VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering

Project Report on

SilentCue - Sign Language Recognition for Deaf and Nonverbal

In partial fulfillment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai Academic Year 2024-25

Submitted by

Jiya Gangwani (D17A , Roll no - 17)

Nikhil Dhanwani (D17B , Roll no - 11)

Soham Panjabi (D17B , Roll no - 36)

Chirag Santwani (D17B , Roll no - 47)

Project Mentor

Prof. Ms. Manisha Mathur

(2024-25)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering

Certificate

This is to certify that ***Jiya Gangwani (D17A, 17), Nikhil Dhanwani (D17B, 11), Soham Panjabi (D17B, 36), Chirag Santwani (D17B, 47)*** of Fourth Year Computer Engineering studying under the University of Mumbai have satisfactorily completed the project on "***SilentCue - Sign Language Recognition for Deaf and NonVerbal***" as a part of their coursework of PROJECT-II for Semester-VIII under the guidance of their mentor ***Prof. Ms. Manisha Mathur*** in the year 2024-25 .

This thesis/dissertation/project report entitled ***SilentCue - Hand Gesture Recognition for Deaf and NonVerbal*** by ***Jiya Gangwani (D17A, 17), Nikhil Dhanwani (D17B, 11), Soham Panjabi (D17B, 36), Chirag Santwani (D17B, 47)*** is approved for the degree of **B.E. Computer Engineering.**

Programme Outcomes	Grade
PO1,PO2,PO3,PO4,PO5,PO6,PO7, PO8, PO9, PO10, PO11, PO12 PSO1, PSO2	

Date:
Project Guide: Prof. Ms. Manisha Mathur

Project Report Approval
For
B. E (Computer Engineering)

This thesis/dissertation/project report entitled **SilentCue - Sign Language Recognition for Deaf and NonVerbal** by **Jiya Gangwani (D17A, 17), Nikhil Dhanwani (D17B, 11), Soham Panjabi (D17B, 36), Chirag Santwani (D17B, 47)** is approved for the degree of **B.E. Computer Engineering**.

Internal Examiner

External Examiner

Head of the Department

Principal

Date:
Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

---------- ----------

(Jiya Gangwani, 17) (Nikhil Dhanwani, 11)

---------- ----------

(Soham Panjabi, 36) (Chirag Santwani, 47)

Date:

ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society's Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Prof. Ms. Manisha Mathur** for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

Computer Engineering Department
COURSE OUTCOMES FOR B.E PROJECT

Learners will be to,

Course Outcome	Description of the Course Outcome
CO 1	Able to apply the relevant engineering concepts, knowledge and skills towards the project.
CO2	Able to identify, formulate and interpret the various relevant research papers and to determine the problem.
CO 3	Able to apply the engineering concepts towards designing solutions for the problem.
CO 4	Able to interpret the data and datasets to be utilized.
CO 5	Able to create, select and apply appropriate technologies, techniques, resources and tools for the project.
CO 6	Able to apply ethical, professional policies and principles towards societal, environmental, safety and cultural benefit.
CO 7	Able to function effectively as an individual, and as a member of a team, allocating roles with clear lines of responsibility and accountability.
CO 8	Able to write effective reports, design documents and make effective presentations.
CO 9	Able to apply engineering and management principles to the project as a team member.
CO 10	Able to apply the project domain knowledge to sharpen one's competency.
CO 11	Able to develop professional, presentational, balanced and structured approach towards project development.
CO 12	Able to adopt skills, languages, environment and platforms for creating innovative solutions for the project.

Index

Abstract

SilentCue is an innovative sign language recognition system designed to bridge communication gaps for deaf and non-verbal individuals. The project combines real-time hand gesture recognition with a user-friendly graphical interface, allowing seamless communication in American Sign Language (ASL). By leveraging cutting-edge technologies like OpenCV, MediaPipe, and machine learning, SilentCue interprets hand gestures and translates them into text, facilitating smooth and interactive conversations.

The system is powered by a custom dataset trained specifically for ASL recognition, ensuring high accuracy and adaptability to various gestures. Key features of the project include gesture training and translation, word and sentence formation, interactive controls like case toggling and backspace, and a dynamic word suggestion system powered by NLTK. The real-time video feed and hand landmarks are displayed through an intuitive Tkinter GUI, allowing users to see both the hand gestures and their translated text.

SilentCue also integrates a dual-purpose module for controlling system actions such as mouse movements, scrolling, and volume adjustment through hand gestures. The system uses MediaPipe for accurate hand tracking and pyautogui and pycaw for controlling the system's mouse and volume. This makes the project not only a communication tool but also an assistive technology for everyday tasks, enhancing accessibility for non-verbal users.

Further expanding its functionality, SilentCue allows users to train and add custom gestures, broadening its scope and accommodating personalized sign language needs. The integration of a video call feature via TokBox enables real-time ASL communication, enhancing its utility for social interactions.

Through its innovative use of gesture recognition, machine learning, and interactive design, SilentCue empowers deaf and non-verbal individuals with a powerful tool for communication, making the world more inclusive and accessible.

Chapter 1: Introduction

1.1. Introduction:

SilentCue - Sign Language Recognition for Deaf and Non-Verbal is an innovative project that focuses on bridging the communication gap for individuals who cannot speak. It uses machine learning, computer vision, and real-time hand tracking through MediaPipe to recognize American Sign Language (ASL) gestures and convert them into readable text. The system also includes features like sentence formation, word suggestions, gesture-based system control (like mouse movement and volume adjustment), and custom gesture training, all wrapped in a user-friendly interface built with Tkinter. SilentCue empowers users to communicate more easily, promotes inclusivity, and demonstrates the powerful role technology can play in making the world more accessible.

1.2. Motivation:

Communication barriers faced by deaf and non-verbal individuals are often underestimated in daily life. In environments such as schools, workplaces, hospitals, and public services, the inability to communicate effectively can lead to feelings of isolation and dependency. Although sign language provides a powerful tool for self-expression, not everyone understands it, leading to further gaps. The motivation behind SilentCue is to leverage the advancements in artificial intelligence, computer vision, and machine learning to create a real-time sign language recognition system that does not require expensive equipment or professional interpreters. By building a platform where users can simply use hand gestures captured via a regular webcam and instantly translate them into text, SilentCue aims to empower users with greater independence, dignity, and social inclusion.

1.3. Problem Definition:

Deaf and non-verbal individuals often rely on sign language as their primary mode of communication, but a significant barrier arises when interacting with people who are unfamiliar with sign language. Traditional solutions involve hiring human interpreters, using written communication, or relying on limited sign language translation apps, all of which have their drawbacks in terms of accessibility, privacy, cost, and speed. The main problem is the lack of a widely available, real-time, and user-friendly system that can translate sign language gestures into spoken or written language accurately and efficiently. Furthermore, most current systems either require heavy hardware setups or are restricted to recognizing a very narrow set of gestures, failing to adapt to personal or regional variations in sign language usage. SilentCue addresses this gap by offering a lightweight, efficient, customizable, and real-time solution that can easily adapt to the user's needs.

1.4. Existing Systems:

Existing technologies in the field of sign language recognition show that significant progress has been made, but many limitations still exist. Some systems use smart gloves embedded with sensors to capture hand movements and translate them into text or speech; examples include the SignAloud glove developed at the University of Washington. Other solutions use smartphone applications that recognize static hand gestures through images, like Google's Teachable Machine or limited ASL apps. However, many of these applications only support a predefined set of static signs (primarily alphabets) and do not support fluid sentence formation or dynamic gestures. Additionally, some research projects require the use of depth sensors or infrared cameras, increasing the overall cost and making the system less accessible to the general population. Moreover, many commercial systems lack features like user-driven gesture training, real-time system control (e.g., using gestures for mouse or volume), or integration with communication platforms like video calls.

1.5. Lacuna of the Existing System:

While the existing systems have made valuable contributions, they have significant shortcomings. Most notably, the dependency on specialized hardware such as gloves, Leap Motion sensors, or high-end cameras restricts accessibility and affordability. Many solutions are focused only on recognizing single gestures and fail to provide continuous sentence-building or conversational-level communication. Moreover, user customization — the ability to train new gestures according to individual preference — is often absent, limiting flexibility. Gesture-based control for everyday tasks like controlling the mouse, scrolling web pages, or adjusting volume, which could significantly boost user independence, is largely missing from traditional systems. In addition, the lack of real-time video communication integration in most systems prevents users from using gestures effectively during live interactions.

1.6. Relevance of the Project:

SilentCue is highly relevant to today's world, where the emphasis on inclusivity, accessibility, and assistive technology is stronger than ever. As communication is a basic human right, providing tools that enable free, independent communication for deaf and non-verbal individuals is crucial. By requiring only a basic webcam and offering a rich set of features — including custom gesture training, continuous sentence formation, gesture-based system control, and video call integration — SilentCue becomes a powerful, accessible, and cost-effective solution for real-world use. Furthermore, the project's use of open technologies like OpenCV, MediaPipe, Tkinter, and lightweight machine learning models ensures that it can be deployed on a wide variety of systems without heavy computational requirements. As society moves towards more inclusive environments, SilentCue stands as an example of how AI and computer vision can be harnessed for meaningful, impactful solutions that genuinely improve quality of life.

Chapter 2: Literature Survey

A. Overview of literature survey:

The literature survey for the project "SilentCue - Sign Language Recognition for Deaf and Non-Verbal" examines advancements in hand gesture recognition systems, focusing on machine learning, deep learning, and multimodal sensor integration. Research has demonstrated the effectiveness of Convolutional Neural Networks (CNNs) for static gesture recognition and Long Short-Term Memory (LSTM) networks for dynamic gestures. Real-time performance is crucial, and solutions like MediaPipe combined with deep learning have been shown to reduce latency and enhance accuracy, though environmental factors still pose challenges. Additionally, multimodal systems incorporating computer vision and other sensors, such as wearables, have been explored to improve recognition accuracy under varying conditions. Despite significant progress, challenges remain in achieving robust, scalable, and real-time systems that can handle the complexities of sign language translation in diverse real-world scenarios.

2.1. Research Papers :

1. Real-Time Hand Gesture Recognition for Improved Communication

Abstract:

This paper focuses on real-time hand gesture recognition systems using Convolutional Neural Networks (CNN). The system processes webcam footage to identify sign language gestures, enabling communication between hearing-impaired individuals and others. The study introduces a robust approach using a deep learning model trained on a large dataset of gestures.

Inference:

The study highlights the efficacy of CNNs in recognizing hand gestures in real-time, which is essential for applications like sign language recognition. This research emphasizes real-time performance, accuracy, and low-latency, which are crucial for interactive applications. However, limitations on gesture types and the need for controlled environments were noted.

2. Machine Learning-Based Gesture Recognition for Communication

Abstract:

The paper proposes a deep learning-based gesture recognition model to bridge the communication gap between deaf and non-verbal individuals and people unfamiliar with sign language. The approach focuses on utilizing machine learning algorithms to interpret various sign gestures, offering translation in text or speech.

Inference:

This paper offers valuable insights into using machine learning models for gesture recognition, improving the communication flow. The deep learning model shows strong results in recognizing dynamic gestures, suggesting that machine learning can provide a scalable solution for real-time sign language translation. The system's adaptability could be an advantage in diverse environments.

3. Real-Time Sign Language Translation Systems: A Review Study

Abstract:

This review article analyzes the evolution of real-time sign language translation systems. It compares several approaches and technologies, including vision-based recognition and wearable devices, assessing their effectiveness and limitations. It focuses on systems that provide real-time translation from sign language to text or speech.

Inference:

The review suggests that while significant progress has been made in real-time sign language translation, challenges remain in accuracy, scalability, and real-time processing. The paper encourages the integration of multimodal systems, such as combining computer vision with wearables, to improve the system’s robustness and flexibility.

4. A Survey of Advancements in Real-Time Sign Language Translators

Abstract:

This survey paper explores recent advancements in real-time sign language translation systems, particularly focusing on incorporating Internet of Things (IoT) technologies. The study reviews several approaches and discusses challenges related to accuracy, language-specific signs, and interactive user experiences.

Inference:

The integration of IoT is a notable trend, enhancing accessibility and interactivity. While current systems are limited to simple gesture recognition, the inclusion of IoT could lead to innovative applications, such as integrating sign language with smart home technologies or wearables. The study emphasizes the need for real-time performance and the adaptation to diverse sign languages.

5. Real-Time Sign Language Recognition Using Deep Learning

Abstract:

The paper proposes the use of Long Short-Term Memory (LSTM) networks for real-time sign language recognition. It focuses on continuous gesture recognition, allowing for dynamic communication, and integrates temporal data processing to handle the sequential nature of sign language.

Inference:

LSTM networks are particularly suited for recognizing sign language as they can capture the temporal aspect of gestures. This study highlights the importance of incorporating time-sequence data for real-world sign language recognition. It also points out the potential for improving real-time recognition.

6. Sign Language Recognition Based on Computer Vision

Abstract:

This paper provides a comprehensive review of various hand gesture techniques for sign language recognition using computer vision. It delves into methods of hand segmentation, feature extraction, and hand tracking, evaluating their applications in real-time systems and their accuracy under different environmental conditions.

Inference:

The paper suggests that computer vision methods, including hand segmentation and keypoint detection, are effective but still have limitations under challenging conditions (e.g., varying light, occlusion). This indicates that integrating deep learning models with traditional computer vision techniques could improve accuracy in dynamic environments.

7. Sign Language Recognition with Multimodal Sensors and Deep Learning

Abstract:

This paper explores the fusion of multimodal sensor data (e.g., 2-axis bending sensors and computer vision) for sign language recognition. By combining these data streams, the system achieves higher recognition accuracy even under occlusion or noisy conditions, which is common in real-world scenarios.

Inference:

The study confirms that multimodal systems can significantly improve sign language recognition accuracy. While traditional computer vision methods may fail in real-time applications with occlusion, using additional sensors helps mitigate these issues. This approach also shows the importance of using diverse input data for more robust systems.

8. Real-Time American Sign Language Interpretation Using Deep Learning

Abstract:

This research introduces a real-time American Sign Language (ASL) interpretation system that uses deep learning for gesture recognition and keypoint tracking for improved accuracy in hand position detection.

Inference:

The use of keypoint tracking with deep learning offers significant improvements in accuracy and efficiency, making the system suitable for real-time applications. The paper suggests that while real-time systems are a challenge, advancements in neural networks are making such applications increasingly feasible.

9. Mediapipe and CNNs for Real-Time ASL Gesture Recognition

Abstract:

This study proposes combining MediaPipe, a powerful framework for real-time hand tracking, with Convolutional Neural Networks (CNNs) for ASL gesture recognition. It demonstrates how this hybrid approach can improve gesture accuracy and processing speed, offering a low-latency solution .

Inference:

The combination of MediaPipe and CNNs offers a powerful framework for real-time gesture recognition. MediaPipe is highly effective for hand tracking, and CNNs can handle complex pattern recognition. This approach could lead to efficient and low-latency systems suitable for interactive environments. The paper also emphasizes the scalability of the system for different sign languages.

10. Image-Based Indian Sign Language Recognition: A Practical Review Using Deep Neural Networks

Abstract:

This paper focuses on using deep neural networks (DNNs) for real-time Indian Sign Language (ISL) recognition. It provides a comprehensive review of the challenges and techniques for accurately recognizing ISL gestures through images, considering the complexity and variation of signs in different regional contexts.

Inference:

The paper shows that deep neural networks are effective for recognizing complex sign languages, including ISL. The key takeaway is the need for region-specific models that can accommodate local variations in signs. The use of image-based systems with DNNs proves to be a practical and scalable solution for recognizing a broader set of gestures and enabling communication.

