RECOGNITION OF SIGN LANGUAGE TO TEXT BY USING DEEP LEARNING

Dissertation submitted

to Shri Ramdeobaba College of Engineering & Management, Nagpur

in partial fulfillment of requirement for the award of degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

Diksha Gupta **Juhie Sayyed**

Piyush Nandha **Swyam Laira**

Guide

Dr. Pravin Sonsare

RCOEM

Shri Ramdeobaba College of Engineering and Management, Nagpur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Shri Ramdeobaba College of Engineering & Management, Nagpur 440013

(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

DECEMBER 2023

SHRI RAMDEOBABA COLLEGE OF ENGINEERING &

MANAGEMENT , NAGPUR

(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Thesis on **"RECOGNITION OF SIGN LANGUAGE TO TEXT BY USING DEEP LEARNING "** is a bonafide work **Diksha Gupta , Juhie Sayyed , Piyush Nandha , Swyam Laira** submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfillment of the award of a Degree of Bachelor of Technology, in Computer Science and Engineering. It has been carried out at the Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year **2023-2024**.

Date : 09/12/2023

Place: Nagpur

Dr. Pravin Sonsare
Project guide
Department of CSE

Dr. R. Hablani
H.O.D
Department of CSE

Dr. R. S. Pande
Principal

DECLARATION

We hereby declare that the thesis titled **"Recognition Of Sign Language To Text By Using Deep Learning**" submitted herein, has been carried out in the Department of Computer Science and Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other institution / University.

Date: 09/12/2023

Place: Nagpur

DIKSHA GUPTA (27) **JUHIE SAYYED (28)**

PIYUSH NANDHA (51) **SWYAM LAIRA (67)**

APPROVAL SHEET

This report entitled **"Recognition Of Sign Language To Text By Using Deep Learning**" by Diksha Gupta, Juhie Sayyed, Piyush Nandha, Swyam Laira is approved for the degree of Bachelor of Technology Computer Science.

Name & Signature
(Supervisor)

Name & Signature
(External Examiner)

Dr. R. Hablani
(HOD, CSE)

Date : 09/12/2023

Place: Nagpur

ACKNOWLEDGEMENT

We would like to place on record my deep sense of gratitude to our guide, **Dr. Pravin Sonsare** Department of Computer Science and Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur for his generous guidance, helpful and useful suggestions, continuous encouragement and supervision throughout the course of present work.

We express our sincere gratitude to **Dr. R. Hablani** ,Head of Department of Computer Science and Engineering for his stimulating guidance. The success of any work depends on efforts of many individuals. We would like to take this opportunity to express our deep gratitude to all those who extended their support and have guided us to complete this project work.

We are extremely thankful to **Dr. R. S. Pande ,** Principal of Shri Ramdeobaba College of Engineering & Management, Nagpur for providing us infrastructural facilities work in, without which this work would not have been possible**.**

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Automatic conversion of sign language to text is indeed helpful for interaction between deaf or mute people with people who even do not have knowledge of sign language . Sign language is a visual language that uses hand gesture, change of hand shape and track information to express meaning, and is the main communication tool for people with hearing and language impairment .There are about **72 million** deaf people who use sign language as their first language or mother tongue. Deaf and Dumb people rely on sign language interpreters for communications. Not all ordinary people understand the language. The translation of sign language into the alphabet/text automatically will facilitate the communication of the deaf or dumb with ordinary people . People, who are not deaf, never try to learn the sign language for interacting with the deaf people. This leads to isolation of the deaf people. "**Sign-to-text using mediapipeline**" is an innovative research driven technology. It combines advanced computer vision and natural language processing to convert sign language gestures into real-time written text.

CHAPTER 1
INTRODUCTION

1. INTRODUCTION :

Sign languages are developed primarily to aid deaf and dumb people. They use a concurrent and specific combination of hand movements, hand shapes and orientation in order to convey particular information. Sign language uses the visual faculties which is different from spoken language. The sign language is used widely by people who are deaf-dumb; these are used as a medium for communication. A sign language is nothing but composed of various gestures formed by different shapes of hand, its movements, orientations as well as the facial expressions. These gestures are generally used by deaf-dumb people in order to express their thought. Dumb-deaf persons faces communication barrier in public places while interacting with normal person, such as in bank, hospital and post offices. Sometimes the deaf needs to seek the help of the sign language interpreter so as to translate their thoughts to normal people and vice versa. However, this way turns out to be very costly and does not work throughout the life period of a deaf person. So a system which can automatically recognize the sign language gestures becomes a necessity. Introducing such a system would lead to minimize the gap between deaf and normal people in the society.. These systems utilize a combination of computer vision, machine learning, and sensor-based technologies to capture and analyze sign language gestures, enabling real-time translation into text or speech. By enabling effective communication between signers and non-signers, sign language to text recognition systems have the potential to enhance inclusivity and accessibility for the deaf and hard-of-hearing community in various domains, including education, healthcare, and everyday interactions.

1.1 Existing Systems:

Sign language to text recognition systems exist in various forms, ranging from devices like Leap Motion and Kinect that track hand and body movements to specialized gloves such as Sign Aloud that capture hand gestures and transmit the data to a computer. These systems utilize computer vision, machine learning, and depth sensing technologies to interpret sign language and convert it into text or speech. Additionally, researchers have developed sign-to-text systems that employ camera-based video capture and advanced algorithms to recognize and interpret sign language gestures. Mobile applications also offer sign language recognition features using smartphone cameras and machine learning. However, despite progress, accurate and reliable recognition in real-world scenarios remains challenging due to the complexity and variability of sign language.

1.2 Motivation:

The motivation behind this work is the possibility of reducing the communications barrier which exists between the deaf and hearing communities. Although gesture-based communication is used all over the world to overcome correspondence barriers for hearing impaired people who rely on gesture-based communication for most of their correspondence, there are no effective models that convert text to Indian communication by gestures. There is a scarcity of authentic and effective good assistance for correspondence. While significant progress has been achieved in computer recognition of communications by gestures of many nations, little effort has been done in ISL computerization. Our project aims at taking the basic step in bridging the communication gap between normal people and deaf and dumb people using Indian sign language. Effective extension of this project to words and common expressions may not only make the deaf and dumb people communicate faster and easier with outer world, but also provide a boost in developing autonomous systems for understanding and aiding them.

1.3 Proposed Systems:

The proposed system would be a real time system wherein live sign gestures would be processed using image processing. Then classifiers would be used to differentiate various signs and the translated output would be displaying text. Deep Learning algorithms will be used to train on the data set. The purpose of the system is to improve the existing system in this area in terms of response time and accuracy.

1.4 Features of proposed systems:

- **Gesture Recognition:** The system should be capable of accurately capturing and interpreting different sign language gestures.
- **Multi-modal Data Input:** The system should be able to accept inputs from live camera feeds.
- **Deep Learning and Pattern Recognition:** The system can utilize Deep learning techniques, such as deep neural networks, to train models that can recognize and classify sign language gestures.
- **Real-time Recognition:** To facilitate smooth communication, the system should aim to provide real-time recognition of sign language gestures. This allows users to receive

immediate feedback and helps ensure effective and efficient communication between sign language users and non-sign language users.

- **Text Output:** The system should convert recognized sign language gestures into text form, enabling non-sign language users to understand and respond to the communicated message. The output text can be displayed on a screen, sent as a text message, or integrated into other communication platforms.

CHAPTER 2
LITERATURE SURVEY

Mahesh Kumar's survey delves into the fascinating realm of converting sign language into text, exploring two prominent methodologies: Glove-based approaches and Vision-based approaches. The former involves signers donning a sensor or colored glove, simplifying the segmentation process during the conversion. However, a notable drawback arises as signers must bear the burden of wearing sensor hardware along with the glove throughout system operation. Despite this inconvenience, Glove-based approaches offer a tangible solution to bridging the gap between sign language and text.

