

## **Project Report**

*On*

### **Conversion of Sign Language to Speech for Teaching Support to Empowering Deaf & Dumb**



*Submitted in partial fulfillment of the requirements of the degree*

*Of*

**Bachelor of Technology**

*In*

**Computer Engineering**

Under the guidance of

**Dr. S.D. Samantaray**

**Submitted By:**

Ayush Pratap Singh      I.D. No. (56068)

Ansh Kumar      I.D. No. (56094)

Shivanshu Rawat      I.D. No. (56930)

***Department of Computer Engineering, College of Technology,  
Govind Ballabh Pant University of Agriculture and Technology,  
Pantnagar-263145, India  
June, 2024***

## **ACKNOWLEDGEMENT**

**“Conversion of Sign Language to Speech for Teaching Support to Empowering Deaf & Dumb”** is not the work of solely our team but it’s a result of combined efforts put in by many people. We would like to take this opportunity to thank all the people who helped us to carry out this project.

First of all, we express our gratitude to our project guide, Dr. S.D. Samantaray without whose help the project would not have been possible. It was his mentorship that encouraged us to expedite our project process and could complete it in time. His precious suggestions and constructive guidance has been indispensable in the completion of this project work.

We would also like to thank Prof. P.K. Mishra, Prof. Jalaj Sharma, Prof. Rajeev Singh, Prof. C.S. Negi, Prof B.K. Singh and Prof. Sunita Jalal for providing all the help and infrastructure needed for the successful completion of this project. They have supported us in this endeavor and appreciated us in our efforts during our project.

Last but not the least, we would also like to thank our friends and family who directly and indirectly supported us during the project work and provided a helpful environment for our project.

|                    |                  |       |
|--------------------|------------------|-------|
| Ayush Pratap Singh | I.D. No. (56068) | ..... |
| Ansh Kumar         | I.D. No. (56094) | ..... |
| Shivanshu Rawat    | I.D. No. (56930) | ..... |



## **CERTIFICATE**

This is to certify that the project work entitled “**Conversion of Sign Language to Speech for Teaching Support to Empowering Deaf & Dumb**” which is submitted by:

| <b>NAME</b>        | <b>ID. No.</b> |
|--------------------|----------------|
| Ayush Pratap Singh | 56068          |
| Ansh Kumar         | 56094          |
| Shivanshu Rawat    | 56930          |

is a record of a student's work carried by them in partial fulfillment of requirements for the award of degree of Bachelor of Technology in the Department of Computer Engineering, College of Technology, G.B. Pant University of Agriculture and Technology, Pantnagar.

**DATE:**

**(Dr. S.D. Samantaray)**  
**Project Guide**

*Department of Computer Engineering, College of Technology,  
Govind Ballabh Pant University of Agriculture and Technology,  
Pantnagar-263145, India*

## **APPROVAL**

This project report entitled “**Conversion of Sign Language to Speech for Teaching Support to Empowering Deaf & Dumb**” is hereby approved as a creditable study of an engineering subject, as a prerequisite to the degree for which it has been submitted.

### **Committee Members:**

1. Dr. S.D. Samantaray

(Head of Department of Computer Engineering)

-----

(Project Guide)

2. Dr. Rajeev Singh

(Associate Professor)

-----

3. Dr. P.K. Mishra

(Associate Professor)

-----

4. Prof. Jalaj Sharma

(Associate Professor)

-----

5. Prof. B.K. Singh

(Associate Professor)

-----

**Signature of Head of Department**

*Department of Computer Engineering, College of Technology,  
Govind Ballabh Pant University of Agriculture and Technology,  
Pantnagar-263145, India*

## **ABSTRACT**

This project addresses the crucial need for effective communication tools for deaf and dumb individuals who rely on sign language as their primary mode of interaction. Sign language, a visual and expressive form of communication using hand gestures, plays a pivotal role in enabling these individuals to convey thoughts and ideas nonverbally. However, the widespread understanding of sign language is limited among the general population, creating significant communication barriers. There are about 15 million such people in India. They are not in the mainstream because of communication problems. This project attempts to break this communication gap and empower them to become part of mainstream occupations.

This project proposes a solution in the form of a real-time sign language-to-speech conversion system, focusing specifically on Indian Sign Language (ISL). ISL, like other regional sign languages, has its own grammar and syntax, making it unique and tailored to local contexts. The system aims to facilitate seamless communication by recognizing and converting ISL signs related to Indian states and their capitals into spoken words. By bridging this gap between sign language and spoken language, the project seeks to enhance accessibility and inclusivity in educational settings and beyond, thereby empowering deaf and dumb individuals to participate more fully in classroom interactions and everyday communication.

The scarcity of teachers in Indian schools is a pressing issue, with a reported shortfall of nearly 1 million teachers, particularly in rural areas where the student-teacher ratio remains critically high. This gap significantly hampers the quality of education. According to the 2011 Census, there are about 2.68 crore disabled people in India, many of whom face barriers to employment, including teaching. Our project aims to empower individuals with hearing and speech impairments to become effective educators. By converting sign language into audible speech, this technology can enable them to teach in classrooms, thus helping to bridge the educational gap and integrate a significant portion of the disabled population into the mainstream workforce.

Utilizing state-of-the-art technologies such as Mediapipe for hand tracking, OpenCV for image preprocessing, and Long Short-Term Memory (LSTM) networks for gesture recognition, our system ensures high accuracy and real-time processing. The recognized gestures are converted into text and subsequently into audible speech using text-to-speech (TTS) technology, delivering clear and contextually accurate audio output.

By leveraging a high-resolution webcam and powerful software tools, we ensure compatibility across different platforms and safeguard user data with stringent security measures. The system was trained and tested on the data set comprising of Indian states and their capitals and the average accuracy was found to be 97% using the LSTM model. The performance of the developed system has been found higher than the existing methods applied to the same data sets previously.

## **TABLE OF CONTENTS**

|   |           |
|---|-----------|
| <b>1. Introduction.....</b>   | <b>08</b> |
| 1.1. History of Sign Languages.....                                       | 10        |
| 1.2. Types of Sign Languages Around the World.....                        | 11        |
| 1.3. Deep Learning Algorithms Available For Sign Language Conversion..... | 13        |
| 1.4. Project Aim and Objective.....                                       | 14        |
| 1.5. Advantages of Converting Sign Language to Speech.....                | 15        |
| <b>2. Literature Review.....</b>  | <b>17</b> |
| <b>3. Problem Specification.....</b>                                      | <b>21</b> |
| 3.1. Technical Feasibility.....   | 22        |
| 3.2. Behavioral Feasibility.....  | 22        |
| 3.3. Economic Feasibility.....  | 22        |
| 3.4. Operational Feasibility.....   | 22        |
| <b>4. Requirement Analysis.....</b>                                       | <b>23</b> |
| 4.1. Software Requirements.....   | 23        |
| 4.2. Hardware Requirements.....   | 24        |
| 4.3. Functional Requirements.....   | 25        |
| 4.4. Non-Functional Requirements.....                                     | 26        |
| <b>5. System Design.....</b>  | <b>27</b> |
| 5.1. System Flowchart.....  | 27        |
| 5.2. Data Flow Diagrams.....  | 29        |
| 5.3. Developing a User Interface.....                                     | 31        |
| <b>6. Implementation.....</b>   | <b>32</b> |
| <b>7. Testing.....</b>  | <b>41</b> |
| 7.1. Functional Testing.....  | 41        |
| 7.2. Structural Testing.....  | 41        |
| 7.3. Non-Functional Testing.....  | 42        |
| 7.4. Security Testing.....  | 42        |
| <b>8. Results and Discussion.....</b>                                     | <b>43</b> |
| 8.1. Installation Process.....  | 43        |
| 8.2. Project Screenshots and Use Case Example.....                        | 46        |
| 8.3. Opportunities and Challenges.....                                    | 51        |
| <b>9. Summary.....</b>  | <b>52</b> |
| <b>10. Literature Cited.....</b>  | <b>53</b> |
| <b>11. Bio-Data of Students.....</b>                                      | <b>54</b> |
| <b>APPENDIX I.....</b>  | <b>55</b> |
| <b>APPENDIX II.....</b>   | <b>56</b> |
| <b>APPENDIX III.....</b>  | <b>61</b> |

## **LIST OF FIGURES**

|  |    |
|--|----|
| Figure 01. Fingerspelling in ISL.....      | 09 |
| Figure 02. Word level sign vocabulary..... | 09 |
| Figure 03. English Alphabet in ISL.....    | 12 |
| Figure 04. System Flowchart Diagram.....   | 27 |
| Figure 05. Use case Diagram.....           | 29 |
| Figure 06. DFD Level 1 Diagram.....        | 29 |
| Figure 07. DFD Level 2 Diagram.....        | 30 |
| Figure 08. Sequence Diagram.....           | 31 |
| Figure 09. Applying Gaussian Blur.....     | 34 |
| Figure 10. Data Preprocessing.....         | 35 |
| Figure 11. Mediapipe Landmark System.....  | 36 |
| Figure 12. LSTM Basic Structure.....       | 36 |
| Figure 13. RNN vs. LSTM.....               | 38 |
| Figure 14. Gradient Problems in RNN.....   | 39 |
| Figure 15. Python Installer Window.....    | 43 |
| Figure 16. Python version check.....       | 43 |
| Figure 17. Installing pip.....             | 44 |
| Figure 18. Installing pytsx3.....          | 44 |
| Figure 19. Installing opencv-python.....   | 45 |
| Figure 20. Installing tensorflow.....      | 45 |
| Figure 21. Graphical User Interface.....   | 46 |
| Figure 22. Validation Window.....          | 47 |
| Figure 23. Real Time Validation.....       | 50 |

## **INTRODUCTION**

Communication is the process of exchanging thoughts and messages through various methods such as speech, signals, behavior, and visuals. For deaf and dumb individuals, this exchange relies heavily on the use of hand gestures to express different ideas and concepts nonverbally. These gestures, which are interpreted through vision, form the basis of sign language. Sign language is a comprehensive and structured form of communication, using specific hand movements and positions to convey words and sentences. It allows deaf and dumb people to effectively communicate with others, making it an essential tool for their interaction and expression. Different regions have their own versions of sign language, such as American Sign Language (ASL) and Indian Sign Language (ISL), each with its own syntax and grammar.

Sign language serves as a vital mode of communication for millions of deaf and hard-of-hearing individuals globally. However, a significant communication barrier exists due to the limited understanding of sign language among the general population. This project proposes a real-time sign language to speech conversion system specifically focused on Indian Sign Language (ISL) to bridge this gap. Our system aims to facilitate seamless communication by enabling real-time recognition and conversion of ISL signs representing Indian states and their capitals into spoken words.

Sign language is a visual language and consists of 3 major components:

| <b>Fingerspelling</b>                  | <b>Word level sign vocabulary</b>       | <b>Non-manual features</b>                              |
|--|---|---|
| Used to spell words letter by letter . | Used for the majority of communication. | Facial expressions and tongue, mouth and body position. |

This project aims to develop a system that converts sign language into speech using Long Short-Term Memory (LSTM) networks and MediaPipe for data capturing. This system is designed to assist individuals who are deaf or hard of hearing by translating their sign language gestures into spoken words, thus facilitating better communication with those who do not understand sign language.

**Objective:** To create a real-time application that accurately recognizes and translates sign language gestures into audible speech.

**Significance:** The project addresses the communication barrier faced by the deaf community, promoting inclusivity and improving their interactions with the broader society.

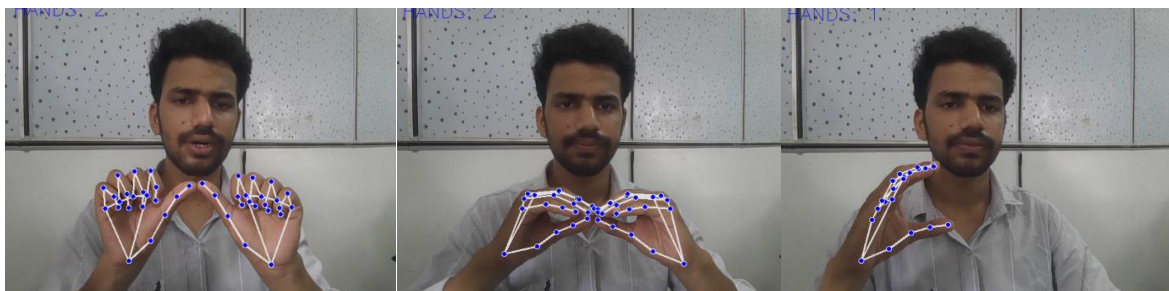


We leverage the power of deep learning for sign language classification. Our system utilizes OpenCV, a renowned open-source library, for real-time video capture. MediaPipe, another powerful open-source framework, is employed for accurate hand detection within the captured video frames. The keypoints extracted from these detected hands using MediaPipe provide a rich feature set that captures the nuances of hand gestures in ISL signs. These extracted keypoints are then stored as a feature vector database. This database serves as the training ground for our Long Short-Term Memory (LSTM) model. LSTMs are a type of recurrent neural network that excel at handling sequential data, making them ideal for tasks like sign language recognition where the order of hand movements plays a crucial role.

Our LSTM model is trained on a dataset specifically curated to recognize signs representing Indian states and their capitals in ISL. This domain-specific focus allows for a more targeted and efficient approach to sign language recognition. By successfully classifying these signs, the model can then convert them to the corresponding spoken words, enabling real-time communication for users of ISL.

For example:

1. Fingerspelling – This is used to spell words letter by letter. Below is the representation of ‘A’, ‘B’ and ‘C’ in Indian Sign Language (ISL).



A

B

C

**Figure 1. Fingerspelling in ISL**

2. Let’s convert the statement “Bhubaneswar is the capital of Odisha” to Indian Sign Language (ISL). Here, we will use the word level sign language vocabulary i.e., for every significant word we have corresponding signs.



Bhubaneswar

capital

Odisha

**Figure 2. Word level sign vocabulary**

## **1.1 HISTORY OF SIGN LANGUAGES**

Sign languages, far from being recent inventions, boast a rich and dynamic history interwoven with the experiences of deaf communities around the world. The following lines explore the fascinating journey of sign languages, tracing their early origins, the challenges and advancements they faced, and their eventual recognition as full-fledged languages.

### **Early Traces: The Dawn of Sign Language Use**

While pinpointing the exact origin of sign language remains elusive, evidence suggests its existence stretches back centuries, perhaps even millennia. One of the earliest references comes from a dialogue in Plato's *Cratylus* (5th century BC), where Socrates ponders the use of gestures for communication if speech were unavailable.

Historical accounts from various cultures also hint at the use of sign systems. Monks in some religious orders employed signing for silent communication during prayer or study. Native American tribes in North America, like the Great Plains tribes, developed elaborate sign languages that transcended spoken language barriers and facilitated communication between groups with different spoken languages.

However, it's important to distinguish between these early signing systems and modern sign languages. These early forms often lacked the complexity and grammatical structure that define modern languages. They might have primarily focused on conveying basic ideas or mimicking spoken words rather than serving as independent communication systems.

### **The Enlightenment and the Rise of Formalized Sign Education**

The 17th and 18th centuries witnessed a significant shift in attitudes towards deafness and sign language. Pioneering educators like Pedro Ponce de León in Spain and Juan Pablo Bonet in the 1600s began developing formalized methods of sign language instruction. Bonet's work, *Reducción de las letras de los sordos* (1620), even included the first documented manual alphabet, a system for fingerspelling letters using handshapes.

However, this period also saw a rise in the oralism movement, which advocated for teaching deaf people to speak and lipread, often at the expense of sign language. This approach, championed by figures like Samuel Heinicke in Germany, dominated deaf education for much of the 19th century. Oralism was based on the belief that speech was superior to sign language and that deaf people could be integrated into mainstream society by learning to speak.

### **The 19th Century: Struggles and the Fight for Recognition**

Despite the dominance of oralism, sign language persisted and continued to evolve within deaf communities. The 19th century saw the establishment of schools specifically for deaf students, fostering environments where sign language flourished. Deaf educators like Laurent Clerc in the United States played a pivotal role in promoting sign language and advocating for its use in education.