Chapter 3 : Requirement gathering for the proposed system

3.1 Introduction to Requirement gathering

Requirement gathering is a crucial phase in the System Development Life Cycle (SDLC) that lays the foundation for the successful development of a project. It involves systematically identifying, analyzing, documenting, and validating the needs and expectations of the users and stakeholders. For "SilentCue - Sign Language Recognition for Deaf and Non-Verbal," requirement gathering ensures that the system addresses the core needs of users such as deaf, mute, and non-verbal individuals by providing an accessible way to communicate through hand gestures. It also identifies technical necessities like real-time hand tracking, gesture classification, custom gesture training, and GUI development. Thorough requirement gathering ensures that SilentCue remains focused, user-friendly, and scalable to meet future needs, like support for multiple languages or expanded gesture sets.

3.2 Functional Requirements

Functional requirements specify what the system should do and how it should behave in response to user interactions. For SilentCue, the key functional requirements are:

- **Hand Gesture Recognition**
    - Detect and classify static and dynamic hand gestures in real-time using MediaPipe and OpenCV.
- **Translation to Text**
    - Recognized gestures must be translated into corresponding text outputs on the screen.
- **Custom Gesture Training**
    - Allow users to create and train their own gestures, associating them with specific words or phrases.
- **Word and Sentence Formation**
    - Enable users to form complete sentences using recognized gestures, with options for adding space, backspace, clear, and save functionality.
- **Cursor, Scroll, and Volume Control**
    - Allow users to control the mouse, scroll pages, and adjust system volume using specific hand gestures.
- **Video Call Integration**
    - Integrate TokBox iframe for real-time video communication using gestures.

- **Visual Feedback**
    - Show visual indicators such as detected landmarks, bounding boxes, gesture name, translation status, and frame rate (FPS).
- **Mode Switching**
    - Recognize and switch between different operational modes (Cursor Control, Scroll, Volume Control) based on gestures.

3.3 Non functional Requirements

Non-functional requirements describe how the system performs rather than specific behaviors. For SilentCue, the non-functional requirements are:

- **Performance**
    - The system should maintain smooth and real-time responsiveness with minimal lag (<100ms delay).
- **Usability**
    - The GUI must be intuitive, accessible to people of different age groups, and easy to navigate without prior technical knowledge.
- **Accuracy**
    - Gesture recognition should have an accuracy rate of at least 90% in controlled environments.
- **Security**
    - Any data stored (like custom gestures) must be securely handled and user privacy should be maintained.
- **Scalability**
    - The system should allow for future updates like adding new gestures, multiple languages, or advanced gesture animations.
- **Reliability**
    - The application should work consistently across different hardware setups without frequent crashes or errors.
- **Portability**
    - The system should be able to run on various operating systems such as Windows, Linux, and macOS.

3.4. Hardware, Software , Technology and tools utilized

Hardware Requirements:

- Processor: Intel Core i3 or higher (Core i5 or better recommended)
- RAM: Minimum 4GB (8GB recommended)
- Storage: Minimum 2GB of free space
- Webcam: Integrated or external HD camera
- GPU (optional): For better performance in hand tracking tasks

Software Requirements:

- Operating System: Windows 10 or later, Ubuntu 18.04 or later, macOS 10.14 or later
- Programming Language: Python 3.x
- Web Technologies: HTML, CSS, JavaScript (for GUI)
- Libraries: OpenCV, MediaPipe, pyautogui, pycaw, scikit-learn, Tkinter

3.5 Constraints

1. **Environmental Dependency**
   Gesture recognition accuracy may vary based on lighting conditions, camera quality, and background noise.

2. **Real-Time Processing**
   Real-time tracking and recognition require efficient models and optimized code to avoid lag on mid-range devices.

3. **Dataset Limitations**
   Lack of large publicly available datasets for some custom gestures may affect the training process.

4. **User Variability**
   Differences in hand sizes, skin tones, and gesture styles among users can affect recognition accuracy.

5. **Internet Dependency**
   TokBox-based video calls require stable internet connectivity for smooth performance.

6. **Hardware Requirement**
   Lower-end systems without GPUs may experience slower processing speeds, affecting the user experience.

Chapter 4: Proposed Design

4.1 Block diagram of the system

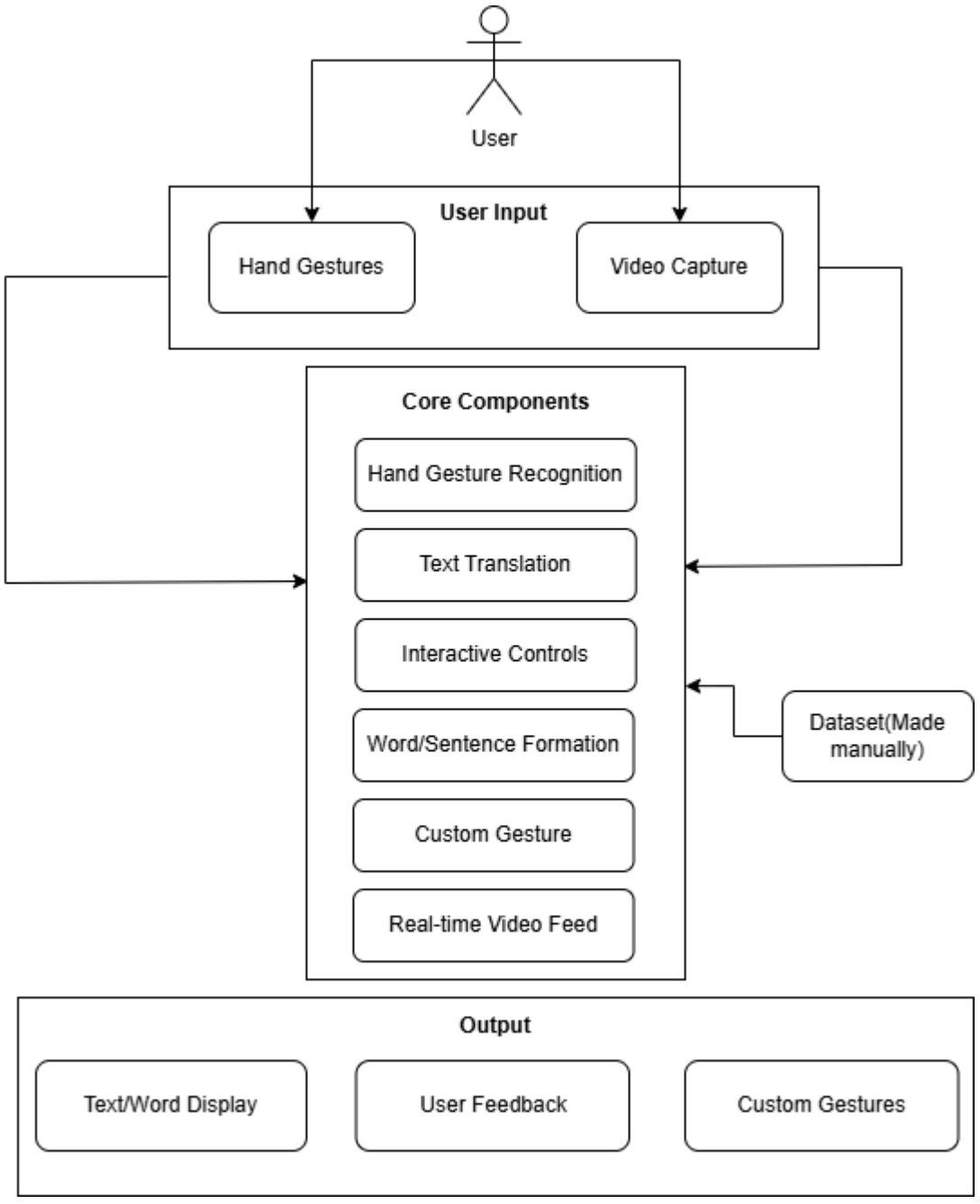


Fig 4.1: Block Diagram

Figure 4.1 shows the workflow of the SilentCue system.The user provides input through hand gestures and live video capture. These inputs are processed by the core components, which handle gesture recognition, text translation, interactive controls, word/sentence formation, and custom gesture training using a manually created dataset. A real-time video feed supports user interaction. The final output includes text/word display, user feedback, and integration of custom gestures, enabling smooth and personalized sign language communication.

4.2 Modular design of the system

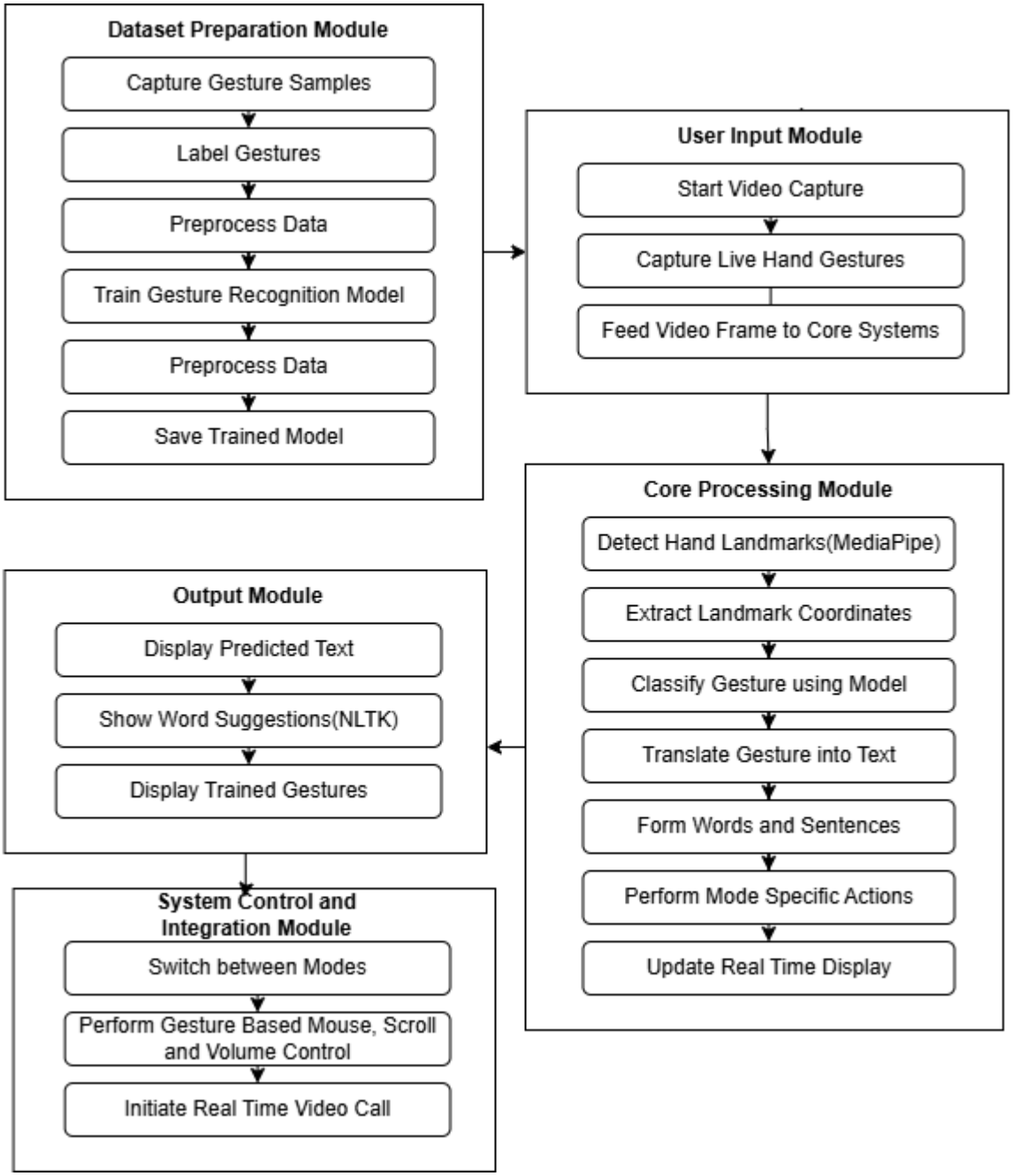


Fig 4.2: Modular Diagram

1. Dataset Preparation Module

- **Dataset Collection:** Hand gesture images are captured using a webcam for both standard ASL gestures and user-defined custom gestures.
- **Labeling:** Each captured sample is labeled according to the gesture it represents.
- **Preprocessing:** Collected images are resized, normalized, and optionally augmented to enhance model training.
- **Model Training:** A machine learning model is trained on the preprocessed dataset to recognize gestures accurately.

- **Model Saving:** The trained model is saved using serialization (e.g., pickle) for real-time prediction in the core system.

2. User Input Module

- **Start Webcam Feed:** The system activates the webcam and begins capturing live video frames.
- **Perform Hand Gestures:** The user performs hand gestures in front of the webcam.
- **Send Frames to Core System** Captured frames are continuously passed to the core processing module for analysis.

3. Core Processing Module

- **Hand Landmark Detection:** MediaPipe is used to detect hand landmarks (x, y coordinates) in the video frames.
- **Feature Extraction:** Landmark coordinates are extracted to serve as features for classification.
- **Gesture Prediction:** The trained model classifies the gesture based on the extracted features.
- **Text Translation:** Recognized gestures are mapped to corresponding letters, words, or custom labels.
- **Word and Sentence Formation:** Recognized characters are concatenated into meaningful words and sentences.
- **Mode-specific Actions:** Based on user mode (Translation, Cursor, Scroll, Volume), appropriate actions are triggered.
- **Real-Time GUI Update:** The predicted output is updated and displayed instantly on the GUI.

4. Output Module

- **Display Text Predictions:** The GUI shows the recognized letters, formed words, and full sentences.
- **Word Suggestions:** Using NLTK, word suggestions are generated based on partial word input.
- **Display Trained Gestures:** A list of user-trained custom gestures and their translations is displayed.

5. System Control and Integration Module

- **Mode Switching:** The user can switch between translation mode, cursor control mode, scroll mode, and volume control mode using gestures.
- **Gesture-Based Controls:** Gestures are used to move the mouse, scroll pages, click, and adjust volume via pyautogui and pycaw libraries.
- **Real-Time Video Communication:** TokBox integration allows users to initiate and participate in live video calls.

4.3 Detailed Design

1. Level 0 - Data Flow Diagram

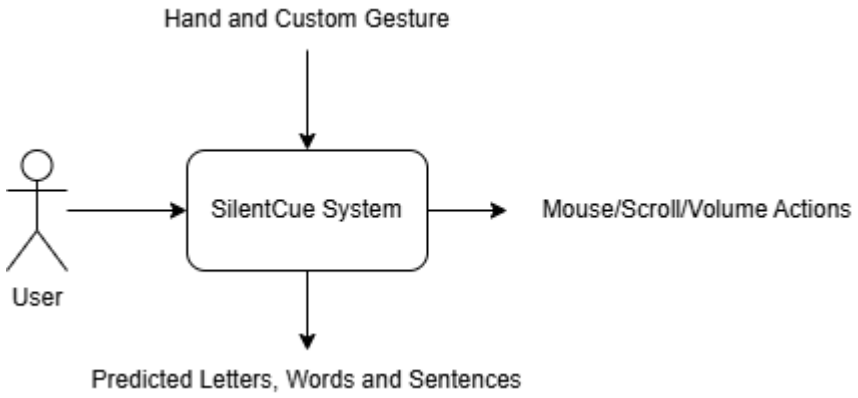


Fig 4.3: DFD Level 0

SilentCue System (Central Process):

- This is the main process that encompasses all the system's functionalities. At Level 0, SilentCue is treated as a "black box" without diving into its internal workings. It handles gesture recognition, translation, gesture-based system control, and video communication in a unified manner.

User (External Entity):

- This represents the person interacting with the system. The user provides various inputs like hand gestures, custom gesture data for training, mode selections (such as translation or control modes), and video call initiation. The user also receives the processed outputs such as translated text, system actions (like cursor movement or volume adjustment), and video call windows.

Data Flows:

- Gesture Input: The user performs hand gestures or provides custom gesture data for the system to recognize and translate.
- Mode Selection: The user chooses the operation mode — whether to translate gestures into text, control the system, or initiate a video call.
- Translated Output: The system provides real-time translation of gestures into text (letters, words, or sentences).
- System Control Output: Based on the detected gestures, the system controls the mouse, scrolls, or adjusts volume accordingly.