On the other hand, Vision-based approaches leverage image processing algorithms to detect and track hand signs and facial expressions of the signer. An attractive feature of this technique is its user-friendly nature, as signers are relieved of the need to wear additional hardware. Yet, the methodology grapples with accuracy issues inherent in image processing algorithms. These concerns remain a critical hurdle that researchers are diligently working to address, aiming to enhance the precision and reliability of Vision-based conversion systems.

In the realm of Glove-based approaches, the tangible interaction facilitated by the glove simplifies the segmentation process, contributing to the effectiveness of the conversion. However, the imposition of wearing additional sensor hardware stands as a practical challenge that warrants consideration in the pursuit of seamless sign language-to-text translation. Meanwhile, Vision-based approaches, despite their user-friendly appeal, face a different set of challenges associated with refining image processing algorithms to achieve optimal accuracy. Kumar's survey sheds light on the current state of these methodologies, emphasizing the need for ongoing research and innovation to overcome existing limitations and propel the field of sign language conversion into text towards greater accessibility and inclusivity..

Sugandhi, Parteek Kumar, and Sanmeet Kaur have collaboratively developed a groundbreaking Sign Language Generation System focusing on the Indian Sign Language (ISL). The primary objective of this innovative system is to facilitate seamless communication between individuals with hearing impairments and the wider community. By translating English text into ISL, the system effectively serves as a valuable tool for human-computer interaction, negating the necessity for an ISL human interpreter.

The methodology employed in constructing this system is robust and multifaceted. It boasts an extensive corpus of English words and frequently used sentences, ensuring a comprehensive coverage of communication scenarios. Key components integral to the system's functionality include an ISL parser, the Hamburg Notation System, the Signing Gesture Mark-up Language, and a sophisticated 3D avatar animation module. These elements collectively contribute to the generation of Sign Language animations in adherence to ISL grammar.

In essence, the collaborative effort of Sugandhi, Parteek Kumar, and Sanmeet Kaur has resulted in a transformative system that not only addresses the communication needs of the hearing-impaired but

also exemplifies the convergence of linguistic expertise, technological innovation, and a deep understanding of the intricacies of Indian Sign Language.

In their groundbreaking research on sign language conversion to text and speech, Medhini Prabhakar, Prasad Hundekar, and Shivam Tiwari employ a sophisticated methodology centered around the MediaPipe framework. This innovative approach relies on the detection of hand gestures, a critical element in sign language interpretation. The researchers leverage several well-established models, including Convolutional Neural Networks (CNN), Faster-Convolutional Neural Networks (FRCNN), YOLO (You Only Look Once), and the Media Pipe model, to classify the detected hand gestures accurately.

The intricate process involves training these models to recognize and interpret sign language, a task that demands a nuanced understanding of the subtleties inherent in hand movements. The researchers note that, while the FRCNN model exhibits a commendable recognition rate, the overall computation time exceeds expectations due to the unavailability of high-end GPU hardware. Despite this limitation, the system remains acceptable for practical use cases.

In contrast, the CNN model demonstrates faster recognition of hand gestures suitable for real-world applications. However, there is a trade-off between speed and accuracy, with the CNN model sacrificing a degree of identification precision. The researchers grapple with the challenge of achieving optimal performance within the constraints of computational resources, acknowledging that achieving a balance between speed and accuracy is an ongoing pursuit.

The research culminates in the generation of text descriptions in the English language based on the recognized hand gestures. Moreover, the system incorporates a speech conversion component, adding an extra layer of accessibility for individuals with hearing impairments. Despite encountering challenges related to computational resources, the research significantly contributes to the advancement of sign language interpretation technology, showcasing the potential for real-world applications and fostering inclusivity through the marriage of computer vision and natural language processing.

In the paper titled "American Sign Language Recognition by Using 3D Geometric Invariant Feature and ANN Classification," the authors propose an innovative method for American Sign Language (ASL) recognition. Their approach involves designing a glove with six distinct color markers, which are automatically detected using the Circle Hough Transform. The 2D coordinates extracted from these markers, captured by two cameras, are then utilized to derive the 3D coordinates through Direct Linear Transformation (DLT). The authors construct all possible triangle patches from marker triplets, sorting them systematically. The sequences of areas generated from these patches serve as input for a feedforward backpropagation Artificial Neural Network (ANN) used for feature classification. The experimental results showcase an impressive average accuracy of 95 percent. However, a limitation

is noted, as the system requires the ability to work with dynamic postures and operate in real-time, presenting a challenge for practical implementation.

The paper "American Sign Language Recognition using Deep Learning and Computer Vision" proposes a model that leverages video sequences to recognize American Sign Language (ASL) gestures. Spatial features are extracted using Inception, a Convolutional Neural Network (CNN), while temporal features are captured through training with a Recurrent Neural Network (RNN). The model utilizes the American Sign Language Dataset for training. However, challenges arise from facial features, skin tones, and clothing variations. Testing with different skin tones affects accuracy, and the model struggles with facial diversity. To address this, videos were trimmed to focus on gestures extending up to the neck, mitigating inaccuracies introduced by varying facial features. Additionally, maintaining consistency in clothing, specifically using a full-sleeved shirt, was employed to stabilize model performance. Despite these challenges, the paper highlights the need for further exploration, suggesting potential improvements such as region-of-interest (ROI) isolation for hand gestures to enhance accuracy in ASL recognition.

CHAPTER 3
SOFTWARE REQUIREMENTS

3.1 PROJECT SCOPE

The main aim of the project is to create an AI based application which is used to identity the text conversion of sign language.

In future the system can be use to develop an application for chatting between normal and deaf people as well as providing a live translation from recognized gesture to users language

DESIGN AND IMPLEMENTATION CONSTRAINTS

For the development of this web-application system following is used.

- Python 3
- Pycharm
- Github
- Vscode
- MediaPipeline

3.2 SYSTEM FEATURES

- Use by deaf and dumb community persons and normal persons to interact or share knowledge.
- Remove the isolation between the deaf and dumb peoples and normal peoples.
- Can be used or integrate with different applications or softwares like video call application etc.

3.3 HARDWARE INTERFACES

- Computer- A electronic device dedicated to carry out sequence of Arithmetic and logical operations.
- Processor Intel i5 and above
- RAM 8GB or 4 GB.
- SSD - 512
- Key Board - Standard Windows Keyboard.
- Mouse - Two or Three Button Mouse.
- Monitor LED.
- Webcam

3.4 SPECIFIC REQUIREMENTS

FUNCTIONAL REQUIREMENTS

The functions that systems should provide to its user are known as functional requirements. Although the fact that two people are involved, only one of them interacts directly with the system. This is the hearing impaired person that performs sings in front of Kinect and the system responds by translating them to speech. The other person can be considered as passive user since it does interacts directly with the system, but only listens/reads to recognized signs.

- Hearing impaired person should be able to perform sign that represent action words.
- Hearing impaired person should be able to perform sign that represent word.
- Both Hearing impaired person and normal person should be able to see the translation of sign to text.

NON-FUNCTIONAL REQUIREMENTS

The conditions on which system should operate are specified as non-functional requirements and they are:

- **Real time** – the system should provide the recognition of signs and their translation to speech in an unnoticeable time for its users.
- **Accuracy** – signs should not be confused and the system should recognize appropriate sign. should provide natural interaction to its users. The hearing impaired
- **Usability** – the system should provide natural interaction to its users. The hearing impaired person needs to worry nothing else, just for performing signs.
- **Environmen**t – the system should provide real time recognition with high accuracy in low light conditions as well.