A pivotal moment arrived in 1880 with the International Congress on Education of the Deaf in Milan, Italy. This controversial congress overwhelmingly endorsed oralism, leading to a period

of decline in sign language use in educational settings. However, the decision also galvanized the deaf community, leading to a growing movement for the recognition of sign language as a legitimate language in its own right.

### **The 20th Century and Beyond: Recognition and Flourishing**

The 20th century witnessed a gradual shift back towards the acceptance and appreciation of sign language. Linguistic research began to document the complex grammar and structure of sign languages, dismantling the notion that they were merely rudimentary forms of communication.

The latter half of the 20th century saw a resurgence of sign language use in deaf education. The signing rights movement gained momentum, advocating for equal access to education and communication in sign language. National sign languages, distinct from spoken languages of their geographical regions, gained recognition. Indian Sign Language (ISL), for instance, emerged as a distinct language with its own grammar and vocabulary.

Technological advancements further empowered deaf communities. Closed captioning on television and video relay services enabled greater access to information and communication. The rise of the internet created new online communities where deaf people could connect and share their language and culture.

### **The Future of Sign Languages: Continued Growth and Innovation**

Today, sign languages are thriving languages used by millions of deaf and hard-of-hearing individuals worldwide. Efforts are underway to document and preserve lesser-known sign languages at risk of disappearing. Technological advancements like machine translation for sign language hold promise for further breaking down communication barriers.

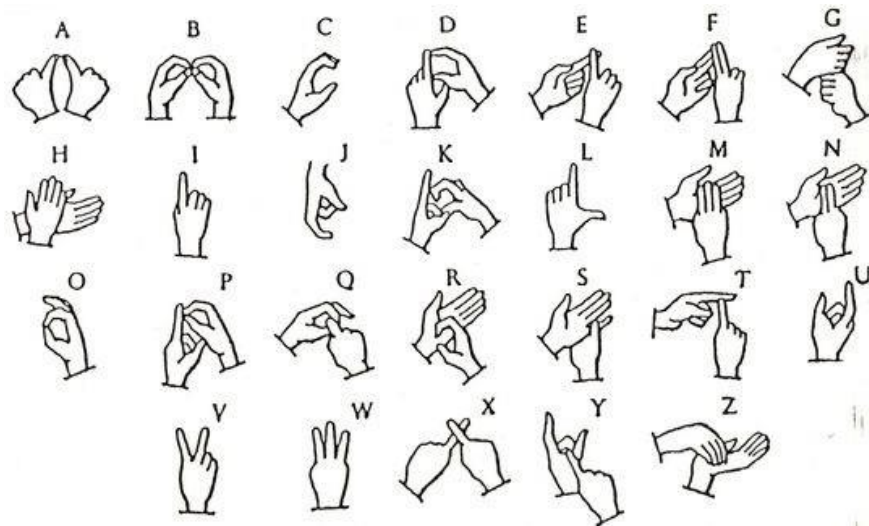
The history of sign languages is a testament to the resilience and ingenuity of deaf communities. From their early roots to their modern-day recognition, sign languages have continuously evolved and adapted, serving as vibrant communication systems and expressions of cultural identity for deaf people around the world.

## **1.2 TYPES OF SIGN LANGUAGES AROUND THE WORLD**

Sign languages are rich, fully developed languages that use visual-manual modality to convey meaning. They have their own syntax, grammar, and lexicon. Each country or region typically has its own sign language, which has evolved to suit the cultural and linguistic needs of its users. Below are some of the most recognized types of sign language across the globe.

### **Indian Sign Language (ISL)**

Indian Sign Language (ISL) is used across the Indian subcontinent. It is a standardized form of sign language that incorporates regional variations. ISL is increasingly being recognized and promoted to improve accessibility and communication for the deaf community in India.



**Figure 1. English Alphabet in ISL**

### **American Sign Language (ASL)**

American Sign Language (ASL) is predominantly used in the United States and parts of Canada. It has a unique grammar and syntax, different from English. ASL is a natural language with its own set of rules for pronunciation, word order, and complex grammar. It is not a direct translation of English but a fully developed language that can express abstract concepts and emotions.

### **British Sign Language (BSL)**

British Sign Language (BSL) is the sign language used in the United Kingdom. Like ASL, BSL has its own syntax and grammar, which are quite different from spoken English. One notable difference between BSL and ASL is the use of two hands for spelling out words in BSL, whereas ASL uses one hand.

### **Australian Sign Language (Auslan)**

Australian Sign Language (Auslan) is used in Australia and is closely related to BSL. Auslan incorporates elements of Irish Sign Language (ISL) as well. Like other sign languages, Auslan has its own grammar and lexicon, which are different from the English language.

### **International Sign (IS)**

International Sign (IS) is a pidgin form of sign language that is used at international meetings and events, such as the World Federation of the Deaf conferences. IS is not a natural language but a simplified, highly visual form of communication that incorporates elements from various sign languages to facilitate understanding across different linguistic backgrounds.

## 1.3 DEEP LEARNING ALGORITHMS AVAILABLE FOR SIGN LANGUAGE CONVERSION

Sign language conversion, the process of translating sign language gestures into spoken language or text, has become a crucial area of research with the potential to revolutionize communication accessibility for deaf and hard-of-hearing individuals. Deep learning algorithms, with their ability to learn complex patterns from vast amounts of data, are at the forefront of this endeavor. This section explores some of the prominent deep learning algorithms employed in sign language conversion systems.

### 1.3.1 Convolutional Neural Networks (CNNs) for Recognizing Hand Gestures

Convolutional Neural Networks (CNNs) excel at image and video recognition tasks, making them well-suited for the initial stage of sign language conversion – identifying hand gestures within video frames. Here's how CNNs contribute:

**Feature Extraction:** CNNs automatically learn relevant features from video frames. These features might capture the shape, orientation, and movement of hands, providing a robust representation of the hand gestures in a sign.

**Spatial Relationships:** CNNs can inherently capture the spatial relationships between different parts of the hand within a frame. This is crucial for differentiating between signs that rely on subtle variations in hand posture.

By processing video frames through a CNN, the system can extract a high-level representation of the hand gesture, forming the foundation for subsequent classification or translation tasks.

### 1.3.2 Recurrent Neural Networks (RNNs) for Capturing Temporal Dynamics

While CNNs excel at capturing spatial features, sign language also involves a temporal component – the sequence of hand movements that define a particular sign. Here's where Recurrent Neural Networks (RNNs) come into play:

**Sequential Data Processing:** Unlike CNNs, RNNs are specifically designed to handle sequential data like sign language gestures. They process information from each video frame while considering the context of previous frames, allowing them to capture the temporal dynamics of hand movements.

**Long Short-Term Memory (LSTM) Networks:** A specific type of RNN called Long Short-Term Memory (LSTM) networks is particularly adept at dealing with long-term dependencies in sequential data. This is crucial for sign language recognition, where the order of hand movements can significantly impact the meaning of the sign.

By feeding the CNN-extracted features from each video frame into an RNN or LSTM network, the system can learn the temporal sequence of hand movements and classify the overall sign language gesture.

### 1.3.3 Encoder-Decoder Architectures for Sign Language Translation

Once a sign language gesture is recognized, the next step is to convert it into spoken language or text. This translation task is often tackled using encoder-decoder architectures:

**Encoder-Decoder Structure:** An encoder network, typically a CNN or RNN, processes the video frame sequence representing the sign language gesture. This network condenses the information into a latent representation capturing the essence of the sign.

**Decoder Network:** A decoder network, often another RNN, takes the latent representation from the encoder and generates the corresponding spoken language or text output. The decoder network might utilize a vocabulary of words or phonemes to construct the translated output.

Encoder-decoder architectures, particularly those incorporating attention mechanisms that allow the decoder to focus on specific parts of the encoded representation, have shown significant promise in sign language translation tasks.

### 1.3.4 Generative Adversarial Networks (GANs) for Sign Language Synthesis

An emerging area of research explores using Generative Adversarial Networks (GANs) for sign language conversion. GANs consist of two competing neural networks:

**Generator Network:** This network aims to generate realistic-looking video of a person signing a specific word or phrase based on the provided spoken language or text input.

**Discriminator Network:** This network acts as a critic, evaluating the generated video and attempting to distinguish it from real sign language videos.

Through an adversarial training process, the generator network continuously improves its ability to produce realistic sign language video outputs based on the spoken language or text input. While still under development, GAN-based approaches hold promise for creating more natural-looking sign language representations.

Deep learning algorithms offer a powerful toolkit for sign language conversion. From recognizing hand gestures with CNNs to capturing temporal dynamics with RNNs and translating signs with encoder-decoder architectures, these algorithms are pushing the boundaries of communication accessibility. As research progresses, we can expect even more sophisticated deep learning techniques to emerge, paving the way for seamless and natural sign language conversion systems.

## **1.4 PROJECT AIM AND OBJECTIVES**

The aim of our project is to develop an advanced and user-friendly system that translates Indian Sign Language (ISL) gestures into spoken language, thereby facilitating enhanced communication and educational opportunities for deaf students. This project specifically focuses on teaching students about Indian states and their capitals.

## **1.5 ADVANTAGES OF CONVERTING SIGN LANGUAGE TO SPEECH**

The ability to convert sign language to speech offers a range of significant advantages, fostering greater inclusivity and communication accessibility for deaf and hard-of-hearing individuals. This section explores some of the key benefits of this technology.

### **1.5.1 Enhanced Accessibility and Communication**

One of the most compelling advantages of sign language to speech conversion lies in its ability to bridge the communication gap between deaf and hearing communities. In everyday interactions, such as ordering food at a restaurant, asking for directions, or participating in meetings, sign language conversion technology can provide real-time translation, enabling deaf individuals to engage more confidently and independently.

This improved accessibility extends to educational settings as well. Students who use sign language can benefit from lectures and classroom discussions being simultaneously converted to speech, fostering a more inclusive learning environment.

### **1.5.2 Breaking Down Barriers in Employment and Social Interactions**

Sign language conversion technology can empower deaf and hard-of-hearing individuals in the workplace. During job interviews, meetings, or presentations, real-time conversion can ensure clear and equal communication, promoting greater participation and career advancement.

Social interactions, which can sometimes be challenging for deaf individuals due to communication barriers, can also be significantly enhanced by sign language conversion technology. Casual conversations, attending cultural events, or simply interacting with strangers become more accessible and less isolating.

### **1.5.3 Reduced Reliance on Sign Language Interpreters**

While sign language interpreters play a vital role in facilitating communication, their availability can be limited, particularly in certain situations or locations. Sign language conversion technology offers an alternative solution, potentially reducing dependence on interpreters and increasing communication autonomy for deaf individuals.

However, it's important to acknowledge that sign language conversion technology is not intended to replace sign language interpreters entirely. Interpreters offer a nuanced understanding of both spoken and signed languages and can provide context and cultural awareness that machine translation may struggle with.

#### **1.5.4 Emergency Situations and Improved Safety**

Sign language conversion technology can be a valuable tool in emergency situations. During a fire alarm, medical emergency, or any situation requiring clear and immediate communication, the ability to convert sign language to speech can ensure that deaf individuals receive critical information and instructions promptly.

This technology can also enhance overall safety by enabling deaf individuals to access important announcements or warnings that might otherwise be communicated verbally.

#### **1.5.5 Promoting Inclusivity and Social Integration**

Sign language conversion technology fosters a more inclusive society by removing communication barriers between deaf and hearing individuals. This inclusivity extends to various aspects of life, from education and employment to social interactions and public services.

By promoting understanding and facilitating communication, sign language conversion technology can help break down social stigma and empower deaf individuals to participate more fully in all areas of society.

In conclusion, sign language conversion technology offers a powerful tool for bridging the communication gap between deaf and hearing communities. The advantages of this technology range from improved accessibility and social integration to enhanced safety and independence for deaf individuals. As the technology continues to develop and become more sophisticated, its potential to empower deaf communities and create a more inclusive world will only continue to grow.



## **LITERATURE REVIEW**

### **2.1 Methods proposed by experts:**

#### **1. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton**

They proposed a solution based on Computer Vision and Machine Learning. This approach forms the core of many sign language conversion systems. A foundational paper by Yann LeCun, Yoshua Bengio, and Geoffrey Hinton titled "Deep Learning" (2015) provides a comprehensive overview of Deep Learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) that are extensively used for feature extraction and classification in sign language recognition. Additionally, researchers like Jingyi Li et al. in their paper "Convolutional Neural Networks for Sign Language Recognition: A Review" (2017) [IEEE Access] delve deeper into the application of CNNs for sign language tasks.

#### **2. Karen D Xu, PS Stewart, F Xia, CT Huang**

While CNNs excel at capturing spatial information within a single frame, sign language recognition requires understanding the temporal dynamics of hand movements over time. This is where RNNs, particularly LSTMs, come into play. As highlighted in research by Karen Xu et al. titled "Sign Language Recognition with Recurrent Neural Networks" (2016), LSTMs possess a unique architecture that allows them to learn long-term dependencies within sequential data like sign language gestures. By processing a sequence of video frames, LSTMs can capture the evolving hand postures and movements that define a specific sign.

#### **3. Efstratios Ioannidis et al.**

This approach is explored in research by Efstratios Ioannidis et al. titled "Wearable Sensor-Based Systems for Sign Language Recognition" (2018) [Sensors]. They propose a system using wearable sensors on the hands and arms for sign language recognition, highlighting the advantages of this approach in specific scenarios.

#### **4. Sudipta Roy et al.**

Glove-based techniques are investigated in a paper by Sudipta Roy et al. titled "A Novel EMG-based System for Sign Language Recognition" (2019) [Biomedical Signal Processing and Control]. Their research explores using data gloves equipped with electromyography (EMG) sensors to capture muscle activity data for sign recognition.

## **5. Mohammad Khamis et al.**

The merits of combining different approaches are addressed in research by Mohammad Khamis et al. titled "Sign Language Recognition Using Sensor Fusion and Deep Learning" (2020) [IEEE Transactions on Industrial Electronics]. Their work explores integrating computer vision with sensor data for sign language recognition, demonstrating the potential benefits of hybrid systems.

### **2.2 Basic steps in involved in the general solution:**

In the recent years there has been tremendous research done on the hand gesture recognition.

With the help of literature survey done we realized the basic steps in hand gesture recognition are :-

- Data acquisition
- Data preprocessing
- Feature extraction
- Gesture classification

#### **2.2.1 Data acquisition**

The different approaches to acquire data about the hand gesture can be done in the following ways:

##### **1. Use of sensory devices**

This method employs electromechanical devices to accurately capture hand configuration and position. Various glove-based techniques can extract detailed information, but they tend to be expensive and lack user-friendliness. These devices are typically used in specialized applications, such as virtual reality or high-end motion capture, where the precision they offer justifies their cost and complexity.

##### **2. Vision based approach**

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

The main challenge of vision-based hand detection is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin-colour possibilities as well as to the variations in view points, scales, and speed of the camera capturing the scene.

### **3. Ultrasound-Based Systems**

Ultrasound-based systems use ultrasonic sensors to detect hand position and movement by emitting and receiving sound waves. Devices such as ultrasonic rings or bracelets can provide gesture recognition without requiring visual line of sight, which is beneficial in scenarios where visual tracking is impractical. However, these systems have limited range and can be susceptible to interference from other ultrasonic sources, potentially affecting accuracy. Despite these limitations, ultrasound-based systems offer a unique solution for gesture recognition in specific contexts.

#### **2.2.2 Data preprocessing and Feature extraction for vision based approach:**

The various approaches and techniques for the data preprocessing and feature extraction task for vision based approach have been discussed below:

- In [1], the approach for hand detection combines threshold-based color detection with background subtraction. To distinguish between faces and hands, an Adaboost face detector can be utilized, as both involve similar skin tones. This method improves accuracy by effectively isolating the hand regions in images.
- To extract the necessary image for training, a Gaussian blur filter can be applied using OpenCV, as described in [3]. This filter helps reduce noise and detail in the image, making it easier to focus on the significant features required for hand gesture recognition.
- As mentioned in [4], instrumented gloves can be used to capture hand movements more accurately. This approach reduces the computation time for preprocessing and provides more concise and accurate data compared to applying filters on video-extracted data. The gloves directly measure hand configurations, offering high precision in gesture data collection.
- We have utilized the Mediapipe handtracking module to capture hand keypoints. This module efficiently detects hand landmarks in real-time and stores the relevant hand keypoints in a numpy array. This structured format facilitates easy manipulation and analysis of the data, enhancing the overall accuracy and efficiency of the hand gesture recognition process.

