HAND GESTURE RECOGNITION AND VOICE CONVERSION FOR DEAF AND DUMB

A PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

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

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

Ch. V. N. S. Sai Divyesh	**(HT No. 19JN1A0524)**
G. V. Hemanth Kumar	**(HT No. 19JN1A0537)**
B. Vishwaroop Achari	**(HT No. 19JN1A0510)**
D. Esak Adbuta Kumar	**(HT No. 20JN5A0506)**
D. Deeven	**(HT No. 19JN1A0531)**

Under the Esteemed Guidance of

Mrs. V. KUSUMA PRIYA, M.Tech.,

Assistant Professor

Department of Computer Science and Engineering

SREE VENKATESWARA COLLEGE OF ENGINEERING

NAAC 'A' Grade Accredited Institution, An ISO 9001::2015 Certified Institution (Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu) NORTH RAJUPALEM(VI), KODAVALURU(M), S.P.S.R NELLORE (DT) – 524 316

2019-2023

SREE VENKATESWARA COLLEGE OF ENGINEERING

NAAC 'A' Grade Accredited Institution, An ISO 9001::2015 Certified Institution
(Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu)
NORTH RAJUPALEM(VI), KODAVALURU(M), S.P.S.R NELLORE (DT) – 524 316

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Project report entitled as **"HAND GESTURE RECOGNITION AND VOICE CONVERSION FOR DEAF AND DUMB"** submitted by **Ch. V. N. S. Sai Divyesh (HT No. 19JN1A0524), G. V. Hemanth Kumar (HT No. 19JN1A0537), B. Vishwaroop Achari (HT No. 19JN1A0510), D. Esak Adbuta Kumar (HT No. 20JN5A0506), D. Deeven (HT No. 19JN1A0531)** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** by **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** during the academic year **2022-2023**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of the Guide

Mrs. V. KUSUMA PRIYA

Signature of Head of the Dept.

Dr. K. VENKATA NAGENDRA

Principal

Dr. P. KUMAR BABU

External Viva-Voce conducted on: _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

The satisfaction and elation that accompany the successful completion of any task would be incomplete without the mention of the people who have made it a possibility. It is a great privilege to express our gratitude and respect to all those who have guided and inspired us during the course of the project work.

We express indebtedness to our honourable chairman **Dr. P. BABU NAIDU** garu who provided all the facilities and necessary encouragement during the course of study.

We express profound sense of gratitude and sincere thanks to our beloved principal **Dr. P. KUMAR BABU** garu for motivating us and providing necessary infrastructure to complete the project.

We convey our thanks to the Head of the Department **Dr. K. VENKATA NAGENDRA** garu for his timely help and guidance whenever required.

We take this opportunity to express my sincere deep sense of gratitude to our Coordinator **Mr. P. MOHAN**, Assistant Professor, Department of Computer Science & Engineering for his significant suggestions and help in every respect to accomplish the project report.

We find immense pleasure in expressing profound gratitude and thanks to our Guide **Mrs. V. KUSUMA PRIYA**, Assistant Professor, Department of CSE for standing by our side all through the project.

We also thank all staff members of the Department of CSE for their support. We also place on record special thanks to our parents and friends who were with us throughout the course of our project.

Project Associates

Ch. V. N. S. Sai Divyesh (HT No. 19JN1A0524)

G. V. Hemanth Kumar (HT No. 19JN1A0537)

B. Vishwaroop Achari (HT No. 19JN1A0510)

D. Esak Adbuta Kumar (HT No. 20JN5A0506)

D. Deeven (HT No. 19JN1A0531)

Dept. of Computer Science and Engineering

Sree Venkateswara College of Engineering,

NH 16 Bypass Road, North Rajupalem, Nellore.

DECLARATION

We hereby declare that the project entitled **"HAND GESTURE RECOGNITION AND VOICE CONVERSION FOR DEAF AND DUMB"** has been done by us under the guidance of **Mrs. V. KUSUMA PRIYA, Assistant Professor, Department of Computer Science and Engineering**. This project work has been submitted to **SREE VENKATESWARA COLLEGE OF ENGINEERING** as a part of partial fulfilment of the requirements for the award of degree of **Bachelor of Technology**.

We also declare that this project report has not been submitted at any time to another Institute or University for the award of any degree.

Project Associates

Ch. V. N. S. Sai Divyesh (HT No. 19JN1A0524)

G. V. Hemanth Kumar (HT No. 19JN1A0537)

B. Vishwaroop Achari (HT No. 19JN1A0510)

D. Esak Adbuta Kumar (HT No. 20JN5A0506)

D. Deeven (HT No. 19JN1A0531)

LIST OF CONTENTS

ABSTRACT

Individuals primarily communicate with one another. Dumb and deaf people use sign language to communicate with others. These individuals have difficulty communicating their message to ordinary people. Deaf and dumb people believe they are unable to communicate because of a lack of communication skills, and as a result, they are unable to express their emotions. Because most individuals aren’t educated in sign language, communicating in an emergency can be extremely challenging. As a consequence, the challenge may be solved by converting hand gestures into human-hearing sounds and text. Vision and non-vision approaches are two of the most commonly used methods for detecting hand movements or gestures. In a vision-based approach, a camera will be used for gesture detection, whereas sensors will be employed in a non-vision-based technique. In this study, a vision-based technique was used. This device detects and locates hand motions in order to keep a communication channel open with others. Using convolutional neural networks, this research develops a gesture recognition system. This study looks into the advantages and disadvantages of hand motion recognition.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S. NO.	ABBREVIATION	DESCRIPTION
1.	SVM	Support Vector Machine
2.	CNN	Convolutional Neural Networks
3.	NIC-CSD	National Informatics Centre - Centre for Software Development
4.	SDLC	Software Development Life Cycle
5.	RTM	Requirements Traceability Matrix
6.	UML	Unified Modeling Language
7.	TTS	Text-To-Speech

CHAPTER 1

INTRODUCTION

Sign language is becoming more popular as a technique to communicate with those who are unable to communicate verbally. It is a language in which hand motions are used to express alphabets and words. The vision technique has been the most extensively used method for sign recognition in recent decades. It's a technology that uses a camera to identify data transmitted by finger motions. It is the most commonly used visual-based method. Vision-based sign recognition systems have taken a lot of time and effort to develop all over the world. The two vision-based gesture recognition systems are direct and indirect. Previously, for the recognition of hand gestures, a vision-based approach was used. However, the ambient influence on the detected picture is significant in this approach. The hand motion is detected and converted into speech and text. One of the most important challenges that this one-of-a kind personality suffers from is the communication gap between a disabled person and an ordinary person. Due to a lack of communication, deaf and dumb people are unable to express their feelings. Hand Gesture Recognition and Voice Conversion (HGRVC) technology identifies and monitors the hand motions of the deaf and dumb, allowing them to converse with others. Webcams are used to detect hand movements. With the help of pre-processing, the images are then converted to normal size. The goal of this study is to create a system that can translate hand gesture into speech and text. Hand gesture is analysed as part of the identification. The technology provides text output, which helps deaf people, and also speech output, which helps blind people and humans communicate more effectively.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system for the project is **Support Vector Machine (SVM)**. The data is divided into two or more categories using the **Classifier model** which is a supervised learning method based on kernels. For binary classification, SVM was developed and utilized. Support vector machines' training step involves building a model, mapping each class's decision boundaries, and locating the hyperplane that divides the various classes. By enhancing the hyperplane margin, the distance between the classes may be increased while maintaining classification accuracy. The implementation's core is libsvm. The fit time complexity is more than quadratic with the number of samples, making it difficult to scale to datasets with more than a few tens of thousands of samples. The management of the multiclass support is done using the one-vs-one approach.

DISADVANTAGES OF EXISTING SYSTEM

- SVM is less adaptable. SVM requires handcrafted feature extraction which can be time-consuming and challenging.
- SVM has less interactive user interface.
- SVM has lower recognition capability.
- SVM is less efficient and less accurate.

2.2 PROPOSED SYSTEM

The system we propose for the project is **Convolutional Neural Networks (CNN)**. The recommended model is **Sequential model** because this deep learning model is founded on a sequence of layers. The model passes information from the input layer to the hidden layer and finally to the output layer. Convolution's main goal is to find the features in our image using a

feature detector and arrange them in a feature map in order to retain the spatial link between pixels.

ADVANTAGES OF PROPOSED SYSTEM

- CNN is more adaptable. CNN can learn more complex features directly from raw data, which can save time and effort in feature extraction.
- CNN has more user-friendly interface compared to SVM. It will be designed to be easy to use for individuals with no technical expertise and it will provide clear feedback on the recognized gestures.
- CNN has higher recognition capability than SVM. CNN is a deep learning model that can automatically learn and extract features from data, which can lead to better performance in complex tasks like hand gesture recognition.
- CNN is more accurate and more efficient when compared to SVM.

2.2.1 FEASIBILITY STUDY

TECHNICAL FEASIBILITY

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or

exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC. There is nominal expenditure and economic feasibility for certain.

OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

2.2.2 MODULES

The proposed approach consists of the following modules:

1. **Upload Hand Gesture Dataset:** This project takes advantage of Kaggle's hand gesture recognition dataset. Various hand gesture images are included in this dataset. The data collection includes different hand movements.
2. **Preprocess Dataset:** Data preprocessing is a technique for transforming raw data into clean data. When data is acquired from several sources, it is in raw format, making analysis impossible.
3. **Model Generation:** In machine learning, model generation is an iterative process in which the machine learning models are constantly trained and tested to find the optimal one for a particular job. There are a variety of machine learning models to choose from, and the one to choose is mostly determined by the task at hand. In this system, Convolutional Neural Networks (CNN) is employed.

4. **Train CNN Gesture Images:** We're utilising deep learning technique CNN to train the gesture photos.
5. **Sign Language Recognition from Webcam:** Vision-based and non-vision-based approaches are the two types of gesture recognition techniques. Vision-based techniques employ a web camera and markers to identify signs. Vision-based technique is used for the gesture recognition in this system.
6. **Extract image from Webcam:** Input is taken through a webcam in this system, and then segments the palm and fingers to identify and recognize motions.
7. **Convert image to binary or grey format and background removal:** It's critical to separate items from their surroundings and convert them to binary images. In pattern recognition, binary images are utilised as inputs to the feature extraction process and play a crucial part in the development of unique features that may be used to discriminate between different classes.
8. **Extract features from image:** Parts or patterns of an item in a picture that assist in identifying it are known as "features". Feature extraction is a step in the dimensionality reduction process, which involves dividing and condensing a large amount of raw data into smaller groupings.
9. **Recognition and play audio:** Finally, the hand motions that have been identified are transformed into voice and text.

2.3 SOFTWARE REQUIREMENT SPECIFICATION

2.3.1 FUNCTIONAL REQUIREMENTS

The major functional requirements of the system are as follows:

- **Hand Gesture Recognition:** The system should be able to accurately recognize and classify hand gestures in real-time to enable communication between the user and the system. The system should have a high recognition rate for different hand gestures, and should be able to recognize gestures in different lighting conditions.

- **Voice Conversion:** The system should be able to convert written text or language into spoken language to facilitate communication between the user and the system. The system should generate natural-sounding speech that is easy to understand.
- **User Interface:** The system should have a user-friendly interface that is easy to navigate, and enables the user to interact with the system easily. The interface should be designed to be accessible to users with different abilities, such as those with visual or motor impairments.
- **Real-Time Processing:** The system should be able to perform gesture recognition and voice conversion in real-time to provide immediate feedback to the user. The system should respond quickly to user inputs, and should minimize latency and delay.
- **Accuracy and Reliability:** The system should be accurate and reliable in recognizing gestures and converting speech to ensure effective communication with the user. The system should have a high recognition rate for different hand gestures, and should generate natural-sounding speech with high accuracy.
- **Adaptability:** The system should be adaptable to different environments and lighting conditions to ensure reliable performance in various settings. The system should be able to adjust to different lighting conditions, and should work equally well in indoor and outdoor settings.

2.3.2 NON-FUNCTIONAL REQUIREMENTS

The major non-functional requirements of the system are as follows:

- **Performance:** The system should perform efficiently and quickly, with minimal latency, to ensure smooth and uninterrupted communication between the user and the system. The system should be designed to handle a large number of users simultaneously, without compromising on performance.
- **Scalability:** The system should be scalable, and should be able to accommodate increasing numbers of users as the project progresses.

The system should be designed to handle additional features and functionality as required, without affecting performance or stability.

