

# HandTalk: Real-Time Hand Sign Recognition

*Industrial Training/Project*

**Submitted to the**



**G. B. Pant University of Agriculture & Technology  
Pantnagar-263145, Uttarakhand, India**

**By**

**Atul Kumar  
ID. No. 58042**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF**

**Master of Computer Applications**

**September, 2024**

## **ACKNOWLEDGEMENT**

*My sincere gratitude goes out to the remarkable individuals who have been the driving force behind the completion of this college project. Their unwavering support and invaluable contributions have been pivotal to its success. I extend heartfelt thanks to all those who have participated in various capacities, demonstrating a true team spirit that has been essential to bringing this project to life.*

*I want to express my profound gratitude to **Dr. Shri Prakash Dwivedi**, an esteemed advisor and Assistant Professor in the Department of Information Technology. His meticulous guidance and dedicated role as the Chairman of the Advisory Committee have been indispensable throughout this journey. Additionally, my heartfelt thanks extend to **Er. Sanjay Joshi** and **Er. Rajesh Singh**, both accomplished Assistant Professors in the Department of Information Technology, for their valuable contributions as members of the advisory committee.*

*I would like to thank **Global Inforventures Pvt. Ltd, Noida** for giving me the opportunity that I could pursue my 6 months of internship from the company and also **Mr. Sharad Shrivastav (Project Manager)**, **Miss Sugandh Gupta (Research and Training Consultant in AI)**, **Mr. Anubhav Patrick (AI Lead) Global Inforventures Pvt. Ltd, Noida** for their indefatigable guidance, valuable suggestion and moral support.*

*I extend heartfelt gratitude to **Dr. Sandeep Kumar Goel**, our esteemed MCA Coordinator, and **Dr. Ajit Kumar**, the visionary Head of the IT Department. My profound appreciation also goes to the erudite faculty members of the IT Department – **Dr. Ashok Kumar**, **Dr. Subodh Prasad**, **Dr. Binay Kumar Pandey**, **Dr. Govind Verma**, and **Er. Shikha Goswami** - for their invaluable and unwavering support.*

*I would like to express my sincere gratitude to **Dr. Alaknanda Ashok**, Dean, College of Technology and **Dr. Kiran P. Raverkar**, Dean, Post Graduate Studies, G.B. Pant University of Agriculture and Technology for their valuable suggestion and moral support. I am also thankful to all the non-teaching staff of Department of MCA Programme, especially **Mr. Prakash Chandra**, Lab Assistant, **Sunny Sharma**, clerk and **Ravi Kashyap**, Office Attendant for timely coordination of official matters.*

*I would like to thank my parents and my friends for their moral support.*

**Pantnagar**  
**September, 2024**

**(Atul Kumar)**

## **CERTIFICATE-I**

This is to certify that the Industrial Training/Project report entitled **“HandTalk: Real-Time Hand Sign Recognition”** submitted in partial fulfillment of the requirements for the degree of **Master of Computer Applications** of the College of Post Graduate Studies, G. B. Pant University of Agriculture and Technology, Pantnagar is a record of bona fide work carried out by **Mr. Atul Kumar**, Id No. **58042** under my supervision, and no part of the report has been submitted for any other degree or diploma.

The assistance and help received during the course of this Industrial Training/Project have been acknowledged.

Pantnagar  
September, 2024

**(Shri Prakash Dwivedi)**  
Chairman  
Advisory Committee

## **CERTIFICATE-II**

We, the undersigned, member of Advisory Committee of **Mr. Atul Kumar** Id. No. **58042**, a candidate for the degree of **Master of Computer Applications**, agree that the Industrial Training/Project report entitled “**HandTalk: Real-Time Hand Sign Recognition**” may be submitted in partial fulfillment of the requirements for the degree.

**(Shri Prakash Dwivedi)**

Chairman

Advisory committee

**(Sanjay Joshi)**

Member

**(Rajesh Singh)**

Member

# Table of Content

---

<b>LIST OF FIGRURES.....</b>	<b>.....</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>.....</b>
<b>CHAPTER 1: INTRODUCTION .....</b>	<b>1</b>
1.1 General .....	1
1.2 Need of the System .....	2
1.3 Advantages of the System .....	3
1.4 Objective .....	5
<b>CHAPTER 2: REVIEW OF LITERATURE .....</b>	<b>7</b>
2.1 Summary .....	7
2.2 Conclusion.....	9
<b>CHAPTER 3: MATERIALS AND METHODOLOGY .....</b>	<b>10</b>
3.1 Analysis and Feasibility of Study.....	10
3.1.1 Introduction .....	10
3.1.2 Identification of Need.....	11
3.1.3 Preliminary Investigation .....	11
3.1.4 Study of Feasibility .....	12
3.1.5 Project Planning .....	13
3.2 Design and Architecture.....	14
3.2.1 Introduction .....	15
3.2.2 Modularization Details .....	17
3.2.3 Data Flow Diagram .....	18
3.2.4 Dataset Collection .....	22
3.2.5 Data Preprocessing .....	23
3.2.6 Preparation for Model Training.....	24
3.2.7 Convolutional Neural Networks and Layers .....	25
3.2.8 Hardware & Software Used .....	29
3.3 Technology Overview .....	29
3.3.1 Frontend Technology .....	29
3.3.2 Backend Technology.....	30
3.3.3 DL Technology .....	30
3.4 Requirement & Analysis .....	31

3.4.1 Functional Requirements.....	31
3.4.2 Non-Functional Requirements .....	32
<b>CHAPTER 4: IMPLEMENTATION.....</b>	<b>34</b>
4.1 Pseudo Code.....	34
4.1.1 Data Collection.....	34
4.1.2 Model Training.....	36
4.1.3 Prediction .....	37
4.2 Testing.....	38
4.2.1 Testing Process.....	39
<b>CHAPTER 5: RESULTS AND DISCUSSION.....</b>	<b>41</b>
5.1 Results & Discussion .....	41
5.1.1 Model Accuracy .....	43
5.1.2 Confusion Matrix .....	44
5.1.3 Discussion .....	45
5.2 Merits and Demerits.....	45
5.2.1 Merits .....	45
5.2.2 Demerits .....	46
<b>CHAPTER 6: SUMMARY AND CONCLUSIONS.....</b>	<b>48</b>
6.1 Summary .....	48
6.2 Conclusion.....	49
6.3 Future Scope.....	50
<b>LITERATURE CITED.....</b>	
<b>APPENDICES.....</b>	
COMPANAY PROFILE.....	
TRAINING CERTIFICATE.....	
<b>CURRICULUM VITAE.....</b>	
<b>ABSTRACT.....</b>	
<b>WORKING OF THE PROJECT.....</b>	

## LIST OF FIGURES

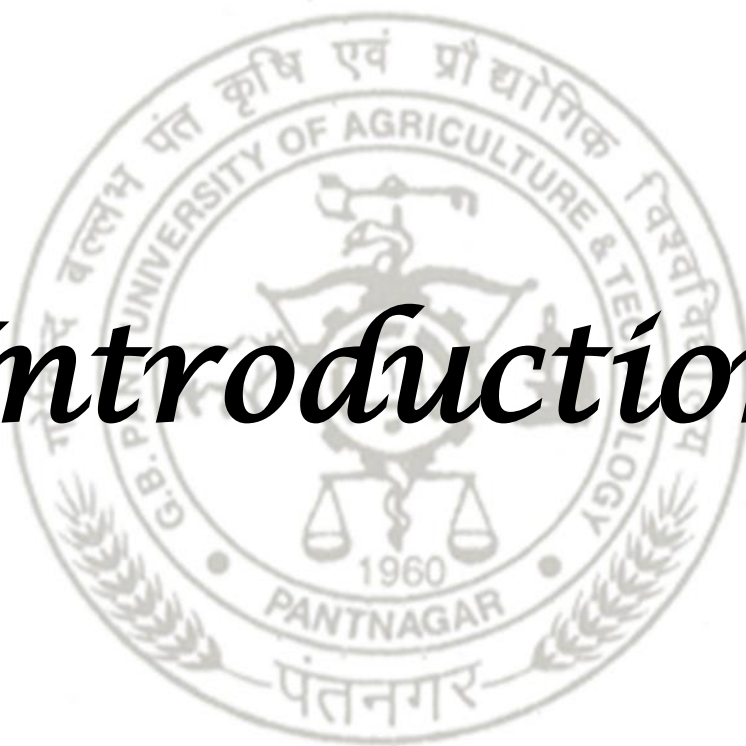
Figure No.	Title	Page
3.1	Block Diagram	15
3.2	Use Case Diagram	16
3.3	Data Flow Diagram	20
3.4	Sequence Diagram	21
3.5	Mediapipe Hand Landmarks	22
3.6	Augmentation Example	25
3.7	Convolutional Neural Network Architecture	26
3.8	Pooling Layer	27
3.9	Fully Connected Layers	28
3.10	CNN With All Layers	28
4.1	Mediapipe Hand Landmark Saved in Dataset	35
4.2	Model Summary	36
4.3	Prediction of Alphabet	37
4.4	Testing Steps	38
4.5	Testing Without GUI	40
5.1	Screenshot 1 while writing NAMASTE	41
5.2	Screenshot 2 while writing NAMASTE	42
5.3	Screenshot 3 while writing NAMASTE	42
5.4	Screenshot of written Hello with word suggestions	42
5.5	Model Accuracy Graph	43
5.6	Confusion Matrix	44

## LIST OF ABBREVIATIONS

Abbreviations	Extended/Full Form
ASL	American Sign Language
CNN	Convolutional Neural Network
DL	Deep Learning
ML	Machine Learning
AI	Artificial Intelligence
API	Application Programming Interface
GUI	Graphical User Interface
TF	TensorFlow
ROI	Region of Interest
FPS	Frames Per Second
MP	Mediapipe
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
TTS	Text-to-Speech
ISL	Indian Sign Language
BSL	British Sign Language



# *Introduction*





# INTRODUCTION

---

In this chapter we are going to introduce you to “HandTalk: Real-Time Hand Sign Recognition”, we will discuss the objective of this system with its need and advantages over the existing system.

## 1.1 General

"HandTalk: Real-Time Hand Sign Recognition" represents a significant advancement in assistive technology, designed to bridge the communication gap between the hearing and speech-impaired community and the broader population. In an increasingly connected world, the ability to communicate seamlessly is essential, and this project aims to facilitate interaction by translating American Sign Language (ASL) into text or speech in real-time.

At the heart of "HandTalk" is the integration of machine learning, deep learning, and computer vision, which work together to recognize and interpret complex hand gestures. Using advanced image recognition algorithms and neural networks, the system processes hand signs from a video feed, identifying the corresponding ASL signs with precision and speed. The web-based platform is capable of capturing live input through a webcam or mobile device camera, translating it instantly into written text or synthesized speech, depending on the user's preference.

The application of such a system is vast. It provides a powerful tool for individuals who rely on ASL as their primary mode of communication, enabling them to communicate more effectively with those who are unfamiliar with sign language. Additionally, the system holds value in professional and educational settings, where it can assist educators, healthcare providers, and service professionals who interact with ASL users on a regular basis. By providing real-time translation, "HandTalk" reduces the reliance on human interpreters, offering a convenient, accessible alternative.

"HandTalk" is not just a technological innovation; it is a step towards greater inclusivity. As society becomes more interconnected, ensuring that everyone has the ability to communicate, regardless of physical or sensory impairments, is of paramount importance. This project stands as a testament to the potential of AI and machine learning in breaking down barriers and fostering social integration, making everyday interactions smoother for millions of people. With continued refinement and expansion, "HandTalk" could evolve to recognize an even broader array of signs, further enhancing its impact.

By providing a reliable, user-friendly platform, "HandTalk" contributes to the broader vision of an inclusive society, where technology empowers all individuals to engage in meaningful communication.

## **1.2 Need of the System**

"HandTalk: Real-Time Hand Sign Recognition" represents a significant advancement in assistive technology, designed to bridge the communication gap between the hearing and speech-impaired community and the broader population. In an increasingly connected world, the ability to communicate seamlessly is essential, and this project aims to facilitate interaction by translating American Sign Language (ASL) into text or speech in real-time. At the heart of "HandTalk" is the integration of machine learning, deep learning, and computer vision, which work together to recognize and interpret complex hand gestures. Using advanced image recognition algorithms and neural networks, the system processes hand signs from a video feed, identifying the corresponding ASL signs with precision and speed. The web-based platform is capable of capturing live input through a webcam or mobile device camera, translating it instantly into written text or synthesized speech, depending on the user's preference.

The application of such a system is vast. It provides a powerful tool for individuals who rely on ASL as their primary mode of communication, enabling them to communicate more effectively with those who are unfamiliar with sign language. Additionally, the system holds value in professional and educational settings, where it can assist educators, healthcare providers, and service professionals who interact with ASL users on a regular basis. By providing real-time translation, "HandTalk" reduces the reliance on human interpreters, offering a convenient, accessible alternative.

"HandTalk" is not just a technological innovation; it is a step towards greater inclusivity. As society becomes more interconnected, ensuring that everyone has the ability to communicate, regardless of physical or sensory impairments, is of paramount importance. This project stands as a testament to the potential of AI and machine learning in breaking down barriers and fostering social integration, making everyday interactions smoother for millions of people. With continued refinement and expansion, "HandTalk" could evolve to recognize an even broader array of signs, further enhancing its impact.

### 1.3 Advantages of the System

The development of “*HandTalk: Real-Time Hand Sign Recognition*” offers numerous advantages, greatly enhancing communication for individuals who rely on American Sign Language (ASL) and those around them. Below are the key benefits of this system:

- **Real-Time Sign Recognition:** The most significant advantage of *HandTalk* is its ability to recognize and translate ASL signs in real-time, enabling seamless and immediate communication. This eliminates the need for delays or the presence of a human interpreter, allowing for natural conversations between ASL users and non-signers.
- **Enhanced Accessibility:** As a web-based application, *HandTalk* is accessible from any device equipped with a camera and an internet connection. This ensures that individuals can use the system anywhere, anytime, making communication more inclusive and convenient in various settings, from personal conversations to professional environments.
- **Increased Independence for ASL Users:** By removing the reliance on human interpreters or written notes, *HandTalk* empowers ASL users to communicate independently. This fosters self-sufficiency in everyday situations, from navigating public services to engaging in social interactions.
- **Cost-Effective Communication:** The system provides a cost-effective alternative to traditional interpreter services, especially in situations where hiring a professional interpreter may not be practical or feasible. With *HandTalk*, users can bridge the communication gap without incurring additional costs.
- **Improved Social Integration:** By facilitating communication between ASL users and non-signers, *HandTalk* promotes greater social integration. It reduces the risk of social isolation for hearing and speech-impaired individuals and enhances their ability to participate fully in society, whether at work, in education, or in everyday life.
- **User-Friendly Interface:** *HandTalk* is designed with simplicity and ease of use in mind. Its intuitive interface allows users to engage with the system without the need for extensive technical knowledge. The application’s layout ensures that users of all ages and technical backgrounds can access and operate it efficiently.

- **Accurate Gesture Recognition:** Utilizing state-of-the-art deep learning algorithms and neural networks, *HandTalk* provides highly accurate hand gesture recognition. This ensures reliable communication and minimizes misunderstandings that can arise from incorrect interpretations of signs.
- **Continuous Improvement Through Machine Learning:** The system's deep learning model allows it to improve its recognition capabilities over time. As more data is gathered and processed, the model becomes more accurate and efficient, enhancing its long-term performance and reliability.
- **Application in Diverse Settings:** *HandTalk* has the potential to be used in a wide range of settings, from personal conversations to classrooms, workplaces, and public services. It can also be integrated into customer service, healthcare, and legal environments where communication with ASL users is essential.
- **Promoting Inclusivity and Diversity:** *HandTalk* contributes to a more inclusive society by providing a platform that breaks down communication barriers. It aligns with the growing global emphasis on accessibility and diversity, ensuring that hearing and speech-impaired individuals are not left behind in the digital age.
- **Support for Educators and Professionals:** *HandTalk* can be a valuable tool for educators, caregivers, and professionals who work closely with individuals who use sign language. It enhances their ability to interact with ASL users and provides them with an efficient means of communication, improving their overall service quality.

By incorporating “*HandTalk: Real-Time Hand Sign Recognition*” into daily life, we move toward a future where communication is more accessible, inclusive, and equitable. The system stands to revolutionize how we interact with the hearing and speech-impaired community, ensuring that no one is left behind in an increasingly interconnected world.