#### **2.2.3 Gesture classification:**

Gesture Classification is the main step involved in the process of conversion of sign language to text. Following are the methods for gesture classification:

- In [1], Hidden Markov Models (HMM) are employed for classifying gestures, addressing the dynamic aspects of gestures. The system extracts gestures from video

sequences by tracking skin-color blobs corresponding to the hand in a body-face space centered on the user's face. The objective is to recognize two classes of gestures: deictic (pointing) and symbolic. The image undergoes filtering using a fast look-up indexing table, after which skin-color pixels are clustered into blobs. These blobs are statistical objects based on the location ( $x, y$ ) and the colorimetry ( $Y, U, V$ ) of the skin-color pixels, facilitating the identification of homogeneous areas.

- In [2], a Naïve Bayes Classifier is utilized for static hand gesture recognition, offering an effective and fast method. This approach classifies different gestures based on geometric invariants derived from image data post-segmentation, making it independent of skin color. Gestures are extracted from each video frame against a static background. The process involves segmenting and labeling the objects of interest, extracting geometric invariants, and classifying gestures using a K-Nearest Neighbor algorithm with distance weighting (KNNDW) to provide data for a locally weighted Naïve Bayes classifier.

- According to a paper titled “Human Hand Gesture Recognition Using a Convolution Neural Network” by Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen from the Institute of Automation Technology at National Taipei University of Technology, the authors construct a skin model to isolate the hand in an image and then apply binary thresholding. After obtaining the thresholded image, they calibrate it around the principal axis to center the image. This centered image is then fed into a Convolutional Neural Network (CNN) model for training and prediction. Their model, trained on seven hand gestures, achieves an accuracy of approximately 95% for these gestures.

- Sign Language Recognition with Recurrent Neural Networks (2016) by Karen Xu et al. This paper explores the use of Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, for sign language recognition. The research demonstrates the advantage of LSTMs in handling sequential data like sign language gestures. LSTMs can learn the temporal dynamics of hand movements over time, which is crucial for accurate classification.

## **PROBLEM SPECIFICATION**

Sign languages serve as vital communication systems for millions of deaf and hard-of-hearing individuals globally. However, a significant communication barrier exists due to the limited understanding of sign language among the general population. This disparity can have profound implications on the daily lives of deaf individuals, hindering their ability to:

- **Access Information and Education:** Educational institutions and workplaces often rely heavily on spoken communication. Without access to sign language interpreters or real-time conversion technology, deaf individuals face challenges in understanding lectures, participating in discussions, or receiving critical information during meetings and presentations.
- **Engage in Social Interactions:** Casual conversations, attending cultural events, or simply interacting with strangers can be significantly more difficult for deaf individuals due to the communication barrier. This lack of accessibility can lead to social isolation and hinder opportunities for building relationships.
- **Maintain Independence and Safety:** Everyday situations like ordering food at a restaurant, asking for directions, or interacting with emergency services can become obstacles for deaf individuals if there's no way to communicate their needs effectively. This reliance on others can limit their independence and autonomy.

The problem specification for sign language to speech conversion lies in addressing these communication barriers. We need a reliable and efficient system that can accurately translate sign language gestures into spoken words in real-time. This system should ideally:

- **Recognize a Wide Range of Signs:** The system should be able to handle a comprehensive vocabulary of signs used in everyday communication, including regional variations and slang.
- **Account for Signer Independence:** Signing styles can vary depending on the individual and region. The system should be robust enough to recognize signs accurately regardless of the signer's specific style.
- **Function in Real-Time:** For seamless communication, the conversion process needs to happen with minimal delay, allowing for natural back-and-forth conversations.
- **Be Adaptable to Different Environments:** The system should function effectively in various lighting conditions and background noise levels to ensure its practicality in real-world scenarios.
- **Offer User-Friendly Interfaces:** The system's design should be user-centric, catering to the needs and preferences of deaf individuals. This could involve options for different output modalities (visual cues, text displays) or adjustable recognition sensitivity.

Developing a sign language to speech conversion system that meets these specifications has the potential to significantly improve the lives of deaf and hard-of-hearing individuals. It can

foster greater inclusivity, communication accessibility, and independence, empowering them to participate fully in all aspects of society.

The primary aim of this project is to enhance the educational experience for deaf students by providing an accessible and interactive tool for learning about Indian states and their capitals. By converting ISL gestures into speech, the system bridges the communication gap and enables students to engage with geographic content in a meaningful way. This not only aids in their academic development but also promotes inclusivity and equal access to education.

### **3.1 Technical Feasibility**

The proposed system is technically feasible as it can be developed easily with the help of available technology. However, significant problems can be caused by gathering the test data. The test data gathered by the team was gathered from kaggle.com and the rest was captured and modified to fulfill the requirements. Also significant resources are needed to train the resources for instance, enough RAM and a GPU of 16 GB VRAM. The latter can be taken care of by Google Collab which provides these resources but with a time constraint.

### **3.2 Behavioral Feasibility**

Psychologically it is observed that people are inherently resistant to change and computers have been known to facilitate change. This system is meant to reduce the time required to sort fruits based on human consumption and can be looked upon by the employees or the workers as a threat. However, this will not only reduce the time required to sort fruits but also help the company allocate the same manpower to different areas in the company hence optimizing the workforce. The system is quite easy to use and learn due to its simple interface. Users require no special training for operating the system.

### **3.3 Economic Feasibility**

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and savings that are expected from a system and compare them with cost, a decision is made to design and implement the system. This part of the feasibility study gives the economic justification of the system. The system being developed is economic with respect to School or Colleges point of view. It is cost effective in the sense that it has eliminated the paper work completely. The system is also time effective because the calculations are automated which are made at the end of the month or as per the user requirement. The result obtained contains minimum errors and are highly accurate as the data is required.

### **3.4 Operational Feasibility**

Automation makes our life easy. The proposed system is highly user friendly and is much easier to interact with. Only little instruction is required for the user so that he can easily operate the system.

## **REQUIREMENT ANALYSIS**

### **4.1 Software Requirements**

#### **4.1.1 Python**

Python is the primary programming language used for developing the solution. It is chosen for its simplicity, versatility, and extensive libraries that support machine learning, computer vision, and data manipulation.

**Version:** Python 3.7 or higher is recommended to ensure compatibility with the latest libraries and features.

#### **4.1.2 Packages/Libraries**

- **Mediapipe:**

Mediapipe is a cross-platform framework developed by Google for building multimodal machine learning pipelines. In this project, it is used for real-time hand-keypoint and landmark detection.

Mediapipe provides pre-trained models for hand tracking, which accurately detect and track hand landmarks in video streams. This is crucial for capturing the necessary data for gesture recognition.

- **Keras:**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It simplifies the process of building and training neural networks.

Keras is used for compiling and training the LSTM (Long Short-Term Memory) based neural network, which is essential for recognizing the temporal patterns in sign language gestures.

- **OpenCV:**

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It contains more than 2500 optimized algorithms for various computer vision tasks.

OpenCV is utilized for image processing tasks such as applying filters, preprocessing the video frames, and enhancing the quality of the captured images before feeding them into the neural network.

- **Numpy:**

Numpy is a fundamental package for scientific computing with Python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Numpy is used for performing array operations, handling numerical data, and managing the hand keypoints data captured by Mediapipe. It helps in efficient data manipulation and storage.

## **4.2 Hardware Requirements**

### **4.2.1 CPU**

A robust CPU is essential for handling the computational load required for real-time video processing and neural network operations.

**Specification:**

Intel: Intel Core i3 or greater.

Apple: Apple silicon chip M1 or greater.

These CPUs provide the necessary processing power to run the machine learning models and perform image processing tasks efficiently. They ensure smooth execution and quick response times, which are crucial for real-time applications.

### **4.2.2 Memory**

Adequate RAM is necessary to handle the data processing and storage requirements of the application.

**Specification:** 8GB or more.

Sufficient memory ensures that the system can manage large arrays of image data and intermediate computations without slowing down or crashing. It supports the seamless execution of resource-intensive tasks.

### **4.2.3 Camera or Webcam**

A camera or webcam is required to capture the hand gestures in real-time. Any standard camera or webcam capable of capturing high-resolution video would suffice.

High-quality video input is essential for accurate hand landmark detection and gesture recognition. A good camera ensures that the system receives clear and detailed images, which improves the overall performance and accuracy of the gesture recognition process.



These software and hardware components together form the foundation of our system, enabling it to function effectively and provide accurate and real-time translations from sign language to spoken language.

### 4.3 Functional Requirements

Functional Requirements Specification describes what is required to meet the users' business needs. It specifies which actions the design must provide in order to benefit the system's users. These are determined by the needs, user, and task analysis of the current system. Functional requirements needed in this Conversion of Sign language to speech using LSTM Networks are as follows:

- **Hand Gesture Detection:**

The system accurately detects hand gestures using a webcam or other video input devices. This involves implementing the Mediapipe handtracking module to capture hand keypoints in real-time. The system is capable of distinguishing between different gestures used in Indian Sign Language (ISL).

- **Gesture Preprocessing:**

The raw hand tracking data is converted into a usable format for further processing. This involves normalizing and filtering the data to reduce noise and improve accuracy. Additionally, the system segments the gesture sequences to clearly identify individual gestures.

- **Gesture Recognition and Classification:**

Long Short-Term Memory (LSTM) networks have been used to classify gestures based on their dynamic sequences. The system handles variability in gesture execution, such as different speeds and angles, and ensure high accuracy in recognizing specific ISL gestures associated with Indian states and their capitals.

- **Speech Synthesis:**

Recognized gestures are converted into corresponding text, which are then used by text-to-speech (TTS) technology to generate audible speech output. The speech synthesis is clear and contextually accurate due to the use of pyttsx3 module, particularly for educational content related to Indian states and capitals.

- **Real-Time Processing:**

The system is able to process video input and output speech in real-time, ensuring minimal latency to provide immediate feedback and translation.

- **User Interface:**

A user-friendly interface has been developed, accessible to deaf students. This interface displays real-time feedback of recognized gestures and corresponding speech output, along with options for users to start, pause, and stop the recognition process.

## **4.4 Non-Functional Requirements**

Non-functional requirements are requirements that are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time etc. They may specify system performance, security, availability, and other emergent properties. This means that they are often more critical than individual functional requirements. System users can usually find ways to work around a system function that doesn't really meet their needs. However, failing to meet a non-functional requirement can mean that the whole system is unusable. Non-functional requirements needed in Conversion of Sign language to speech using LSTM Networks are as follows:

- **Efficiency:**

The proposed deep learning based system provided an accuracy of about 97% and proved to be much more efficient when we used the LSTM training model instead of CNN model which had an accuracy of below 80%.

- **Accuracy:**

The deep learning based model is reliable and provides results accurately 97% of the time. The model was tested with two different deep learning based artificial convolutional neural networks, but the LSTM model proved to be more reliable. Thus, we used that model in our testing and implementation.

- **Maintainability**

It is fair to say that a system that undergoes changes with time is better suited and preferred over others. The system that we proposed has the capability to undergo changes and bug fixes in the future.

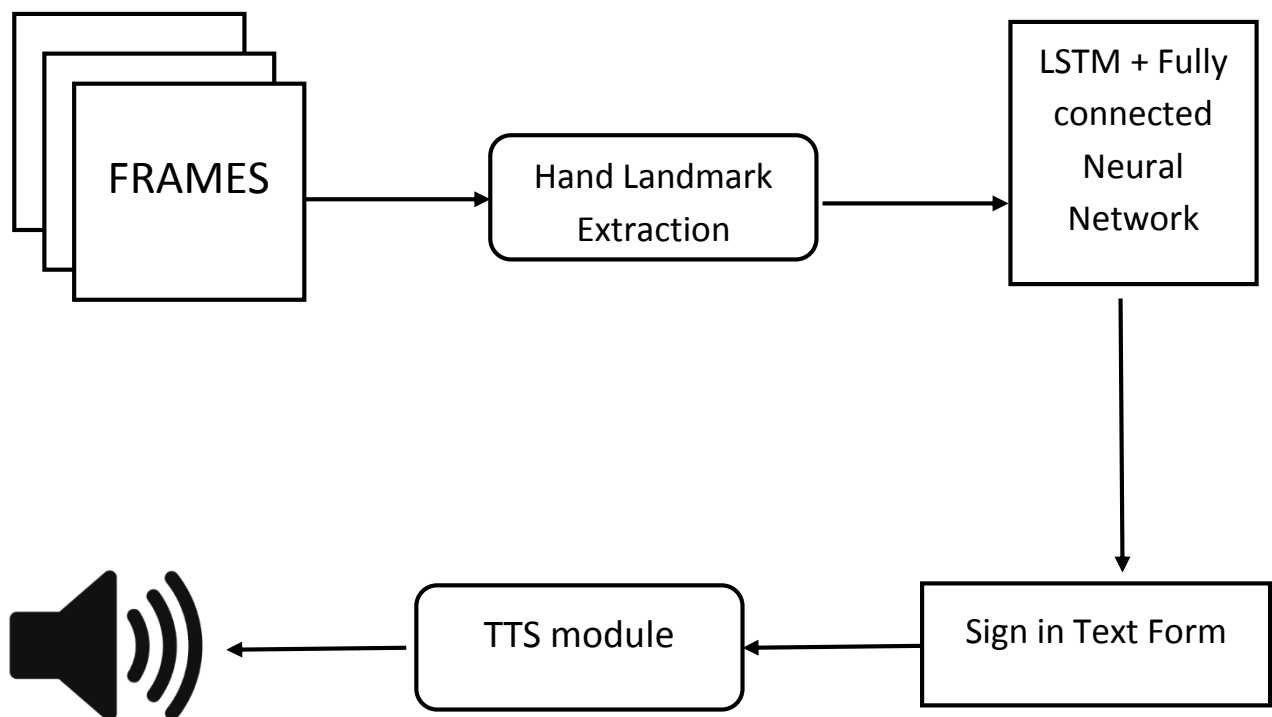
- **Portability**

It is the usability of the same software in different environments. The prerequisite of portability is the generalized abstraction between the application logic and the system interfaces. The proposed system fulfills the portability requirement. The system that we proposed can be used in different environments. The model can be used on different platforms including Linux, Windows, macOS etc.

## SYSTEM DESIGN

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to an effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel.

### 5.1 System Flowchart



**Figure 4. System Flowchart Diagram**

The design for the project can be understood through the following components, as depicted in the above flowchart:

**1. Frames:** This is the initial stage where frames are captured from the real-time video input. Each frame contains an image of the hand gesture being performed.

The frames act as the raw data input for the system. These are the individual snapshots taken from the continuous video stream provided by the camera.

## **2. Hand Landmark Extraction**

This component uses Mediapipe's hand-tracking model to detect and extract key landmarks of the hand from each frame.

Mediapipe processes the frames and identifies the key points (landmarks) on the hand, such as finger tips, joints, and the wrist. The extracted landmarks are represented as coordinates, which are then used as features for further processing.

## **3. LSTM + Fully Connected Neural Network (NN)**

The core of the system where the sequence of hand landmarks is analyzed to recognize the sign language gestures.

**LSTM (Long Short-Term Memory):** This type of recurrent neural network is well-suited for learning temporal patterns. It processes the sequence of hand landmarks over time to capture the dynamic aspect of sign language gestures.

**Fully Connected NN:** After the LSTM processes the sequence, the output is passed to a fully connected neural network layer that classifies the gesture based on the learned patterns.

The combination of LSTM and a fully connected network enables the system to accurately recognize complex sign language gestures from the temporal sequence of hand movements.

## **4. Sign in Text Form**

The recognized gesture is converted into a textual representation.

Once the LSTM and the fully connected network classify the gesture, the corresponding text (e.g., the name of an Indian state or its capital) is generated. This text represents the meaning of the recognized sign.

