SIGN LANGUAGE RECOGNITION USING IMAGE BASED HAND GESTURE RECOGNITION TECHNIQUE

A PROJECT REPORT

Submitted by

SARATH KUMAR U [211420104246]

THARANI T [211420104287]

THIRUGNANAM B [211420104290]

in partial fulfillment for the award of the degreeof

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report titled "**SIGN LANGUAGE RECOGNITION USING IMAGE BASED HAND GESTURE RECOGNITION TECHNIQUE**", is the bonafide work of "**SARATH KUMAR U, THARANI T** and **THIRUGNANAM B"** who carried out the work under my supervision.

Signature of the HOD with date

DR L. JABASHEELA M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Signature of the Supervisor with date

MRS. V. ANITHA MOSES M.E.,
PROFESSOR,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Certified that the above candidate(s) was examined in the End Semester Project Viva- Voce Examination held on .....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We Sarath Kumar U (211420104246), Tharani T (211420104287) and Thirugnanam B (211420104290) hereby declare that this project report titled “Sign language detection system using machine learning” , under the guidance of Mrs. V. Anitha Moses Moses M.E., is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SARATH KUMAR U-211420104246

THARANI T-211420104287

THIRUGNANAM B-2114210104290

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Dr. V. Subedha M.Tech., Ph.D.,** and **Mrs. V. Anitha moses, M.E**, and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

SARATH KUMAR U-211420104246

THARANI T-211420104287

THIRUGNANAM B-211410104290

ABSTRACT

The Sign Language Detection System Leveraging Machine Learning stands at the forefront of technological innovations aimed at facilitating communication for the hearing-impaired. This groundbreaking application addresses the unique needs of the deaf community by harnessing the power of advanced machine learning algorithms. The system's core functionality revolves around interpreting and recognizing sign language gestures in real-time, offering a transformative bridge between the signing and non-signing communities.

At its heart, the system relies on a diverse and extensive dataset encompassing a broad spectrum of sign language gestures. This rich dataset empowers the machine learning model to comprehend and generalize across various sign languages and dialects, ensuring adaptability and inclusivity. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) serve as the backbone of the system, effectively capturing both spatial and temporal features from video inputs to accurately interpret the intricacies of sign language expressions.

What sets this system apart is its dynamic nature. It continuously evolves through the integration of deep learning techniques, ensuring not only real-time interpretation but also the ability to learn and adapt to new signs. Furthermore, the system incorporates features like adaptive learning, allowing users to personalize the model to suit individual signing styles, thereby enhancing the user experience.

In essence, the Sign Language Detection System signifies a monumental leap forward in fostering inclusivity across educational, professional, and social realms. By breaking down communication barriers, this system contributes to the creation of a more connected and inclusive world for individuals with hearing impairments.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	DEFINITION
1	ASL	American Sign Language
2	CSL	Common sign Language
3	SQL	Structured Query Language

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

In a world where human connection thrives on communication, the deaf and mute community faces unique challenges. The World Health Organization reports that 466 million individuals, approximately 5% of the global population, grapple with hearing and speech disabilities. Addressing this need, the Sign Language Detection System emerges as a transformative solution, employing machine learning and computer vision to enable seamless communication for those with hearing and speech challenges. This insight illuminates the isolating experiences of the deaf and mute, emphasizing the urgent need for innovative solutions. Sign language, a nuanced visual-gestural language, serves as a vital means of expression for those with hearing impairments. However, translating this form of communication poses a challenge, necessitating advanced technologies like the Sign Language Detection System.

Operating on a dual classification approach, the system focuses on Isolated sign language recognition and Continuous sign language recognition. The former analyzes individual steady signs, forming the foundation for essential communication elements. Continuous sign language recognition tracks dynamic gestures and signs based on consecutive movements, providing a fluid representation of communication. This paper primarily explores Isolated sign language recognition, utilizing real-time images of 24 alphabetical signs in American Sign Language.At its core, the system integrates computer vision, image processing, and machine learning, with the convolutional neural network (CNN) playing a key role. This deep learning method identifies image features, extracting intricate details like finger shapes through multiple layers of abstraction. The result is a sophisticated classification system that categorizes images, offering a granular understanding of signed gestures.