- **Usability:** The system should be easy to use, with a simple and intuitive interface that enables the user to navigate the system easily. The system should be designed to be accessible to users with different abilities, such as those with visual or motor impairments.
- **Reliability:** The system should be reliable, with minimal downtime or disruptions. The system should be designed to handle errors and exceptions gracefully, and should be able to recover quickly from any failures.
- **Maintainability:** The system should be easy to maintain, with clear documentation and well-structured code. The system should be designed to facilitate future enhancements and modifications, without affecting stability or performance.
- **Portability:** The system should be portable, and should be able to run on different platforms and devices, such as mobile phones, tablets, and laptops. The system should be designed to be platform-agnostic, without compromising on performance or functionality.
- **Security:** The system should be designed to protect user data and maintain the confidentiality of information exchanged between the user and the system. The system should be secure against external attacks and hacking attempts, and should have robust authentication and authorization mechanisms in place.

2.3.3 LITERATURE SURVEY

"Real-time two-way communication approach for hearing impaired and dumb person based on image processing"

In the recent years, there has been rapid increase in the number of deaf and dumb victims due to birth defects, accidents and oral diseases. Since deaf and dumb people cannot communicate with normal person so they have to depend on some sort of visual communication. Gesture shows an expressive movement of body parts such as physical movements of head, face, arms, hand

or body which convey some message. Gesture recognition is the mathematical interpretation of a human motion by a computing device. Sign language provide best communication platform for the hearing impaired and dumb person to communicate with normal person. The objective of this research is to develop a real time system for hand gesture recognition which recognize hand gestures, features of hands such as peak calculation and angle calculation and then convert gesture images into voice and vice versa. To implement this system, we use a simple night vision webcam with 20-megapixel intensity. The ideas consisted of designing and implement a system using artificial intelligence, image processing and data mining concepts to take input as hand gestures and generate recognizable outputs in the form of text and voice with 91% accuracy.

"Computer vision-based approach for Indian Sign Language character recognition"

Deaf and dumb people communicate among themselves using sign languages, but they find it difficult to expose themselves to the outside world. This paper proposes a method to convert the Indian Sign Language (ISL) hand gestures into appropriate text message. In this paper the hand gestures corresponding to ISL English alphabets are captured through a webcam. In the captured frames, the hand is segmented and the state of fingers is used to recognize the alphabet. The features such as angle made between fingers, number of fingers that are fully opened, fully closed or semi closed and identification of each finger are used for recognition. Experimentation done for single hand alphabets and the results are summarised.

"AAWAAZ: A communication system for deaf and dumb"

The paper proposes a framework for recognizing hand gesture which would serve not only as a way of communication between deaf and dumb and mute people, but also, as an instructor. Deaf and dumb individuals lack in proper communication with normal people and find it difficult to properly express themselves. Thus, they are subjected to face many issues in this regard.

The sign language is very popular among them and they use it to express themselves. Thus, there is a need of a proper translator. The deaf and dumb are not idle as past, they are working outside and doing great at it. So, an efficient system must be set up, to interact with them, to know their views and ideas. The framework here, acts as a communication system for deaf and dumb individuals. It would take the sign language as an input which would display the result not only in the form of text but also in the form of audio. Similarly, if there is any input in the form of text, it would display the corresponding image.

"Two Way Communicator between Deaf and Dumb People and Normal People"

One of the most precious gifts of nature to human beings is the ability to express themselves by responding to the events occurring in their surroundings. Every normal human being sees, listens and then reacts to the situations by speaking himself out. But there are some unfortunate ones who are deprived of this valuable gift. This creates a gap between the normal human beings and the deprived ones. This application will help for both of them to communicate with each other. The system is mainly consisting of two modules, first module is drawing out Indian Sign Language (ISL) gestures from real-time video and mapping it with human-understandable speech. Accordingly, second module will take natural language as input and map it with equivalent Indian Sign Language animated gestures. Processing from video to speech will include frame formation from videos, finding region of interest (ROI) and mapping of images with language knowledge base using Correlation based approach then relevant audio generation using Google Text-to- Speech (TTS) API. The other way round, natural language is mapped with equivalent Indian Sign Language gestures by conversion of speech to text using Google Speech-to-Text (STT) API, further mapping the text to relevant animated gestures from the database.

"Design and Development of Hand Gesture Recognition System for Speech Impaired People"

All over world, deaf and dumb people face struggle in expressing their feelings to other people. There are various challenges experienced by speech and hearing-impaired people at public places in expressing themselves to normal people. The solution to this problem is determined in this paper, by the usage of the Indian sign language symbols which are generic to all deaf and dumb people in India. The gestures illustrated by the Indian sign language symbols will be conquered with the support of the flex sensors and accelerometer. The movements included during gesture representation are rotation, angle tilt, and direction changes. The flex sensor and the accelerometer are incorporated over fingers and wrist respectively to acquire their dynamics, these sensors are fitted over the data glove. These voltage signals will then be processed by microcontroller and sent to voice module, where the words voice outputs are stored and play backed equivalent to each word values to produce the appropriate voice words with the help of the speaker.

"Human action recognition using DFT"

Action is any meaningful movement of the human and it is used to convey information or to interact naturally without any mechanical devices. Human action recognition is motivated by some of the applications such as video retrieval, Human robot interaction, to interact with deaf and dumb people etc. In any Action Recognition System, some pre-processing steps are done for removing the noise caused because of illumination effects, blurring, false contour etc. Background subtraction is done to remove the static or slowly varying background. In this paper, multiple background subtraction algorithms are tested and then one of them is selected for the further process of action recognition. Background subtraction is also known as foreground/background segmentation or foreground extraction. The next step is the feature extraction which deals with the extraction of the important feature (like corner points, optical flow, shape, motion vectors etc.) from the image frame. The proposed

novel action recognition algorithm uses Discrete Fourier Transform (DFT) of the small image block.

2.3.4 SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

The following are the software requirements needed to develop the project:

- **Operating System** : Windows 7 or 8 (minimum)
- **Programming Language** : Python

HARDWARE REQUIREMENTS

The following are the hardware requirements needed to develop the project:

- **Processor** : Intel Core i3 (minimum)
- **Webcam** : 720 pixels (minimum)
- **RAM** : 2 GB (minimum)
- **Hard disk** : 256 GB (minimum)
- **Keyboard** : Standard Windows Keyboard
- **Mouse** : Standard Mouse
- **Monitor** : SVGA

CHAPTER 3

SYSTEM DESIGN

3.1 INTRODUCTION

The most creative and challenging phase of the life cycle is system and design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in the implementation of the candidate system. The design may be defined as "The process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient details to permit its physical realization".

The design's goal is how the output is to be produced and in what format samples of the output and input are also presented. Second input data and database files have to be designed to meet the requirements of the proposed output. The processing phase is handled through the program construction and testing. Finally, details related to justification of the system and an estimate of the impact of the candidate system on the users and the organization are documented and evaluated by management as a step towards implementation.

The importance of software design can be stated in a single word **"Quality"**. Design provides us with representation of software that can be assessed for quality. Design is the only way that we can accurately translate a customer's requirements into a finished software product or system. Without design, we risk building an unstable system, that might fail, or maybe difficult to test, or one whose quality can't be tested. So, it is an essential phase in the development of a software product.

3.2 SDLC METHODOLOGY

The document plays a vital role in the Software Development Life Cycle (SDLC) as it describes the complete requirements of the system. It can be used by the developers and will be the basic during testing phase. Any

changes made to the requirements in future will have to go through the formal change approval process.

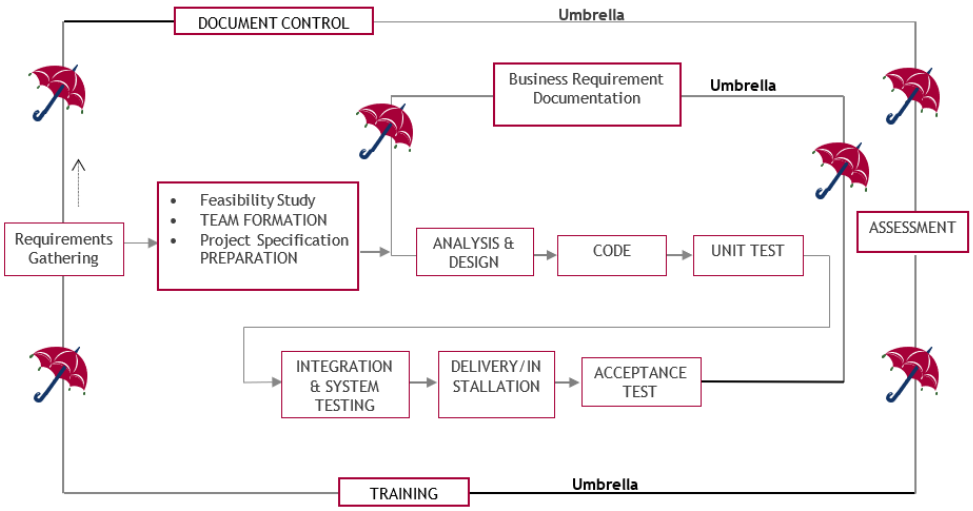


Fig 3.2.1: Umbrella Model

Stages in SDLC

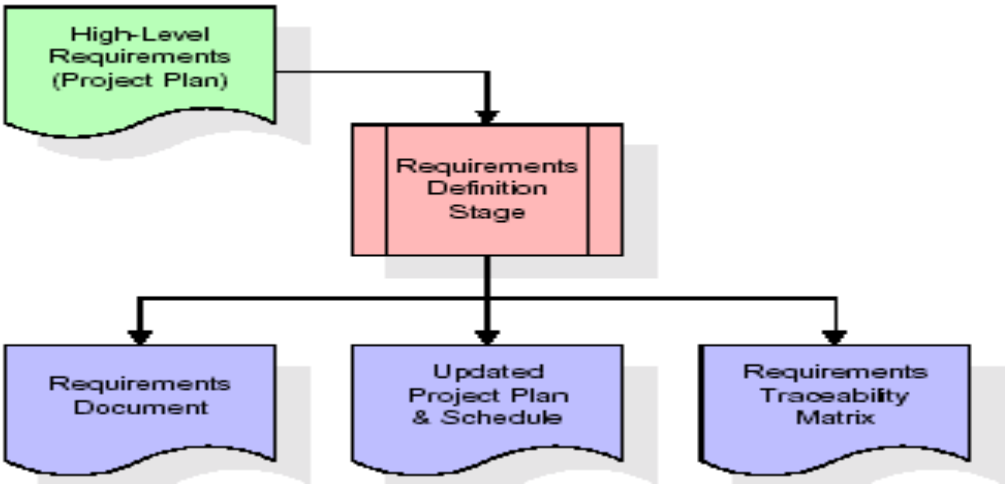
- Requirements Gathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

Requirements Gathering Stage

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements.

These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class

hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

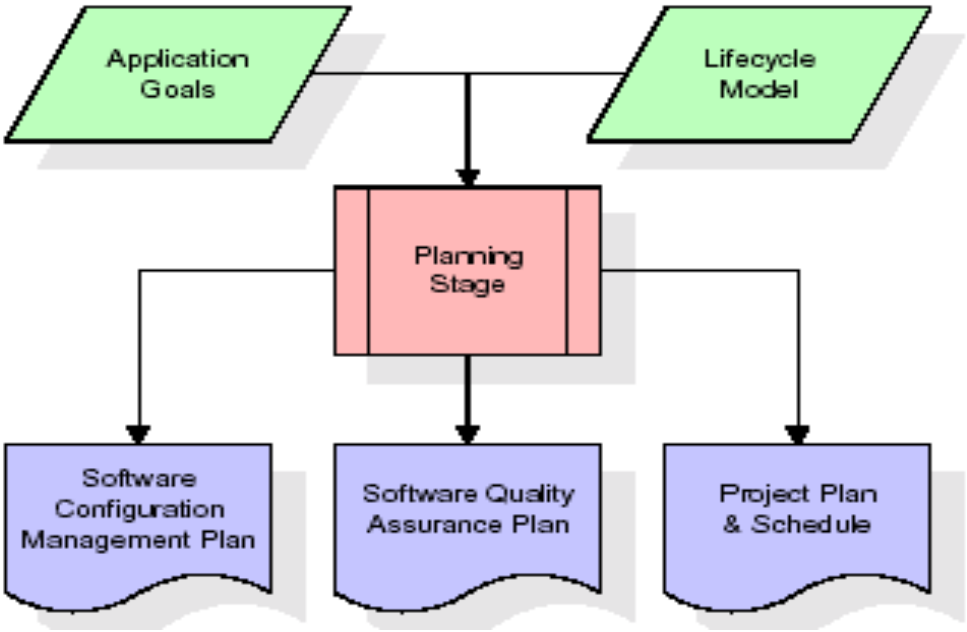
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



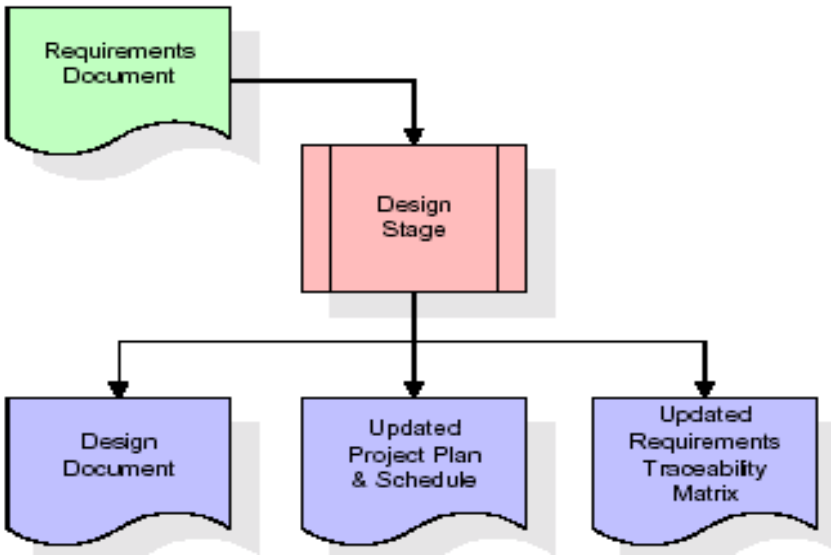
The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product

requirements to be developed during the requirements definition stage flow from one or more of these goals.