## **5. TTS Module (Text-to-Speech)**

This module converts the text output into spoken words.

The textual representation of the recognized gesture is fed into a text-to-speech engine. The TTS module synthesizes speech from the text, producing audible output that can be heard by the user.

This is crucial for enabling communication and learning for deaf students, as it bridges the gap between sign language and spoken language.

## **Overall Flow:**

**Input:** Real-time video is captured and broken down into individual frames.

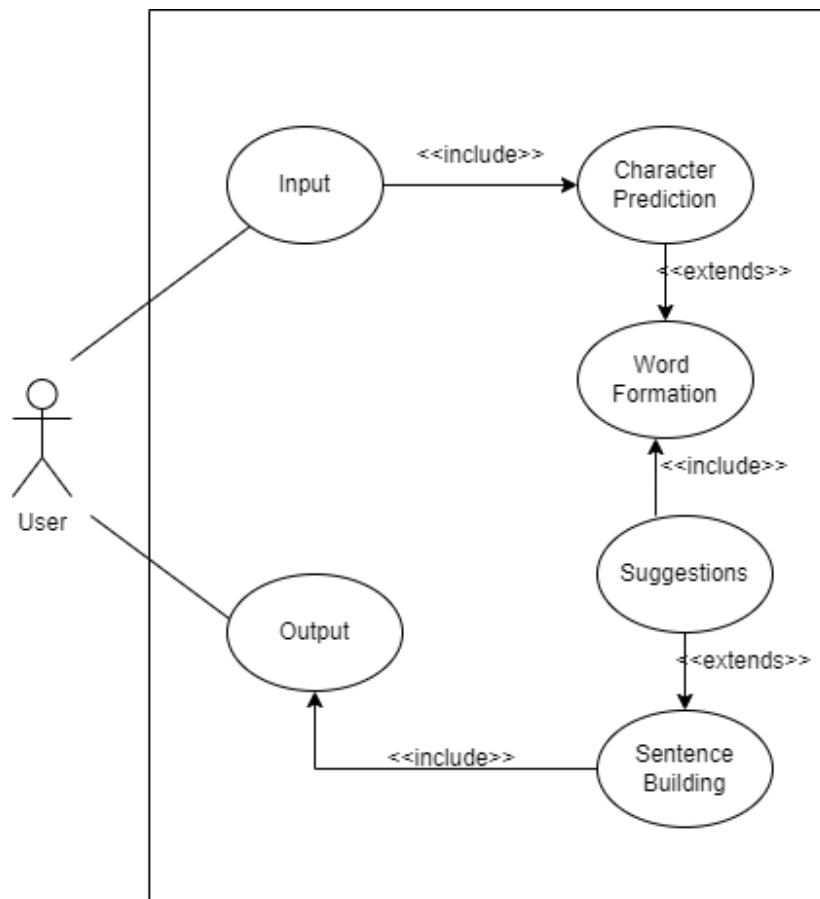
**Hand Landmark Extraction:** Each frame is processed to extract hand landmarks.

**Gesture Recognition:** The sequence of landmarks is fed into the LSTM + fully connected neural network to classify the gesture.

**Text Generation:** The classified gesture is converted into its corresponding text form.

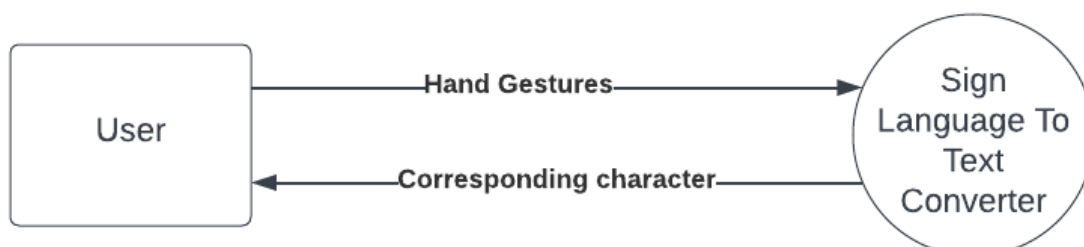
**Speech Output:** The text is transformed into speech using the TTS module, allowing the recognized gesture to be heard.

## 5.2 Data Flow Diagrams



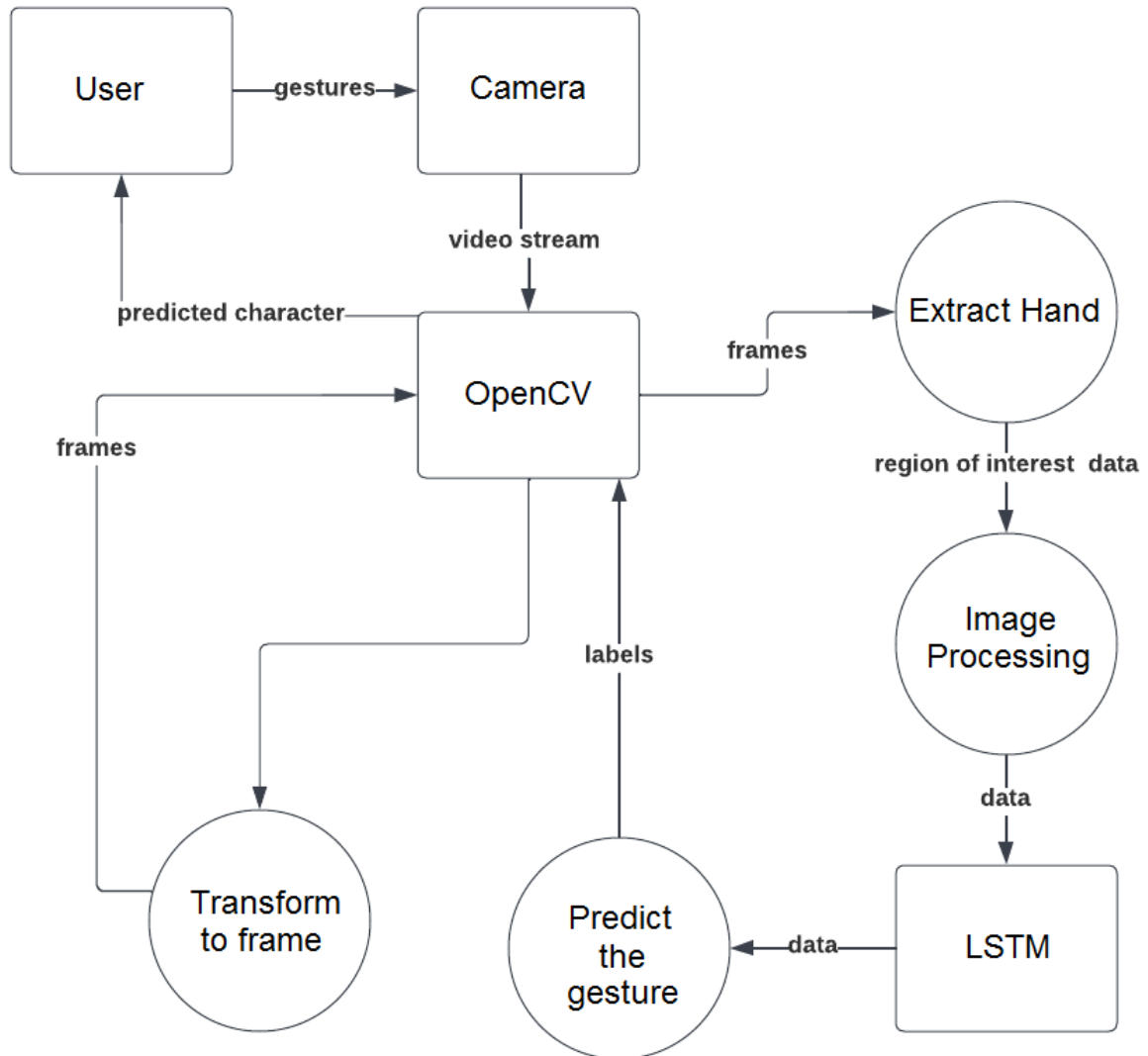
**Figure 5. Use case Diagram**

### DFD Level 1



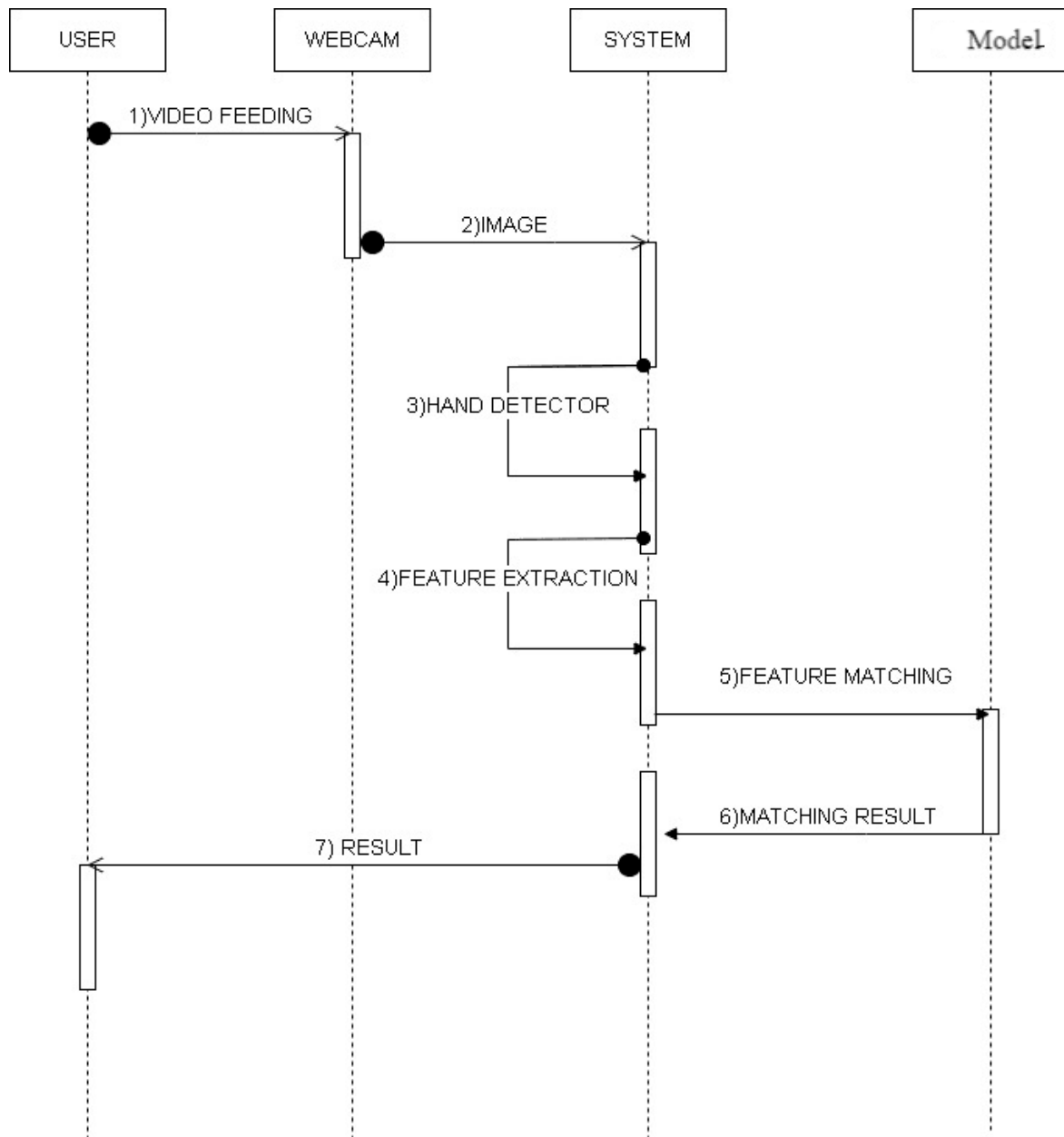
**Figure 6. DFD Level 1 Diagram**

## DFD Level 2



**Figure 7. DFD Level 2 Diagram**

The above DFD Level 2 diagram illustrates the detailed flow of Sign Language to Speech conversion system. It starts with the **User** performing gestures captured by the **Camera** as a video stream. This stream is processed by **OpenCV** library, which extracts frames and sends them for **Hand Extraction** and **Image Processing**. The processed data is fed into an **LSTM** network for gesture prediction. The **Predict the Gesture** module labels the data, which is then transformed back into frames and provided to the user as a **Predicted Character**. OpenCV also integrates these results, ensuring accurate real-time recognition and interaction.



**Figure 8. Sequence Diagram**

The sequence diagram outlines the process of converting sign language to speech. The **User** initiates the process by feeding video to the **Webcam**. The webcam captures the video and sends the **Image** data to the **System**. The system performs **Hand Detection** and proceeds with **Feature Extraction** from the detected hand. These features are sent to the **Model** for **Feature Matching**. Once the model matches the features, it sends the **Matching Result** back to the system, which then produces the final **Result** for the user. This sequence ensures accurate and efficient conversion of sign language gestures to speech.

### 5.3 Developing a User Interface

The next step is to develop a User Interface that helps us to interact with the model and fetch the results. UI can be developed using libraries like PyQt or Tkinter. The group developed the UI using Tkinter which is a cross-platform GUI toolkit, implemented as a Python plug-in.

## **IMPLEMENTATION**

The implementation phase of the project is the development of the designs produced during the design phase. It is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, and evaluation of change over methods. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. In a network backup system no additional resources are needed. Implementation is the final and the most important phase. The system can be implemented only after thorough testing is done and if it is found to be working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transactions while using the new system.

### **System Testing**

It is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies a test plan is carried out on each module. The various tests performed are unit testing, integration testing and user acceptance testing.

### **Unit Testing**

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by the system. The various controls are tested to ensure that each performs its action as required.

### **Integration Testing**

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the Server module and Client module options are integrated and tested. This testing provides the assurance that the application is a well-integrated functional unit with smooth transition of data.



## **User Acceptance Testing**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required. Apart from planning, major tasks of preparing the implementation are education and training of users. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective.

## **User Training**

After the system is implemented successfully, training of the user is one of the most important subtasks of the developer. For this purpose user manuals are prepared and handed over to the user to operate the developed system. Thus the users are trained to operate the developed system. Both the hardware and software securities are made to run the developed systems successfully in future. In order to put new application system into use, the following activities were taken care of:-

- a. Preparation of user and system documentation.
- b. Conducting user training with demo and hands on.
- c. Test run for some period to ensure smooth switching over the system

The users are trained to use the newly developed functions. User manuals describing the procedures for using the functions listed on the menu are circulated to all the users. It is confirmed that the system is implemented up to users' needs and expectations.

## **Security and Maintenance**

Maintenance involves the software industry captive, typing up system resources. It means restoring something to its original condition. Maintenance follows conversion to the extent that changes are necessary to maintain satisfactory operations relative to changes in the user's environment. Maintenance often includes minor enhancements or corrections to problems that surface in the system's operation. Maintenance is also done based on fixing the problems reported, changing the interface with other software or hardware, enhancing the software. Any system developed should be secured and protected against possible hazards. Security measures are provided to prevent unauthorized access of the database at various levels. Password protection and simple procedures to prevent unauthorized access are provided to the user.

## **Step by Step implementation of the app**

The entire project can be broken down into following modules:

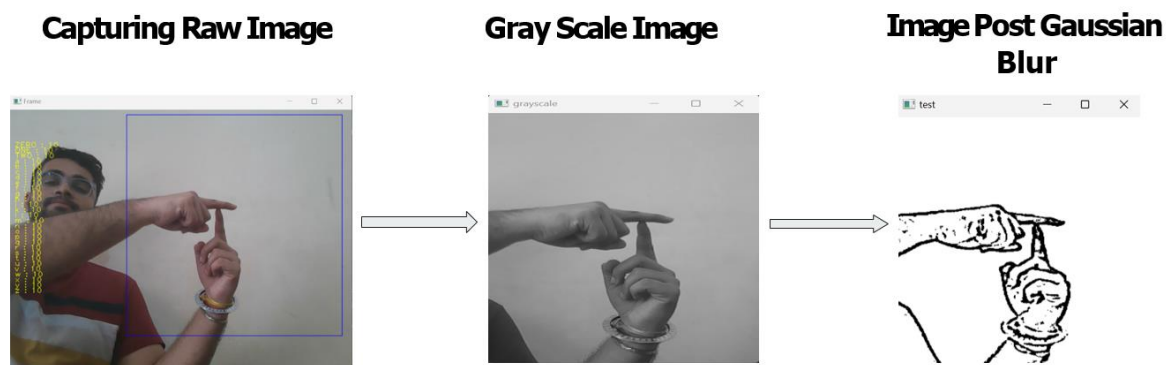
- Data Acquisition
- Data pre-processing and Feature extraction
- Gesture Classification
- Text To Speech Translation

## Data Acquisition

The process of acquiring data about hand gestures can be approached in mainly two ways – using sensory devices and vision based approach.

