Sign Language Translation for Deaf

A PROJECT REPORT

Submitted by

Mahir Chaudhari (22BECE30037)

Mohsinali Datardi (22BECE30075)

Krutik Gami (22BECE30093)

Kumar Parva (22BECE30153)

In fulfillment for the award of the degree

of

COMPUTER ENGINEERING

LDRP Institute Of Technology and Research,Gandhinagar

Kadi Sarva Vishwavidyalaya

Kadi Sarva Vishwavidyalaya

Jan-Feb, 2025

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CERTIFICATE

This is to certify that the Project work entitled "**Sign Language Translation for Deaf**" has been carried out by **<u>Mohsinali Datardi</u> (22BECE30075)** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering Semester-6 of Kadi Sarva Vishwavidyalaya University during the academic year 2025-26.

Prof. Tejasvee Gupta	**Prof. Ashishkumar Patel**
Internal Guide	**Head of Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CERTIFICATE

This is to certify that the Project work entitled "**Sign Language Translation for Deaf**" has been carried out by **Mahir Chaudhari (22BECE30037)** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering Semester-6 of Kadi Sarva Vishwavidhyalaya University during the academic year 2025-26.

Prof. Tejasvee Gupta	**Prof. Ashishkumar Patel**
Internal Guide	**Head of Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH

GANDHINAGAR

CERTIFICATE

This is to certify that the Project work entitled "**Sign Language Translation for Deaf**" has been carried out by **Krutik Gami (22BECE30093)** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering Semester- 6 of Kadi Sarva Vishwavidyalaya University during the academic year 2025-26.

Prof. Tejasvee Gupta	**Prof. Ashishkumar Patel**
Internal Guide	**Head of Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CERTIFICATE

This is to certify that the Project work entitled "**Sign Language Translation for Deaf**" has been carried out by Kumar Parva **(22BECE30153)** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering Semester-6 of Kadi Sarva Vishwavidyalaya University during the academic year 2025-26.

Prof. Tejasvee Gupta	**Prof. Ashishkumar Patel**
Internal Guide	**Head of Department**
LDRP ITR	**LDRP ITR**

Presentation-I for Project-I

1. Name & Signature of Internal Guide	
2. Comments from Panel Members	
3. Name & Signature of Panel Members	

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who have contributed to the successful completion of our Sign Language Translation for Deaf project. Without their support, guidance, and encouragement, this project would not have been possible.

First of all we are thankful to the LDRP Institute of Technology and Research for undertaking this project. We would like to extend our deepest appreciation to Prof. Tejasvee Gupta, our project supervisor, for providing us with invaluable guidance throughout the project. Your expertise, patience, and continuous support were instrumental in shaping our project and enhancing our understanding of Sign Language Translation for Deaf.

We would like to express our gratitude to our classmates and friends who provided us with moral support and valuable suggestions during the course of this project.

Mahir Chaudhari (22BECE30037)

Mohsinali Datardi (22BECE30075)

Krutik Gami (22BECE30093)

Kumar Parva (22BECE30153)

ABSTRACT

The creative project "Sign Language Translation for Deaf " aims to create a hand gesture detection system based on AI/ML that will help people with speech impairments communicate more effectively. This technology converts hand motions into text by using cutting-edge computer vision and deep learning algorithms, giving people who are mute a smooth means of communication.

In order to guarantee excellent accuracy and dependability, the project entails building a sturdy convolutional neural network (CNN) model trained on a variety of hand gesture datasets. The system includes an easy-to-use OpenCV.js-integrated real-time gesture recognition, and a Flask-powered Flask backend for data administration and model inference.

Real-time gesture detection, text display of motions that have been recognized, an easy-to-use user interface, and a training module for adding new gestures are some of the key features. To further enhance the user experience and system performance, future enhancements will include text-to-speech conversion, support for more sophisticated gestures, and continuous learning capabilities.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1 INTRODUCTION

- INTRODUCTION
- AIMS AND OBJECTIVE OF THE WORK
- BRIEF LITERATURE REVIEW
- PROBLEM DEFINITION
- PLAN OF THEIR WORK

1.1 Introduction

A vital component of human contact, communication is essential for conveying needs, wants, and feelings. Using traditional ways of communication might be difficult for people who are speech challenged. The " Sign Language Translation for Deaf " project intends to create an AI/ML-based hand gesture recognition system that converts hand gestures into text, providing a useful and effective solution for individuals who are mute, in an effort to close this communication gap. Using deep learning and computer vision, the "Sign Language Translation for Deaf " system can recognize and comprehend a large variety of hand signals in real-time. Through the use of a convolutional neural network (CNN) trained on a wide range of hand gesture datasets, the system guarantees a high degree of accuracy and consistency in gesture recognition. After the gestures are identified, text is generated. To summarize, " Sign Language Translation for Deaf" is a noteworthy progression in assistive technology, providing a scalable and efficient means of improving communication for those with speech impairments. This research demonstrates how AI and machine learning may be used to develop inclusive technologies that give people more power and enhance their quality of life.

1.2 Aims and objective of the work

Build a Reliable Model for Gesture Recognition:

- To train the model, gather and preprocess a variety of hand gesture datasets.
- Convolutional neural networks (CNNs) are being developed and trained with TensorFlow/Keras to identify a wide range of hand movements with accuracy.
- To improve model performance, apply techniques like hyperparameter tuning and data augmentation.

