

SIGN-LANGUAGE RECOGNITION SYSTEM

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Computer Science and Engineering



MENTOR:

Mr. Pratipal Singh

Dept. of Computer Science & Engineering

SUBMITTED BY:

Aman Jain (19ESKCS032)

Anshika Mittal (19ESKCS038)

Arunabh Jain (19ESKCS046)

Aryan Sharma (19ESKCS047)

Department Of Computer Science & Engineering

Swami Keshvanand Institute of Technology, M & G, Jaipur

Rajasthan Technical Kota, Jaipur

Session 2022-23



**Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur
Department of Computer Science and Engineering**

CERTIFICATE

This is to certify that **Mr. Aman Jain (19ESKCS032)**, a student of B.Tech (Computer Science & Engineering) VIII Semester has submitted his Project Report entitled “**Sign Language Recognition System**” under my guidance

Mentor:

Mr. Pratipal Singh
Dept. of Computer
Science & Engineering

Coordinator:

30/5/23

Ms. Anjana Sangwan
Dept. of Computer
Science & Engineering



**Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur
Department of Computer Science and Engineering**

CERTIFICATE

This is to certify that **Ms. Anshika Mittal (19ESKCS038)**, a student of B.Tech (Computer Science & Engineering) VIII Semester has submitted his Project Report entitled “**Sign Language Recognition System**” under my guidance

Mentor:

Mr. Pratipal Singh
Dept. of Computer
Science & Engineering

Coordinator:

Ms. Anjana Sangwan
Dept. of Computer
Science & Engineering



**Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur
Department of Computer Science and Engineering**

CERTIFICATE

This is to certify that **Mr. Arunabh Jain (19ESKCS046)**, a student of B.Tech (Computer Science & Engineering) VIII Semester has submitted his Project Report entitled “**Sign Language Recognition System**” under my guidance

Mentor:

Mr. Pratipal Singh
Dept. of Computer
Science & Engineering

Coordinator:

Ms. Anjana Sangwan
Dept. of Computer
Science & Engineering



**Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur**
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **Mr. Aryan Sharma (19ESKCS047)**, a student of B.Tech (Computer Science & Engineering) VIII Semester has submitted his Project Report entitled “**Sign Language Recognition System**” under my guidance

Mentor:

Mr. Pratipal Singh
Dept. of Computer
Science & Engineering

Coordinator:

Ms. Anjana Sangwan
Dept. of Computer
Science & Engineering



**Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur
Department of Computer Science and Engineering**

DECLARATION

We hereby declare that the report of the project entitled “SIGN-LANGUAGE RECOGNITION SYSTEM” is a record of an original work done by us at Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur under the mentorship of Mr. Pratipal Singh (Dept. of Computer Science & Engineering) and coordinator Ms. Anjana Sangwan (Dept. of Computer Science & Engineering). This project report has been submitted as the proof of original work for the particular fulfillment of the requirements for the award of the degree Bachelor of Technology (B.Tech) in the Department of Computer Science. It has not been submitted anywhere else, under any other program to the best of our knowledge.

Team Members:

Aman Jain (19ESKCS032)
Anshika Mittal (19ESKCS038)
Arunabh Jain (19ESKCS046)
Aryan Sharma (19ESKCS047)

Signature:

Aman
Anshika
Arunabh Jain
Aryan

Acknowledgement

A project of such vast coverage cannot be realized without help from numerous sources and people in the organization. We take this opportunity to express our gratitude to all those who have been helping us in making this project successful.

We are highly indebted to our faculty mentor Mr. Pratipal Singh. He has been a guide, motivator and source of inspiration for us to carry out the necessary proceedings for the project to be completely successful. We would also like to thank Ms. Anjana Sangwan for her co-operation, encouragement, valuable suggestions and critical remarks that galvanized our efforts in the right direction.

We would also like to convey our sincere thanks to Prof. (Dr.) Mukesh Kumar Gupta, HOD, Department of Computer Science and Engineering, for facilitating, motivating and supporting us during each phase of development of the project. Also, we pay our sincere gratitude to all the Faculty Members of Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur and all our Colleagues for their co-operation and support.

Last but not least we would like to thank all those who have directly or indirectly helped and cooperated in accomplishing this project

Team Members:

Aman Jain (19ESKCS032)
Anshika Mittal (19ESKCS038)
Arunabh Jain (19ESKCS046)
Aryan Sharma (19ESKCS047)

Signature:

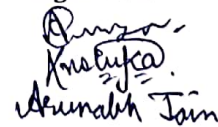

Aryan

Table of Contents

1. INTRODUCTION	1
1.1 Problem Statement & Objective	1
1.2 Investigation & Analysis.....	2
1.3 Introduction to Project	4
1.4 Proposed Solution	4
1.5 Scope of the Project	5
2. SYSTEM REQUIREMENTS SPECIFICATIONS	6
2.1 Overall Description.....	6
2.1.1 Product Perspective.....	6
2.1.1.1 System Interfaces.....	7
2.1.1.2 User Interfaces	7
2.1.1.3 Hardware Interfaces.....	7
2.1.1.4 Software Interfaces	8
2.1.1.5 Communication Interfaces.....	9
2.1.1.6 Memory Constraints	9
2.1.1.7 Operations.....	9
2.1.1.8 Project Function.....	10
2.1.1.9 User Characteristics	10
2.1.1.10 Constraints	10
2.1.1.11 Assumption & Dependencies	10
3. SYSTEM DESIGN SPECIFICATION	11
3.1 System Architecture.....	11
3.2 Module Decomposition Description	12
3.3 High Level Design Diagrams.....	17
3.3.1 Use-Case Diagram	17
3.3.2 Activity Diagram	18
3.3.3 Data Flow Diagram.....	19

3.3.4 Sequence Diagram	20
4. METHODOLOGY & TEAM	21
4.1 Introduction to Waterfall Framework	21
4.2 Team Members, Roles & Responsibilities.....	23
5. CENTERING TESTING SYSTEM	24
5.1 Functionality Testing	24
5.2 Performance Testing	25
5.3 Usability Testing	26
6. TEST EXECUTION SUMMARY	27
7. PROJECT SCREENSHOTS	29
8. PROJECT SUMMARY AND CONSLUSION	34
8.1 Conclusions.....	34
9. FUTURE SCOPE	35
REFERENCES	36
PROJECT LINKS	36

List of Figures

1. Figure 2.1 System Description.....	6
2. Figure 3.1 System Architecture.....	11
3. Figure 3.2.1 Example Training Dataset	12
4. Figure 3.2.2 Workflow for Pre-Processing	13
5. Figure 3.2.3 Model Summary	14
6. Figure 3.2.4 Loss and Accuracy Curve.....	15
7. Figure 3.2.5 Deployment Window.....	16
8. Figure 3.3.1 Use-Case Diagram	17
9. Fig 3.3.2: Activity Diagram	18
10. Fig 3.3.3: Data-Flow Diagram	19
11. Fig 3.3.4: Sequence Diagram.....	20
12. Fig 4.1: Waterfall Model with Feedback	21
13. Fig 7.1: Home Page.....	29
14. Fig 7.2: Output Window – ‘H’ for getting text “HELLO”	30
15. Fig 7.3: Output Window – ‘E’ for getting text “HELLO”	30
16. Fig 7.4: Output Window – ‘L’ for getting text “HELLO”	31
17. Fig 7.5: Output Window – ‘L’ for getting text “HELLO”	31
18. Fig 7.6: Output Window – ‘O’ for getting text “HELLO”	32
19. Fig 7.7: Output Window – “LOL”	32
20. Fig 7.8: Output Window – “WOW”	33
21. Fig 7.9: Output Window – “HOW”	33

List of Tables

1. Table 2.1 Minimum Client Side Hardware Interface.....	7
2. Table 2.2 Minimum Server Side Hardware Interface	7
3. Table 2.3 Recommended Client Side Hardware Interface.....	8
4. Table 2.4 Recommended Server Side Hardware Interface	8
5. Table 2.5 Minimum Software Interfaces	8
6. Table 2.6 Recommended Software Interfaces	9
7. Table 4.1 Roles and Responsibilities	23
8. Table 6.1 Test Case Summary	28

Chapter 1

Introduction

1.1 Problem Statement & Objective

Problem: The problem addressed by the sign language recognition tool is the communication barrier faced by individuals who are deaf or hard of hearing. Sign language is the primary means of communication for the deaf community, but it can be challenging for non-sign language users to understand and communicate with them effectively. The lack of widespread sign language knowledge among the general population further exacerbates this issue. Therefore, there is a need for a technology-based solution that can recognize and interpret sign language gestures to facilitate effective communication.

Reasons: The main reason for developing a sign language recognition tool is to bridge the communication gap between individuals who use sign language and those who do not. This tool aims to provide a means for non-sign language users to understand and interact with individuals who are deaf or hard of hearing. By leveraging computer vision and machine learning techniques, it becomes possible to analyze and interpret sign language gestures, ultimately enabling more inclusive and accessible communication for the deaf community.