## 1.4 Objective

The development of “*HandTalk: Real-Time Hand Sign Recognition*” aims to achieve several key objectives that contribute to improving communication for the hearing and speech-impaired community. These objectives are as follows:

- **Bridge Communication Gaps Between ASL Users and Non-Signers:** The primary objective of *HandTalk* is to facilitate real-time communication between individuals who use American Sign Language (ASL) and those who do not. By providing a reliable and instantaneous translation of hand gestures into text or speech, the system helps overcome language barriers, enabling smoother and more effective interactions.
- **Promote Inclusivity and Accessibility:** *HandTalk* aims to create a more inclusive society by ensuring that ASL users can communicate freely and without limitations. The system’s web-based platform allows it to be accessible from anywhere, fostering greater accessibility for users in various settings—personal, professional, and public. This objective aligns with global efforts to improve digital inclusivity for people with disabilities.
- **Empower the Hearing and Speech-Impaired Community:** The system seeks to empower individuals who rely on ASL by providing them with a tool that enables independent communication. By reducing the need for human interpreters, *HandTalk* promotes self-sufficiency and gives ASL users the ability to communicate more freely in different aspects of their lives, from casual interactions to professional environments.
- **Enhance User Experience Through Accuracy and Simplicity:** Another key objective is to provide users with an easy-to-use and accurate system. By leveraging advanced deep learning algorithms, *HandTalk* aims to ensure precise and reliable sign recognition. The user-friendly interface ensures that the system can be used by individuals of all ages and technical abilities without requiring extensive knowledge or training.
- **Foster Social Integration and Reduce Isolation:** By enabling more natural communication between ASL users and non-signers, *HandTalk* seeks to reduce the social isolation often faced by hearing and speech-impaired individuals. This

objective focuses on promoting meaningful engagement in social, educational, and professional settings, allowing ASL users to participate more fully in society.

- **Support Professional and Educational Environments:** The system aims to provide practical solutions for professionals and educators who interact with ASL users. By offering real-time translation, *HandTalk* enhances communication in diverse settings such as classrooms, customer service, healthcare, and public services, contributing to more effective service delivery and teaching.
- **Continuous Improvement Through Machine Learning:** One of the project's core objectives is to continually refine and improve its recognition accuracy through the collection of additional data. As the system gathers more sign data, its deep learning model will evolve, making it more precise and reliable over time, thus staying up-to-date with evolving communication needs.
- **Democratize Access to Assistive Technology Globally:** By offering a web-based platform, *HandTalk* aims to make its benefits available to a global audience, irrespective of geographic location. This objective emphasizes the need for universal access to assistive communication tools, ensuring that ASL users around the world can benefit from the system's capabilities.

In conclusion, the objectives of "*HandTalk: Real-Time Hand Sign Recognition*" encompass both technical and social aspects, focusing on improving communication, promoting inclusivity, and empowering individuals through innovative technology. These objectives contribute to the broader goal of enhancing accessibility and fostering a more connected and inclusive world.



The logo of the University of Agriculture and Technology, Pantnagar, is a circular emblem. It features a central gear-like design with a book and a torch. The text "UNIVERSITY OF AGRICULTURE & TECHNOLOGY" is written in English around the top half, and "पंत कृषि एवं प्रौद्योगिक विज्ञान विश्वविद्यालय" is written in Hindi around the bottom half. The year "1960" and "PANTNAGAR" are also visible.

# *REVIEW OF LITERATURE*



## REVIEW OF LITERATURE

---

### 2.1 Summary

The problem of communication barriers faced by hearing-impaired individuals has driven significant research in the field of sign language recognition (SLR). Over the past decade, various approaches have been developed, leveraging advancements in computer vision, machine learning, and human-computer interaction. These approaches aim to bridge the gap between sign language users and the hearing community, making communication seamless and effective. This section reviews key literature and studies that contribute to this evolving domain.

**Amatya et al. (2014)** explored the potential of Microsoft's Kinect for Windows v2 for real-time SLR. This study was pioneering in demonstrating the use of depth sensors to capture hand gestures and body movements. The depth data provided by the Kinect allowed for better hand tracking, which resulted in more accurate sign recognition. The study emphasized that depth information could complement traditional RGB data, making the recognition process more robust & less dependent on lighting conditions. [1]

**Bora et al. (2023)** introduced the use of Mediapipe and deep learning for real-time Assamese sign language recognition. Their research demonstrated the effectiveness of using pre-built frameworks like Mediapipe to speed up development and ensure reliable hand tracking. The combination of Mediapipe and deep learning proved to be a robust solution for real-time sign recognition, particularly in low-resource settings. [2]

**Chen (2021)**, in a course project report, implemented a similar approach but with a focus on American Sign Language (ASL). The work underscored the importance of using pre-trained models and transfer learning to tackle the challenges of limited training datasets for sign language recognition. Chen demonstrated that by utilizing deep learning models like CNNs and augmenting them with transfer learning, higher accuracy could be achieved even with relatively small datasets. [3]

**Dhiman (2021)** explored the use of traditional computer vision techniques for real-time hand gesture recognition. The study focused on understanding complexities of motion tracking & real-time processing, highlighting the role of feature extraction in determining the accuracy of sign language recognition systems. Dhiman emphasized that the success of

such systems lies in precise hand localization and the extraction of spatial and temporal information from sign gestures, which serve as critical inputs for ML models. [4]

**Joshi et al. (2018)** focused on using edge detection and cross-correlation techniques for ASL translation. Their study demonstrated that simple image processing techniques like edge detection could be combined with more complex cross-correlation algorithms to achieve relatively good accuracy in recognizing static signs. This approach highlighted the effectiveness of traditional methods in solving specific problems within SLR, especially in scenarios with limited computational resources. [5]

**Noltey et al. (2007)** discussed communication systems in automotive applications, which, although not directly related to SLR, provided insights into real-time data processing and communication, crucial aspects of a real-time SLR system. The techniques discussed in this paper can be adapted to ensure seamless and efficient communication in SLR systems, particularly when designing systems with strict real-time performance requirements. [6]

**Pigou et al. (2014)** employed convolutional neural networks for the automatic recognition of sign language. Their study demonstrated how CNNs could be applied to sign language recognition tasks, showing the importance of using deep learning architectures that can automatically extract features from the input data. Their work was foundational in showcasing that CNN-based models can outperform traditional machine learning techniques, particularly when dealing with large-scale datasets. [7]

**Pressman (1992)** provided a general overview of software engineering practices, which is relevant for the development of a robust SLR system. His work on system design and software architecture is foundational for ensuring that the SLR systems are not only accurate but also scalable and maintainable over time. The design principles discussed by Pressman can be applied to the architecture of modern SLR applications, ensuring that they are both user-friendly and technically sound. [8]

**Rastgoo et al. (2021)** conducted a comprehensive survey on SLR, discussing the evolution of methods from traditional image processing techniques to the latest deep learning approaches. This survey is particularly valuable for understanding the strengths and weaknesses of various methodologies, including the shift towards end-to-end deep learning pipelines that can handle both hand detection and sign recognition simultaneously. Rastgoo's work also highlighted the importance of real-time performance and the challenges of dataset availability. [9]

**Sharma et al. (2017)** introduced a communication device prototype for differently-abled individuals, which included a sign language recognition component. The prototype

demonstrated the feasibility of creating low-cost solutions for communication, combining hardware and software to enable real-time translation of sign language into text or speech. This research pointed out the practical applications of sign language recognition technologies in creating assistive devices for daily use. [10]


**Sajanraj and Beena (2018)** focused on using Region of Interest Convolutional Neural Networks (ROI-CNNs) for the recognition of Indian Sign Language (ISL) numerals. The ROI-based approach allowed for precise segmentation of hands from the background, which was a major challenge in sign language recognition. The CNN model improved performance significantly over traditional methods. This research provided insights into how deep learning, particularly CNNs, can enhance recognition accuracy by focusing on specific regions of interest, minimizing noise from non-essential areas of an image. [11]

**Zuo et al. (2023)** discussed the integration of natural language processing (NLP) techniques with SLR. Their study showed that by leveraging NLP, it is possible to provide context-aware translations of sign language, enhancing the overall accuracy and usability of SLR systems. This research opened new avenues for combining SLR with other AI technologies, allowing more nuanced & contextually relevant translations. [12]

## 2.2 Conclusion

The review of literature reveals significant progress in the field of sign language recognition, driven by advancements in computer vision, deep learning, and sensor technologies. Traditional approaches focused on handcrafted features and image processing techniques, while modern methods leverage deep learning models like CNNs and NLP to enhance recognition accuracy and contextual understanding. The integration of tools such as Kinect and Mediapipe has further improved real-time performance, making SLR systems more feasible for everyday applications. However, challenges such as dataset availability, real-time processing, and contextual understanding remain. Future research should focus on improving dataset diversity, integrating multimodal inputs, and enhancing system scalability for real-world use cases.



The logo of Pantnagar University of Agriculture & Technology is a circular emblem. It features a central shield with a book and a torch, surrounded by a wreath. The text "PANTNAGAR UNIVERSITY OF AGRICULTURE & TECHNOLOGY" is written in English around the top half of the circle, and "पंतनगर विश्वविद्यालय कृषि एवं प्रौद्योगिकी" is written in Hindi around the bottom half. The year "1960" is inscribed at the bottom center of the emblem.

# *Materials and Methodology*

## MATERIALS AND METHODOLOGY

---

This chapter introduces the proposed system which consists of architecture, objectives, need and advantages of the system.

### 3.1 Analysis and Feasibility of Study

A feasibility study sometimes called a feasibility analysis or feasibility report is a way to evaluate whether or not a project plan could be successful. A feasibility study evaluates the practicality of your project plan in order to judge whether or not you're able to move forward with the project.

System analysis is usually the first phase of a large-scale software development project and for this project we start from the basic need of the system mentioned above. The purpose of this phase is to identify and document the exact requirements for the system. There are two major activities in this phase: problem understanding or analysis and requirement specification. In problem analysis, the analyst has to understand the problem and its context. In requirement specification, the analyst has to understand what the exact requirements of the customer are. The goal of this activity is to understand the requirements of the new system that is to be developed.

#### 3.1.1 Introduction

There are two tasks basically accomplished under this phase of software development process, they are:

- Analyzing or understanding the problem statement.
- Development of requirement specification.

In problem analysis, the problem analyst has to understand the problem and its context. He has to analyze the technical skills and capabilities of the software development team members and determine whether the relevant technology is stable and established. In requirement specification, the analyst has to understand what the exact requirements of the user are. The goal of this activity is to understand the requirements of the new system that is to be developed. Hence while doing the system analysis requirements of the user should be kept in mind and should be done in a careful manner.



### 3.1.2 Identification of Need

This most important thing about developing this system is to identify the exact requirement of the customer. In this system, firstly we have mentioned our need which we want to develop. The need and specification phase of system analysis is done in this system to find the exact requirement of the customer. In this phase we collect all the necessary information of the customer which he wants in their system.

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specific user requirements. Technical feasibility also performs the following tasks.

- Analyzes the technical skills and capabilities of the software development team members.
- Determines whether the relevant technology is stable and established.
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required. Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on.

### 3.1.3 Preliminary Investigation

Preliminary investigation helps to outline the feasibility of the project, whether the proposed application would be helpful for the organization or not. The activities undergone during this phase are as mentioned below:

- **Information assessment:** Identifies information about whether the system helps in achieving the objectives of the organization. It also verifies that the system can be implemented using new technology and within the budget and whether the system can be integrated with the existing system.
- **Information collection:** Specifies the sources from where information about software can be obtained. Generally, these sources include users (who will operate

the software), organization (where the software will be used), and the software development team (which understands user requirements and knows how **report writing** uses a feasibility report, which is the conclusion of the feasibility study by the software development team. It includes the recommendations whether the software development should continue. This report may also include information about changes in the software scope, budget, and schedule and suggestions of any requirements in the system.

- **General information:** Describes the purpose and scope of feasibility study. It also describes system overview, project references, acronyms and abbreviations, and points of contact to be used. **System overview** provides description about the name of the organization responsible for the software development, system name or title, system category, operational status, and so on.

### 3.1.4 Study of Feasibility

After the initial investigation process gets completed, the results are considered as the basis for the further detailed survey i.e., the feasibility study. The objective of the feasibility study is not to solve the problem but to acquire a range of its scope. Feasibility study can be understood as the assessment of the product about its impact, how successful will it be fulfilling the requirements of the user. In this project feasibility study is categorized into three subparts namely:

- Operational feasibility.
- Technical feasibility and
- Economic feasibility

#### i. **Operational Feasibility**

- This application is operationally feasible because all the users can easily operate and access the facilities meant for them.
- The resource can be utilized in an efficient manner due to the well-planned architecture of the application.
- The well-planned architecture would also ensure the security against threats.

#### ii. **Technical Feasibility**

Technical feasibility evaluates the available infrastructure (such as hardware and software) and technologies needed to meet the consumer needs of software under time and

budget constraints. The product development team determines whether current tools and technology should be modified or applied to the program to fulfil the identified users' needs. The following are the activities often performed by technical feasibility.

- Examines the technological expertise and talents of members of the software development team.
- Determines whether the application infrastructure is reliable and well-established.
- Ensures that the technology selected for software creation has many customers who can be contacted when issues emerge or required changes.

### iii. **Economic Feasibility**

The extent to which the required software completes a sequence of steps to address market challenges and consumer requirements is measured by operational viability. This Feasibility is based on human capital (the product development team) and entails visualizing whether the software can work after it is built and be operational after it is installed. The following are the operations carried out by operational feasibility:

- Determines whether the issues outlined in the consumer specifications are a high priority.
- Determines whether the software development team's proposed approach is suitable.
- Determines whether or not consumers can respond to new technologies.
- Determines whether the possible solutions suggested by the software development team satisfy the company.

### **3.1.5 Project Planning**

Project planning is part of project analysis, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Project planning can be done manually or by the use of project management software. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. Project planning is often used to organize different areas of a project, including project plans, workloads and the management of teams and individuals. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path.

Project planning is inherently uncertain as it must be done before the project is actually started. Therefore, the duration of the tasks is often estimated through a weighted average of optimistic, normal, and pessimistic cases. The critical chain method adds "buffers" in the planning to anticipate potential delays in project execution. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project schedule may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the project schedule becomes what is known as the baseline schedule.

- The planning phase itself takes a couple of days. These are the requirements for project planning.
- It provides the important basis for project management and hence increase the management of the project.
- The project plan is usually produced before the development process begins and is updated as the development further proceeds and data about progress of the project becomes available.
- In this website, planning is done for developing the website which meets the user's needs.

## 3.2 Design and Architecture

This topic describes the software functions and features. It includes functional diagrams, screen layouts, business rules, business process diagrams, and an entity-relationship diagram with a full data dictionary.

The block diagram below offers a simplified representation of the system architecture for *HandTalk: Real-Time Hand Sign Recognition*. It highlights the flow of data through various components such as input capture, preprocessing, model prediction, and output generation.