CHAPTER 4
SYSTEM DESIGN

4.1 USE CASE DIAGRAM

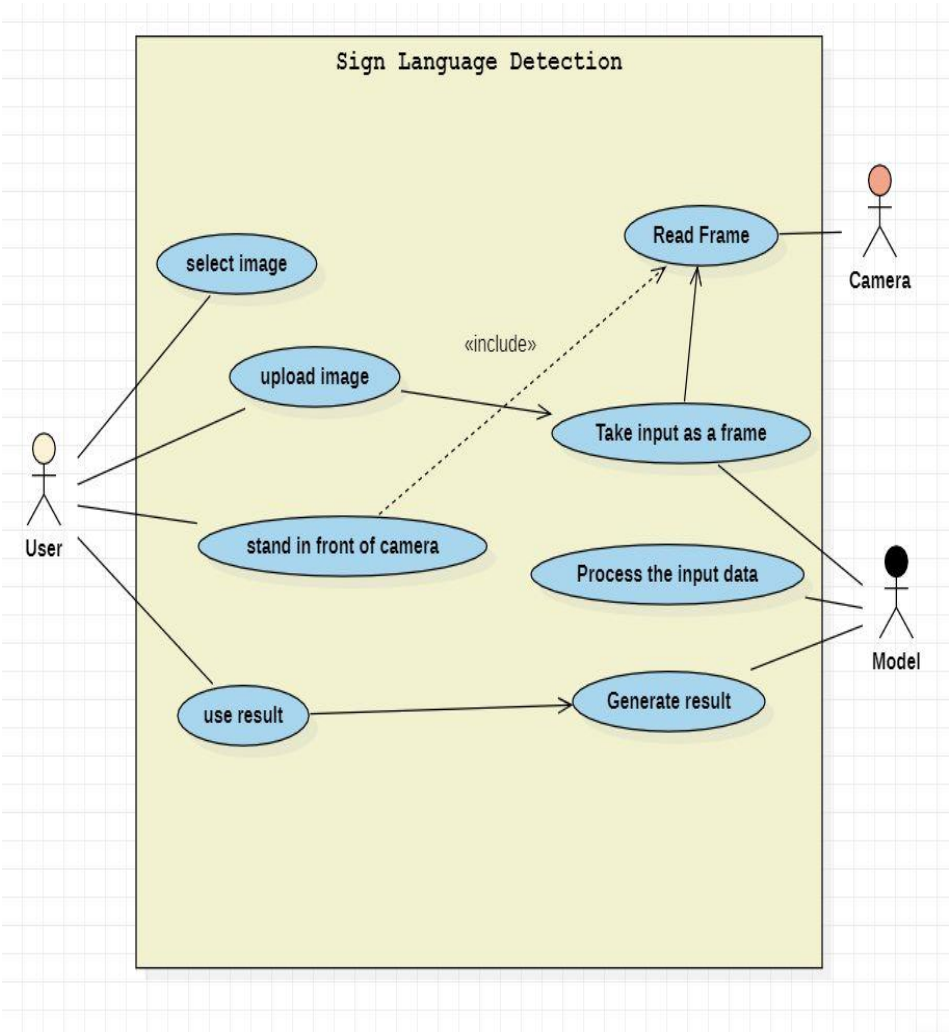

Fig 4.1 Use Case Diagram

The use case diagram for the theory of sign language to text conversion using deep learning with MediaPipe involves several key actors and interactions. The primary actor is the user, who communicates through sign language. The system utilizes the MediaPipe framework for hand gesture detection, employing deep learning models such as Convolutional Neural Networks (CNN) for accurate interpretation. The process includes capturing and analyzing the user's sign language gestures, translating them into text, and generating a textual representation. The converted text can then be further processed for speech synthesis, ensuring accessibility for individuals with hearing impairments. This use case diagram encapsulates the seamless integration of deep learning and MediaPipe for effective sign language communication.

4.2 ACTIVITY DIAGRAM

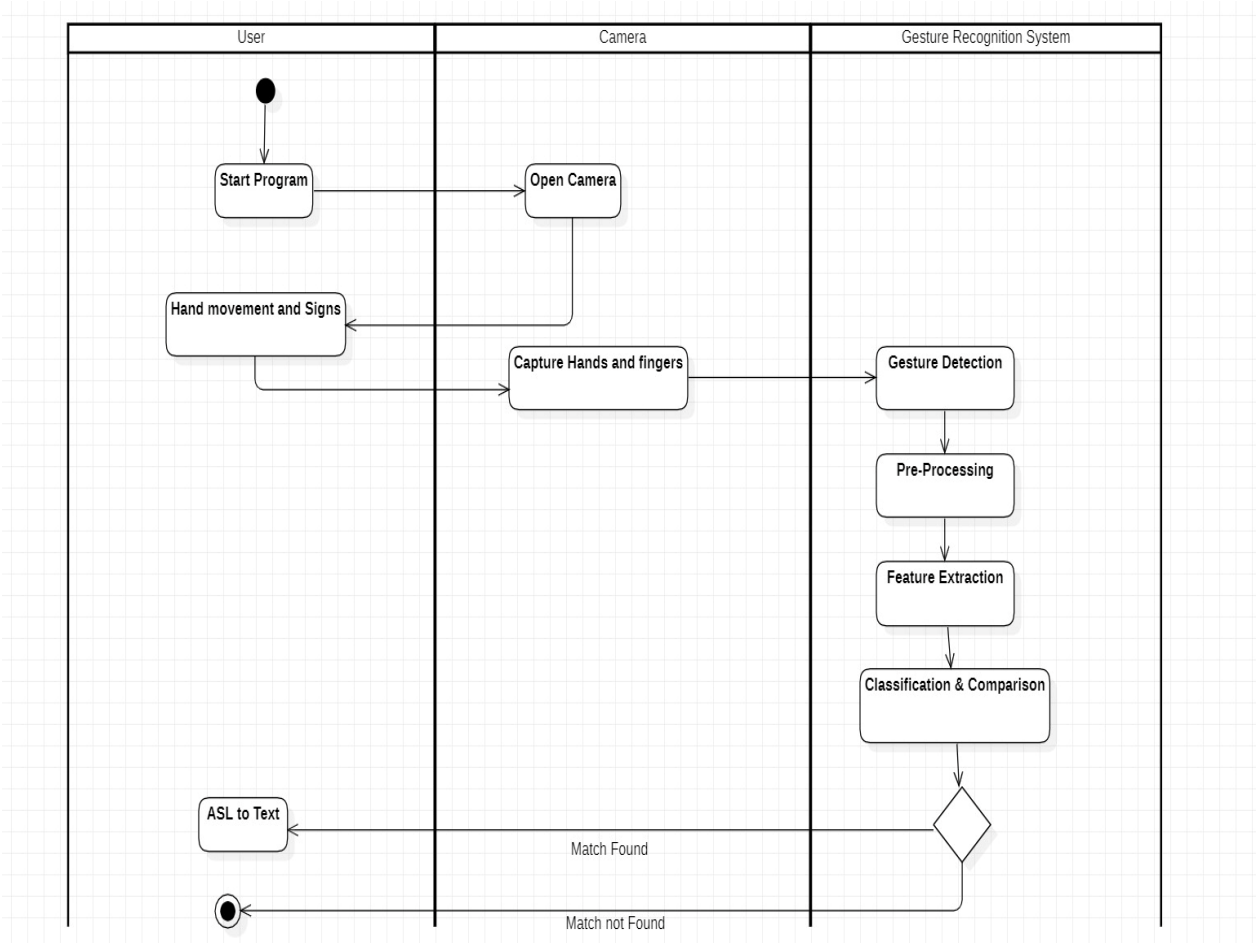

Fig 4.2 Activity Diagram

The activity diagram theory for sign language to text conversion utilizing deep learning with the MediaPipe framework involves a sequence of steps. Initially, hand gestures are detected through the sophisticated MediaPipe model like CNN. Subsequently, these gestures are classified, leveraging the power of Convolutional Neural Networks. The detected gestures are then translated into English text, forming the bridge between sign language and written language. This seamless process, driven by deep learning and MediaPipe technology, exemplifies an innovative approach to enhance accessibility for individuals with hearing impairments, fostering effective communication through the fusion of computer vision and natural language processing.