Solution: The sign language recognition tool utilizes Python, OpenCV (Open Source Computer Vision Library), and Flask (a micro web framework) to provide a solution for recognizing and interpreting sign language gestures. The tool employs computer vision techniques to capture and process video input from a camera or any video source. It analyzes the video frames to detect and track the hand or specific markers corresponding to sign language gestures. OpenCV offers a rich set of image processing and computer vision algorithms that aid in detecting and recognizing hand shapes, gestures, and movements.

To improve the accuracy and robustness of the system, machine learning models can be trained using datasets of sign language gestures. Convolutional Neural Networks (CNNs) or other deep learning architectures can be employed to classify and recognize different signs. These models learn from annotated data and can identify the specific gestures made by the user.

The Flask framework is used to create a web-based interface for the sign language recognition tool. It allows users to access the system through a web browser, facilitating easy and intuitive interaction. The tool can display the recognized sign language gestures in real-time, aiding non-sign language users in understanding and responding appropriately.

Objective: The main objectives of the sign language recognition tool are:

- a) To provide a means of effective communication between sign language users and non-sign language users.
- b) To recognize and interpret sign language gestures accurately and in real-time.
- c) To improve accessibility and inclusivity for individuals who are deaf or hard of hearing.
- d) To reduce the communication barrier and enable better interaction and understanding between different individuals.
- e) To create a user-friendly interface through the Flask web framework for easy access and interaction with the tool.
- f) To leverage computer vision and machine learning techniques to develop a robust and accurate sign language recognition system.

1.2 Investigation & Analysis

The investigation and analysis for the sign language recognition tool can encompass several aspects, including the technical feasibility, usability, impact, and potential challenges. Here are some key points to consider:

Technical Feasibility:

The investigation should include an analysis of the technical requirements and resources needed to develop the sign language recognition tool using Python, OpenCV, and Flask. This may

involve assessing the availability of suitable hardware (e.g., cameras) and the computational capabilities required for real-time processing.

It is essential to explore the compatibility of the chosen technologies and libraries to ensure seamless integration and efficient performance.

Consideration should be given to the availability of datasets for training machine learning models to recognize sign language gestures. Analyze the quality, size, and diversity of existing datasets or explore the feasibility of creating custom datasets.

Usability:

Evaluate the user experience of the sign language recognition tool. Conduct user studies or gather feedback from individuals who are deaf or hard of hearing, as well as non-sign language users, to understand the ease of use, intuitiveness, and effectiveness of the tool.

Assess the accuracy and reliability of the gesture recognition system. Measure the recognition rates and identify any limitations or challenges in accurately detecting and interpreting different sign language gestures.

Consider the accessibility of the web-based interface developed using Flask. Ensure that it is user-friendly and compatible with different devices and browsers.

Impact:

Analyze the potential impact of the sign language recognition tool on individuals who are deaf or hard of hearing. Assess how it can enhance their ability to communicate effectively with non-sign language users, increasing inclusivity and reducing the communication barrier.

Consider the broader societal impact of the tool, such as enabling individuals who are deaf or hard of hearing to participate more actively in various domains, including education, employment, and social interactions.

Challenges:

Identify and analyze potential challenges and limitations of the sign language recognition tool. These may include varying lighting conditions, occlusions, hand shape variations, and the need for continuous improvement and updates to the machine learning models.

Assess the computational requirements for real-time processing and explore potential optimizations to enhance performance.

Investigate potential ethical considerations, such as privacy and data security, particularly when

handling video input.

By conducting a thorough investigation and analysis, you can gain a comprehensive understanding of the technical feasibility, usability, impact, and challenges associated with the sign language recognition tool. This information will support the development process and provide valuable insights for further enhancements and future iterations of the tool.

1.3 Introduction to Project

There have been several advancements in technology and a lot of research has been done to help the people who are deaf and dumb. Deaf and hard-of-hearing persons, as well as others who are unable to communicate verbally, utilize sign language to communicate within their communities and with others. Aiding the cause, Deep learning, and computer vision can be used to make an impact on this cause. This can be very helpful for the deaf and dumb people in communicate with others as knowing sign language is not something that is common to all, this can be extended to creating automatic editors, where the person can easily write by just hand gestures.

1.4 Proposed Solutions

Proposed System

So, we have come up with a solution to above mentioned problems, with a product fall under the category of human-computer interaction (HCI). In this sign language recognition project, we create a sign detector, which detects vast multitude of other signs and hand gestures including the alphabets and can generate text on the basis of hand gestures. We have developed this project using OpenCV and Keras modules of python and also used CNN.

This is divided into following parts:

1. Taking the dataset from Kaggle.
2. Training a CNN on the captured dataset.
3. Deploying the model on web app using Flask.
4. Predicting the data with use of image.

1.5 Scope of the Project

Sign Language is a language which allows mute people to communicate. The ease of communication offered by this language, however, disappears when one of the interlocutors, who may or not be mute, does not know Sign Language and a conversation starts using that language. Thus, this project helps deaf and mute people to communicate with any person. This project helps them to convert hand gestures into alphabets which help them to communicate with the person. It is deployed on web app which takes hand gestures of user from web cam and predicts the letter and stores the history.

Chapter 2

System Requirements Specifications

2.1 Overall Description

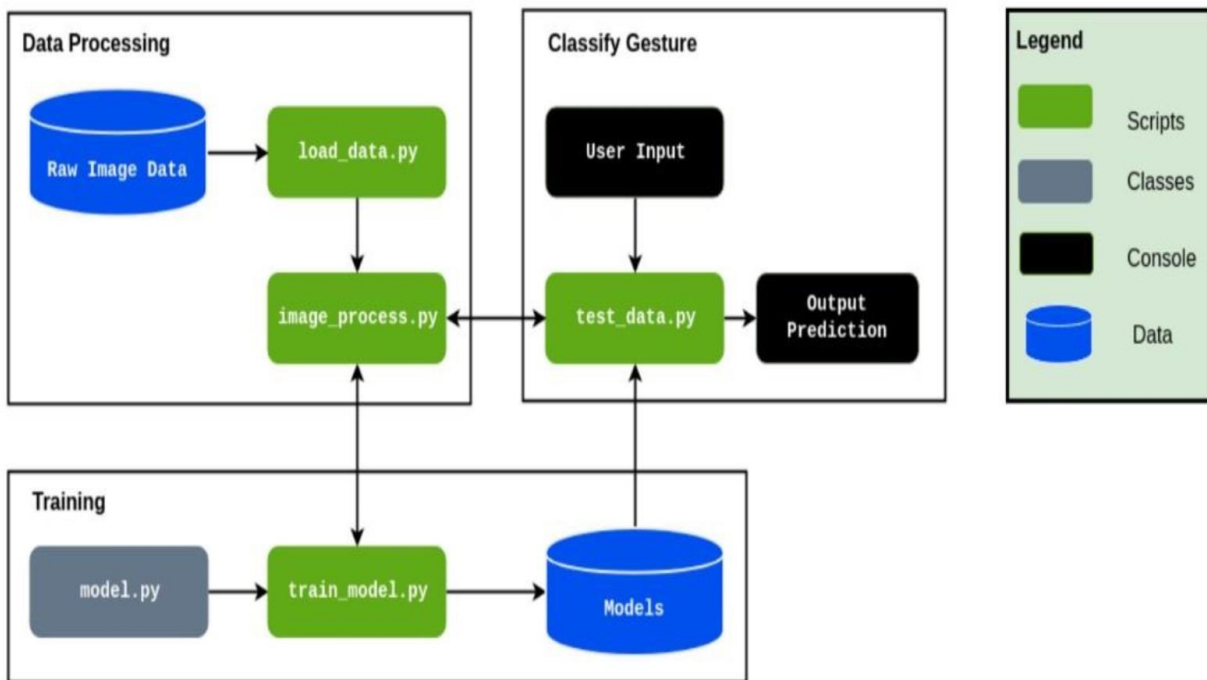


Fig 2.1: System Description

2.1.1 Product Perspective

Sign language is the only tool of communication for the person who is not able speak and hear anything. Sign language is a boon for the physically challenged people to express their thoughts and emotion. In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabets 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. Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different

signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

2.1.1.1 System Interfaces

List each system interface and identify the functionality of the system (hardware and software both) to accomplish the system requirement and interface description to match the system.

2.1.1.2 User Interfaces

The application will have a user friendly and menu-based interface. Following screens will be provided:

- It contains the problem statement, so that user can get friendly.