The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high-level estimates of effort for the out stages.

Designing Stage

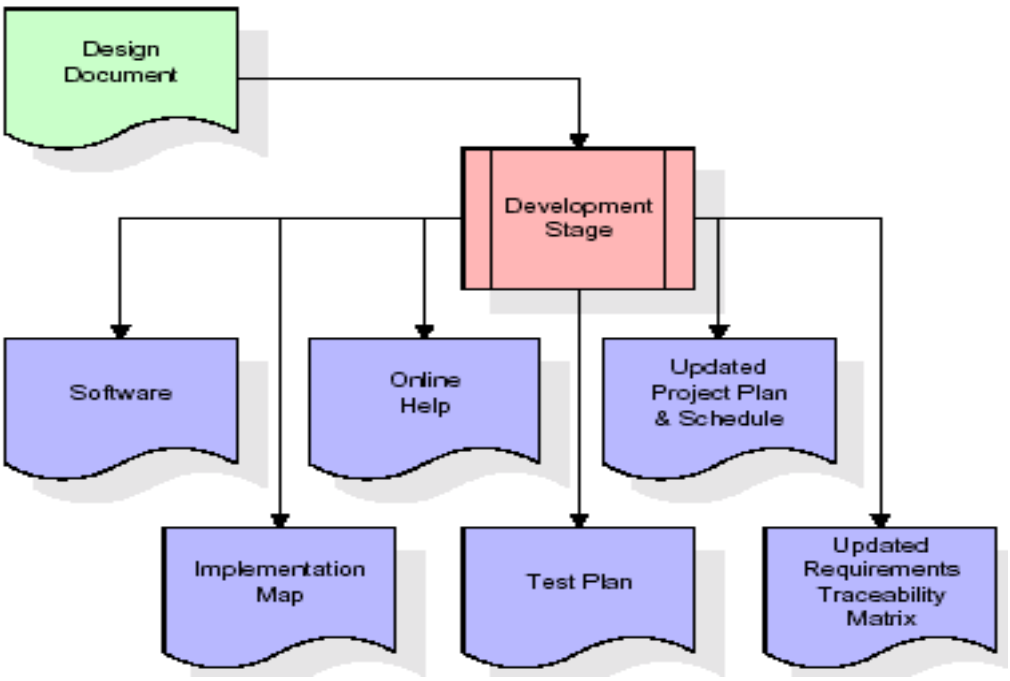
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development (Coding) Stage

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artefacts will be produced. Software artefacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artefacts, and an online help system will be developed to guide users in their interactions with the software.



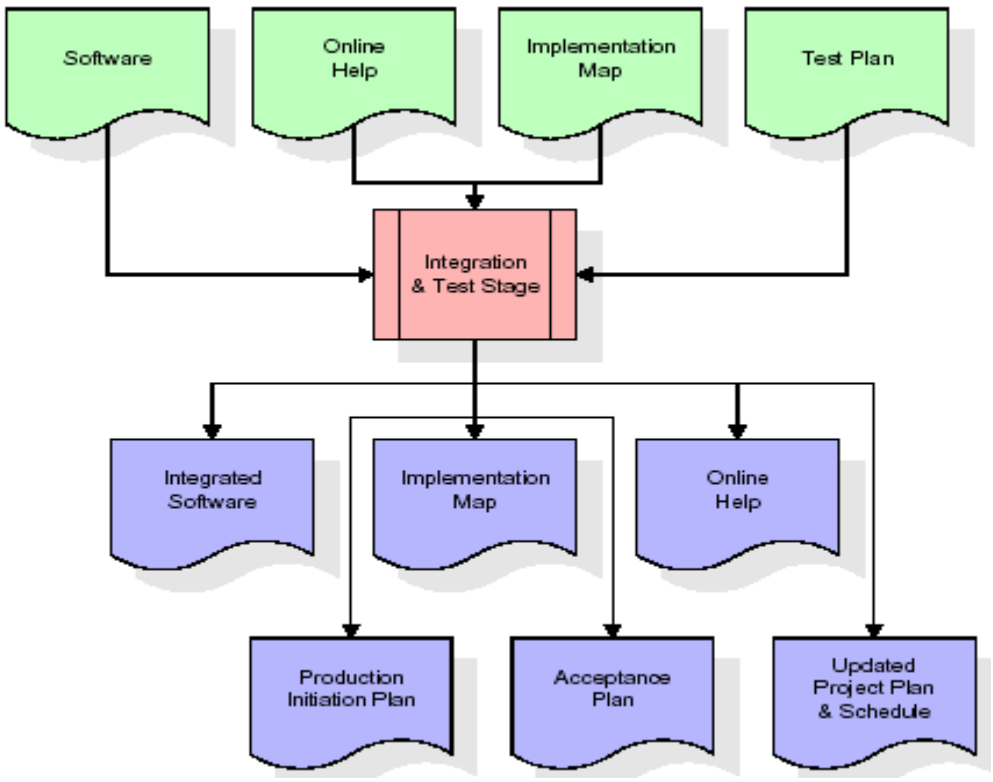
The RTM will be updated to show that each developed artefact is linked to a specific design element, and that each developed artefact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional

set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles.

The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

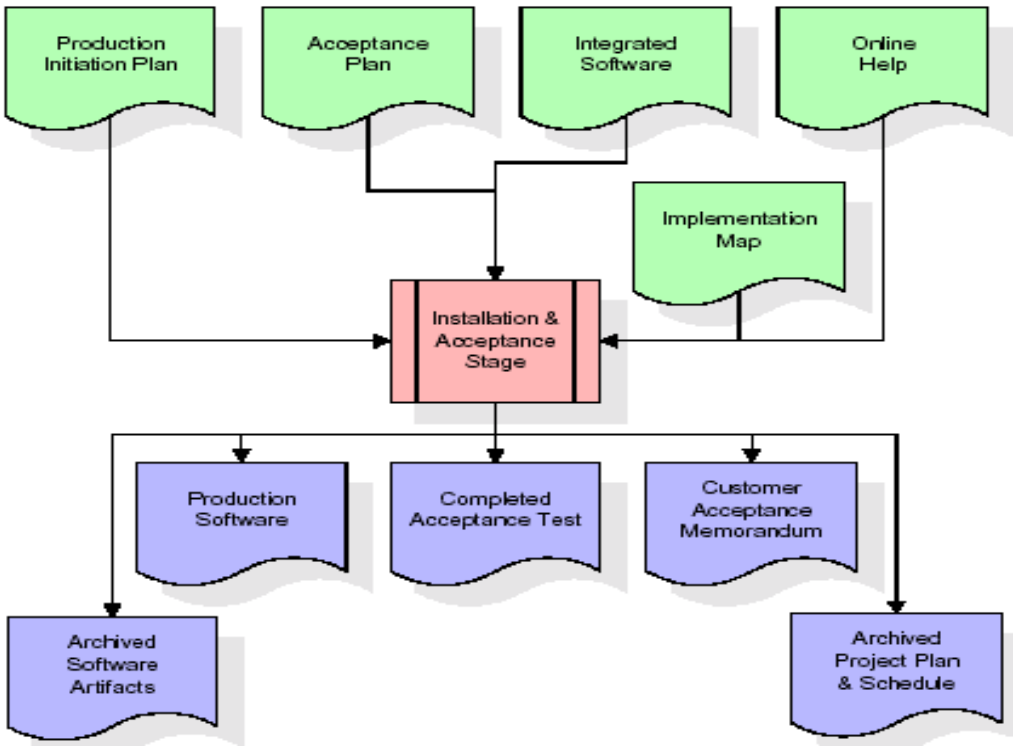


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

Installation & Acceptance Test Stage

During the installation and acceptance test stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the

actual labour data into the project schedule and locks the project as a permanent project record. At this point, the PDR “locks” the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance Stage

Outer rectangle represents maintenance of a project. Maintenance team will start with requirement study, understanding of documentation. Later employees will be assigned work and they will undergo training on that particular assigned category. There is no end for this life cycle. It will be continued so on like an umbrella i.e., there is no ending point to the umbrella sticks.

3.3 INPUT AND OUTPUT REPRESENTATION

INPUT REPRESENTATION

Hand gestures are represented using image processing techniques such as computer vision or machine learning algorithms. The system uses a camera or a depth sensor to capture the hand gesture images, which are then processed and analyzed using computer vision techniques. The hand gesture is represented as a series of data points that represent the position, orientation, and movement of the hand.

OUTPUT REPRESENTATION

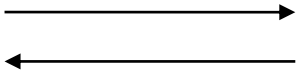
The output for hand gesture recognition can be represented using different techniques, such as displaying the recognized gesture as an image, video, or animation, or text. The system can also provide auditory feedback by converting the recognized hand gesture into a voice or sound output that represents the meaning of the gesture.

3.4 UML DIAGRAMS

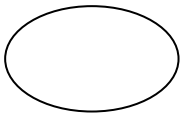
DATA FLOW DIAGRAM

Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also known as Data Flow Graph or a Bubble Chart. The basic notations used to create a DFD are as follows:

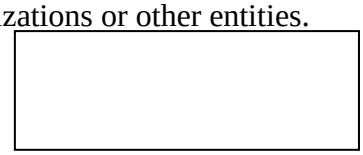
1. **Data Flow:** The movement of data in a specific direction from an origin to a destination.



2. **Process:** People, procedures, or devices that use or produce (transform) the data.



3. **Source:** External sources or destinations of data, which may be people, programs, organizations or other entities.



4. **Data Store:** Here, data is stored or referenced by a process in the system.

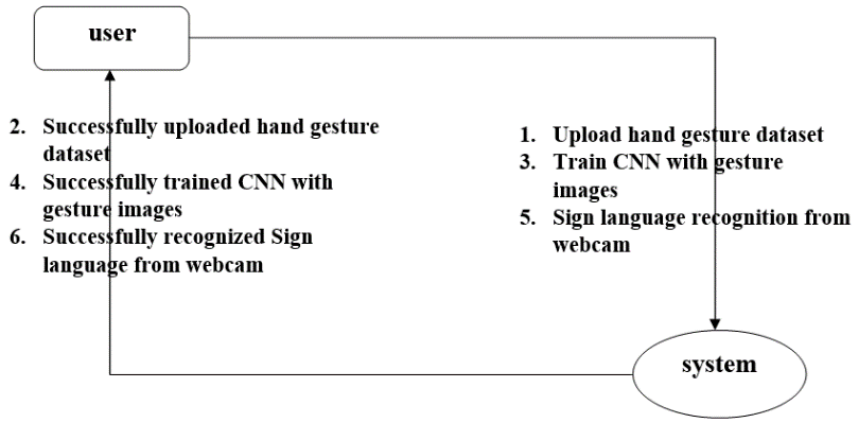


Fig 3.4.1: Data Flow Diagram

UML DIAGRAMS

Unified Modeling Language (UML) is a language available to perform modeling of software. A model is simplification of reality. A model provides the blue print of the system and encompasses detailed plans.