Real-time identification empowers individuals with hearing and speech impairments, providing a seamless means to convey messages in a world often silent to their needs. Envisioning an inclusive future, the technology opens avenues for converting sign language into on-screen text or integrating voice-enabled functionality, expanding communication possibilities. This introduction delves into the Sign Language Detection System, a groundbreaking innovation poised to redefine communication paradigms and foster inclusivity for the deaf and mute community, guiding us toward a future.

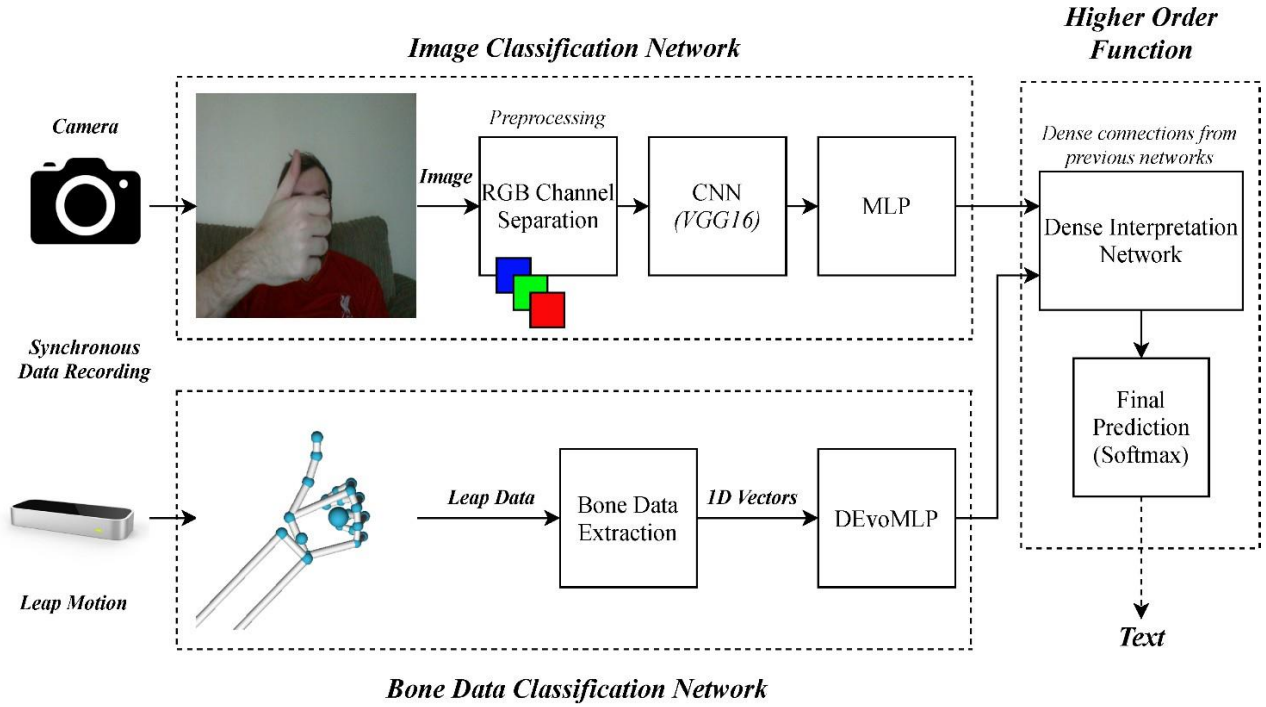


Fig 1.1 System Architecture

1.2 PROBLEM DEFINITION

The problem definition of a Sign Language Detection System involves creating a technology that can accurately interpret and translate sign language gestures into meaningful and understandable text or spoken language. Sign language is a vital mode of communication for individuals with hearing impairments, and developing a reliable system to interpret these gestures can enhance accessibility and inclusion.

