

# **Action Detection for Sign Language Gestures**

A major project report submitted in partial fulfillment of the requirement for the  
award of degree of

**Bachelor of Technology**  
in  
**Computer Science & Engineering**

*Submitted by*

**Amit Sharma (201236)**

**Vedanta Koul (201464)**

*Under the guidance & supervision of*

**Dr. Aman Sharma**

**Assistant Professor (SG)**



**Department of Computer Science & Engineering and Information  
Technology**

**Jaypee University of Information Technology, Wagnaghat, Solan -  
173234 (India)**

# CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled “**Action Detection for Sign Language Gestures**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2023 to May 2024 under the supervision of **Dr. Aman Sharma**, Assistant Professor (SG), Department of Computer Science and Engineering and Information Technology.

Amit Sharma (201236)  
Vedanta Koul (201464)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Aman Sharma  
Assistant Professor (SG)  
Department of Computer Science and Engineering

# ACKNOWLEDGEMENT

First, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing to make it possible to complete the project work successfully.

I am grateful and wish my profound indebtedness to Dr Aman Sharma, Assistant Professor (SG), Department of CSE & IT, Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of my supervisor in the field of “Data Science” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non- instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents and brother.

Amit Sharma (201236)

Vedanta Koul (201464)

Project Group No. 14

# TABLE OF CONTENT

<b>Chapter 1: Introduction</b>	<b>Page No.</b>
1.1 Introduction	1
1.2 Problem Statement	7
1.3 Objective	9
1.4 Significance and Motivation of the Project Work	9
1.5 Organization of Project Report	11
<b>Chapter 2: Literature Surveys</b>	
2.1 Overview of Relevant Literature	14
2.2 Key Gaps in the Literature	17
<b>Chapter 3: System Development</b>	
3.1 Requirements and Analysis	20
3.2 Project Design and Architecture	28
3.3 Data Preparation	30
3.4 Implementation	31
3.5 Key Challenges	37
<b>Chapter 4: Testing</b>	
4.1 Test Cases and Outcomes	40
<b>Chapter 5: Results and Evaluation</b>	
5.1 Results	42
<b>Chapter 6: Conclusions and Future Scope</b>	
6.1 Conclusion	44
6.2 Future Scope	45
<b>References</b>	47
<b>Plagiarism Report</b>	51

# LIST OF ABBREVIATIONS

1. ML: Machine Learning
2. XGB: Extreme Gradient Boosting
3. KNN: K-Nearest Neighbor
4. LGBM: Light Gradient Boosting Machine
5. MLP: Multi-Layered Perceptron
6. SVC: Support Vector Classifier
7. SGD: Stochastic Gradient Descent
8. GBM: Gradient Boosting Algorithm
9. RF: Random Forest
10. ANN: Artificial Neural Networks
11. NB: Naïve Bayes
12. NN: Neural Network
13. PIDD: Pima Indians Diabetes Database

# LIST OF FIGURES

	<b>Page No.</b>
Fig 1: Sign language gestures	3
Fig 2: The hand landmark model of mediapipe	21
Fig 3: Basic Structure of LSTM	23
Fig 4: Forget gate	24
Fig 5: Input gate	25
Fig 6: Output gate	25
Fig 7: Basic flow of the project using simple keras model	28
Fig 8: Detailed flow of the project using LSTM	29
Fig 9: Architecture of the Hand Gesture Recognition System	31
Fig 10: List of libraries used	31
Fig 11: Model Prediction for gesture “Thankyou”	40
Fig 12: Model prediction for gesture “I’m fine”	41

# LIST OF GRAPHS

	<b>Page No.</b>
Graph 1: epoch_categorical_accuracy	42
Graph 2: epoch_loss	43

# LIST OF TABLES

**Table 1:** Identified Gaps in Action Detection for Sign Language Gestures Research

Page No. 14



# ABSTRACT

This major project centers around the critical domain of Action Detection for Sign Language Gestures, addressing the unique communication needs of individuals with hearing impairments. The project involves creating a custom dataset that has been carefully selected to capture the minute details of various hand motions, facial expressions, and body language used in sign language. The system consists of four steps, including segmentation, feature extraction, classification, and verification. The hand region is separated from the background using the background subtraction method, and the palm and fingers are identified to recognize hand gestures. The deep learning model is trained using information such as hand motion size, shape, and color to recognize four different gestures, including the fist, palm, two-finger, and three-finger movements. To evaluate the effectiveness of the system, we compare its performance to other state-of-the-art gesture detection algorithms, and its accuracy and speed are measured. The results show that our proposed system outperforms other algorithms, achieving high recognition accuracy and fast processing speed. The LSTM NN algorithm is a good fit for this project, as it can learn and extract features from data. It has been effectively used for a variety of applications in machine learning, including speech recognition and language modeling. Therefore, our proposed solution can contribute significantly to the fields of artificial intelligence and communication. The proposed smart system that utilizes the LSTM algorithm for hand gesture recognition can facilitate communication with individuals who use sign language as their primary form of speech. By achieving high recognition accuracy and fast processing speed, this system can be applied in various fields and help reduce the communication gap between sign language users and non-users.

# CHAPTER-1

## INTRODUCTION

### 1.1 Introduction

Communication is essential for connecting and sharing feelings among people. For most, spoken language is the primary mode of communication. However, not everyone communicates this way. Some individuals, such as those who are hard of hearing or deaf, use sign language. Sign language is a visual language that uses hand gestures, facial expressions, and body postures to convey meaning. It is not just a handy tool for communication but also a critical part of many cultures and societies. It takes communication between people to connect and express feelings. Spoken language is the primary means of communication for the majority of us. However, not all individuals communicate in this manner. Some people utilise sign language, such as those who are hard of hearing or deaf. Sign language is a visual language that communicates meaning using hand gestures, facial expressions, and body postures.

It's an integral component of many cultures and societies in addition to being a useful means to communicate. People who use sign language will find it simpler to communicate with others thanks to this project's use of technology. Our goal is to develop a system that can recognize and decipher sign language gestures automatically. This entails teaching computers to recognise and comprehend sign language through the use of cutting-edge artificial intelligence technology known as deep learning. Developing such a system presents several challenges. The AI must be able to differentiate between subtle movements and understand the nuances of facial expressions and postures that are integral to sign language. Additionally, it must be adaptive to different sign languages, as various countries and regions have their own versions of sign language.

This project's basic concept is that we think technology can aid in removing obstacles to communication. Develop a system that can recognise sign language gestures in real-time is our special objective. Through this action, we hope to improve accessibility to communication for those who use sign language, facilitating their interactions with non-sign language users. Why does this matter? Indeed, there are issues with the ways that sign language is currently interpreted. Although human translators are invaluable, they might not always be accessible, and occasionally they could find it difficult to fully express the intricacies of sign language. Other solutions require gloves or markers, which might be cumbersome in daily life. With the use of deep learning models,

our project seeks to automatically and effectively recognise and comprehend sign language gestures, thereby overcoming these difficulties. But why is this important? There are several challenges associated with the current methods of interpreting sign language. Human translators, while invaluable, are not always available. Furthermore, they may encounter difficulties in conveying the subtleties and complexities inherent in sign language. Other existing solutions often rely on gloves or markers to track hand movements, which can be cumbersome and impractical for everyday use. Our project intends to address these issues through the use of deep learning models. Deep learning, a branch of artificial intelligence, works by mimicking the neural networks of the human brain. This allows it to process and analyze vast amounts of data to identify patterns and make accurate predictions. By training deep learning models with vast datasets of sign language gestures, we aim to create a system that can automatically and accurately recognize and interpret these gestures. In addition to facilitating everyday interactions, our system could aid in teaching and learning sign language. It could serve as an educational tool, helping both deaf and hearing individuals to learn and practice signing, enhancing overall awareness and understanding of sign language.

Our project is driven by the aspiration to enhance societal well-being, transcending mere technological improvements. It is crucial to create a tool that empowers the deaf and hard-of-hearing community, rather than just developing an impressive piece of technology. We aim to facilitate smoother and more natural communication with them, aligning with the broader principle that technology should enhance accessibility and inclusivity for everyone. As we embark on developing an Action Detection system for sign language gestures, we must acknowledge the interplay between cultural, linguistic, and technological elements. These factors are integral to our endeavor and shape the foundation of our project. This initial chapter serves to introduce the challenges we are addressing, the objectives we have set, the problems we are solving, and the structure of this project report. In subsequent chapters, we will explore the relevant literature, detail our methodology, outline data collection and processing procedures, describe the technologies employed, and present our findings. Together, these sections will narrate the project's conception, development, and the lessons learned along the way. The inspiration behind this project is the desire to bridge communication gaps and make interactions more seamless. For the deaf and hard-of-hearing community, communication with non-sign language users often involves significant hurdles. Human translators, though invaluable, aren't always available and might struggle to convey the intricacies of sign language. Moreover, existing solutions that involve gloves or tracking markers can be inconvenient for daily use.

Our project aims to overcome these obstacles by developing a real-time sign language recognition system using deep learning models. Deep learning, a subset of artificial intelligence, excels at recognizing patterns in large datasets, making it well-suited for interpreting the complex gestures of sign language. By training these models on extensive datasets of sign language videos, we aim to create a system that can automatically and accurately interpret sign language gestures. The development process involves several key components. First, we require a comprehensive and diverse dataset of sign language gestures. This dataset must capture a wide range of signs to ensure the AI can recognize gestures accurately, despite variations in speed, style, or individual signing characteristics. Second, robust algorithms are needed to process this data, learn from it, and make precise predictions.

Finally, the resulting system must be integrated into a user-friendly interface, accessible on common devices such as smartphones and tablets. Our ultimate goal is to produce a practical and accessible tool that facilitates communication between sign language users and others, thereby promoting inclusivity. Imagine a world where a deaf person can enter any public space and communicate effortlessly without depending on predefined arrangements or cumbersome equipment. Such a tool would be immensely beneficial in emergency situations, where immediate communication is crucial. Beyond enhancing daily interactions, our system could serve as an educational resource. It could assist both deaf and hearing individuals in learning and practicing sign language, promoting greater awareness and understanding.



**Fig 1.1.1 Sign language gestures [29]**

In this research, we aim to develop an intelligent system capable of recognizing hand gestures and human movements using computer vision, a subfield of artificial intelligence (AI). Our goal is to train a computer vision program to "understand" scenes or features within images to recognize and locate hands—a crucial first step in any hand processing system. The process begins by identifying and isolating the hand region in the original images captured by input devices. This involves several stages: detecting the presence of a hand in the image, extracting various features to represent hand motions, and finally comparing these features to recognize the gesture. By teaching computers to perform these tasks, we hope to build a system that can automatically detect and interpret hand gestures. One of the main challenges in developing this system is ensuring accurate hand detection and gesture recognition. The system must be able to identify hands in diverse conditions, such as varying lighting, different backgrounds, and hands in different positions or orientations. To address these challenges, we leverage deep learning algorithms, which are highly effective at pattern recognition and can learn from vast amounts of data. We will start by creating a substantial dataset of images that include various hand gestures in multiple scenarios. This dataset will be used to train our deep learning models to recognize and interpret the gestures accurately. The training process involves feeding the images into the model, enabling it to learn the patterns associated with each gesture.

Next, we extract features from the hand images. These features could include information about the hand's shape, position, and movement. Feature extraction is critical because it transforms the raw image data into a format that the model can effectively process and interpret. Once the features are extracted, they are compared against known gesture patterns to identify the specific gesture being performed. To make the system more practical and user-friendly, we aim to include a data storage and verification component. This will allow the system to store historical gesture data and verify new inputs against this stored data. By doing so, we can enhance the system's ability to recognize and learn from repeated gestures, making it more accurate and reliable over time.

The ultimate aim of this project is to facilitate better communication between sign language users and those who do not understand sign language. By developing a system that can recognize and interpret hand gestures in real-time, we can bridge the communication gap and create more inclusive interactions. This intelligent hand gesture recognition system has numerous potential applications. It can be used in educational settings to assist in teaching and learning sign language, making it easier for both deaf and hearing individuals to communicate. In workplaces, it could enable more inclusive communication without the need for human interpreters. Public service

environments, such as hospitals and government offices, could also benefit from such a system by offering more accessible communication for sign language users.

Hand gesture recognition has long been a central focus of research in computer vision and artificial intelligence (AI). One of the most prevalent methods for detecting hand gestures involves computer vision techniques, which gather and analyze visual data of hand movements. However, the task of hand detection is inherently challenging due to variations in hand pose, orientation, location, and scale, as well as differing lighting conditions. These variations necessitate robust systems capable of accurate recognition. The primary approach in hand gesture recognition involves several key steps. Initially, the system must detect and isolate the hand from the rest of the image. This step is complicated by the natural variability in how hands can appear; they can be positioned or oriented in countless ways, which makes consistent detection difficult. Different lighting can also play a significant role, casting shadows or creating glare that can obscure hand features. Once the hand is detected, the next step is feature extraction. This involves identifying key characteristics of the hand and its movement, such as the contour, edges, and the spatial relationship between different parts of the hand. Extracting these features enables the system to create a model of the hand gesture that can be compared against known gestures in a database.