Facilitate Efficient Conversation:

- Convert recognised hand movements into real-time text that appears on a screen.
- Make sure there is low latency and high precision in the system to allow efficient and seamless communication.

Conduct Comprehensive Testing:

- To make sure the system is reliable and functional, thoroughly evaluate its units and integrations.

Develop a plan for future improvements:

- Examine whether more complex movements and gesture combinations can be supported.
- Create a voice synthesis module to enable speech recognition of text.
- Apply machine learning techniques to learn and improve continuously depending on feedback from users.

1.3 Brief Literature Review

Systems for recognising hand gestures based on AI/ML are becoming more and more important for improving communication among those with speech impairments. Real-time hand gesture interpretation has become much more capable thanks to developments in computer vision and machine learning, which allow hand gestures to be translated into text or speech. This development demonstrates how these technologies have the potential to greatly improve speech-impaired individuals' accessibility and communication abilities.

Despite these developments, a number of current systems still have issues with hardware dependencies and the requirement for regulated environments, which might limit their accessibility and usability. Furthermore, the effectiveness and general user happiness of the current solutions may be impacted by their lack of adequate customisation and flexibility possibilities.

In order to overcome these obstacles, the " Sign Language Translation for Deaf " project offers a feature-rich gesture detection system that is easy to use and accessible. " Sign Language Translation for Deaf " makes technology more adaptable and user-friendly in a variety of contexts by doing away with the need for specialized hardware by utilizing a browser-based approach with OpenCV.js.

"Sign Language Translation for Deaf " is a noteworthy development in gesture detection technology. It offers a more flexible and user-centric tool for communication among speech-impaired people, positioning it as a considerable improvement over current alternatives due to its novel approach to accessibility, customisation, and user involvement.

1.4 Problem Definition

A basic human need is to be able to communicate effectively, although people who have speech problems frequently struggle to express themselves. While helpful, traditional communication aids like text-to-speech software or sign language interpreters might not completely meet the demand for accessible, user-friendly, real-time communication solutions. For people who use sign language in particular, hand gestures provide a natural way to communicate. However, because of variations in user-specific characteristics, background conditions, and gesture performance variability, real-time recognition and translation of these gestures into text or speech remains a challenging task. Utilizing deep learning and computer vision, the "Sign Language Translation for Deaf " system can detect and interpret a large variety of hand signals in real-time. Through the use of a convolutional neural network (CNN) trained on a wide range of hand gesture datasets, the system guarantees a high degree of accuracy and consistency in gesture recognition. After the movements are identified, text is generated and shown on a screen to facilitate smooth conversation.

1.5 Plan of their work

1. Establishing the Project and Initial Planning:

   Project Initialization:

   - Configure the environment,tools,and project repository.

   Requirement analysis:

   - Specify the system's needs and requirements in detail, including the gestures that the system must recognize.

   Data Collection:

   - Locate and compile hand gesture datasets that are pertinent to the project, including both static and dynamic gestures.

2. Collecting and Processing Data:

   Data Acquisition:

   - Gather and aggregate video and picture data for arrange of hand movements.

Data Preprocessing:

- To get the data ready for model training, clean it up and perform preprocessing operations including normalization, augmentation, and labeling.

3. Development of the Model:

Model Selection:

- Based on the requirements for gesture recognition,select suitable deeplearning architectures, such as Convolutional Neural Networks (CNNs) or 3D CNNs.

Training:

- Using TensorFlow/Keras, create and train the selected model. To increase accuracy, use approaches like data augmentation and hyperparameter adjustment.

Evaluation:

- Using validation datasets, evaluate the model's performance and make necessary adjustments based on accuracy and reliability measures.

4. Analyzing and Improving:

Unit and Integration Testing:

- Toverifysystemfunctionality,thoroughlytesteachindividualcomponentand how they work together.

User Acceptance Testing:

- Assess the system with people who have speech him pairments inorder to get input and make any required modifications.

5. Training and Documentation:

Documentation:

- Create training materials ,user guides,and technical documentation.

6. Training Sessions:

- Tointroduceendusersandstakeholderswiththesystem,arrangetraining.

7. Future Improvements (Continued):

Extensions of Gestures:

- Provideassistanceforincreasinglyintricatemovementsandcombinations.

Voice Analysis:

- To further improve communication possibilities, consider adding a text-to-speech module.

Continuous Improvement:

- Apply machine learning strategies for ongoing education and system enhancement in response to user input.

This strategy ensures that the "Sign Language Translation for Deaf " hand gesture detection system is developed, tested, and implemented in an organized manner, effectively meeting the needs of those with speech impairments.

2 TECHNOLOGY AND LITERATURE REVIEW

- TOOLS AND TECHNOLOGY
- PROJECT SCHEDULING

2.1 Tools and Technology

Streamlit:

- An open-source framework for creating dynamic web apps related to machine learning and data science. It makes it easier to deploy machine learning models by enabling you to write little code to construct dashboards and user interfaces.