2) Level 1 Data Flow Diagram

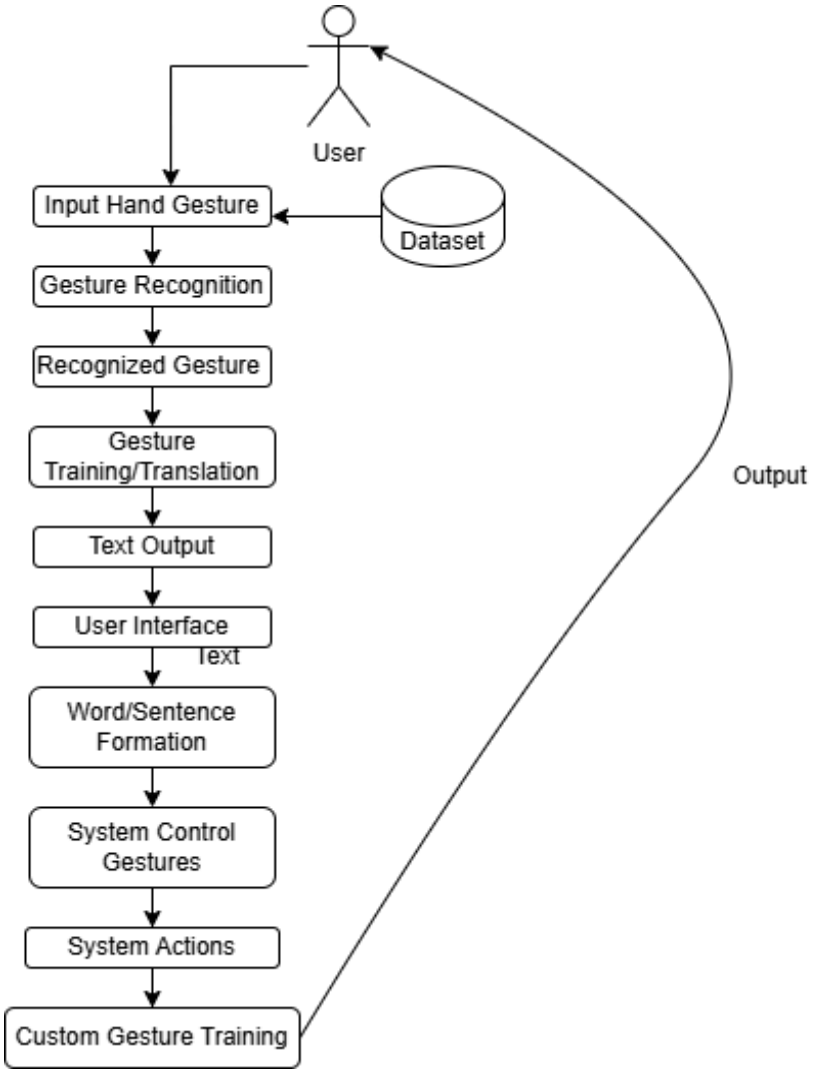


Fig 4.4: DFD Level 1

Level 1 Data Flow Diagram for SilentCue - Sign Language Recognition

The Level 1 Data Flow Diagram (DFD) for SilentCue illustrates how the system works to recognize sign language gestures and translate them into text, facilitate system control through gestures, and support video communication. This diagram breaks down the major components and their interactions, reflecting the system's primary processes.

1. User Input Hand Gesture

The process begins with the user providing a hand gesture as input. This gesture could be part of the American Sign Language (ASL) or a custom gesture that the user has trained the system to recognize. The system uses **MediaPipe** and **OpenCV** to capture and process the hand gesture.

2. Gesture Recognition

The input gesture is passed to the **Gesture Recognition** process, which identifies the gesture using a trained model. The recognition process is powered by machine learning models, ensuring that gestures are accurately detected and classified.

3. Recognized Gesture

Once the gesture is recognized, the system identifies the specific ASL letter or sign. This recognized gesture is then passed to the **Translation Process** to convert it into a meaningful output, such as a word or sentence.

4. Gesture Training and Translation

If the recognized gesture is a part of the custom gestures trained by the user, it is passed to the **Gesture Training** process, allowing the system to expand its database of gestures. The **Translation Process** converts the recognized gesture into a translated text or word, which is displayed in the **User Interface** for the user to read.

5. Text Output and User Interface

The system outputs the translated text through the **User Interface**, allowing the user to see the corresponding ASL translation. The user can interact with the interface, which supports word and sentence formation, and use interactive controls like backspace or case toggling.

6. System Control Gestures

SilentCue also supports controlling system actions such as mouse movements, scrolling, and volume adjustment through hand gestures. These gestures are captured and passed to the **System Action** process, which then uses libraries like **pyautogui** and **pycaw** to perform the actions on the user's computer.

7. Custom Gesture Training

If the user wishes to add new, personalized gestures, they can train the system to recognize these custom gestures. The **Custom Gesture Training** process allows users to input and store new gestures in the **Gesture Database**, ensuring that the system can adapt to individual needs.

8. Database Interactions

- **Gesture Database**: Stores recognized gestures, including both predefined and custom gestures, allowing the system to reference and retrieve gesture data during recognition and training processes.

- **Translation Database**: Stores translations of recognized gestures to facilitate quick access and improve efficiency in the translation process.

Summary:

The Level 1 DFD for SilentCue demonstrates the flow of data between various processes like gesture recognition, translation, system control, and video communication. By breaking down the system into distinct processes, it highlights how SilentCue adapts to users' needs, helping them communicate effectively through ASL while also offering assistive technology features like system control and video calls.

3) Activity Diagram

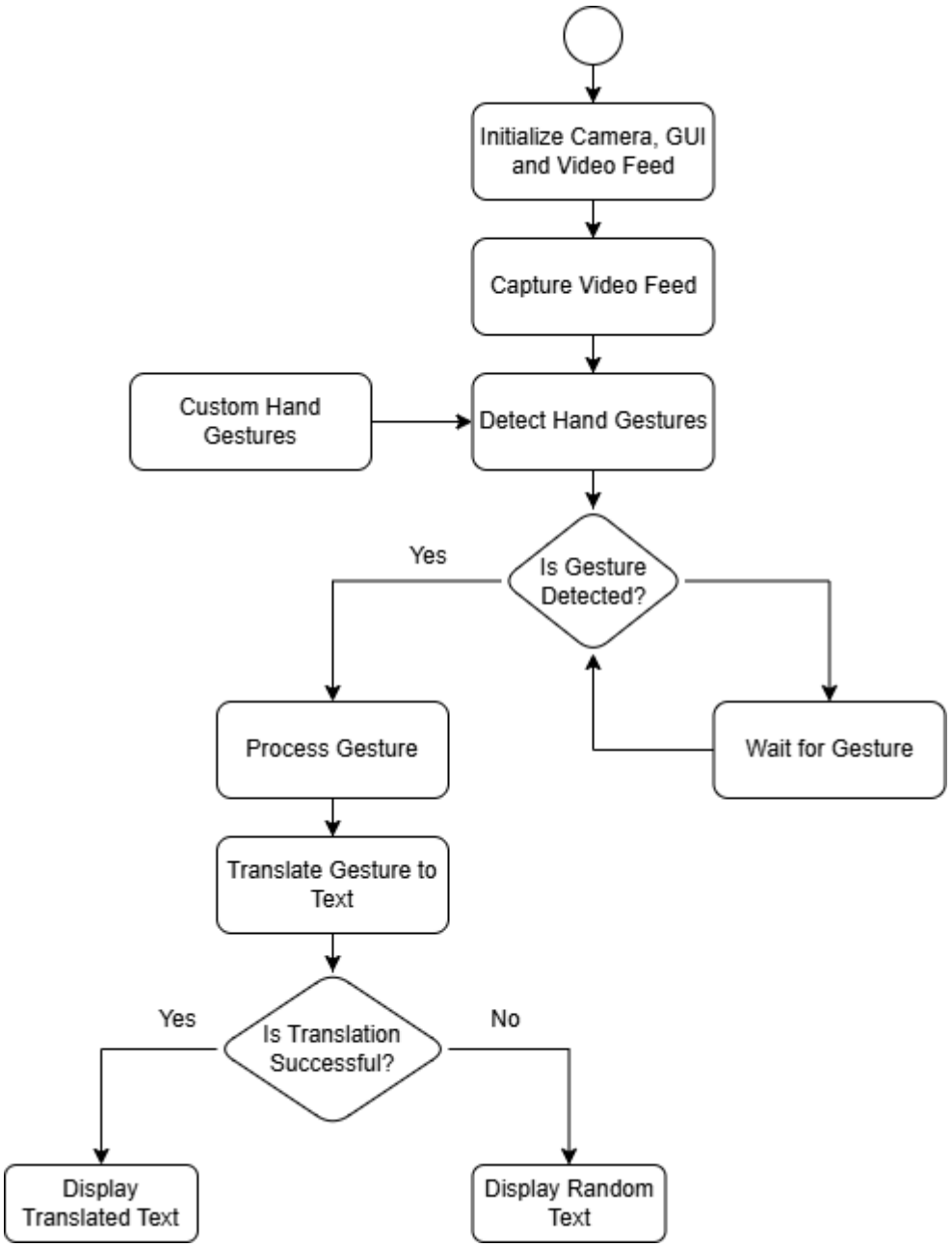


Fig 4.5: Activity Diagram

- The Hand Gesture Translation System starts by initializing the camera, graphical user interface (GUI), and the video feed.
- The system captures a continuous video feed in real-time from the camera.
- Using the captured feed, it detects hand gestures, including any predefined custom gestures.
- The system checks if a hand gesture is detected:
    - If no gesture is detected, it waits and continues monitoring.
    - If a gesture is detected, it proceeds to the next step.
- Once a gesture is detected, the system processes the gesture to extract meaningful features.
- The processed gesture is then translated into text using a gesture-to-text translation mechanism.
- The system verifies if the translation is successful:
    - If the translation is successful, it displays the translated text to the user.
    - If the translation fails, the system displays a random text output as a fallback.
- The system continues this cycle to allow real-time hand gesture translation and display.

Chapter 5: Implementation of the Proposed System

5.1. Methodology employed for development

The methodology followed for the development of the SilentCue - Sign Language Recognition System is modular and iterative. The system was divided into several interdependent modules, including hand gesture recognition and translation, system control via gestures (mouse, scroll, and volume), and a custom gesture training and translation interface. Each module was developed and tested individually, ensuring proper functionality before integration into the final system.

A real-time, data-driven approach was used, where gesture recognition was based on a custom dataset of American Sign Language (ASL) gestures, and gesture control actions were mapped to system-level functions using live hand landmark detection. The development process followed a logical flow, starting from gesture detection and prediction to interactive control and user-defined gesture translation, ensuring accuracy, usability, and real-time performance at each stage.

5.2 Algorithms and Flowcharts for the Respective Modules Developed

For each module, specific algorithms and flowcharts were designed to define the logic and process flow.

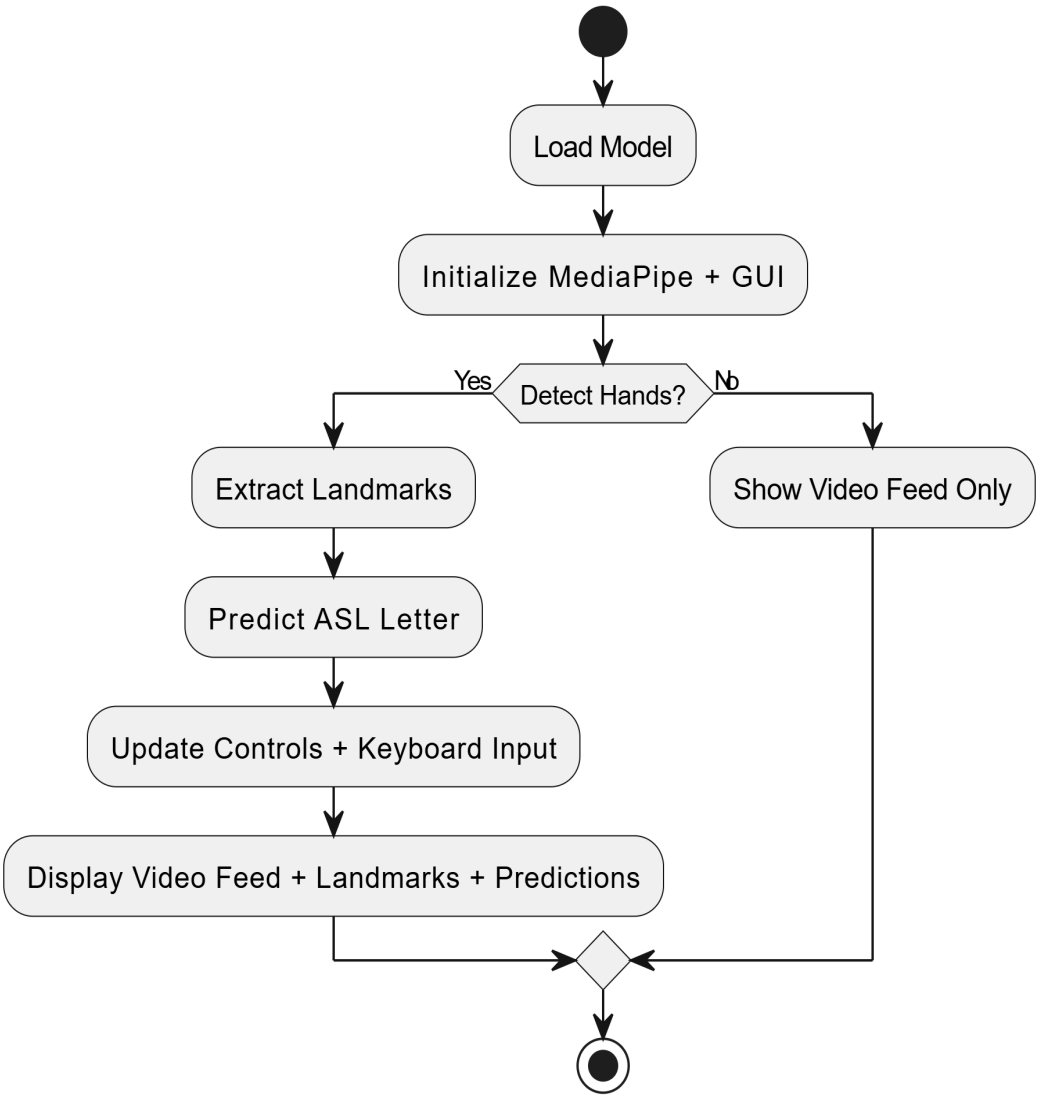
ASL Gesture Recognition and Translation

The **ASL Gesture Recognition and Translation** module begins when the user starts interacting with the system via the webcam. The system first captures the video feed and uses MediaPipe to detect hand landmarks in real-time. Once hand landmarks are detected, the system passes the coordinates of these landmarks to the pre-trained machine learning model.

The model then predicts the corresponding American Sign Language (ASL) letter based on the gesture formed by the user. As the user continues making gestures, the system continuously updates the predicted ASL letter, displaying it on the graphical user interface (GUI) using Tkinter.

The system also manages word formation by recognizing continuous gestures and displaying them as text, updating as new gestures are made. Suggestions for words are provided based on the current sequence of gestures, helping speed up typing. The system also allows the user to interact with the GUI for actions like case toggling, clearing the word/sentence, or saving the sentence to a file.

Finally, the system ensures smooth operation by adding slight delays between predictions to avoid constant switching of letters. The model is trained on a custom dataset tailored to recognizing ASL gestures, ensuring accurate translation of the user’s gestures into text.