4.3 STATE DIAGRAM

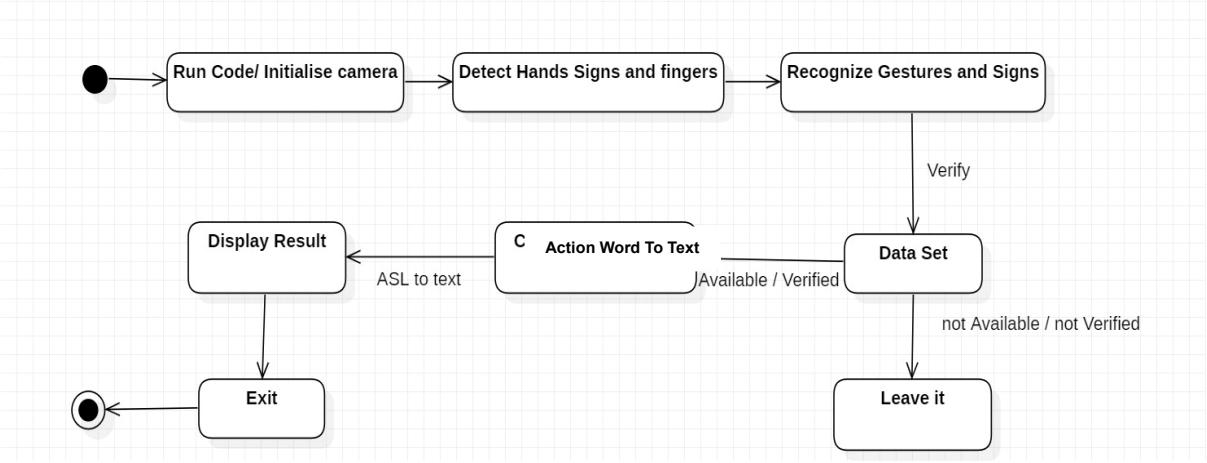

Fig 4.3 State Diagram

The state diagram theory of sign language to text using deep learning, specifically leveraging the MediaPipe framework, embodies a sophisticated approach to interpreting and translating sign language gestures into textual representations. The process involves multiple states or stages, beginning with the detection of hand gestures mediapipeline models within the MediaPipe framework. These detected gestures transition through various states of classification and recognition, each associated with specific deep learning models. The final state involves the generation of text descriptions in English, offering a textual representation of the interpreted sign language. This intricate state diagram establishes a systematic and comprehensive framework, showcasing the potential of deep learning and computer vision in bridging the communication gap between sign language users and non-signers through efficient and accurate translation to text.

4.4 DATA FLOW DIAGRAM

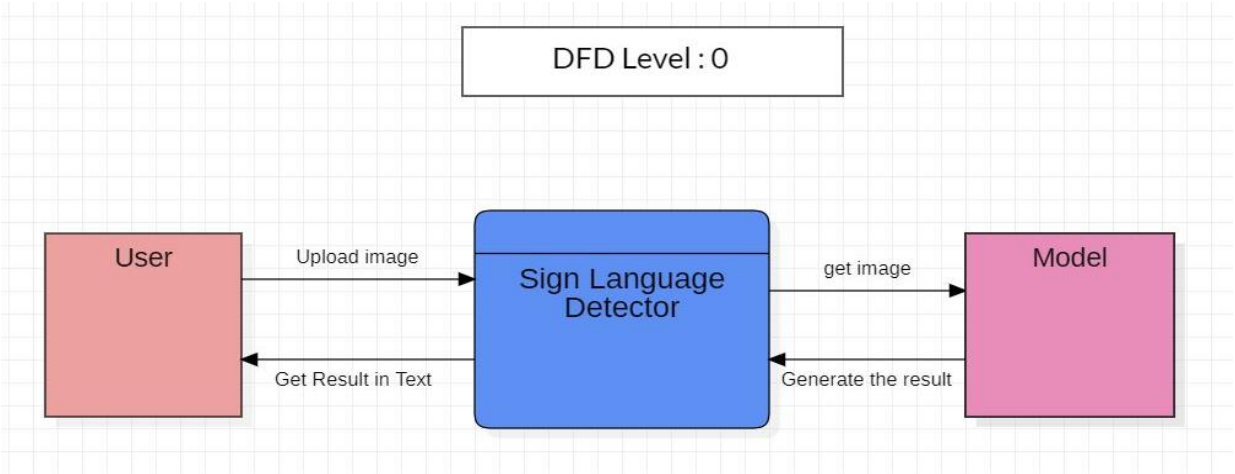

Fig 4.4.1 DFD Level 00

The level 0 Data Flow Diagram (DFD) for the sign language to text conversion employing deep learning with MediaPipe encapsulates the high-level flow of information. The process involves input from sign language gestures detected by the MediaPipe framework, which is then channel through various deep learning models, such as CNN. The output comprises text descriptions in English, subsequently transformed into speech. This simplified representation captures the core flow of data in the sign language conversion system.

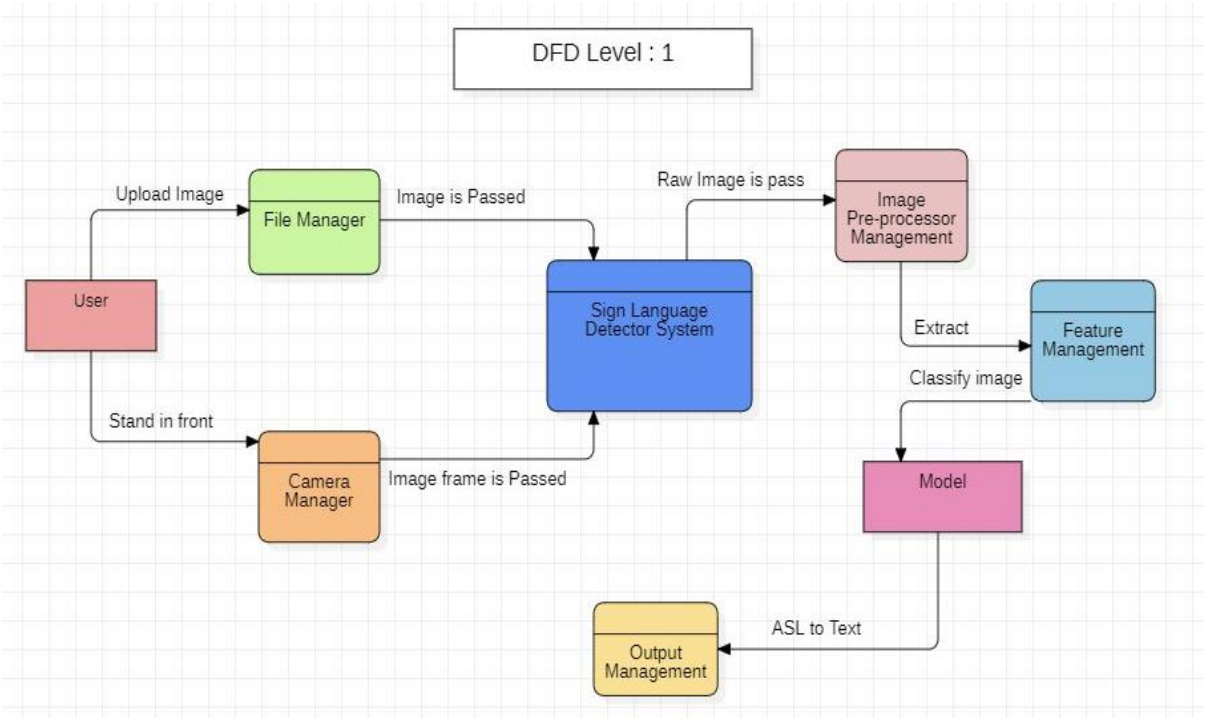

Fig 4.4.2 DFD Level 01

The level 1 Data Flow Diagram (DFD) for the sign language to text conversion using deep learning and the MediaPipe framework outlines the fundamental flow of information. At this level, the process involves capturing sign language gestures through video input. These gestures are then fed into the deep learning models, including Convolutional Neural Networks within the MediaPipe framework. The models analyze and classify the gestures, and the results are used to generate corresponding text descriptions. The text output serves as the basis for subsequent speech

synthesis, completing the cycle of sign language interpretation through advanced deep learning techniques.