Frameworks for Model Training:

- TensorFlow/Keras:
  Popular deep learning frameworks for model construction and training. While Keras gives a high-level API to facilitate the design and experimentation of models more easily, TensorFlow offers a rich environment for building machine learning models.
- PyTorch: An alternative to TensorFlow that is well-liked for its adaptability and simplicity of usage, particularly in research environments, is PyTorch. Dynamic computation graphs are made possible by PyTorch, which is useful for testing and training models.

Tools for Collaboration and Development:

- Jupyter Notebook:
  Jupyter Notebooks are useful for developing and testing models prior to Stream-lit deployment, albeit they are mostly utilized for exploratory data analysis and model training.
- Git/GitHub: To facilitate collaboration and version control, make sure that model code and associated scripts are properly maintained and distributed among team members.

Preprocessing and Data Handling:

- Pandas: An effective library for data analysis and manipulation that may be used to prepare data before training models.
- NumPy: Alibrarythatsupportsmassivemulti-dimensionalarraysandmatricesfor numerical computations

2.2 Project Scheduling

Project scheduling involves separating the total work in a project into separate activities and judging the time required to complete these activities. Usually, some of these activities are carried out in parallel.

I. Work Breakdown Structure

- Work Breakdown Structure is used to decompose a given task set recursively into small activity [Fig 2.I].

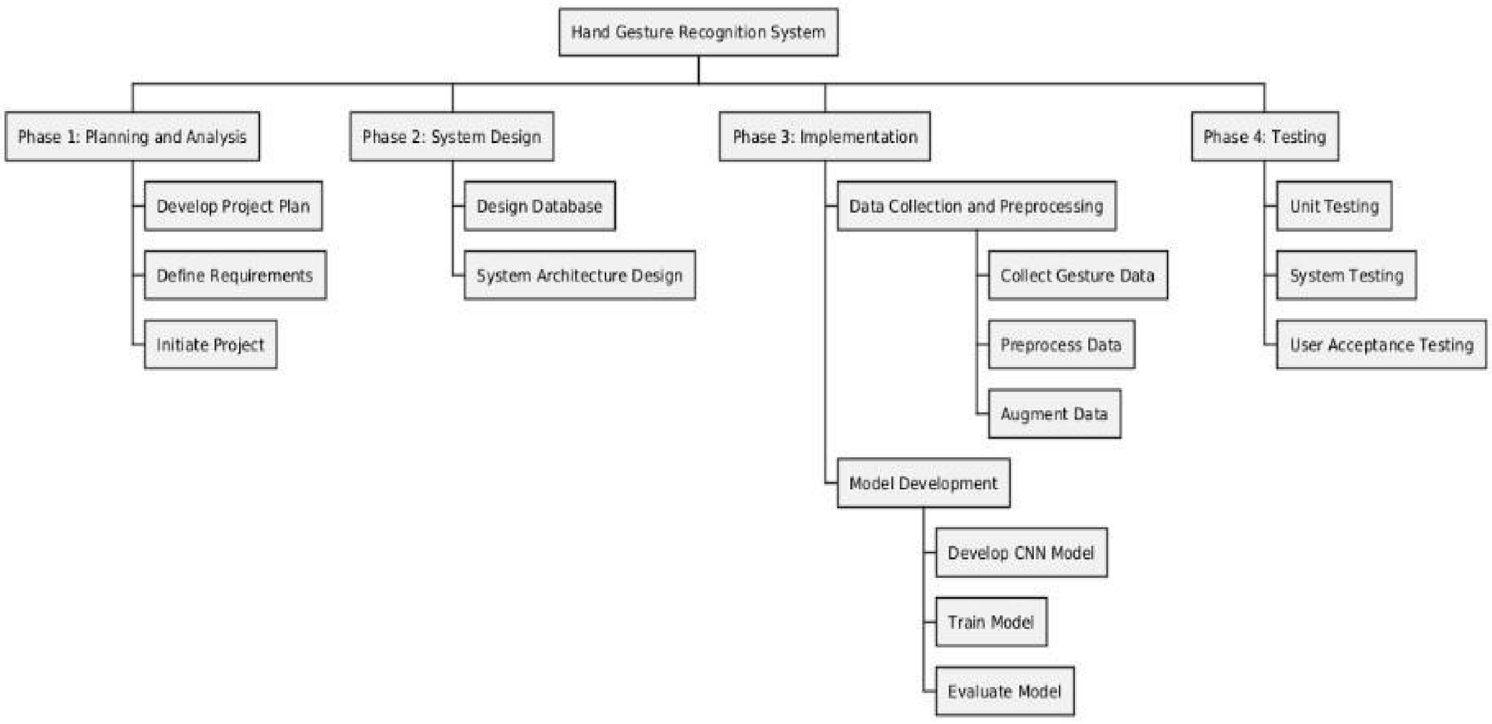

Fig 2.I - Work Breakdown Structure

3 SYSTEM REQUIREMENTS STUDY

- USER CHARACTERISTICS
- HARDWAREANDSOFTWAREREQUIREMENTS
- ASSUMPTIONS AND DEPENDENCIES

3.1 User Characteristics

Primary Users:

1. Speech-Impaired Individuals:
   - The main target audience for "Sign Language Translation for Deaf " is those with speech problems who primarily communicate through hand gestures or sign language. They might prefer different communication styles and have differing degrees of technological familiarity.
2. Technical Proficiency:
   - Varied Technical Skills:
     The system should be built with an intuitive interface to support users of varying technical proficiency, from those who are familiar with digital tools and applications to those who have little to no experience with technology.
3. Communication Needs:
   - Diverse Communication Preferences: Depending on their particular disability, users may require different forms of communication. To make sure it satisfies the needs of a wide audience, the system should include a variety of hand gestures and sign language.
   - Real-Time Communication: For efficient and seamless communication, users need to be able to translate gestures into words in real-time.
4. Physical Abilities:
   - Hand Dexterity:
     Users will have varied levels of hand dexterity and may employ different styles of hand gestures. A vast variety of gestures, including changes in speed and accuracy, must be recognised by the system.
   - Mobility Variations: It should be possible for some users to perform motions only because of physical limits; the system should be able to accommodate a wide range of movements and gestures.
5. Accessibility Considerations:
   - Visual and Auditory Needs: It should be possible for users with visual or hearing impairments to interact with the system. For instance, feedback mechanisms should be available and text on the screen should have movable font and contrast.

6. Environmental Context:

- Diverse Environments:
  Users will use the system in various environments, including homes, workplaces, and public spaces. The system should be robust enough to handle different lighting conditions and backgrounds.
- Device Compatibility: The application should be compatible with various devices, including smartphones, tablets, and computers, to ensure accessibility across different platforms.

3.2 Software and Hardware Requirements

Software and Hardware Requirements are used to describe the minimum hardware and software requirements to run the Software. These requirements are described below.

3.2.1 Software Requirements

Development Frameworks and Libraries:

- TensorFlow/Keras:
  For developing and refining deep learning models. Keras offers a high-level API to streamline model creation, whereas TensorFlow is an extremely potent open-source toolkit.
- PyTorch: An alternative deep learning framework noted for its dynamic computation graphs and ease of use in research.
- Scikit-learn: For additional machine learning algorithms and model evaluation.

Data Processing and Visualization:

- Pandas: For data manipulation and preprocessing.
- NumPy: For numerical operations and array manipulations.
- Matplotlib/Seaborn: For data visualization and plotting.

Development Tools:

- Jupyter Notebook: For interactive development, experimentation, and visualization of model training processes.

- VS Code/PyCharm: Integrated Development Environments (IDEs) for writing and managing code.

Libraries and Tools for Computer Vision:

- OpenCV: For image processing and augmentation, essential for preprocessing hand gesture data.

Version Control and Collaboration:

- Git/GitHub: For version control and collaboration on code and model changes.

3.2.2 Hardware Requirements

Computing Power:

- High-performance CPU:
  A multi-core CPU (such as the AMD Ryzen 7/9 or Intel i7/i9) for effective computing, particularly for data preprocessing and model training.
- GPU: A specialized GPU (such as the NVIDIA GTX 1660 or RTX 20xx/30xx series) to speed up deep learning model training. When training, GPUs take a lot less time than CPUs.
- RAM: 32 GB or more of RAM is advised for managing huge datasets and complicated models, although at least 16 GB is sufficient.

Storage:

- SSD: Multi-core processor with sufficient processing power.
- External Storage: Additional storage solutions for backup and archiving of datasets and models.

Additional Hardware:

- Webcamor Camera: If pre-collected datasets are not being used, to record hand motions during data collection.
- External Mouse and Keyboard: Peripherals with ergonomic design to boost output during extended development sessions.

It is possible to efficiently design, train, and test the "Sign Language Translation for Deaf " hand gesture recognition model while maintaining project efficiency and performance by making sure that these hardware and software requirements are satisfied.

3.3 Assumptions and Dependencies

3.3.1 Assumptions

- It is assumed that sufficient diversity and labeled examples of high-quality hand gesture datasets are either already available or can be gathered. The performance of the model will be greatly impacted by the quality and variety of the data. It is assumed that
- preprocessing the data—which includes labeling, augmentation, and normalization—can be done successfully. Perhaps the selected deep learning model architectures (e.g.,
- CNNs) are suitable for hand gesture recognition and will deliver acceptable performance. It predicts that the model will converge during training, and regularization and
- hyperparameter adjustment will be used to control overfitting or underfitting. The hardware (CPUs, GPUs, RAM) that is available is thought to be adequate for training the
- model in a fair amount of time. It is expected that the necessary computing resources will be allocated while employing cloud resources.

3.3.2 Dependencies

- For model training, it is essential to have access to large datasets or to be able to gather and annotate gesture data. Access to datasets and tools for preprocessing and data labeling are examples of dependencies. The availability and correct operation of
- computer vision libraries (OpenCV) and machine learning frameworks (TensorFlow/Keras, PyTorch) are prerequisites for the model's effective training. Dependencies include data processing and visualization tools (Pandas, NumPy) and
- integrated development environments (IDEs) such as VS Code or PyCharm. Dependence on sufficient computing resources for effective model training and experimentation, such
- as powerful GPUs and RAM.