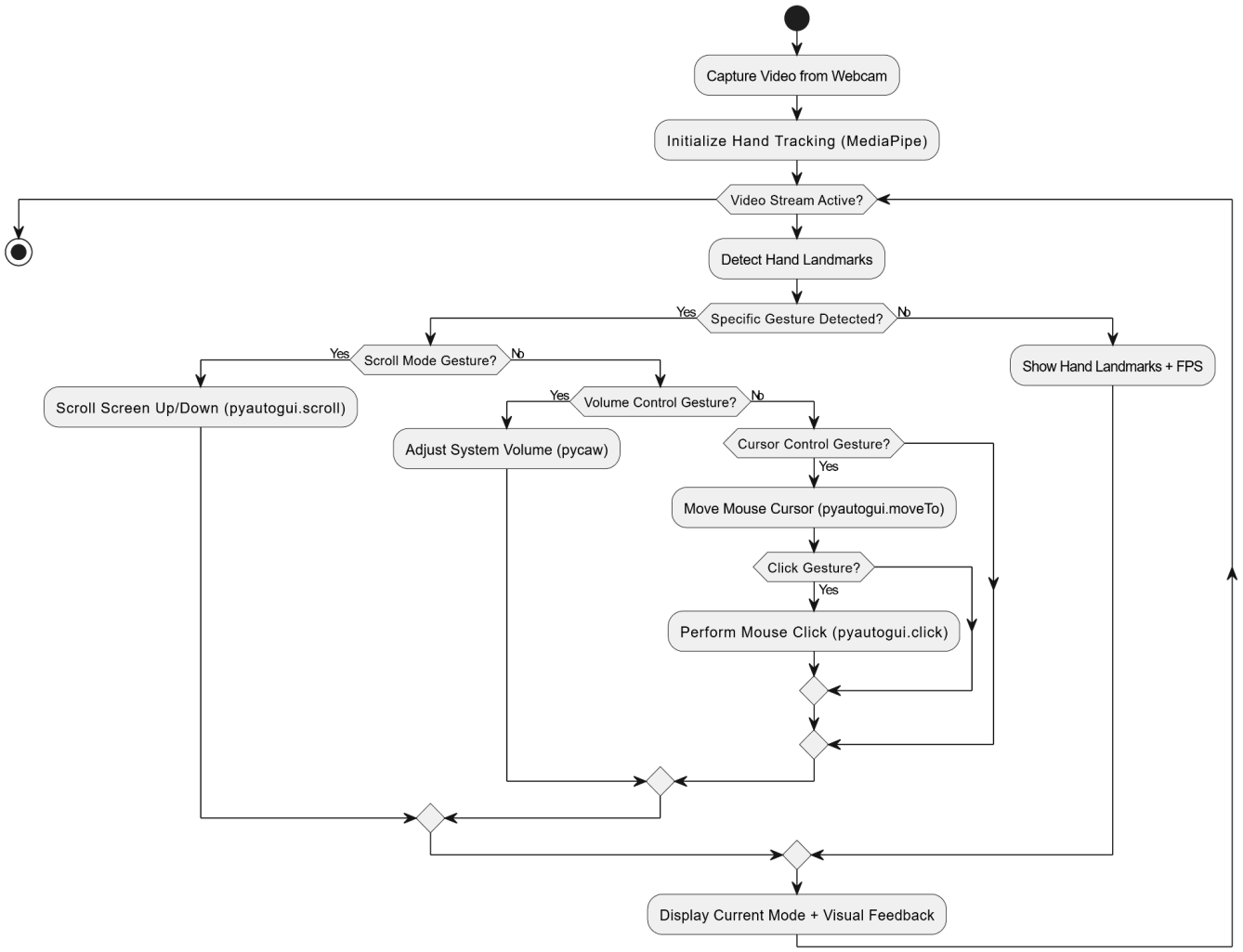


Hand Gesture Control for Mouse, Scroll, and Volume

The **Hand Gesture Control** module begins when the system captures the video feed and uses MediaPipe to detect hand landmarks in real-time. The system continuously monitors the positions of the landmarks, identifying the user's hand gestures. Based on the gesture detected, the system switches between different modes: scroll, volume control, and cursor control.

If the system detects a gesture for scroll mode (e.g., index finger raised for scroll up), it triggers the corresponding action (scroll up or down) using pyautogui. In volume control mode, when the system detects a pinch gesture (thumb and index finger), it adjusts the system's volume through pycaw, mapping the pinch distance to a volume range. For cursor control mode, the system tracks the index finger's position and moves the mouse cursor accordingly. If the thumb-down gesture is detected, a mouse click is simulated.

The system provides visual feedback, showing the active mode and any actions being performed, such as the volume level or mouse position. It ensures smooth operation by updating the interface in real-time and switching modes as needed based on the user's hand gestures.



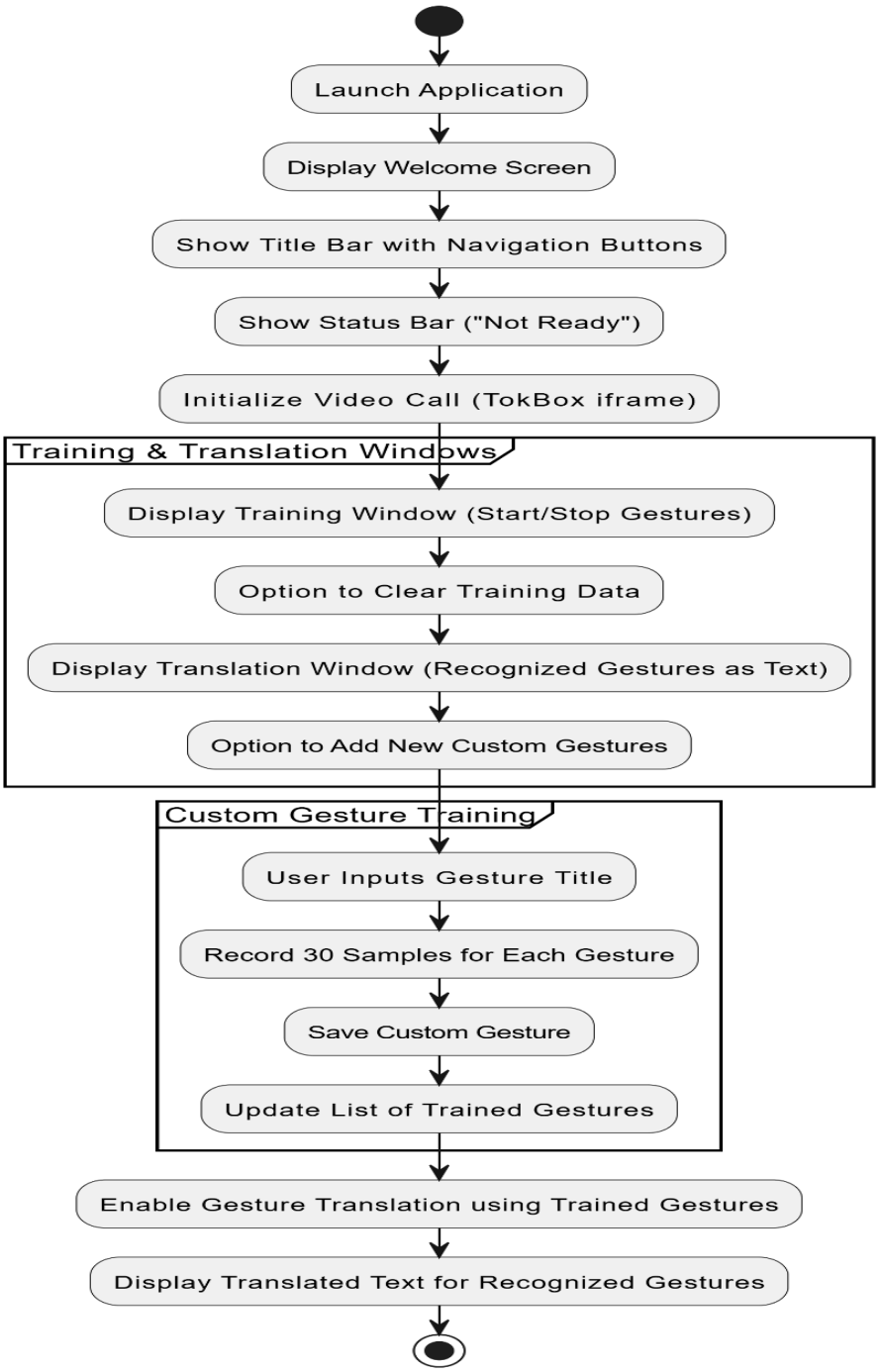
Gesture Training & Translation (User Interface)

The **Gesture Training & Translation** module starts when the user interacts with the system to either train new custom gestures or translate existing ones. Upon launching the app, the user is greeted with a welcome screen, and the interface displays the current stage (e.g., "Train Gestures") in the title bar. The user can navigate through various stages of training and translation via the interface buttons.

To train new custom gestures, the user selects the option to add a new gesture, and the system prompts the user to record 30 samples of the gesture. These samples are then processed, and the gesture is added to the list of trained gestures, each associated with a title and its translation. The trained gestures are displayed as cards in the interface, allowing the user to see all recognized gestures and their corresponding translations.

In the translation phase, the system listens for recognized gestures, translating them into corresponding text and displaying them in real-time on the screen. The user can add more custom gestures as needed, expanding the system's gesture vocabulary. If desired, the user can also clear training data to re-train gestures. The system ensures that trained gestures are saved and available for future use, continuously updating the translation display as new gestures are performed.

The user interface is built using HTML, CSS, and JavaScript, while the backend leverages Python with OpenCV and MediaPipe for gesture recognition, allowing the system to be intuitive and easy to navigate for users.



5.3 Datasets Source and Utilization

The datasets used in this system consist of:

- **Custom ASL Gesture Dataset**: A proprietary collection of hand gesture images and videos, specifically focused on American Sign Language (ASL) gestures.

- **Custom Gesture Training Dataset**: A set of hand gestures recorded and labeled by the user, used for training custom gestures that expand the system's vocabulary.

- **Word Suggestions Dataset**: A list of common words sourced from the NLTK word list, used for suggesting possible word completions based on recognized ASL gestures.

Utilization:

- The **Custom ASL Gesture Dataset** is used to train the machine learning model for accurate ASL gesture recognition, allowing real-time translation of gestures into text. This dataset was tested alongside publicly available datasets to ensure robustness and generalization.

- The **Custom Gesture Training Dataset** is utilized when users add their own gestures to the system, enabling the system to recognize and translate newly trained gestures into text after sufficient training samples are provided.

- The **Word Suggestions Dataset** is used during word formation to suggest possible words based on the gestures currently being made, speeding up the communication process and improving user experience.

Source:

- The **Custom ASL Gesture Dataset** was created by the development team, focusing on capturing real-time ASL gestures with OpenCV. It was then tested with publicly available datasets to validate the system's performance and accuracy in recognizing ASL gestures.

- The **Custom Gesture Training Dataset** is manually generated by the user through the system's gesture recording feature.

Chapter 6: Testing of the Proposed System

6.1 . Introduction to testing

Testing is a critical phase in software development that ensures the system performs as expected and meets user requirements. It helps in identifying bugs, verifying functionalities, and validating performance. Although SilentCue is still in its development stage, a structured approach towards testing has been conceptualized to ensure the system's readiness for real-world usage. This phase includes planning different levels of testing such as unit testing, integration testing, system testing, and user acceptance testing.

6.2 Types of Tests Considered

- **Unit Testing**

  Each individual module, such as hand landmark detection, gesture classification, and text translation, was tested separately to ensure they performed as intended. Testing was done using dummy gesture samples to verify correct detection and translation.
- **Integration Testing**

  After unit testing, modules were integrated and tested together. For example, after a gesture is recognized, the translation output and GUI update were verified in sequence to ensure smooth flow.
- **System Testing**

  The complete system was run end-to-end to check overall functionality, including the custom gesture training module, real-time hand tracking, word formation interface, and gesture-to-text translation.
- **User Acceptance Testing (UAT)**

  The application was informally tested by a small group of users (friends and classmates) to gather feedback on usability, accuracy, and performance. Based on their feedback, minor UI improvements and gesture sensitivity adjustments were conceptualized.

6.3 Various Test Case Scenarios Considered

Sr. No	Test Scenario	Description	Expected Outcome
1	Basic Gesture Recognition	Test recognition of standard gestures like "start" and "stop" under normal conditions.	Gestures are correctly detected and translated.

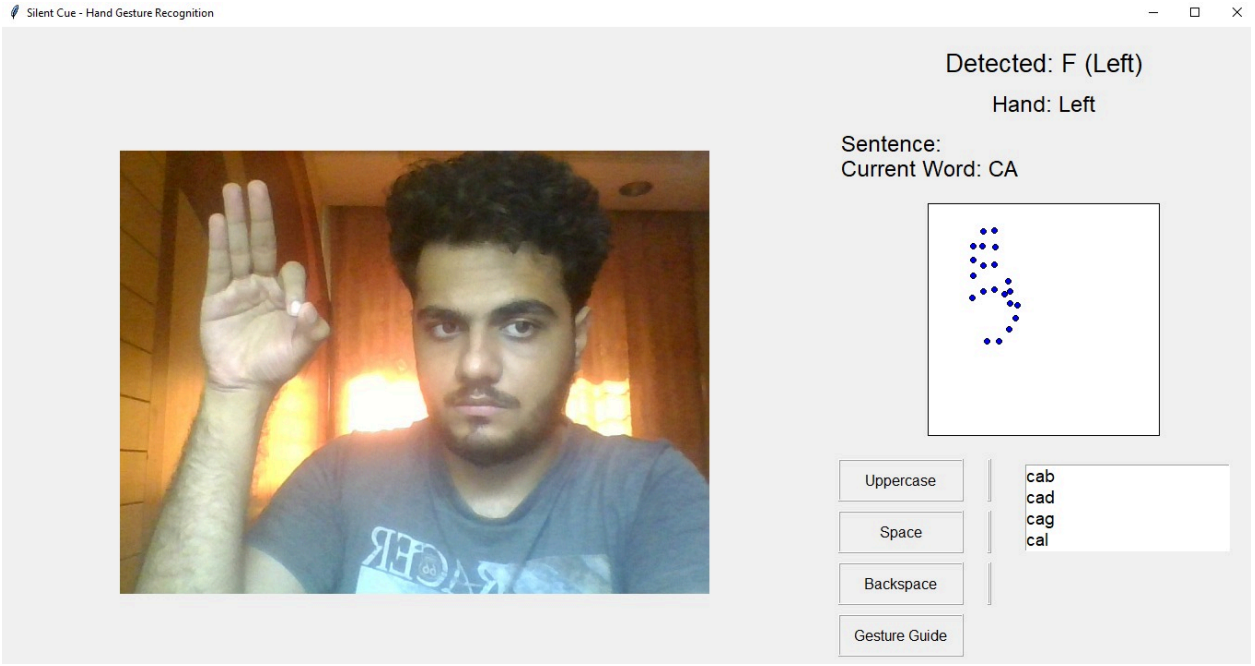
2	Custom Gesture Training	Train a new custom gesture and check for recognition accuracy after training.	Custom gestures are successfully recognized.
3	Real-Time Translation	Perform continuous gestures to test real-time translation performance.	Gesture-to-text happens with minimal delay.
4	Mode Switching	Switch between Cursor, Scroll, and Volume control modes using different gestures.	System switches modes correctly based on gestures.
5	Edge Case Handling	Perform gestures with rapid movement, partial hand visibility, or multiple hands in frame.	System handles the situation or prompts the user appropriately.
6	Low-Light Condition Detection	Perform gestures under low-light or dim background conditions.	Gestures are detected but slight accuracy drop may occur.
7	System Response Time	Measure the delay between gesture performed and system response.	System responds within acceptable delay (0.5-1 second).
8	Background Complexity Test	Test system performance with cluttered or dynamic backgrounds (e.g., moving objects).	Gestures were still detected but background simplicity preferred.