2.1.1.3 Hardware Interfaces

- A network connection (internet / intranet) is required to make the web service accessible on other systems connected over the network.
- Other hardware interface specifications are as follows

Minimum Requirements:

Client Side			
	Processor	RAM	Disk Space
Google Chrome	Intel Pentium III or AMD -800 MHz	1 GB	100 MB

Table 2.1 Minimum Client Side Hardware Interface

Server Side			
	Processor	RAM	Disk Space
Flask	Intel Pentium III or AMD -800 MHz	1 GB	3.5 GB

Table 2.2 Minimum Server Side Hardware Interface

Recommended Requirements:

Client Side			
	Processor	RAM	Disk Space
Google Chrome	Intel i3 Processor or AMD Ryzen 3	4 GB	1 GB

Table 2.3 Recommended Client Side Hardware Interface

Server Side			
	Processor	RAM	Disk Space
Flask	Intel i3 Processor or AMD Ryzen 3	4 GB	3.5 GB

Table 2.4 Recommended Server Side Hardware Interface**2.1.1.4 Software Interfaces**

- Any Microsoft Windows 7 and higher (Windows 7 / 8 / 8.1 / 10) or equivalent Linux based operating system with minimum kernel support 3.X.
- Crystal Reports 8 for generation and viewing of reports
- We have chosen web application for its best support and user-friendliness.
- To implement the project we have chosen VS Code for its more interactive support.

Minimum Requirements:

Software Tool	Version	Purpose of Use
Operating System	Windows 7 and higher or Linux with kernel 3.x and higher	Installation and operational platform
Web Browser	Google Chrome, Brave and other higher compatible	Access to the web application
Web Server	Node Server	Running the web application over intranet

Table 2.5 Minimum Software Interface

Recommended Requirements:

Software Tool	Version	Purpose of Use
Operating System	Windows 8 and higher or Linux with kernel 3.x and higher	Installation and operational platform
Web Browser	Google Chrome, Brave and other higher compatible	Access to the web application
Web Server	Node Server	Running the web application over intranet

Table 2.6 Recommended Software Interface**2.1.1.5 Communication Interfaces**

- Client (customer) on Internet will be using HTTP/HTTPS protocol.
- Client (system user) on Internet will be using HTTP/HTTPS protocol.

2.1.1.6 Memory Constraints

- At least 256 MB of RAM and 2 GB of space on hard disk will be required for running the application on client end.
- Similarly, a minimum of 2048 MB of RAM and 20 GB of space on hard disk will be required for running the application on server end.

2.1.1.7 Operations

- Web application stores history (previously predicted signs) when in use.
- Application detects hand gestures of user for predicting the alphabet according to American sign language.
- Model is based on CNN (convolutional neural network).

2.1.1.8 Project Functions

- This sign language conversion system includes user interaction.
- It captures hand gestures of user to predict the sign from web cam.
- Model is deployed using flask framework and model is trained on CNN.
- Project includes interactive front end with problem statement.

2.1.1.9 User Characteristics

- Technical Expertise: User should be comfortable using general purpose applications on a computer.
- User have to use hand gestures for predicting the result.
- Educational Level: User should be at least graduate and comfortable with English.

2.1.1.10 Constraints

- GUI is only in English.
- Limited to HTTP/HTTPS.
- Needed web camera interaction of user for getting hand gestures.
- White background is needed for accurate prediction.

2.1.1.11 Assumption & Dependencies

- The project code will not change.
- White background is needed for accurate prediction.
- User has to set his hand inside ROI (Region of interest) for accurate predictions.

Chapter 3

System Design Specifications

3.1 System Architecture

System architecture presents the schematic view of the complete system along with its major components and their connectivity. The overall architecture of the proposed system will be as follows.

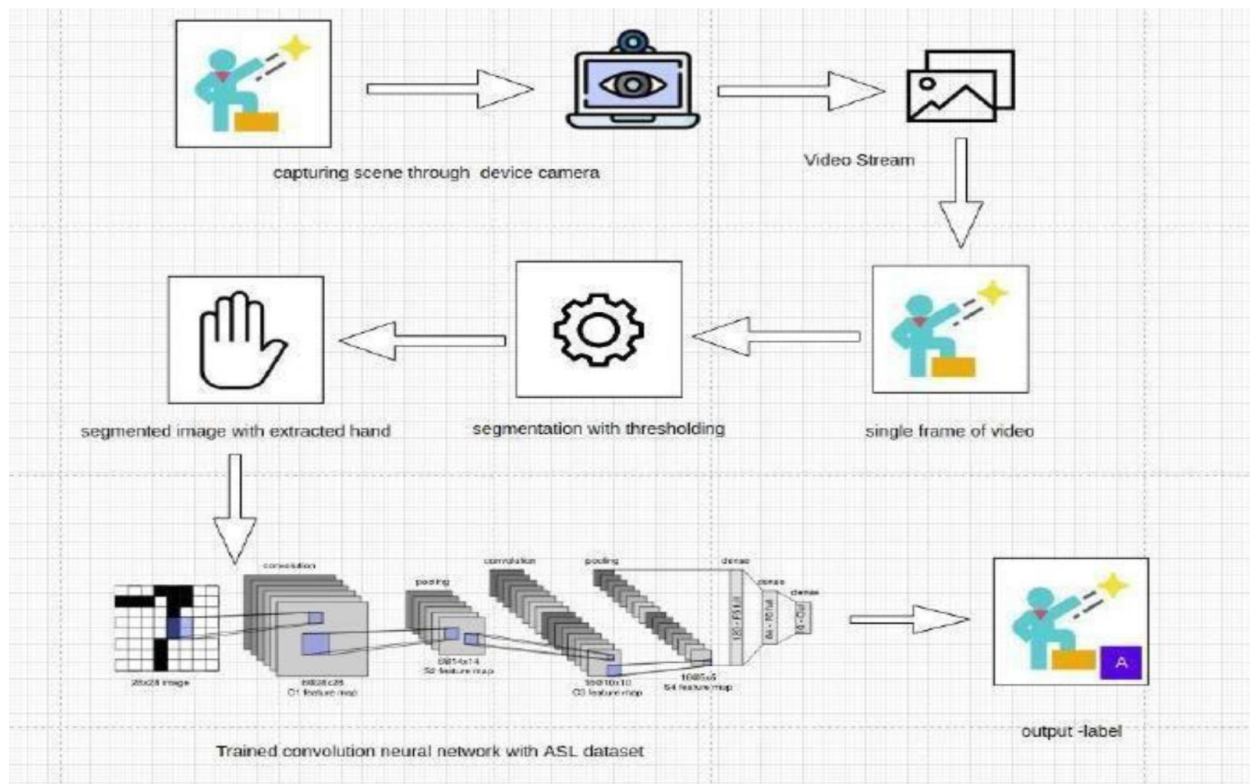


Fig 3.1: System Architecture

3.2 Module Decomposition

The proposed system can be majorly decomposed into following modules:

- Data gathering
- Data cleaning, manipulation and pre-processing
- Model building
- Model Training and Testing
- Model Deployment using Flask

3.2.1 Data Gathering

For any deep-learning model to work well a good dataset is important. Here we have alphabetical characters in English language represented in American Sign Language. The data set is open-sourced and available free to use on <https://www.kaggle.com/grassknotted/asl-alphabet>. The dataset consists of 3000 unique images of 26 alphabetical characters namely A-Z. The dataset also consists of some special signs which are helpful for us to differentiate between words when the whole model is ready to deploy. These special signs can be implemented in an app for increasing the user friendliness. The 3 special signs included are 'del', 'space' and 'nothing' each with 3000 images for training. For the validation part we will be randomly choosing 20% of the whole training dataset and test on 28 unique images already available with the dataset. (The 'del' image is missing here). A sample from the dataset is shown here:



Fig 3.2.1: Example Training Dataset

3.2.2 Data Cleaning, Manipulation and Pre-Processing

The batch size: Number of training examples that are went through the model one iteration are 32. With this now we have the training batch of size 2175 and validation size of 544. For 69600 images belonging to 29 classes for the training set and 17400 images for the validation set. Initially before passing the images to the model all images will be turned to tensors and reshaped to a size of (200 x 200) with 3 colour channels. The images are being converted into tensors to use the power of GPU's (Graphical processing unit). These lines from NVIDIA will make your idea clearer. According to NVIDIA, tensors operate on their GPUs to deliver an order of magnitude better performance with lower precisions such as TF32 and FP16. NVIDIA provides direct support for CUDA-X libraries. This approach is fully automated, reducing training to convergence times while retaining accuracy.