After feature extraction, the gesture must be recognized by comparing the extracted features with pre-defined gesture models. This typically involves machine learning algorithms that have been trained on large datasets of hand gestures. The goal is for the system to accurately match the current gesture with the most similar one in its database. Recognizing hand gestures accurately has numerous practical applications. In robotics, for example, it allows humans to interact more naturally with robots by using hand signals instead of verbal commands or physical controllers. In AI, it can enhance user interfaces, making them more intuitive and accessible. Gesture recognition systems can also play a crucial role in communication, particularly for individuals who rely on nonverbal methods, such as the deaf and the blind.

For the deaf and hard-of-hearing community, hand gestures are an essential mode of communication through sign language. Developing systems that can recognize sign language gestures can significantly improve accessibility, allowing for smoother interactions with hearing individuals without the need for a human interpreter. Such systems can be implemented in various settings, from educational institutions to workplaces and public services, to foster more inclusive

environments. For the blind, who may rely on tactile hand gestures or other nonverbal cues, gesture recognition systems can offer new ways to interact with technology. By detecting and interpreting these gestures, the system can provide auditory feedback or other forms of assistance, enhancing independence and accessibility. There is a growing demand for tools that can accurately recognize hand gestures due to these wide-ranging applications. Advanced technologies, such as deep learning and neural networks, are increasingly being employed to improve the robustness and accuracy of these systems. By learning from extensive datasets, these AI models can better handle the variability in hand appearances and environmental conditions.

We are utilizing the Long Short-Term Memory (LSTM) algorithm, which has proven highly effective at recognizing hand motions, through sophisticated methods and methodologies. Our proposed system aims to provide a quick and accurate way to recognize hand gestures in live video and involves four main stages: segmentation, feature extraction, classification, and verification. In the segmentation stage, we use the background subtraction method to isolate the hand region from the rest of the image. This step is crucial as it helps to remove any extraneous background elements that could interfere with accurate gesture detection. The subsequent step, feature extraction, involves identifying various characteristics of the hand to describe its motions. This includes details about the hand's position, orientation, and shape. Effectively extracting these features is essential for the system to accurately represent hand gestures. Once features are extracted, the classification stage begins. Here, the system compares the extracted data with pre-defined hand motions stored in its database. This comparison allows the system to identify the specific hand gesture being performed. Finally, in the verification stage, the system records the gesture data and verifies it against input from the user to ensure accuracy before proceeding. The proposed system is expected to make significant contributions to the fields of computer vision and artificial intelligence, especially in communication with sign language users. By accurately recognizing hand movements in real-time video, we can develop a more effective form of communication for individuals who rely on sign language. This capability could greatly enhance interactions for the deaf and hard-of-hearing community, providing them with more seamless integration into everyday activities.

Beyond its applications in sign language recognition, our system has the potential to impact various other fields. In virtual reality and gaming, for instance, hand gesture recognition can offer more immersive and intuitive user experiences. Users would be able to interact with virtual environments more naturally, using hand gestures to control actions within a game or virtual space. In robotics, hand gesture recognition can facilitate more sophisticated human-robot interactions, allowing

humans to command robots through simple hand signals for tasks where verbal instructions may be impractical or inefficient. The hand gesture recognition system proposed in this research has the potential to revolutionize how humans and computers interact. By utilizing advanced computer vision techniques and the LSTM algorithm, our goal is to develop a system that is precise, efficient, and capable of enhancing communication for all users. The ability to recognize hand gestures accurately and in real-time can not only improve communication for sign language users but also open up new possibilities in various high-tech fields. Through this project, we aspire to create a tool that is not only technologically advanced but also practical and beneficial for everyday use. The implementation of such a system could lead to significant advancements in how we interact with technology, making these interactions more natural and inclusive. Our research emphasizes the importance of integrating cutting-edge technologies to foster a more connected and accessible world, reinforcing the idea that the ultimate goal of technological innovation should be to enhance the human experience across all areas of life.

## **1.2 Problem Statement**

Effective communication can be particularly challenging for individuals with hearing disabilities, especially those who rely on sign language. While sign language is a rich and expressive form of communication, it is difficult to recognize and understand accurately due to the lack of effective tools. Current systems often struggle to capture the subtle nuances of various hand gestures, facial expressions, and body language that are crucial for accurate interpretation. Furthermore, the absence of a comprehensive dataset specifically designed for sign language gestures complicates the development of reliable models. Existing databases may not adequately reflect the diversity of sign language across different regions, cultural backgrounds, and personal communication styles. Our project aims to address these challenges by creating a customized dataset tailored for Action Detection in Sign Language Gestures. This dataset will incorporate a wide range of sign language expressions to ensure inclusivity and effectiveness. The primary goal is to leverage deep learning models to develop a system capable of recognizing and understanding sign language gestures, thereby improving communication for people with hearing impairments. One of the main challenges in this endeavor is creating a dataset that accurately represents the diversity of sign languages. This includes variations in gestures across different regions, cultural backgrounds, and individual preferences. By including a broad spectrum of sign language expressions, we aim to create a more comprehensive and representative dataset that can serve as a robust foundation for training our deep learning models.



Feature extraction is another critical aspect of our project. This involves identifying key characteristics of hand gestures, such as position, orientation, and shape, as well as incorporating facial expressions and body language. Accurately capturing these features is essential for the model to interpret sign language gestures correctly. To achieve this, we will employ advanced computer vision techniques and machine learning algorithms, including Long Short-Term Memory (LSTM) networks, known for their effectiveness in sequence prediction tasks. The next step is to train our deep learning models using the customized dataset. This training process involves feeding the dataset into the model, allowing it to learn the patterns and nuances associated with different sign language gestures. One of the significant challenges here is achieving high accuracy in action detection, sensitivity to various gestures, and specificity in reducing false positives and false negatives. By fine-tuning the model's performance, we aim to create a system that can recognize sign language gestures with high precision. Our project holds the potential to significantly improve communication tools for the hearing-impaired community. By developing a reliable and accurate sign language recognition system, we can make interactions smoother and more inclusive for individuals who rely on sign language. This has wide-ranging applications, from enhancing accessibility in educational and workplace settings to facilitating better communication in public services and social interactions.

Sign language recognition has long faced challenges due to the varied real-world conditions in which it must function. Factors such as different lighting conditions, diverse backgrounds, and multiple people in a single frame often cause current systems to falter. These practical difficulties hinder the adoption of reliable and accurate sign language interpreting systems in daily use. Furthermore, there is no standardized method for evaluating the performance of these models, which adds another layer of complexity. As a result, there is often a gap between the performance of a model in a controlled laboratory environment and its usefulness in real-world settings. Our project recognizes the necessity of a comprehensive solution that addresses not only the complexities of sign language gestures but also the practical issues associated with real-world deployment. The foundation of our approach is the creation of a robust dataset that encompasses a wide range of scenarios. This dataset is crucial for training the deep learning model to handle variability and enhance its generalization abilities.

To tackle these challenges, our project will employ advanced deep learning techniques, including convolutional and recurrent neural networks. These techniques are well-suited to managing the intricacies of sign language gestures and the variability of real-world conditions. Convolutional

neural networks (CNNs) are particularly effective for processing images and recognizing patterns, making them ideal for analyzing the visual aspects of sign language. Recurrent neural networks (RNNs), on the other hand, excel in handling sequential data, which is essential for understanding the temporal dynamics of sign language gestures. The ultimate goal of this project is to bridge the communication gap for individuals with hearing impairments by developing an efficient action recognition system for sign language gestures. Success will not only be measured by the model's accuracy but also by its applicability in everyday situations and its ability to adapt to changing real-world conditions. This means that the system must perform well in diverse environments, maintain high levels of accuracy despite variations in lighting and background, and effectively differentiate between gestures made by different people. To ensure the practicality of the solution, rigorous testing will be conducted under various conditions that mimic real-world scenarios. This includes evaluating the model's performance in different lighting environments, with varying background complexity, and in the presence of multiple individuals. By doing so, we aim to refine the system to handle the full spectrum of challenges it may encounter in daily use. Moreover, the project will explore methods for establishing standardized evaluation criteria that more accurately reflect the performance of sign language recognition systems in real-world settings. This will help ensure that the models developed are not only theoretically sound but also practically useful.

### **1.3 Objective**

1.3.1 To study the existing techniques / tools for hand gesture detection.

1.3.2 To propose deep learning algorithm for hand gesture detection.

1.3.3 Testing and validation of proposed algorithm.

### **1.4 Significance and Motivation of the Project Work**

The potential impact on the lives of those with hearing impairments makes our project on Action Detection for Sign Language Gestures both significant and highly motivated. Effective communication is a fundamental aspect of human connection, allowing individuals to share their thoughts, feelings, and ideas. People who rely on sign language often face challenges due to the limitations of current technology used for sign language interpretation. Our project aims to address these shortcomings by developing an accurate and reliable system, thus enabling individuals with hearing impairments to communicate more easily and inclusively across various settings. The determination to enhance the lives of those who are hearing impaired drives our project's purpose. Despite advances in technology, existing sign language interpretation tools struggle to fully

understand and interpret the intricacies of sign language gestures. Our initiative seeks to bridge this gap by improving the accuracy and reliability of sign language action recognition. We plan to achieve this through the creation of a customized dataset and the application of advanced deep learning algorithms.

Furthermore, this project is propelled by the broader goal of promoting accessibility and inclusivity. By offering better tools for sign language interpretation, we help ensure that individuals with hearing impairments can enjoy equal opportunities and break down communication barriers. This can vastly improve their integration into societal activities and interactions, making communication more fluid and inclusive. The significance of this project extends into the professional and educational realms as well. In educational settings, proficient sign language interpretation is critical to ensuring that students with hearing impairments have equal access to learning opportunities. Effective communication in these environments allows for a richer educational experience and can significantly impact academic performance and social integration. In professional environments, the ability to accurately interpret sign language gestures is equally crucial. Enhanced sign language recognition can open up new job opportunities for individuals with hearing impairments, enabling them to participate fully in the workforce. This leads to more productive and collaborative workspaces where communication barriers are minimized, and diversity is embraced.

By focusing on these objectives, our project does not just aim to develop a better technological tool but rather to make a substantial positive impact on real lives. The creation of a robust and adaptable sign language recognition system will cater to the nuanced needs of its users, ensuring it performs reliably in various real-world conditions. This will be achieved by training the system using a comprehensive dataset that reflects diverse environments and by employing cutting-edge deep learning techniques such as convolutional and recurrent neural networks.

Our project, focused on the development of a reliable system for sign language action detection, is poised to make an immediate impact on individuals with hearing impairments. Beyond this direct benefit, it represents a significant advancement in the field of assistive technology. By creating a dependable solution for interpreting sign language, we not only provide valuable tools for the deaf community but also set a benchmark for future innovations in human-computer interaction. This aligns with global efforts towards fostering accessibility and inclusivity. Driven by a strong societal mission, our initiative aims to leverage cutting-edge technology to enhance the lives of those with

hearing impairments. It is more than just a project about detecting sign language gestures – it embodies our commitment to using technology to foster inclusivity and effect positive social changes. By integrating state-of-the-art technology to meet the unique needs of the deaf community, we chart an inspiring course for further research and innovation.

Moreover, our project has the potential to inspire creativity and collaboration within the larger community. The progress made in developing accurate sign language interpretation tools opens doors for educators, developers, and researchers to work together to refine and enhance these tools. Through the application of advanced deep learning techniques and the creation of specialized datasets, we pave the way for future advancements in computer vision and human-computer interaction. This project underscores the necessity of ensuring that technological advances are accessible to all, recognizing that marginalized groups should not be left behind in technological progress. Addressing the specific needs of individuals with hearing impairments, we contribute to a more inclusive technological landscape. Our commitment extends beyond technical innovation; we are equally dedicated to the ethical aspects of technology development. By enhancing communication for the hard of hearing, we also integrate important considerations of data security, privacy, and fair representation into our research. This ensures our work aligns with broader discussions about responsible technology development and its applications.

## **1.5 Organization of Project Report**

### Chapter 2: Literature Survey

- 2.1 Overview of Relevant Literature
  - Introduction to existing literature on Action Detection for Sign Language Gestures, exploring various methodologies, datasets, and technologies in the context of deep learning.
- 2.2 Key Gaps in the Literature
  - Identification and discussion of gaps or limitations in existing literature concerning action detection in sign language gestures, paving the way for the unique contribution of our project.

### Chapter 3: System Development

- 3.1 Requirements and Analysis
  - In-depth analysis of the specific requirements for developing an Action Detection system for Sign Language Gestures, emphasizing the need for a self-created dataset.

- 3.2 Project Design and Architecture
  - Presentation of the overall design and architecture of our system, the integration of the self-created dataset and the chosen deep learning model.
- 3.3 Data Preparation
  - Explanation of the process involved in creating the specialized dataset, capturing diverse hand movements, and body language in sign language gestures.
- 3.4 Implementation
  - Inclusion of relevant code snippets, algorithms, tools, and techniques used in implementing the deep learning model tailored for action detection in sign language.
- 3.5 Key Challenges
  - Discussion of challenges faced during the development process, focusing on the unique aspects of creating a self-generated dataset and optimizing the deep learning model for sign language gestures.