Electromechanical devices can be employed to provide precise hand configuration and position. Various glove-based approaches are used to extract information. However, these methods are often expensive and not user-friendly due to the complexity and discomfort of wearing such devices.

In vision-based methods, a computer webcam serves as the input device to observe and capture information about hands and/or fingers. This approach relies solely on a camera, allowing for a natural interaction between humans and computers without the need for additional devices, thereby reducing costs. The main challenge of vision-based hand detection is managing the significant variability in human hand appearance due to numerous hand movements, different skin colors, and variations in viewpoints, scales, and camera speed.



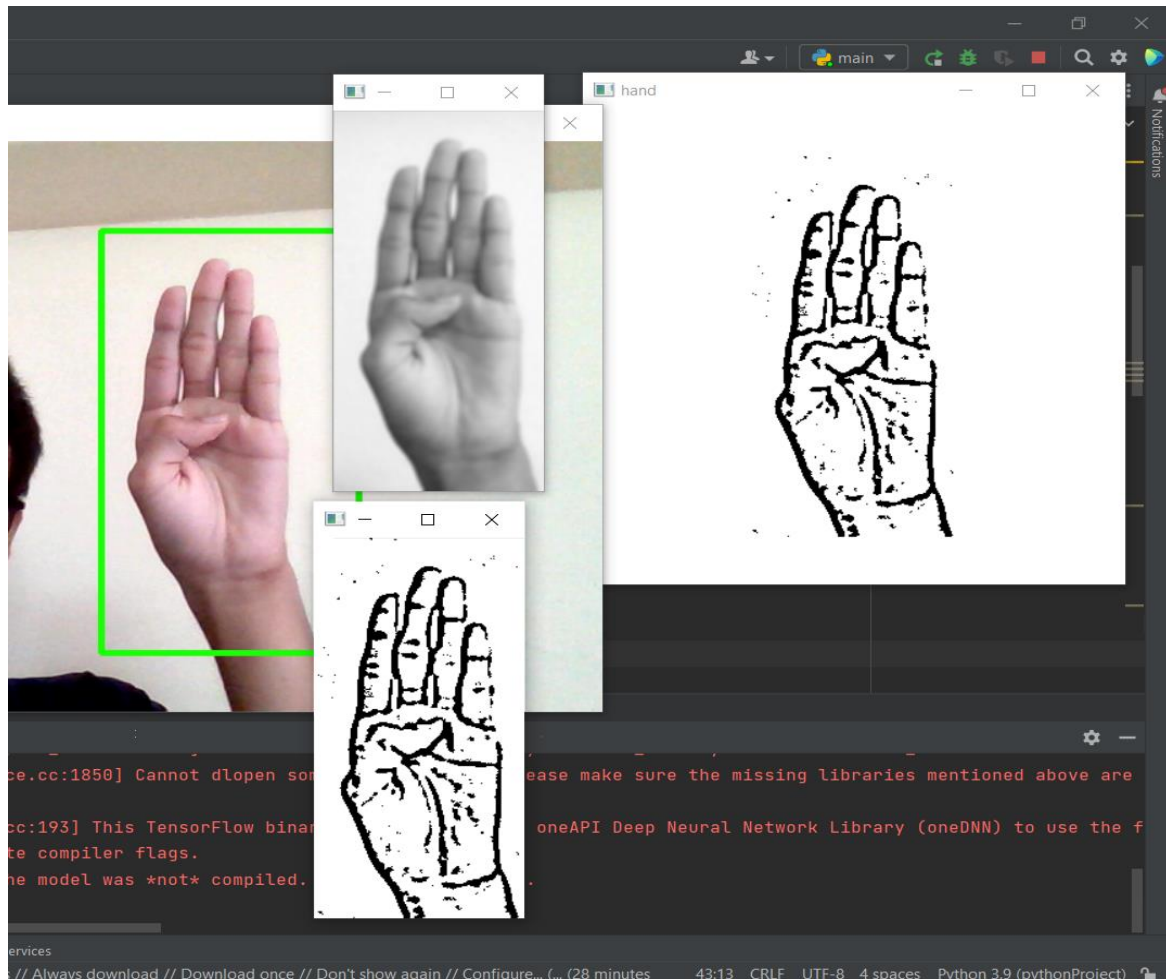
**Figure 9. Applying Gaussian Blur**

## Data Pre-processing and Feature Extraction

In this approach for hand detection, we start by capturing an image using a webcam. For detecting the hand within the image, we utilize the MediaPipe library, which is specifically designed for image processing. Once the hand is identified, we extract the region of interest (ROI) and crop the image accordingly. The cropped image is then converted to grayscale using the OpenCV library, followed by the application of a Gaussian blur filter to reduce noise. This filter can be easily applied using the OpenCV library. Subsequently, the grayscale image is converted to a binary image using threshold and adaptive threshold methods.

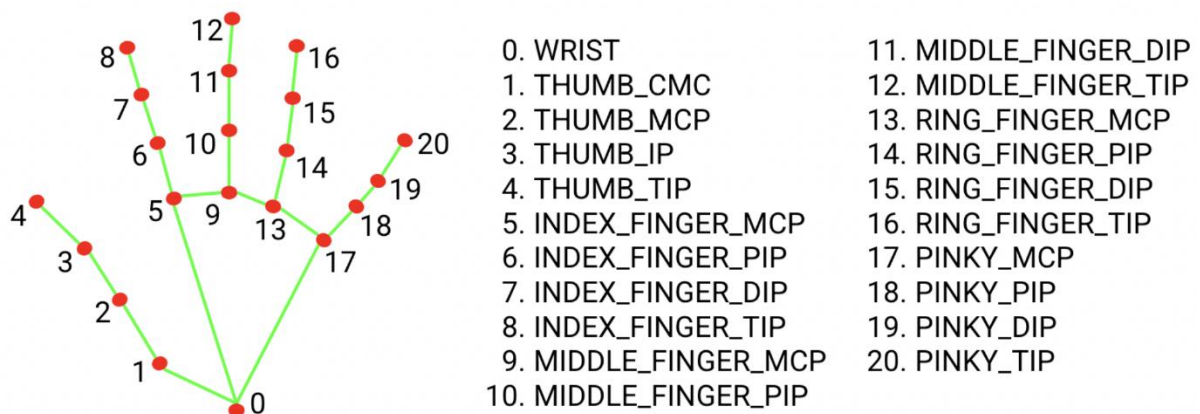
We have collected images of different signs representing Indian states and their capitals in Indian Sign Language (ISL). However, this method has several limitations: the hand must be positioned against a clean, soft background and in proper lighting conditions for accurate results. In real-world scenarios, such ideal backgrounds and lighting conditions are not always available.

In this approach, firstly we detect hand from image that is acquired by webcam and for detecting a hand we used media pipe library which is used for image processing. So, after finding the hand from image we get the region of interest (Roi) then we cropped that image and convert the image to gray image using OpenCV library after we applied the gaussian blur. The filter can be easily applied using open computer vision library also known as OpenCV. Then we converted the gray image to binary image using threshold and Adaptive threshold methods.



**Figure 10. Data Preprocessing**

To address these challenges, we explored different approaches and arrived at an interesting solution. We first detect the hand in the frame using MediaPipe and extract the hand landmarks present in the image. These landmarks are then stored in a numpy array. This method significantly improves the robustness of our system against varying background and lighting conditions, ensuring more accurate and reliable hand gesture recognition.



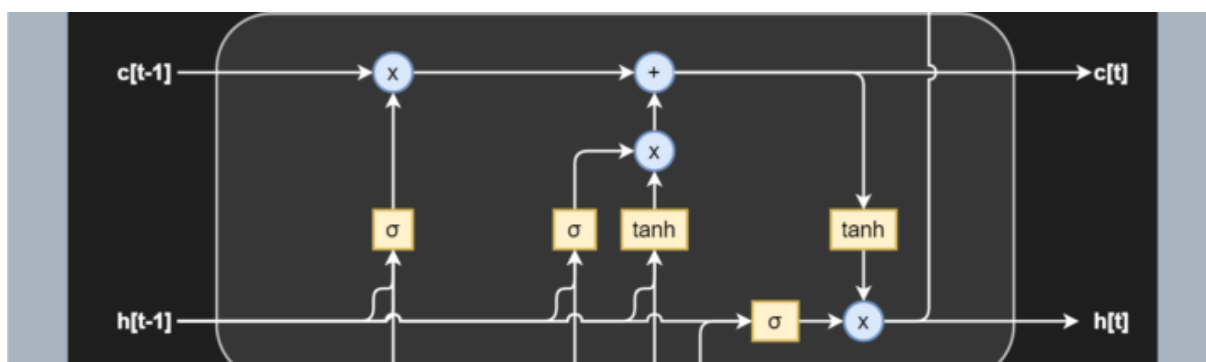
**Figure 11. Mediapipe Landmark System**

By doing this we tackle the situation of background and lightning conditions because the mediapipe library will give us landmark points in any background and mostly in any lightning conditions.

### **Gesture Classification:**

#### **Long Short Term Memory (LSTM) Networks**

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are well-suited for learning from sequences of data. They are designed to overcome the limitations of traditional RNNs, which struggle with long-term dependencies due to the problem of vanishing gradients. LSTMs achieve this by incorporating memory cells that can maintain information over long periods and gates that control the flow of information into and out of these cells.



**Figure 12. LSTM Basic Structure**

#### **Components of an LSTM Network**

1. **Cell**—Every unit of the LSTM network is known as a “cell”. Each cell is composed of 3 inputs—

$x(t)$ —token at timestamp  $t$

$h(t-1)$ —previous hidden state

$c(t-1)$ —previous cell state,

and 2 outputs—

$h(t)$ —updated hidden state, used for predicting the output

$c(t)$ —current cell state

**2. Gates**—LSTM uses a special theory of controlling the memorizing process. Popularly referred to as gating mechanism in LSTM, what the gates in LSTM do is, store the memory components in analog format, and make it a probabilistic score by doing point-wise multiplication using sigmoid activation function, which stores it in the range of 0–1. Gates in LSTM regulate the flow of information in and out of the LSTM cells.

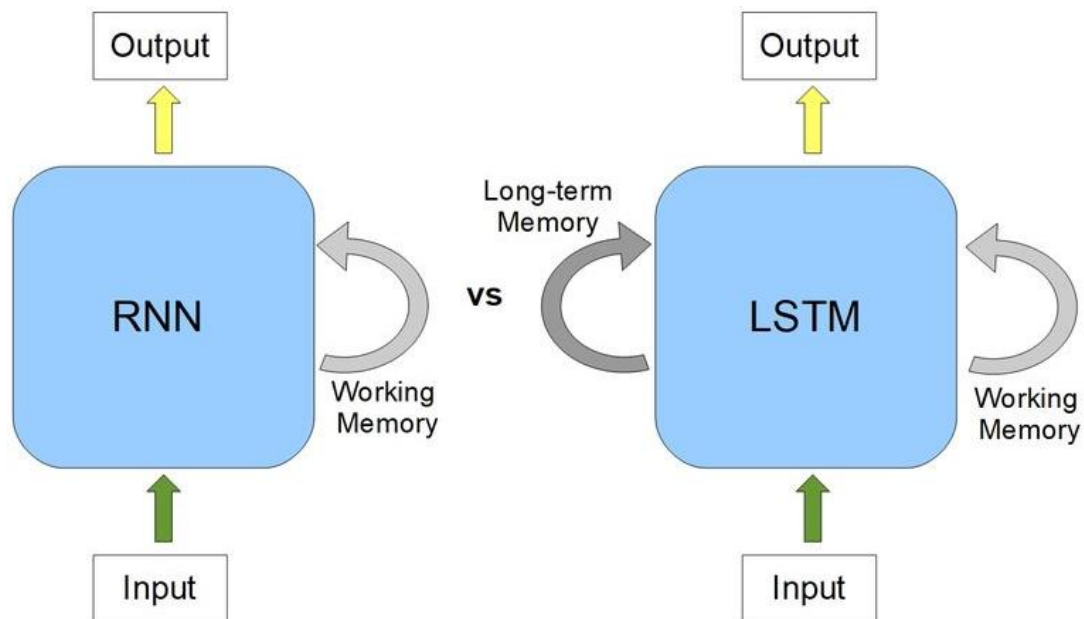
Gates are of 3 types—

- **Input Gate**—This gate lets in optional information necessary from the current cell state. It decides which information is relevant for the current input and allows it in.
- **Output Gate**—This gate updates and finalizes the next hidden state. Since the hidden state contains critical information about previous cell inputs, it decides for the last time which information it should carry for providing the output.
- **Forget Gate**—Pretty smart in eliminating unnecessary information, the forget gate multiplies 0 to the tokens which are not important or relevant and lets it be forgotten forever.

### How LSTM Works?

1. **Forget Gate:** It uses a sigmoid function to filter out irrelevant information from the previous cell state.
2. **Input Gate:** It updates the cell state with the new information that is considered relevant.
3. **Cell State Update:** The cell state is updated by adding the filtered previous state and the new candidate values.
4. **Output Gate:** It generates the output by filtering the updated cell state through a sigmoid function and then applying a tanh function.

## Why does LSTM outperform RNN?



**Figure 13. RNN vs. LSTM**

RNNs have quite massively proved their incredible performance in sequence learning. But, it has been remarkably noticed that RNNs are not sporty while handling long-term dependencies.

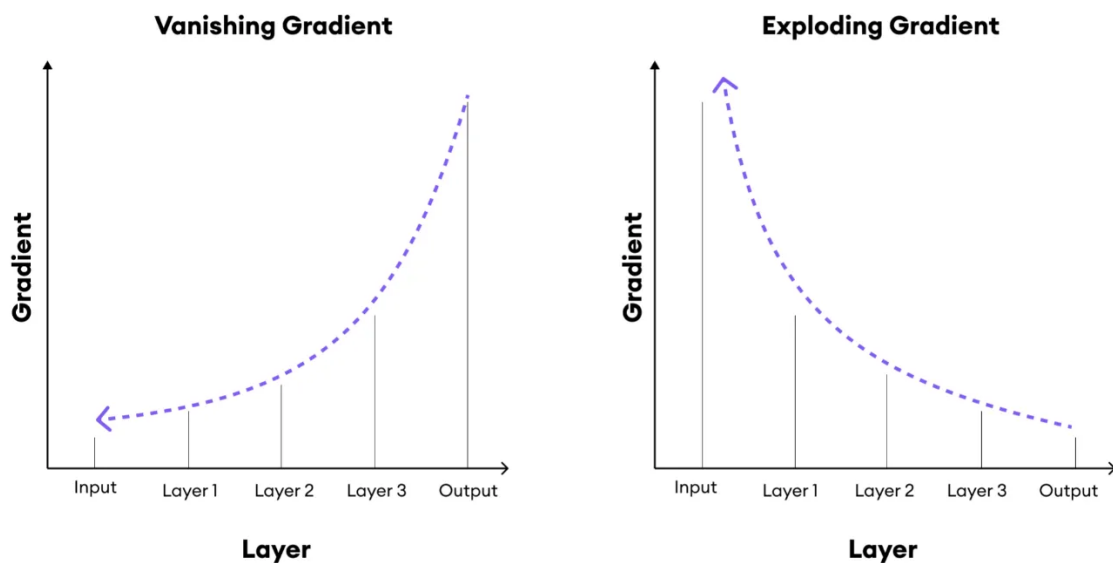
Why so?