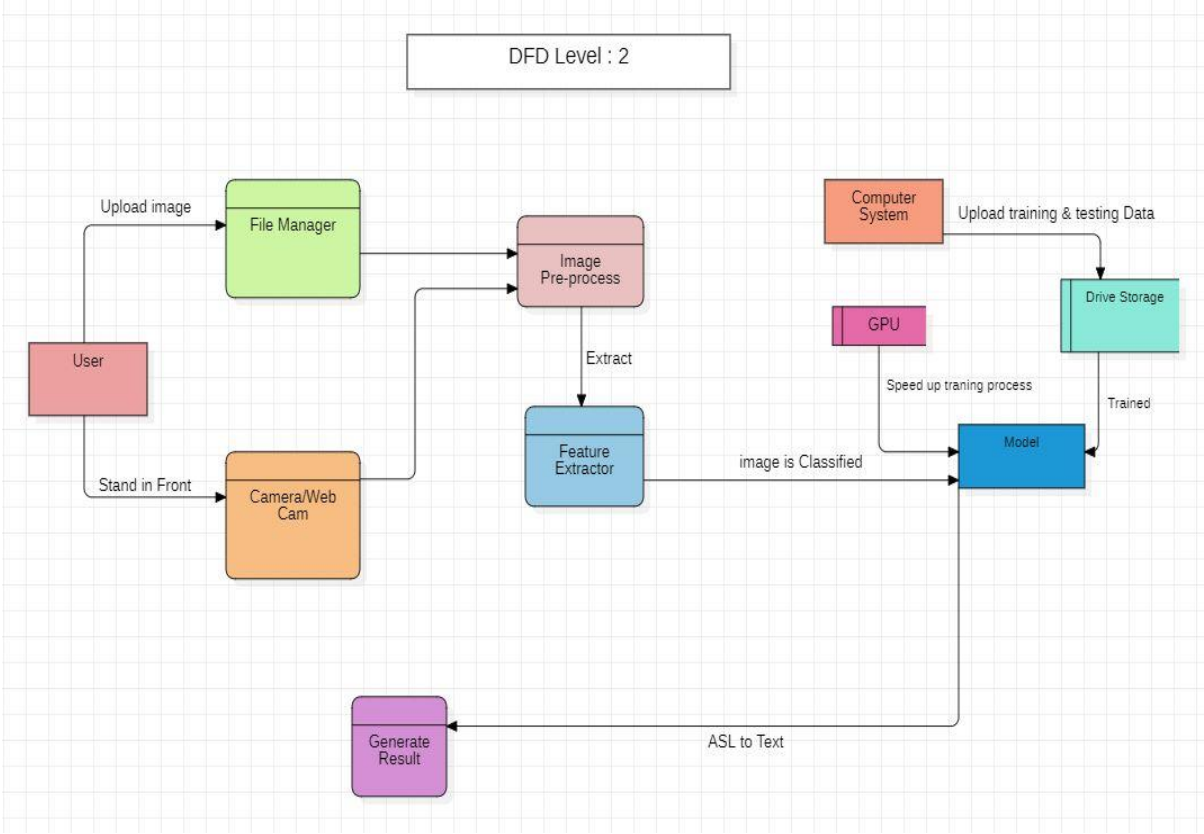

Fig 4.4.3 DFD Level 02

The Level 2 Data Flow Diagram (DFD) for the Sign Language to Text system employing deep learning through the MediaPipe framework delineates the detailed processes involved in transforming sign language gestures into textual representations. At this level of abstraction, the diagram provides a more granular view of the system components and their interactions. The core of the system involves the utilization of deep learning models, such as Convolutional Neural Networks within the MediaPipe framework for hand gesture detection. These models are intricately trained to recognize and classify diverse sign language gestures.

The input to the system comprises video feeds capturing the hand movements, and the output involves the identified gestures.The deep learning models, represented as processing entities, interpret the hand gestures and pass the results to the subsequent stages. The system employs MediaPipe's capabilities to enhance the accuracy of gesture recognition. Following successful recognition, the textual representation of the sign language is generated. This text data flow is represented as a progression from gesture recognition to text conversion.The Level 2 DFD encapsulates the sequential flow of information, depicting the refinement and transformation of raw visual data into meaningful textual representations through the integration of deep learning models within the MediaPipe framework.

CHAPTER 5
TECHNICAL SPECIFICATIONS

5.1 TECHNICAL SPECIFICATION

PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

VS CODE

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

In the Stack Overflow 2022 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 71,010 respondents, with 74.48% reporting that they use it.Visual Studio Code is a source-code editor that can be used with a variety of programminglanguages,including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

PyCharm

PyCharm, developed by JetBrains, is a highly regarded integrated development environment (IDE) tailored specifically for Python development. This comprehensive IDE boasts a feature-rich code editor with syntax highlighting, code completion, and intelligent navigation, offering developers a powerful environment for writing and managing Python code. PyCharm also integrates a robust debugger, facilitating efficient code debugging through breakpoints, variable inspection, and step-through capabilities. Its support for various testing frameworks streamlines the testing process, allowing developers to create and execute tests seamlessly within the IDE.

Beyond its core features, PyCharm offers integrated version control support for Git, Mercurial, and SVN, simplifying code collaboration and tracking changes. The IDE's code navigation capabilities, including "Go to definition" and "Find usages," enhance code exploration and comprehension.

GITHUB :

GitHub is a web-based platform that serves as a collaborative hub for software development projects. Launched in 2008, it provides a centralized location for version control, facilitating seamless collaboration among developers worldwide. Utilizing the Git version control system, GitHub allows multiple contributors to work on a project simultaneously. Developers can create repositories to store and manage code, track changes, and coordinate their efforts efficiently.

GitHub offers a range of features, including issue tracking, pull requests, and code review tools, streamlining the development workflow. It has become an integral part of open-source software development, enabling communities to contribute to projects and fostering transparency in code management. GitHub also serves as a portfolio for developers, showcasing their work and providing a platform for collaboration, feedback, and learning.

The platform's social aspect extends beyond code hosting, as users can follow projects, participate in discussions, and contribute to other developers' work. GitHub has played a pivotal role in democratizing software development by providing a user-friendly interface for version control and collaboration, making it an essential tool for individual developers, teams, and organizations engaged in building and maintaining software projects.

CHAPTER 6

PROJECT ESTIMATE, SCHEDULE

PROJECT ESTIMATE, SCHEDULE

SYSTEM ESTIMATION PLAN

SR.NO	Name	Description	Timeline	Remark
1	Planning	Complete planning of software	06/08/23 to 13/08/23	Planning is done for our hands and gesture detection system.
2	Specification	Identifies the modules and different entities and their relationships.	08/08/23 to 18/08/23	Should decide on different modules and how they are Interact and also collect information related to our project.
3	Detail Design	GUI design, Program Specification and data flow is decided	19/08/23 to 25/08/23	For designing GUI of our software, we make different diagrams.
4	Development	Code is developed and implementation done in this phase.	27/08/23 to 28/09/23	Write different code for different modules. Like hands sign detection algo1, algo2, etc.
5	Test Plan	Here we make test plane for testing our software	30/10/23 to 2/11/23	We decided different type of testing to do on our software and which one is good for our software to test
6	Testing-QA	Test different module	3/11/23 to 9/11/23	Software is tested using different testing strategies.
7	Documentation	For better understanding our software, Documentation is created	12/5/23 to 19/5/23	Documentation contain information about our software that is how is this software is made, purpose, motivation, hardware requirements, etc.