#### Chapter 4: Testing

- 4.1 Testing Strategy
  - Overview and discussion of the testing strategy applied to evaluate the accuracy and reliability of our Action Detection system.
- 4.2 Test Cases and Outcomes
  - Presentation of outcomes and insights derived from the testing phase, demonstrating the system's effectiveness.

#### Chapter 5: Results and Evaluation

- 5.1 Results
  - Presentation and interpretation of findings obtained from the system's performance, showcasing its accuracy in detecting sign language gestures.

#### Chapter 6: Conclusions and Future Scope

- 6.1 Conclusion
  - Summarization of key findings, limitations, and contributions made by the project in the realm of Action Detection for Sign Language Gestures.

- 6.2 Future Scope
  - Exploration of potential avenues for future research and development, considering enhancements, broader applications, and further advancements in sign language recognition technology.

# Chapter 2: Literature Survey

## 2.1 Overview of Relevant Literature

This chapter takes a good look at what other researchers have done in the world of sign language and technology. Nandy et al. [5] proposed a technique of identifying and recognizing Indian Sign gestures from gray scale images. Their approach consists of turning an input gray-scale frame that captures the signing gestures from each frame of the video feed and employing a directional histogram for deriving features from each frame. Clustering is used last, whereby the signs are classified into some among the pre-planned groups depending on the attributes they possess. According to the authors, the 36-bin histogram method proved to provide better results in their investigation and showed a 100% sign recognition rate.

It suggested the neural network architecture for detecting and processing sign language as well as producing text in real-time from a video stream. Pre-photos processing, hand pose & motion information extraction and framing represent three major design stages of this system. Certain hand qualities are indicated by a hand's POI of a point of interest [ 6 ]. In this approach, the authors' neural network architecture had 55 different features which consisted of CNN layers of predictive signs. For this research, the model was trained and tested for using the English alphabet starting with A to Z. It asserted achieving a 100% recognition rate and 48% immunity against distortion. First, creating a dataset with 1200 samples for ten static signs and letters (suggested by Chen , [8]) he proceeded with the model. The first stage included edge segmentation that helped to determine the hand's edge prior to changing the RGB images to YUQ/YIQ color space for differentiating among gestures. The convex hull approach was used to recognize the hand and identify the fingers. In the end, a neural network was used for classification. This approach thus translated into a mere error margin of about 1.8%.

Using the idea of India sign language, Sharma et al. [8] designed this method that connects with those whose speech and hearing is impaired. The data were then pre-processed to the conversion of RGB to grayscale in the matlab environment after taking the picture [8]. Afterwards, a Sobel detector involving a 3×3 filter identified the images' edges. Finally, the reduced image containing 600 pixels was submitted to hierarchical centroid approach resulted in 124 characteristics. The two classification methods used included neural networks and KNNs. The obtained accuracy was

97.10% by using this methodology.

Agarwal et al.,[9] developed a method of utilizing sensor glove for converting signed signs into phrases which could be understood by an individual without speech impediment. Individuals employed the sensor gloves for this purpose of enacting the gestures. When the gestures and the database matched, the recognized gesture was sent for parsing to form a sentence. Initial accuracy for a given version of the application was at 33.33%. Therefore, 100% accuracy was achieved by version 2 in dealing with the simple and continuous tense.

Wazalwar and others in 2017 designed a way of reading sign language information from an input video by framing and segmentation. They used the P2DHMM algorithm for manual tracking and track camshift for automated tracking. The indications were identified with a Haar cascade classifier. The WordNet POS tagger then identified the sig and tagged each word in it. Therefore, it provided the final text, which formed the sentence gradually. With respect to this, the LALR parser generated an important statement in English.

As explained by Shivashankara and Sri-nath, they developed a tool that read symbols and hand signs using an American sign language [11]. YCbcr was adopted as a model for the performance optimization of the skins color clustering by the authors [11]. As a result, the model was used for pre-processing an image. The study discovered that the centroid of a pre-processed picture was recognized as the starting point of that gesture. The peak offset of gesture was used to determine it and the gesture was recognized later on. This model showed an overall accuracy of 93.05%.

Camgoz et al. [12] proposed a series showing signs and speaking by sequence-to-sequence conversion instead of single-to-single matching. Following the analogy of a typical neural machine translation process, the inventors of this visionary approach recreated its tokenization and embedding phases. This is shown in the CNN architecture for the neural machine translation step when transforming signed videos into spoken languages [12], with attentional encoder/decoder which provides probability distribution functions modeling each spoken language given a signed video. The authors first used word embedding to change sparse vectors into compact ones so that related semantic words would be side by side. The use of encoder-decoder phase helped to maximise on conditional probability. Encoding produced the sign movies' features as a fixed size vector. The inputs included the word embedding and the previously hidden state for the decoding stage. This aided in word prediction. Moreover, the authors integrated an attention mechanism in the decoding phase in order to tackle problems related to dependency length and vanishing gradient. They created PHOENIX14T, a set of continuous sign language translations.



Mariappan and Gomathi's [13] proposed a novel approach for recognizing signed utterances in Indian Sign Language (IS). They proposed a method that utilizes ROI-based tracking of an open CV-skin segmentation function based on this observation. Signs prediction was done using FCM electronic. From ten distinctive subjects they collected a dataset consisting of 50 sentences and 80 words for training and testing. In this way, they did some further morphologic operations with the binary image created from skin segmentation. The aim was to improve its features and perform some additional filtering on color images for reducing noise in digital sources.

Therefore, they applied the FCM method and developed 40 words derived from ISL with the accuracy score of 75%.

Mittal et al. [14] leveraged leap motion in continuous sign language recognition system based on LSTM. Four gated lstm cells together with two dimensional convolutional neural network were applied for sign sentence recognition while assigning each word with its own tag. When using three essential gates for input, a forgot gate generated an output at its previous event  $t-1$  and was awarded a label, particularly in conjunction with its word or phrase. They used a special sign, the dollar (\$), to denote this difference between the successive signs of the series. The researchers utilised the RESET flag which appeared next to \$, denoting a step-over from one sign to another. A modified LSTM model was therefore used to recognize sign sentences with an accuracy of up to seventy-two percent. They obtained a 89.50% sign word recognition accuracy rating.

To this end, de Costet et al. employed Transformer network for sign recognition in films, including gestures of mouth and brows [15]. Instead of that, they come up with proposals such as the pose transformer network, Video Transformer Network for motion recognition, and multimodal transformer network for sign recognition which incorporate more than one neural network. To detect isolated signs in their multi-model-based signing language recognition system, Jiang et al. [16] used skeleton base graph strategy. Isolated sign recognition was proposed by the authors based on SAM-SLR framework; while SL-GCN and SSTCN were used as the key points skeletonization and feature extraction models respectively [16]. To evaluate the proposed framework, AULTS dataset was employed.

BLSTM-3DRN has been used by Liao et al. [17] to recognize dynamic sign language. In order to identify gestures over DEVISIGN\_D (Chinese hand sign language), the authors employed a bi-directional LSTM model that is serialised in three stages: hand detection, spatio-temporal information representation, and fingerprinting identification [17]. In the on-going recognition of

syntax constructions in sign language, Adaloglou proposed the ResNet with B-LSTM that was dubbed I3D [18]. Three types of annotations and a suggested scheme were applied by them to several instances of RGB + D data concerning primarily Greek SL. Table one shows how various deep learning models, especially when using a hybrid CNN/LSTM, operate on different datasets.

## **2.2 Key Gaps in the Literature**

The body of research on Action Detection for Sign Language Gestures has advanced the discipline and demonstrated a wide range of techniques and tools. Nevertheless, a thorough analysis identifies a number of significant gaps that indicate areas in need of additional research and development. The following are the gaps that were found:

### **1. Limited Exploration of Self-Created Datasets:**

Although several research works have showcased creative methods with pre-existing datasets [5][7][13], there is a noticeable void in investigating the possibilities of custom-made datasets designed especially for Action Detection in Sign Language Gestures. Self-generated datasets provide a more comprehensive depiction of a wide range of sign language expressions, taking into account individual communication styles and regional variances in signs.

### **2. Limited Exploration of Real-time Sign Language Gesture Recognition:**

Real-time action detection receives less attention in the literature than offline processing of sign language gestures [5][8][11]. For practical applications like assistive technology and communication tools, it is imperative to develop systems that can recognise and interpret gestures in sign language in real-time.

### **3. Inadequate Attention to Noise Immunity:**

Although some research [6][7] indicate outstanding accuracy rates, a thorough investigation into the models' resistance to noise and outside influences is lacking. To improve the resilience of Action Detection systems, real-world issues like changes in illumination and background clutter must be taken into account.

### **4. Limited Exploration of Multimodal Approaches:**

The majority of the literature on sign language recognition [15][16][17] relies on visual characteristics. Investigating multimodal strategies that incorporate auditory, visual, or other

sensory modalities is noticeably lacking, nevertheless. An interpretation of sign language expressions that is more comprehensive may result from taking into account non-manual elements like lip and eyebrow locations.

### **5. Insufficient Benchmarking Across Diverse Sign Languages:**

Most of the research focuses on certain sign languages, including American Sign Language (ASL) [11] and Indian Sign Language (ISL) [13]. Considering the variety of linguistic patterns and gestures, there is a notable lack of benchmarking and evaluating the efficacy of models over a wider range of sign languages.

This table offers a succinct summary of the gaps that have been found, along with the relevant literature references that back up each gap.

**Table 2.2.1: Identified Gaps in Action Detection for Sign Language Gestures Research**

<b>Gap</b>	<b>Description</b>	<b>Examples in Literature</b>
1. Limited Exploration of Self-Created Datasets	Lack of investigation into custom-made datasets for Action Detection in Sign Language Gestures, limiting the diversity of expressions.	[5][7][13]
2. Limited Exploration of Real-time Sign Language Gesture Recognition	Insufficient focus on real-time processing, crucial for practical applications like assistive technology.	[5][8][11]
3. Inadequate Attention to Noise Immunity	Lack of thorough examination of model resilience to real-world challenges such as changes in illumination and background clutter.	[6][7]
4. Limited Exploration of Multimodal Approaches	Predominance of visual-centric approaches with limited incorporation of auditory or other sensory modalities.	[15][16][17]
5. Insufficient Benchmarking Across Diverse Sign Languages	Concentration on specific sign languages, such as ASL and ISL, without benchmarking models across a broader range of linguistic patterns and gestures.	[11][13]

# Chapter-3

## System Development

### 3.1 Requirements and analysis

#### 3.1.1 Hardware Requirements

- **A Web Camera**

Our webcam is an adaptable and user-friendly tool designed to efficiently and easily record hand gestures. This webcam's high-resolution sensor produces clear, sharp images, which are necessary for precise gesture interpretation. In real-time processing, it offers a fluid and responsive user experience with a typical frame rate of 30 frames per second (FPS). The webcam's wide field of vision enables it to record a large area and supports a variety of user interactions.

With its plug-and-play design, this webcam is easy to use and connects to computers and other devices using conventional USB ports. Because of its lightweight and compact design, it may be positioned freely for the best possible gesture recognition in a variety of scenarios. Although it doesn't have the depth-sensing capabilities of more specialised cameras, its accessibility and price make it a good option for projects with less demanding depth requirements or budgetary restrictions.

#### 3.1.2 Software and library requirements

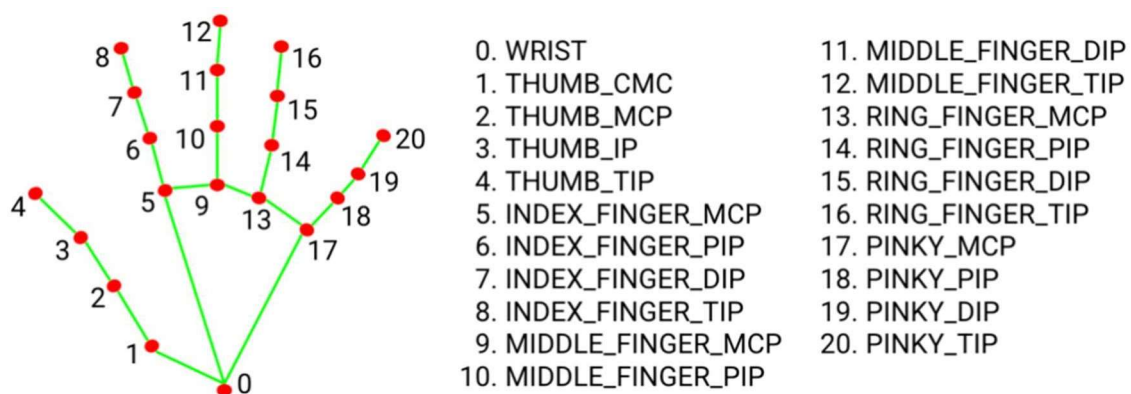
##### 1. OpenCV [2G]

- A flexible open-source software library for computer vision and machine learning is called OpenCV (Open Source Computer Vision Library). It offers a comprehensive range of tools and functions for processing images and videos, and is designed in C++ and Python. This enables developers to construct apps for tasks like object detection, facial recognition, and gesture analysis. Modern deep learning models and conventional computer vision methods are also included in OpenCV's extensive library of algorithms. OpenCV, which is widely used in both academia and industry, is a vital tool for people working in the computer vision and image processing sectors since it makes it easier to develop applications ranging from robots and augmented reality to medical imaging.
- OpenCV provides a wide range of computer vision algorithms beyond image processing. These consist of image stitching, object recognition, and feature detection. Modern methods and ancient algorithms are also supported, providing flexibility for a wide range of applications. The ability to develop models for tasks like classification, regression, and

clustering is made possible by the integration of machine learning capabilities within OpenCV. OpenCV performs well in 3D applications as well as 2D image processing. For computer vision jobs that need depth perception, the library offers tools for camera calibration. It also facilitates 3D reconstruction from several photos, which helps with augmented reality and structure-from-motion applications.