6.4 Inference Drawn from the Test Cases

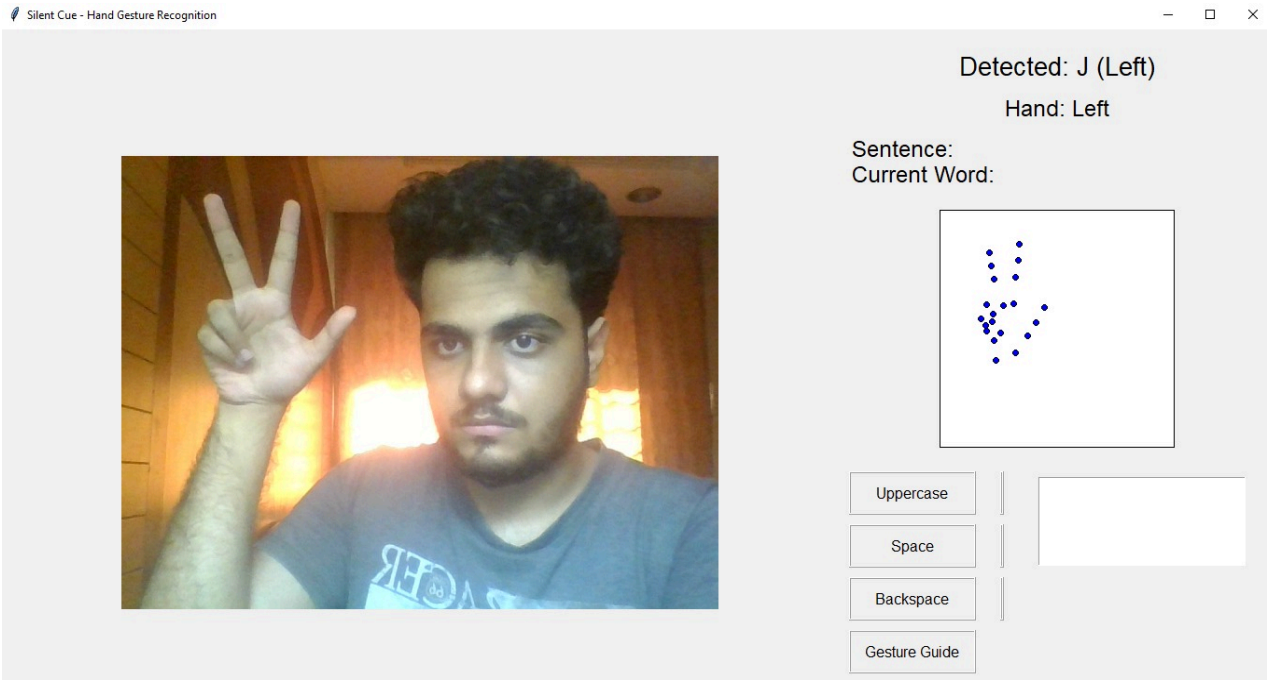
- Standard and custom gestures are accurately recognized under normal conditions.
- Real-time translation and mode switching work smoothly with minimal delay.
- The system handles edge cases and low-light scenarios reasonably well, with slight performance drops
- Response time remains within 0.5–1 second, maintaining user experience expectations.
- Gestures are detectable in complex backgrounds, but simpler backgrounds enhance accuracy.

Chapter 7: Results and Discussion

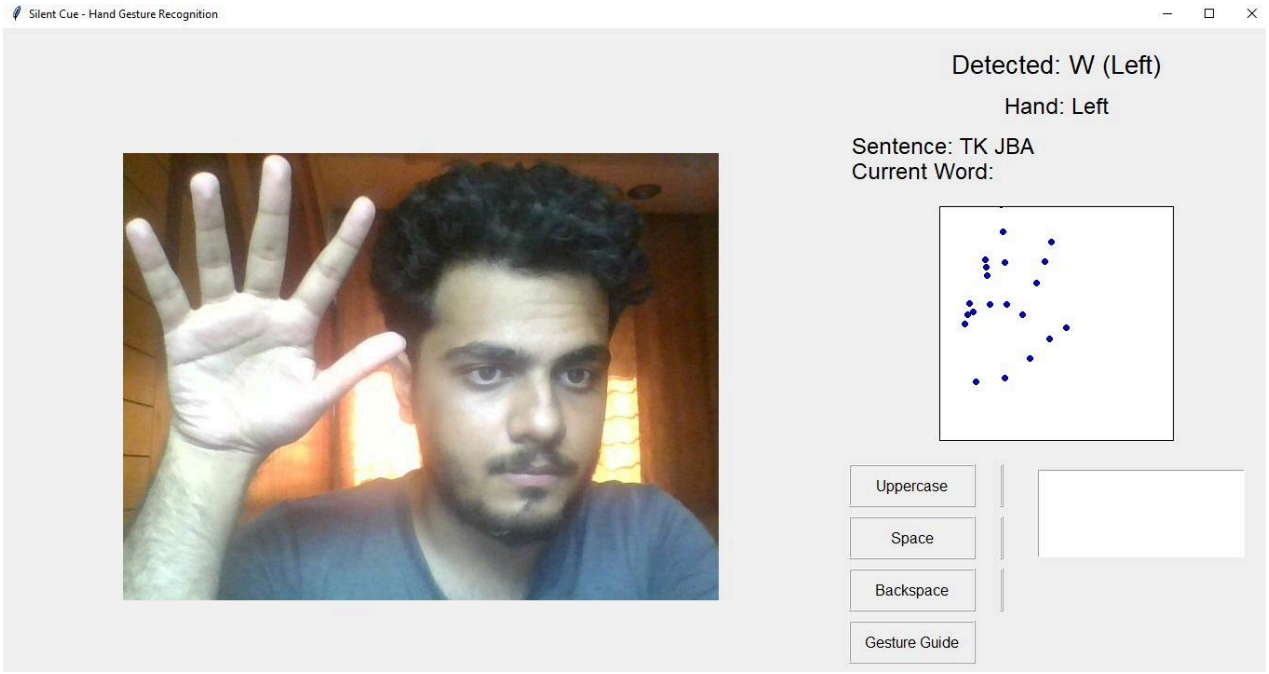
7.1. Screenshots of User Interface (UI) for the respective module



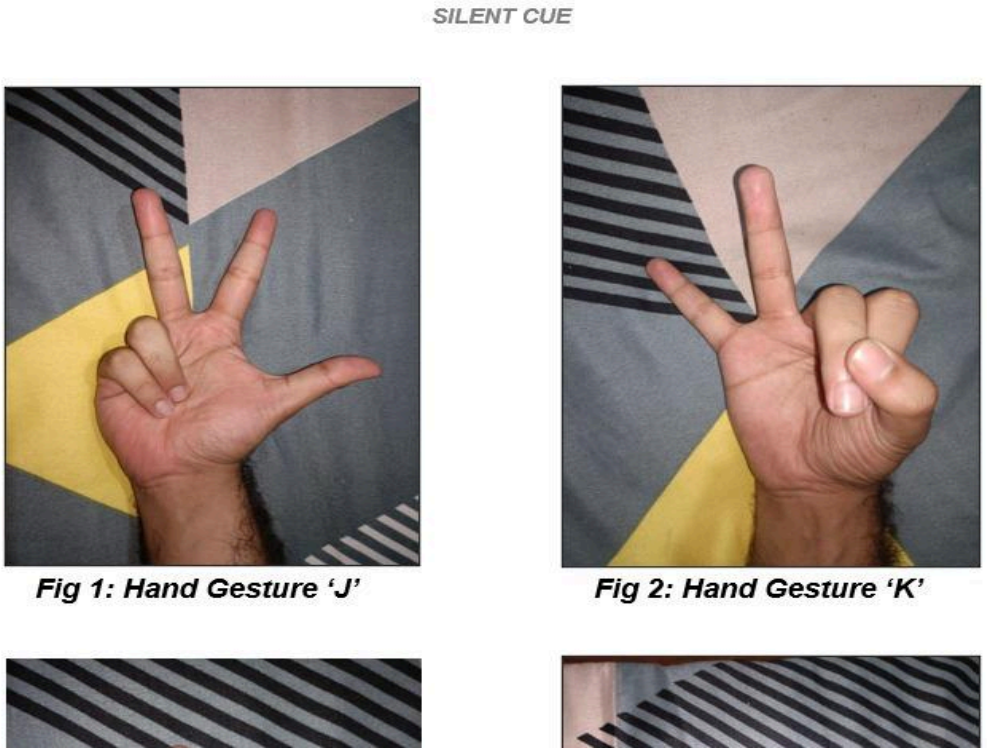
7.1(a) Word Formation and Live Suggestion