4 SYSTEM DIAGRAMS

- CLASS DIAGRAM
- USE CASE DIAGRAM
- SEQUENCE DIAGRAM
- ACTIVITY DIAGRAM

4.1 Class Diagram

A class diagram is a graph of classified elements connected by their various static relationships. It is shown here for the Sign Language Translation for Deaf . This includes the System and the End-users as its main classes. Here are three different fields: Class, Attributes, and Operations. Class shows the class name, i.e., System, End-user, Data and Sign Clip. They are connected with each other through links and their relation with each other is shown through the numbers represented on the link; here * indicates zero or more multiplicity. Here Data class is connected to System through a Composition link which is the collaboration of all participants who are part of one composite class. Attributes provide the details of the Class while Operations show all possible operations respective classes can do in the system.

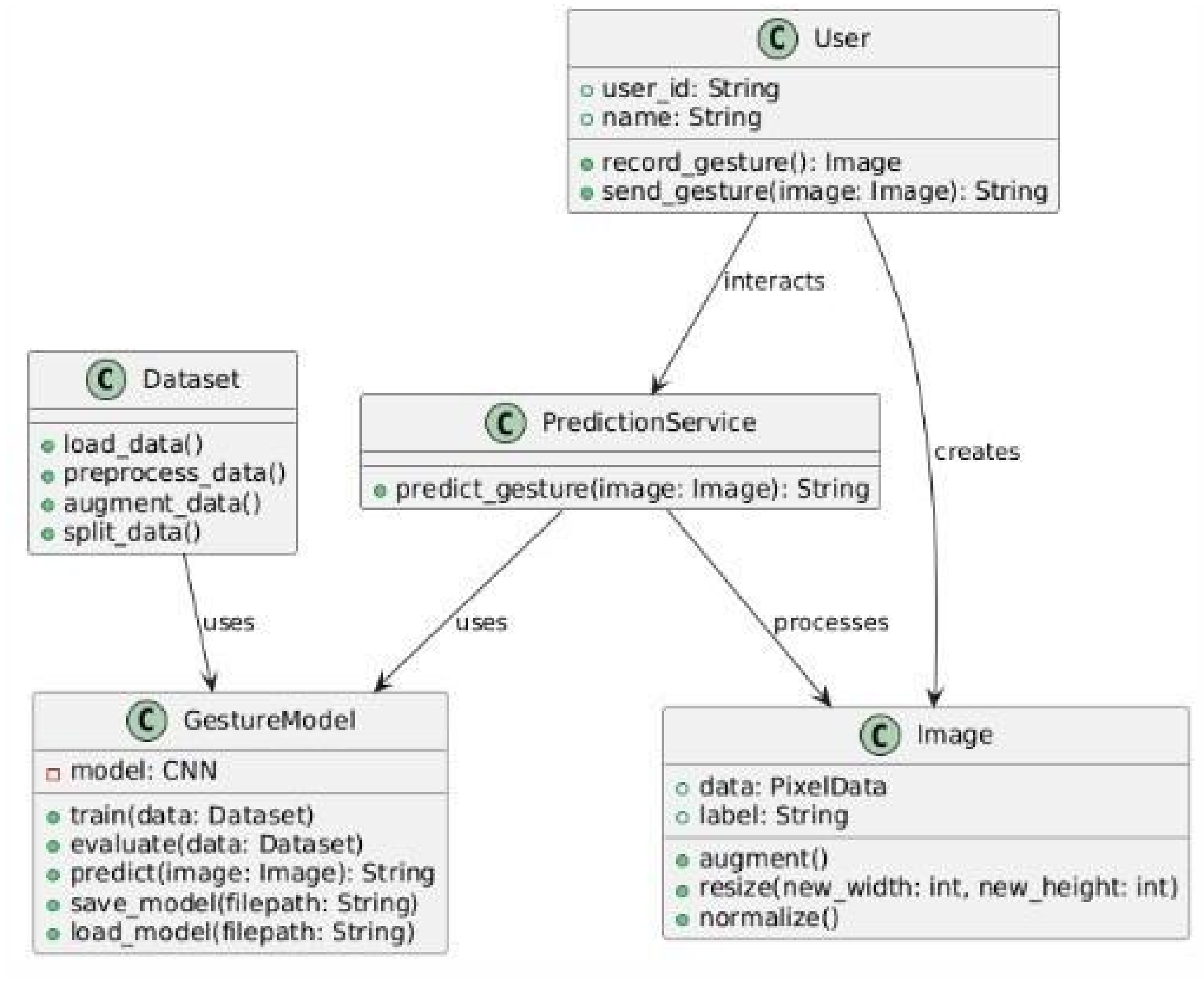

Fig 4.1 - Class diagram

4.2 Use Case Diagram

A use case diagram, in the structure of the "Sign Language Translation for Deaf " project, shows how users and the system interact, with particular attention to functions connected to gesture recognition and model training. The Speech-Impaired User, who uses hand gestures to communicate with the system, and the System Administrator, who oversees the dataset and trains the model, are the main actors. The two main use cases are Data Management, where the administrator uploads and analyzes gesture datasets, and Gesture Recognition, where the user enters hand gestures to be recognised and translated into text. Model training, in which the system administrator starts the model's training with the data gathered, and feedback collection, in which users contribute to increase system accuracy, are two other crucial use cases.