## 2. MediaPipe [27]

- Google developed MediaPipe, a robust open-source library that makes it easier to create cross-platform perceptual computing apps. MediaPipe, a company that specialises in real-time machine learning solutions, provides a large selection of pre-built models and components for applications including position estimation, hand tracking, face detection, and augmented reality effects. This library makes complicated computer vision jobs easier by giving developers a high-level framework that makes it possible for them to incorporate complex algorithms without requiring a lot of specialised knowledge. Because of MediaPipe's adaptability and effectiveness, perceptual computing technologies are now more accessible. It may be used in a wide range of applications, such as gesture detection, virtual try-on experiences, and interactive multimedia projects.
- Because of its modular nature, developers may easily customize and extend it to create bespoke perception pipelines. Because MediaPipe is cross-platform compatible, it can be easily deployed on a variety of devices and is appropriate for a wide range of applications, including augmented reality, HCI, content creation, health, fitness, and robotics. Providing effective solutions for a wide range of perception tasks, MediaPipe is a potent tool in the machine learning landscape thanks to its emphasis on real-time performance and integration with TensorFlow Lite.



**Fig 3.1.2.1. The hand landmark model of mediapipe [30]**

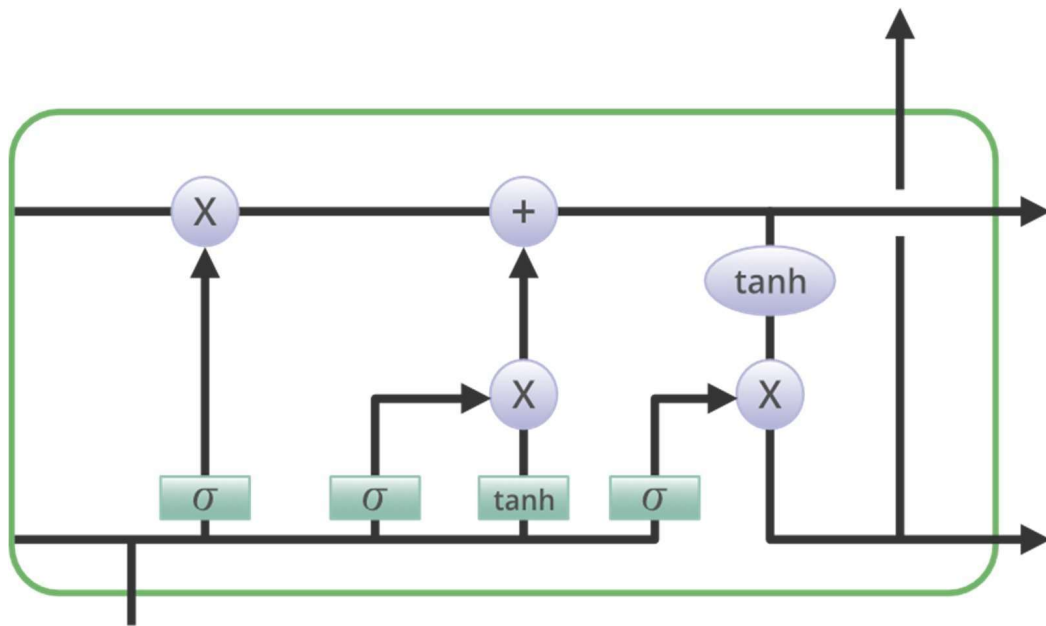
### 3. Long Short-Term Memory(LSTM)

- In order to deal with long dependencies in a sequence, a certain RNN known as LSTM or Long Short-term Memory was developed. In 1997, Hochreiter and Schmidhuber introduced LSTM, one of the most sophisticated memory cells that can store data for prolonged periods of time. This innovation addresses the shortcomings of conventional RNNs, which struggle to absorb and retain information from distant time steps.
- For instance, LSTM networks can be applied to various sequential data types including speech, text, and time series. Firstly, the ability of LSTM to learn complex longitudinal connections is important in areas like sound recognition, language transcription, and sequential predictions. LSTMs use a memory cell controlled by three crucial gates: input, forget, output. Unlike traditional RNNs, this GRU uses one hidden-state that gets transferred across time.
- The input gate allows for addition of meaningful information to be stored in the memory cell through control over inputs of data. On the other hand, the forget gate controls the deletion of unnecessary data in memory cells to make sure that only vital information remains. Simultaneously, the output gate controls the data output of the memory cell. This leads to a complicated gateway that enables LSTM net to make sound decision on what to keep or forget as it passes the network, thereby prompts the network to learn on long term dependency.
- The deep LSTM networks are created by stacking multiple LSTM layers that have been perfected in order to accurately model complex sequential patterns. The network will use a hierarchical approach so that it is able to pick up intricate designs in the incoming information. Besides, LSTMs can also be combined with other neural networks like CNN for use on image and video based apps. Due to their flexibility, one should take into consideration using the LSTM algorithm in various professions as it leads to substantial progress in the field of ML and AI.

Structure of LSTM:

The LSTM is a chain structure made up of several memory blocks called cells and four neural

networks.



**Fig 3.1.2.2 Basic Structure of LSTM [28]**

The gates perform memory modifications, whereas the cells store information. Three gates are present.

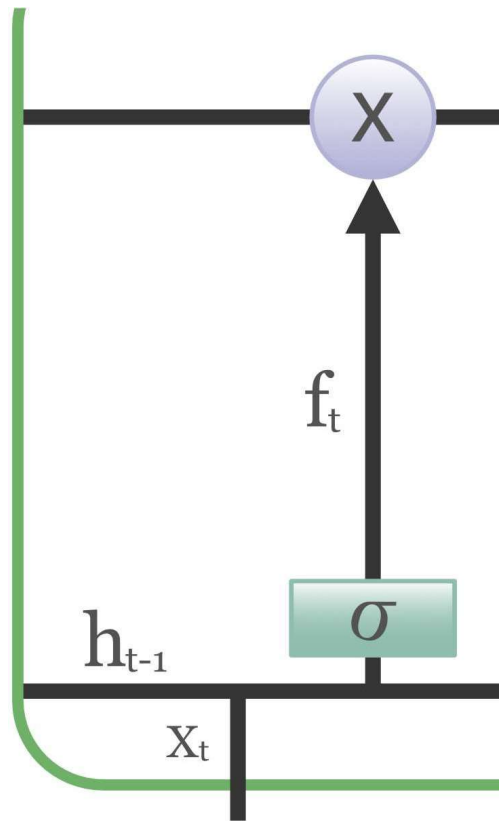
**1. Forget Gate:** Forget gate discards old information within the cell state. Gate that gets the inputs at a certain time, namely  $x_t$  and  $h_{t-1}$  which multiplies weight matrices and add bias at last. Once subjected to an activation function it becomes a two-state value. If the output for a particular cell state is zero, the details are no longer important or of any benefit, while if it is one, then preservation for future application takes place. The forget gate's equation is as follows:

$$\text{Where: } f_t = (W_f * [h_{t-1}, x_t] + b_f)$$

- $W_f$  represents the weight matrix that links to the forget gate.
- The computation is indicated by the notation  $[h_{t-1}, x_t]$ , representing the concatenation of the previous hidden state and the current input.
- Bias of the forget gate is  $b_f$ .



- This is referred to as the sigmoid activation function and it is denoted by  $\sigma$ .

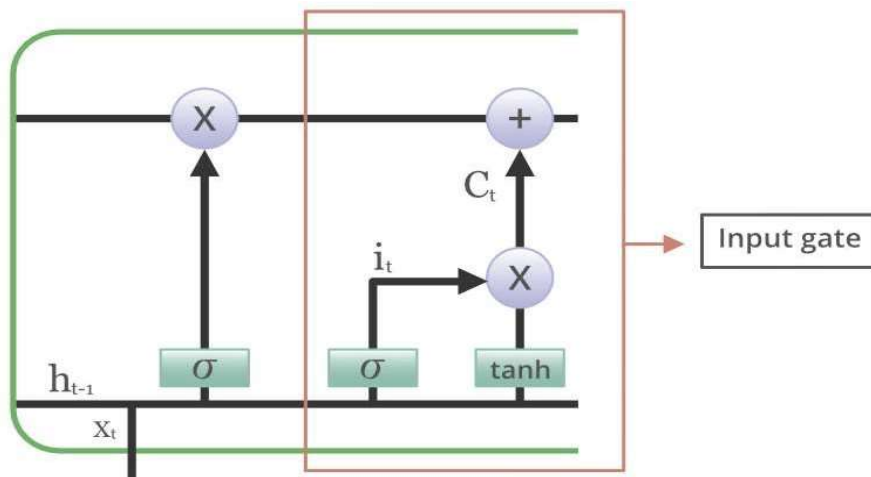


**Fig 3.1.2.3. Forget Gate [28]**

**2. Input gate:** It updates the cell's state by providing relevant data to the input gate. The process starts with regulating information by using a sigmoid function that selects the items for storing in a way similar to the forget gate by using input values,  $x_t$  and  $h_{t-1}$ . Subsequently, a vector consisting of all conceivable values derived from  $x_t$  and  $h_{t-1}$  is built based on the application of tanh function yielding an output between -1 and +1. Finally, the wanted useful information can be determined through multiplication of the vector values with the controlled values. The input gate's equation is as follows:

$$\begin{aligned}
 W_i * [h_{t-1}, x_t] + b_i &= \sigma(i_t) \\
 W_c * [h_{t-1}, x_t] + b_c &= \hat{C}_{\tanh} \\
 C_t \text{ is equal to } f_t \odot C_{t-1} + i_t &\triangleq \hat{C}_t
 \end{aligned}$$

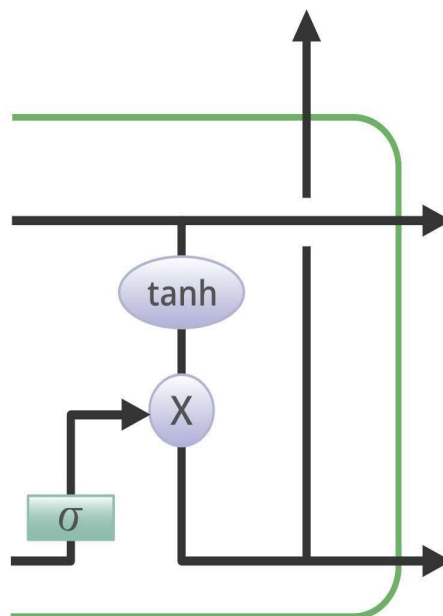
where:  $\tanh$  is the tanh activation function, and  $\odot$  indicates element-wise multiplication.



**Fig 3.1.2.4. Input Gate [28]**

**3. Output gate:** An output gate retrieves useful data from the content cell's state to display as output. For instance, the cell is initially built using the tanh function. Next, the data is separated using  $x_t$  and  $h_{t-1}$  based on their value of relevance, and then through a sigmoid function, it is made controllable. Lastly, they are multiplied so as to serve as the inputs/outputs for the subsequent cell in transmitting the values of the vector and controlled values. The output gate's equation is as follows:

$$W_o * [h_{t-1}, x_t] + b_o = o_t$$



**Fig 3.1.2.5. Output Gate [28]**

### **The advantages of LSTM:**

- Long- term dependencies can be detected using LSTM networks. Each has a ‘memory cell’ that is capable of storing data over some time duration.
- Standard RNNs face problems arising due to gradient vanishing and exploding while training models that are long term. This problem is resolved in gated recurrent units and LSTM networks, which determine what information is kept and what is discarded using gating mechanisms.
- Despite there being long pauses between critical happenings in the sequence, LSTM enables the model to learn and remember the important contextual information. Consequently, LSTM is used in situations where contextual knowledge is essential. say, computer translation.

### **The disadvantages of LSTM:**

- LSTMs are more expensive in terms of computational cost compared to less complicate structures like feed forward neural networks. Therefore, in cases involving huge datasets/resources, they may not be as scalable.
- However, it is worth noting that as they are computational complex, it often takes more time to train LSTM networks than simpler ones. Therefore, long training periods and sufficient data are typically required for a LSTM to train accordingly.
- The processing of the sentences occurs in a sequence by single word and hence, it is difficult to parallelize.

### **Several well-known uses for LSTM include:**

- There are numerous types of RNN, including among others LSTM – a particularly powerful variant of the RNN which has been successfully utilized in multiple areas. Here are a few well-known uses for LSTM:
- Language Modelling: The application of LSTMs to handle some natural language processing tasks such as text summarization, machine translation, and language modelling has been done. They can be made to understand the word relationship in a sentence so that they will generate meaningful sentences.
- Speech Recognition: LSTMs have also attempted to solve speech recognition issues including transcriptions of text-to-speech and spoken commands. We can teach them to

perceive sound patterns associating them with the corresponding text.