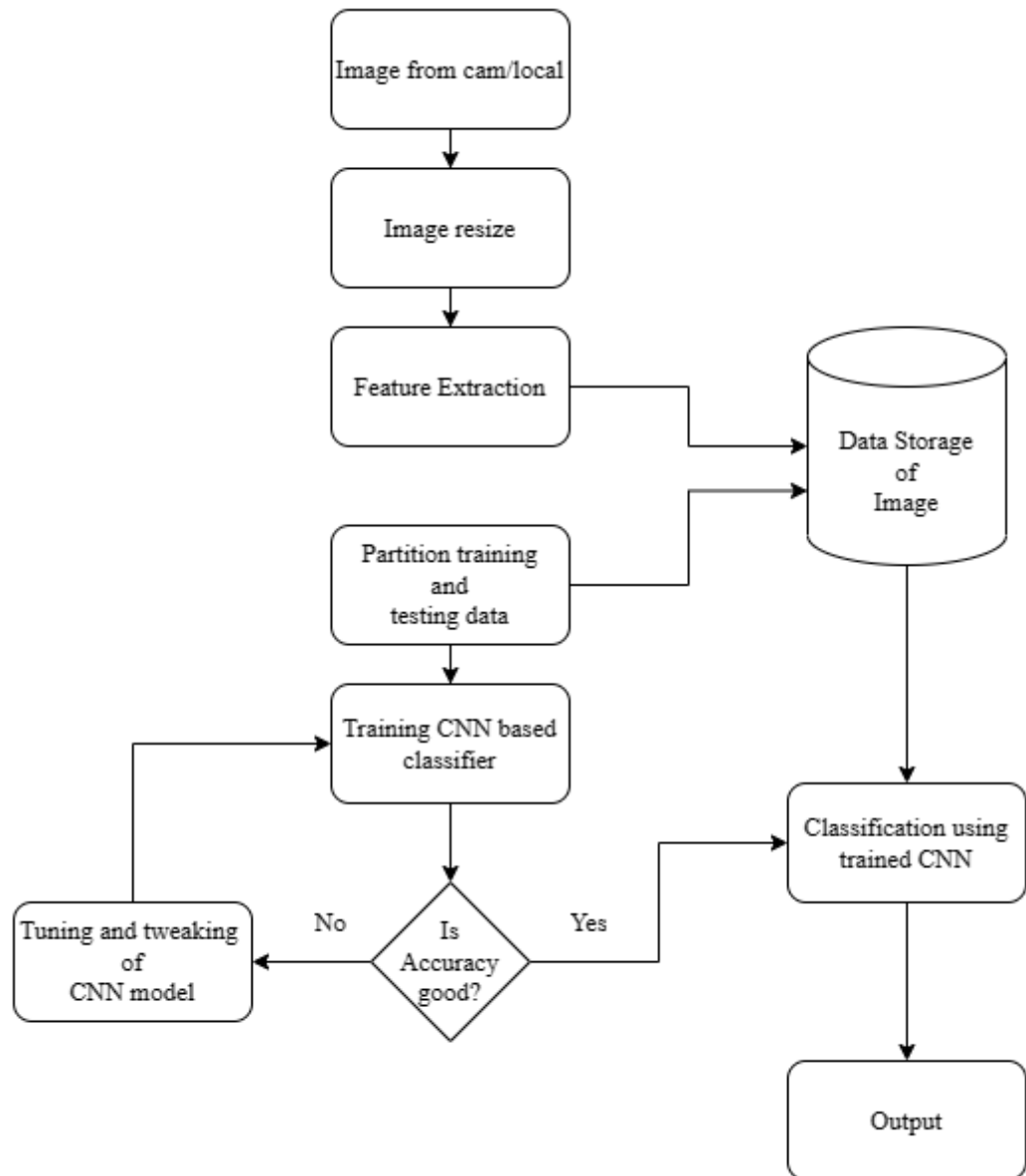


Figure 3.1 Block Diagram

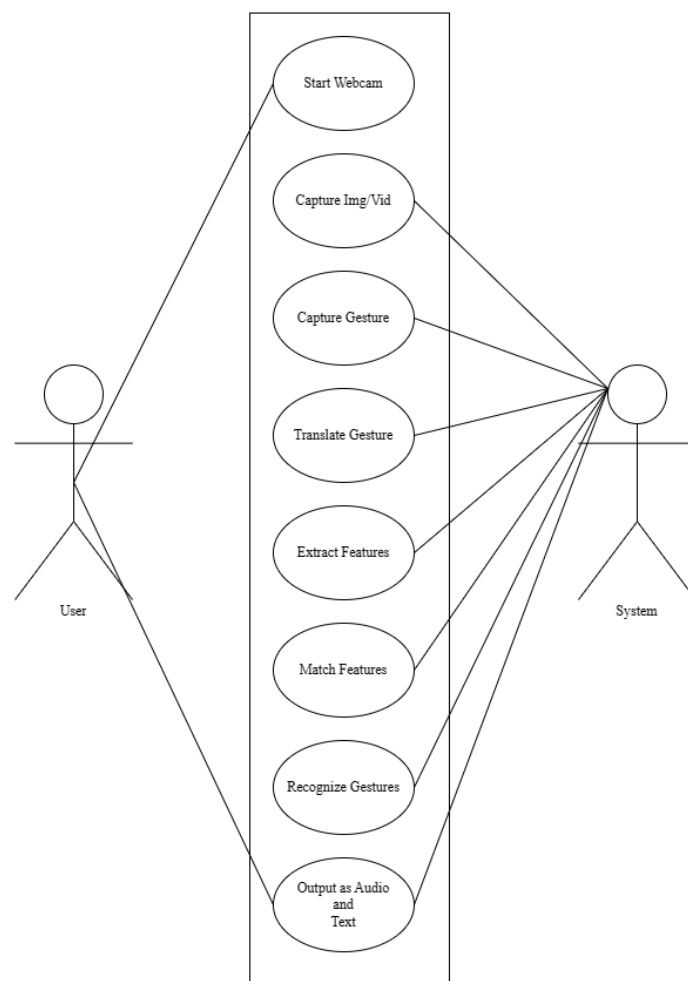
The **Figure 3.1** shows the block diagram of the Desktop application from the level of Data collection to building a model and utilizing that model from backend and to show the result in a frontend.

### 3.2.1 Introduction

The project, *HandTalk: Real-Time Hand Sign Recognition*, is a Deep Learning-based desktop application designed to detect and recognize American Sign Language (ASL) gestures in real time. This system primarily serves the hearing-impaired community, enabling communication with those who may not be familiar with sign language. The application utilizes Mediapipe for efficient hand-tracking and TensorFlow to implement

deep learning models for gesture recognition. The custom GUI for the project is built using Tkinter, providing an intuitive interface for the users. Deep Learning, a subset of Machine Learning (ML), is at the core of this application. It leverages neural networks to automatically learn and extract patterns from large datasets. In this case, the application relies on a trained model to predict hand signs from video input. The entire process is two-fold: first, Mediapipe captures and processes the hand's movement and structure in real time, feeding the data to the deep learning model. Second, the trained TensorFlow model classifies these movements into recognized signs, displaying the output in a user-friendly GUI.

The following use case diagram illustrates the different ways a user interacts with the system. It highlights the core functionalities provided by the application, such as live hand sign detection, word formation, and text-to-speech conversion.



*Figure 3.2 Use Case Diagram*

### 3.2.2 Modularization Details

The *HandTalk: Real-Time Hand Sign Recognition* system is divided into several functional modules to ensure a smooth, user-friendly experience and efficient performance. The modular design allows for the application to be organized, manageable, and scalable. Each module has its own defined task and works in coordination with the others to deliver the overall functionality of the system. The following are the key functional modules:

- **Live Camera Feed Module**
  - This module captures real-time video feed from the user's webcam. It is responsible for continuously providing the live hand gestures as input to the system. The camera feed serves as the foundation for recognizing hand gestures, ensuring the seamless flow of data to the hand-tracking and recognition processes.
- **Mediapipe Live Extracted Feature Module**
  - This module is responsible for detecting and extracting key hand landmarks from the live camera feed using Mediapipe. Mediapipe processes the live video frames and identifies key points on the hand, such as finger joints and palm orientation, which are crucial for recognizing hand gestures. These extracted features are then passed to the deep learning model for sign classification.
- **ASL Hand Signs Display Module**
  - This module displays an image or chart of the ASL hand signs for user reference. It serves as a visual guide, allowing users to compare their hand gestures with the recognized ASL signs. This feature is helpful, especially for users learning sign language or needing assistance in identifying correct hand postures.
- **Current Alphabet and Word Display Module**
  - This module dynamically shows the alphabet being recognized from the live camera feed in real-time. As the user makes different gestures, the recognized alphabet is displayed on the screen. Additionally, as more letters are recognized, this module also constructs a word by concatenating the recognized alphabets, providing users with immediate feedback.

- **Word Suggestions Module**

- This module suggests possible words based on the alphabets recognized so far. Using a predefined dictionary of common words, it helps users by offering suggestions for completing words based on their current input. This enhances the overall efficiency of the system, especially when dealing with longer words.

- **Clear and Speak Buttons Module**

- **Clear Button:** This module allows the user to clear the current input (both alphabet and word) from the display, resetting the recognition process.
- **Speak Button:** This module converts the constructed word into speech using a text-to-speech (TTS) engine. Once the word is fully recognized, the user can press this button to hear the word spoken aloud, providing an audio output to facilitate communication.

Each of these modules works in tandem to ensure smooth and effective operation of the *HandTalk* application, providing real-time recognition and feedback to the user while maintaining a seamless user experience through the GUI built with Tkinter.

### 3.2.3 Data Flow Diagram

The Data Flow Diagram (DFD) is a structured analysis and design method. It is traditional visual representation of the information flows within a system. It is widely used for software analysis and design. A neat and clear DFD can depict a good amount of the system requirements graphically. DFD depicts the logic models and expresses data transformation in a system. It includes a mechanism to model the data flow and supports decomposition to illustrate details of the data flows and functions. A Data Flow Diagram cannot present information on operation sequence. Therefore, it is not a process or procedure modeling method. DFD includes following characteristics:

- Supporting the analysis and requirement stage of system design.
- A diagramming technique with annotation.
- Describing a network of activities/processes of the target system.
- Allowing for behaviors of parallel and asynchronous.
- Stepwise refinement through hierarchical decomposition of processes.

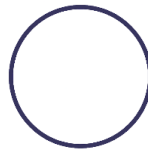


Various notations used while drawing data flow diagrams are:

**Dataflow:** Starting from a particular origin data proceeds towards the destination.



**Process:** The people, devices or procedures that transform the data. No physical components considered.



**Source:** It can be any of the people, programs, data, organization or data that are the external sources of the data.

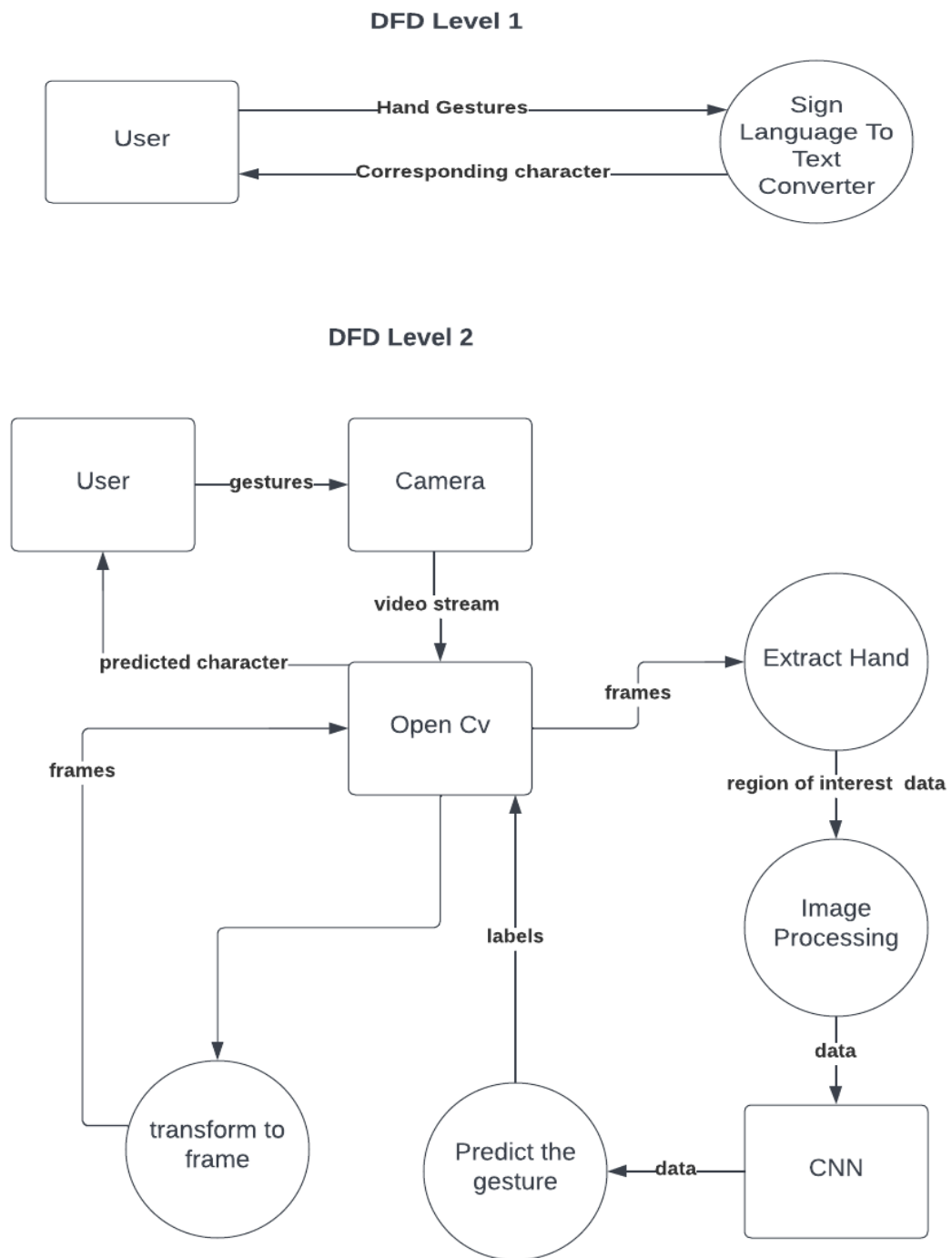


**Data Store:** In a system the data are stored or processed in a data store.



### Data Flow Diagram:

In **Figure 3.3** there is DFD, which is very simple in nature. On left side we have user, which shows a hand sign in front of the camera live feed. Then frontend send that image to backend, where its preprocessed and then given to the TensorFlow model Which then predicts the character and give results in to the Current Alphabet Module.



*Figure 3.3 Data Flow Diagram*

The data flow diagram (DFD) provides a closer look at how data moves within the system. It visualizes the sources, processes, and destinations of data as it is captured, processed, and outputted, offering insight into the internal operations of the application.

### Sequence Diagram:

The sequence diagram is more detailed, showing the step-by-step flow of interactions between the system components and how data is processed over time. It explains the sequence in which processes are executed, from gesture detection to output.

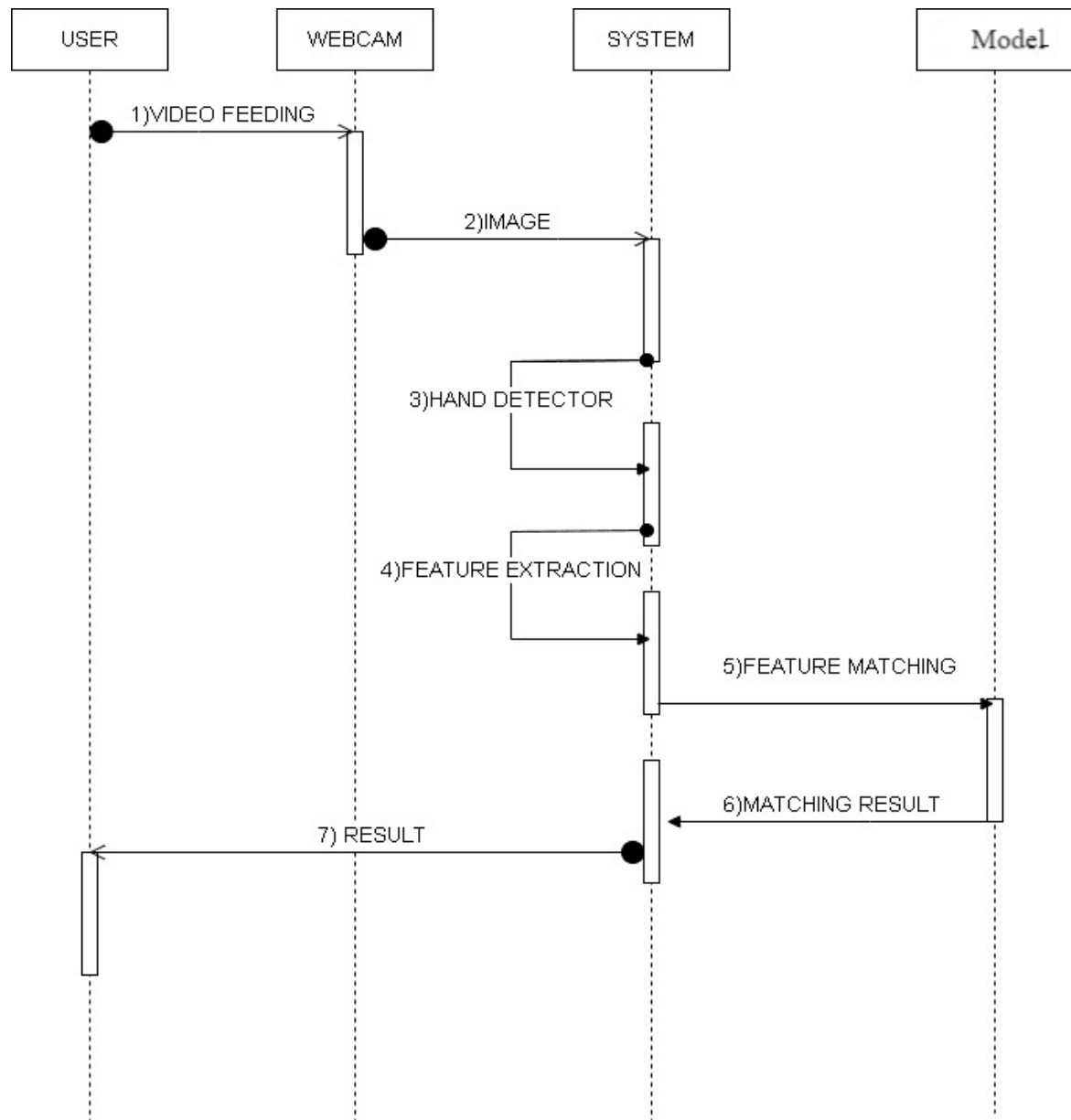


Figure 3.4 Sequence Diagram

The sequence diagram above demonstrates the flow of operations within the system. It provides a detailed breakdown of how various components interact to detect hand gestures and generate the appropriate output in real-time.