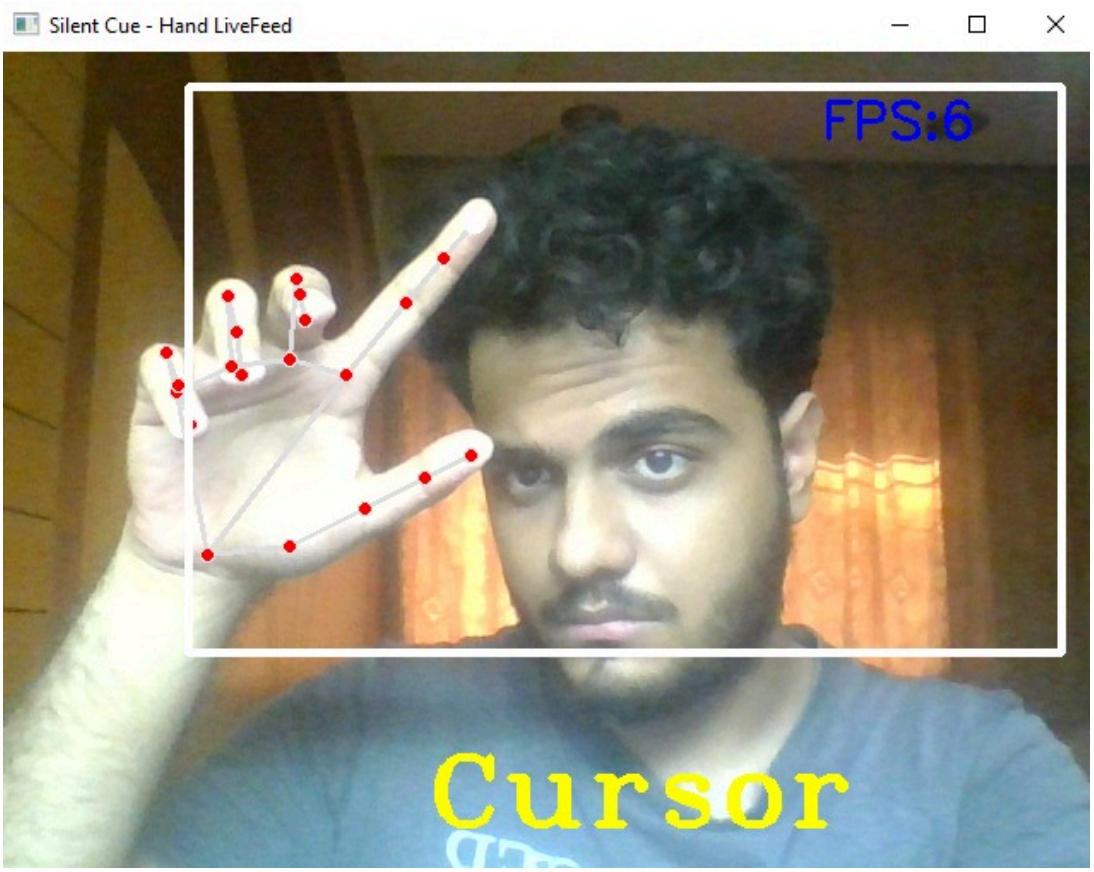
7.1(b) GUI and Gesture Detection



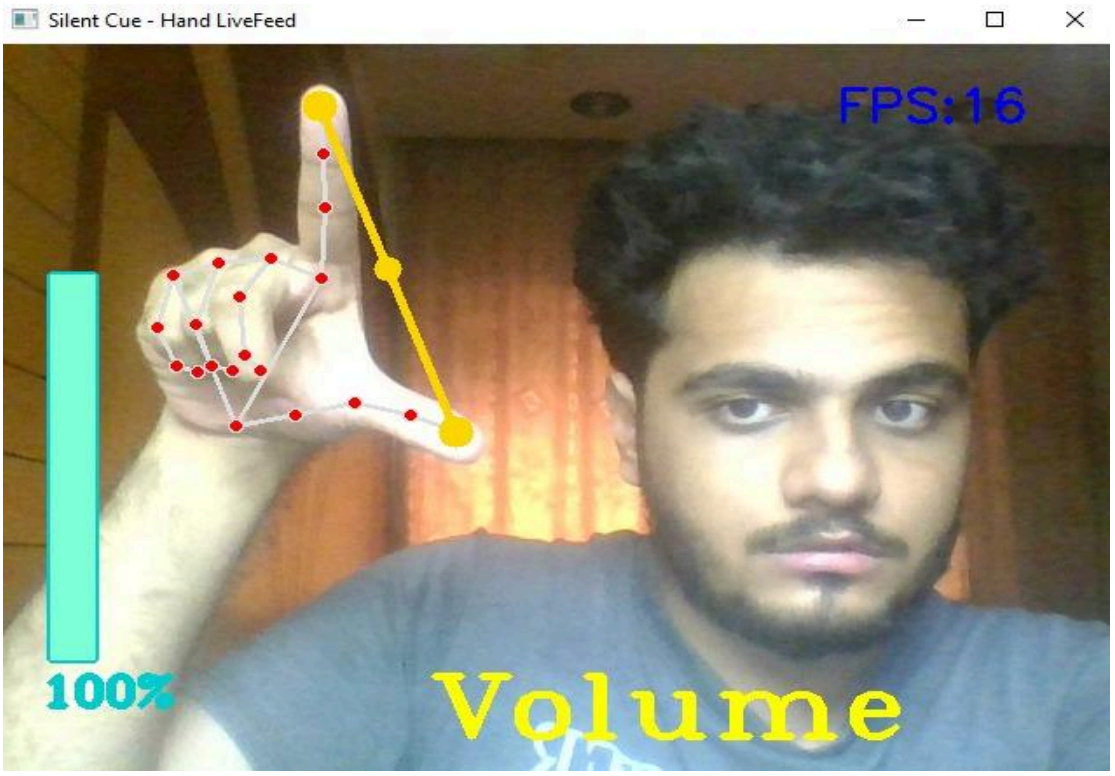
7.1(c) Sentence Formation



7.1(d) Custom Hand Gestures



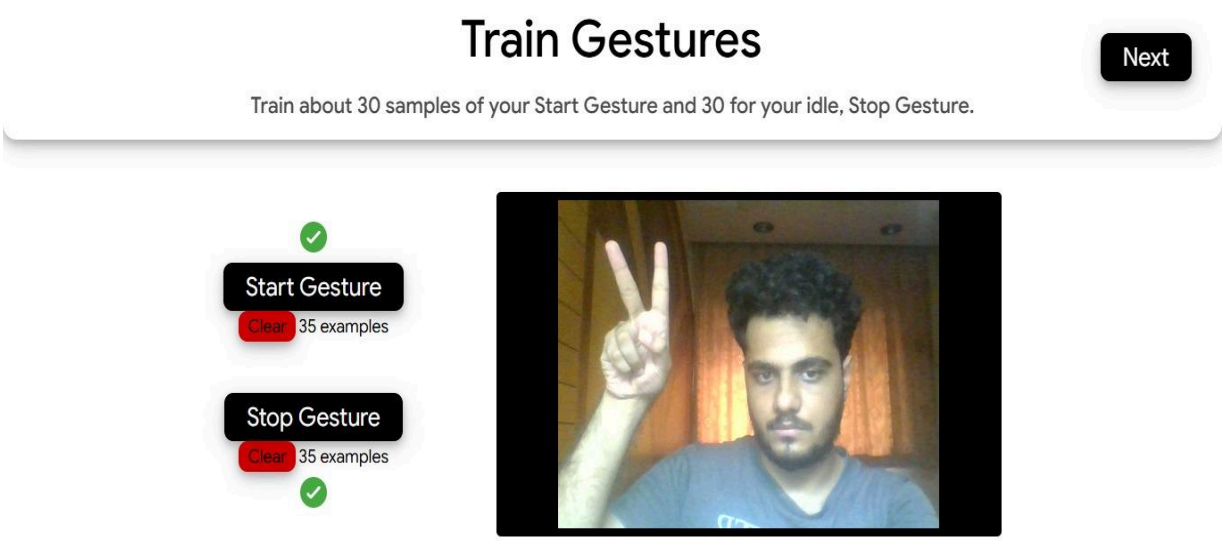
7.1(e) Mouse Movement: Cursor



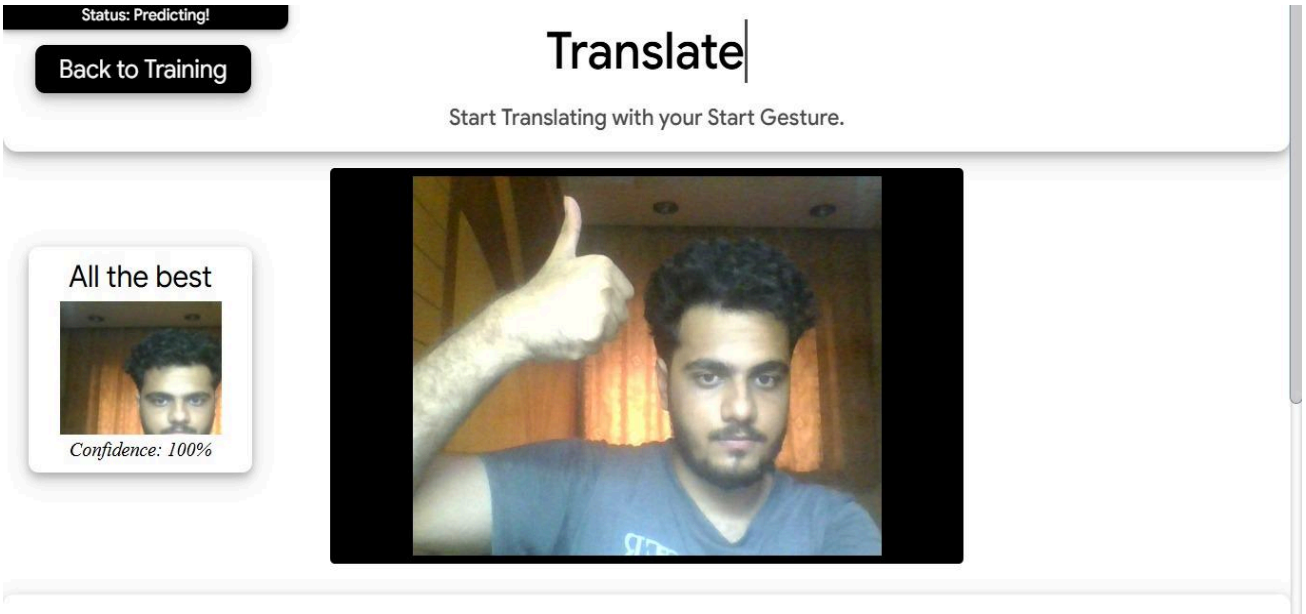
7.1(f) Mouse Movement: Volume

7.1(g) Mouse Movement: Scroll Up

7.1(h) Mouse Movement: Scroll Down



7.1(i) Training Custom and Common Gestures



7.1(j) Common Word Detection

7.2. Performance Evaluation measures

1. Accuracy of Gesture Recognition

The accuracy of the system in recognizing hand gestures is a critical performance metric. It reflects how closely the system's gesture detection matches the actual intended gestures made by the user.

Accuracy = (Correct Gesture Recognitions / Total Gestures Performed) × 100

2. Precision and Recall

If the system classifies gestures into specific actions (like cursor control, scroll, volume control, or custom gestures), precision and recall help measure the classification's effectiveness.

- **Precision** measures how many gestures identified by the system were correctly recognized.
- **Recall** measures how many actual performed gestures were successfully detected by the system.

  Precision = True Positives / (True Positives + False Positives)

  Recall = True Positives / (True Positives + False Negatives)

3. Response Time

This measures the average time taken by the system to recognize a gesture and perform the corresponding action. Lower response times indicate better real-time performance and usability for gesture interaction.

Average Response Time = Total Response Time / Number of Gestures Detected

4. System Throughput

Throughput refers to the number of gesture actions (cursor moves, scrolls, volume adjustments, translations) the system can process per minute. It helps evaluate the system's ability to handle continuous gestures without lag.

Throughput = Number of Gestures Processed / Time Taken

5. Feedback Quality

The effectiveness of the visual feedback (hand landmarks, mode indicators) and functional response (e.g., mouse movement, volume change) is assessed. Effective feedback ensures better user interaction and system trust.

Measurement criteria:

- User ratings on gesture recognition accuracy
- Alignment between performed gesture and system response (without user confusion)

6. Error Rate

This metric captures the number of times the system fails to detect a gesture correctly or misclassifies it into the wrong mode. A lower error rate indicates a more reliable and stable recognition model.

Error Rate = (Number of Recognition Errors / Total Gesture Attempts) × 100

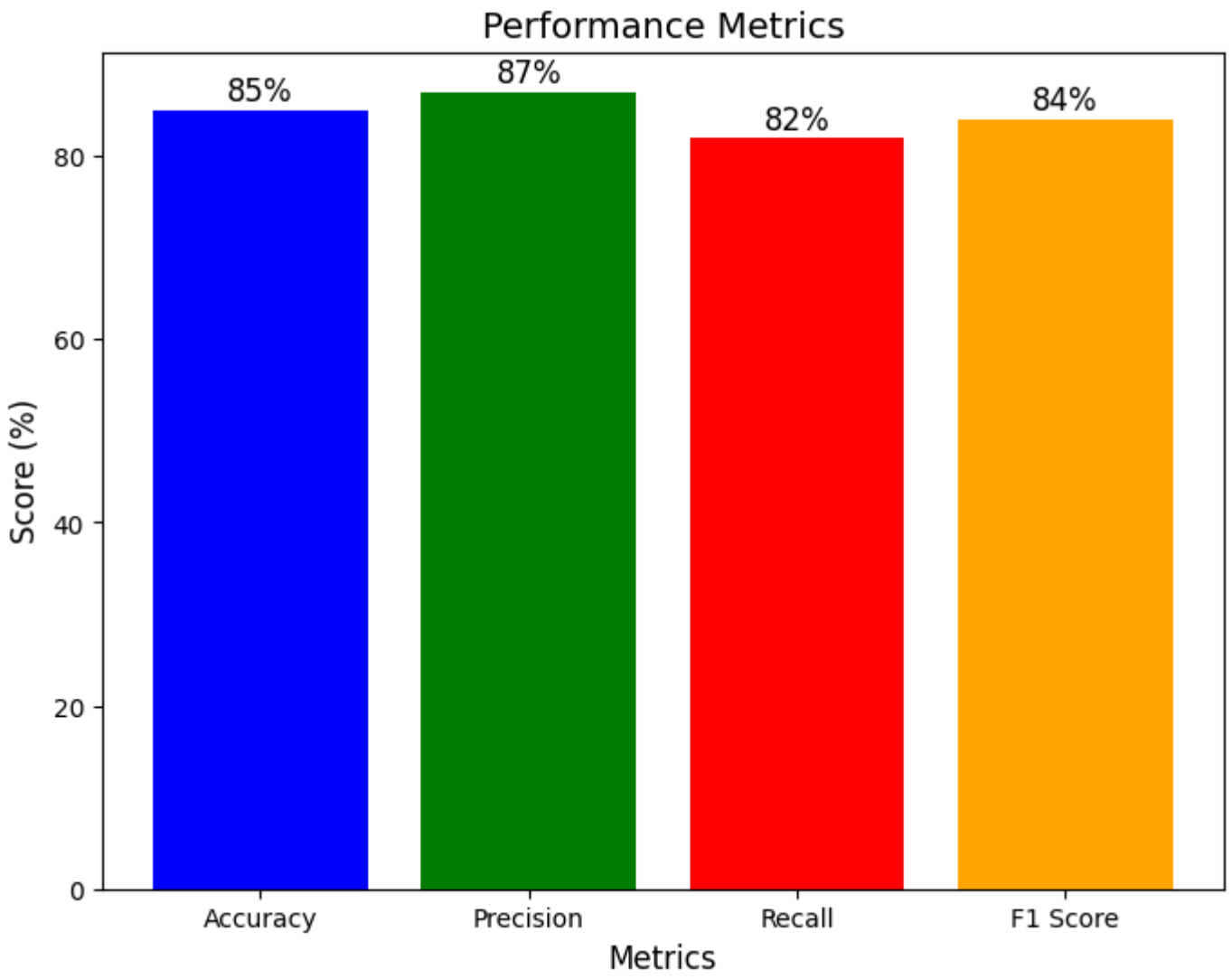


Fig 7.2(a) Calculated Metrics

7.3. Input Parameters / Features considered

Gesture Input Parameters

Users can set up personalized gesture recognition by providing inputs like hand gesture categories, specific gesture types (e.g., "Hello," "Goodbye," "All the Best," "Peace"), and environmental conditions (e.g., lighting, background). These parameters help adapt the gesture recognition model to the user's unique needs and ensure the system provides accurate and responsive gesture classification. The complexity of recognition adjusts based on user input, including whether they are a beginner or more advanced in hand gesture communication.

7.4 Comparison of results with existing systems

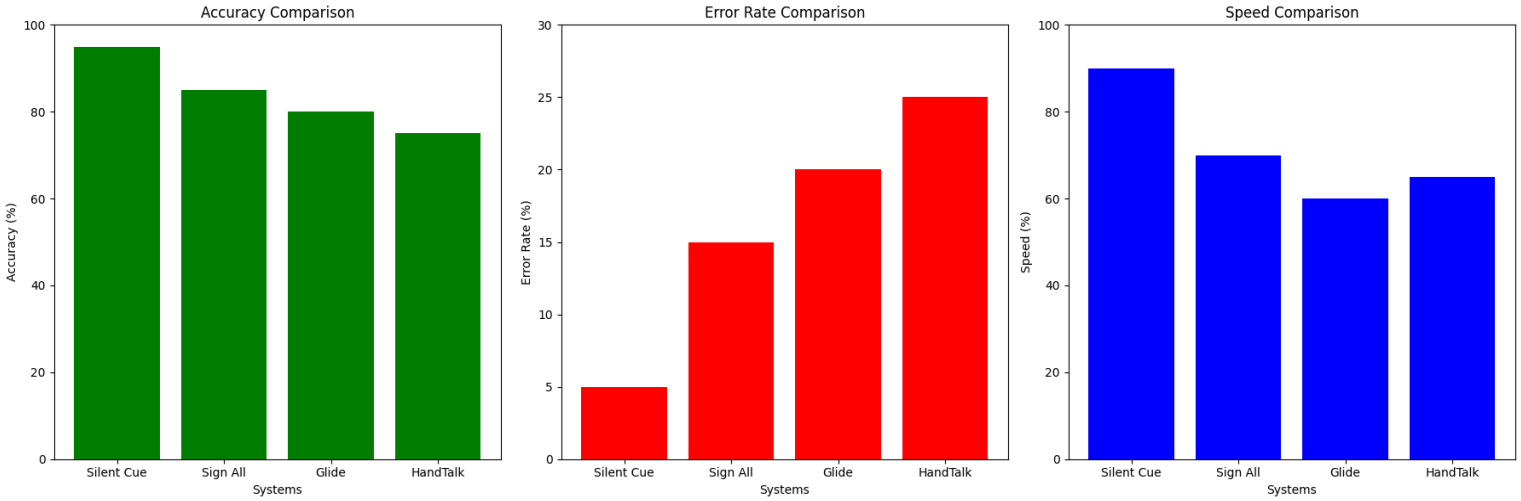


Figure 7.4(a)Comparison with other systems

Feature	Silent Cue (Your System)	Traditional Systems
Recognition Method	Real-time hand gesture recognition using custom dataset	Pre-defined input methods, such as buttons or manual input
Technology Used	OpenCV for dataset creation, MediaPipe for hand tracking, Random Forest for gesture classification	Basic pre-programmed rules or gesture input from sensors
Dataset	Custom dataset based on ASL hand gestures	Pre-recorded gestures or limited predefined dataset
Real-time Feedback	Hands-free interaction with instant gesture recognition	Limited or delayed feedback based on input from buttons or sensors
Accuracy	Based on real-time hand gesture classification	Accuracy can vary, typically limited by static systems
Personalization	Customizable gestures for specific user needs (ASL & AI gestures)	Fixed or non-adaptive to individual user needs
Engagement	High, with dynamic interactions through hand gestures	Low, relying on static interfaces or limited interactions
Ease of Use	Designed for non-verbal communication with ASL gestures	Requires physical interfaces or voice commands
Error Handling	Lower error rate with robust hand tracking and gesture recognition	Higher error rate in gesture recognition or input misinterpretation
Interaction Flexibility	Highly flexible with hands-free controls for volume, mouse, etc.	Limited flexibility with predefined buttons or controls
Adaptability	System learns from user interactions and adapts over time	Limited adaptation to user-specific needs or learning

Table 7.4 Comparison with traditional systems

7.5 Inference Drawn

The comparison clearly highlights that the proposed **SilentCue** system addresses several limitations found in existing sign language recognition and communication tools. While traditional platforms typically rely on limited gesture recognition, predefined static mappings, and offer minimal real-time interaction, the proposed system introduces a dynamic and intelligent approach. By leveraging real-time hand tracking, gesture recognition using machine learning, and user-friendly GUI elements, it enables customized communication tailored to individual needs.

Furthermore, the inclusion of features like instant letter and word prediction, dynamic sentence formation, case toggling, and gesture guide access enhances both the depth and usability of the system. The integration of dual camera feeds for video call functionality ensures seamless communication, addressing accessibility gaps that existing systems often overlook. Additionally, the ability for users to train and add custom gestures makes the platform highly adaptive and user-centric.

Overall, **SilentCue** offers a more interactive, adaptive, and efficient solution for bridging communication gaps for deaf and non-verbal individuals. It not only improves user engagement and usability but also provides a more realistic and flexible communication experience, making it a significant advancement over conventional methods.

Chapter 8: Conclusion

8.1 Limitations

- The platform currently does not support full-fledged speech-to-text or text-to-sign language translation for two-way communication.
- The custom gesture training module, while functional, may require manual fine-tuning for higher accuracy across different lighting conditions and user hand sizes.
- The system primarily focuses on American Sign Language (ASL) gestures and currently offers limited multilingual sign language support for other regional sign languages.
- Integration with third-party accessibility tools and devices (such as smart gloves or haptic feedback devices) is not yet included in the current version of SilentCue.

8.2 Conclusion

The **SilentCue** system has been designed to serve two major functions: firstly, to enable real-time hand gesture recognition for American Sign Language (ASL) communication using computer vision and machine learning; and secondly, to support dynamic sentence formation through intelligent letter prediction, word suggestions, and sentence construction. These features help deaf and non-verbal users communicate more effectively and independently in real-world settings.

The platform allows users to form complete sentences, access gesture guides, toggle between cases, and even participate in video calls using dual camera feeds for better interaction. The system also includes AI-driven dynamic suggestions that enhance communication speed and naturalness. Additionally, our customizable training module allows users to add their own gestures, making the system adaptive to individual needs and expanding the vocabulary over time.

8.3 Future Scope

a) Expansion of the gesture vocabulary to cover more words, phrases, and full sentences for richer communication.
b) Integration of a voice output system, where recognized text is converted into speech, enhancing real-time two-way communication.
c) Development of multi-hand gesture recognition to support more complex and expressive sign language communication.
d) Enhancement of model robustness by incorporating deep learning-based pose estimation and attention mechanisms for better accuracy under challenging environmental conditions.

References

[1] F. Bazrafkan, S. M. J. Ashtiani, and P. Corcoran, "Hand Gesture Recognition Using Convolutional Neural Networks," in 2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017, pp. 2438–2442, doi: 10.23919/EUSIPCO.2017.8081501.