Building Blocks of the UML

The vocabulary of the UML encompasses 3 kinds of building blocks:

- **Things in the UML:** Things are the abstractions that are first-class citizens in a model. There are 4 kinds of things in the UML:
    1. Structure things
    2. Behavioral things
    3. Grouping things
    4. Annotational things
- **Relationships in the UML:** Things can be connected logically or physically with the help of relationships in an object-oriented model. There are 4 kinds of relationships in the UML:
    1. Dependency
    2. Association
    3. Generalization
    4. Realization

- **Diagrams in the UML:** A diagram is a graphical representation of a set of elements. There are 8 kinds of diagrams in the UML:
    1. Class Diagram
    2. Use case Diagram
    3. Sequence Diagram
    4. Collaboration Diagram
    5. Activity Diagram
    6. Component Diagram
    7. Deployment Diagram
    8. State chart Diagram

CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class.
- The middle part contains the attributes of the class.
- The bottom part gives the methods or operations the class can take or undertake.

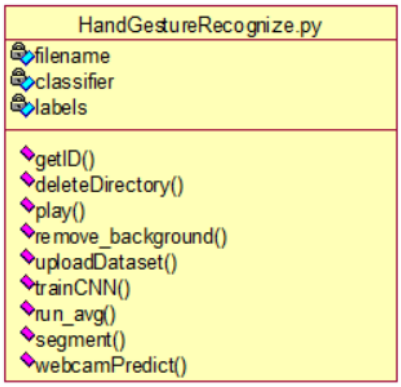


Fig 3.4.2: Class Diagram

USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

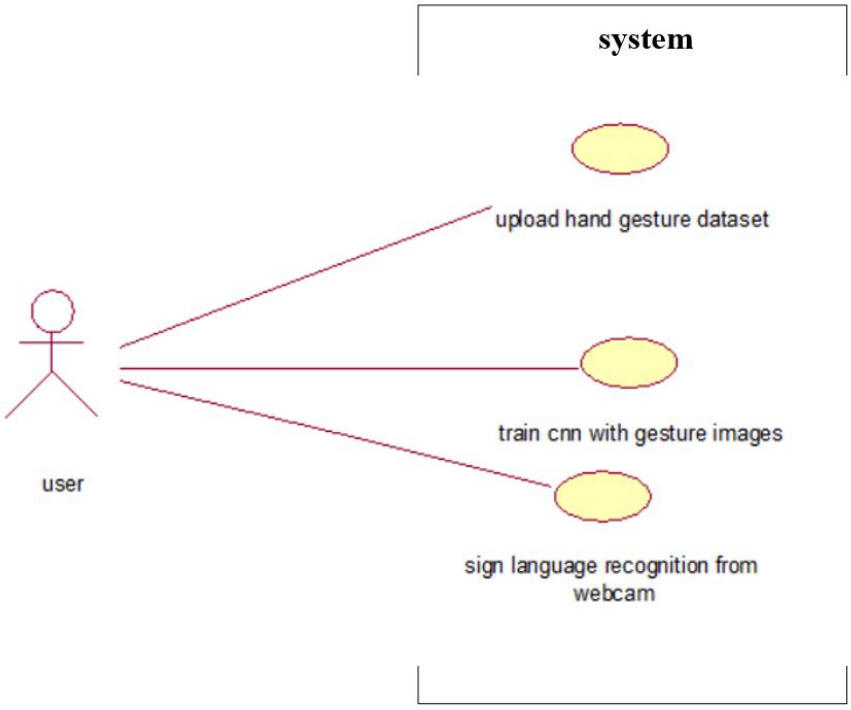


Fig 3.4.3: Use case Diagram

SEQUENCE DIAGRAM

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in 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**, **event scenarios**, and **timing diagrams**.

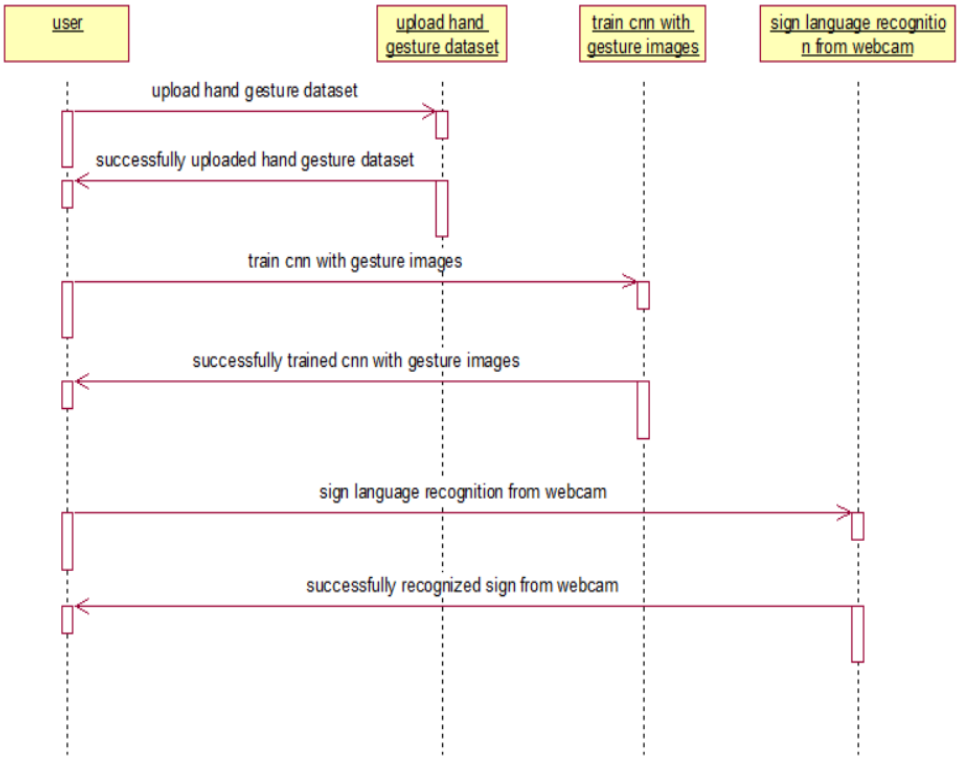


Fig 3.4.4: Sequence Diagram

COLLABORATION DIAGRAM

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.

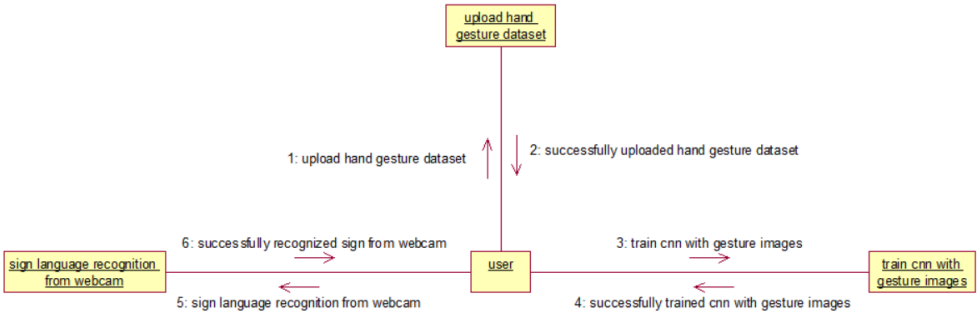


Fig 3.4.5: Collaboration Diagram

ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

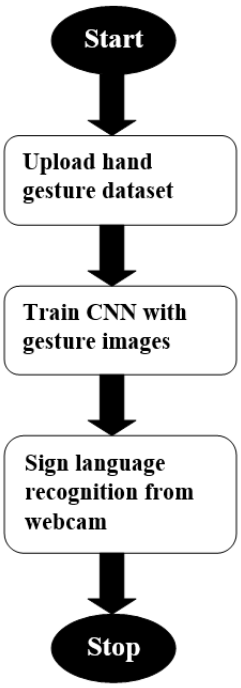


Fig 3.4.6: Activity Diagram

STATE CHART DIAGRAM

A state chart diagram describes a state machine which shows the behavior of classes. It shows the actual changes in state, not processes or commands that create those changes and is the dynamic behavior of objects over time by modelling the lifecycle of objects of each class. It describes how an object is changing from one state to another state. There are 2 states in a state chart diagram: Initial state and Final state.

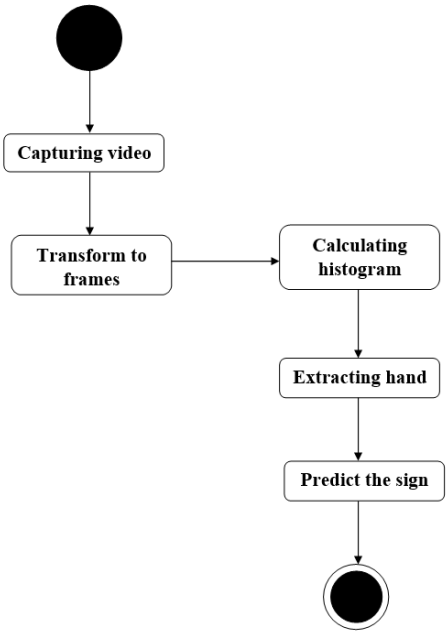


Fig 3.4.7: State Chart Diagram

3.5 SYSTEM ARCHITECTURE

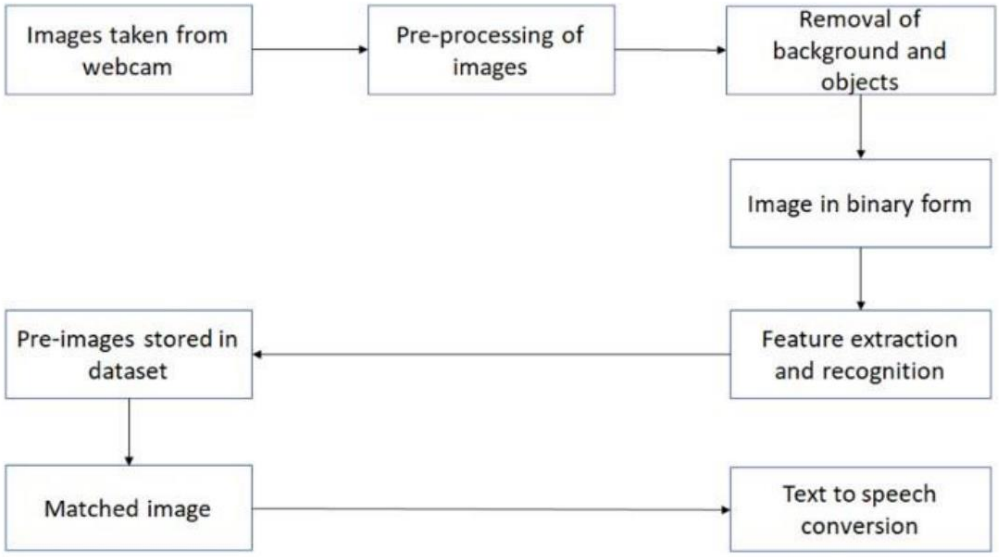


Fig 3.5.1: System Architecture

Fig 3.5.1 illustrates the preferred method's system architecture. It denotes the system's whole physical installation. Each block in the architecture is explained as described below:

- A camera records the user's input.
- After taking a frame of a picture obtained with the webcam, the image is preprocessed and the background noise is eliminated.
- The image has now been converted to binary format, which allows features to be retrieved and identified.
- The obtained features are compared with the dataset.
- The gesture with the highest matching rate is taken into account and transformed into text and voice.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 ALGORITHM DEVELOPMENT

Convolutional Neural Networks Algorithm

A CNN (Convolutional Neural Network) is a Deep Learning technology that may take an image as input and prioritize or discriminate between numerous attributes or objects in the picture. A CNN requires much less preprocessing than other classification methods. Simple procedures are used to manually engineer filters. If given enough training, CNN can learn these filters and attributes. The structure of a CNN was influenced by the architecture of the visual cortex, which is comparable to the connection pattern of neurons in the human brain. Individual neurons can only respond to stimuli that fall inside the receptive field, which is a small fraction of the visual field. A number of comparable fields can be stacked on top of one another to cover the whole visual field.

Algorithm

Step-1: Start.

Step-2: Execute the Python file in the terminal.

Step-3: Click on **Upload Hand Gesture Dataset** button.

Step-4: Select the path for the Dataset and upload the dataset.

Step-5: Preprocess the dataset.

Step-6: Click on **Train CNN with Gesture Images** button.

Step-7: Click on **Hand Gesture Recognition from Webcam** button.

Step-8: Image Extraction from Webcam.

Step-9: Convert image into Binary or gray format.

Step-10: Background Removal of image.

Step-11: Feature extraction from image.

Step-12: Result is displayed on the screen in text.