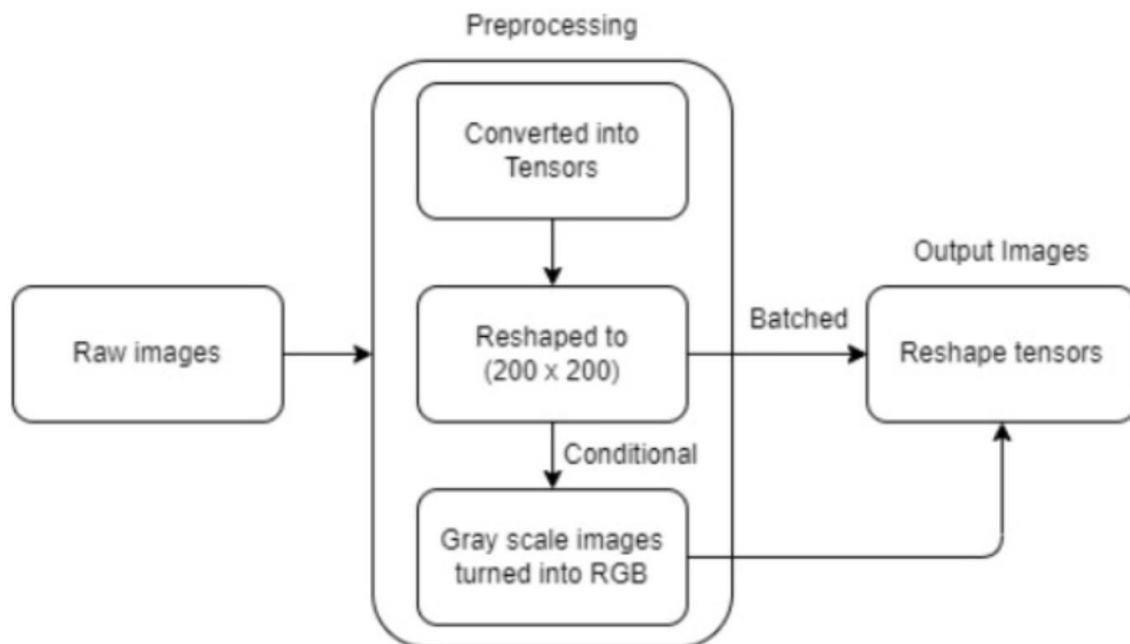


Fig 3.2.2: Workflow for Pre-Processing

3.2.3 Model Building

We will be using a transfer learnt model that will have its own trained weights and then we will use these already learnt weights to train our model on our character signs dataset. The modelling steps will include creating an input layer which takes images of shape (200 x 200 x 3). This layer will be called 'input layer'. Next, we will pass this input layer to our base model with training set to false so that the pretrained weights don't get updated. Next layer will be a global average pooling layer following a dense layer with 29 nodes. 29 nodes to get the output probability of each class using the 'softmax' activation. Lastly, we will condense all these layers into a single model.

The model summary layer wise is –

```
Model: "model"
_____
Layer (type)                Output Shape                Param #
=====
input_layer (InputLayer)    [(None, 200, 200, 3)]      0

efficientnetb0 (Functional) (None, None, None, 1280)    4049571

global_average_pooling (Glo (None, 1280)                0
balAveragePooling2D)

output_layer (Dense)        (None, 29)                  37149

=====
Total params: 4,086,720
Trainable params: 37,149
Non-trainable params: 4,049,571
```

Fig 3.2.3: Model Summary

The 'efficientnetb0' layer here is the one that contains the pre-trained weights with a dense architecture. The model workflow looks something like this. A layer in a deep learning model is a structure that holds information and processes the information in the current state then this

same information is passed to other layers that are consecutively present in the model to process further information. Many types of layers are present in an architecture some of them are Convolutional neural networks, max pooling and global average pooling. Fully connected layer and ReLU layers are also present. RNN layer in the RNN model and deconvolutional layer in autoencoder.

3.2.4 Model Training and Testing

For a single training sample in the dataset, the error is determined using the loss function. The cost function calculates the average of all the loss functions in the training set. The optimizer is an algorithm that optimizes the weights and biases such that we move towards the minimum in the graph. In multi-class classification tasks, categorical cross-entropy is a loss function. These are challenges in which a sample can only belong to one of many possible categories, and the model must choose which one it belongs to.

It's usually used to calculate the difference between two probability distributions. where \hat{y}_i is the model output's i th scalar value, y_i is the matching target value, and output size is the number of scalar values in the model output This loss is a good indicator of how easily two discrete probability distributions can be distinguished from one another. In this case, y_i represents the chance that event i will occur, and the sum of all y_i equals 1, implying that only one event will occur. When the distributions become closer to one other, the minus sign ensures that the loss gets lower.

The training loss closes at 0.0037 whereas validation loss closes at 0.3185. The training accuracy reaches 0.9999 whereas validation accuracy reaches 0.9054. The model has slightly over-fitted.

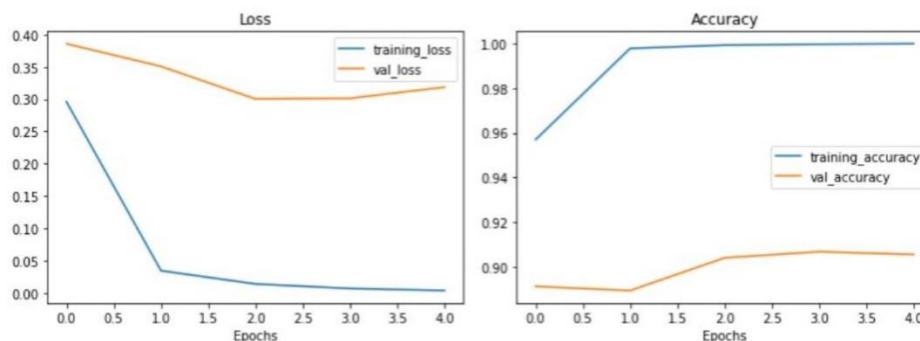


Fig 3.2.4: Loss and Accuracy Curve

3.2.5 Model Deployment using Flask

Machine learning deployment is the process of deploying a machine learning model in a real environment. The model is frequently integrated with apps via an API and can be utilised in a variety of contexts. Deployment is a crucial stage in delivering operational value for an organisation from machine learning.

The model will be deployed to the web with flask API. A local server will be initiated that will take the live inputs from the webcam or any other live video sourcing device and make predictions on the same. The predictions will be returned and displayed on the web-page creating an interactive loop between the user and the model. The input is being taken in form of signs or gestures, the user will require a small-time gap of 5 seconds to transition between the first sign and second sign. To create user experience at its best an itch effect of black screen in case of successful image capture will be shown.

