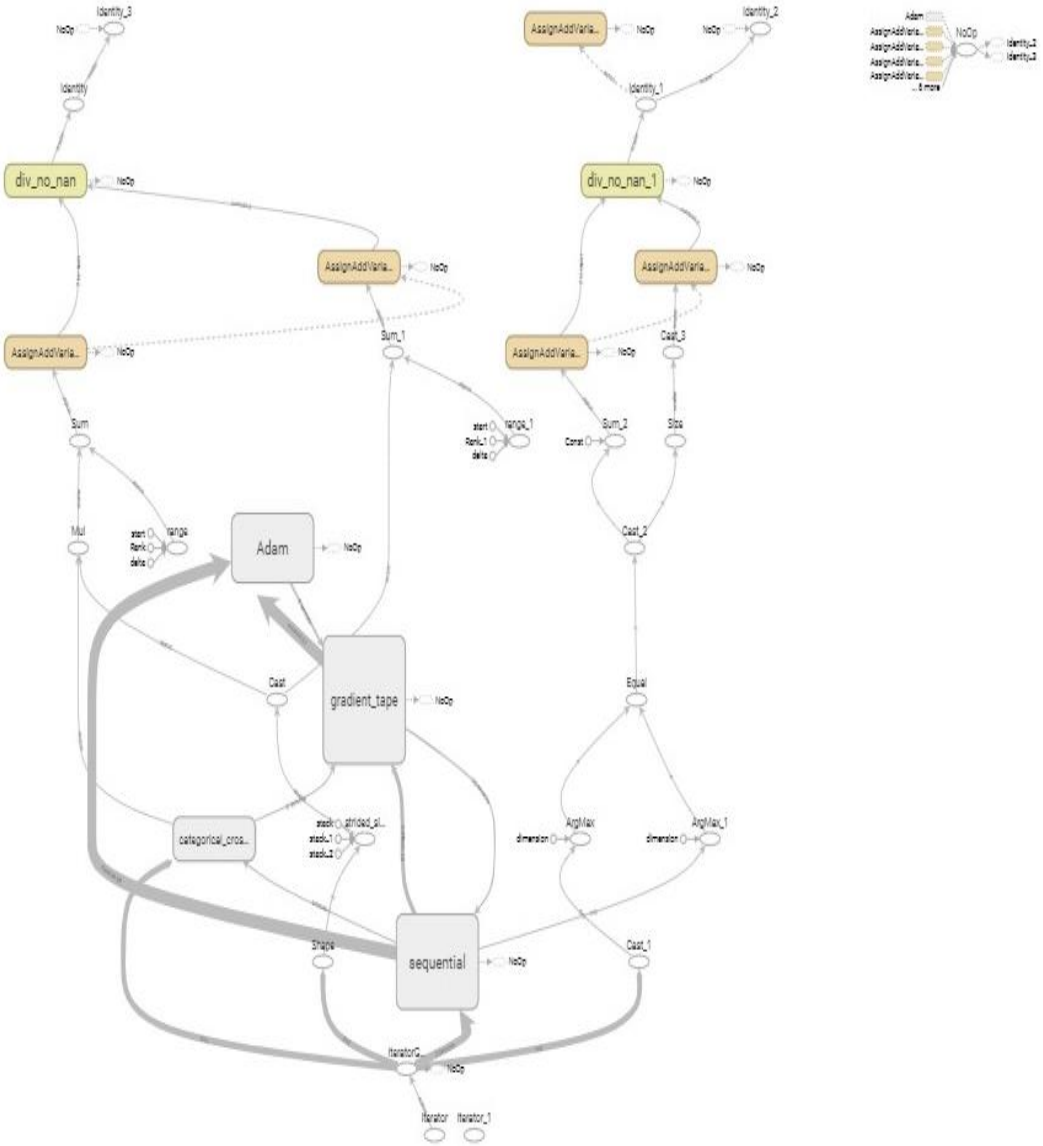


Figure 6: Data flow graph built using TensorFlow for the proposed system.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