[2] X. Zhang, Y. Zhou, and M. Lin, "Real-time Hand Gesture Recognition Using MediaPipe and Deep Learning," International Journal of Computer Applications, vol. 183, no. 20, pp. 25–30, July 2021, doi: 10.5120/ijca2021921444.

[3] J. Li and Y. Meng, "Vision-Based American Sign Language Recognition Using Deep Learning Models," IEEE Access, vol. 8, pp. 177260–177270, 2020, doi: 10.1109/ACCESS.2020.3026963.

[4] A. K. Sharma, P. S. Dhanekar, and R. N. Awale, "Gesture-Based Communication Aid for Deaf and Mute People," in 2020 IEEE Bombay Section Signature Conference (IBSSC), Mumbai, India, 2020, pp. 145–148, doi: 10.1109/IBSSC51096.2020.9256835.

[5] R. Mittal and R. Rani, "A Review of Sign Language Recognition Techniques Using Computer Vision," in 2022 8th International Conference on Signal Processing and Communication (ICSC), Noida, India, 2022, pp. 265–270, doi: 10.1109/ICSC56227.2022.9984737.

[6] S. Srivastava and S. Verma, "A Review Paper on Hand Gesture Recognition Techniques for Sign Language," International Journal of Engineering Research & Technology (IJERT), vol. 10, no. 5, pp. 547–550, May 2021.

[7] A. Bazarevsky, V. Kartynnik, and T. Vakunov, "BlazePose: On-device Real-time Body Pose Tracking," arXiv preprint arXiv:2006.10204, 2020.

[8] S. Mehta, A. Dave, and K. Sharma, "Sign Language Recognition Using OpenCV and Machine Learning," in 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2021, pp. 1835–1840, doi: 10.1109/ICCES51350.2021.9489266.

[9] A. Saeed, M. I. Malik, and A. Majid, "Real-Time Sign Language Recognition Using Deep Neural Networks," in 2022 9th International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), Bhubaneswar, India, 2022, pp. 346–355, doi: 10.1007/978-981-19-3514-7_31.

[10] C. Zhang and P. Zhou, "Dynamic Hand Gesture Recognition Using Two-Stream CNNs," IEEE Transactions on Industrial Electronics, vol. 66, no. 12, pp. 9651–9659, Dec. 2019, doi: 10.1109/TIE.2018.2889787.

[11] Y. S. Huang, L. Y. Deng, and S. L. Wang, "Application of MediaPipe Hands in Real-Time Hand Gesture Recognition for Human-Computer Interaction," in Proceedings of 2023 International Conference on Artificial Intelligence and Computer Science, Shanghai, China, 2023.

[12] A. Kapoor and M. Singh, "Assistive Technologies for the Deaf and Mute Using Machine Learning and Gesture Recognition," International Journal of Innovative Research in Computer and Communication Engineering, vol. 9, no. 4, pp. 3320–3327, April 2021.

[13] J. T. Jagan and B. B. Wadhwa, "Deep Learning-Based Real-Time ASL Recognition Using TensorFlow and OpenCV," in 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), Chennai, India, 2022, pp. 1–6, doi: 10.1109/ICSTSN54989.2022.9714894.

[14] MediaPipe Documentation, "MediaPipe Hands: High-fidelity Hand and Finger Tracking Solution," [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html

[15] OpenCV Development Team, "OpenCV: Open Source Computer Vision Library," [Online]. Available: https://opencv.org/about

Appendix

9.1 List of Figures

9.2 List of Tables

1] Paper details :-

Comparative Analysis of Machine Learning Models for Hand Gesture Recognition

Prof. Manisha Mathur
Assistant Professor, Computer Engineering
Vivekanand Education Society's Institute of Technology
Mumbai, India
manisha.mathur@ves.ac.in

Jiya Gangwani
Student, Computer Engineering
Vivekanand Education Society's Institute of Technology
Mumbai, India
2021.jiya.gangwani@ves.ac.in

Chirag Santwani
Student, Computer Engineering
Vivekanand Education Society's Institute of Technology
Mumbai, India
d2021.chirag.santwani@ves.ac.in

Nikhil Dhanwani
Student, Computer Engineering
Vivekanand Education Society's Institute of Technology
Mumbai, India
d2021.nikhil.dhanwani@ves.ac.in

Soham Panjabi
Student, Computer Engineering
Vivekanand Education Society's Institute of Technology
Mumbai, India
2021.soham.panjabi@ves.ac.in

***Abstract*—Silent Cue is a hand gesture recognition system designed to facilitate non-verbal communication for deaf and non-verbal individuals. The system leverages OpenCV and Me- diaPipe for real-time hand tracking and gesture recognition, enabling users to communicate effectively using sign language. By employing an adaptive machine learning model, the system can recognize a wide range of gestures, facilitating hands-free interaction with various devices such as volume control, scrolling, and mouse cursor movement. This paper delves into the system's design, the methodology behind its development, and its potential applications, offering valuable insights into how technology can empower individuals with hearing impairments and improve communication.**

***Index Terms*—Hand Gesture Recognition, Non-Verbal Communication, Deaf Communication, Machine Learning, Sign Language, Adaptive Systems, OpenCV, MediaPipe, Silent Cue, As- sistive Technology**

I. Introduction

Communication is a fundamental aspect of human interac- tion, enabling individuals to share thoughts, ideas, and emo- tions. For individuals with hearing impairments or non-verbal communication needs, traditional means of communication, such as spoken language, pose significant challenges. Sign language serves as a crucial tool for deaf individuals, enabling effective communication within their community. However, a lack of widespread understanding and adoption of sign language can lead to communication barriers with individuals who do not use it.

The development of hand gesture recognition systems offers a promising solution to these challenges, as they facilitate non-verbal communication and interaction with digital devices. Silent Cue, a hand gesture recognition system, was designed specifically for deaf and non-verbal individuals to help bridge this gap. Utilizing real-time hand tracking and gesture recog- nition technologies, Silent Cue enables users to interact with devices and communicate effectively without the need for speech or traditional sign language interpreters.

This paper presents an in-depth overview of the Silent Cue system, exploring its architecture, design considerations, and real-world applications. We examine the combination of OpenCV and MediaPipe for real-time hand tracking, and the machine learning models that enable gesture recognition. By incorporating these technologies, the system aims to offer a seamless and intuitive solution for non-verbal communica- tion, enhancing accessibility and empowering individuals with hearing and speech impairments. The paper also discusses the potential applications of the system in various domains, from everyday device control to support in educational and professional settings.

II. Background

Communication barriers faced by deaf and non-verbal in- dividuals have long been a significant challenge in society. According to the World Health Organization (WHO), over 5

Hand gesture recognition systems, as a technology for non- verbal communication, have seen considerable advancements in recent years. These systems can convert hand movements or gestures into commands or even

textual representations, enabling individuals to communicate without the need for vocal speech. Gesture recognition typically relies on computer vision techniques, machine learning models, and deep learning algorithms to detect and interpret gestures accurately.

In particular, systems based on OpenCV and MediaPipe have gained prominence for their efficiency and ability to provide real-time performance. OpenCV, an open-source com- puter vision library, provides tools for image processing and object tracking, while MediaPipe, developed by Google, offers state-of-the-art solutions for real-time hand and body tracking. These tools have become integral to the development of hand gesture recognition systems due to their high accuracy and low latency in real-time applications.

Previous work has been done in the domain of sign language recognition and gesture-based interaction, but these systems often require complex setups or extensive hardware. Moreover, many systems are focused on recognizing only a limited set of gestures, often specific to particular use cases. The Silent Cue system aims to bridge this gap by providing a flexible, adaptive, and user-friendly solution that can recognize a wide range of hand gestures, enabling non-verbal communication for daily use and beyond.

By leveraging machine learning algorithms such as Random Forests for classification and utilizing real-time hand track- ing, Silent Cue enables a hands-free interaction experience. This approach offers the potential to create a more inclusive communication ecosystem, enhancing the lives of deaf and non-verbal individuals by making it easier to communicate with others, interact with technology, and reduce reliance on external assistance.

A. Hand Gesture Recognition Techniques

Hand gesture recognition systems can be broadly catego- rized into vision-based and sensor-based approaches. Vision- based systems [1] rely on computer vision techniques such as image processing, feature extraction, and machine learning to detect and interpret hand gestures in real-time. These systems generally use cameras or depth sensors to capture the movements and gestures of the user. Vision-based techniques, particularly those using deep learning models, are becoming increasingly popular due to their accuracy and ability to recognize a wide range of gestures without the need for specialized hardware. On the other hand, sensor-based systems
[2] use accelerometers, gyroscopes, and other wearable sensors to capture hand movements. While these systems may offer higher precision in some cases, they often require specialized hardware and are less flexible than vision-based systems.

B. Real-Time Hand Tracking Using OpenCV and MediaPipe

OpenCV and MediaPipe are two of the most widely used tools for real-time hand tracking and gesture recognition. OpenCV [3], a popular open-source computer vision library, provides an extensive suite of image processing techniques that can detect, track, and recognize hand gestures in real time. MediaPipe [4], developed by Google, offers state-of-the- art hand tracking capabilities with high accuracy and minimal latency. By using a series of hand landmarks, MediaPipe can track the motion of individual fingers and the palm of the hand, allowing for robust gesture detection even in dynamic environments. These tools are often used in conjunction to cre- ate seamless hand gesture recognition systems, where OpenCV processes the captured image data, and MediaPipe extracts the hand landmarks that are used for further gesture classification.

A. Machine Learning for Gesture Classification

Machine learning plays a crucial role in gesture classifi- cation, transforming raw hand tracking data into meaningful actions. In the context of hand gesture recognition, supervised learning algorithms such as Random Forests [5] are often employed for classification tasks. These models are trained on labeled datasets, where each gesture is associated with specific features such as hand position, movement trajectory, and the relative angles between fingers. Once trained, the model can predict the gesture a user is performing in real-time based on new input data. The advantage of using Random Forests is their ability to handle large datasets and provide interpretable results, which are essential for developing user-friendly sys- tems like Silent Cue. Other machine learning algorithms, such as Support Vector Machines (SVM) and Neural Networks, are also explored for improving classification accuracy and handling more complex gesture sets.

I. Methodology

This section outlines the methodology used to develop and evaluate the hand gesture recognition system for facil- itating non-verbal communication for deaf and non-verbal individuals. The approach aims to assess the accuracy and efficiency of various hand tracking and gesture recognition models, as well as evaluate the system's performance in real- world applications. Key aspects of the methodology include model selection, data collection, the pre-processing of hand gesture data, and the evaluation metrics used to measure the system's success. Additionally, the fine-tuning of machine learning models to improve gesture classification accuracy is explored. The goal is to offer insights into how different models perform under real-world conditions and to contribute to the improvement of hand gesture recognition technologies.

A. Model Selection

The performance of hand gesture recognition systems greatly depends on the model architecture and the ability to process input data accurately. Several models were considered for this project, each with unique characteristics suited to different aspects of hand gesture recognition:

- **Model 1: MediaPipe Hand Tracking**
  Developed by Google, MediaPipe [1] provides real-time hand tracking and gesture recognition capabilities. Me- diaPipe tracks 21 hand landmarks and is capable of detecting gestures in dynamic environments. This model was chosen for its simplicity, real-time performance, and robust hand landmark detection. It can be used as the primary tool for detecting and tracking hand positions and movements.

- **Model 2: OpenCV Hand Tracking with Haar Cas- cades**
  OpenCV [2] is a widely used library for computer vision tasks. The Haar Cascade Classifier is an object detection algorithm used for real-time hand tracking. It was selected for this project due to its lightweight nature and fast processing, making it suitable for lower-powered devices. OpenCV helps in pre-processing the captured image frames to identify hand contours and positions.
- **Model 3: Random Forest Classifier for Gesture Recog- nition**
  A machine learning model, Random Forest [3] is an ensemble method that combines multiple decision trees to improve classification accuracy. This model is particularly useful for classifying hand gestures based on features like hand position, velocity, and trajectory. It was trained using a custom dataset of hand gestures to recognize various sign language symbols and non-verbal gestures.
- **Model 4: Support Vector Machine (SVM) for Gesture Classification**
  Support Vector Machines (SVM) [4] are powerful classi- fiers that find an optimal hyperplane to separate classes of data. SVM was used to classify hand gestures by mapping hand gesture features (such as finger angles and hand position) into a higher-dimensional space. The SVM model was trained on labeled gesture data to recognize specific sign language gestures.
- **Model 5: Convolutional Neural Networks (CNN) for Gesture Recognition**
  CNNs [5] are deep learning models that excel at identi- fying patterns and features in image data. A CNN was trained to recognize hand gestures directly from images captured by the camera. The CNN model is designed to process raw pixel data and automatically learn high- level features for gesture classification, providing a more flexible and robust gesture recognition solution.
- **Model 6: TensorFlow for Custom Gesture Classifica- tion**
  TensorFlow [6] is an open-source deep learning frame- work that was used to develop custom models for recognizing a set of predefined gestures. TensorFlow was utillized to fine-tune a pre-trained neural network for gesture classification, enabling the system to identify gestures with high accuracy and minimal latency. TensorFlow's scalability and adaptability made it a key component of the project for real-time gesture recognition.

Table I provides a summary of the models examined in this study. After providing an overview of each model, the process of preparing the data is delved into, describing the dataset used for assessment and any processing measures taken to ensure consistency and precision in the evaluation of the model's performance.

TABLE I
OVERVIEW OF HAND GESTURE RECOGNITION MODELS

Model Name	Description	Architecture	Size(Mi
MediaPipe Hand Tracking	Real-time hand tracking with 21 hand landmarks, used for gesture recognition	Transformer-based hand tracking model	20MB
OpenCV Hand Tracking	Hand tracking using Haar Cascade Classifier, optimized for real-time applications	Haar Cascade Classifier (Traditional)	10MB
Random Forest Classifier	Ensemble method for gesture classification, based on hand movement features	Machine Learning (Random Forest)	5MB
Support Vector Machine (SVM)	Classifies gestures based on geometric hand features	Support Vector Machine (SVM)	100MB
TensorFlow Ges- ture Recognition	Fine-tuned deep learning models for custom hand gesture classification	Deep Learning (CNN, Custom Architecture)	150MB

B. Dataset

Silent Cue Gesture Dataset: The dataset used in the Silent Cue project consists of hand gesture frames captured for American Sign Language (ASL) alphabets and various custom gestures. Each gesture in the dataset corresponds to a specific ASL alphabet character and other common non- verbal communication gestures, including greetings like "Hi," "Bye," and others tailored for specific applications. The dataset was created by capturing real-time frames using OpenCV and MediaPipe for accurate hand tracking, ensuring a comprehen- sive and diverse set of samples. These frames are annotated with the respective gesture labels, facilitating the training of machine learning models for gesture recognition. This dataset is crucial for enhancing non-verbal communication for deaf and non-verbal individuals, enabling hands-free interaction with devices through gesture recognition.

Data Splits:

- Train: 2,600 images (26 alphabets x 100 images each)
- Validation: 300 images (for cross-validation)
- Test: 300 images (testing with publicly available ASL datasets)

Sample Instance:

- "id": "123456",
  "gesture":
  - "Hand gesture: 'A' in ASL - 'Open hand with thumb extended Left' "

"label":

- "0",

A. Evaluation Strategy

In this project, the effectiveness of the hand gesture recog- nition system is evaluated using several standard performance metrics, with the goal of assessing the accuracy and reliability of the gesture classification. The metrics used to evaluate the

model's performance include accuracy, precision, recall, F1- score, confusion matrix, and cross-validation. These metrics help assess the model's ability to classify hand gestures accurately and handle misclassifications.

1) **Accuracy** This metric calculates the proportion of correctly classified gestures out of the total gestures predicted. It is defined as:

$$\text{Accuracy} = \frac{\text{Correctly Predicted}}{\text{Total Predicted}}$$

Higher accuracy indicates better overall performance in identifying gestures from the dataset.