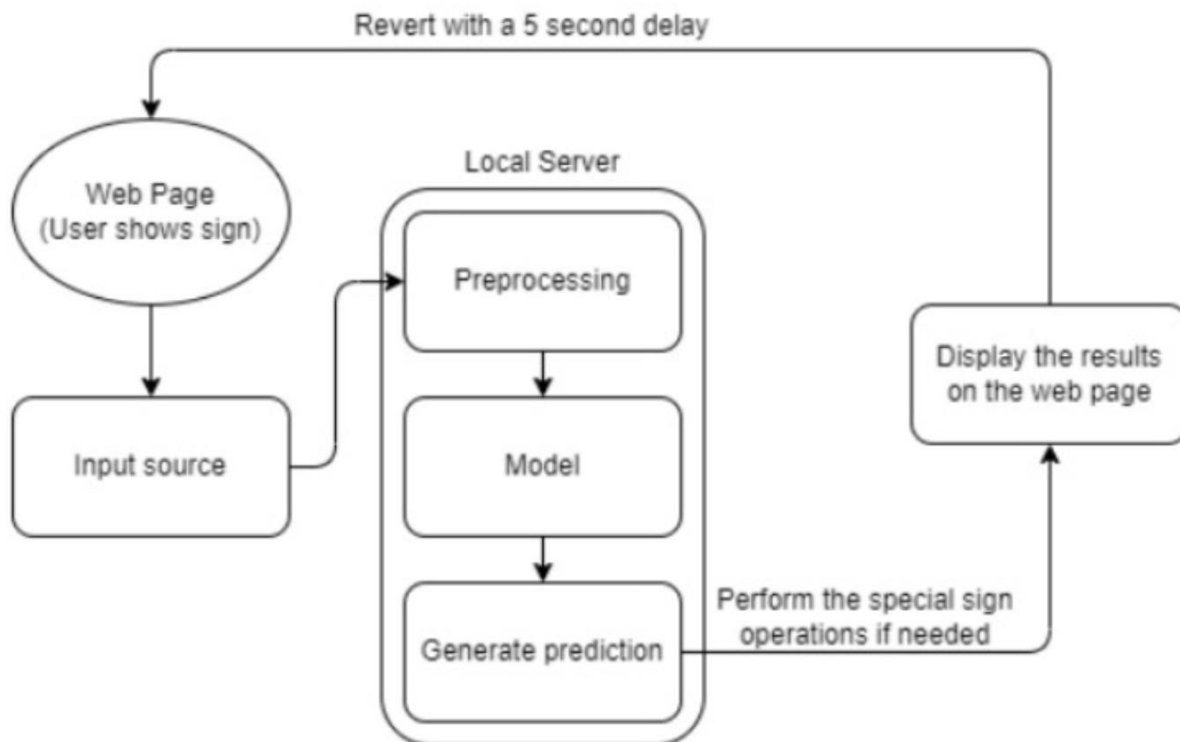


Fig 3.2.5: Deployment Window

3.3 High Level Design Diagrams

3.3.1 Use – Case Diagrams

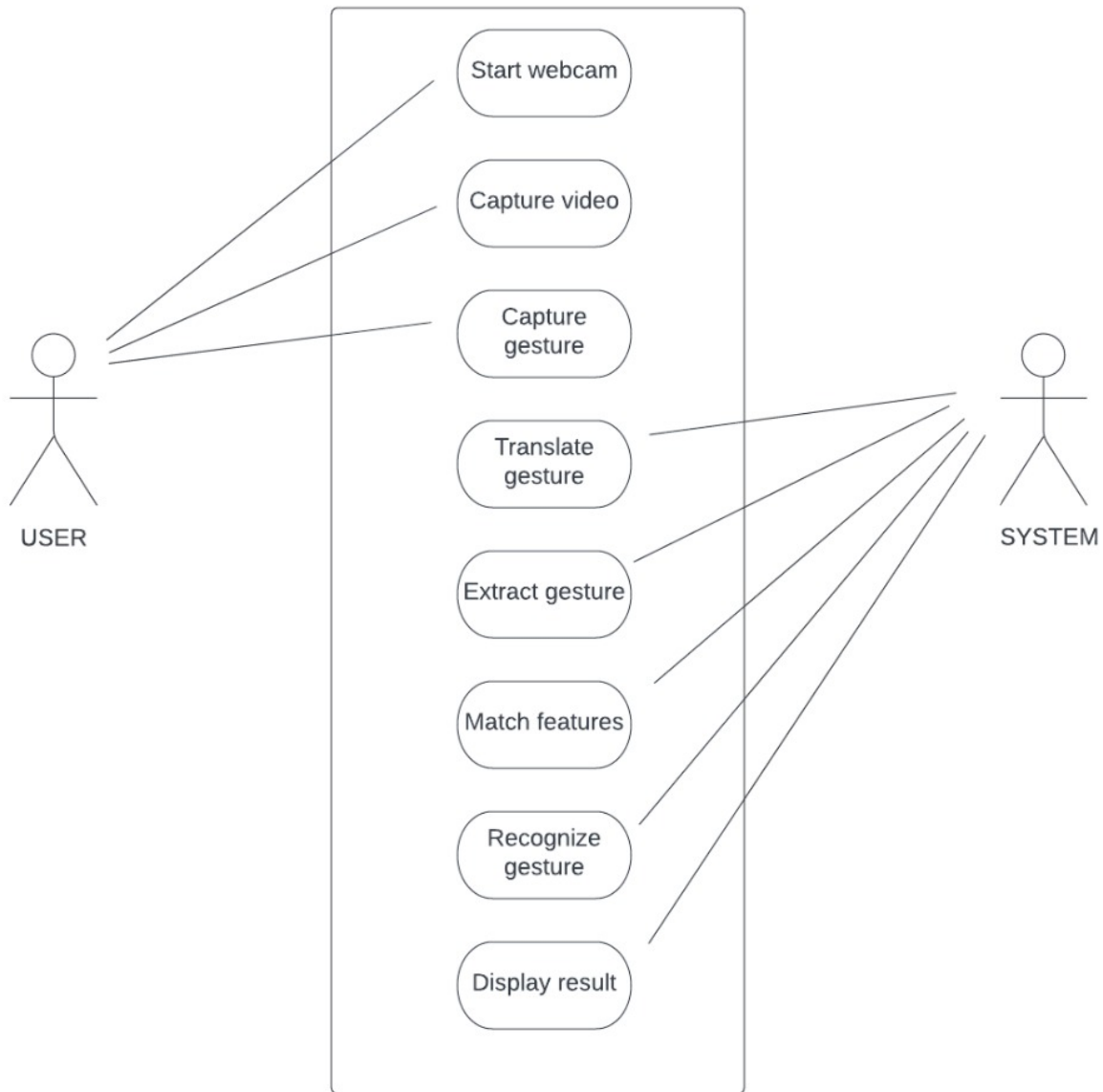


Fig 3.3.1: Use - Case Diagram

3.3.2 Activity Diagram

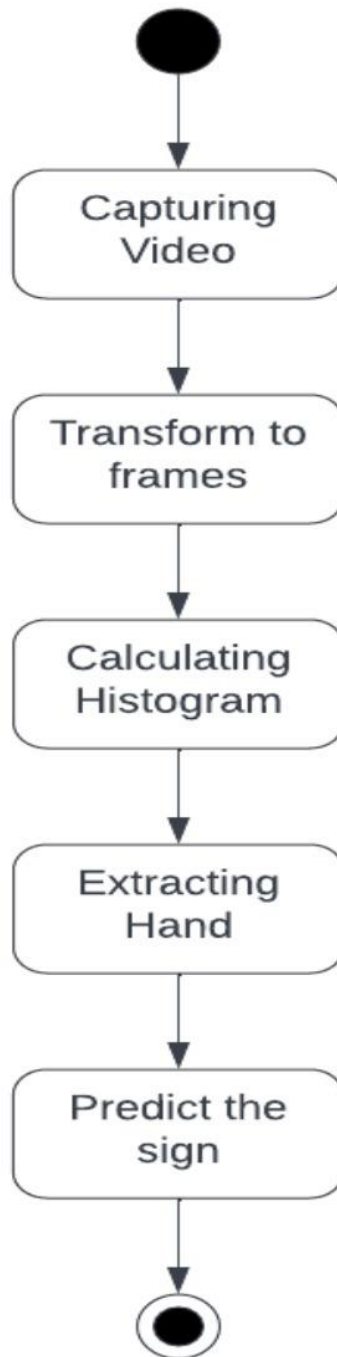


Fig 3.3.2: Activity Diagram

3.3.3 Data Flow Diagram

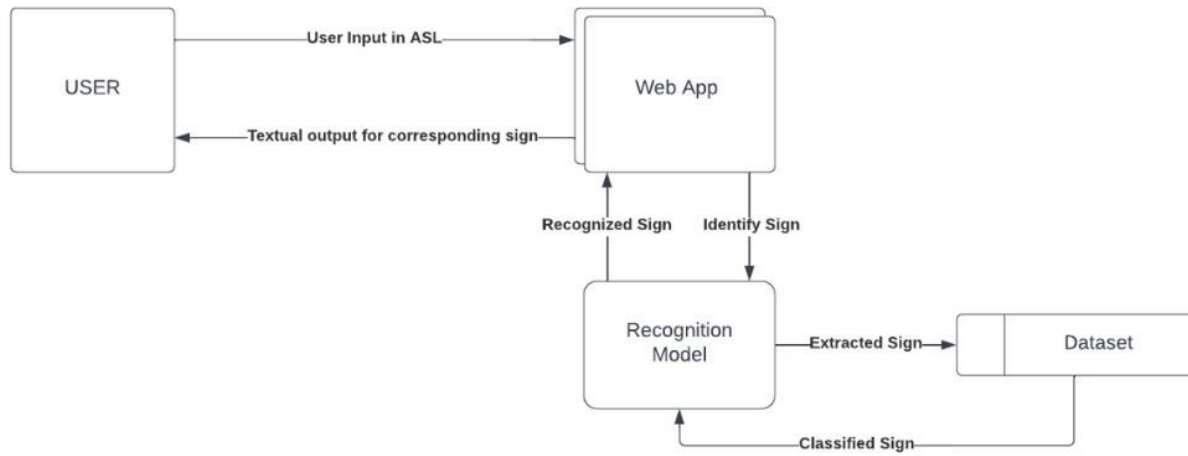


Fig 3.3: Data Flow Diagram

3.3.4 Sequence Diagram

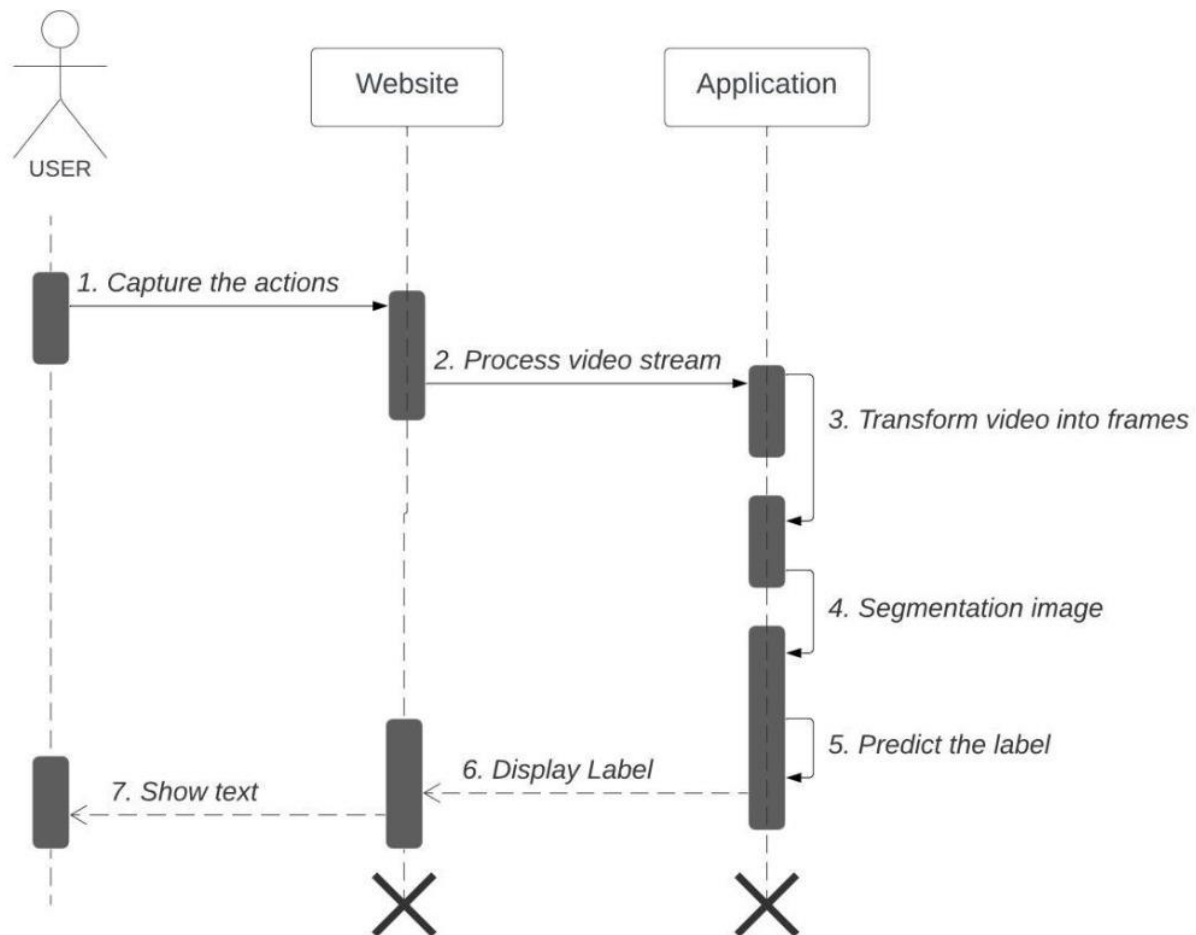


Fig 3.3.4: Sequence Diagram

Chapter 4

Methodology and Team

4.1 Introduction to Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.

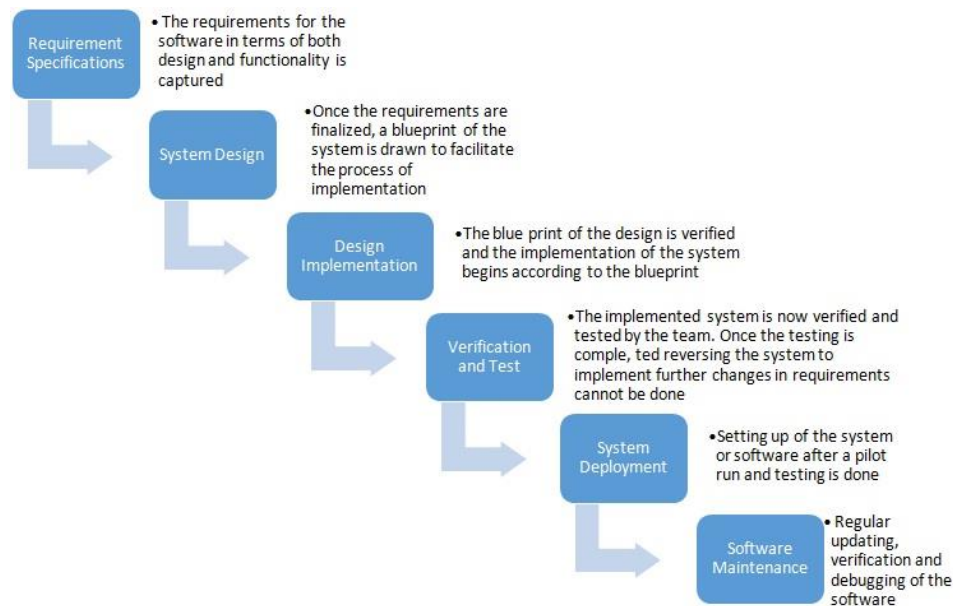


Fig 4.1: Waterfall Model with Feedback

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors.

Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.

Waterfall Model Pros & Cons

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

4.2 Team Members, Roles & Responsibilities

Team Members	Project Role	Responsibilities
Aman Jain	Team Member	Data gathering and manipulation, model building
Anshika Mittal	Team Member	Model training, model testing, model evaluation
Arunabh Jain	Project Manager	Front-end and designing of the website using HTML, CSS, JavaScript
Aryan Sharma	Team Member	Back-end, creating APIs through flask and project deployment

Table 4.1: Roles and Responsibilities

Chapter 5

Centering Testing System

The designed system has been testing through following test parameters.

5.1 Functionality Testing

In testing the functionality of the web sites the following features were tested:

- **Links**

- a) Internal Links:

All internal links of the website were checked by clicking each link individually and providing the appropriate input to reach the other links within.

- b) External Links:

Till now no external links are provided in our website but for future enhancement we will provide the links to the candidate's actual profile available online and link up with the elections updates online etc.

- c) Mail Links:

No mail links are provided in our website till this stage but this is also a future enhancement of our website to trigger mails to people for keeping them updated about the online registration dates, the polling dates and other details.

- d) Broken Links:

Broken link are those links which so not divert the page to specified page or any page at all. By testing the links on our website there was no link found on clicking which we did not find any page.

- **Forms**

- a) Field validation

Checks on dates have been applied. For e.g. The Date of birth should be less than the current date and after that we have checked the age to be greater than or equal to 18 years (the eligible age for casting vote). Checks have been applied on starting and ending dates, i.e. the starting dates of elections or registrations should always fall before their respective ending dates.

- b) Error message for wrong input

Error messages have been displayed as and when we enter the wrong details (e.g. Dates), and when we do not enter any detail in the mandatory fields. For example: when we enter wrong password we get error message for acknowledging us that we have entered it wrong and when we do not enter the username and/or password we get the messages displaying the respective errors.

- c) Optional and Mandatory fields