- **Time Series Forecasting:** The use of LSTMs in time series forecasting tasks, as well as energy usage, stock price, and weather prediction. Through this, they can predict the future as it is always possible to find patterns within time series data.
- **Anomaly Detection:** For instance, researchers have applied LSTMs with detecting different forms of corruption and network security compromise. These systems can also be trained to recognise peculiarities in the data and, flagging them accordingly.
- **Systems for Recommenders:** The application of LSTM to music, book, and film suggestion. They can detect how a user behaves and relate it with suggesting relevant options.
- **Video Analysis:** Video analysis has also been achieved using LSTM on tasks like object detection, activity recognition, and action classification. The CNNs along with these can also assist in analysing video for valuable information.

### 3.2 Project design and architecture

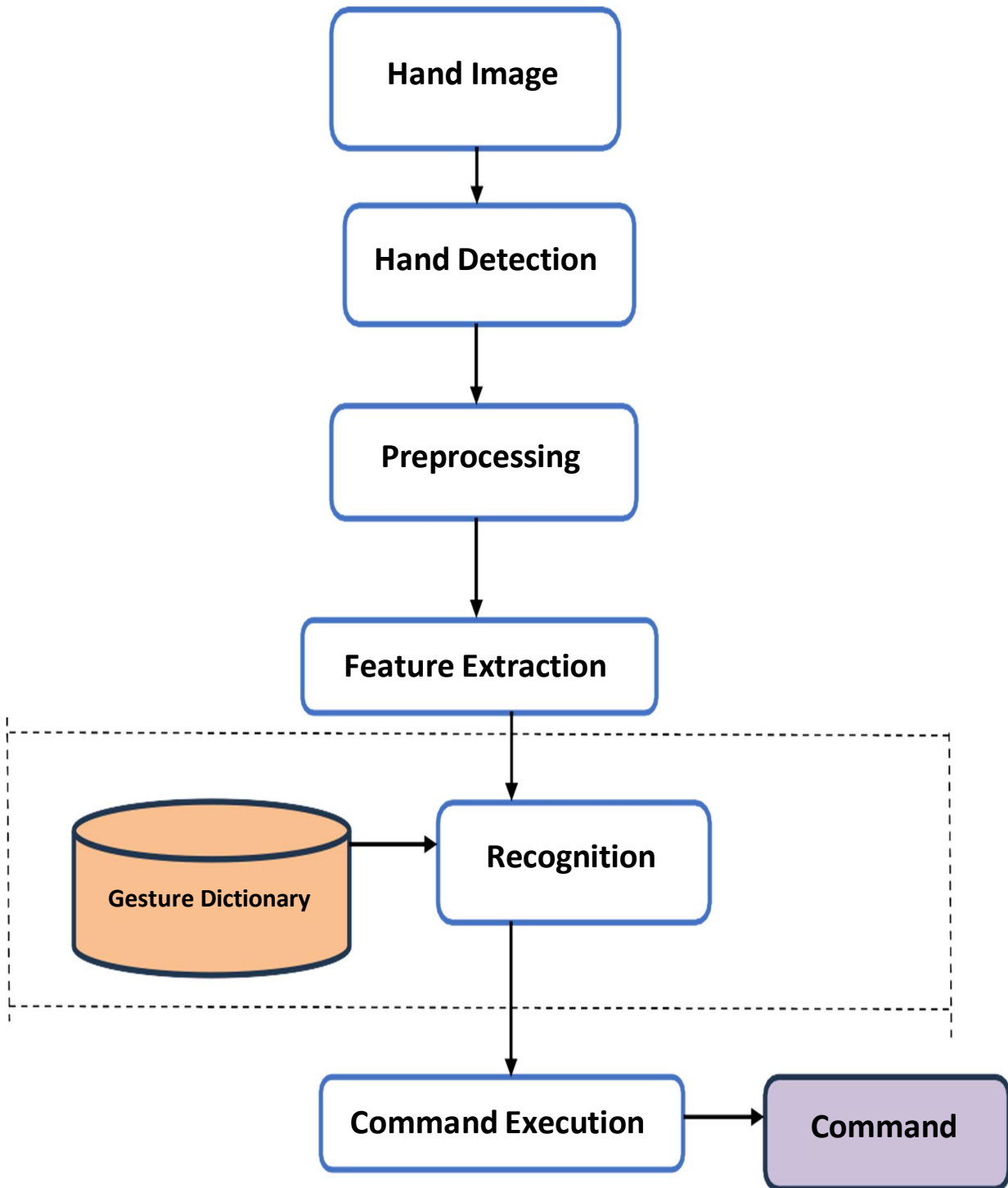
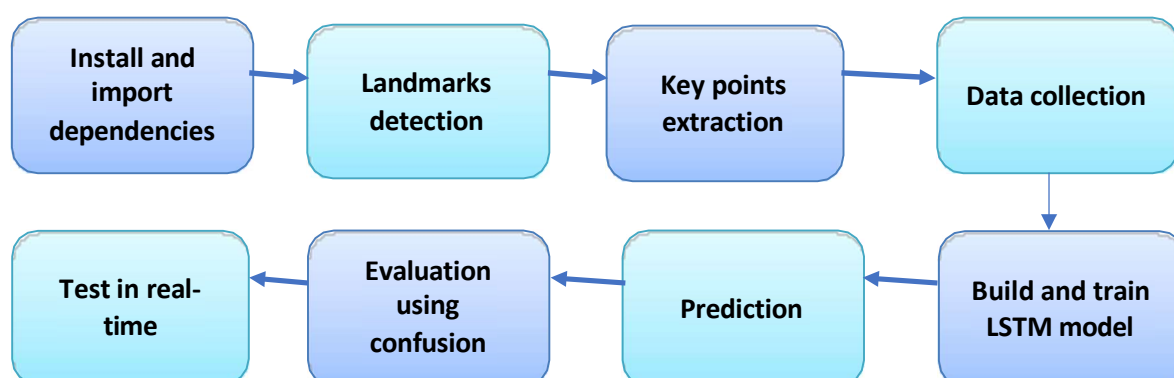


Fig 3.2.1. Basic flow of the project using simple keras model

This is an overview of how we extracted facial, posing, and hand points using MediaPipe and then used an LSTM neural network to predict the gesture:

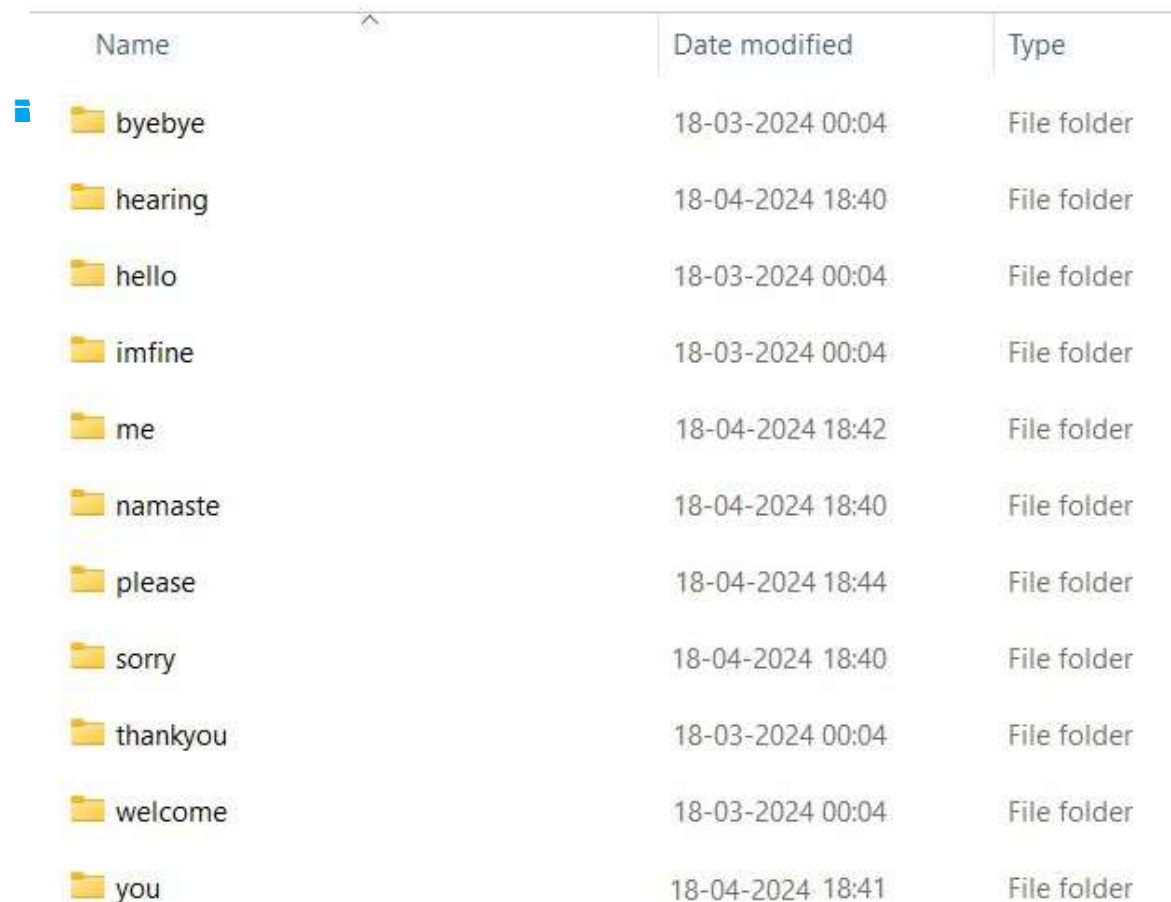
- A machine learning pipeline can be built using the MediaPipe framework to process time-series data, such as audio and video. For a wide range of applications, including face detection, multi-hand tracking, object detection and tracking, and many more, it offers pre-trained models and APIs.
- We employed a solution called MediaPipe Holistic, which integrates hand, pose, and face models into a single pipeline. For every one of these parts, it produces a set of landmarks that stand in for the important parts of the hands, body, and face.
- The landmarks for every frame were obtained by feeding the video frames into the MediaPipe Holistic pipeline. We then normalized and flattened the landmarks into a feature vector for each frame.
- To predict the gesture from the feature vectors, we employed an LSTM neural network. Long Short-Term Memory, or LSTM for short, is a kind of recurrent neural network that can pick up on long-term dependencies in the input. We chose LSTM because it can capture the temporal dynamics of the gestures, which are sequential in nature.
- Using a labeled dataset of gestures, each represented by a series of feature vectors, we trained the LSTM network. To obtain the probability distribution over the possible gesture classes, we applied a softmax activation function on the output layer.
- We assessed the accuracy, precision, recall, and F1-score of the LSTM network using a test set of gestures. In order to examine the network's performance, we also plotted the ROC curve and the confusion matrix.



**Fig 3.2.2. Detailed flow of the project using LSTM**

### 3.3 Data Preparation

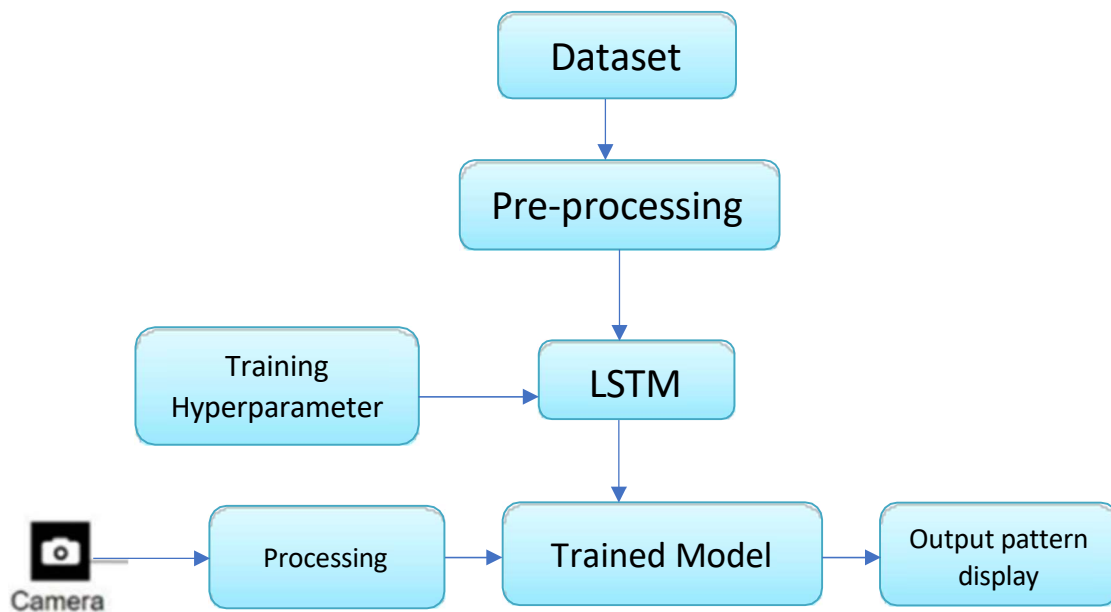
The first step is to create data collection folders - we gathered our own dataset for training and testing our deep learning model in the following manner: Using the `os.path.join('data folder name')` method, we created a folder for the dataset. We then defined a set of gestures ([“hello”, “byebye”, “welcome”, etc]) that will be used to train our recognition model. We specify the number of videos to be collected (30) and the number of frames (30) for each video to be captured. After we finish creating the gesture folder, the next step is to collect data folders- a loop is then run to create the 30 videos for each sign gesture. E.g. For each of the gesture folder there are video folders as shown in the figure below:



Name	Date modified	Type
byebye	18-03-2024 00:04	File folder
hearing	18-04-2024 18:40	File folder
hello	18-03-2024 00:04	File folder
imfine	18-03-2024 00:04	File folder
me	18-04-2024 18:42	File folder
namaste	18-04-2024 18:40	File folder
please	18-04-2024 18:44	File folder
sorry	18-04-2024 18:40	File folder
thankyou	18-03-2024 00:04	File folder
welcome	18-03-2024 00:04	File folder
you	18-04-2024 18:41	File folder

**Fig 3.3.1. folders of classes of gestures**

## 3.4 Implementation



**Fig 3.4.1. Architecture of the Hand Gesture Recognition System**

### 3.4.1 Importing and Installing Dependencies

Installing and importing certain dependencies was the first step. Six distinct dependencies have been installed by us:

- *tensorflow*
- *matplotlib* (to visualize images easily)
- *mediapipe* (to extract hand, face, pose landmarks)
- *OpenCV*
- *scikit-learn* (for splitting data into training and testing sets)
- *matplotlib* (for visualising images)

```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
import time
import mediapipe as mp
```

**Fig 3.4.1.1 List of libraries used**



### 3.4.2 Detecting Hand, Pose and Face landmarks (using MP Holistics)

First, we will confirm that OpenCV can correctly access our webcam. Essentially, all we are doing is preparing a video capture, after which we will render each and every frame on the screen. As a result, even though we are only looping through one frame, it will appear to be a video because we are actually stacking several frames together.