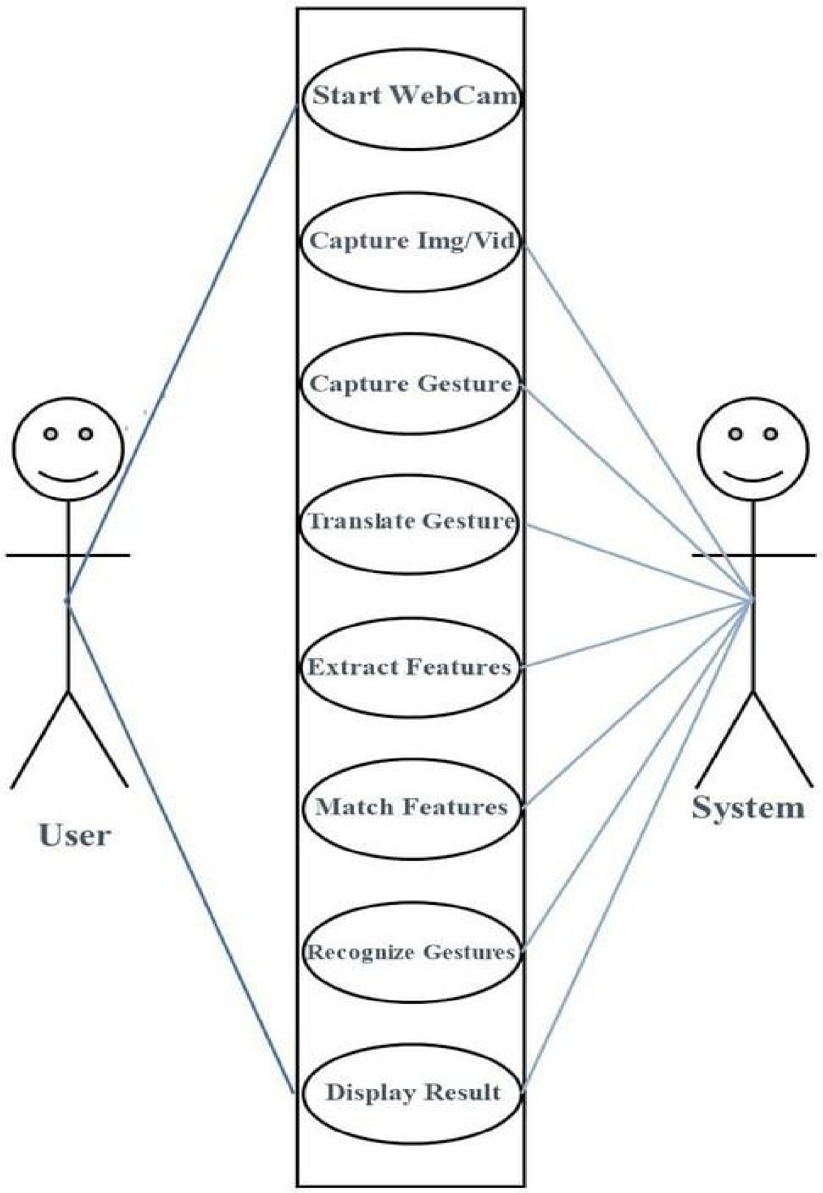

Fig 4.2 - Use case diagram

4.3 Sequence Diagram

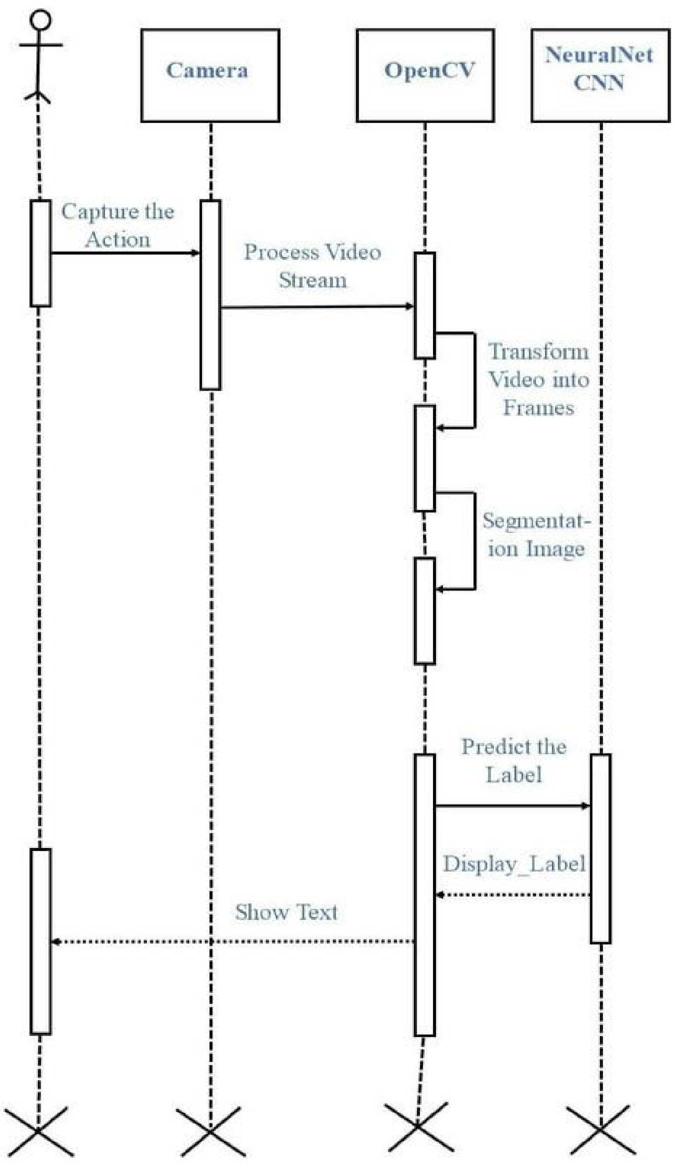

Fig 4.3 - Sequence diagram

4.4 Activity Diagram

An activity diagram is a special case of a state diagram in which all (or at least most) of the states are action states and in which all (or at least most) of the transitions are triggered by completion of the actions in the source states. Below are the activity diagrams for the actions performed by the end-user and the response of the system.