All the mandatory fields have been marked with a red asterisk (*) and apart from that there is a display of error messages when we do not enter the mandatory fields. For example: As the first name is a compulsory field in all our forms so when we do not enter that in our form and submit the form we get an error message asking for us to enter details in that particular field.

5.2 Performance Testing

Performance testing can be applied to understand the website's scalability, or to benchmark the performance in the environment of third party products such as servers and middleware for potential purchase. This can only be done once it is put into use on the actual internet server and tested by the users.

Till now it is done using the null modem on two systems.

The system load includes:

- a) What is the number of users per time?
- b) Checking for peak loads and how system behaves.
- c) Amount of data accessed by user.

This is done using only 2 systems for now so cannot be tested for load unless we deploy it on a real server machine.

5.3 Usability Testing

Usability testing is the process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction.

- a) Ease of learning
- b) Navigation
- c) Subjective user satisfaction
- d) General appearance

As system is not put into the real time use so it's not yet tested for usability.

Chapter 6

Test Execution Summary

The Execution Test Summary Report is a crucial document that provides an overall view of the testing process from start to end. This report is typically generated at the end of the testing process and is given to the client for their understanding and review. The Test Summary Report contains important information about the testing process, including the test case IDs that were generated, the total number of resources consumed during testing, the number of test cases that passed and failed, and the overall status of the test cases.

The Test Summary Report contents are:

1. Test Case ID generated = PRO1, PRO2, PRO3, PRO4, PRO5
2. Total number of resources consumed = 3
3. Passed Test Cases = 5
4. Failed Test cases = 0
5. Status of Test Cases = Passed

This information is valuable for the client as it provides them with a clear understanding of the testing process and its outcomes. The client can use this information to assess the quality of the software being tested and to make informed decisions about its release.

In addition to the information provided in this summary report, there may be other details included in a Test Summary Report. These details may include information about the testing environment, the testing tools used, any issues or defects that were identified during testing, and any recommendations for future improvements.

The Test Summary Report is an important document that provides transparency and accountability in the testing process. It allows all stakeholders to have a clear understanding of the testing outcomes and to make informed decisions about the software being tested.

S. No.	Test Case ID	Test Case Description	Expected Outcome	Test Case Status	No. of Resources Consumed
1.	PRO1	Creating hand gesture of W using ASL	W	PASS	Monitor, Keyboard, Webcam
2.	PRO2	Creating hand gesture of L using ASL	L	PASS	Monitor, Keyboard, Webcam
3.	PRO3	Creating hand gesture of O using ASL	O	PASS	Monitor, Keyboard, Webcam
4.	PRO4	Creating hand gesture of V using ASL	V	PASS	Monitor, Keyboard, Webcam
5.	PRO5	Creating hand gesture of H using ASL	H	PASS	Monitor, Keyboard, Webcam