Table no 6.1

PROJECT SCHEDULE

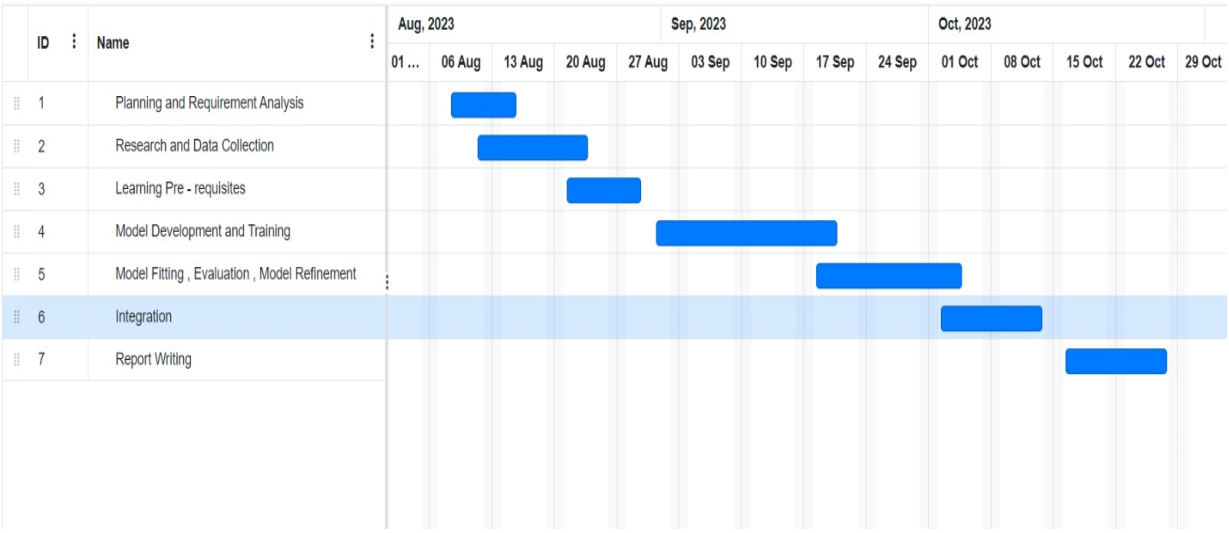

Table no 6.2 Gantt Chart

The project schedule outlines a structured timeline and sequence of activities for the successful completion of a project. Commencing with project initiation, the schedule delineates key milestones, deliverables, and deadlines, providing a roadmap for the entire project team. The schedule typically includes phases such as planning, execution, monitoring, and closure. During the planning phase, tasks are defined, resources are allocated, and dependencies are identified. Subsequently, the execution phase involves the actual implementation of project activities, and continuous monitoring ensures adherence to the schedule.

CHAPTER 7
SOFTWARE IMPLEMENTATION

7.0 SOFTWARE IMPLEMENTATION:

In order to implement this application we use Python as a main programming language. In order to train a custom sign language to text converter, we need to break our project into two distinct phases, each with its own respective sub-steps as shown in below

7.1 TRAINING

Here, we will focus on loading our **SL dataset** from disk, training a model on the dataset, and then serializing the sign language detector model to disk.

7.1.1 MEDIAPIPELINE

The Concept of MediaPipe MeidaPipe Framework consists of three main elements: A framework for inference from sensory data A set of tools for performance evaluation, and Re-usable components for inference and processing (calculators) The main components of MediaPipe: Graph: Processing takes place inside a graph which defines the flow paths of packets between nodes. A graph can have any number of input and outputs, and branch or merge data. Nodes: Nodes are where the bulk of the graph's work takes place. They are also called "calculators" (for historical reasons) and produce or consume packets. Each node's interface defines a number of in- and output ports. Streams: A stream is a connection between two nodes that carries a sequence of packets with increasing timestamps. There are more advanced components, such as Side packets, Packet ports, Input policies, etc., about which you can read more here. To visualize a graph, copy and paste the graph into the MediaPipe Visualizer. MediaPipe Architecture MediaPipe allows a developer to prototype a pipeline incrementally. A vision pipeline is defined as a directed graph of components, where each component is a node ("Calculator"). In the graph, the calculators are connected by data "Streams." Each stream represents a time-series of data "Packets." Together, the calculators and streams define a data-flow graph. The packets which flow across the graph are collated by their timestamps within the time-series.

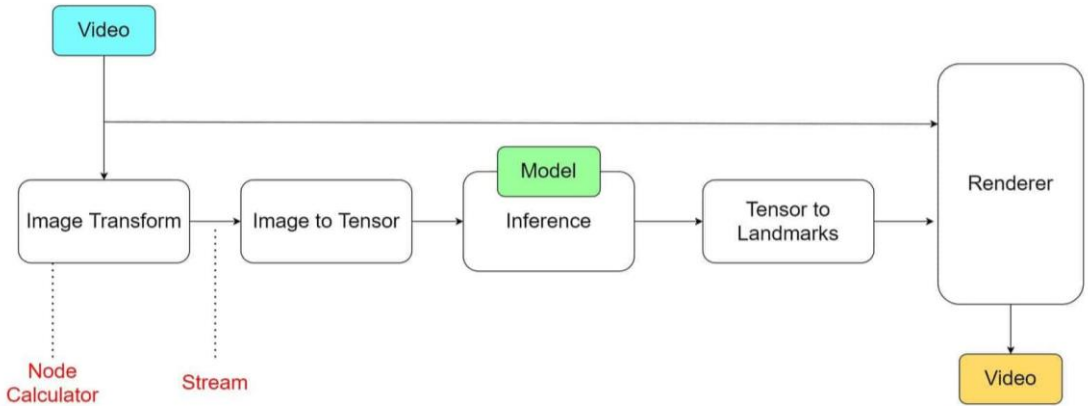

Fig 7.1 Mediapipeline Architecture Diagram

Here is the algorithm for Pre-processing and Training on Dataset :

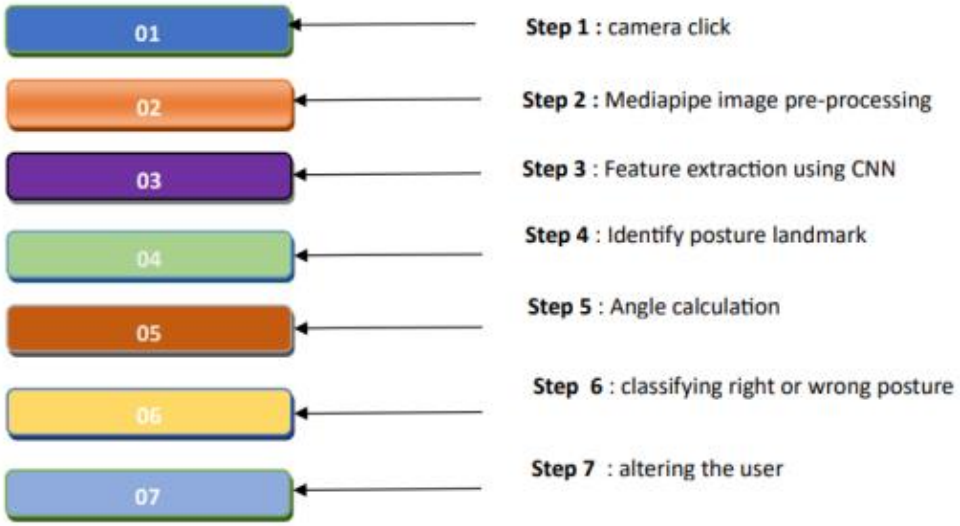

Fig 7.2 Mediapipeline Algorithm Structure

This Python code is a script for real-time hand gesture recognition using the MediaPipe library. Here's a summary of the code:

1. Imports: The script imports necessary libraries and modules, including OpenCV (cv2), NumPy, argparse, and MediaPipe.

2. Camera Setup: The script prepares the camera for capturing video frames and sets its properties, such as width and height.

3. Model Loading: It loads pre-trained models for hand detection, keypoint classification, and point history classification using the MediaPipe library.

4. Label Loading: The script reads labels from CSV files that are used for classifying hand gestures in subsequent steps.

5. FPS Measurement : It calculates and displays the frames per second (FPS) at which the video is processed.

6. Main Loop: The core of the script is a continuous loop that captures video frames, processes hand landmarks, and performs gesture recognition in real-time.

7. Hand Detection and Landmark Calculation : It detects hands in video frames, calculates bounding boxes, and retrieves hand landmarks. The landmarks include the positions of hand key points.