### **Vanishing Gradient Problem**

Recurrent Neural Networks uses a hyperbolic tangent function, what we call the tanh function. The range of this activation function lies between  $[-1,1]$ , with its derivative ranging from  $[0,1]$ . Now we know that RNNs are a deep sequential neural network. Hence, due to its depth, the matrix multiplications continually increase in the network as the input sequence keeps on increasing. Hence, while we use the chain rule of differentiation during calculating backpropagation, the network keeps on multiplying the numbers with small numbers. And guess what happens when you keep on multiplying a number with negative values with itself? It becomes exponentially smaller, squeezing the final gradient to almost 0, hence weights are no more updated, and model training halts. It leads to poor learning, which we say as “cannot handle long term dependencies” when we speak about RNNs.

### **Exploding Gradient Problem**

Similar concept to the vanishing gradient problem, but just the opposite of the process, let's suppose in this case our gradient value is greater than 1 and multiplying a large number to itself makes it exponentially larger leading to the explosion of the gradient.



**Figure 14. Gradient Problems in RNN**

**The detailed workflow of the approach which we used for this project is:**

1. **Data Capture:** Using a webcam, video frames are captured and processed to detect hand landmarks using MediaPipe.
2. **Pre-processing:** The landmarks are stored in a numpy array, representing the sequential data input for the LSTM network.
3. **Training:** The LSTM network is trained using labeled sequences of hand landmarks, each corresponding to a specific ISL gesture.
4. **Prediction:** In real-time, the LSTM network predicts the gesture based on the sequence of incoming hand landmarks.
5. **Output Generation:** The predicted gesture is converted into text and then into speech, providing an audible output.

We trained our LSTM network model on a total of **62 signs** including the signs for the Indian states and their capitals in Indian Sign Language (ISL).

### **Advantages of Using LSTM in our Project**

- **Handling Long Sequences:** LSTMs can manage long sequences of hand movements without losing context, making them ideal for sign language recognition.
- **Memory Efficiency:** LSTMs can retain and forget information as needed, making the model efficient in handling the dynamics of hand gestures.
- **Improved Accuracy:** The ability of LSTMs to understand temporal dependencies improves the accuracy of gesture recognition.

By leveraging the capabilities of LSTM networks, our project aims to create an efficient and accurate system for converting Indian Sign Language gestures into speech, thereby facilitating communication and education for deaf students.

### **Text To Speech Translation**

The model translates known gestures into words. We have used **pyttsx3 library** of Python to convert the recognized words into the appropriate speech. The text-to-speech output is a simple workaround, but it's a useful feature because it simulates a real-life dialogue.



## **TESTING**

Testing is a crucial phase in the development of this project. It ensures that the system meets its requirements and functions correctly in various conditions. The primary goal is to identify and rectify defects, validate functionality, and ensure the system's reliability and performance. The testing phase involves various techniques and strategies to cover different aspects of the system.

### **Types of Testing**

#### **7.1 Functional Testing**

Functional testing focuses on verifying that the system behaves according to the specified requirements. It involves testing the system's features and functionalities to ensure they work as intended.

##### **Black Box Testing**

Black box testing involves testing the system without knowledge of the internal workings of the application. The tester only knows the inputs and the expected outputs. In this project, black box testing involved testing the user interface, the gesture detection accuracy, and the speech synthesis output. Test cases were designed based on the functional requirements to validate the system's behavior.

Various Functional Tests that were conducted are as follows:

- Test the system's ability to detect and interpret different hand gestures accurately.
- Verify the correctness of the text output generated from recognized gestures.
- Ensure the text-to-speech (TTS) module produces clear and accurate speech.

#### **7.2 Structural Testing**

Structural testing, also known as white box testing, involves testing the internal structures or workings of an application, as opposed to its functionality.

##### **White Box Testing**

White box testing involves testing the internal logic, code structure, and design of the application. The tester has full visibility of the internal processes and logic. For this project, white box testing included unit testing, integration testing, and code review.

Various Structural Tests that were conducted are as follows:

- Unit Testing: Testing individual components such as the hand landmark extraction module, the LSTM network, and the TTS module to ensure each unit functions correctly in isolation.

- **Integration Testing:** Testing the interaction between different modules (e.g., verifying that the output from the hand detection module correctly feeds into the gesture recognition module).
- **Code Review:** Reviewing the code for logic errors, coding standards, and potential security vulnerabilities.

### 7.3 Non-Functional Testing

Non-functional testing evaluates the system's performance, reliability, and other quality attributes.

**Performance Testing:** Performance testing assesses how well the system performs under various conditions, including load and stress. For this project, performance testing involved testing the real-time processing capability, latency in gesture recognition, and the responsiveness of the TTS module.

**Usability Testing:** Usability testing ensures that the system is user-friendly and meets the needs of the end-users. For this project, usability testing will involve testing the user interface with real users, particularly deaf students, to gather feedback on the system's ease of use and overall experience.

### 7.4 Security Testing

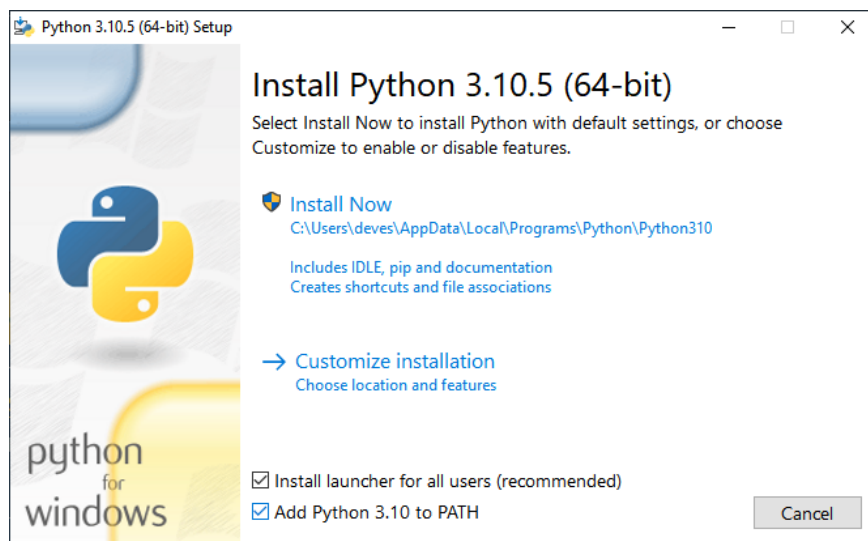
Security testing ensures that the system protects user data and is resilient against potential threats. For this project, security testing involved verifying data protection mechanisms, ensuring secure storage of gesture data, and testing for vulnerabilities.

## **RESULTS AND DISCUSSION**

### **8.1 Installation process:**

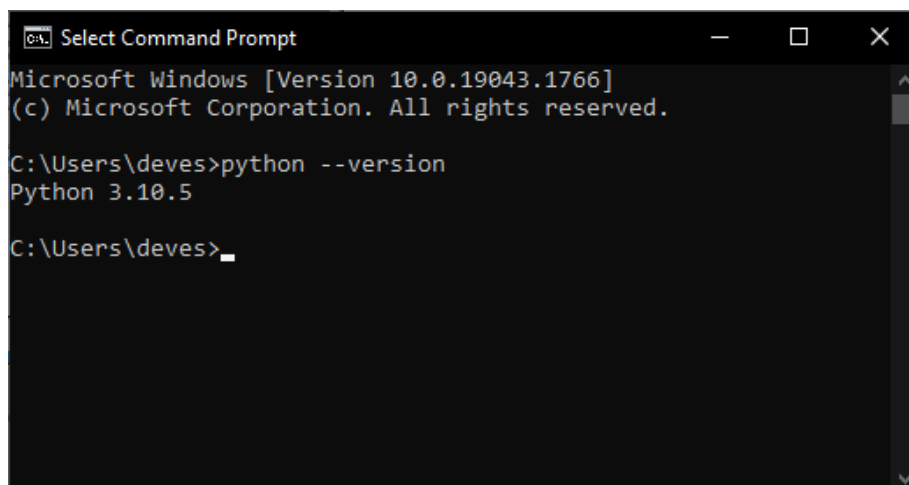
#### **Installing Python**

1. Go to the python website and download the latest version.  
<https://www.python.org/downloads/>
2. Run the Installer



**Figure 15. Python Installer Window**

3. Check python installed using command prompt.

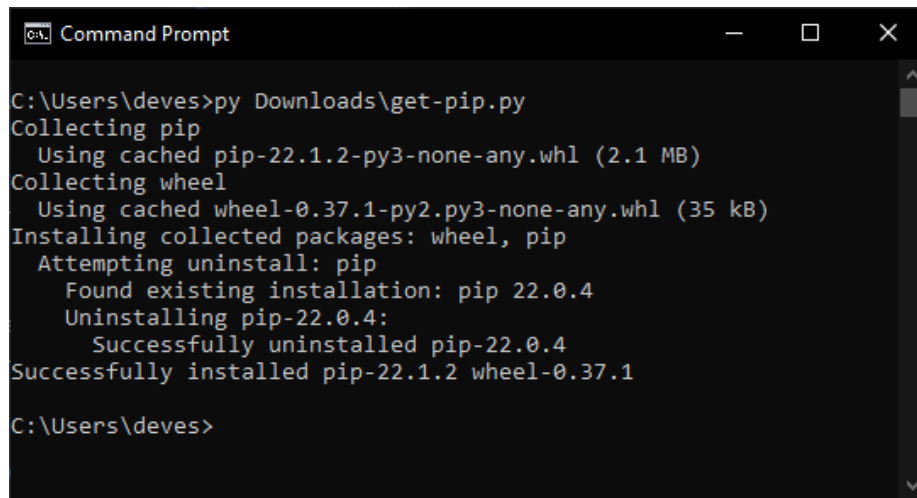


**Figure 16. Python version check**

#### **Installing pip**

<https://pip.pypa.io/en/stable/installation/>

1. Download the script from the following link:  
<https://bootstrap.pypa.io/get-pip.py>
2. Run the downloaded script from the download location



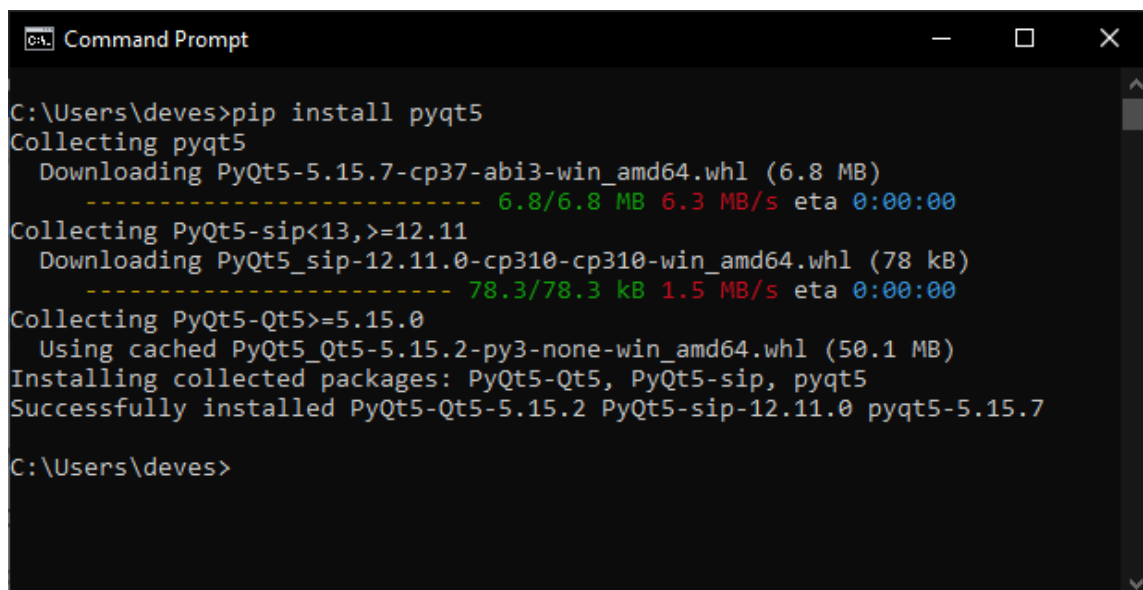
```
C:\Users\deves>py Downloads\get-pip.py
Collecting pip
  Using cached pip-22.1.2-py3-none-any.whl (2.1 MB)
Collecting wheel
  Using cached wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.4
    Uninstalling pip-22.0.4:
      Successfully uninstalled pip-22.0.4
Successfully installed pip-22.1.2 wheel-0.37.1

C:\Users\deves>
```

**Figure 17. Installing pip**

## Installing python libraries and modules

1. Installing pyqt5  
Command: > pip install pyqt5



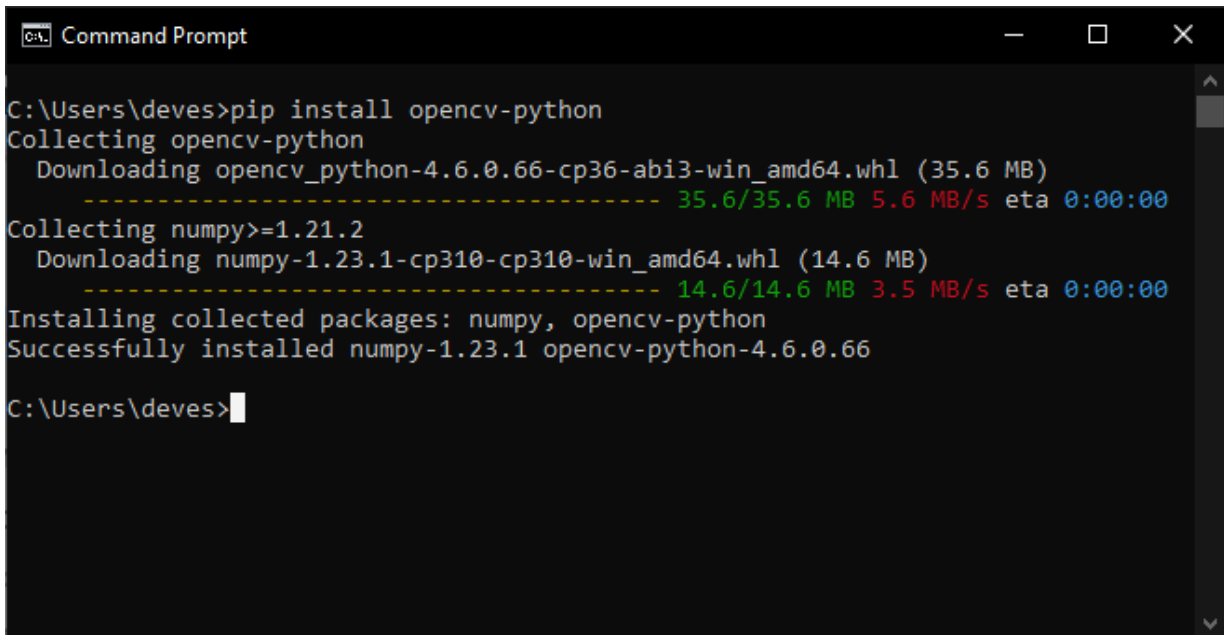
```
C:\Users\deves>pip install pyqt5
Collecting pyqt5
  Downloading PyQt5-5.15.7-cp37-abi3-win_amd64.whl (6.8 MB)
----- 6.8/6.8 MB 6.3 MB/s eta 0:00:00
Collecting PyQt5-sip<13,>=12.11
  Downloading PyQt5_sip-12.11.0-cp310-cp310-win_amd64.whl (78 kB)
----- 78.3/78.3 kB 1.5 MB/s eta 0:00:00
Collecting PyQt5-Qt5>=5.15.0
  Using cached PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
Installing collected packages: PyQt5-Qt5, PyQt5-sip, pyqt5
Successfully installed PyQt5-Qt5-5.15.2 PyQt5-sip-12.11.0 pyqt5-5.15.7

C:\Users\deves>
```