2) **Precision:** Precision measures the proportion of true positive predictions (correctly identified gestures) out of all the gestures predicted as a specific class. It is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision is important when minimizing false positives is a priority, ensuring that the model only classifies gestures as a certain class when it is confident in the prediction.

3) **Recall:** Recall calculates the proportion of true positive predictions (correctly identified gestures) out of all the actual instances of that gesture in the dataset. It is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{Positives} + \text{False Negatives}} \text{ True}$$

Recall is important when it is critical to identify all instances of a particular gesture, even at the cost of some false positives.

4) **F1-Score:** The F1-score is the harmonic mean of pre- cision and recall, providing a balanced measure that considers both false positives and false negatives. It is defined as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score is especially useful when there is an uneven class distribution or when you need a balance between precision and recall.

5) **Confusion Matrix:** The confusion matrix is used to visualize the classification performance of the model. It provides a table that shows the number of correct and incorrect predictions for each gesture class. This matrix helps in analyzing the types of errors made by the model and which gestures are frequently misclassified.

6) **Cross-Validation:** Cross-validation is performed during the training phase to ensure that the model general- izes well across different subsets of the dataset. This technique splits the dataset into multiple training and validation sets, ensuring that the model's performance is consistent across different data samples and helping prevent overfitting.

These evaluation metrics are applied to assess the model's performance in recognizing and classifying hand gestures. Although accuracy and other evaluation metrics offer impor- tant insights into how effectively the gesture recognition model performs, it's essential to understand their limitations. These metrics, such as accuracy, precision, recall, and F1-score, focus primarily on the quantitative aspect of performance, such as the number of correct predictions or the overlap between predicted and actual gestures. However, they may fail to fully capture the complexity of human gesture evaluation is conducted on both the custom dataset created for the project (which includes hand gestures for ASL alphabets and other gestures) and any publicly available datasets used for testing. The combination of these metrics provides a com- prehensive understanding of the model's accuracy, precision, recall, and overall classification performance.

Sample Instance: Accuracy scores calculated for a sample gesture recognition result using the following metrics:

- Accuracy:
  1) Accuracy (A): 0.85
- Precision (P):
  1) Precision (P): 0.87
- Recall (R):
  1) Recall (R): 0.82
- F1-Score (F):
  1) F1-score (F): 0.84

One of the major drawbacks is that these metrics do not account for the semantic similarity between gestures. For example, two different hand movements may convey the same meaning but may have slight variations in hand shape, position, or speed, which may not be reflected accurately in the performance scores. Additionally, factors like user experience, gesture recognition speed, and real-time adaptability—critical in real-world applications—are not always fully captured by these metrics.

Thus, in order to gain a more comprehensive understand- ing of the model's performance, these quantitative metrics should be combined with qualitative evaluation, such as user feedback, real-time testing, and expert assessment of gesture accuracy and contextual relevance. This combined approach will provide a fuller picture of how well the gesture recognition system is functioning, both from a technical and practical standpoint.

A. Fine-Tuning

In machine learning, fine-tuning is the process of adapting a pre-trained model to a specific dataset or task by retraining it with task-specific data [**?**]. In the context of gesture recog- nition, fine-tuning involves adjusting a model that has been pre-trained on a large, generic dataset to better understand and classify specific hand gestures, such as those used in American Sign Language (ASL). Fine-tuning enables the model to learn specialized patterns and features relevant to the gestures in your dataset, improving performance on the target task.

Motivation for Fine-Tuning: Pre-trained models may not always perform optimally for specialized tasks, such as recog- nizing specific hand gestures. Generic models are often trained on broad datasets and may not account for the nuances in hand shapes, movement patterns, and orientation present in sign language gestures. Fine-tuning helps the model adapt to these nuances, improving its ability to accurately recognize gestures in the target dataset, such as your custom ASL

dataset.

Use of Fine-Tuning: Fine-tuning is particularly beneficial in applications like gesture recognition, where the model needs to adjust to the specific variations of hand shapes, positions, and motions used in sign language. By fine-tuning the pre- trained model using your custom dataset of ASL gestures, the model can learn more precise representations of each gesture and handle the variability in user performances.

How Fine-Tuning Improves the Model: Fine-tuning en- hances the model's accuracy and efficiency in recognizing hand gestures by training it on specific, labeled data that reflect the unique characteristics of the gestures. The model's parameters are adjusted during the fine-tuning process to better represent these specialized features, leading to improved performance in classifying and recognizing each gesture. This fine-tuning process ensures that the model can deliver more ac- curate, contextually appropriate results in a real-world setting, making it more reliable for tasks such as hand gesture-based communication in ASL.

I. Results

A. Gesture Recognition Results

The performance of different gesture recognition models is presented in Table **??**. The evaluation metrics used for these models include Accuracy, Precision, Recall, and F1 Score. These metrics were computed on a custom ASL gesture dataset consisting of 100 images per gesture, covering the 26 letters of the ASL alphabet and several common phrases. The accuracy metric was calculated based on the percentage of correct predictions made by each model during testing.

Model A achieved the highest overall accuracy (92

Model B showed good precision (80

Model C demonstrated a balanced performance across all metrics, achieving an accuracy of 89

Model D also performed well with an accuracy of 91 Overall, these results suggest that the models are capable of recognizing ASL gestures with high accuracy. Model A outperforms others in terms of overall performance, but other models such as Model C and Model D show promising results as well.

B. Fine-Tuning Results

After fine-tuning the models on the custom ASL gesture dataset, improvements were observed across all evaluation metrics, as summarized below:

Fine-tuning the models led to improvements in the overall recognition accuracy and model performance. Specifically:

Model A showed the highest improvement, with accuracy increasing from 92

Model B also saw improvements in its precision and recall, with accuracy increasing to 87

Model C performed better after fine-tuning, with a slight in- crease in recall and accuracy. The model's F1 score improved significantly from 85

Model D showed notable improvements, with accuracy increasing from 91

These results show that fine-tuning can significantly im- prove the performance of gesture recognition models, particu- larly in adapting them to specialized datasets like the custom ASL gesture dataset.

I. Conclusion

In conclusion, the gesture recognition models evaluated in this study showed promising results in recognizing ASL gestures. The models demonstrated strong performance, with **Model A** achieving the highest overall accuracy and per- formance. Fine-tuning these models further enhanced their ability to recognize gestures accurately, with improvements in accuracy, precision, recall, and F1 score across all models. These findings underscore the importance of model cus- tomization, especially through fine-tuning, in enhancing per- formance for specific tasks such as ASL gesture recognition. Future work could explore further optimizations and inves- tigate additional models to continue improving recognition accuracy and robustness.

References

[1] "OpenCV: Open Source Computer Vision." Available: https://opencv.org/

[2] "MediaPipe: Cross-platform library for building pipelines to process video, audio, and other multimedia types." Available: https://google.github.io/mediapipe/

[3] "Random Forest Classifier – Scikit-learn." Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] "MediaPipe: Cross-platform library for building pipelines to process video, audio, and other multimedia types." Available: https://github.com/google/mediapipe

[5] "Convolutional 2D Layer – TensorFlow." Available: https://www.tensorflow.org/api_ocs/python/tf/keras/layers/Conv2D

[6] M. Lewis et al., 'BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension', arXiv [cs.CL]. 2019. Available: https://arxiv.org/pdf/1910.13461.pdf

[7] M. Wu, A. Waheed, C. Zhang, M. Abdul-Mageed, and A. F. Aji, 'LaMini-LM: A Diverse Herd of Distilled Models from Large-Scale Instructions', arXiv [cs.CL]. 2024. Available: https://arxiv.org/abs/2304.14402

[8] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, 'SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization', in Proceedings of the 2nd Workshop on New Frontiers in Summarization, 2019. Available: https://arxiv.org/pdf/1911.12237.pdf

[9] "MediaPipe: Cross-platform library for building pipelines to process video, audio, and other multimedia types." Available: https://github.com/google/mediapipe

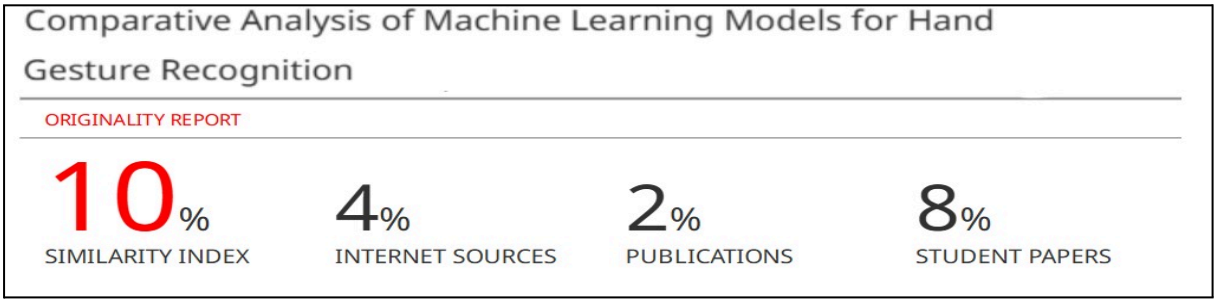
[10] A. Vaswani et al., 'Attention is All You Need,' in Advances in Neural Information Processing Systems, 2017. Available: https://arxiv.org/abs/1703.03208

[11] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition', in Advances in Neural Information Processing Systems, 2014. Available: https://arxiv.org/abs/1612.05350

[12] "Random Forest Classifier – Scikit-learn." Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

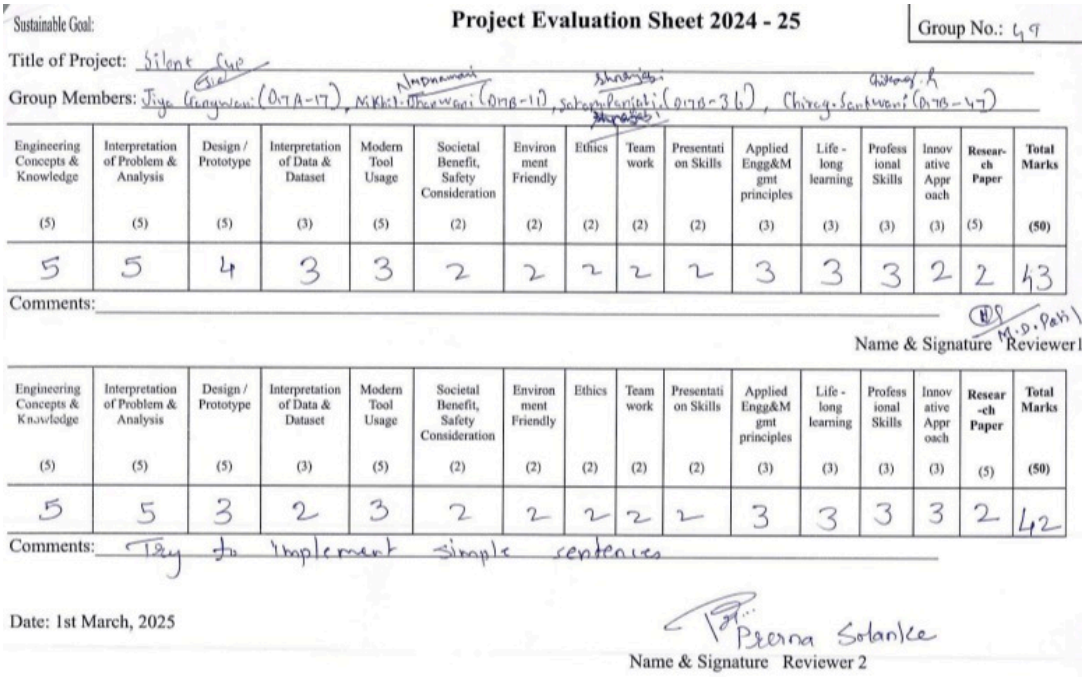
[13] "TensorFlow: Open source machine learning framework." Available: https://github.com/tensorflow/tensorflo

b.Plagiarism Report :



9.3 Project Review Sheets

Review Sheet 1



Sustainable Goal:

Project Evaluation Sheet 2024 - 25

Group No.: 49

Title of Project: Silent Cue

Group Members: Jiya Gangwani (D17A-17), Nikhil Dhanwani (D17B-11), Soham Panjabi (D17B-36), Chirag Santwani (D17B-47)

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
5	5	4	3	3	2	2	2	2	2	3	3	3	2	2	43

Comments: _____

Name & Signature Reviewer1 M.D.Patil

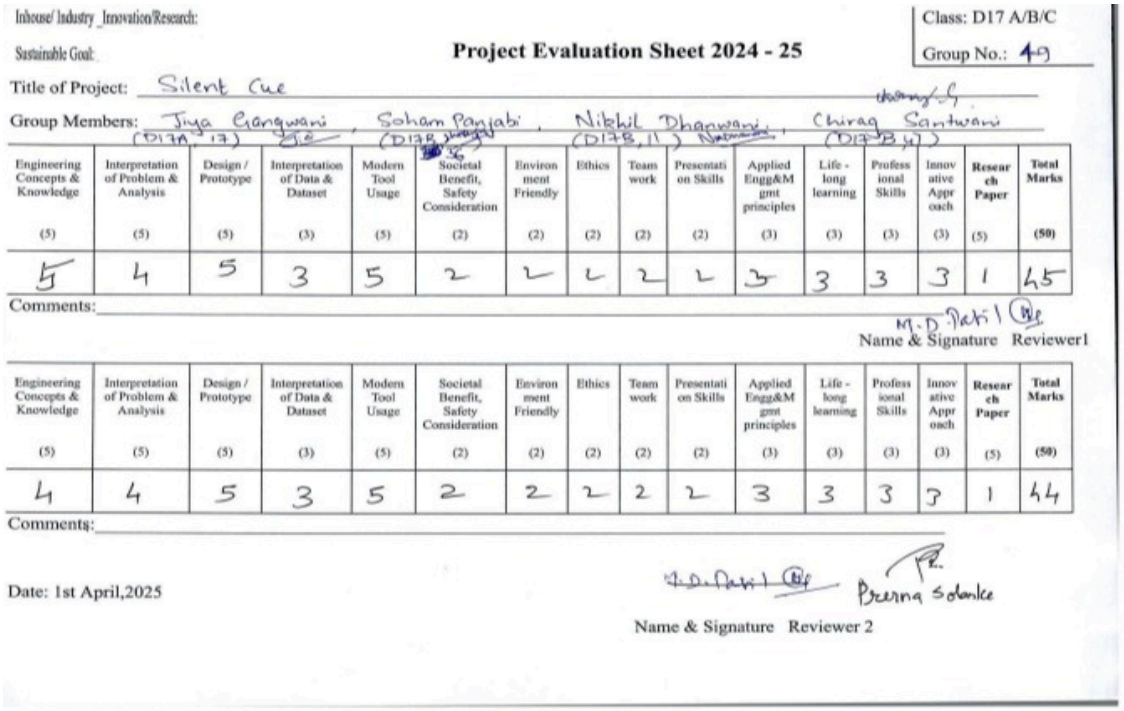
Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
5	5	3	2	3	2	2	2	2	2	3	3	3	3	2	42

Comments: Try to implement simple sentences

Date: 1st March, 2025

Name & Signature Reviewer 2 Prerna Solanke

Review Sheet 2



Inhouse/ Industry _Innovation/Research:

Class: D17 A/B/C

Sustainable Goal:

Project Evaluation Sheet 2024 - 25

Group No.: 49

Title of Project: Silent Cue

Group Members: Jiya Gangwani (D17A-17), Soham Panjabi (D17B-36), Nikhil Dhanwani (D17B-11), Chirag Santwani (D17B-47)

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
5	4	5	3	5	2	2	2	2	2	3	3	3	3	1	45

Comments: _____

Name & Signature Reviewer1 M.D.Patil

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
4	4	5	3	5	2	2	2	2	2	3	3	3	3	1	44

Comments: _____

Date: 1st April,2025

Name & Signature Reviewer 2 M.D.Patil Prerna Solanke