Step-13: Text is converted into audio.

Step-14: Go to step-8 until the window is closed.

Step-15: End.

Implementation of the Algorithm

Implementation of the proposed approach consists of following 4 steps:

1. **Preprocess the Hand Gesture Dataset:** In the first step, a function was written that saved the path to the hand gestures dataset in a variable and loaded all image-containing folders into arrays. To remove the noise, the mean filter method is utilized. In this simple sliding window, the average of the window's pixel values replaces the centre value. For segmentation, the threshold method is used. It divides pixels in an image by comparing each pixel's intensity to a preset value. A contour is just a line that links all sections of the same color or intensity.
2. **Modeling and Training Gesture Images:** CNN Algorithm is employed to model the gesture images. The basic methods employed by CNN are: convolution, maximum pooling, and flattening. **Convolution** is the technique of altering an image by applying a kernel over the entire picture to each pixel and its local neighbours. **Max Pooling** is a convolution method in which the kernel convolutions the region with the greatest value. The process of converting data into a one-dimensional array for usage in the following layer is known as **flattening**.
3. **Webcam Image Extraction and Feature detection:** At this stage, picture frames are read from the camera and image processing algorithms are applied to them, after which characteristics are retrieved from the image frames and motions are detected by comparing them to those in the dataset.

4. **Voice and text translation of hand gestures:** Both text and speech versions of the predicted gesture are produced. The text is shown on the screen and also fed into Google Voice to provide an audio output.

4.2 TECHNOLOGY DESCRIPTION

INTRODUCTION TO PYTHON

This project uses **Python** version **3.7**. Python is a high-level programming language that is widely used in various fields, including artificial intelligence and computer vision. Python has a very simple and elegant syntax. It is much easier to read and write Python programs compared to other languages. Python has a large number of libraries and frameworks that can be used to build complex and sophisticated applications quickly and efficiently.

In this project, Python will be used to develop the computer vision algorithms that recognize and classify hand gestures. Python libraries such as **OpenCV** and **TensorFlow** can be used to implement these algorithms. OpenCV is a computer vision library that provides a wide range of functions for image processing and object detection, while TensorFlow is a machine learning library that is widely used for deep learning tasks such as image recognition and natural language processing.

Python will also be used to develop the voice conversion system. Text-To-Speech (TTS) libraries such as **pyttsx3** or **gTTS** can be used to convert the text generated by the hand gesture recognition system into audible speech. Additionally, Python's audio processing libraries, such as **PyAudio** and **SpeechRecognition**, can be used to capture audio input from a microphone and convert it to text, enabling the system to recognize speech and convert it to text for further processing.

Overall, Python's versatility and powerful libraries make it an excellent choice for developing the project. Its ability to handle both computer vision and audio processing tasks efficiently will be critical in building a system that is accurate and reliable for users.

INTRODUCTION TO OPENCV

This project uses **OpenCV** module version **4.5.1.48**. OpenCV is a popular open-source library for computer vision applications that provides a wide range of functions for image and video processing. It has various algorithms and techniques that can be used to detect, recognize, and track objects in real-time, making it an ideal choice for building a hand gesture recognition system.

In this project, OpenCV will be used to recognize and track the movement of the user's hand, which will be captured by a camera. OpenCV can be used to extract the features of the hand and track its movement using techniques such as contour detection, thresholding, and edge detection. These features can then be used to recognize various hand gestures used in sign language.

OpenCV also offers a wide range of pre-trained models and machine learning algorithms that can be used for gesture recognition. These models can be trained using large datasets of hand gestures, which can help the system learn to recognize new gestures accurately.

OpenCV can also be used to filter and preprocess the captured video or image data to remove noise and enhance the quality of the input. This is particularly important in low-light or noisy environments where the quality of the input may be poor.

Overall, OpenCV is an essential technology for the development of the project. Its ability to detect and track the user's hand, recognize gestures accurately, and preprocess input data makes it a valuable tool for building a robust and reliable hand gesture recognition system.

INTRODUCTION TO TENSORFLOW

This project uses **TensorFlow** module version **1.14.0**. TensorFlow is a popular open-source library for building and training machine learning models. In this project, TensorFlow will be used to develop machine learning algorithms that can recognize hand gestures accurately.

TensorFlow offers a wide range of tools and functions for building and training deep learning models. In the case of this project, TensorFlow's convolutional neural network (CNN) architecture can be used to train a model to recognize hand gestures accurately. The CNN model can be trained using large datasets of hand gestures, enabling it to learn and recognize new gestures with high accuracy.

TensorFlow also provides pre-trained models that can be used as a starting point for building the gesture recognition system. These pre-trained models can be fine-tuned using transfer learning techniques to improve their accuracy and adapt them to the specific requirements of the project.

In addition to its machine learning capabilities, TensorFlow can also be used to implement the voice conversion system. TensorFlow's text-to-speech (TTS) library can be used to convert text generated by the gesture recognition system into audible speech. Additionally, TensorFlow's speech recognition library can be used to capture audio input from a microphone and convert it to text, enabling the system to recognize speech and convert it to text for further processing.

Overall, TensorFlow is a powerful technology for the development of the project. Its ability to build and train machine learning models and its pre-built models make it an ideal choice for developing the gesture recognition system. TensorFlow's TTS and speech recognition libraries can be used to develop the voice conversion system, enabling the system to convert text to speech and vice versa accurately.

4.3 SAMPLE CODE

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog
```

```
from tkinter.filedialog import askopenfilename
import cv2
import random
import numpy as np
from keras.utils.np_utils import to_categorical
from keras.layers import MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import model_from_json
import pickle
import os
import imutils
from gtts import gTTS
from playsound import playsound
import os
from threading import Thread


main = tkinter.Tk()
main.title("Hand Gesture Recognition and Voice Conversion using CNN")
main.geometry("1200x600")


global filename
```

```
global classifier


bg = None

playcount = 0


#names = ['Palm','I','Fist','Fist Moved','Thumbs up','Index','OK','Palm
Moved','C','Down']

names = ['C','Thumbs Down','Fist','I','Ok','Palm','Thumbs up']


def getID(name):
    index = 0
    for i in range(len(names)):
        if names[i] == name:
            index = i
            break
    return index


bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)


def deleteDirectory():
    filelist = [ f for f in os.listdir('play') if f.endswith(".mp3") ]
    for f in filelist:
        os.remove(os.path.join('play', f))
```

```
def play(playcount,gesture):
    class PlayThread(Thread):
        def __init__(self,playcount,gesture):
            Thread.__init__(self)
            self.gesture = gesture
            self.playcount = playcount

        def run(self):
            t1 = gTTS(text=self.gesture, lang='en', slow=False)
            t1.save("play/"+str(self.playcount)+".mp3")
            playsound("play/"+str(self.playcount)+".mp3")

    newthread = PlayThread(playcount,gesture)
    newthread.start()

def remove_background(frame):
    fgmask = bgModel.apply(frame, learningRate=0)
    kernel = np.ones((3, 3), np.uint8)
    fgmask = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=fgmask)
    return res

def uploadDataset():
```

```
    global filename
    global labels
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");


def trainCNN():
    global classifier
    text.delete('1.0', END)
    X_train = np.load('model/X.txt.npy')
    Y_train = np.load('model/Y.txt.npy')
    text.insert(END,"CNN is training on total images: "+str(len(X_train))+"\n")
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            classifier = model_from_json(loaded_model_json)
        classifier.load_weights("model/model_weights.h5")
        classifier._make_predict_function()
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
```

```
        f.close()
        acc = data['accuracy']
        accuracy = acc[9] * 100
        text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))
    else:
        classifier = Sequential()
        classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
        classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
        classifier.add(Flatten())
        classifier.add(Dense(output_dim = 256, activation = 'relu'))
        classifier.add(Dense(output_dim = 5, activation = 'softmax'))
        print(classifier.summary())
        classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
        hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=10, shuffle=True, verbose=2)
        classifier.save_weights('model/model_weights.h5')
        model_json = classifier.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        f = open('model/history.pckl', 'wb')
```

```
        pickle.dump(hist.history, f)
        f.close()
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[9] * 100
        text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))


def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)


def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    ( cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)


def webcamPredict():
    global playcount
    oldresult = 'none'
    count = 0
    fgbg2 = cv2.createBackgroundSubtractorKNN();
    aWeight = 0.5
    camera = cv2.VideoCapture(0)
    top, right, bottom, left = 10, 350, 325, 690
    num_frames = 0
    while(True):
        (grabbed, frame) = camera.read()
        frame = imutils.resize(frame, width=700)
        frame = cv2.flip(frame, 1)
        clone = frame.copy()
        (height, width) = frame.shape[:2]
        roi = frame[top:bottom, right:left]
        gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

```
        gray = cv2.GaussianBlur(gray, (41, 41), 0)
        if num_frames < 30:
            run_avg(gray, aWeight)
        else:
            temp = gray
            hand = segment(gray)
            if hand is not None:
                (thresholded, segmented) = hand
                cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
                #cv2.imwrite("test.jpg",temp)
                #cv2.imshow("Thesholded", temp)
                #ret, thresh = cv2.threshold(temp, 150, 255, cv2.THRESH_BINARY
+ cv2.THRESH_OTSU)
                #thresh = cv2.resize(thresh, (64, 64))
                #thresh = np.array(thresh)
                #img = np.stack((thresh,)*3, axis=-1)
                roi = frame[top:bottom, right:left]
                roi = fgbg2.apply(roi);
                cv2.imwrite("test.jpg",roi)
                #cv2.imwrite("Dataset/Fist/"+str(count)+".png",roi)
                #count = count + 1
                #print(count)
                img = cv2.imread("test.jpg")
                img = cv2.resize(img, (64, 64))
```

```
            img = img.reshape(1, 64, 64, 3)
            img = np.array(img, dtype='float32')
            img /= 255
            predict = classifier.predict(img)
            value = np.amax(predict)
            cl = np.argmax(predict)
            result = names[np.argmax(predict)]
            if value >= 0.99:
                print(str(value)+" "+str(result))
                cv2.putText(clone, 'Gesture Recognized as: '+str(result), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
                if oldresult != result:
                    play(playcount,result)
                    oldresult = result
                    playcount = playcount + 1
            else:
                cv2.putText(clone, '', (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
            cv2.imshow("video frame", roi)
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)
        num_frames += 1
        cv2.imshow("Video Feed", clone)
        keypress = cv2.waitKey(1) & 0xFF
        if keypress == ord("q"):
```

```
            break
    camera.release()
    cv2.destroyAllWindows()


font = ('times', 20, 'bold')
title = Label(main, text='Hand Gesture Recognition and Voice Conversion using CNN',anchor=W, justify=CENTER)
title.config(bg='yellow4', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)



font1 = ('times', 13, 'bold')
upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)
upload.place(x=50,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='yellow4', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50,y=150)
```

```
markovButton = Button(main, text="Train CNN with Gesture Images",
command=trainCNN)

markovButton.place(x=50,y=200)

markovButton.config(font=font1)


predictButton = Button(main, text="Sign Language Recognition from
Webcam", command=webcamPredict)

predictButton.place(x=50,y=250)

predictButton.config(font=font1)


font1 = ('times', 12, 'bold')

text=Text(main,height=15,width=78)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)


deleteDirectory()

main.config(bg='magenta3')

main.mainloop()
```

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for the software engineer.

Testing Objectives

- **Validation:** Ensuring that the software meets the specified requirements and performs its intended functions.
- **Verification:** Checking that the software meets the quality standards and guidelines set by the organization.
- **Reliability:** Ensuring that the software is reliable and works consistently under various conditions.
- **Usability:** Ensuring that the software is user-friendly and easy to use.
- **Security:** Ensuring that the software is secure and protected against unauthorized access or attacks.
- **Performance:** Ensuring that the software performs efficiently and quickly under various workloads.
- **Maintainability:** Ensuring that the software is easy to maintain and update.

Testing Principles

- Testing should start early in the software development life cycle.
- Testing should be conducted in a systematic and planned manner.
- Testing should be based on the requirements and specifications.
- Test cases should be designed to cover all possible scenarios and conditions.
- Testing should be independent of the development process.
- Testing should be repeatable and consistent.

- Defects and issues should be tracked and managed throughout the testing process.

5.2 TESTING STRATEGIES

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation is given for the fields. Then the system testing takes place which makes sure that all components of the system function as a unit. The test data should be chosen such that it passes through all possible conditions.