4.6(i) Activity diagram for Recording and Sending Gesture

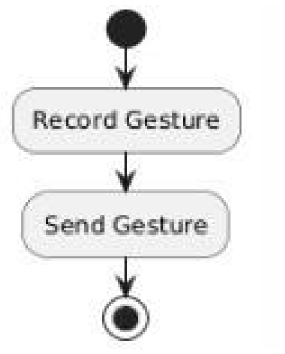

Fig 4.6(i) - Activity diagram for Recording and Sending Gesture

4.6(ii) Activity diagram for Preprocessing Gesture Data

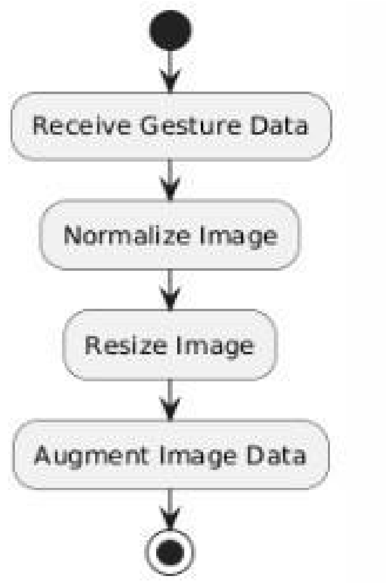

Fig 4.6(ii) - Activity diagram for Preprocessing Gesture Data

4.6(iii) Activity diagram for Predicting Gesture

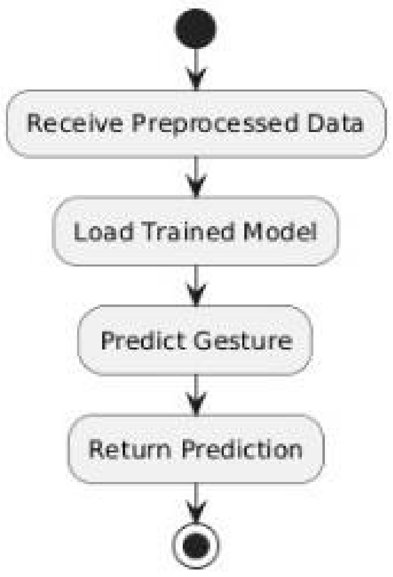

fig 4.6(iii) - Activity diagram for Predicting Gesture

4.6(iv) Activity diagram for Training Model

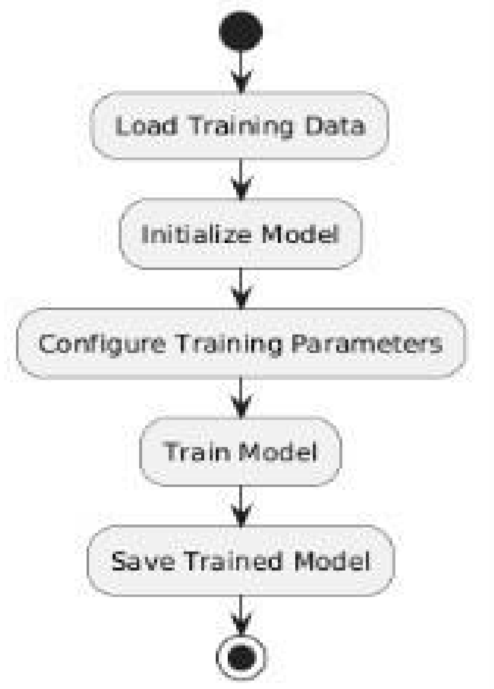

fig 4.6(iv) - Activity diagram for Training Model

5 DATADICTIONARY

- DATADICTIONARY

5.1 Data Dictionary

The different fields or data components that are utilized in a Sign Language Translation for Deaf , along with their definitions, data types, and any constraints or linkages, are usually listed in a data dictionary. As an illustration, consider this:

Table 5.1 - Input Data Dictionary

Field Name	DataType	Description
Image ID	STRING	Unique identifier for each hand gesture image.
Image Data	BINARY	Raw pixel data of the hand gesture image.
Label	STRING	The label or class of the hand gesture.
TimeStamp	DATETIME	Date and time when the image was captured.

Table 5.2 - Processed Data Dictionary

Field Name	DataType	Description
Normalized Image	ARRAY	Normalized pixel data of the image.
Feature Vector	ARRAY	Extracted features from the image used for training.
Data Format	STRING	Format of the processed data.
Image Dimensions	TUPLE	Dimensions of the processed image.
Preprocessing Steps	LIST	Steps applied during preprocessing.