Table 6.1: Test Case Summary

Chapter 7

Project Screen Shots

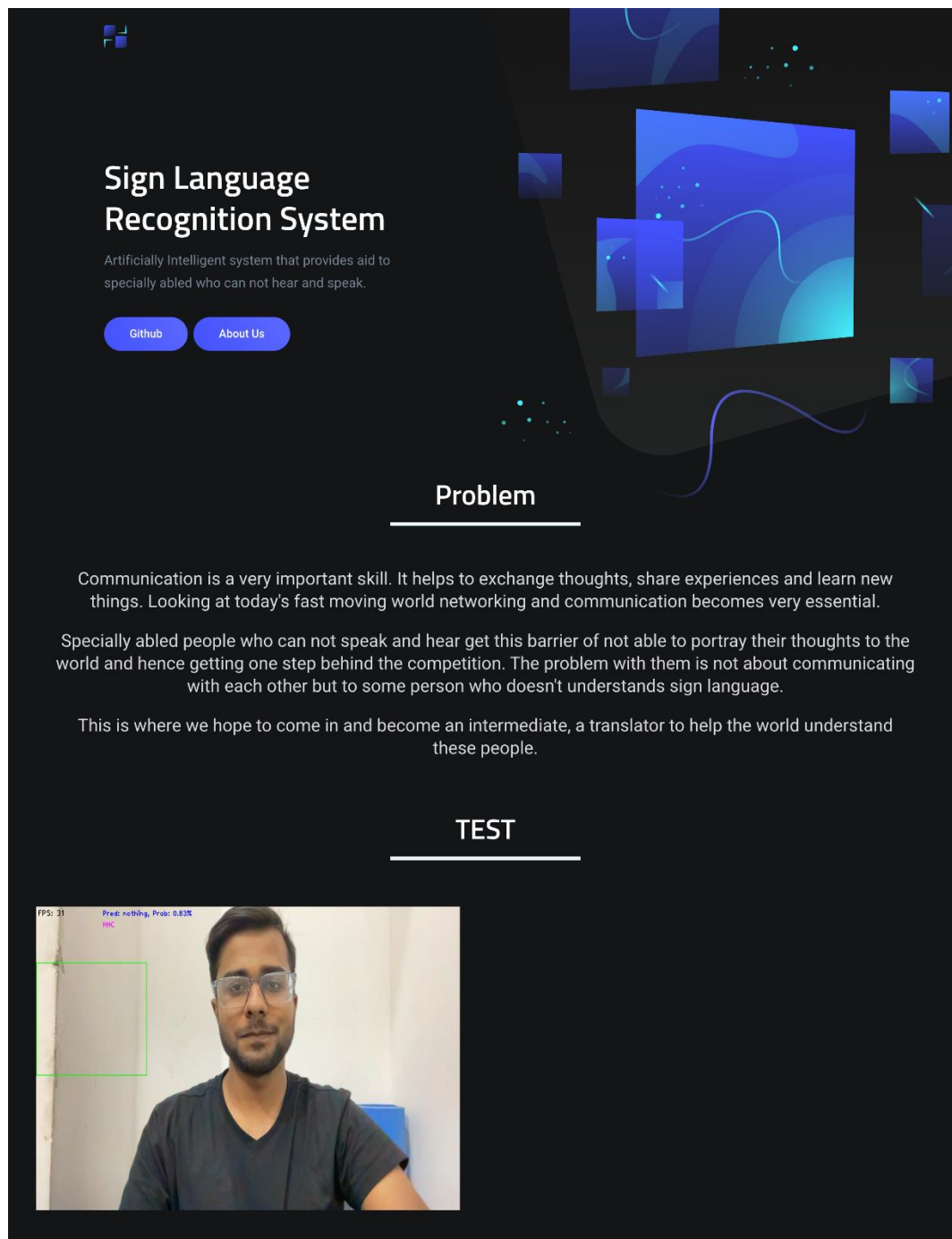


Fig 7.1: Home Page

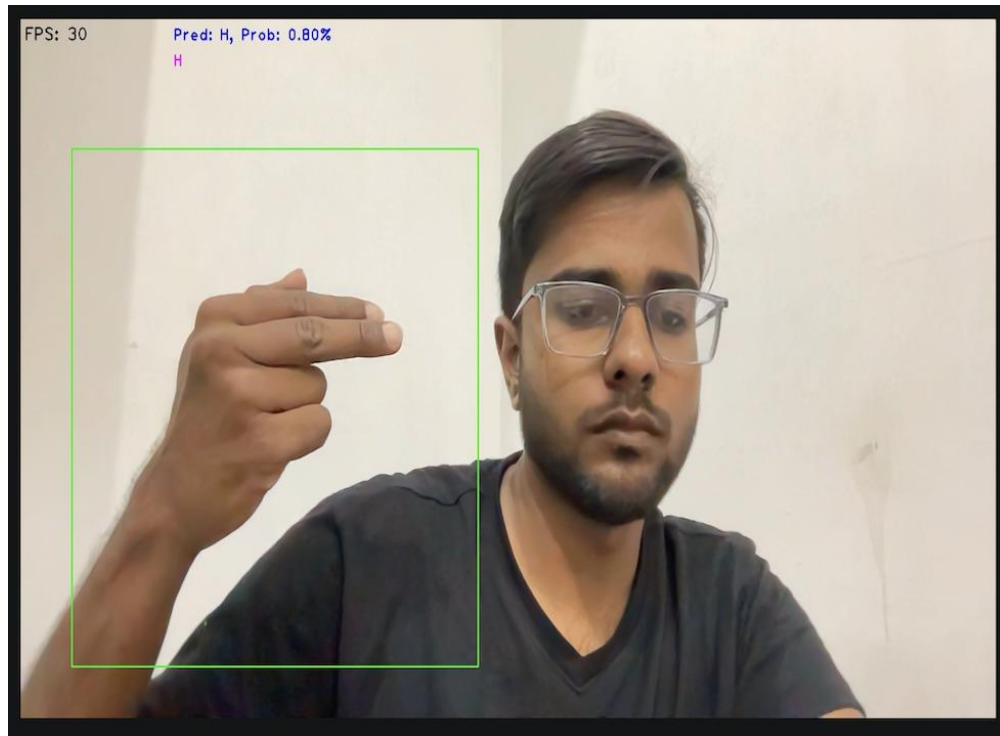


Fig 7.2: Output Window – ‘H’ for getting text “HELLO”

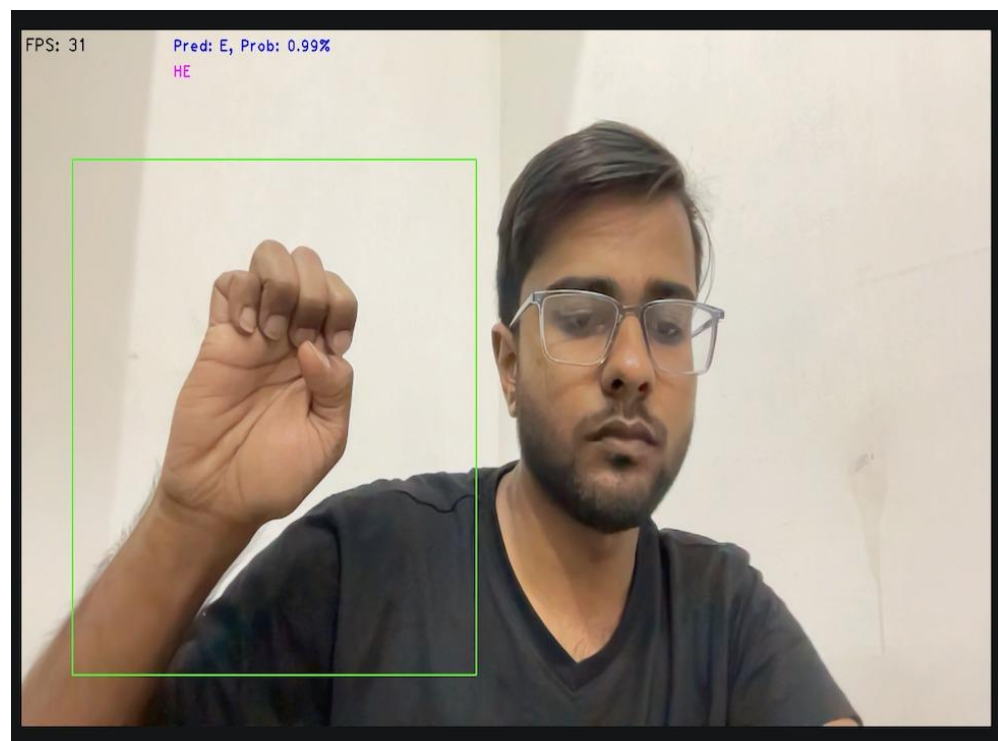


Fig 7.3: Output Window – ‘E’ for getting text “HELLO”



Fig 7.4: Output Window – ‘L’ for getting text “HELLO”



Fig 7.5: Output Window – ‘L’ for getting text “HELLO”