**Figure 18. Installing pyqt**

## 2. Installing opencv-python

Command: > pip install opencv-python



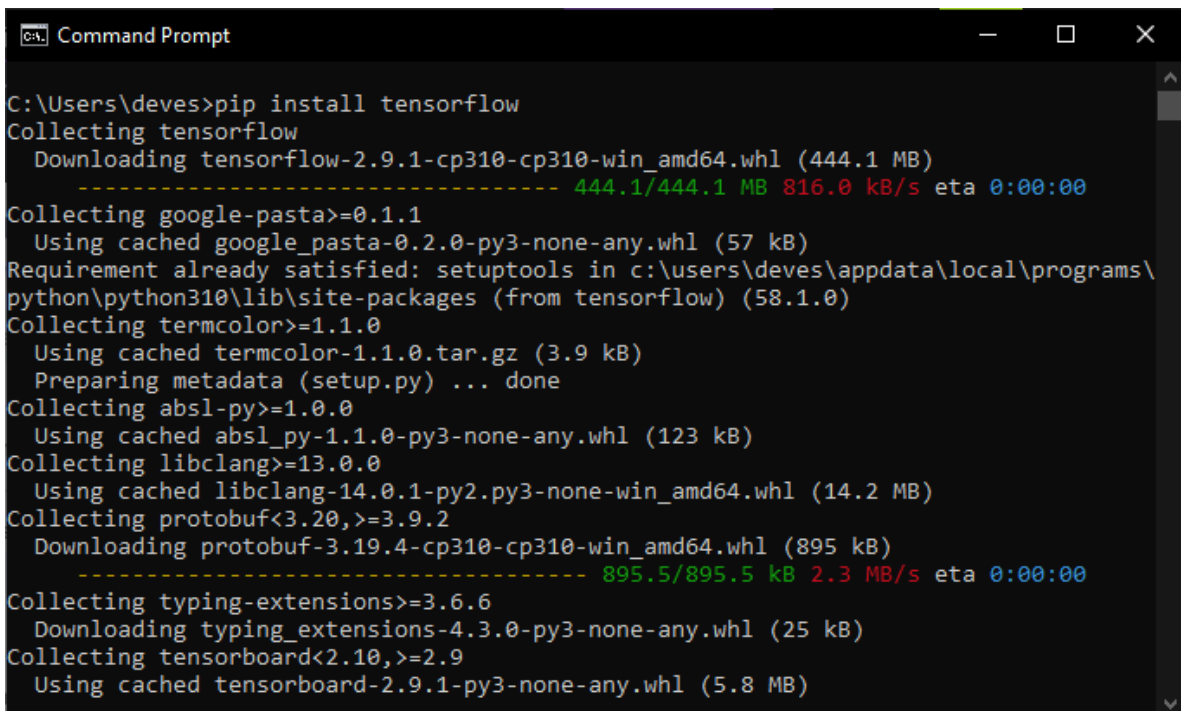
```
C:\Users\deves>pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
    ----- 35.6/35.6 MB 5.6 MB/s eta 0:00:00
Collecting numpy>=1.21.2
  Downloading numpy-1.23.1-cp310-cp310-win_amd64.whl (14.6 MB)
    ----- 14.6/14.6 MB 3.5 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.23.1 opencv-python-4.6.0.66

C:\Users\deves>
```

**Figure 19. Installing opencv-python**

## 3. Installing TensorFlow

Command: > pip install tensorflow



```
C:\Users\deves>pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.9.1-cp310-cp310-win_amd64.whl (444.1 MB)
    ----- 444.1/444.1 MB 816.0 kB/s eta 0:00:00
Collecting google-pasta>=0.1.1
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: setuptools in c:\users\deves\appdata\local\programs\python\python310\lib\site-packages (from tensorflow) (58.1.0)
Collecting termcolor>=1.1.0
  Using cached termcolor-1.1.0.tar.gz (3.9 kB)
  Preparing metadata (setup.py) ... done
Collecting absl-py>=1.0.0
  Using cached absl_py-1.1.0-py3-none-any.whl (123 kB)
Collecting libclang>=13.0.0
  Using cached libclang-14.0.1-py2.py3-none-win_amd64.whl (14.2 MB)
Collecting protobuf<3.20,>=3.9.2
  Downloading protobuf-3.19.4-cp310-cp310-win_amd64.whl (895 kB)
    ----- 895.5/895.5 kB 2.3 MB/s eta 0:00:00
Collecting typing-extensions>=3.6.6
  Downloading typing_extensions-4.3.0-py3-none-any.whl (25 kB)
Collecting tensorboard<2.10,>=2.9
  Using cached tensorboard-2.9.1-py3-none-any.whl (5.8 MB)
```

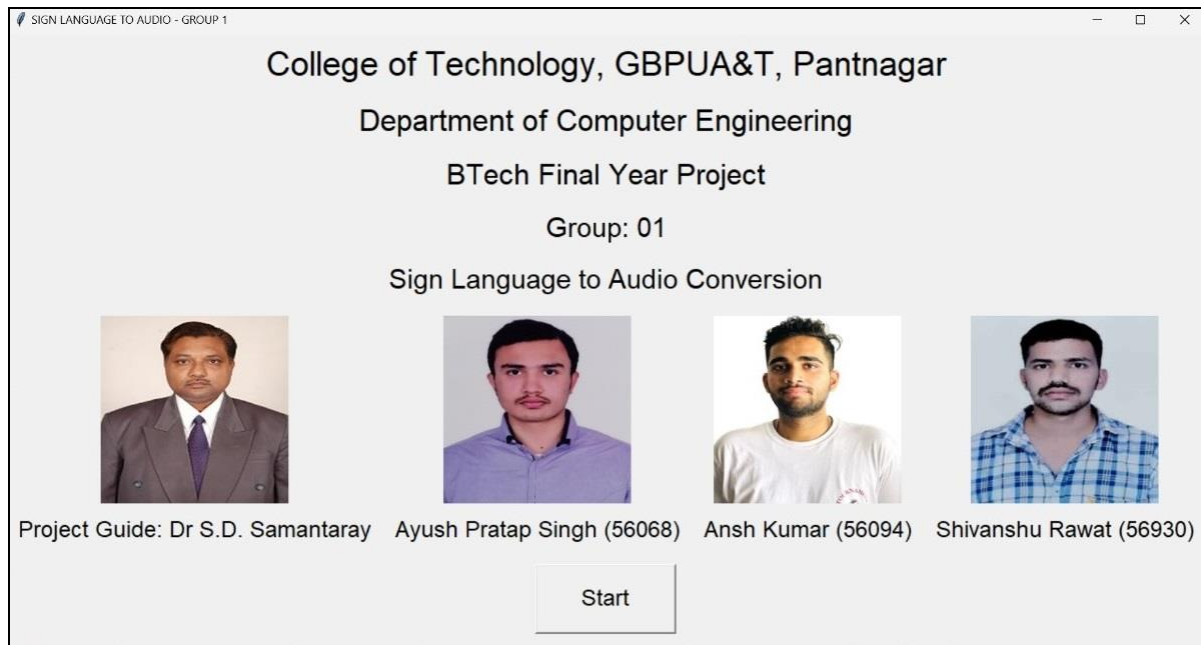
**Figure 20. Installing tensorflow**

## Run the project

Command: > python -u "d:\FinalYearProject\main\_app.py"

## 8.2 Project Screenshots and Use Case Example

We have created the following user interface for the project displaying the photos and names of the project guide and the members of our project group. We have given the “Start” button at the bottom, clicking upon which the actual validation window opens as can be seen in below demonstration screenshots.



**Figure 21. Graphical User Interface**

Below is the screenshot of the Validation Window which has following components:

### **1. Output Window**

This window shows the real-time video captured by the webcam or camera.

### **2. Prediction**

This is the word or set of words that the LSTM model predicts after analyzing the real time sign obtained from the webcam.

### **3. Train Button**

Training of the LSTM model will start on the collected dataset upon clicking this button.

### **4. Validate Button**

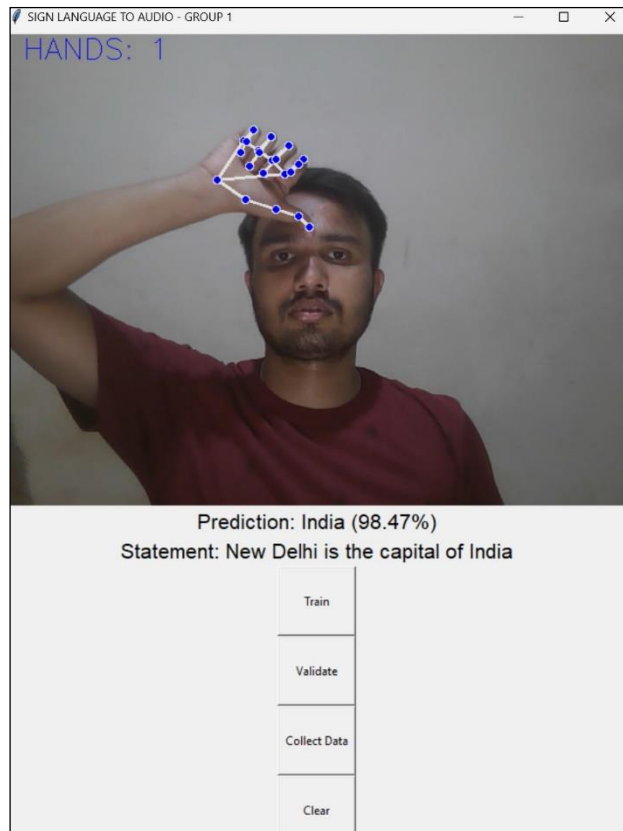
Actual validation or inference will start upon clicking this button.

### **5. Collect Data Button**

Data collection will start upon clicking this button.

### **6. Clear Button**

Whole statement will get cleared upon clicking this button.

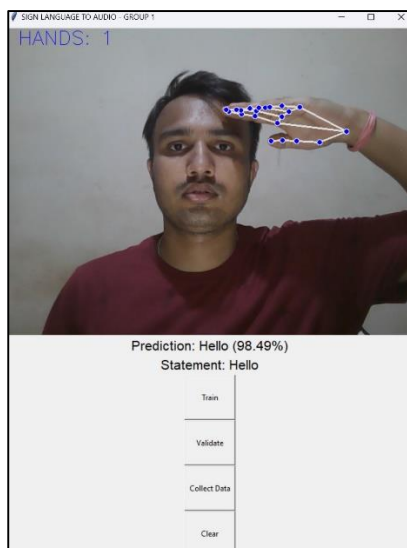


**Figure 22. Validation Window**

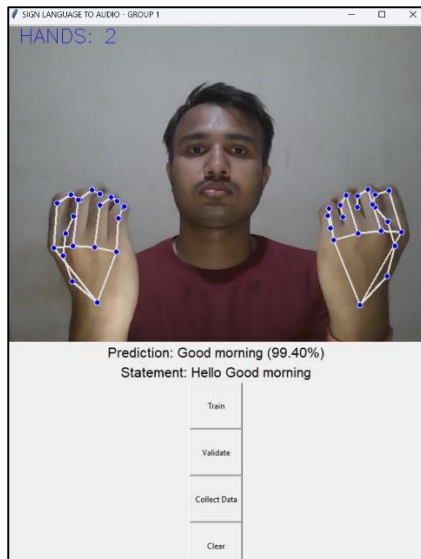
### Use Case Example

Imagine a classroom where a passionate teacher, who is deaf and dumb and uses sign language, stands in front of a diverse group of students. Equipped with our sign language to speech conversion system, the teacher begins the lesson confidently. The teacher has to teach students about Indian states and their capitals, for this, he has prepared the following script:

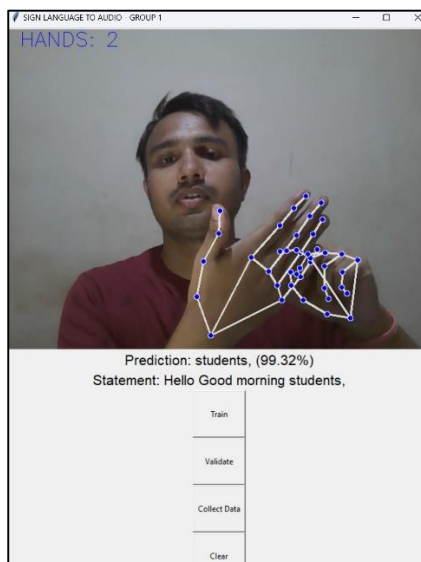
“Hello! Good morning students, today I am going to teach you about Indian states and their capitals. New Delhi is the capital of India. ....”



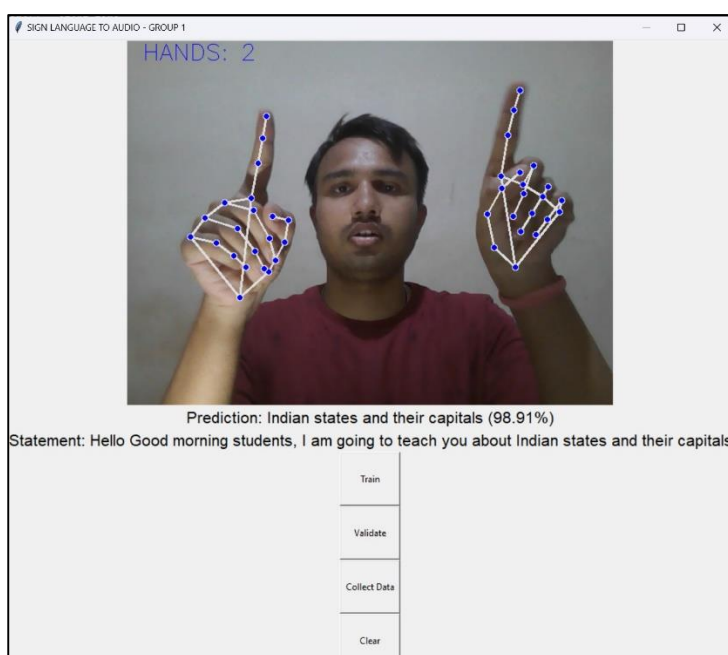
"Hello!" The teacher signs with a warm smile, and immediately, the system converts the gesture into text form with a accuracy of 98.49%.



ISL sign for “Good morning” is detected with 99.40% accuracy and it is appended to the Statement.



ISL sign for “students” is detected with 99.32% accuracy and it is appended to the Statement.

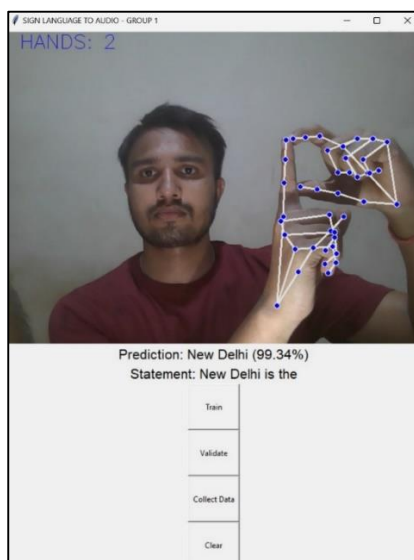


ISL sign for “Indian states and their capitals” is detected with 98.91% accuracy and it is appended to the Statement.

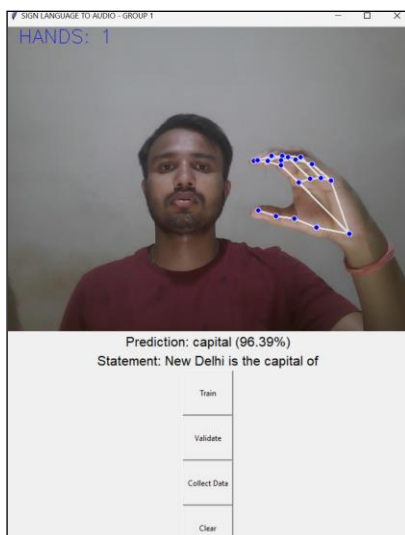




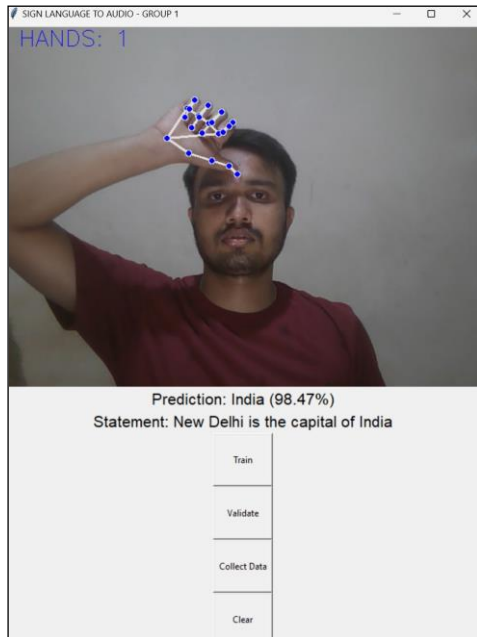
ISL sign for “OK” is detected with 98.70% accuracy. After detecting “OK”, the system calls the TTS (Text-to-speech) module and converts the formed statement into audible speech.