Table 5.3 - Model Configuration Dictionary

Field Name	DataType	Description
Hyperparameters	DICT	Dictionary of hyperparameters used for training.
Architecture	STRING	Details of the model architecture.
Training Epochs	INT	Number of training iterations.
Learning Rate	FLOAT	Learning rate used during training.
Batch Size	INT	Size of the training batches.

Table 5.4 - Training Metrics Data Dictionary

Field Name	DataType	Description
Accuracy	FLOAT	Model accuracy during training.
Loss	FLOAT	Loss value calculated during training.
ValidationScore	FLOAT	Model performance on validation data.
Precision	FLOAT	Precision of the model for classification tasks.
Recall	FLOAT	Recall of the model for classification tasks.

Table 5.5 - Hardware Data Dictionary

Field Name	DataType	Description
Hardware ID	STRING	Unique identifier for each payment record
Specifications	DICT	Detailed specifications (e.g., resolution,

Table 5.6 - Software Data Dictionary

Field Name	DataType	Description
Software ID	STRING	Unique identifier for each software component.
Software Name	STRING	Name of the software (e.g., TensorFlow, OpenCV)
License Type	STRING	Type of license
Installation Path	STRING	Path where the software is installed.
Version	STRING	Version number of the software.

Table 5.7 - Checkpoint Data Dictionary

Field Name	DataType	Description
Checkpoint ID	STRING	Unique identifier for each model checkpoint.
Model Version	STRING	Version of the model at the checkpoint.
Save Date	DATE	Date when the checkpoint was saved.
TrainingEpoch	INT	Training epoch at which the checkpoint was saved.
Metrics	DICT	Metrics at the checkpoint.

Table 5.8 - Validation Data Dictionary

Field Name	DataType	Description
ValidationID	STRING	Unique identifier for each validation run.
Dataset	STRING	Name or ID of the validation dataset.
Metrics	DICT	Performance metrics from validation.
ValidationDate	DATE	Date when the validation was performed.
Comments	STRING	Additional comments or notes about the validation.

Table 5.9 - Testing Data Dictionary

Field Name	DataType	Description
Test ID	STRING	Unique identifier for each test case.
Test Type	STRING	Type of test (e.g., unit test, integration test).
Test Results	DICT	Results of the test
Test Date	DATE	Date when the test was executed.
Tester	STRING	Name or ID of the person who performed the test.

This is a simplified example, and in a real-world scenario, the data dictionary would likely include additional details such as constraints, validation rules, and relationships between tables. It serves as a valuable reference for developers, database administrators, and other stakeholders involved in the project.

6. DISCUSSION AND CONCLUSION

- RESULT AND DISCUSSION
- CONCLUSION

Result and Discussion

The "Sign Language Translation for Deaf" project effectively created a hand gesture recognition system based on AI/ML designed to help people with speech impairments communicate. The system used a convolutional neural network (CNN) model to perform strong real-time gesture detection after an extensive 12-week development process. High accuracy in identifying specified hand gestures was shown by the model, and performance measures showed an excellent trade-off between recall and precision. With the help of React.js and OpenCV.js, real-time gesture detection was skilfully incorporated into a user-friendly interface that guaranteed fluid interaction and prompt response. The backend, which was driven by Flask, TensorFlow/Keras, Python, and Flask, handled model inference and API queries effectively, producing text output for gesture recognition quickly and accurately. Positive comments underlined the system's functional precision and intuitive design, and user testing validated its efficacy in promoting communication.

Conclusion

The "Sign Language Translation for Deaf" system has proven to be a valuable tool for bridging communication gaps for speech-impaired individuals. By leveraging advanced computer vision and deep learning techniques, the project has created a practical and accessible solution that translates hand gestures into text. Along with thorough testing and validation, has resulted in a reliable and efficient application. The project's outcomes validate its potential to enhance communication for users with speech impairments, offering a foundation for future improvements such as supporting more complex gestures and integrating voice synthesis. Overall, "Sign Language Translation for Deaf" stands as a significant step forward in the realm of assistive technologies, demonstrating both technical success and meaningful impact.

7 REFERENCES

- REFERENCES

7.1 References

Here are some references and resources that can help in the development of a Sign Language Translation for Deaf

Websites:

1 https://ieeexplore.ieee.org/document/9825192

2 https://ieeexplore.ieee.org/document/8602809

3 https://link.springer.com/article/10.1007/s11760-023-02626-8

4 https://www.computervision.zone/courses/hand-sign-detection-

Youtube:

5 https://www.youtube.com/watch?v=a99p_fAr6e4&list=PL0FM467k5KSyt5o3ro2fyQG

4 t-6zRkHXRv

6 https://www.youtube.com/watch?v=wa2ARoUUdU8&list=PL0FM467k5KSyt5o3ro2fy

QGt-6zRkHXRv&index=2

7. https://www.youtube.com/watch?v=pDXdlXlaCco&list=PL0FM467k5KSyt5o3ro2fyQ

Gt-6zRkHXRv&index=4

GitHub:

8 https://github.com/nicknochnack/RealTimeObjectDetection

9 https://github.com/kinivi/hand-gesture-recognition-mediapipe

Open AI:

10 https://chatgpt.com/c/e0b7886c-b903-4e7b-a9d3-4b2fe6ccd82a