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

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

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

        # Draw landmarks
        draw_styled_landmarks(image, results)

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

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

Fig 3.4.2.1 Video capture using MP holistics

- *cap = cv2.VideoCapture(0)*
  - We can utilize OpenCV's function, specifying device value 0, to access our webcam.
- *cap.isOpen()*
  - Verify whether we're accessing the webcam before starting a loop.
- *cap.read()*
  - Obtain the present frame from the webcam, providing two results: a return value and the current frame.
- *cv2.imshow()*
  - Display the resulting frame on the screen.
- *cv2.waitKey()*
  - Pause until a key is pressed on the keyboard.

- `cap.release()`
  - Free up the webcam resources.
- `cv2.destroyAllWindows()`
  - This will shut down our frame.

We then make it unwritable(saves a little memory) and then do Detection and finally set it to Writable before converting it from RGB to BGR. Next, we carry our detection, change it back to writable, and rearrange its color scheme from RGB to BGR so as to take less space.

**def mediapipe\_detection(image, model)**

By default, OpenCV reads feeds in BGR channel format when they are received. However, the input must be in RGB format when utilizing the mediapipe to actually make a detection. So, we are going to make that transformation using OpenCV.

- `cv2.cvtColor()`
  - transforms an image into a different color. An input image is transformed from one color to another by this function. When converting from RGB to another color space, the channels' order needs to be stated clearly (RGB OR BGR).

We can employ Mediapipe for our detection process and subsequently project these identified keypoints onto the image. Currently, the landmarks in the frame are not visible. To address this, we will define a function:

**def draw\_landmarks(image, results)**

### 3.4.3 Extract Keypoint values

Up to this point, we've obtained landmarks for the left and right hands, facial features, and body positions.

To enhance resilience, particularly when no values are obtained, it's crucial to extract these landmarks in a more robust manner. To achieve this, we concatenate them into a NumPy array. In cases where no values are present, we create a NumPy zeros array with the same shape.

The input data for this action detection model comprises 30 arrays, each containing 1662 values (30, 1662). Each of these arrays represents the landmark values (1662 values) from a single frame. To streamline this process, we create separate placeholder arrays for the left hand, right hand, face, and pose. We extract the different landmarks for each keypoint and consolidate them into a flattened array using the function:

```
def extract_keypoints(results)
```

### **3.4.4 Setting up folders for data collection**

Our next step involves establishing folders for our array compilation, utilizing extracted keypoints to interpret sign language. We've employed 30 frames of data, encompassing 30 sets of 1662 keypoints to identify specific actions.

The distinctive aspect of action detection, as opposed to other computer vision tasks, lies in the use of a sequence of data rather than a single frame for detection. Essentially, we've gathered data for 10 unique actions, acquiring 30 videos for each action (e.g., hello, thanks, I love you). Each video sequence comprises 30 frames, and each frame is characterized by 1662 landmark values. In summary, our dataset comprises 10 actions, each with 30 video sequences, 30 frames per sequence, and 1662 landmarks per frame.

**Number of sequences=30 (number of videos)**

**Sequence length =30 (30 frames per video)**

Initially, we established the data path for storing our data and defined the "actions" variable to represent the various actions targeted for detection.

Subsequently, we implemented a loop iterating through the number of actions and sequences, creating the corresponding folders accordingly.

### **3.4.5 Gather sequences of keypoints.**

We are going to take a break between each video that is collected. Having breaks between each sequence collection allows us to reset and reposition our-self to collect the action from start to

finish. After taking the data input, the frames are stored as NumPy arrays at the data path location.

### 3.4.6 Preprocess and create data labels

To pre-process our data and create labels, first we are going to import 2 more dependencies:

- *train\_test\_split* from sklearn (Enables the segmentation of data into training and testing sets, facilitating training on one portion and evaluating on a distinct segment of our dataset.)
- *to\_categorical* function from keras utilities (To generate labels, especially beneficial when converting our data into one-hot encoded format.)

Subsequently, a label map will be established to denote each unique action. The process involves consolidating all the data, creating a comprehensive array that encompasses all information. In essence, this results in a collection of 10 sets, each containing 30 arrays. Within each of these arrays, there are 30 frames, and each frame is characterized by 1662 values representing our keypoints.

Initially, two empty arrays will be generated:

- *sequences* - represents our feature data or our X data
- *labels* - represent our labels or Y data

Therefore, we will be using these features to train a model that will capture the relationship between the labels.

We iterate through each action, then through each sequence, creating an empty array named "window" to encompass all frames of that particular sequence. Subsequently, for each frame, we utilize `numpy.load()` to load the frame.

The "sequences" array holds 300 different videos, with each one consisting of 30 frames. Following this, we initiate the preprocessing of the gathered data. To facilitate ease of manipulation, the sequences are stored within a NumPy array.

Moving forward, we conduct training and testing partitions by employing `train_test_split()` with a test size of 5%, segregating a subset of our data for testing purposes.