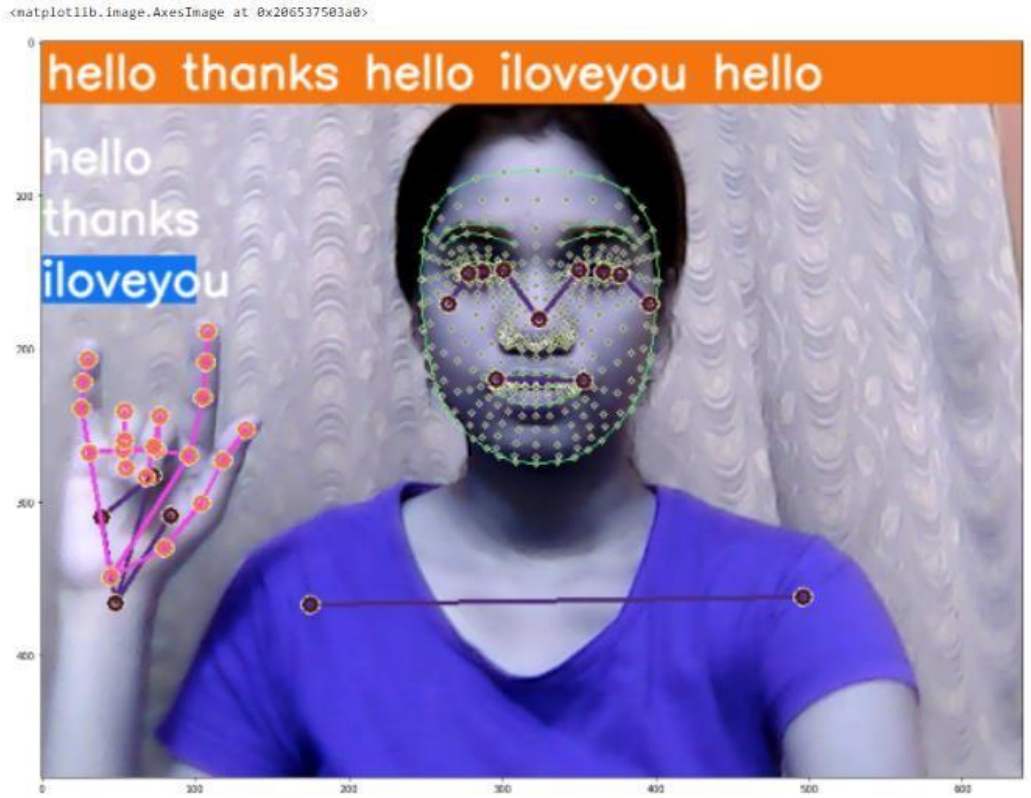


Figure 7: Performing real-time sign language recognition using OpenCV

V. FUTURE ENHANCEMENT

The future scope of this work can be extended:

- To implement other sign languages such as American Sign Language and Indian Sign Language.
- To further train the LSTM model to recognize alphabets and symbols.
- To enhance the model to detect facial expressions.
- To display sentences instead of words which could be a more appropriate sign language translation which also increases readability.
- To add a greater number of training data which results in a higher accuracy score.
- To convert signs to speech.
- To develop a proficient system completely that could help the deaf and dumb people.

VI. CONCLUSION

From classifying signs and numbers, the Sign Language Recognition System can be progressed to a system that can recognize dynamic movements in continuous sequences of images. Nowadays, both researchers and developers are focusing their efforts on developing a wide vocabulary for sign language recognition systems. They differ in their classification methods and the model being trained for detecting sign language as each one of them uses their customized working model. Because of the differences in sign language between countries and the conditions set, fair comparisons between various models are limited. The majority of the country's sign language variations are dependent on their grammar and how they portray each word.

No matter what whichever model being used can break the bridge gap between the people with hearing impairments and the general public from which many can be benefited during real-time activities such as education, global meetings, and medical and legal appointments.

VII. REFERENCES

[1] Aditya Das, Shantanu Gawde, Khyati Suratwala, Dr. Dhananjay Kalbande (2018, February). Facial expression recognition from video sequences: temporal and static modelling. Computer Vision and Image Undertaking 91.

[2] http://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html

[3] https://mediapipe.dev/

[4] https://ieeexplore.ieee.org/document/8537248

[5] A Practical Introduction to Computer Vision with OpenCV-WILEY Continuous dynamic Indian Sign Language gesture recognition with invariant backgrounds by Kumud Tripathi, Neha Baranwal, G. C. Nandi at 2015 Conference on Advances in Computing, Communications and Informatics (ICACCI).

[6] https://www.researchgate.net/publication/342331104_Sign_Language_Recognition_Using_Deep_Learning_and_Computer_Vision

[7] https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember- the-past-55e54c

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 05/40500037818

Date: 07/05/2022

Certificate of Publication

*This is to certify that author "**Arugula Sismai**" with paper ID "**IRJMETS40500037818**" has published a paper entitled "SIGN LANGUAGE RECOGNITION" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 05, May 2022***

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 05/40500037818

Date: 07/05/2022

Certificate of Publication

*This is to certify that author "**Asritha Diddi**" with paper ID "**IRJMETS40500037818**" has published a paper entitled "SIGN LANGUAGE RECOGNITION" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 05, May 2022***

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 05/40500037818

Date: 07/05/2022

Certificate of Publication

*This is to certify that author "**Chavali Anusha**" with paper ID "**IRJMETS40500037818**" has published a paper entitled "SIGN LANGUAGE RECOGNITION" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 05, May 2022***

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 05/40500037818

Date: 07/05/2022

Certificate of Publication

*This is to certify that author "**Chinta Sundara Sreya**" with paper ID "**IRJMETS40500037818**" has published a paper entitled "SIGN LANGUAGE RECOGNITION" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 05, May 2022***

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 05/40500037818

Date: 07/05/2022

Certificate of Publication

*This is to certify that author "**Prof. B. Prajna**" with paper ID "**IRJMETS40500037818**" has published a paper entitled "SIGN LANGUAGE RECOGNITION" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 05, May 2022***

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com