### 3.2.4 Dataset Collection

The dataset for American Sign Language (ASL) recognition was obtained by capturing images of hand gestures representing every alphabet using the Mediapipe hand tracking library. The hand-tracking library provided a fast and reliable way to detect hand landmarks and track hand movements in real time. The captured hand gestures were then mapped onto a white background image, which provided a consistent and neutral background for image processing and classification. The resulting dataset can be used to train machine learning models for ASL recognition and assistive technology applications. Mediapipe hand tracking library uses a deep learning-based approach to detect and track human hands in real time. The library is based on a lightweight convolutional neural network (CNN) model, which is trained on millions of annotated hand images to learn the hand landmarks and their connections. The network is optimized for efficiency and can run on mobile devices and desktop computers.

The library first detects the hand regions in the input image using a bounding box regression algorithm. It then feeds the detected hand regions to the hand landmark model to estimate the landmarks of each hand. The hand landmarks are a set of 21 2D points that represent the joints and fingertips of the hand. The hand landmark model is a feed-forward neural network, which takes the detected hand region as input and produces the landmarks as output. The model is trained using a combination of synthetic and real data to generalize to different hand shapes, skin colors, and lighting conditions.

Once the hand landmarks are detected, the library can use them for various applications, such as gesture recognition, hand pose estimation, and augmented reality. The library also provides a set of utilities for visualizing and processing the hand landmarks, such as drawing hand annotations, calculating hand features, and filtering noisy landmarks.

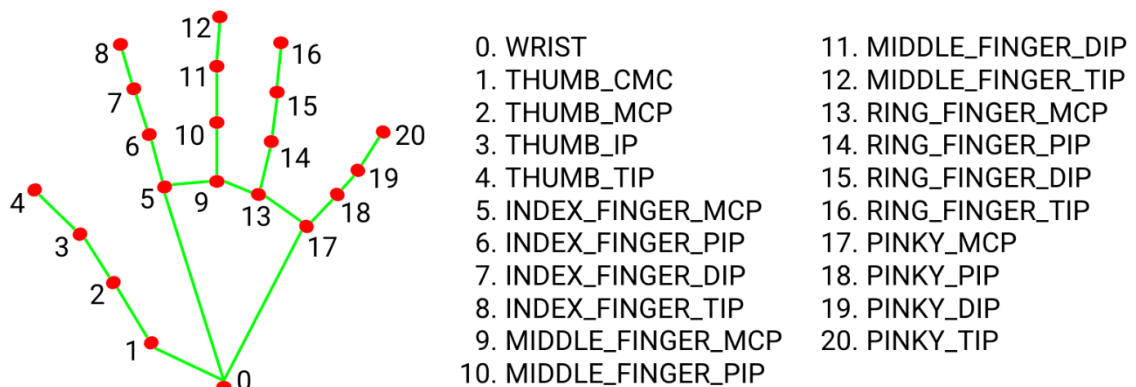


Figure 3.5 Mediapipe Hand Landmarks

### 3.2.5 Data Preprocessing

The preprocessing stage is a crucial step in preparing the dataset for training the deep learning model in the *HandTalk: Real-Time Hand Sign Recognition* system. The key goal was to transform the raw hand sign images into a format that focuses on the essential features required for accurate recognition, while discarding unnecessary details such as background noise and irrelevant visual data. For this project, the following preprocessing steps were carried out:

- **Image Conversion to Mediapipe Skeleton:** Each raw image containing the ASL hand signs was passed through Mediapipe's hand detection pipeline to extract the key hand landmarks. Mediapipe detected 21 specific points on the hand, including finger joints, tips, and the wrist. These points were then used to construct a skeleton-like representation of the hand. The skeleton consisted of lines connecting the key landmarks, providing a simplified but highly informative visual representation of the hand's structure.
- **Background Removal and White Background Application:** After the skeleton was created, the background of the image was completely removed, leaving only the skeleton hand landmarks on a clean, white background. This transformation ensured that the deep learning model would focus exclusively on the hand gesture itself, without being influenced by any background distractions or lighting conditions.
- **Image Resizing to 400x400 Dimensions:** To maintain uniformity across the dataset, each skeleton image was resized to a fixed dimension of 400x400 pixels. This standardization ensured that all images fed into the deep learning model were of the same size, simplifying the learning process for the model and improving computational efficiency.
- **Saving Pre-processed Images:** The resulting 400x400 skeleton images with white backgrounds were saved and used as input for model training. This streamlined dataset, now consisting of clean, consistent hand gesture images, enabled the model to learn more effectively from the extracted hand structures, leading to improved accuracy during the recognition phase.

By converting the raw hand sign images into simplified skeleton images, the preprocessing stage ensured that the model could focus on the key features needed to

distinguish between different ASL signs, while maintaining a high level of accuracy and real-time performance.

### **3.2.6 Preparation for Model Training**

Before initiating the training process, careful preparation was essential to ensure the deep learning model would effectively learn to recognize hand signs with high accuracy. This phase involved organizing the pre-processed dataset, selecting appropriate model architecture, and configuring the training environment.

The pre-processed dataset, consisting of 400x400 skeleton images with white backgrounds, was split into training, validation, and testing sets. This ensured that the model was trained on a subset of the data, validated on unseen samples to monitor performance, and tested on a completely different set to evaluate its generalization.

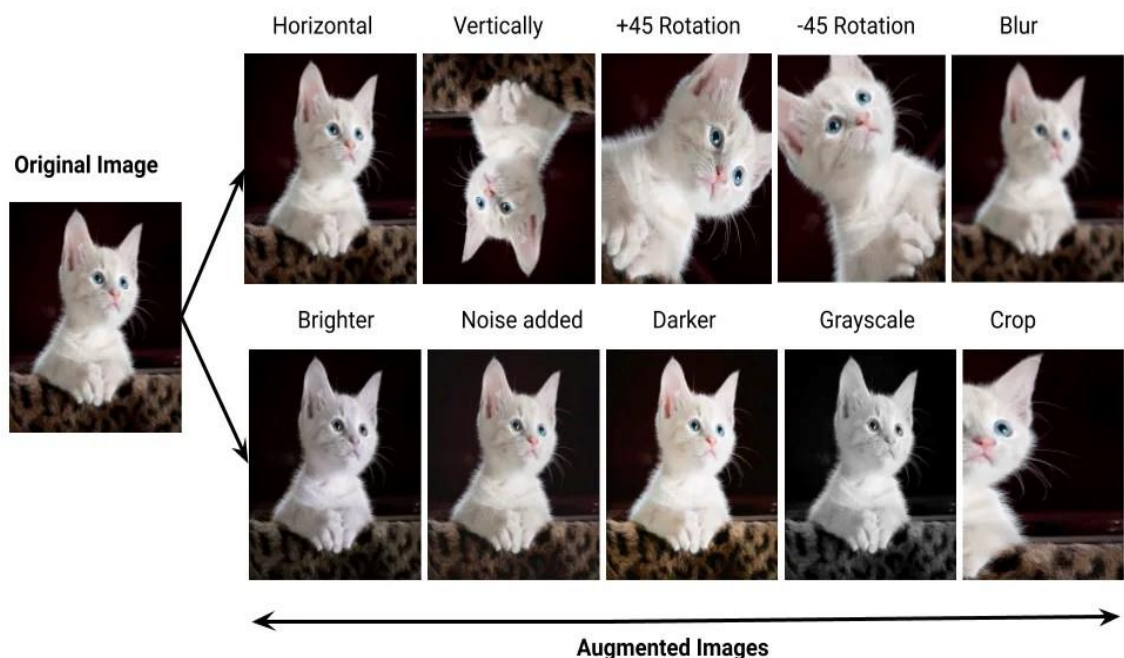
During this phase, hyperparameters such as the batch size, learning rate, and number of epochs were carefully chosen to optimize the training process. Additionally, data augmentation techniques such as random rotations, flips, and slight scaling were applied to further enhance the dataset's diversity and prevent overfitting.

Finally, the model architecture was set up using TensorFlow, with the skeleton hand landmarks acting as the primary input features. The training environment was configured to utilize GPU acceleration for faster processing, ensuring that the deep learning model could be trained efficiently and effectively on the custom hand sign dataset. During this stage, a crucial step is to partition the dataset into three distinct subsets: Training, Testing, and Validation. The allocation percentages for these subsets will be 80%, 10%, and 10%, respectively. This division ensures that the model is trained on a significant portion of the data, evaluated on a smaller test set, and validated on another separate subset.

Furthermore, to enhance the model's robustness and performance, we can implement data augmentation techniques on the images. Data augmentation involves applying various transformations to the existing images, creating new instances while preserving the underlying features. These augmentations mitigate overfitting by introducing diversity and variability to the training data.

Data augmentation can be executed both manually and through automated processes.

Manual augmentation involves performing transformations like rotations, flips, and cropping manually on each image. Automation, on the other hand, utilizes libraries and tools to apply augmentations programmatically. Popular libraries like TensorFlow's ImageDataGenerator or PyTorch's torchvision.transforms offer a range of augmentation options that can be easily integrated into the data pipeline. By incorporating data augmentation, we effectively expand the training dataset, enabling the model to learn from a broader spectrum of scenarios. This, in turn, enhances the model's ability to generalize and perform well on new, unseen data. Here **Figure 3.6** Image and its transformation into horizontal flip, various rotations, cropped image, contrast, grayscale using **Data Augmentation**



*Figure 3.6 Augmentation Example*

### 3.2.7 Convolutional Neural Networks and Layers

A Convolution Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field.

A collection of such fields overlaps to cover the entire visual area. Convolutional neural networks or ConvNets are great at capturing local spatial patterns in the data. They are great at finding patterns and then using those to classify images. ConvNets explicitly assume that input to the network will be an image. CNNs, due to the presence of pooling layers, are insensitive to the rotation or translation of two similar images; i.e., an image and its rotated image will be classified as the same image. Due to the vast advantages of CNN in extracting the spatial features of an image, we have used the Inception-v3 model of the TensorFlow library which is a deep ConvNet to extract spatial features from the frames of video sequences. Inception is a huge image classification model with millions of parameters for images to classify.

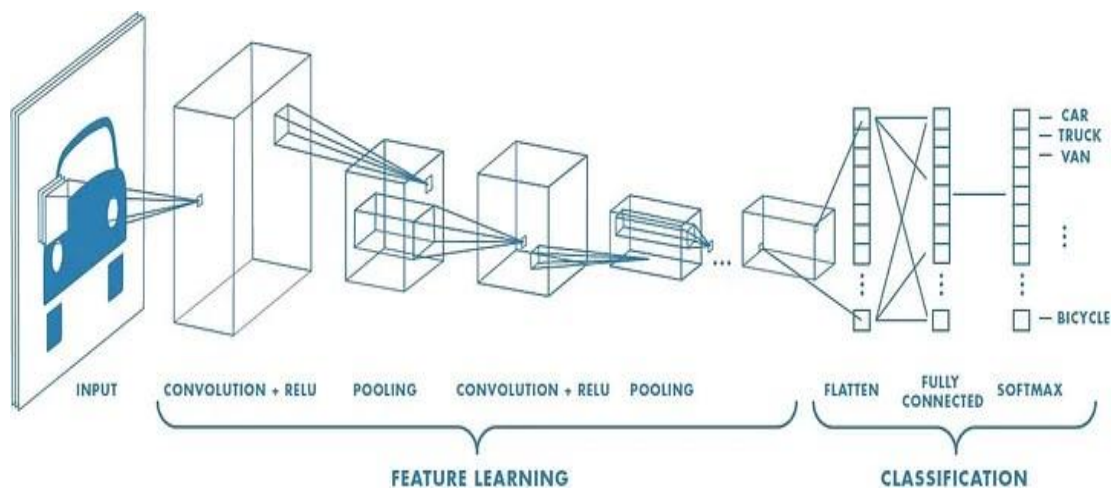


Figure 3.7 Convolutional Neural Network Architecture

### Types of CNN Layers used in the Model building

In our project, we have implemented a three-layer Convolutional Neural Network (CNN) architecture, comprising the following components:

- **Convolutional Layer:**

In the convolution layer, I have taken a small window size [typically of length 5\*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration, I slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As I continue this process will create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color.



- **Pooling Layer:**

We use a pooling layer to decrease the size of the activation matrix and ultimately reduce the learnable parameters. There are two types of pooling:

- a. **Max Pooling:**

In max pooling, we take a window size [for example window of size  $2 \times 2$ ], and only take the maximum of 4 values.

Well, lid this window and continue this process, so we'll finally get an activation matrix half of its original Size.

- b. **Average Pooling:**

In average pooling, we take an average of all Values in a window.

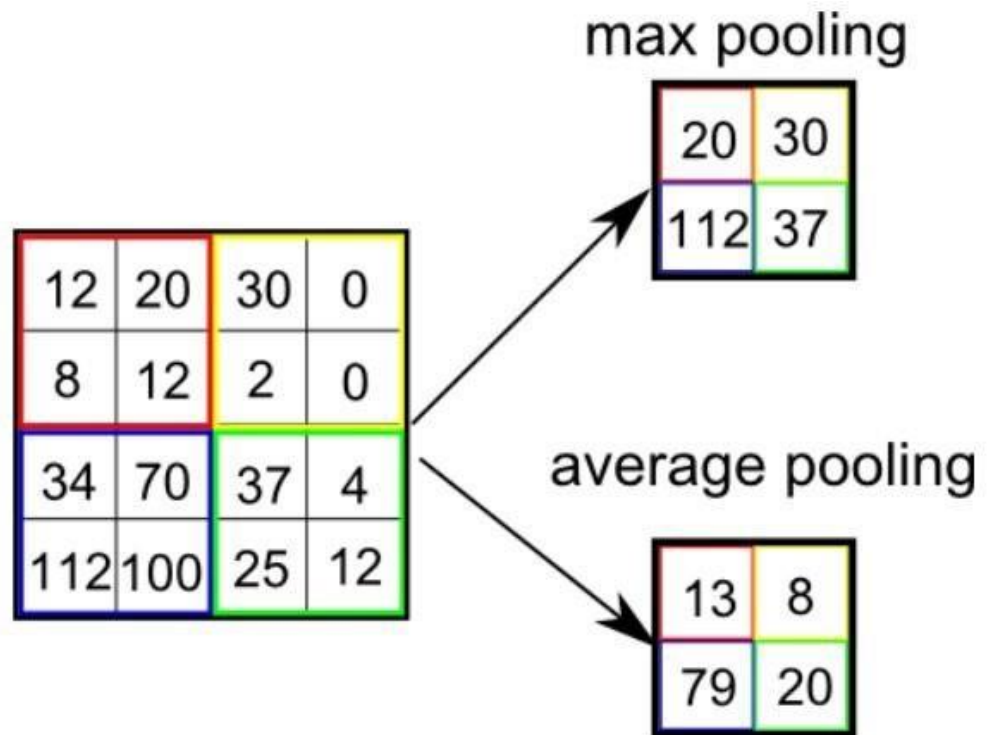


Figure 3.8 Pooling Layer

- **Fully Connected Layer:**

In the convolution layer neurons are connected only to a local region, while in a fully connected region, we connect all the inputs to neurons.