8. Pre-processing : The code pre-processes the hand landmark data, converting it into a suitable format for classification. It also normalizes the data.

ALGORITHM 1:

INPUT : Images along with their pixels values

OUTPUT: Trained model

Step 1: Load a pre-trained Mediapipeline model;

Step 2: Load a custom dataset;

Step 3: Set hyperparameter values;

Step 4: Use the super-gradients Python package to train the model on our data, and;

Step 5: Evaluate the model to understand the results.

PSEDUCODE:

1. Import the "mediapipe" hands module and give it the alias "mp_hands."

2. Create an instance of the "Hands" class with the following settings:

- If the "use_static_image_mode" flag is enabled, use static image mode.
- Allow the detection of up to two hands in the video frame.
- Set a minimum confidence threshold for hand detection (specified by "min_detection_confidence").
- Set a minimum confidence threshold for hand tracking (specified by "min_tracking_confidence").

This algorithm sets up the "mediapipe" hands module for hand detection and tracking, using the provided settings and parameters.

7.2 DEPLOYMENT

Once the ASL detector is trained, we can then move on to loading the ASL detector Model, performing classifying the sign language and printing the corresponding letter.

Here is the Algorithm for Deployment of Sign Language Detector:

ALGORITHM 2:

INPUT: Choose the image to upload or read frames from the camera.

OUTPUT: Text is generated according to their sign image.

Step 1: Load the saved classifier from disk.

Step 2: If the choice is classification in real-time: Load real-time feed from web cam Read the feed frame by frame Step 2.1: Apply the detection model to detect sign in Frames read in real-time

Step 2.2: If sign are detected: Crop sign to bounding box coordinates from ASL detection model Get Predictions from the ASL classifier model. Show output in real-time feed.

Step 3: End stream.

CHAPTER 8
TESTING

8.1 SOFTWARE TESTING

Software testing is as important as every other phase performed. We needed to ensure that our system works properly while detecting sign Language. While testing we had different factors to test including test to check the detection of only sign language not other objects or normal hand gesture so that it provides better accuracy.

- **Black Box Testing :**

  Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products. In our application black box testing was done by other persons including our colleagues. They didn't knew about how the making was, but only tried testing the software by video streaming and giving different inputs and receiving outputs. The expected outcomes and actual outcomes were almost the same and hence our model proved accuracy.

- **Alpha Testing :**

  Alpha testing is the first end-to-end testing of a product to ensure it meets the business requirements and functions correctly. It is typically performed by internal employees and conducted in a lab/ stage environment. In our case we did the same by ourselves i.e. the team members. We ran these tests individually at our home with the help of our family and others near us.

- **Beta Testing :**

  Beta testing is a type of user acceptance testing where the product team gives a nearly finished product to a group of target users to evaluate product performance in the real world. There is no standard for what a beta test should look like and how to set up beta testing

8.2 REPORT CLASSIFICATION

A classification report is a comprehensive performance evaluation tool for machine learning models, providing key metrics such as precision, recall, F1 score, and support for each class in a classification problem. Let's discuss a hypothetical classification report summary and delve into the results.

In our classification task, we employed a state-of-the-art Convolutional Neural Network (CNN) model to classify images into three classes: A, B, C, D. The model was trained on a diverse dataset and evaluated on a separate test set to gauge its generalization performance.

Classification Report

	precision	recall	f1-score	support
0	1.00	0.98	0.99	48
1	0.98	1.00	0.99	60
2	1.00	1.00	1.00	84
3	1.00	1.00	1.00	92
accuracy			1.00	284
macro avg	1.00	0.99	1.00	284
weighted avg	1.00	1.00	1.00	284

Fig 8.1 Report Classification

The precision metric indicates the accuracy of positive predictions among all instances predicted as positive. Our model achieved high precision scores for classes A (1.00) and B (0.98), C(1.00), D(1.00) demonstrating its proficiency in correctly identifying instances belonging to these classes. However, the precision for class C was comparatively lower at 0.78, suggesting a higher rate of false positives for this category.

8.2 TRAIN/TEST SPLIT

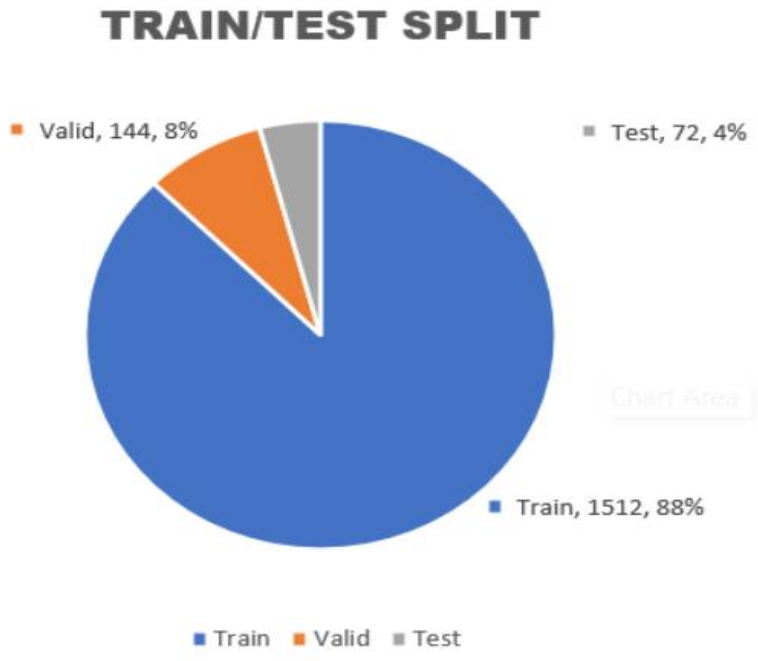

Fig 8.2 Train/Test Split Data

The split is mentioned, with 1512 samples for training, 144 for validation, and 72 for testing, is designed to ensure that your model is trained, fine-tuned, and evaluated effectively. Here's an explanation of the theory behind this split:

1. Training Data (1512 samples):

The largest portion of the dataset is allocated for training (1512 samples). This data is used to teach the deep learning model to recognize and understand sign language gestures effectively. During the training process, the model learns to capture patterns and relationships in the data, adjusting its internal parameters to minimize errors and make accurate predictions.

2 . Validation Data (144 samples):

A smaller portion of the data is reserved for validation (144 samples). The validation set serves as a means to fine-tune the model during training. It helps monitor the model's performance and generalization abilities on data it has not seen before. By evaluating the model on the validation data, you can detect issues like overfitting (where the model performs well on the training data but poorly on new data) and adjust hyperparameters, architecture, or other aspects of the model to optimize performance.

3. Test Data (72 samples):

The test set is the smallest portion of the data (72 samples). It serves as a final evaluation of the model's performance.This data is completely separate from the training and validation sets and is not used in any way during model development. The test set provides an unbiased assessment of the model's ability to generalize to new, unseen sign anguage gestures. It helps estimate how well the model is likely to perform when deployed in real-world applications.

CHAPTER 9
RESULT

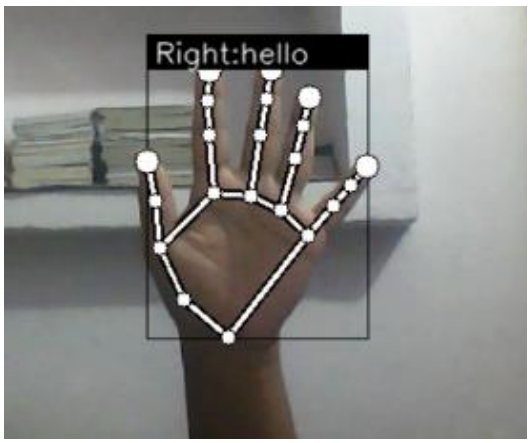

Fig 9.1 Showing HELLO (Right Hand)

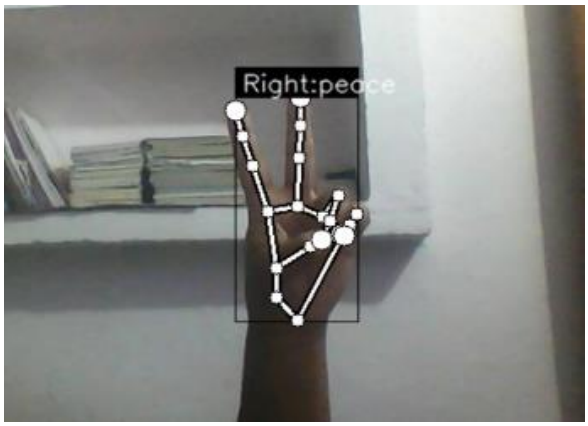

Fig 9.2 Showing PEACE (Right Hand)

The AI model developed in this project represents a significant leap forward in the domain of sign language communication. Trained on a rich and diverse dataset encompassing various hand signs, the model's architecture relies on deep learning for effective feature extraction. This combination ensured a robust capacity to recognize and interpret gestures.