Challenges in this domain include the vast diversity of sign languages across different regions, variations in individual signing styles, and the dynamic nature of sign language expressions. The system must be capable of recognizing and interpreting a wide range of gestures, considering factors like facial expressions, body movements, and hand shapes.

Another aspect of the problem is real-time processing, as communication in sign language often requires quick and fluid exchanges. The system needs to be efficient enough to provide timely translations to facilitate seamless communication.

Furthermore, ensuring the accuracy and reliability of the detection system is crucial, as misinterpretations could lead to misunderstandings. Overall, the problem definition encompasses developing a robust, adaptable, and accurate Sign Language Detection System to bridge communication gaps and promote inclusivity for the deaf and hard-of-hearing community.

1.3 OBJECTIVE

The Sign Language Detection System using machine learning addresses the pervasive communication challenges faced by the deaf and mute community. With approximately 5% of the global population, amounting to 466 million individuals, experiencing hearing and speech disabilities, there is an urgent need for a robust solution. Traditional communication methods often fall short in meeting the unique needs of this demographic, leading to isolation and limited interaction. The problem at hand is twofold: first, the intricate nature of sign language, a visual-gestural language, poses a substantial obstacle in translation for those unfamiliar with its nuances. Second, the lack of real-time, accessible communication tools hinders the ability of individuals with hearing and speech impairments to effectively convey messages in a world predominantly designed for verbal communication.3The Sign Language Detection System aims to bridge this communication gap by leveraging machine learning, computer vision, and image processing. The specific challenge lies in developing a system capable of accurately recognizing both Isolated and Continuous sign language, ensuring real-time identification of gestures and signs to facilitate seamless communication for the deaf and mute community. Addressing these issues is crucial for fostering inclusivity and empowering individuals with hearing and speech challenges to engage more fully in social, educational, and professional spheres

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2

LITERATURE REVIEW

1. Nearest neighbour classification of Indian sign language gestures using kinect camera

Authors: Zafar Ahmed Ansari et.al.

Description

This paper conducted extensive research to accurately categorize gestures in Indian Sign Language, classifying 140 gestures classes, including finger-spelling numbers, alphabets and common phrases. The dataset employed a Kinect sensor, capturing RGB images (640&480 resolution) and corresponding depth data.However, the method of depth-based palm detection had inconsistencies, rendering the dataset unsuitable for convolutional neural network training

2. Structured Feature Network for continuous Sign Language RecognitionAuthors: Zhaoyang Yang et.al.

Description

This paper delves into method for recognizing sign language, where a “gloss” is defined as a unit composed of gestures, motions, and facial expressions. Sign Language Recognition(SLR) is categorized into isolated SLR and continuous SLR. Isolated SLR

Involves segmenting frames or signs individually, predicting by considering a single running gloss.

3. Sign Language Recognition Using Image Based Hand Gestures Recognition Technique

Authors: Ashih S. Nikam et.al.

Description

This paper explores the utilization of hand gestures as a communication method in sign language. Each gesture carries a specific meaning, and complex meanings can be derived through the combination of basic elements. Sign language approaches can broadly be categorized into vision-based and sensor -based methods. In this study, a vision-based approach is adopted, consisting of two phases: sign detection and sign recognition

4. Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images

Authors: Aditiya das et.al.

Description

This paper focuses on sign language detection through image classification, employing a machine learning algorithm, specifically a Convolutional Neural Network (CNN). The image dataset primarily consists of static gestures representing alphabets and numbers, successfully detected using the proposed neural network.

5. **A Persian Sign Language (PSL) recognition using wavelet transform and neural networks.**

Authors: Karami et.al.