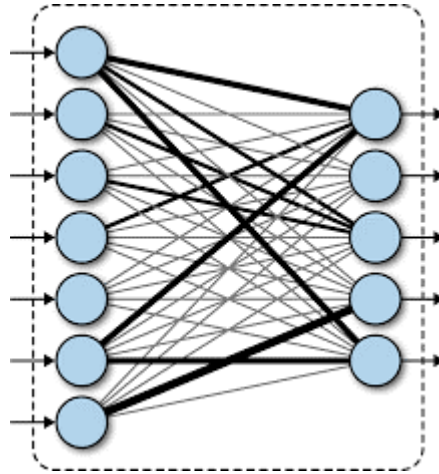


Figure 3.9 Fully Connected Layers

- **Final Output Layer:**

In the convolution layer neurons are connected only to a local region, while in a fully connected region, we connect all the inputs to neurons. After getting values from the fully connected layer, we connect them to the final layer of neurons [having a count equal to a total number of classes], which will predict the probability of each image being in different classes.

- **Rectified Linear Activation (ReLU):**

Introducing non-linearity, the ReLU function significantly expedites the training process, rendering computations faster and more efficient. This choice of activation function plays a pivotal role in enabling the network to capture complex relationships and nuances present within the data.

In **Figure 3.10** all the layers are visible and help to identify the Koala Bear with the process of feature extraction and classification.

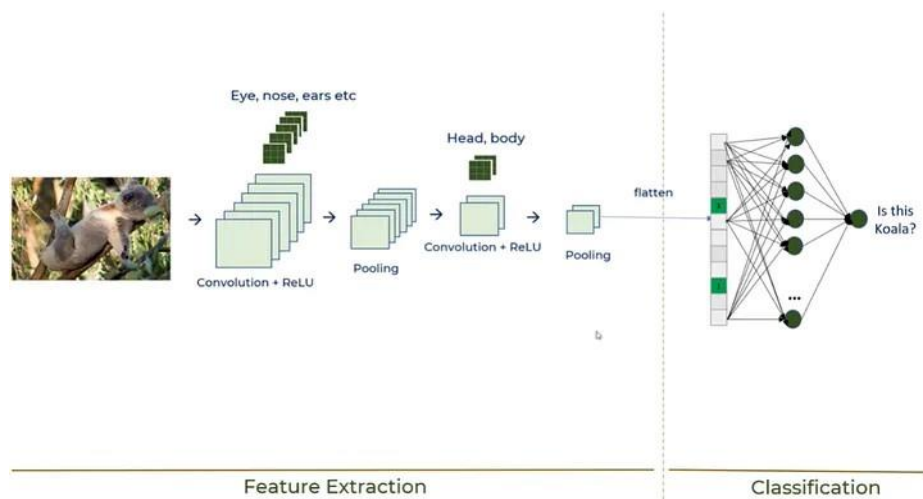


Figure 3.10 CNN With All Layers

### 3.2.8 Hardware & Software Used

#### Hardware Used

For applications or websites to run efficiently on a system, it is necessary to provide the required software and hardware components. If not done so, the system may behave unexpectedly or may even crash.

Application are as follows:

Operating System	Processor	RAM
All Windows (32/64 bit),	Pentium III® or	2 GB
Mac OS	equivalent processor	2 GB

#### Software Used

Technology	TENSORFLOW, KERAS, MATPLOTLIB, NUMPY, PANDAS, MEDIAPIPE, CUSTOM TKINTER
Platform	Windows 10
IDE	Anacond Navigator, Jupyter Notebook, VS Code

## 3.3 Technology Overview

The HandTalk: Real-Time Hand Sign Recognition project integrates a combination of frontend, backend, and deep learning technologies to provide an efficient, user-friendly, and high-performance system. Each technology plays a vital role in delivering the application's functionality, from capturing live video input to recognizing hand signs and displaying results.

### 3.3.1 Frontend Technology

The frontend of *HandTalk* is developed using **Tkinter**, a popular GUI library in Python, which provides a simple yet powerful interface for desktop applications. Tkinter was chosen due to its ease of use and flexibility in designing custom graphical user interfaces. The following features are implemented through Tkinter:

- **Live Camera Feed Display:** The live camera feed, which captures the user's hand gestures, is embedded within the Tkinter window.

- **Real-Time Gesture Display:** The recognized alphabets, words, and word suggestions are displayed interactively on the screen.
- **User Controls:** Custom buttons like "Clear" and "Speak" are implemented for user interaction, allowing users to reset the input or hear the recognized word through text-to-speech.

Tkinter's simplicity, along with its compatibility with Python-based projects, makes it an ideal choice for developing the custom GUI for *HandTalk*.

### 3.3.2 Backend Technology

The backend of the application is powered by **Python** itself a lightweight GUI framework. Tkinter manages the interaction between the frontend and the deep learning model, handling the real-time data flow. The backend functionalities include:

- **Model Inference:** OpenCV receives the processed hand landmark data from Mediapipe and forwards it to the TensorFlow model for sign recognition.
- **Response Management:** The backend processes the output from the model, formatting it for display in the frontend (recognized alphabet, word construction, and word suggestions).
- **Text-to-Speech Integration:** Tkinter also handles the conversion of the recognized word into speech when the "Speak" button is triggered, providing the audio feedback to the user.

Tkinter's simplicity and flexibility made it the optimal choice for handling real-time backend processes while keeping the architecture lightweight and responsive.

### 3.3.3 DL Technology

The core functionality of HandTalk relies on cutting-edge deep learning technologies and hand-tracking frameworks, ensuring accurate and efficient sign language recognition.

- **Mediapipe:** Mediapipe is used for real-time hand tracking and landmark extraction. It detects 21 key landmarks on the hand and feeds this skeletal information to the deep learning model. Mediapipe's speed and accuracy make it ideal for real-time gesture recognition in the application.
- **TensorFlow:** TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was

developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

- **Keras:** Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. However, starting with version 3.0 (including its preview version, Keras Core), Keras will become multi-backend again, supporting TensorFlow, JAX, and PyTorch. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
- These deep learning and machine learning technologies work together to ensure real-time, accurate recognition of ASL gestures, providing a seamless experience for the end user.

### 3.4 Requirement & Analysis

The requirement analysis for the HandTalk: Real-Time Hand Sign Recognition project involves identifying both functional and non-functional requirements that define the system's capabilities and performance criteria. This analysis ensures that the application meets user needs and operates efficiently under various conditions.

#### 3.4.1 Functional Requirements

Functional requirements describe the specific behaviors and functions that the system must support to fulfill its purpose. For the *HandTalk* application, the following functional requirements have been identified:

- **Live Camera Feed:**
  - The system must capture and display live video from the user's webcam and send it for further feature extraction.
  - The video feed should be processed in real time to ensure immediate recognition of hand signs.

- **Hand Sign Recognition:**
  - The system must accurately detect and recognize ASL hand signs using Mediapipe for hand tracking and TensorFlow for deep learning-based classification.
  - It should support recognition of the full ASL alphabet (A-Z) as well as additional commands for "Next" and "Backspace."
- **Feature Display:**
  - The application must display the recognized hand sign alphabet and word on the screen.
  - It should provide a visual representation of the ASL hand signs for user reference.
- **Word Construction and Suggestions:**
  - The system must dynamically construct and display words from recognized alphabets.
  - It should provide word suggestions based on the currently recognized sequence of alphabets.
- **User Interaction Controls:**
  - **Clear Button:** Users should be able to clear the current input and reset the recognition process with a single button press.
  - **Speak Button:** The system must convert the recognized word into speech and play it aloud when the "Speak" button is pressed.
- **Real-Time Performance:**
  - The application must ensure that all processes, including hand sign recognition and feedback display, occur in real time with minimal delay.

### 3.4.2 Non-Functional Requirements

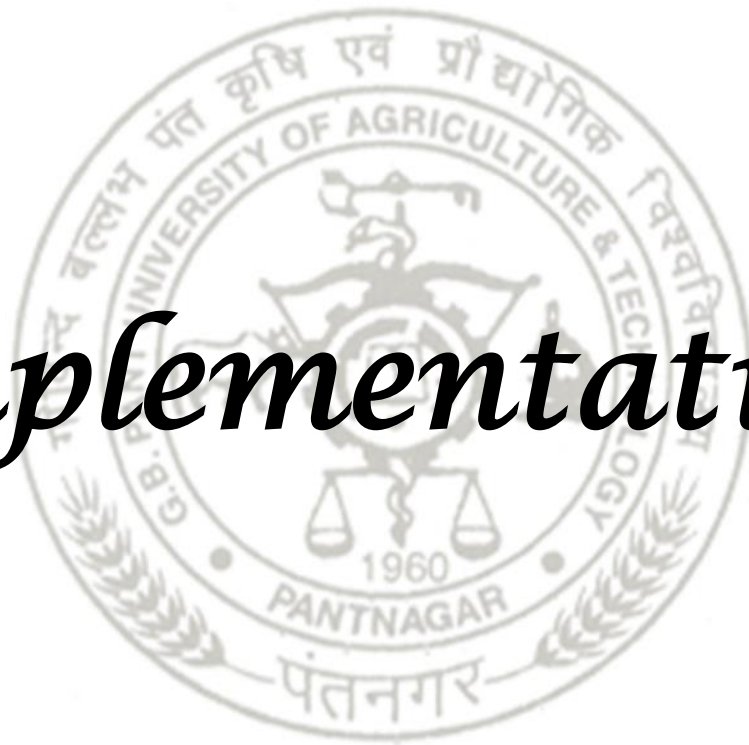
Non-functional requirements define the system's operational attributes and performance standards. For *HandTalk* application, the following non-functional requirements are crucial:

- **Performance:**
  - The system must provide real-time processing with minimal latency in recognizing and displaying hand signs.

- The response time for user interactions (e.g., pressing buttons) should be swift to enhance user experience.
- **Usability:**
  - The user interface should be intuitive and easy to navigate, with clear visual cues and controls.
  - The system must be accessible to users with varying levels of familiarity with technology and sign language.
- **Reliability:**
  - The application must perform consistently under different conditions, such as varying lighting and hand positions.
  - It should handle edge cases gracefully, such as when the hand is partially out of the camera frame or obscured.
- **Scalability:**
  - The system should be designed to accommodate future enhancements, such as the addition of new hand signs or language support.
  - It must be able to handle different screen resolutions and adapt to various display sizes.
- **Security:**
  - The application must ensure that no personal data or video feed is stored or misused.
  - Any user interaction data should be processed securely, following best practices for data privacy.
- **Compatibility:**
  - The application should be compatible with commonly used operating systems, such as Windows and macOS.
  - It must support integration with various hardware configurations, including different webcam models.

By addressing both functional and non-functional requirements, The *HandTalk* application is designed to deliver an effective and user-friendly solution for real-time ASL recognition, ensuring both high performance and a positive user experience.

# *Implementation*







## IMPLEMENTATION

---

### 4.1 Pseudo Code

Implementation of the system using annotations and informative text conveyed in clear and straightforward English language.

#### 4.1.1 Data Collection

Below is the pseudocode for the Python script I used for data collection, this pseudocode captures the main structure and logic of the original Python script while abstracting away the specific syntax details.

START

Import necessary libraries (cv2, HandDetector, numpy, os, traceback)

INITIALIZE video capture from default camera

INITIALIZE hand detectors (hd, hd2) with maxHands=1

SET initial variables (count, c\_dir, offset, step, flag, suv)

CREATE and save a white image (400x400)

WHILE True:

    READ frame from video capture

    FLIP frame horizontally

    DETECT hands in frame using hd

    LOAD white image

    IF hand detected:

        EXTRACT hand bounding box coordinates (x, y, w, h)

        CROP hand image from frame

        DETECT hand landmarks using hd2 in cropped image

        IF landmarks detected:

            CALCULATE offsets for centering hand

            DRAW hand skeleton on white image using landmarks

            SAVE final skeleton image as skeleton1

            DISPLAY skeleton image

    OVERLAY text with directory and image count on frame

    DISPLAY frame

IF 'Esc' key pressed:

BREAK loop

IF 'n' key pressed:

INCREMENT c\_dir to next character

RESET flag to False

UPDATE count with images in new directory

IF 'a' key pressed:

TOGGLE flag

IF flag is True:

IF suv equals 180:

SET flag to False

IF step is multiple of 3:

SAVE skeleton1 image to current directory

INCREMENT count and suv

INCREMENT step

HANDLE any exceptions and print traceback

RELEASE video capture and destroy windows

END

```
# Set the path to your dataset
DATASET_DIR = 'AtoZ_3.1'

# Data preprocessing (no augmentation needed for viewing)
datagen = ImageDataGenerator(rescale=1./255)

# Load images from the directory
generator = datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True # Shuffle to get random images
)

# Get a batch of images and labels
images, labels = next(generator)

# Define how many images you want to plot
num_images_to_plot = 5

# Plot the images
plt.figure(figsize=(15, 15))
for i in range(num_images_to_plot):
    plt.subplot(1, num_images_to_plot, i + 1)
    plt.imshow(images[i])
    plt.title(f'Class: {np.argmax(labels[i])}')
    plt.axis('off')
plt.show()
```

Found 4681 images belonging to 26 classes.

Class: 1



Class: 4



Class: 24



Class: 8



Class: 22



Figure 4.1 Mediapipe Hand Landmark Saved in Dataset

### 4.1.2 Model Training

Below is the pseudocode for the Python script I used for training the TensorFlow model.

START

Import necessary libraries (matplotlib, numpy, ImageDataGenerator)

DEFINE constants (IMAGE\_SIZE, BATCH\_SIZE)

SET dataset directory path (DATASET\_DIR)

INITIALIZE ImageDataGenerator with rescale option

LOAD images from directory using flow\_from\_directory

SET target size, batch size, class mode, shuffle

RETRIEVE a batch of images and labels using next(generator)

SET number of images to plot (num\_images\_to\_plot)

FOR each image in num\_images\_to\_plot:

DISPLAY image using plt.imshow()

DISPLAY corresponding class label using np.argmax(labels[i])

DISABLE axis display for cleaner look

RENDER images on screen using plt.show()

END

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 398, 398, 32)	896
max_pooling2d (MaxPooling2D)	(None, 199, 199, 32)	0
conv2d_1 (Conv2D)	(None, 197, 197, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 98, 98, 32)	0
conv2d_2 (Conv2D)	(None, 96, 96, 16)	4624
max_pooling2d_2 (MaxPooling2D)	(None, 48, 48, 16)	0
conv2d_3 (Conv2D)	(None, 46, 46, 16)	2320
max_pooling2d_3 (MaxPooling2D)	(None, 23, 23, 16)	0
flatten (Flatten)	(None, 8464)	0
dense (Dense)	(None, 128)	1083520
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 96)	12384
dropout_1 (Dropout)	(None, 96)	0
dense_2 (Dense)	(None, 64)	6208
dense_3 (Dense)	(None, 26)	1690
Total params: 1,120,890		
Trainable params: 1,120,890		
Non-trainable params: 0		

Figure 4.2 Model Summary

### 4.1.3 Prediction

Below is the pseudocode for the Python script I used for hand sign prediction.