The following is the description of the testing strategies, which were carried out during the testing period.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. When the software is developed before it is given to user to use, the software must be tested whether it is solving the purpose for which it is developed or not. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom to top starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared

with the results that are prepared manually. The comparison shows that the resulting proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system, the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system, all the modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

Acceptance Testing

Acceptance Testing would involve testing the software with end-users or stakeholders to verify that it meets their requirements and expectations. This would include verifying that the software correctly recognizes and converts hand gestures and speech, and that it provides an intuitive and user-friendly interface. The acceptance testing would also verify that the software meets accessibility requirements and is accessible to people with disabilities. The testing would be conducted in a real-world environment to simulate the actual usage scenarios of the software.

Security Testing

Security testing involves testing the application's security features to ensure that it is protected against unauthorized access and attacks. For this project, security testing can be used to test the application's security features, ensuring that it is protected against unauthorized access and attacks.

5.3 TEST CASES

The following table illustrates various test cases of this project:

Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status and Test Priority
			Step	Expected	Actual	
01	Upload Hand Gesture Dataset	Test whether the Hand Gesture Dataset is uploaded or not	If the Hand Gesture Dataset may not be uploaded	We cannot do further operations	Hand Gesture Dataset uploaded. We will do further operations	High and High
02	Train CNN with Gesture Images	Test whether the CNN is trained with Gesture Images or not	If the CNN may not be trained with Gesture Images	We cannot do further operations	CNN is trained with Gesture Images. We will do further operations	High and High
03	Recognize Gesture from Webcam	Test whether the Gesture is recognized from Webcam or not	If the Gesture from Webcam may not be recognized	We cannot do further operations	Gesture is recognized from Webcam. We will do further operations	High and High

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 EXECUTION SCREENSHOTS

The project contains the folder named “Dataset” which consists of 7 different hand gestures namely: **‘C’, ‘Thumbs Down’, ‘Fist’, ‘I’, ‘Ok’, ‘Palm’, ‘Thumbs Up’.** The folder ‘C’ which consists of various ‘C’ hand gesture images used for training is as shown in Fig 6.1:

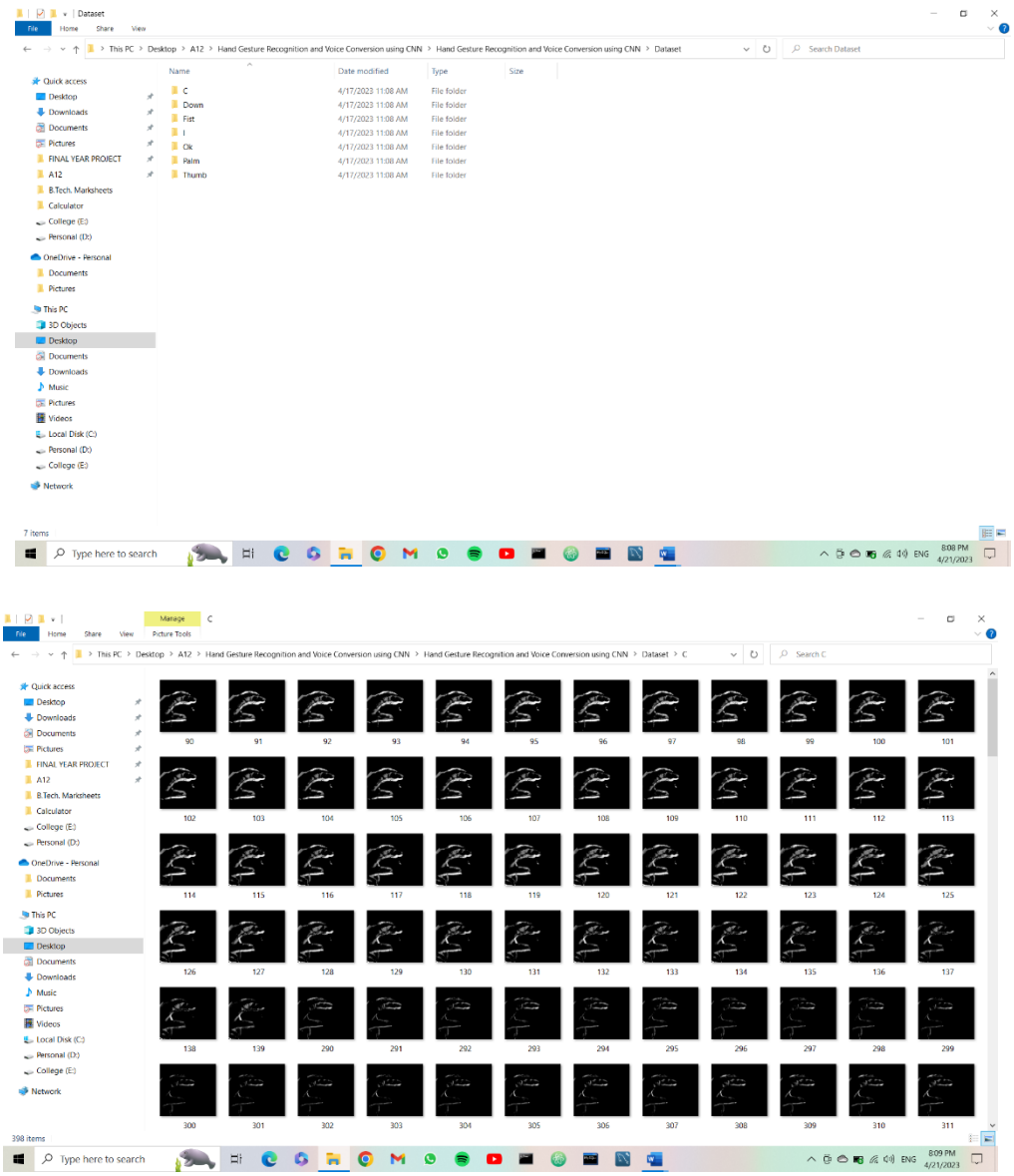


Fig 6.1: Dataset used for Training

Similarly, there will be images used for training the remaining gestures: 'Thumbs Down', 'Fist', 'I', 'Ok', 'Palm', 'Thumbs Up' in their respective folders.

Step-1: Double click on **run.bat** file to execute the project and the following window is opened which consists of 3 buttons namely,

1. **Upload Hand Gesture Dataset**
2. **Train CNN with Gesture Images** and
3. **Sign Language Recognition from Webcam** as shown in Fig 6.2:

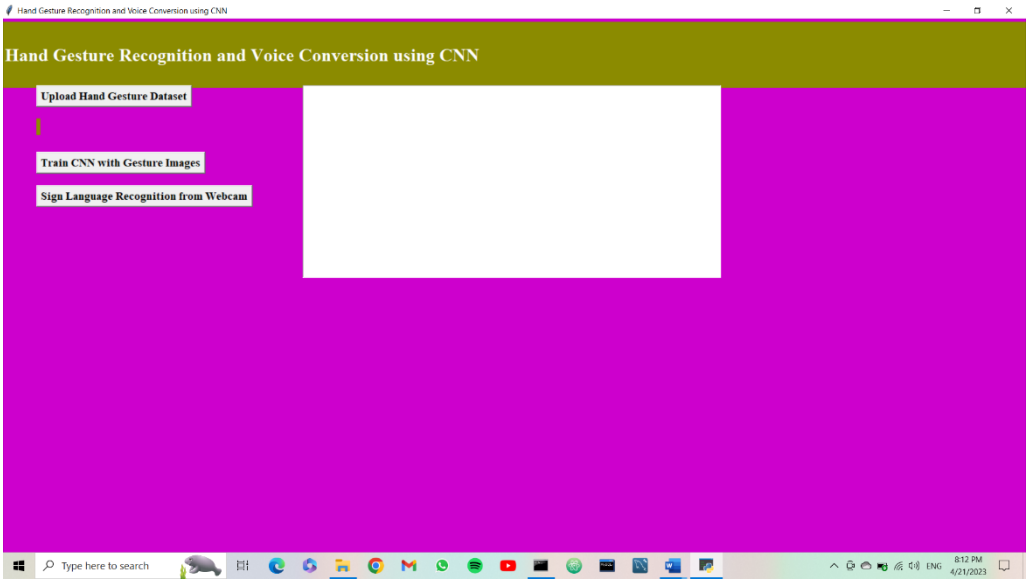


Fig 6.2: Application Interface

Step-2: Click on **Upload Hand Gesture Dataset** button to upload the dataset present in the "Dataset" folder in the current working directory as shown in Fig 6.3:

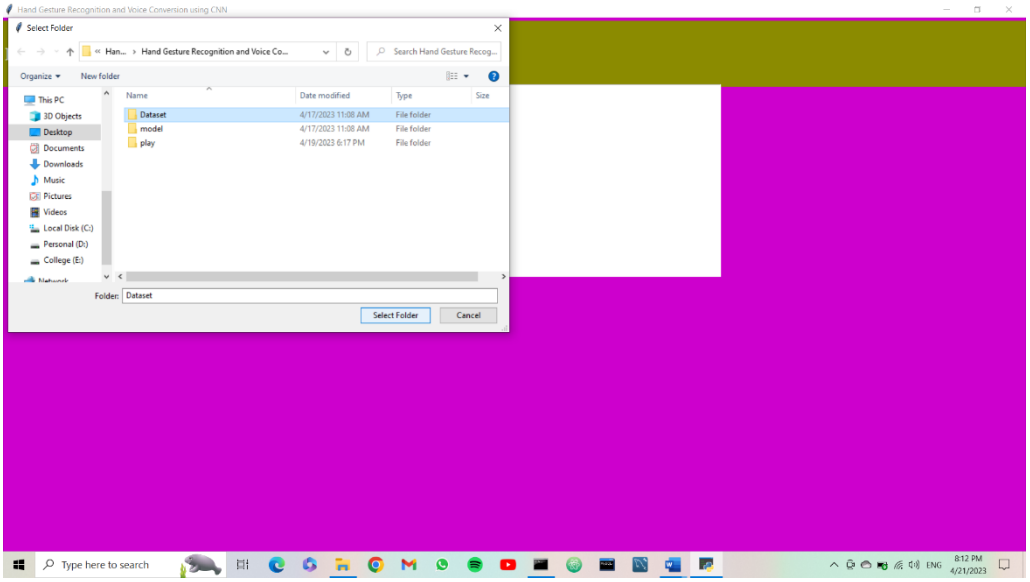


Fig 6.3: Upload Dataset for Training

After uploading the dataset successfully, a message is displayed with the path of the dataset uploaded as **"Dataset loaded"** as shown in Fig 6.4:

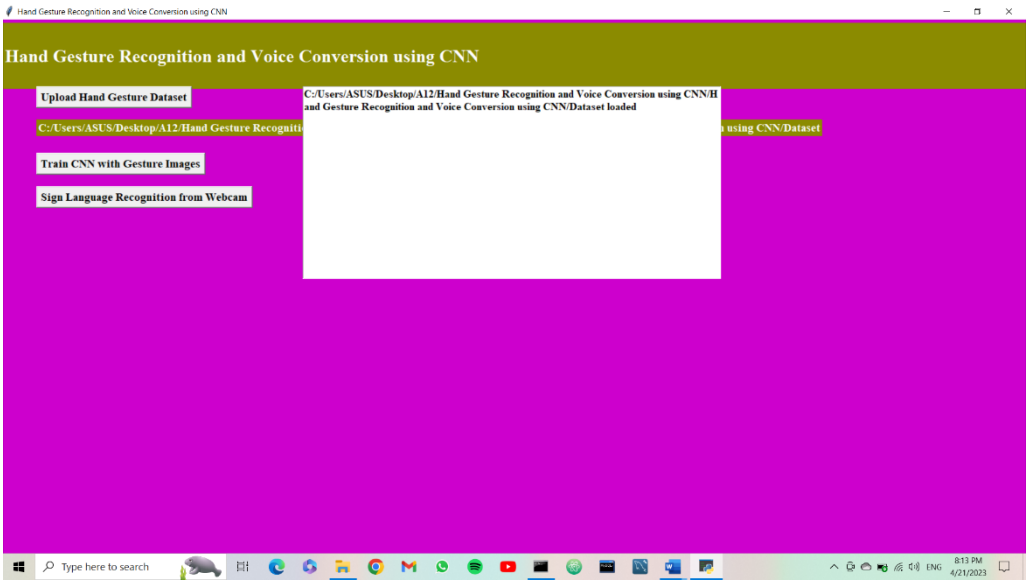


Fig 6.4: Dataset Loaded message

Step-3: Click on **Train CNN with Gesture Images** button to train the uploaded dataset using CNN algorithm and find the Model Prediction Accuracy as shown in Fig 6.5:

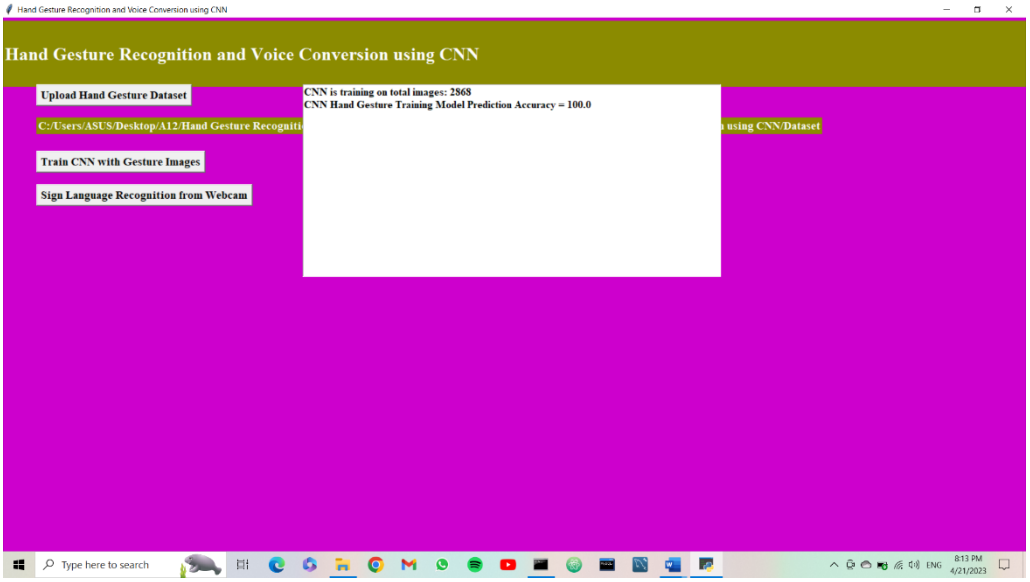


Fig 6.5: Dataset Trained message

Step-4: Click on **Sign Language Recognition from Webcam** button to start giving hand gestures so that a **Video Feed** and a **Video Frame** are displayed as shown in Fig 6.6:

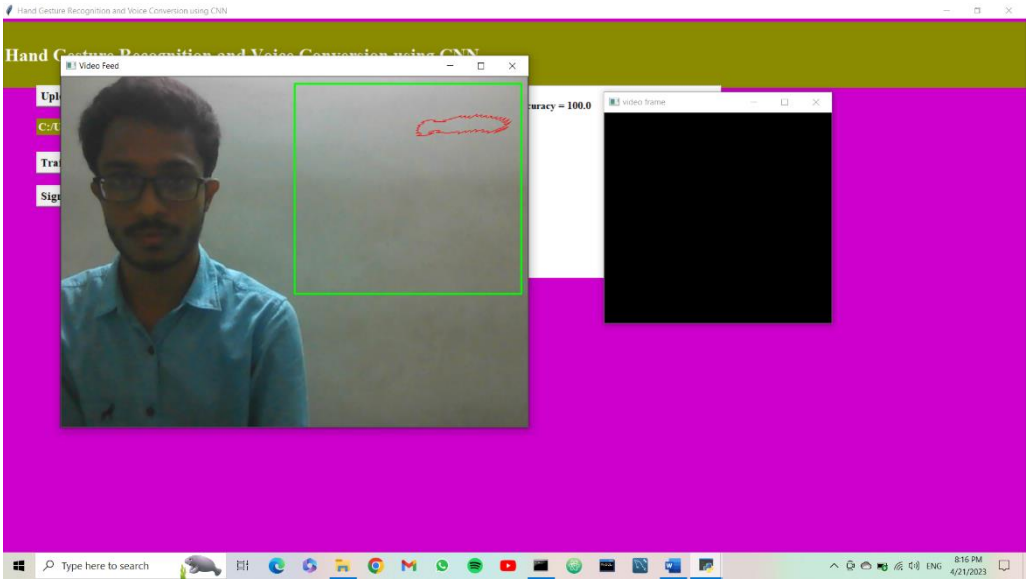


Fig 6.6: Empty Output screen

While giving the gestures as input, there will be 2 kinds of output generated for each gesture:

1. **Text corresponding to the recognized gesture:** The text is displayed as **Gesture Recognized as: Palm** in the Video Feed and a grayscale image of the gesture can be seen in the Video Frame.
2. **Voice corresponding to the recognized gesture:** The voice is heard as **Palm** either from the built-in speakers of the system or from the speakers connected externally to the system.

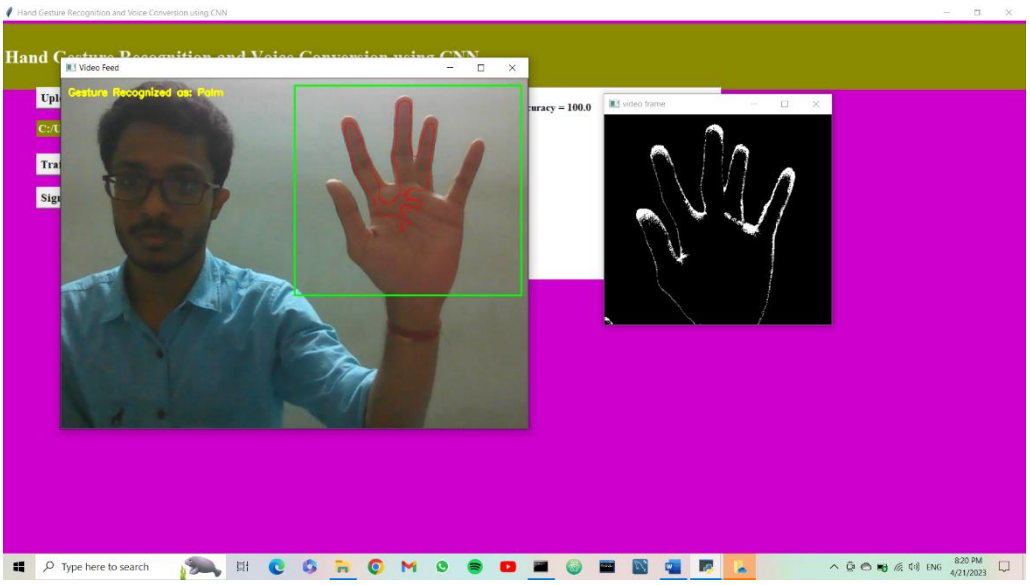


Fig 6.7: Palm Gesture Recognition

In the similar manner, all the remaining gestures i.e., ‘C’, ‘Thumbs Down’, ‘Fist’, ‘I’, ‘Ok’, ‘Thumbs Up’ can be recognized by the system with both text and voice output as shown in the Fig 6.8 to Fig 6.13:

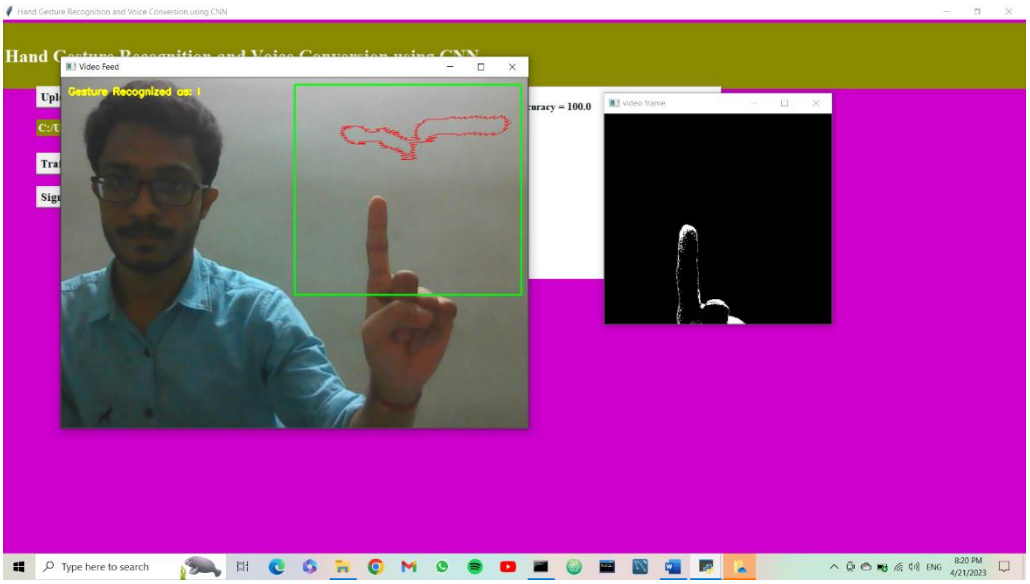


Fig 6.8: I Gesture Recognition

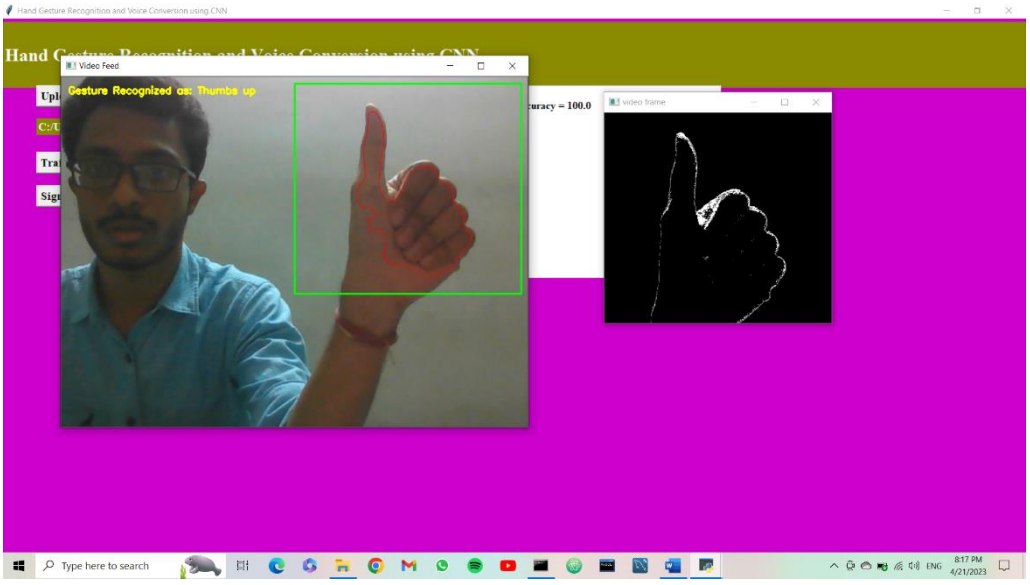


Fig 6.9: Thumbs Up Gesture Recognition

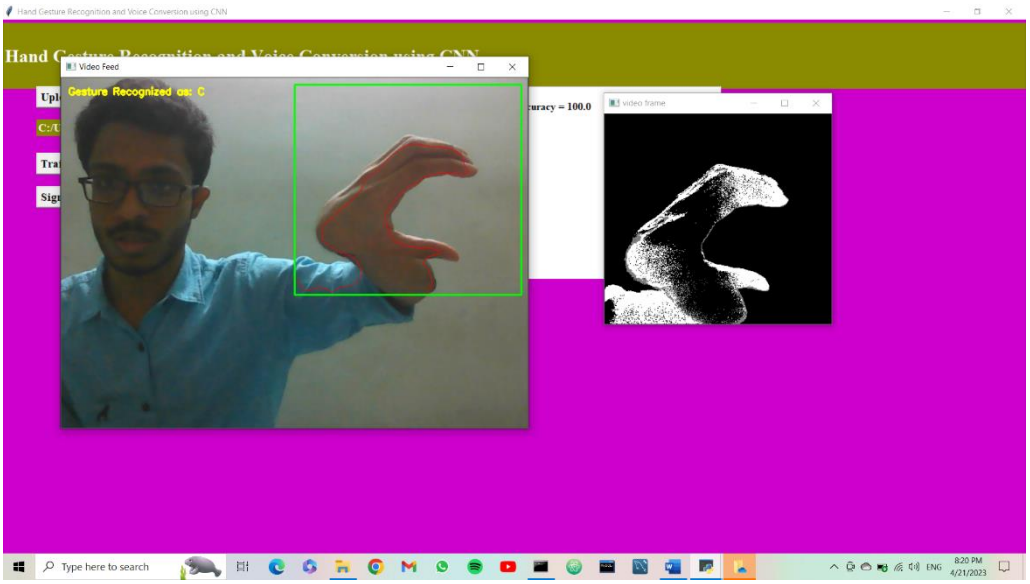


Fig 6.10: C Gesture Recognition

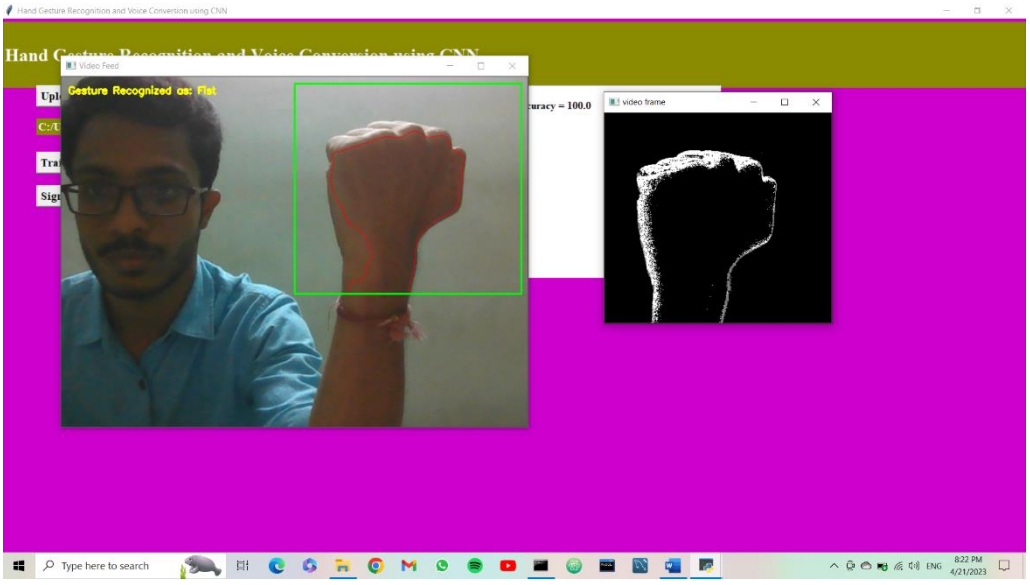


Fig 6.11: Fist Gesture Recognition

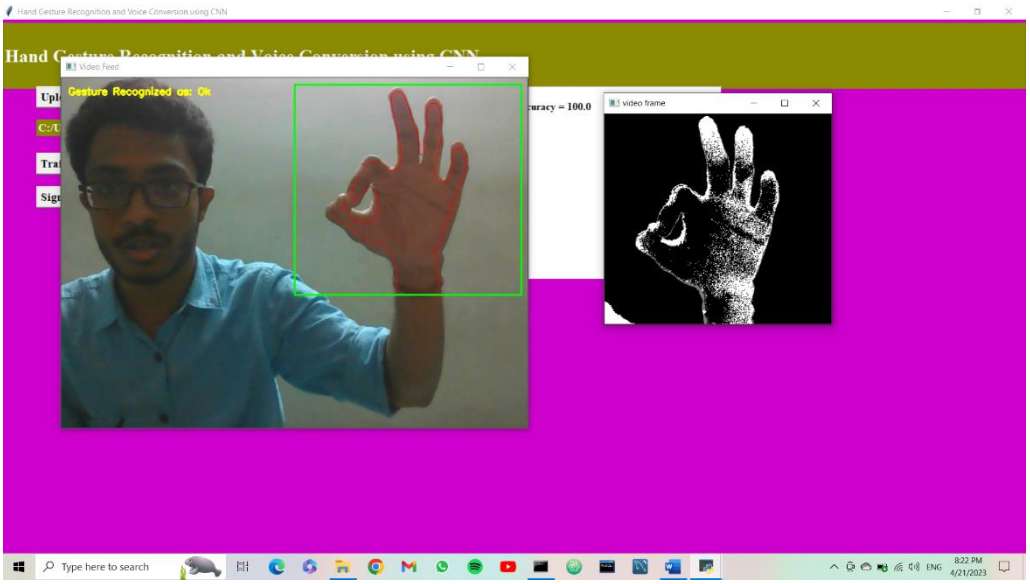


Fig 6.12: Ok Gesture Recognition

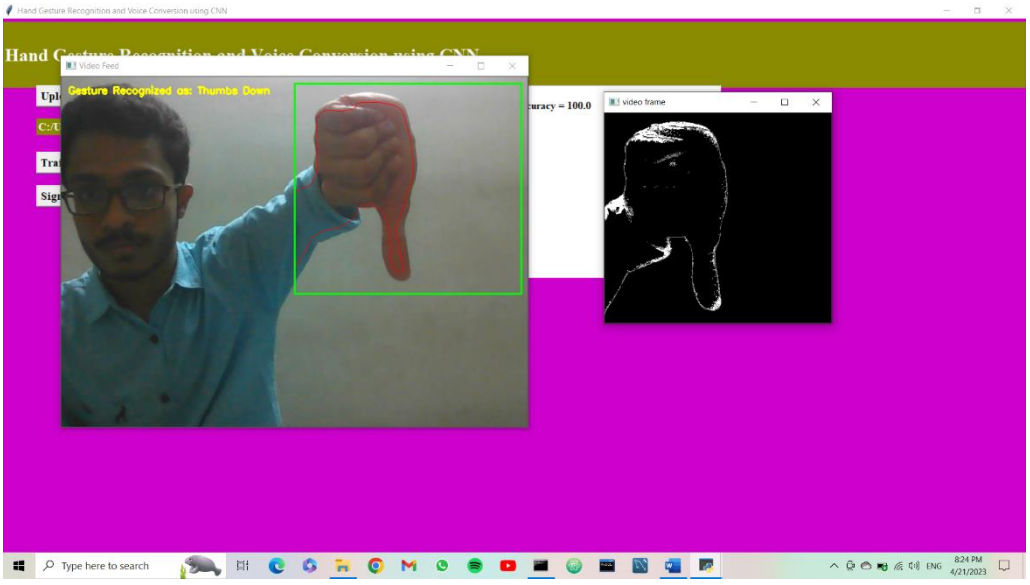


Fig 6.13: Thumbs Down Gesture Recognition

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

We developed a CNN model for sign language recognition. Our model learns and extracts both spatial and temporal features by performing 3D convolutions. The developed deep learning architecture extracts multiple types of information from adjacent input frames and then performs convolution and subsampling separately. Hand gestures to speech and text translation have been employed in this approach to allow the reduction of hardware components as a simple and practical technique to establish human-computer interaction. Overall, the approach tries to help people in need while maintaining societal significance. The system's user-friendly design assures that it can be used by anybody without difficulty or complication. The application is low-cost and does not require the use of costly technologies. The communication gap between blind and deaf persons is bridged using this technology.

7.2 FUTURE ENHANCEMENT

To boost user engagement and make the system more robust, the app may be linked with other mobile and IoT devices. One possible future enhancement is the integration of artificial intelligence (AI) to enhance the accuracy of hand gesture recognition and voice conversion. With AI, the system can learn from previous interactions and improve its recognition and conversion abilities.

Another potential future enhancement is the development of a mobile application that can be easily accessed and used by individuals who are deaf and mute. The mobile application can allow for more convenient and accessible communication between the deaf and mute community and the hearing community. Furthermore, the project can also be expanded to include other sign languages and dialects, making it more inclusive and accessible to a wider audience.

CHAPTER 8

REFERENCES

[1] P. Surekha, N. Vitta, P. Duggirala and V. S. S. Ambadipudi, "Hand Gesture Recognition and voice, text conversion using," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 167-171, doi: 10.1109/ICAIS53314.2022.9743064.

[2] Shinde, Shweta S., Rajesh M. Autee, and Vitthal K. Bhosale. "Real-time two-way communication approach for hearing impaired and dumb person based on image processing." Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on. IEEE, 2016.

[3] Shangeetha, R. K., V. Valliammai, and S. Padmavathi. "Computer vision-based approach for Indian Sign Language character recognition." Machine Vision and Image Processing (MVIP), 2012 International Conference on. IEEE, 2012.

[4] Sood, Anchal, and Anju Mishra. "AAWAAZ: A communication system for deaf and dumb." Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2016 5th International Conference on. IEEE, 2016.

[5] Ahire, Prashant G., et al. "Two Way Communicator between Deaf and Dumb People and Normal People." Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on. IEEE, 2015.

[6] S. Karthik, S. Arumugam, and R. Thirumalvalavan, "Hand Gesture Recognition for Sign Language Translation System: A Review," in 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Chennai, India, 2018, pp. 551-555. doi: 10.1109/ICOEI.2018.8553827.

[7] K. H. Lin, C. H. Tsai, and C. C. Wang, "Real-time American Sign Language hand gesture recognition using a depth-sensing camera," in 2014 IEEE Symposium on Multimedia, Taichung, Taiwan, 2014, pp. 185-190. doi: 10.1109/ISM.2014.39.

[8] K. M. Baharul Islam, T. Bora, and D. K. Bhattacharyya, "Design of an Efficient Hand Gesture Recognition System for Sign Language Communication," in 2018 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2018, pp. 906-909. doi: 10.1109/ICICCT.2018.8473139.

[9] R. K. Sahu and S. Mishra, "Real-Time Hand Gesture Recognition for Sign Language Translation System," in 2019 2nd International Conference on Computing, Communication, and Automation (ICCCA), Greater Noida, India, 2019, pp. 784-789. doi: 10.1109/CCAA.2019.8776967.

[10] N. S. Saini and S. Saini, "Real Time Sign Language Recognition using Image Processing Techniques," in 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016, pp. 3663-3667. doi: 10.1109/ICEEOT.2016.7755267.

[11] K. Singh and K. Singh, "Real-Time Hand Gesture Recognition System for Sign Language Translation," in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 742-747. doi: 10.1109/ICCS47637.2019.9035136.

[12] M. A. Razak and M. R. B. Mohamed, "Design and Development of a Voice Conversion System for Deaf and Dumb," in 2015 International Conference on Computing and Communication Technologies (ICCCT), Chennai, India, 2015, pp.

[13] J. Wang, Y. Zhang, Z. Lv, and J. Cao, "Real-time hand gesture recognition based on deep learning and OpenCV," Multimedia Tools and Applications, vol. 79, no. 5-6, pp. 3591-3606, Jan. 2020.

[14] K. Zhang and X. Peng, "A novel hand gesture recognition algorithm based on a hybrid CNN-LSTM network," Journal of Visual Communication and Image Representation, vol. 72, pp. 112-123, Nov. 2020.

[15] Y. Zhang, C. Zhang, and H. Zhang, "Real-time sign language recognition using a convolutional neural network," in Proceedings of the 3rd International Conference on Multimedia Systems and Signal Processing, Beijing, China, 2020, pp. 1-5.

[16] S. Deshmukh, K. Dhingra, and P. Vashisht, "Design and implementation of a voice conversion system for deaf and dumb," in Proceedings of the 4th International Conference on Computing, Communication and Networking Technologies, Chennai, India, 2018, pp. 1-6.

[17] N. Alotaibi, N. S. Mekala, and S. M. Ahmed, "A real-time hand gesture recognition system using deep learning for sign language communication," in Proceedings of the 5th International Conference on Computing, Communication and Security, New York, NY, USA, 2020, pp. 1-6.

[18] L. Zou, X. Yang, and Y. Chen, "Real-time sign language recognition using a hybrid model of convolutional neural network and hidden Markov model," Journal of Visual Communication and Image Representation, vol. 60, pp. 324-333, Dec. 2019.

[19] H. Kim, J. Lee, and K. Lee, "A voice conversion system for Korean sign language using a deep neural network," in Proceedings of the 11th International Conference on Computational Intelligence and Communication Networks, Ghaziabad, India, 2019, pp. 1-5.

[20] S. S. Rautaray and A. Agrawal, "Vision-based hand gesture recognition: A review," Artificial Intelligence Review, vol. 51, no. 3, pp. 345-430, Sep. 2019.

[21] Raj, Jennifer S., and Mr C. Vijesh Joe. "Wi-Fi Network Profiling and QoS Assessment for Real Time Video Streaming." IRO Journal on Sustainable Wireless Systems 3, no. 1 (2021): 21-30.

[22] Tripathi, Milan. "Analysis of Convolutional Neural Network based Image Classification Techniques." Journal of Innovative Image Processing (JIIP) 3, no. 02 (2021): 100-117.

[23] Hamdan, Yasir Babiker. "Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition." Journal of Information Technology 3, no. 02 (2021): 92-107.

[24] J. Rajalakshmi, P. Kumar. "Hand Gesture Recognition using CNN and RNN", International Journal of Recent Technology and Engineering (IJRTE), 2020.

[25] Syed Raquib, Shareef, Mohammed, Mannan Hussain, Akash Gupta, Hakeem Aejaz Aslam. "Hand Gesture Recognition System for Deaf and Dumb", International Journal of Multidisciplinary and Current Educational Research (IJMCER) 2020.