### 3.4.7 Build and train LSTM deep learning model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
model.fit(X_train, y_train, epochs=500, callbacks=[tb_callback])
model.summary()
```

Fig 3.4.7.1 LSTM structure

We will utilise TensorFlow and Keras to train our LSTM neural network.

- *sequential module* (Enable the construction of a sequential neural network.)
- *lstm layer* (Furnishes a temporal element for constructing our neural network, enabling action detection.)
- *dense layer* (A standard fully connected layer.)
- *tensorboard* (Enable logging within TensorBoard for tracking and monitoring the model during training, providing the option to trace its progress.)

Following this, establish a log directory and set up TensorBoard callbacks. Construct the neural network architecture, compile the model using the Adam optimizer and categorical crossentropy loss function, and proceed to fit the model.

Given the multi-class classification nature of the model, we utilize the Adam optimizer and categorical crossentropy loss during compilation. Subsequently, initiate the training process for the model.

### 3.4.8 Make sign language Prediction

By employing the `model.predict(X_test)` function, we make predictions regarding the classes of gestures captured through our webcam. This process involves utilizing the trained model to infer the most likely class labels for the input data in `X_test`. The predictions can then be further analyzed or used for various applications, such as gesture recognition or classification.

### 3.4.10 Evaluation using a confusion Matrix and accuracy

For this we will import a couple of metrics from sci-kit learn to evaluate the performance of our model:

- *Multi\_label\_confusion\_matrix*(gives us a confusion matrix for each one of our different labels. This allows us to evaluate what's being detected as a True Positive and a True negative and what's being detected as false positive and false negative.
- *accuracy\_score* (to evaluate accuracy)

### 3.4.11 Test in real time

Subsequently, in real-time scenarios, the model undergoes testing as it processes gestures performed in front of the webcam. During this phase, the model's predictive capability is put to the test by capturing live gestures and utilizing its learned patterns to forecast the corresponding gestures. This dynamic evaluation enables the model to make immediate predictions based on the gestures it observes, demonstrating its practicality and accuracy in recognizing and categorizing real-time actions.

## 3.5 Key Challenges

While hand gesture detection using deep learning is a powerful and versatile approach, it comes with certain limitations. Some of these limitations include:

### 3.5.1. Data Quality and Quantity:

- **Insufficient Data:** Deep learning models, especially complex ones like LSTMs, often require large amounts of labeled data for effective training. Limited data may result in overfitting or reduced model generalization.
- **Annotation Challenges:** Manually annotating hand gesture datasets can be time-consuming and may introduce human biases. Additionally, obtaining diverse datasets that encompass various lighting conditions, backgrounds, and hand shapes can be challenging.

### 3.5.2. Real-time Processing:

- **Computational Resources:** Real-time hand gesture detection demands significant computational resources. Running deep learning models on resource-constrained devices

may lead to latency issues, limiting their suitability for certain applications.

- **Power Consumption:** Deep learning models, particularly those involving complex architectures, can be power-intensive. This might be a concern for applications deployed on battery-powered devices.

### **3.5.3. Variability in Gestures:**

- **Gesture Diversity:** Hand gestures can vary widely among individuals and cultures. Building a model that accommodates this diversity can be challenging, and achieving high accuracy across all possible gestures is difficult.
- **Fine-grained Movements:** Capturing subtle or fine-grained hand movements accurately may require high-resolution input data and more sophisticated model architectures.

### **3.5.4. Environmental Factors:**

- **Lighting Conditions:** Changes in lighting conditions can impact the performance of hand gesture recognition systems, especially those relying on RGB-based data. Adapting to diverse lighting scenarios is a common challenge.
- **Background Interference:** Cluttered or dynamic backgrounds may interfere with the model's ability to focus on the hand, leading to false positives or negatives.

### **3.5.5. User Interaction Challenges:**

- **Ambiguity in Gestures:** Some gestures may be inherently ambiguous or similar to others, making it challenging for the model to accurately differentiate between them.
- **User Adaptation:** User-specific variations, such as hand size, shape, and motion patterns, may affect the model's performance. Adaptive techniques or user-specific fine-tuning may be necessary.

### **3.5.6. Robustness and Generalization:**

- **Limited Generalization:** Models trained on specific datasets may struggle to generalize to unseen scenarios or diverse user populations. Transfer learning or domain adaptation techniques may be required.
- **Noise Sensitivity:** Deep learning models can be sensitive to noisy input, which may result in erratic predictions. Preprocessing steps and robust architectures can help mitigate this.

Understanding these limitations is crucial for developers to make informed decisions during the design and deployment of hand gesture detection projects. Addressing these challenges often involves a combination of careful dataset curation, model optimization, and consideration of the specific application requirements.



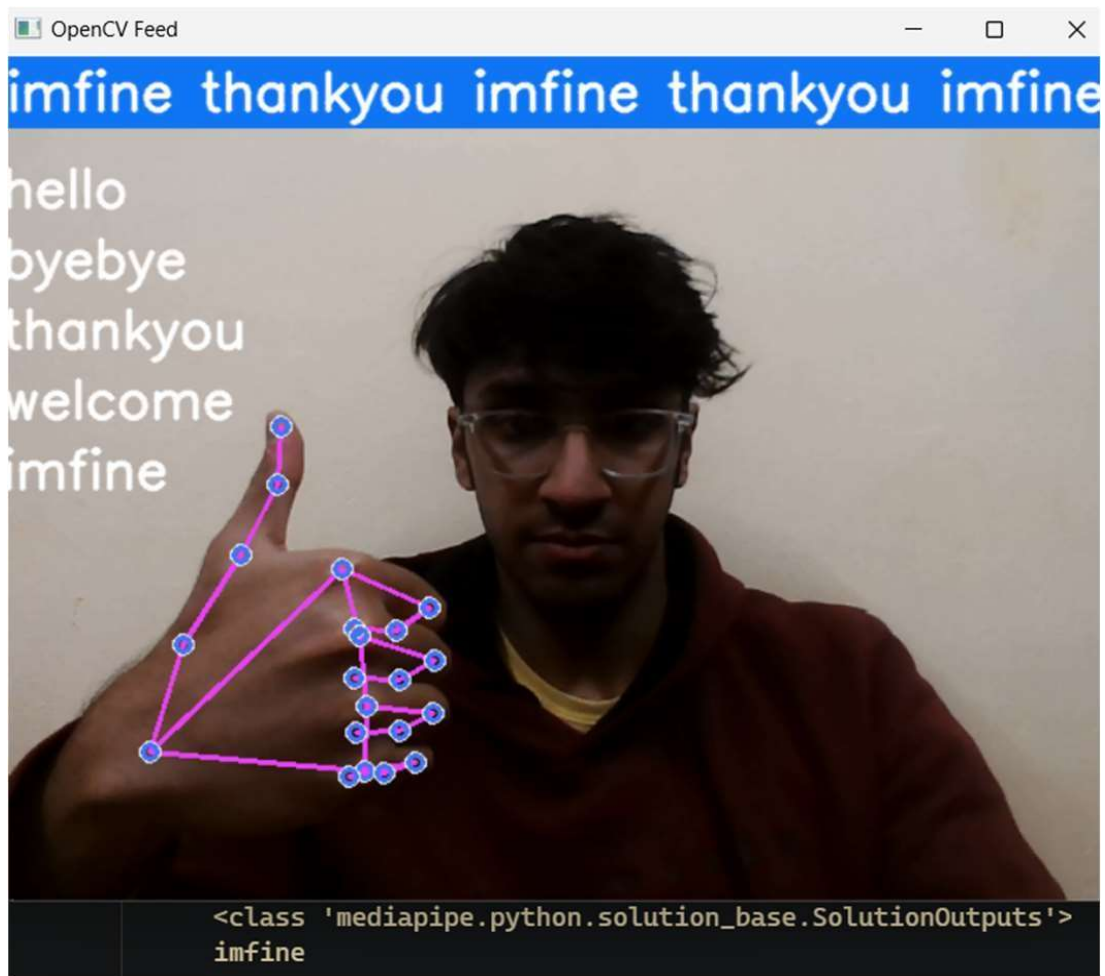
# Chapter 4: Testing

## 4.1 Test Cases and Outcomes

Text-based gesture recognition was demonstrated through a real-time feed. Testing was done in real time for each gesture, and the results looked good. Since our model is trained to make predictions based on 30 frames of important points, the video feed is processed as follows in order to enable it in real-time: It is defined that `sequence=[]` is an empty array. OpenCV is used to capture the real-time feed. A number of previously discussed techniques in this paper, including `mediapipe_detection` and `extraction_keypoints`, are used to carry out the detection and gather the key points in real time. The key points that have been gathered are added to the initial empty array that we defined. `Sequence = [-30]` is the new assignment made to the array. Consequently, it preserves the appended key points for the last thirty frames. The model is used to forecast the result when the sequence array's length is found to be thirty.



**Fig. 4.2.1 : Model prediction for gesture “Thank You”**

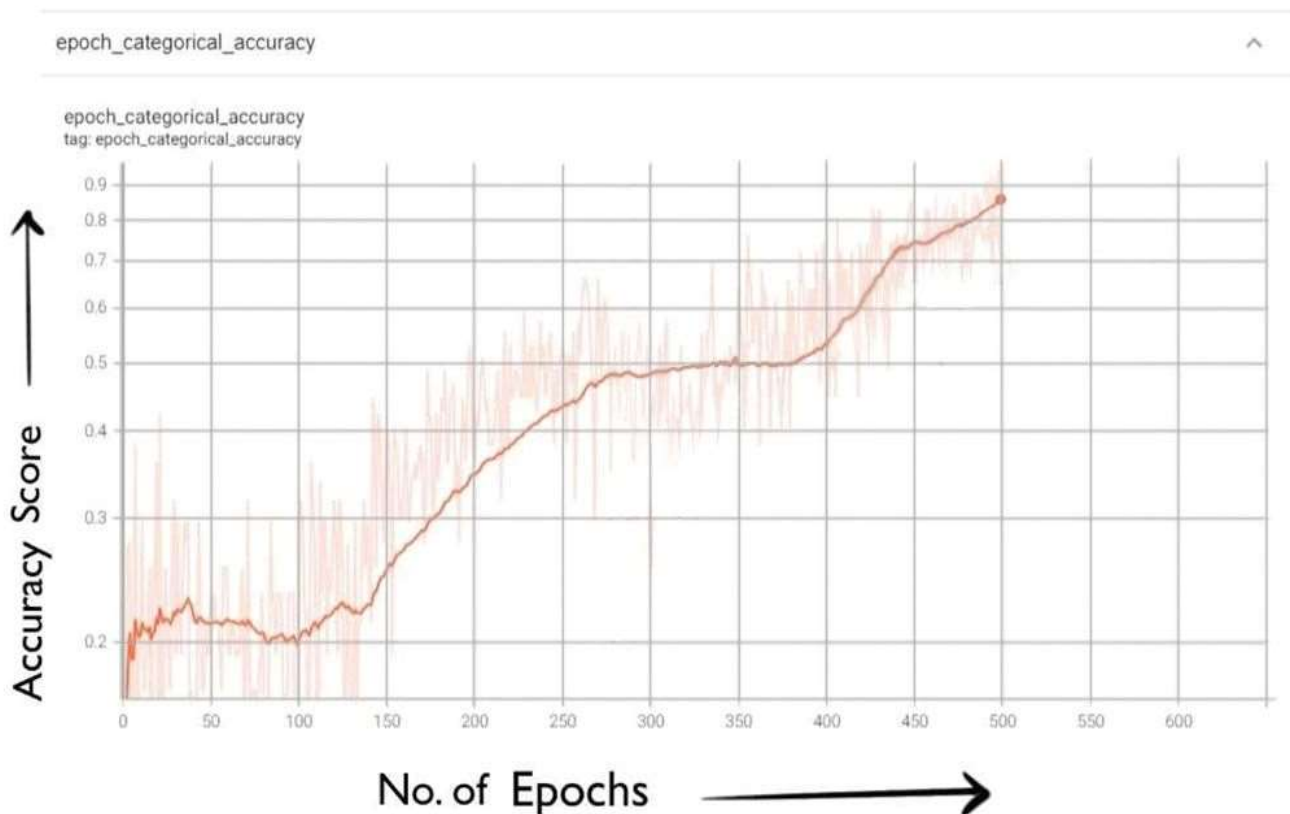


**Fig 4.2.2 : Model prediction for gesture “I’m fine”**

# Chapter 5: Results and Evaluation

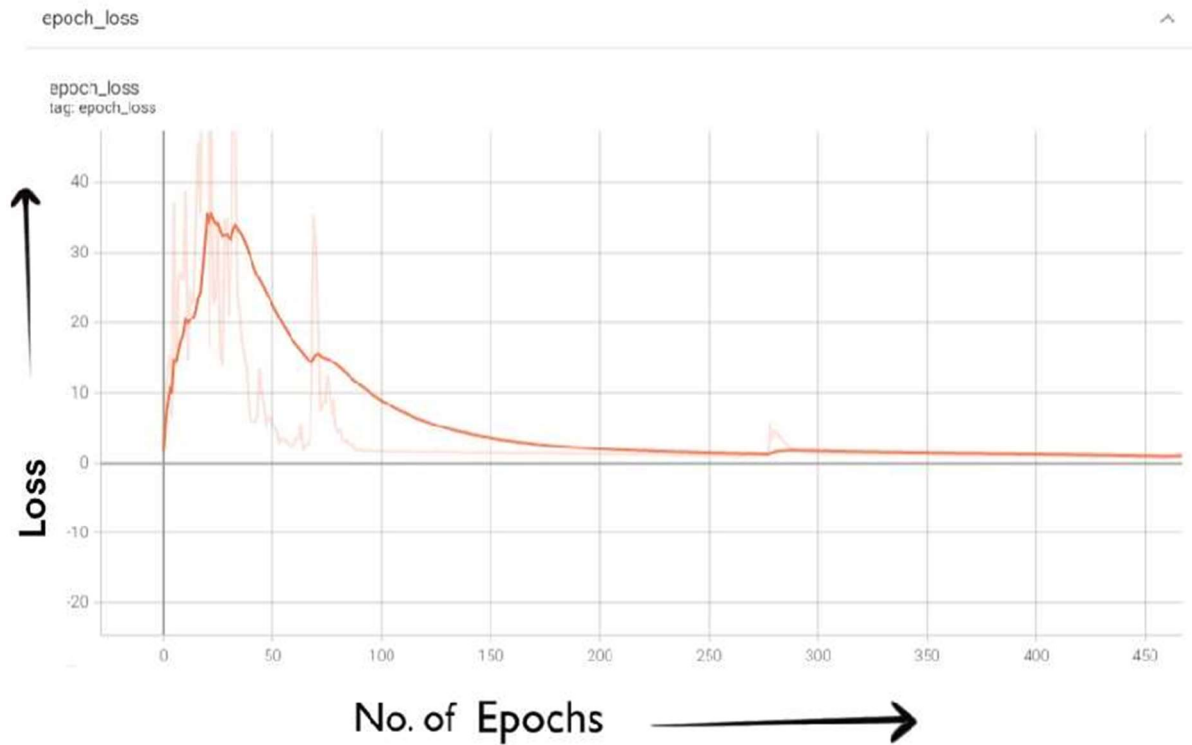
## 5.1 Results

The original dataset, which provided the basis for the testing dataset, was divided into the training and testing datasets. The preferred evaluation metric we use is accuracy. The LSTM model is working as intended.



**Graph 5.1.1: epoch\_categorical\_accuracy**

The LSTM (Long Short-Term Memory) model's accuracy increases to about 87% after 500 epochs because of the network's capacity to recognise complex temporal correlations in the data and its extensive training. The model picks up simple patterns in the early epochs, but as training goes on, it gets better at comprehending intricate sequences. Long-range dependencies can be addressed by the LSTM's improved internal memory cell adaptation made possible by this prolonged exposure. Fine-tuning and regularisation strategies might also be involved in the increase in performance. After this point, it's important to keep an eye out for overfitting because too many epochs could cause the model to become too adept at memorising the training set rather than effectively generalising.



**Graph 5.1.2 : epoch\_loss**

Iterative optimization during training epochs is how an LSTM (Long Short-Term Memory) model reduces loss. The loss shows the difference between the expected and actual numbers, and it starts off high. The model minimises this discrepancy as it uses backpropagation to improve its parameters. The architecture of the LSTM makes it possible to learn sequential relationships efficiently, which is essential for tasks like time series prediction. The model refines its memory cells throughout epochs, improving its capacity to recognise complex patterns. The use of regularisation procedures also aids in preventing overfitting. The model's increased ability to generalise and produce precise predictions on unobserved data is indicated by a declining loss.

# Chapter 6:

## Conclusions and Future Scope

### 6.1 Conclusion

In conclusion, our large-scale project on Action Detection for Sign Language Gestures has produced important discoveries and advances in the field of sign language recognition. We did this by utilising a self-created dataset and deep learning models. The main conclusions, limits, and contributions to the field are outlined in this section.

#### 6.1.1 Key Findings:

##### 6.1.1.1 Effective Utilization of Self-Created Dataset:

The project effectively illustrated the value of using a dataset that the user developed, enabling a more detailed and nuanced description of sign language gestures. This method made sure that a variety of expressions were included and enhanced the model's flexibility to accommodate various signing styles.

##### 6.1.1.2 Real-time Action Detection with LSTM:

Using LSTM in conjunction with cutting-edge tools such as TensorFlow Keras and Mediapipe allowed real-time action identification for sign language gestures. Particularly in the creation of assistive technologies and communication tools for people with hearing impairments, this skill is critical for real-world applications.

##### 6.1.1.3 Multimodal Approach for Holistic Recognition:

The multimodal method that took into account both visual and spatial characteristics was made possible by the integration of Mediapipe and TensorFlow Keras. This method was in line with the objective of developing a more comprehensive comprehension of sign language gestures, including non-manual elements.

#### 6.1.2 Limitations:

##### 6.1.2.1 Dataset Size and Diversity:

Although the self-created dataset turned out to be useful, its size and diversity constraints may limit the model's ability to generalise over a variety of sign languages and user variations.

### **6.1.2.2 Real-world Variability:**

There is still a problem with real-world variability even with attempts to handle external influences. Research and improvement must continue to guarantee the model's flexibility in the face of changing environmental circumstances.

## **6.2 Future Scope:**

The "Action Detection for Sign Language Gestures" big project's trajectory paves the way for a significant and far-reaching future. Researchers can advance sign language recognition and develop more inclusive and adaptable technology for people with hearing impairments by exploring these strategic avenues in greater detail.

### **6.2.1 Expansion of the Dataset:**

The potential for capturing the variety and diversity inherent in sign languages is enormous as the dataset is expanded. To create a dataset that takes into account regional variances, dialectical subtleties, and individual signing styles, researchers should work in conjunction with sign language communities, linguistic experts, and cultural representatives. This inclusive dataset guarantees that the model appeals to a wide range of user groups while also increasing the model's accuracy.

### **6.2.2 Incorporation of Additional Modalities:**

Adding other modalities improves the model's contextual awareness beyond visual signals. Understanding aspects of sign language like hand gestures, facial emotions, and even background details like the tone or tempo of the discourse can help with a more sophisticated interpretation. By using a comprehensive approach, the technology becomes more responsive and intuitive while reflecting the richness of human communication.

### **6.2.3 Enhancement of Real-time Processing:**

The model must be optimised for low latency and effective frame rate processing in order to achieve real-time processing excellence. Utilising the capabilities of devices like smartphones or wearables for real-time sign language detection, researchers can investigate cutting-edge methods in edge computing. For applications like assistive technology or live translations that need real-time communication, this optimisation is essential.