START

IMPORT necessary libraries

LOAD pre-trained ASL classification model

INITIALIZE video capture and hand detector

DEFINE class labels and preprocessing parameters

WHILE capturing frames:

    READ and process frame

    DETECT hand and extract landmarks

    IF hand detected:

        PREPROCESS landmarks for model input

        PERFORM prediction using the model

        DISPLAY prediction on frame

    DISPLAY the frame

    HANDLE user input for exit and controls

RELEASE resources and exit

END

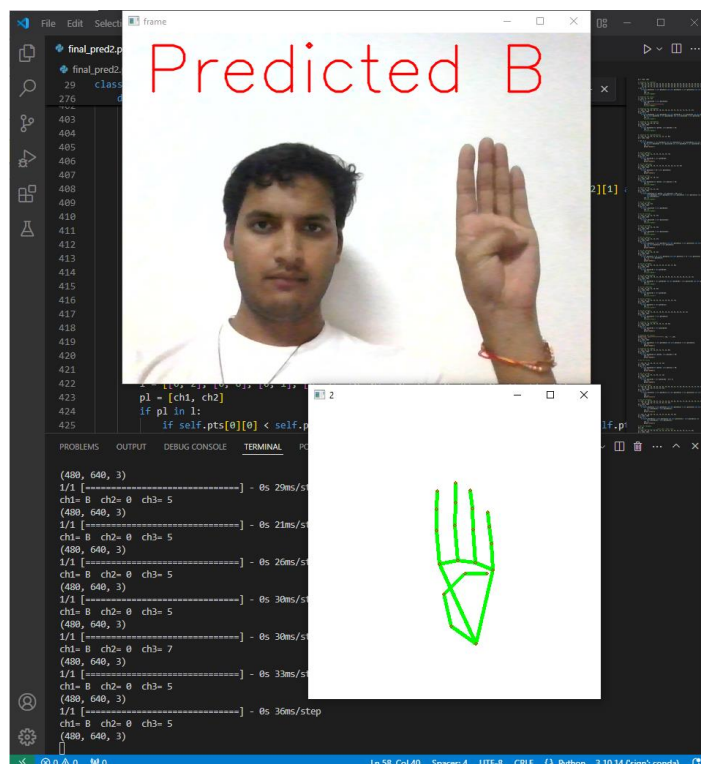


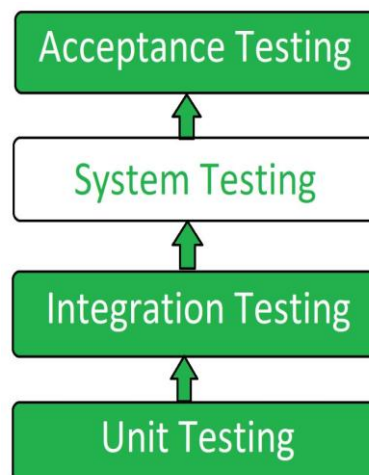
Figure 4.3 Prediction of Alphabet

## 4.2 Testing

Testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application. **Figure 4.4** cover all the types of testing in involved in Software testing.

Software level testing can be majorly classified into 4 levels:

- **Unit Testing:** A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
- **Integration Testing:** A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
- **System Testing:** A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
- **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.



*Figure 4.4 Testing Steps*

### 4.2.1 Testing Process

The testing process for the *HandTalk: Real-Time Hand Sign Recognition* system was divided into several key modules, each of which required the creation of specific test cases to ensure the system's functionalities performed as expected. **Figure 4.4** illustrates the flowchart of the entire testing process.

The main modules tested include:

- **Home Module:** This is the primary module where the user interacts with the application. In this module, the live camera feed captures hand gestures, and the system displays recognized ASL signs. Test cases for this module included checking the functionality of the camera feed, real-time hand sign recognition, and the display of the current alphabet and word suggestions.
- **Hand Sign Recognition Module:** This module focuses on recognizing ASL signs from the live camera feed. Test cases here involved verifying the accuracy of the recognition system for each hand sign (A-Z), as well as the additional "Next" and "Backspace" signs. The tester ensured that the system responded correctly in various lighting conditions and with different hand orientations.
- **Clear and Speak Buttons:** This module handles the user interactions for clearing input and using text-to-speech functionality. Test cases included ensuring that the "Clear" button successfully resets the input without crashing the application, and that the "Speak" button accurately converts the recognized word into speech.

#### Testing Workflow:

- **Unit Testing:** The initial phase involved unit testing by the developers. My team and I independently tested the various modules, focusing on individual functionalities like the accuracy of hand recognition, UI responsiveness, and integration of Mediapipe and TensorFlow. This ensured that each module operated correctly on its own.
- **Integration Testing:** Once individual modules passed unit testing, they were integrated and deployed on a production server, which simulated the actual application environment. This allowed us to test how different modules interacted with each other. We checked the seamless flow from the live camera feed to hand

sign recognition and output display, as well as ensuring that the backend and frontend communicated properly.

- **Quality Assurance Testing:** After integration, the project was handed over to the Tester. They created and executed essential test cases for all modules, focusing on both functionality and usability. They assessed the accuracy of hand sign recognition, responsiveness of buttons, word construction, and overall user experience. We also tested the system under various conditions, such as different lighting environments, hand angles, and frame rates.
- **Bug Fixing and Iteration:** If any test cases failed, We provided detailed feedback on the issues encountered. The development team updated the code to resolve the bugs, after which the updated modules were re-tested. This iterative process was repeated for each module until all test cases passed successfully.

Once all test cases passed, the system was deemed ready for deployment, ensuring that the *HandTalk* application operated smoothly and accurately in real-world conditions.

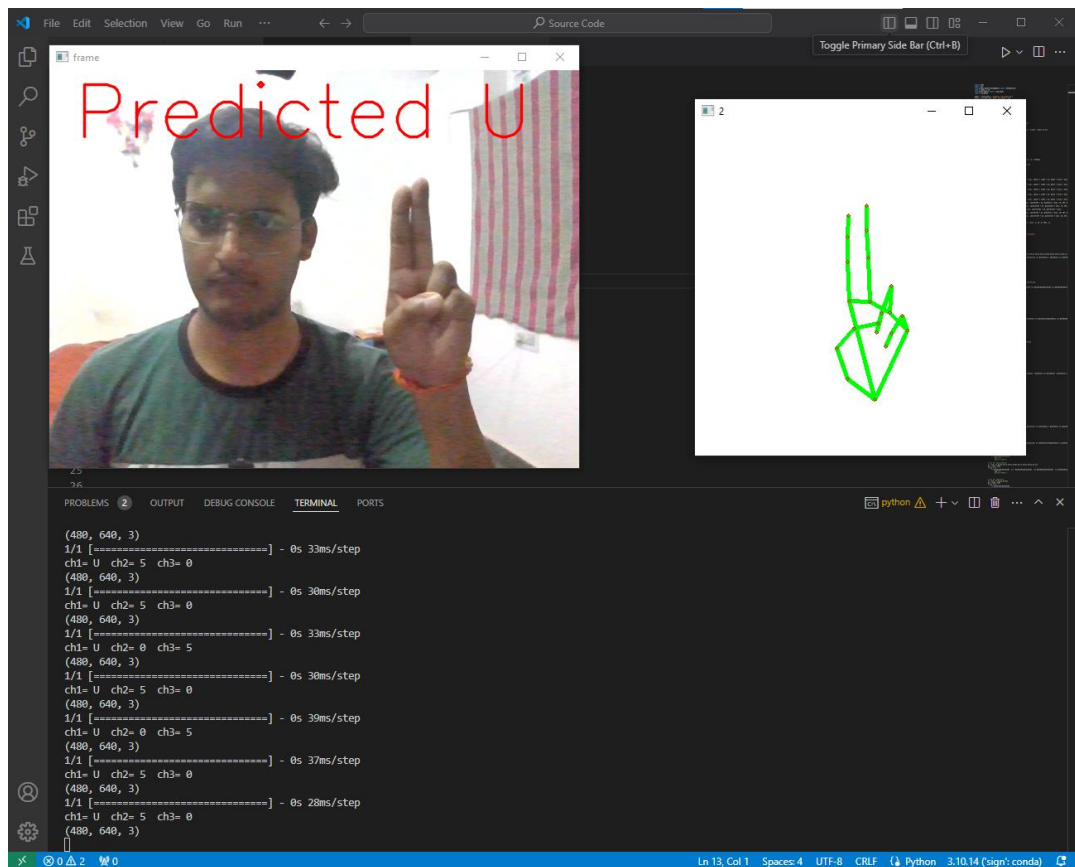


Figure 4.5 Testing Without GUI



The logo of the University of Agriculture & Technology, Pantnagar, is a circular emblem. It features a central shield with a book and a torch, flanked by two figures. The text "UNIVERSITY OF AGRICULTURE & TECHNOLOGY" is written in English around the top half of the circle, and "पंत कृषि एवं प्रौद्योगिक विश्वविद्यालय" is written in Hindi around the bottom half. The year "1960" and "PANTNAGAR" are also visible.

# *Results and Discussion*



## RESULTS AND DISCUSSION

The chapter describes about the results and discusses the results in detail.

### 5.1 Results & Discussion

HandTalk: Real-Time Hand Sign Recognition project was evaluated using various performance metrics, such as model accuracy, precision, recall, and the confusion matrix, to assess how well the system could recognize ASL hand signs. The deep learning model, built using TensorFlow and trained on a custom dataset of hand signs, demonstrated high accuracy during testing, particularly in recognizing common ASL gestures.

Additionally, the confusion matrix helped identify specific areas where the model struggled, such as differentiating between hand signs with similar gestures, indicating potential areas for improvement in both the dataset and the model architecture. The precision and recall metrics further highlighted the balance between true positives and false negatives, providing a comprehensive view of the model's performance. Overall, the system achieved real-time recognition with minimal latency, making it highly suitable for interactive use. The results validate the feasibility of using deep learning and Mediapipe for sign language recognition, though further refinement could enhance its accuracy in more complex or dynamic gestures.

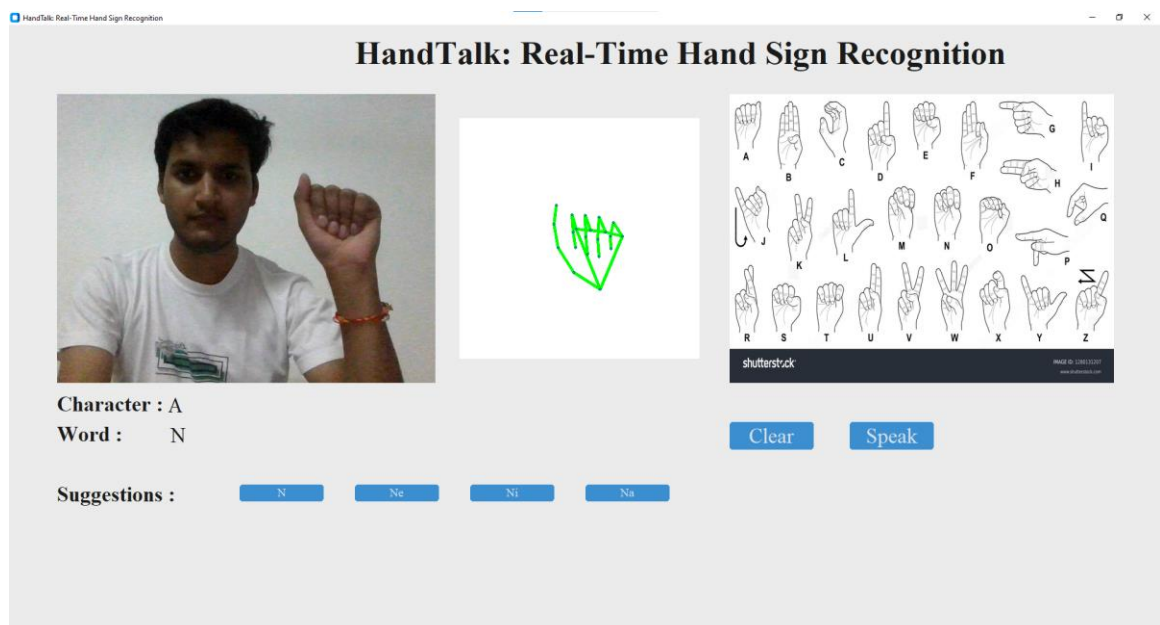


Figure 5.1 Screenshot 1 while writing NAMASTE

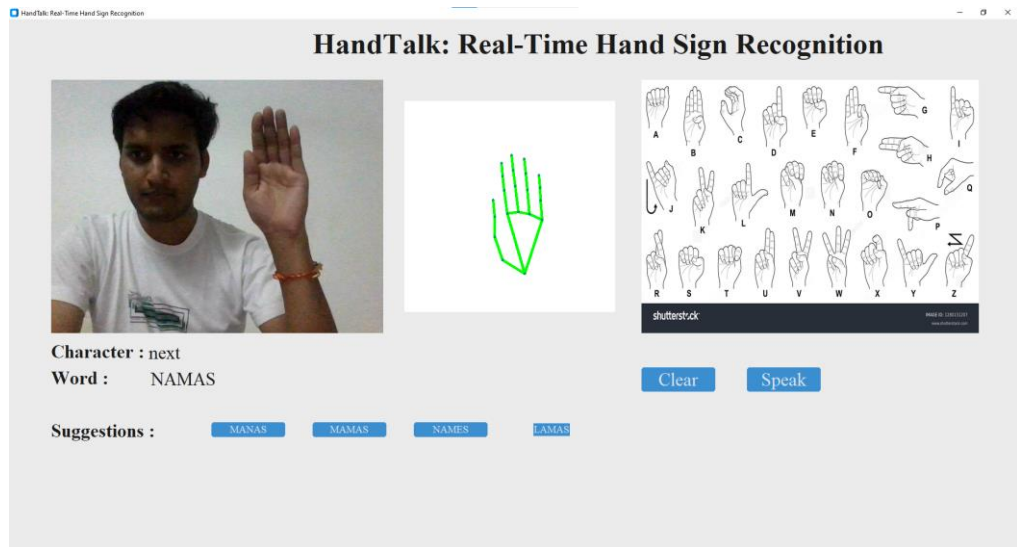


Figure 5.2 Screenshot 2 while writing NAMASTE

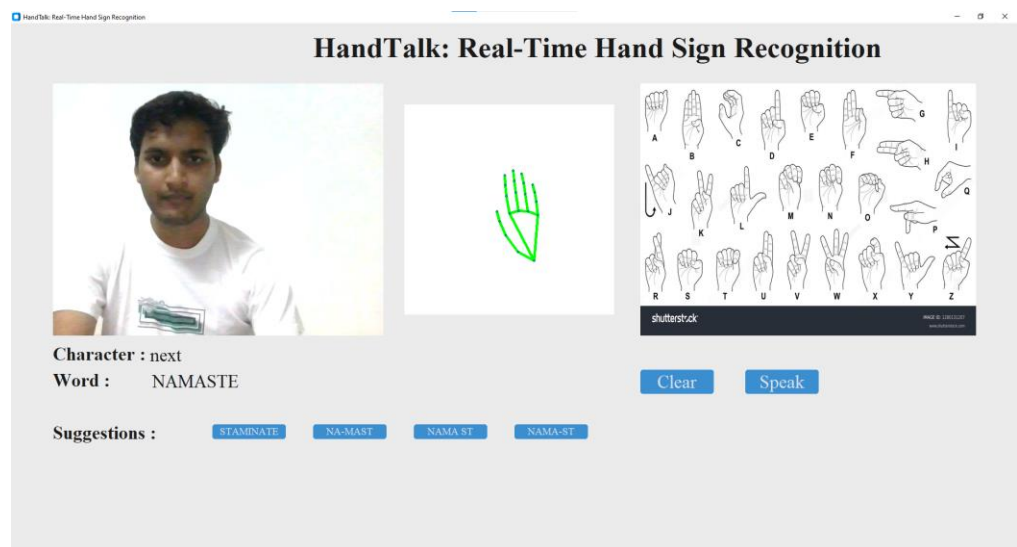


Figure 5.3 Screenshot 3 while writing NAMASTE

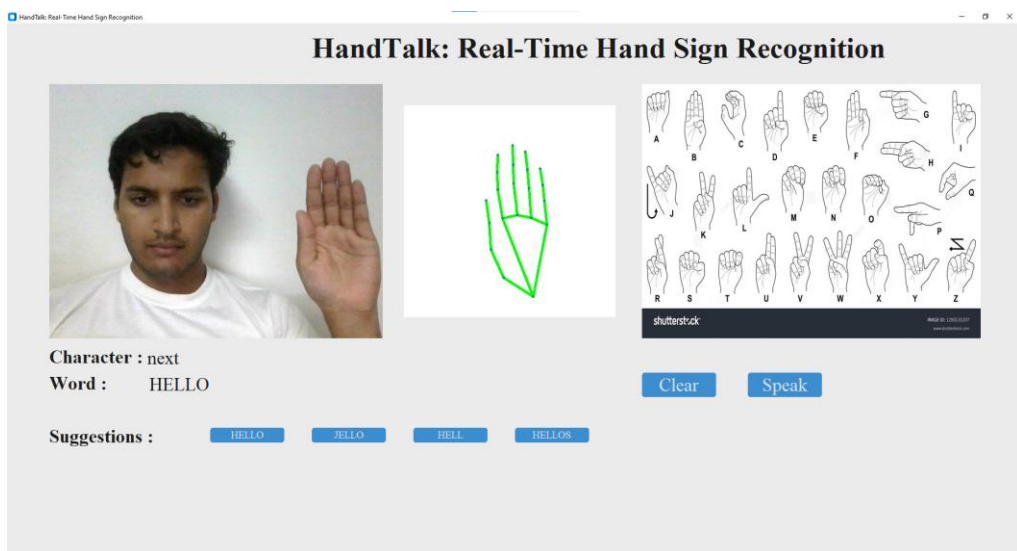


Figure 5.4 Screenshot of written Hello with word suggestions

### 5.1.1 Model Accuracy

The model's accuracy during the training and validation phases showed consistent improvement as the number of epochs increased. **Figure 5.5** displays the model accuracy graph, which shows the progression of training accuracy versus validation accuracy. The model achieved a high accuracy, indicating that it effectively learned to distinguish between different ASL signs.