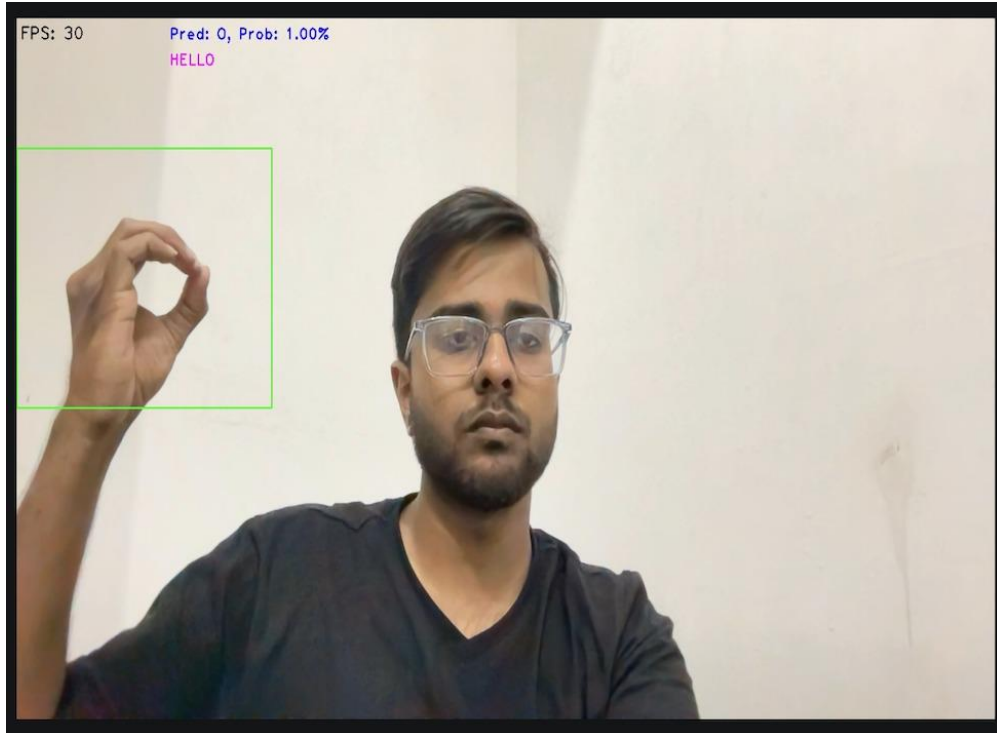


Fig 7.6: Output Window – ‘O’ for getting text “HELLO”

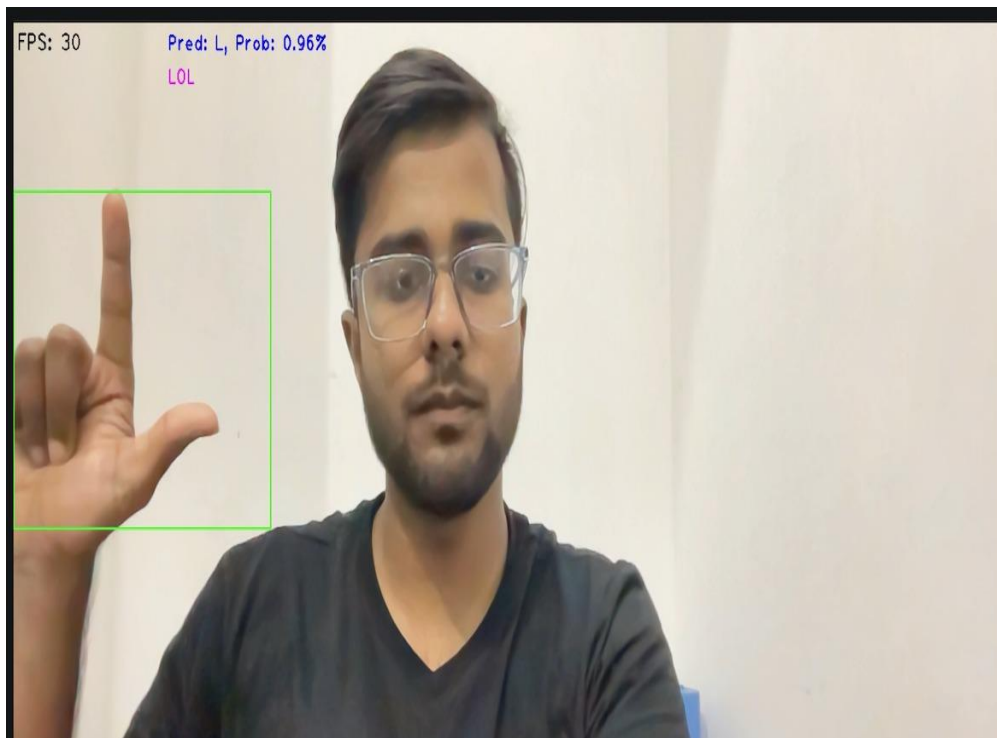


Fig 7.7: Output Window – “LOL”

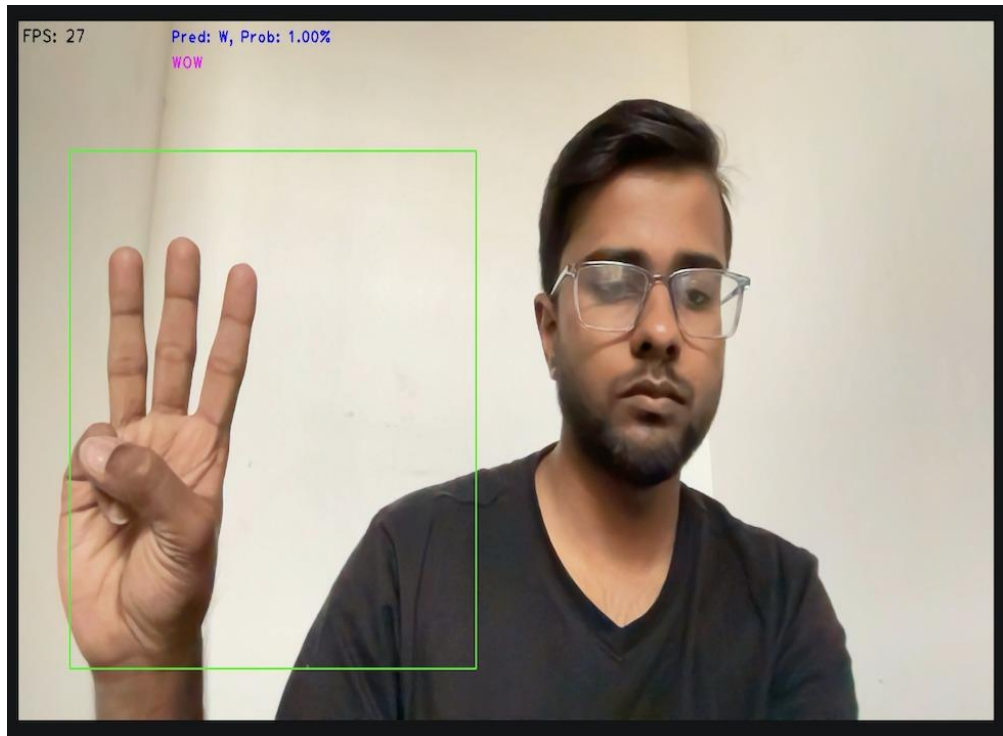


Fig 7.8: Output Window – “WOW”



Fig 7.9: Output Window – “HOW”

Chapter 8

Project Summary and Conclusions

8.1 Conclusion

In this project, we have implemented an automatic sign language gesture recognition system in real-time, using tools learnt in computer vision and machine learning. We learned about how sometimes basic approaches work better than complicated approaches. Some letters were harder to classify in our live demo such as "a" vs "i" since they only differ by a very small edge (the "i" has the pinky pointing up). Although our classification system works quite well as has been demonstrated through tables and images, there's still a lot of scope for possible future work. Increasing the training set size leads to drastic improvement in model performance. This likely increases the robustness of the model through learning more of the possible sample space. With a sufficiently big dataset of 69600 images available for training we were able to get up an accuracy of about 90.54% on the validation data of 17400 images. This clearly shows the power of transfer learning. The model works with a 30+ FPS on a GeForce-MX250. The system can provide an aid to the deaf and mute people with just a single click where they just have to load this website in their browser and start interacting and showing gestures about what they have to say or talk about word to word and the characters will be automatically visible on the screen. This person can now interact with anyone with ease and the job of intermediate human is excluded with making the whole communication process simpler.

Chapter 9

Future Scope

Possible extensions to this project would be extending the sign language conversion system to all alphabets of the ASL and other non-alphabet gestures as well. Having used MATLAB as the platform for implementation, we feel that we can also improve upon the speed of our real-time system by coding in C. The framework of this project can also be extended to several other applications like controlling robot navigation using hand gestures and the like.

A dataset with more variation and a higher quality can really boost the accuracy of our current models. Also, we think that using more complex models like artificial neural networks, or applying deep learning on the HOG vectors should improve the accuracy as they are capable of extracting richer information from these vectors. Increasing the levels of hierarchy with proper hierarchy levels created on the basis of which nodes are being misclassified can result in improvement in accuracy, but we cannot surely claim that as suggested by multiplicative probability rule in our experiment with hierarchical classification.

References

- [1] Tan, M., Le, Q. (2019, May). Efficient net: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.
- [2] J. Mach. Learn. Res. (2011) The TEX document, Adaptive Sub gradient Methods for Online Learning and Stochastic Optimization
- [3] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He (2019): A Comprehensive Survey on Transfer Learning
- [4] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu (2018) : A Survey on Deep Transfer Learning
- [5] Kiprono Elijah Koech (2020): Cross-Entropy Loss Function

Project Links

- **GitHub Repository**
<https://github.com/arunabh-jain/sign-language-recognition-system>