ISL sign for “New Delhi” is detected with 99.34% accuracy and it is appended to the Statement.



ISL sign for “capital” is detected with 96.39% accuracy and it is appended to the Statement.



ISL sign for “India” is detected with 98.47% accuracy and it is appended to the Statement.



ISL sign for “OK” is detected with 98.67% accuracy. After detecting “OK”, the system calls the TTS (Text-to-speech) module and converts the formed statement into audible speech.



**Figure 23. Real Time Validation**

This innovative approach not only empowers the teacher to deliver the lesson effectively but also fosters an inclusive learning environment where students can engage with diverse teaching methods, breaking down barriers and enriching the educational experience for all.

## 8.3 Opportunities and Challenges of Converting Sign Languages to Speech

Deep learning, a subfield of artificial intelligence, offers immense potential for revolutionizing sign language to speech conversion. Its ability to learn complex patterns from vast amounts of data makes it well-suited for the task of recognizing and translating sign language gestures. This section explores the exciting opportunities and significant challenges that we faced while applying deep learning to this domain.

### Opportunities

1. **High Accuracy and Robustness:** Deep learning architectures, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), excel at extracting features from video frames and recognizing patterns in sequential data like hand movements. This capability allows them to achieve high accuracy in sign language recognition, leading to more reliable and effective speech conversion.
2. **Adaptability and Generalization:** Deep learning models possess the ability to learn from diverse datasets encompassing variations in signing styles, hand postures, and backgrounds. This adaptability allows them to generalize well and function accurately even when encountering unseen signs or signer variations.
3. **Real-Time Processing Potential:** Advancements in deep learning architectures and hardware optimization are paving the way for real-time sign language conversion. This enables seamless communication with minimal latency, crucial for natural conversations and fostering inclusivity.

### Challenges

1. **Limited Training Data:** The success of deep learning models heavily relies on the quality and size of training datasets. Creating comprehensive datasets encompassing a vast vocabulary of signs, signer variations, and real-world scenarios remains a challenge for us.
2. **Signer Independence:** Signing styles can vary significantly across individuals and regions. Deep learning models need to be robust enough to overcome these variations and achieve signer-independent recognition for widespread usability.
3. **Computational Demands:** Training and running complex deep learning models can require significant computational resources. Optimizing these models for efficient processing on mobile devices or embedded systems is crucial for real-world implementation.
4. **Environmental Variations:** Lighting conditions, background noise, and occlusions (when parts of the hands are hidden) can significantly impact the accuracy of sign recognition. Deep learning models need to be adaptable enough to function effectively in various environments.

## **SUMMARY**

This project aims to develop a system that accurately translates hand gestures into spoken language, specifically focusing on Indian Sign Language (ISL). The primary goal is to facilitate effective communication and enhance the educational experience for deaf students, particularly in learning about Indian states and their capitals. The system leverages advanced technologies such as computer vision, machine learning, and text-to-speech (TTS) to achieve real-time translation of hand gestures into audible speech.

The core functionality of the project includes detecting hand gestures through a webcam using the Mediapipe handtracking module, preprocessing the captured data to reduce noise, and recognizing gestures using Long Short-Term Memory (LSTM) networks. This ensures high accuracy in interpreting the dynamic sequences of gestures. The system converts the recognized gestures into text and subsequently into speech using TTS technology, providing clear and contextually accurate audio output. Additionally, the project emphasizes real-time processing to offer immediate feedback and translation, and includes a user-friendly interface accessible to deaf students, displaying real-time feedback and providing control options for the recognition process.

The project requires specific hardware and software components. A high-resolution webcam and a computer with adequate processing power are essential for real-time video processing and gesture recognition. Key software includes Mediapipe for handtracking, OpenCV for image processing, and TensorFlow for developing and training the LSTM network. TTS software such as Pyttsx3 is used for converting text into speech. The development environment involves Python IDE like PyCharm or Visual Studio Code, with additional tools for version control and testing.




Effective project management involved establishing a detailed timeline with milestones, assigning specific roles and responsibilities to team members, and maintaining comprehensive documentation throughout the development phases. Testing and validation are critical, with a structured plan to ensure all components function correctly. User testing with deaf students provided us valuable feedback for validating the system's effectiveness in an educational context, allowing for necessary adjustments based on the user experiences.

In summary, this project integrates cutting-edge technology to create a practical, reliable, and user-friendly tool that significantly aids deaf students in their educational journey. By addressing both functional and non-functional requirements, the project ensures a high standard of performance, accuracy, and usability, ultimately enhancing the learning experience for its users.

## **LITERATURE CITED**

- [1] T. Yang, Y. Xu, and “A. , Hidden Markov Model for Gesture Recognition”, CMU-RI-TR-94 10, Robotics Institute, CarnegieMellon Univ.,Pittsburgh,PA, May 1994.
- [2] Pujan Ziaie, Thomas Müller , Mary Ellen Foster , and Alois Knoll“A Naïve Bayes Munich,Dept. of Informatics VI, Robotics and Embedded Systems,Boltzmannstr. 3, DE-85748 Garching, Germany.
- [3][https://docs.opencv.org/2.4/doc/tutorials/imgproc/gaussian\\_median\\_blur\\_bilateral\\_filter/gaussian\\_median\\_blur\\_bilateral\\_filter.html](https://docs.opencv.org/2.4/doc/tutorials/imgproc/gaussian_median_blur_bilateral_filter/gaussian_median_blur_bilateral_filter.html)
- [4] Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.
- [5] [aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/](https://github.com/aeshpande3/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/)
- [6] <http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php>
- [7] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925.
- [8] Zaki, M.M., Shaheen, S.I.: Sign language recognition using a combination of new vision based features. Pattern Recognition Letters 32(4), 572–577 (2011)
- [9] N. Mukai, N. Harada and Y. Chang, "Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning," 2017 Nicograph International (NicoInt), Kyoto, Japan, 2017, pp. 19-24.doi:10.1109/NICOInt.2017.9
- [10] Byeongkeun Kang , Subarna Tripathi , Truong Q. Nguyen ”Real-time sign language fingerspelling recognition using convolutional neural networks from depth map” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)
- [11] <https://opencv.org/>
- [12] <https://en.wikipedia.org/wiki/TensorFlow>
- [13] [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)

**BIO-DATA OF STUDENTS**

|   |  |
|---|--|
|    | <p>           Name : Ayush Pratap Singh<br/>           ID No. : 56068<br/>           Place : Rudrapur         </p> |
|   | <p>           Name : Ansh Kumar<br/>           ID No. : 56094<br/>           Place : Khatima         </p>          |
|  | <p>           Name : Shivanshu Rawat<br/>           ID No. : 56930<br/>           Place : Kashipur         </p>    |

## **APPENDIX I**

### Installation of the Project

The installation process for the sign language to speech conversion system will vary depending on the specific software and hardware involved. However, this guide provides a general framework that can be adapted to most setups.

First, ensure your computer meets the minimum system requirements, such as operating system, processor speed, RAM, and available disk space. Additionally, if the system relies on computer vision for sign recognition, you'll need a webcam with a minimum resolution.

Next, proceed with the software installation. Locate the official download website or repository for the software components and any necessary libraries or language models. Run the installer and follow the prompts, accepting license agreements and choosing an appropriate installation directory. During installation, you might encounter options to customize settings like the spoken output language or advanced user configurations.

Once the software is installed, connect your webcam (if applicable) and ensure it's properly recognized by the system. In some cases, you might need to connect external sensors like data gloves via USB or Bluetooth, potentially requiring additional driver installations.

Finally, complete the post-installation configuration. The software might need to load pre-trained deep learning models for sign language recognition. These models can be large files and take time to download. Language pack installation might also be required for specific spoken output languages. After configuration, thoroughly test the system by performing basic signs and verifying accurate translation into spoken words.

Remember to consult the software's user manuals or online documentation for specific installation instructions and troubleshooting tips. By following these steps and referring to the relevant resources, you can ensure a smooth installation of your sign language to speech conversion system, promoting greater communication accessibility.

## **APPENDIX II**

### **Source Code**

```
import cv2

import numpy as np

import os

import pickle as pkl

import time

import tkinter as tk

import json

import pytsx3

from tkinter import Label, Button, simpledialog

from PIL import Image, ImageTk


from utils.mediapipe_helper import MediapipeHelper

from model.model import load_model_, get_model, logger

from data_loader import load_data

from get_logger import *

from utils.helper import shift_top_inplace


DATASET_DIR = "./dataset"

FONT_SIZE = 1

FRAMES = int(1.5 * 24)

FEATURES = 84

SAMPLES = 25

WEIGHTS_DIR = "./weights"


indian_states = ["Andhra Pradesh", "India", "Arunachal Pradesh", "Assam", "Bihar",
"Chhattisgarh", "Goa", "Gujarat", "Haryana", "Himachal Pradesh", "Jharkhand", "Karnataka",
"Kerala", "Madhya Pradesh", "Maharashtra", "Manipur", "Meghalaya", "Mizoram",
```



```
"Nagaland", "Odisha", "Punjab", "Rajasthan", "Sikkim", "Tamil Nadu", "Telangana",  
"Tripura", "Uttar Pradesh", "Uttarakhand", "West Bengal"]
```

```
indian_state_capitals = ["Amaravati", "New Delhi", "Itanagar", "Dispur", "Patna", "Raipur",  
"Panaji", "Gandhinagar", "Chandigarh", "Shimla", "Ranchi", "Bengaluru",  
"Thiruvananthapuram", "Bhopal", "Mumbai", "Imphal", "Shillong", "Aizawl", "Kohima",  
"Bhubaneswar", "Chandigarh", "Jaipur", "Gangtok", "Chennai", "Hyderabad", "Agartala",  
"Lucknow", "Dehradun", "Kolkata"]
```

### **# Load the classes from a pickle file**

```
with open(os.path.join(WEIGHTS_DIR, "classes.pkl"), "rb") as f:
```

```
    classes = pickle.load(f)
```

```
logger.info(f"Loaded classes -> {json.dumps(classes)}")
```

### **# Initialize video capture**

```
cap = cv2.VideoCapture(0)
```

```
mediapipeHelper = MediapipeHelper()
```

```
model = load_model_()
```

### **# Initialize memory for storing features**

```
memory = np.zeros((1, FRAMES, FEATURES))
```

```
str_array = []
```

```
str_accumulated = ""
```

### **# Initialize start time**

```
start_time = time.time()
```

```
t = 7
```

### **# Initialize pytsx3 engine**

```
engine = pytsx3.init()
```

### **# Function to start the main application**

```
def start_application():  
    start_screen.destroy()  
  
    # Create the main application UI  
  
    global video_label, prediction_label, str_label, train_button, validate_button, collect_button,  
    clear_button
```

### **# Video display label**

```
video_label = Label(root)  
video_label.pack()  
  
prediction_label = Label(root, text="Prediction: ", font=("Helvetica", 16))  
prediction_label.pack()  
  
str_label = Label(root, text="Statement: ", font=("Helvetica", 16))  
str_label.pack()
```

### **# Train button**

```
train_button = Button(root, text="Train", command=train_model, width=10, height=4)  
train_button.pack()
```

### **# Validate button**

```
validate_button = Button(root, text="Validate", command=update_frame, width=10,  
height=4)  
validate_button.pack()
```

### **# Data Collection button**

```
collect_button = Button(root, text="Collect Data", command=collect_data, width=10,  
height=4)  
collect_button.pack()
```

### **# Clear button**

```
clear_button = Button(root, text="Clear", command=clear_statement, width=10, height=4)
```

```
clear_button.pack()
```

### **# Start the video frame update loop**

```
update_frame()
```

### **# Setup the Tkinter window**

```
root = tk.Tk()
```

```
root.title("SIGN LANGUAGE TO AUDIO - GROUP 1")
```

### **# Create the start screen frame**

```
start_screen = tk.Frame(root)
```

```
start_screen.pack(fill="both", expand=True)
```

```
title_label_1 = Label(start_screen, text="College of Technology, GBPUA&T, Pantnagar",  
font=("Helvetica", 27))
```

```
title_label_1.pack(pady=8)
```

```
title_label_3 = Label(start_screen, text="Department of Computer Engineering",  
font=("Helvetica", 24))
```

```
title_label_3.pack(pady=8)
```

```
title_label_2 = Label(start_screen, text="BTech Final Year Project", font=("Helvetica", 23))
```

```
title_label_2.pack(pady=8)
```

```
title_label = Label(start_screen, text="Group: 01", font=("Helvetica", 22))
```

```
title_label.pack(pady=8)
```

```
title_label_4 = Label(start_screen, text="Sign Language to Audio Conversion",  
font=("Helvetica", 22))
```

```
title_label_4.pack(pady=8)
```

```
photo_frame = tk.Frame(start_screen)
```

```
photo_frame.pack()
```

### **# Photo labels and images**

```
photos = ["photo1.jpg", "photo2.jpeg", "photo3.jpeg", "photo4.jpeg"]
```

```
photo_labels = ["Project Guide: Dr S.D. Samantaray", "Ayush Pratap Singh (56068)", "Ansh Kumar (56094)", "Shivanshu Rawat (56930)"]
```

```
for i, photo in enumerate(photos):
```

```
    img = Image.open(photo)
```

```
    img = img.resize((200, 200), Image.LANCZOS) # Use LANCZOS for resizing
```

```
    photo_img = ImageTk.PhotoImage(img)
```

```
    # Create image label
```

```
    img_label = Label(photo_frame, image=photo_img)
```

```
    img_label.image = photo_img # Keep a reference to avoid garbage collection
```

```
    img_label.grid(row=0, column=i, padx=10, pady=10) # Place image label in grid layout
```

```
    # Create label for text below image
```

```
    label_text = photo_labels[i]
```

```
    label = Label(photo_frame, text=label_text, font=("Helvetica", 18))
```

```
    label.grid(row=1, column=i, padx=10) # Place label below corresponding image label
```

```
# Start button
```

```
start_button = Button(start_screen, text="Start", command=start_application, width=10, height=2, font=("Helvetica", 18))
```

```
start_button.pack(pady=20)
```

```
# Start the Tkinter main loop
```

```
root.mainloop()
```

```
# Release the video capture when the Tkinter window is closed
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

## **APPENDIX III**

### **User Manual & Tutorial**

#### **1. PRE-INSTALLATION REQUIREMENTS**

- a. Python
- b. Webcam
- c. The following libraries
  - i. Tensorflow
  - ii. cv2
  - iii. Mediapipe
  - iv. numpy
  - v. pyttsx3
  - vi. Tkinter
  - vii. sys
- d. Laptop

#### **2. APP INSTALLATION & INTERFACE GUIDE**

- a. Download the project code. The user needs to download the project code and the model from the CD provided with this project report.
- b. The user then needs to install python, pip and the required libraries. The libraries can be installed in a single go using the following command.  
`pip install -r requirements.txt`
- c. Once installed, the user needs to open the terminal (cmd or powershell) in the **current directory** and then type **python main\_app.py**

#### **3. DIRECTIONS FOR USE**

This sign language to audio conversion system can be used by running the provided script. The script will initialize the camera, load the pre-trained model, and establish a connection to the text-to-speech engine. Once started, simply hold your hand up in front of the webcam and perform signs. The system will continuously analyze the hand gestures, predict the corresponding signs, and if the confidence level is high enough, translate the sign into spoken words through your computer's speakers. You can also use the provided buttons to train the model on new signs, collect data for future training, or clear the accumulated statement.