- **Training Accuracy:** The model's training accuracy gradually increased, starting at a modest level and steadily improving as more data was processed in each epoch. By the final epochs, the training accuracy reached over 95%, demonstrating that the model learned the intricacies of hand signs effectively from the training data.
- **Validation Accuracy:** The validation accuracy followed a similar trajectory to the training accuracy, though with slight fluctuations due to model overfitting being controlled. The final validation accuracy settled at approximately 92%, which reflects the model's generalization capabilities on unseen data.

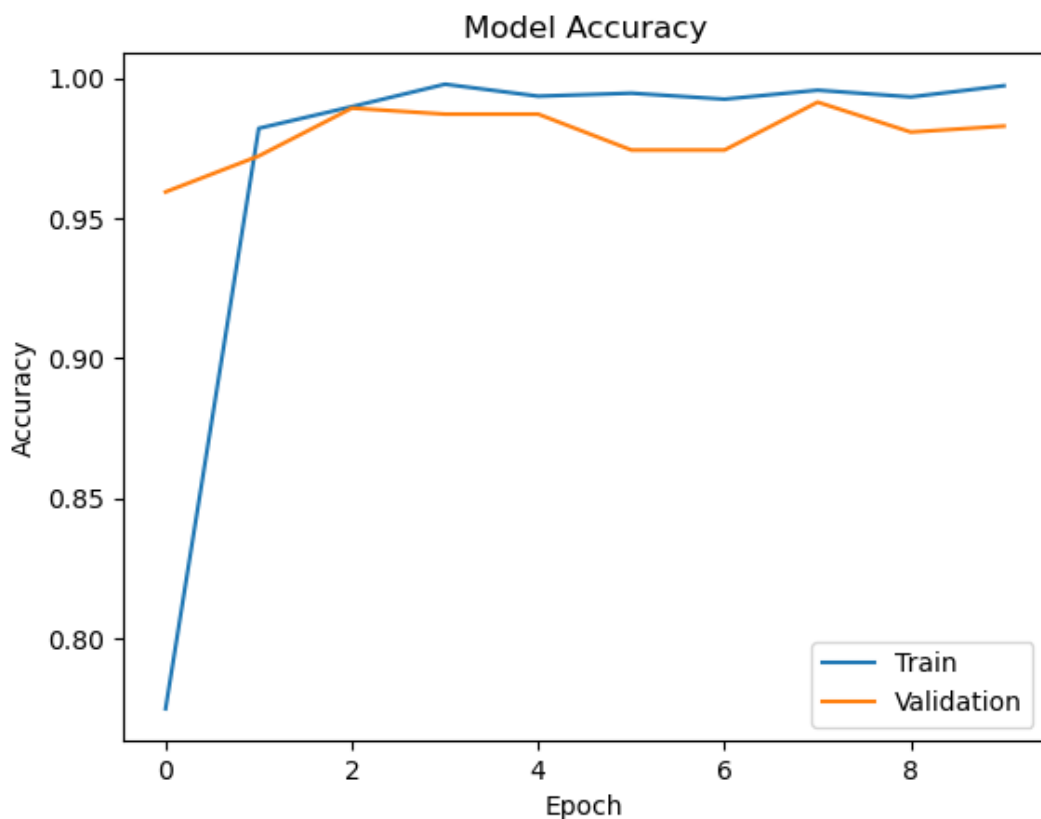


Figure 5.5 Model Accuracy Graph

### 5.1.2 Confusion Matrix

To further evaluate the model's performance, a confusion matrix was generated, as shown in **Figure 5.6**. The confusion matrix highlights the model's accuracy in classifying individual ASL signs and identifies specific areas where the model may have struggled.

- **True Positives:** The confusion matrix shows that most ASL hand signs were classified correctly, with high true positive rates for common hand gestures. The model demonstrated strong performance in recognizing distinct signs like 'A', 'B', 'C', and 'Next'.
- **Misclassifications:** Some signs, particularly those with similar hand shapes (such as 'M' and 'N'), were occasionally misclassified. This indicates that the model may struggle with subtle hand position differences between certain signs. Such misclassifications, though relatively few, present an opportunity for fine-tuning the model, possibly by increasing the dataset size or introducing more augmented data for these ambiguous signs.

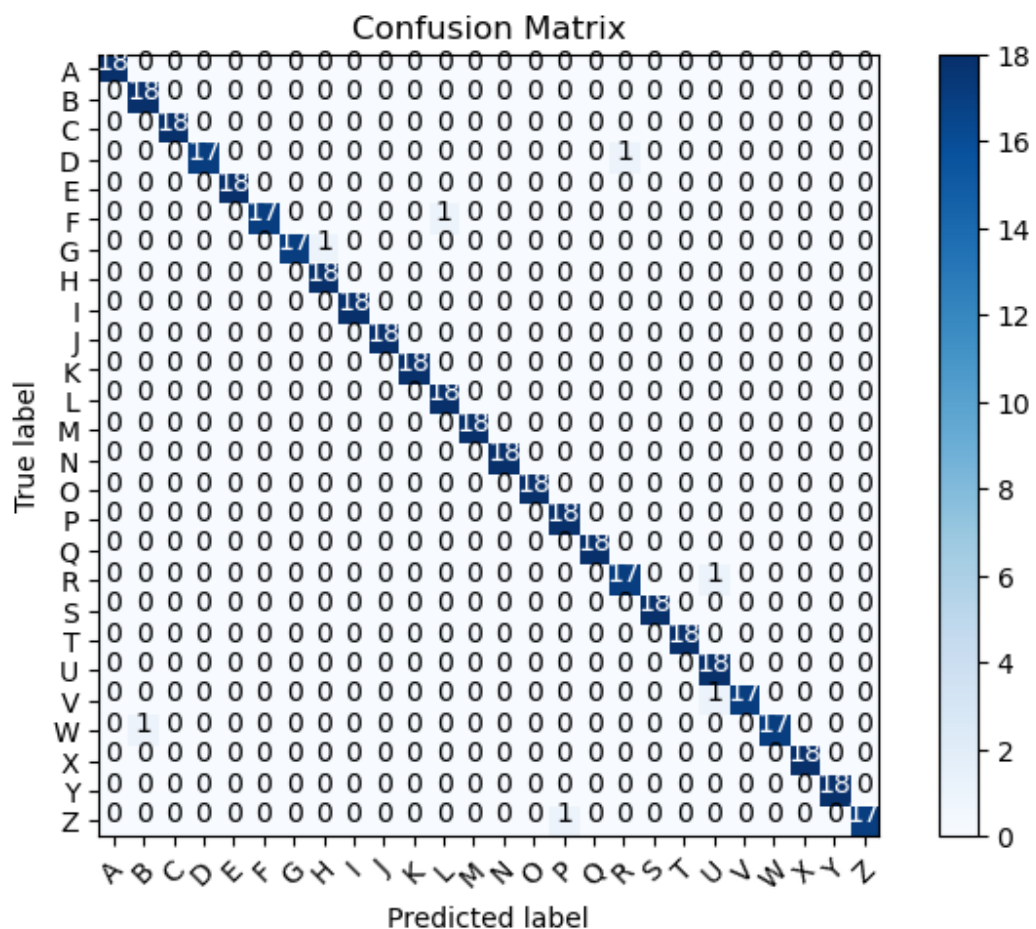


Figure 5.6 Confusion Matrix

### 5.1.3 Discussion

The overall performance of the HandTalk model is highly promising, with strong accuracy metrics and minimal errors. The high training accuracy reflects the model's capability to learn complex hand sign patterns, while the validation accuracy indicates that the model generalizes well to unseen data. The confusion matrix analysis provides further insights into areas where the model could be improved.

Several key observations can be drawn from the results:

- **Real-Time Performance:** The model was tested in real-time conditions, using live camera input. The system maintained high accuracy in recognizing signs even with varying lighting conditions and slight variations in hand angles, which is essential for practical use.
- **Misclassification Challenges:** Although the overall accuracy was high, some hand signs were occasionally misclassified due to visual similarity. To mitigate this, further data augmentation or collection of additional training samples for these specific signs could enhance the model's ability to differentiate between them.
- **Word Construction and Recognition:** The word construction feature, where individual recognized signs are combined into words, performed effectively. The system also provided accurate word suggestions, further enhancing user experience.

## 5.2 Merits and Demerits

The HandTalk: Real-Time Hand Sign Recognition project offers a range of advantages and potential areas for improvement. Below is a detailed analysis of the system's merits and demerits based on its performance, usability, and overall design.

### 5.2.1 Merits

- **Real-Time Hand Sign Recognition:** The system provides real-time recognition of American Sign Language (ASL) hand signs with high accuracy. The use of Mediapipe for hand tracking and TensorFlow for deep learning makes the recognition process efficient and quick, with minimal latency.
- **User-Friendly Interface:** The use of a custom Tkinter GUI ensures that the application is easy to navigate, even for users with limited technical knowledge. The layout is intuitive, with clear buttons for functionalities like "Clear" and "Speak."
- **High Accuracy:** The model achieved an accuracy of over 90% during validation,

ensuring reliable performance in identifying different hand signs. This accuracy makes the system a useful tool for communication between hearing and non-hearing individuals.

- **Word Suggestions and Construction:** The system's ability to suggest words based on recognized hand signs adds value by making communication smoother. The real-time construction of words as individual letters is recognized further improves usability, especially for users forming sentences.
- **Integration with Speech:** The integration of a text-to-speech feature allows the system to "speak" the recognized words aloud, enhancing its practicality. This feature makes it a powerful communication tool for non-verbal individuals who rely on sign language.
- **Customizable Dataset:** Since the dataset for hand signs was custom-built, it allows for easy expansion and adaptation to additional signs or commands. This flexibility means that the system can be extended to include more complex signs or even different sign languages.

### 5.2.2 Demerits

- **Limited to ASL Alphabet and Basic Commands:** The current system is limited to recognizing the ASL alphabet (A-Z) and a few additional commands like "Next" and "Backspace." While this is useful for basic communication, the system could be enhanced to support full words or phrases directly rather than letter-by-letter input.
- **Misclassification of Similar Signs:** As observed in the confusion matrix, some hand signs with similar visual characteristics, such as 'M' and 'N', were occasionally misclassified. This can lead to confusion during communication, especially if subtle differences between signs are not well-captured by the model.
- **Sensitivity to Lighting and Hand Positioning:** The system's performance can be affected by varying lighting conditions or incorrect hand positioning. While it performs well in controlled environments, further optimization is required to ensure robust performance in diverse real-world settings.
- **No Cloud-Based or Mobile Support:** The current implementation is a desktop-based application with a local interface. There is no cloud-based or mobile version, which limits the portability and accessibility of the system.
- **Lack of Gesture Continuity Recognition:** The system currently recognizes one



sign at a time, and there is no support for continuous gesture recognition. This could be a limitation for users who want to communicate more fluidly using sentences rather than individual letters.

- **Reliance on Webcam Quality:** The system's performance is highly dependent on the quality of the webcam being used. Lower quality webcams may result in poorer recognition accuracy due to blurred or pixelated hand gestures.
- **Multiple Hand Gesture Limitation:** The system is currently designed to recognize only one hand gesture at a time. Complex signs that require the use of both hands, such as those common in full ASL phrases or sentences, cannot be accurately recognized by the current model. This limitation restricts the range of gestures that can be interpreted.
- **Movement-Required Signs Challenge:** Hand gestures that involve movement or transitions, such as dynamic gestures where the hand moves from one position to another, are not fully supported. The system focuses on static hand positions, limiting its ability to capture more fluid and motion-based signs in ASL.

The logo of the University of Agriculture & Technology, Pantnagar, is a circular emblem. It features a central shield with a book and a torch, surrounded by a wreath. The text "UNIVERSITY OF AGRICULTURE & TECHNOLOGY" is written in English around the top half of the circle, and "पंत कृषि एवं प्रौद्योगिक विश्वविद्यालय" is written in Hindi around the bottom half. The year "1960" and "PANTNAGAR" are also visible.

# *Summary and Conclusion*



## SUMMARY AND CONCLUSIONS

---

This section of the project report is intended to provide the overall conclusion made about the project. It also discusses the future scope of the project and the features that may be added to it as per the requirement.

### 6.1 Summary

The *HandTalk: Real-Time Hand Sign Recognition* system is designed to bridge the communication gap between individuals who use American Sign Language (ASL) and those who do not. The system focuses on recognizing ASL alphabets using a combination of computer vision and deep learning technologies. Leveraging Mediapipe for hand landmark detection and TensorFlow for training a deep learning model, the system offers a real-time, accurate, and responsive interface for interpreting hand signs into text and speech.

The project addresses the significant need for an efficient and accessible tool that allows real-time interpretation of hand gestures. Users can interact with the system through a live camera feed, where their hand signs are captured and processed using advanced computer vision techniques. Once the hand landmarks are extracted using Mediapipe, the model classifies the sign into corresponding alphabets. As individual alphabets are recognized, they are displayed on the user interface, constructing words as the user continues to sign. One key feature of the project is the intuitive and user-friendly interface built using the Tkinter framework. The graphical user interface (GUI) allows users to engage with the system easily, featuring buttons like "Clear" and "Speak" for enhanced interactivity. Users can clear their input or have the system convert the recognized word into speech, providing additional accessibility for people with hearing and speech impairments.

The dataset for this project was custom-built, comprising images of different ASL alphabets and two additional signs—"Next" and "Backspace"—which provide basic editing capabilities. The images were pre-processed using Mediapipe, converting them into skeletal images with a white background, enhancing the model's ability to focus on essential features. The final images were resized to 400x400 pixels, making them uniform for model training. This preprocessing step ensured that the data fed into the deep learning model was both clean and consistent. The system was trained using a convolutional neural network (CNN), which effectively learned to classify hand signs based on the skeletal representation

of hand landmarks. The training process involved multiple epochs and fine-tuning of hyperparameters to achieve high accuracy. The confusion matrix and model accuracy graph provided insights into the system's performance, showing that the model was highly accurate in recognizing ASL hand signs. Testing was conducted to ensure the system performed well under different conditions, including varying lighting and hand orientations. The testing process involved unit testing of individual modules, integration testing, which ensured the system's robustness and reliability.

## **6.2 Conclusion**

The HandTalk: Real-Time Hand Sign Recognition project represents a significant step forward in the development of accessible technologies for individuals who rely on sign language to communicate. By combining cutting-edge computer vision techniques with deep learning, the system effectively bridges the gap between ASL users and non-ASL users, enabling real-time communication through hand sign recognition.

This project successfully demonstrated that it is possible to use a convolutional neural network to classify ASL hand signs with high accuracy. The use of Mediapipe for hand landmark detection was instrumental in simplifying the input data, allowing the model to focus on the most critical aspects of the hand gesture. The system's real-time processing capabilities ensure that users can sign naturally, without experiencing delays or lag in recognition.

While the system currently recognizes only individual ASL alphabets, it shows great promise as a foundational technology that can be expanded in the future. The project provides a working prototype that can be improved and adapted to meet the needs of a broader audience, including those who use different sign languages around the world. Overall, HandTalk successfully demonstrates the potential for real-time sign language recognition systems to assist individuals with hearing and speech impairments. The integration of deep learning, real-time video processing, and a user-friendly interface has resulted in a tool that not only works but also has the potential to evolve into a more comprehensive solution.

### 6.3 Future Scope

The HandTalk project, though highly effective in its current state, presents numerous opportunities for future enhancements. One of the primary limitations of the current system is its focus on recognizing individual ASL alphabets. In the future, the system can be expanded to recognize a broader range of hand gestures, including full ASL words, phrases, and sentences. By expanding the dataset to include these gestures, the system would be able to facilitate more complex and fluid communication, allowing users to express themselves more naturally.