### **6.2.4 Cross-linguistic Adaptability:**

In order to attain cross-linguistic flexibility, scholars ought to undertake an extensive process of training the model across numerous sign languages. In addition to comprehending the linguistic subtleties, this work entails grasping the cultural background that is inherent in each sign language. The objective is to develop a global model that can easily adjust to the many signing customs found all over the world.

#### **6.2.5 User-specific Customization:**

Personalised models can be designed to meet the individual tastes of each user, taking into account the distinctiveness of each signer. Improved intuitiveness and user-friendliness can be achieved using machine learning algorithms that can adjust to the unique characteristics of each person's signing gestures. Enhancing total communication efficacy, this personalization helps users and technology interact on a deeper level.

#### **6.2.6 Integration of Edge Computing:**

Accessible sign language recognition depends on the incorporation of edge computing technologies. Scholars can concentrate on refining the model for implementation on peripheral devices, guaranteeing that the technology becomes transportable and accessible. This strategy advances the democratisation of assistive technologies and is consistent with the growing trend of decentralised computing.

#### **6.2.7 Evaluation in Real-world Environments:**

Extensive testing in real-world circumstances is necessary to improve the resilience of the model. Extensive assessments should be carried out in a variety of settings, such as changing lighting, intricate backgrounds, and individual differences in signing. This in-person testing guarantees the model's dependability in real-world scenarios by simulating the erratic nature of regular communication.

#### **6.2.8 Collaborative Initiatives and Benchmarking:**

Taking part in joint projects with the larger scientific community encourages a group approach to progress. When models are compared to common standards in benchmarking exercises, researchers should take an active part in the process. Setting standards for performance evaluation promotes openness, encourages healthy competition, and quickens the field's overall advancement.

## References

- [1] M. A. S. M. Miah, J. Shin, Mr. M. Atiq ul Islam, Md. Azizur Rehman and Mr. Y. Okuyama, “Rotation translation and scale invariant sign word recognition using deep learning”, Syst. Sci. Eng., vol. 44, no. 3, pp. 2521-2536, 2023.
- [2] Mahesh A. Khan, Mukesh Mittal, Leena M. Goyal and Santosh Roy, “A deep survey on supervised learning-based human detection and activity classification”, in: Multimedia Tools & Applications, vol. 80, no 18, pp. 27867-27923, Jul. 2021.
- [3] L. Shi, Y. Zhang, J. Cheng, H. Lu “Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition”. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 12026-12035, Jun. 2019.
- [4] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor and J. F. Vélez (2018). Convolutional Neural Networks and Long Short-Term Memory for S
- [5] A.Nandy, J.Prasad ,S.Mondal, P.Chakraborty & G.Nandi.(2016) “Real-Time Recognition Of Isolated Indian Sign Language Gesture,” Commun. Comput. K., Bennani & Yousfi, (2010). Inf. Sci.\*, vol. 70, pp. 102–107.
- [6] P. Mekala, Yanan Gao, Jin Fan, and Amir Davari, “Real-time Sign Language Recognition Based on Neural Network Architecture” in Proceedings of the IEEE 43rd Southeastern Symposium on
- [7] J K Chen, “Sign language Recognition with unsupervised feature learning,” CS229 project final report, Stanford University, Stanford, CA, USA, 2011 [7].
- [8] “Indian Sign Language Recognition Using Neural Networks & KNN Classifiers” by M. Sharma et. al. The International Journal of Obesity & related Metabolic disorders:



Sci., Vol.9(12), 1255-1259, 2014.

[9] S. R. Agarwal, S. B. Agrawal, and A. M. Latif, "Article: "Sentence formulation in NLP engine with the help of Indian sign language based on hand gestures," Int. J. Comput. In J Appol. 116: 18–22, 2015.

[10] Shazalwar, S. S. and Shrawankar, U., "Interpretation of sign language into English using NLP techniques," J. Inf. Optim. Sci., vol. 38, pp. 895–9710, 2017.

[11] S. Shivashankara & S. Srinath, "American Sign Language Recognition System: An Optimal Approach," International Journal of Image Graph and Signal Processing, vol. 10, pp. 18–30, 2018.

[12] Narciso Camgoz, Shadfield, O. Koller, Ney, H., and R. Bowden, "Neural Sign Language Translation," in \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018, 18–22 June 2018, Salt Lake City, UT, USA; IEEE: Piscataway, NJ, USA, 2018.

[13] "Real-Time Recognition of Indian Sign Language," by H. Muthu Mariappan and V. Gomathi, in Proceedings of the International Conference on Computational Intelligence in Data Science, September 6–7, 2019, Haryana, India.

[14] "A Modified LSTM Model for Continuous Sign Language Recognition Using Leap Motion," A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian, and B. B. Chaudhuri, IEEE Sens. J., vol. 19, pp. 7056–7063, 2019.

[15] M. De Coster, J. Dambre, and M. V. Herreweghe, "Sign Language Recognition with Transformer Networks," in Proceedings of the Language Resources and Evaluation (LREC 2020), Marseille, France, May 13–15, 2020, pp. 6018–6024.

[16] S. Jiang et al., \*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition\*, Nashville, TN, USA, June 21–24, 2021; pp. 3413–3423. "Skeleton aware multi-modal sign language recognition."

- [17] Y. Liao and colleagues, "Dynamic Sign Language Recognition Based on Video Sequence with BLSTM-3D Residual Networks," *IEEE Access\**, vol. 7, 2019, pp. 38044–38054.
- [18] N. Adaloglou and T. Chatzis, *\*IEEE Trans. Multimed.\**, vol. 24, pp. 1750–1762, 2022, "A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition."
- [19] Aparna, C. and Geetha, M., "CNN and Stacked LSTM Model for Indian Sign Language Recognition," *\*Commun. Comput. Inf. Sci.\**, vol. 1203, pp. 126–134, 2020.
- [20] Alemi, A. A., Szegedy, C., and Vanhoucke, S., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *\*arXiv 2016\**, arXiv:1602.07261/.
- [21] D. Yang and colleagues, "Deep learning techniques for COVID-19 detection and analysis in medical images," *\*Sci. Rep.\**, vol. 11, p. 19638, 2021. [PubMed] [CrossRef].
- [22] "Transformer-XL: Attentive language models beyond a fixed-length context", arXiv:1901.02860, 2019, Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov.
- [23] "Skeleton-based dynamic hand gesture recognition using an enhanced network with one-shot learning," C. Ma, S. Zhang, A. Wang, Y. Qi, and G. Chen, *Appl. Sci.*, vol. 10, no. 11, pp. 3680, May 2020.
- [24] *Proc. Int. Conf. 3D Vis. (3DV)*, pp. 617-626, Dec. 2021; A. Bigalke and M. P. Heinrich, "Fusing posture and position representations for point cloud-based hand gesture recognition".
- [25] *Visual Computing*, vol. 34, no. 6, pp. 1053-1063, June 2018, C. Ma, A. Wang, G. Chen, and C. Xu, "Hand joints-based gesture recognition for noisy dataset using nested interval unscented Kalman filter with LSTM network".
- [26] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [27] Google Research. "MediaPipe: A Framework for Building Perception Pipelines." v0.8.3, 2021. [Online]. Available: <https://github.com/google/mediapipe>

[28] <https://www.geeksforgeeks.org/understanding-of-lstm-networks/>

[29] <https://www.startasl.com/top-10-25-american-sign-language-signs-for-beginners-the-most-know-top-10-25-asl-signs-to-learn-first/>

[30] <https://www.geeksforgeeks.org/face-and-hand-landmarks-detection-using-python-mediapipe-opencv/>

# JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

## PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_ Name

of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

### UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

### FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at.....(%). Therefore, We are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

### FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

.....

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File)through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## Project\_Report\_Amit

### ORIGINALITY REPORT

<b>15%</b>	<b>13%</b>	<b>%</b>	<b>7%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>www.ir.juit.ac.in:8080</b> Internet Source	<b>4%</b>
<b>2</b>	<b>mdpi-res.com</b> Internet Source	<b>2%</b>
<b>3</b>	<b>www.mdpi.com</b> Internet Source	<b>1%</b>
<b>4</b>	<b>Submitted to Hanoi University</b> Student Paper	<b>1%</b>
<b>5</b>	<b>www.researchgate.net</b> Internet Source	<b>1%</b>
<b>6</b>	<b>Submitted to Jaypee University of Information Technology</b> Student Paper	<b>&lt;1%</b>
<b>7</b>	<b>www.arxiv-vanity.com</b> Internet Source	<b>&lt;1%</b>
<b>8</b>	<b>ieeexplore.ieee.org</b> Internet Source	<b>&lt;1%</b>
<b>9</b>	<b>journal2.uad.ac.id</b> Internet Source	<b>&lt;1%</b>

43	Submitted to Universidad Nacional de Educación a Distancia Student Paper	<1 %
44	cdn.iiit.ac.in Internet Source	<1 %
45	dokumen.pub Internet Source	<1 %
46	ebin.pub Internet Source	<1 %
47	mediatum.ub.tum.de Internet Source	<1 %
48	www.degruyter.com Internet Source	<1 %
49	www.eurchembull.com Internet Source	<1 %
50	www.hillclimber.ai Internet Source	<1 %
51	www.ijana.in Internet Source	<1 %
52	www.journaltocs.ac.uk Internet Source	<1 %

Exclude quotes

Off

Exclude matches

Off

10	<a href="http://www.irejournals.com">www.irejournals.com</a> Internet Source	<1 %
11	<a href="http://epubs.surrey.ac.uk">epubs.surrey.ac.uk</a> Internet Source	<1 %
12	<a href="http://export.arxiv.org">export.arxiv.org</a> Internet Source	<1 %
13	<a href="http://www.google.com">www.google.com</a> Internet Source	<1 %
14	Submitted to University of Hertfordshire Student Paper	<1 %
15	<a href="http://anshumitts.github.io">anshumitts.github.io</a> Internet Source	<1 %
16	<a href="http://www2.mdpi.com">www2.mdpi.com</a> Internet Source	<1 %
17	Submitted to South Eastern University of Sri Lanka Student Paper	<1 %
18	<a href="http://mededu.jmir.org">mededu.jmir.org</a> Internet Source	<1 %
19	<a href="http://profs.info.uaic.ro">profs.info.uaic.ro</a> Internet Source	<1 %
20	Submitted to University of Essex Student Paper	<1 %
21	<a href="http://healthdocbox.com">healthdocbox.com</a> Internet Source	<1 %

		<1 %
22	Submitted to British University In Dubai Student Paper	<1 %
23	link.springer.com Internet Source	<1 %
24	Submitted to GLA University Student Paper	<1 %
25	encyclopedia.pub Internet Source	<1 %
26	Submitted to Dr. B R Ambedkar National Institute of Technology, Jalandhar Student Paper	<1 %
27	aisel.aisnet.org Internet Source	<1 %
28	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
29	digitalcollection.zhaw.ch Internet Source	<1 %
30	www.coursehero.com Internet Source	<1 %
31	www.semanticscholar.org Internet Source	<1 %



31	<a href="http://journal.esj.edu.iq">journal.esj.edu.iq</a> Internet Source	<1 %
32	<a href="http://viso.ai">viso.ai</a> Internet Source	<1 %
33	<a href="http://wiredspace.wits.ac.za">wiredspace.wits.ac.za</a> Internet Source	<1 %
34	<a href="http://www.ijritcc.org">www.ijritcc.org</a> Internet Source	<1 %
35	<a href="#">Gagandeep Kaur, Ritesh Sinha, Puneet Kumar Tiwari, Srijan Kumar Yadav, Prabhash Pandey, Rohit Raj, Anshu Vashisth, Manik Rakhra.</a> "Face Mask Recognition System using CNN Model", Neuroscience Informatics, 2021 Publication	<1 %
36	<a href="http://www.escholar.manchester.ac.uk">www.escholar.manchester.ac.uk</a> Internet Source	<1 %
37	<a href="http://elibrary.almaata.ac.id">elibrary.almaata.ac.id</a> Internet Source	<1 %
38	<a href="#">Sang-Heon Lee, Myoung-Kyu Sohn, Dong-lu Kim, Byungmin Kim, Hyunduk Kim.</a> "Face recognition of near-infrared images for interactive smart TV", Proceedings of the 27th Conference on Image and Vision Computing New Zealand, 2012 Publication	<1 %

39	<a href="http://www-emerald-com-443.webvpn.sxu.edu.cn">www-emerald-com-443.webvpn.sxu.edu.cn</a> Internet Source	<1%
40	<a href="http://www.jetir.org">www.jetir.org</a> Internet Source	<1%
41	Shunji Hashiguchi, Taro Shibasaki. "A Task Estimation Method Based on Image Recognition and Its Application to EMG Prosthetic Hand Control", 2023 9th International Conference on Virtual Reality (ICVR), 2023 Publication	<1%
42	<a href="http://f5.pm">f5.pm</a> Internet Source	<1%
43	<a href="http://web.stanford.edu">web.stanford.edu</a> Internet Source	<1%
44	<a href="http://www.researchgate.net">www.researchgate.net</a> Internet Source	<1%
45	<a href="http://labstic.univ-guelma.dz">labstic.univ-guelma.dz</a> Internet Source	<1%