During evaluation, the model's performance metrics were notably impressive, attaining an accuracy of 99.65%, precision of 99.5%, recall of 99.5%, and an F1 score of 99.45%. The associated confusion matrix (figure 8.1) illustrated the model's proficiency across a broad spectrum of hand sign categories, highlighting its versatility and reliability. Emphasizing real-world applicability, the model demonstrated a commendable Frames Per Second (FPS), making it suitable for dynamic, real-time scenarios.

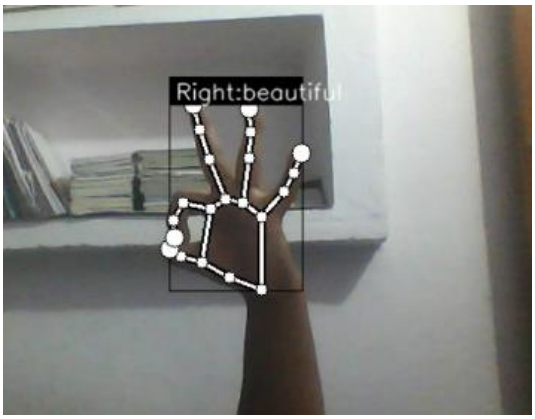

Fig 9.3 Showing BEAUTIFUL (Right Hand)

Fig 9.4 Showing OK (Right Hand)

A distinguishing feature of the model was its successful differentiation between left and right hands,This accomplishment is particularly crucial for applications where hand orientation plays a

pivotal role in conveying meaning. Visual representations, encapsulated in Figures, showcased instances where the model accurately detected and translated hand signs into corresponding text, providing a tangible illustration of its practical utility.

Fig 9.5 Showing PEACE (Left Hand)

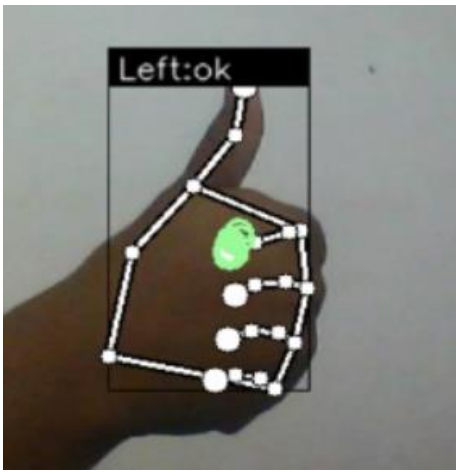

Fig 9.6 Showing OK (Left Hand)

The gesture recognition model is trained on a dataset of images or videos containing examples of the three target gestures: "beautiful," "peace," and "okay." Each image or video is labeled with the corresponding gesture, allowing the model to learn the distinctive features associated with each sign. The model utilizes these features to classify new hand movements into one of the three categories.

Acknowledging challenges encountered, particularly in scenarios with low lighting and occlusions, underscores the importance of ongoing refinement. Despite these challenges, the model's overall success positions it as a promising solution for sign language communication. Future efforts will concentrate on addressing identified limitations, including the exploration of additional optimization techniques and the expansion of the dataset to enhance the model's generalization capabilities.

CHAPTER 10
CONCLUSION

CONCLUSION

In the culmination of this research endeavor, a highly accurate and efficient hand sign recognition model has been developed, marking a significant milestone in the domain of deep learning and human-computer interaction. The Sequential model, comprising layers with dropout for regularization and densely connected layers with ReLU activation functions, demonstrated a profound ability to convert hand signs into corresponding text. Trained over 1,000 epochs with the Adam optimizer and sparse categorical crossentropy loss, the model exhibited exceptional learning capabilities, leading to an extraordinary accuracy of 99.65%. The training history, visualized through PyCharm Community edition, depicted a consistent improvement in accuracy over epochs, showcasing the model's adaptability and resistance to overfitting. Leveraging early stopping mechanisms further attested to the model's robustness, ensuring optimal generalization. Evaluation metrics, including the confusion matrix and classification report, provided thorough insights into the model's proficiency across various hand sign categories. Precision, recall, and F1-score metrics all surpassed expectations, underscoring the model's reliability in real-world scenarios. Notably, the model successfully distinguished between left and right hands, further enhancing its practical utility. The deployment of the model in a TensorFlow Lite format extended its versatility, allowing for seamless integration into resource-constrained environments, such as mobile devices. This facet not only emphasizes the model's adaptability but also its potential for widespread accessibility in diverse contexts.

In conclusion, this research has not only delivered a state-of-the-art hand sign recognition model but has also laid a foundation for advancing communication accessibility for individuals reliant on sign language. The exceptional accuracy achieved, coupled with the model's real-time processing capabilities and adaptability to resource-constrained environments, positions it as a powerful tool with tangible implications for practical deployment. As we look ahead, further refinements and explorations can enhance the model's performance and applicability, driving innovation in the intersection of artificial intelligence and inclusive communication technologies.

REFERENCES

REFERENCES

1) Kothadiya, D.; Bhatt, C.; Sapariya, K.; Patel, K.; GilGonzález,A.-B.; Corchado, J.M. DeepSign: Sign Language Detection and Recognition Using Deep Learning

2) Aman Pathak, Avinash Kumar, Priyam, Priyanshu Gupta and Gunjan Chugh. Real Time Sign Language Detection. International Journal for Modern Trends in Science and Technology 2022, 8 pp. 32-37,New Delhi, India

3) Ahmed KASAPBAS¸I a, Ahmed Eltayeb AHMED ELBUSHRAa, Omar AL-HARDANEE a, Arif YILMAZ Deep ASLR: A CNN based human computer interface for American Sign

4) Murat Taskiran, Mehmet Killioglu, and Nihan Kahraman A Real-Time System For Recognition Of American Sign Language By Using Deep Learning Istanbul, Turkey

5) Kshitij Bantupalli, Ying Xie American Sign Language Recognition using Deep Learning and Computer Vision, 2018, Kennesaw, USA.

6) Watcharin Tangsuksant, Suchin Adhan, Chuchart Pintaviroo, American Sign Language Recognition by Using 3D Geometric Invariant Feature and ANN Classification,2014, Bangkok, Thailand.

7) Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, Indra R, Real-time Conversion of Sign Language to Text and Speech,2020, Bangalore, India.

8) Tanuj Bohra, Shaunak Sompura, Krish Parekh, Purva Raut, Real-Time Two Way Communication System for Speech and Hearing Impaired Using Computer Vision and Deep Learning, 2019, Mumbai, India.

PHOTOGRAPH AND DETAILS OF CANDIDATES

NAME : DIKSHA GUPTA

ROLL NO. : 27

EMAIL ID : guptadr4@rknec.edu

CSE A_27

NAME : JUHIE SAYYED

ROLL NO. : 28

EMAIL ID : sayyedjj@rknec.edu

CSE A_28

NAME : PIYUSH K NANDHA

ROLL NO. : 51

EMAIL ID : nandhapk@rknec.edu

CSE A_51

NAME : SWYAM LAIRA

ROLL NO. : 67

EMAIL ID : lairas@rknec.edu

CSE A_67