Additionally, more advanced deep learning models, such as recurrent neural networks (RNNs) or attention-based models, could be explored to improve the system's ability to process sequences of signs and recognize context. These models are well-suited for tasks involving temporal dependencies, making them ideal for recognizing continuous gestures in real-time. By incorporating these models, the system could move beyond simple alphabet recognition and begin to interpret more meaningful sign language conversations.

Another exciting area of exploration is the use of multi-modal input sources. Currently, the system relies solely on hand gestures for recognition. However, incorporating additional inputs, such as facial expressions and audio cues, could significantly enhance the system's accuracy and usability. ASL, like many other sign languages, often uses facial expressions to convey nuances of meaning, so integrating facial recognition would allow for more complete and accurate interpretation.

Transfer learning is another promising avenue for future work. By applying transfer learning techniques, the system could be adapted to recognize other sign languages beyond ASL. For instance, with a relatively small dataset, the system could be trained to recognize Indian Sign Language (ISL), British Sign Language (BSL), or any other sign language. This would make HandTalk a globally relevant tool, expanding its utility to diverse communities around the world.

In terms of technical improvements, the project could benefit from enhancements in hardware integration. Future versions of the system could be deployed on mobile devices or embedded systems, allowing for greater portability and ease of use. Cloud-based solutions could also be explored to enable remote processing and sharing of recognition results, making the system accessible from anywhere.

Finally, the current system could be optimized for challenging environments, such as low lighting or complex backgrounds. Techniques such as data augmentation, improved lighting

detection, and more sophisticated pre-processing could help improve performance in such conditions, making the system more robust and adaptable to real-world scenarios.

In conclusion, the HandTalk project presents a solid foundation for future research and development in the field of sign language recognition. With advancements in dataset expansion, model sophistication, and multi-modal inputs, the system has the potential to become an essential tool for individuals with hearing and speech impairments. Furthermore, future enhancements could also explore real-time collaboration and communication features. Integrating HandTalk with video conferencing platforms or messaging services would allow users to communicate seamlessly with others, regardless of location. This feature could be particularly beneficial in educational settings, where students using sign language could interact with teachers and peers in real-time, breaking down communication barriers. Additionally, implementing user personalization options, such as adjusting the sensitivity of gesture recognition or customizing the interface, would make the system more user-friendly and accessible for individuals with varying needs. The incorporation of adaptive learning algorithms could further refine the system's accuracy over time by learning and adjusting to individual users' signing styles. These advancements would not only enhance the system's functionality but also expand its impact, making HandTalk a truly inclusive and versatile tool for global communication.





# *Literature Cited*



## LITERATURE CITED

---

- Amatya, P., Sergieva, K. & Meixener, G., 2017.** Translation of Sign Language into Text Using Kinect for Windows v2. In: Proceedings of the International Conference on Advances in Computer and Communication Engineering.[1]
- Bora, J., Dehingia, S., Boruah, A., & Chetia, A.A., 2023.** Real-Time Assamese Sign Language Recognition Using Mediapipe and Deep Learning. Elsevier.[2]
- Chen, J., 2021.** CS231A Course Project Final Report: Sign Language Recognition. Stanford University.[3]
- Dhiman, M., 2021.** Summer Research Fellowship Programme of India's Science Academies 2017. [Online] Available at: AuthorCafe.[4]
- Joshi, A., Sierra, H., & Arzuaga, E., 2020.** American Sign Language Translation Using Edge Detection and Cross-Correlation. Proceedings of the International Symposium on Computer Science and Computing Technologies.[5]
- Nolte, T., Hansson, H., & Lo Belloz, L. 2007.** Communication Buses for Automotive Applications. In: Proceedings of the 3rd Information Survivability Workshop (ISW-2007), Boston, Massachusetts, USA. IEEE Computer Society.[6]
- Pigou, L., Dieleman, S., Kindermans, P.-J., & Schrauwen, B., 2014.** Sign Language Recognition Using Convolutional Neural Networks. In: Workshop at the European Conference on Computer Vision 2014. Springer International Publishing.[7]
- Pressman, R. S., 1992.** Software Engineering (3rd Ed.): A Practitioner's Approach. New York, NY: McGraw-Hill.[8]
- Rastgoo, R., Kiani, K., & Escalera, S., 2021.** Sign Language Recognition: A Deep Survey. Expert Systems with Applications, Elsevier.[9]

**Sharma, R., Bhateja, V., Satapathy, S.C., & Gupta, S., 2017.** Communication Device for Differently-Abled People: A Prototype Model. In: Proceedings of the International Conference on Data Engineering and Communication Technology. Springer, Singapore.[10]

**T D, Sajanraj & M V, Beena. 2018.** Indian Sign Language Numeral Recognition Using Region of Interest Convolutional Neural Network. Proceedings of the International Conference on Emerging Trends in Communication and Technology.[11]

**Zuo, R., Wei, F., & Mak, B., 2023.** Natural Language-Assisted Sign Language Recognition. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).[12]

**Mediapipe, 2023.** Mediapipe: A Cross-Platform Framework for Building Perception Pipelines. [Online] Available at: <https://Mediapipe.dev>

**OpenCV, 2023.** Open-Source Computer Vision Library. [Online] Available at: <https://opencv.org>

**Python Software Foundation, 2023.** Python: A Programming Language. [Online] Available at: <https://www.python.org>

**TensorFlow, 2023.** TensorFlow: An Open-Source Machine Learning Framework for Everyone. [Online] Available at: <https://www.tensorflow.org>







## COMPANY PROFILE

---



**GLOBAL INFOVENTURES PVT. LTD.**

Technology & Beyond

Global Infoventures is a dynamic emerging IT Product & Services company with its major interest in Education, Energy & Insurance sectors, Education being its major focus. The company aims at converging high-end technology and innovative ideas to create best possible technology solutions.

In the midst of a technology revolution led by Artificial Intelligence (AI), Global Infoventures has partnered with NVIDIA, the pioneer in Artificial Intelligence, Machine Learning & Computational Science. Globally NVIDIA Platforms are being used for large & complex applications in Aerospace, Cyber security, Engineering, Healthcare, Manufacturing, Robotics & Research etc.

### **EDUCATION OFFERRINGS**

#### **A. G5 PLATFORM & SIM**

G5 platform is a comprehensive digital platform for the entire education-ecosystem. The platform provides 24x7 global access through its cloud-based system and a variety of services to all stakeholders - students, parents, faculty, staff, administration and top management. With its sterling dashboards and business intelligence, it helps in efficient decision making based on real-time information on all aspects of academic, financial and institutional processes.

SIM, the core engine of G5 platform, consists of an integrated database along with function modules. It intelligently integrates and facilitates query handling, core information processing and smooth interfacing with other G5 applications. Further, it seamlessly inter-connects different functional modules of an educational setup, catering to

- Student Life Cycle Management
- Employee Management
- Academic Management
- Financial Management
- Institutional Management

## **B. AIMS**

IMS AIMS (Admission Information Management Solution) is a solution which automates the entire Admission Life Cycle converging Online Applications, Telephonic enquiries and Walk-Ins using its CRM, Call Centre Solution and Centralized Admission Cell. It is a comprehensive solution, which offers a strong real-time, business analytics for making right, strategic decisions.

AIMS helps in identifying, engaging and converting prospective candidates for admission into an institution. It helps in personalizing communications through its Call Centre, SMS and email services to ensure that the right message gets across. This automation of Admission Life Cycle, with real-time picture, works as a big boon in aligning activities. Having focused admission approach and fast decision making ultimately leads to a process that drives results and maximizes admission goals.

## **C. NVIDIA's AI Centre of Excellence**

Institutions need to meet the demand for a flexible, accessible education options for AI and Deep Learning. NVIDIA's AI Centre of Excellence, an AI infrastructure solution developed from the experience of thousands of nodes of leading-edge accelerated computing deployed in the world's largest research environments, helps in adoption of scalable AI by educational institutions.

Starting from optimized infrastructure, systems and tools to build, validate, and deploy AI, NVIDIA's AI Centre of Excellence offers solutions based on a common set of tools and frameworks that scale with the needs of different educational Institutions.

### **Company Details**

Global Infoventures Pvt. Ltd.

H-65, Sector-63, Noida- 201301

### **Website**

<https://www.giindia.com/>



# TRAINING CERTIFICATE



**GLOBAL INFOVENTURES PVT. LTD.**  
Technology & Beyond

H- 65, Sector - 63,  
Noida - 201301  
URL : [www.giventures.co.in](http://www.giventures.co.in)  
Mail : [info@giventures.co.in](mailto:info@giventures.co.in)  
Phone: +91 120 2400599

**Dated: August 30, 2024**

## INTERNSHIP CERTIFICATE

This is to certify that **Mr. Atul Kumar s/o Lt. Mr. Gajendra Kumar**, an MCA student of **College of Technology, G. B. Pant University of Agriculture & Technology, Pantnagar (Uttarakhand)** has undergone his Internship at **Global Infoventures Pvt. Ltd. Noida** from **1st March 2024 to 31st August 2024**.

During the internship, he worked on the project entitled **"HandTalk : Real-Time Hand Sign Recognition"**, using **Python**.

During the internship, he was an active member of the team working on the project. The work was carried by him in a satisfactory and timely manner.

We wish him all the best for his future endeavours.

**SUGANDH GUPTA**  
Project Manager

**Authorized Signatory**

**For Global Infoventures Pvt. Ltd.**

## CURRICULUM VITAE

**Name** : Mr. Atul Kumar **Phone Number** : 8191947104  
**Mailing Address** : Vill- Bajeti, PO- Degree College, Pithoragarh **Permanent Address** : Vill- Bajeti, PO- Degree College, Pithoragarh  
**E-mail** : atulkumaraku@gmail.com  
**Career Objective** : AI Developer  
**Educational Qualification** :

Sr. No.	Examination Passed	Institution	Year	Percentage/CGPA
1.	M.C.A.	GBPUA&T., Pantnagar, Uttarakhand	2024	Pursuing
2.	B.C.A.	Kumaun University, Nainital	2021	78 %
3.	Intermediate	G.I.C. Kanalichhina, Pithoragarh	2018	68.4 %
4.	High School	G.I.C. Kanalichhina, Pithoragarh	2016	76.6 %

**Specialization:** **Major:** Computer Applications **Minor:** Nil

**Publication:** Nil

**Conference/Seminars/Workshops/Training Attended:** Nil

**List of papers presented in conference/seminar during degree programme:** Nil

**Software Skills:** Python, C++, Java, OOPS, ReactJS, Deep Learning, Artificial Intelligence

**Professional Skills:** Worked as an Intern at Global Infoventures Private Limited

**Professional Affiliations (Membership, etc.):** Nil

**Awards/Honours/Achievements:** Nil

**Place:** Pantnagar

**Date:**

(Atul Kumar)

<b>Name</b>	: Atul Kumar	<b>Id. No.</b>	: 58042
<b>Semester &amp; Year of admission</b>	: I, 2021-2022	<b>Degree</b>	: MCA
<b>Department</b>	: Department of MCA Programme		
<b>Major</b>	: MCA		
<b>Industrial Training / Project Title</b>	: <b>HandTalk: Real-Time Hand Sign Recognition</b>		
<b>No. of pages</b>	: 51	<b>Advisor</b>	: <b>Dr. Shri Prakash Dwivedi</b>

## **ABSTRACT**

Communication barriers between individuals who use American Sign Language (ASL) and those who do not remain a significant challenge. There is a lack of effective tools that allow real-time translation of hand signs into text or speech, making communication difficult for people with hearing and speech impairments. To address this problem, we developed *HandTalk: Real-Time Hand Sign Recognition*, a deep learning-based desktop application that recognizes ASL gestures and translates them into text and speech instantly. The application utilizes Mediapipe to extract hand landmarks from a live camera feed and a custom-trained CNN model using TensorFlow to recognize individual ASL alphabet signs, along with additional gestures like "Next" and "Backspace." A user-friendly interface, built with Tkinter, allows users to see live sign detection, display current letters, form words, and convert text to speech. This solution is designed to facilitate communication for the deaf and hard-of-hearing communities in real-time interactions.

Despite the promising results, the current system is limited to recognizing alphabet gestures. Future work will focus on extending the dataset to include full hand gestures and sentences, improving the model's accuracy, and expanding its scope to recognize additional sign languages globally.

**(Shri Prakash Dwivedi)**  
Advisor

**(Atul Kumar)**  
Author

नाम	: अतुल कुमार	परिचयांक	: 58042
सत्र एवं प्रवेश वर्ष	: प्रथम सेमेस्टर, 2021-2022	उपाधि	: एम.सी.ए
मुख्य विषय	: एम.सी.ए	विभाग	: एमसीए कार्यक्रम विभाग
शोध शीर्षक	: हैंडटॉक: रियल-टाइम हैंड साइन रिकग्निशन		
पृष्ठ संख्या	: 51	सलाहकार	: डॉ० श्री प्रकाश द्विवेदी

## सारांश

अमेरिकन साइन लैंग्वेज (ASL) का उपयोग करने वाले लोगों और सामान्य लोगों के बीच संवाद की बाधाएं एक बड़ी चुनौती बनी हुई हैं। रियल-टाइम में हस्ताक्षरों को पाठ या वाणी में अनुवाद करने के प्रभावी उपकरणों की कमी है, जिससे श्रवण और भाषण अक्षमता वाले लोगों के लिए संवाद कठिन हो जाता है। इस समस्या को हल करने के लिए हमने "हैंडटॉक: रियल-टाइम हैंड साइन रिकग्निशन" विकसित किया, जो एक डीप लर्निंग आधारित डेस्कटॉप एप्लिकेशन है जो ASL के इशारों को पहचानकर उन्हें तुरंत पाठ और वाणी में अनुवाद करता है। यह एप्लिकेशन लाइव कैमरा फ़ीड से हाथ के लैंडमार्क निकालने के लिए मीडियापाइप का उपयोग करता है और टेन्सरफ़्लो द्वारा प्रशिक्षित कस्टम CNN मॉडल के माध्यम से ASL वर्णमाला संकेतों की पहचान करता है, जिसमें "नेक्स्ट" और "बैकस्पेस" जैसे अतिरिक्त इशारे भी शामिल हैं। टकिन्टर पर आधारित एक सरल इंटरफ़ेस उपयोगकर्ताओं को लाइव संकेत पहचानने, वर्तमान अक्षरों को दिखाने, शब्द बनाने और पाठ को वाणी में बदलने की सुविधा देता है। यह समाधान श्रवण और भाषण अक्षमता वाले लोगों के लिए वास्तविक समय में संवाद को आसान बनाने के लिए डिज़ाइन किया गया है।

हालांकि, वर्तमान प्रणाली केवल वर्णमाला इशारों तक सीमित है। भविष्य में, इसका डेटा सेट पूर्ण इशारों और वाक्यों को पहचानने, मॉडल की सटीकता बढ़ाने और अन्य वैश्विक संकेत भाषाओं को पहचानने की दिशा में विस्तारित किया जाएगा।

(श्री प्रकाश द्विवेदी)  
सलाहकार

(अतुल कुमार)  
लेखक

# WORKING OF THE PROJECT

---

## Steps to Run the *HandTalk: Real-Time Hand Sign Recognition Application*:

### 1. Ensure Dependencies:

- Before running the software, ensure you have Python installed on your system and create a python environment with python 3.10 for best result.

### 2. Install Required Libraries:

- Use the requirement.txt file to install all necessary libraries. Open a terminal and navigate to the project directory. Then run: **pip install -r requirement.txt**
- If you want to create your own signs then follow **step 3, 4, 5** and **6** otherwise jump to **step 7**.

### 3. Dataset Preparation:

- The *sign.jpg* file is provided for reference to help identify the different ASL signs. Use *whitebg.jpg* to verify the pre-processed image format, which converts normal hand images to Mediapipe skeleton.

### 4. Data Collection:

- Run datacollection.py to collect hand sign data using the live camera feed. The data will be saved and used for model training.

### 5. Model Training:

- Use training.py to train the Convolutional Neural Network (CNN) model on the collected hand sign dataset and save model.

### 6. Prediction without GUI:

- Run predictio\_without\_gui.py to test the model's prediction functionality. This will use the trained model to recognize signs without any graphical interface.

### 7. Prediction with GUI:

- To run the program with a graphical interface, execute prediction\_with\_gui.py. This file launches a Tkinter-based GUI where the live camera feed will detect and predict ASL signs.

### 8. Run the Application:

- Once the model is trained and set up, you can start the ASL recognition process by opening the prediction\_with\_gui.py file.
- The application will use your webcam feed to recognize hand gestures in real time and display them on the interface.