Description

This comprehensive review introduces a novel system leveraging a MultiLayered Perceptron Neural Network (MLPNN) for the intricate task of recognizing 32 classes within Persian Sign Language (PSL). The classification process is enriched by features extracted through discrete wavelet transform (DWT), contributing to a remarkable recognition accuracy of 94.06%. This amalgamation of MLPNN and DWT signifies a robust approach in overcoming the complexities of PSL, offering a promising avenue for efficient and accurate sign language recognition.

6. **Real-Time Arabic Sign Language (ArSL) Recognition**

Authors: Nadia R.Albelwi et.al.

Description

This study presents a real-time Arabic Sign Language recognition system incorporating a video camera for live input. Hand tracking is achieved through a Haar-like algorithm applied to video frames, and preprocessing techniques, including skin detection and size normalization, are implemented to refine the region of interest. These steps enhance the system's accuracy by isolating the hand and ensuring consistency across various hand sizes. The proposed approach aims to provide effective real-time Arabic Sign Language interpretation, fostering improved communication accessibility for the hearing-impaired within Arabic-speaking communities.

7. A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images

Authors: Geetha M et.al

Description

This paper proposed a novel vision-based recognition system for Indian sign language alphabets and digits, utilizing bspline approximation . The system employs Maximum Curvature Points (MCPs) as control points to approximate the extracted boundary from the region of interest to a B-Spline curve. Through iterative smoothing of the B-spline curve, Key Maximum Curvature Points (KMCPs) are extracted, representing the major contributors to the gesture shape.

8. Video Audio Interface for Recognizing Gestures of Indian Sign Language

Authors: P.V.V Kishore et.al.

Description

This paper proposed a system capable of recognizing and converting ISL gestures from a video feed into English voice and text. The recognition process involved segmenting shapes in the video stream using image processing techniques such as edge detection, wavelet transform, and picture fusion. Shape features were extracted using Ellipsoidal Fourier descriptors, and Principal Component Analysis (PCA) was applied to optimize and reduce the feature set. The system was resulting in a commendable 91% accuracy.

9. Artificial Neural Network Based Method for Indian Sign Language Recognition.

Authors: Adithya. V et.al

Description

This paper proposed a method for the automatic recognition of Indian Sign Language gestures, leveraging digital image processing techniques. The method utilizes the YCbCr color space for hand detection, involving a transformation of the input image. Various techniques, including distance transformation, projection of distance transformation coefficients, Fourier descriptors, and feature vectors, are employed to extract relevant features. Classification of the data is accomplished using an artificial neural network, resulting in an impressive recognition rate of 91.11 percent.

10. Real time Indian Sign Language Recognition System to aid deaf-dumb people.

Authors: Balakrishnan G et al.

Description

This paper proposed a system that translates a set of 32 combinations of binary numbers, representing UP and DOWN positions of the five fingers, into decimal form [22]. The binary numbers undergo conversion into decimal using a binary-decimal conversion algorithm, and subsequently, the decimal values are mapped to their corresponding Tamil letters. Static images of the gestures serve as input to the system, where a Canny-edge detection algorithm is employed to extract palm edges. The Euclidean Distance is then applied to identify the positions of the fingers. Impressively, the system achieved an accuracy of 98.75%.

CHAPTER 3
THEORITICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

The proposed endeavor involves the creation of a sophisticated Sign Language Detection System (SLDS), integrating advanced computer vision and machine learning methodologies for the precise identification and interpretation of sign language gestures. With a focus on augmenting communication accessibility for individuals with hearing impairments, the SLDS will employ a comprehensive dataset of sign language gestures to train a deep learning model, ensuring robust recognition and translation capabilities into textual or auditory forms.

The SLDS will boast real-time sign language recognition, facilitating seamless communication between sign language users and those unfamiliar with the language. Versatility is a key design element, encompassing support for multiple sign languages and compatibility across various devices, including webcams and smartphones, to cater to diverse user scenarios.

The project demands a collaborative effort, involving expertise in computer vision, machine learning, and sign language proficiency to curate an extensive and representative dataset. The developmental phase will encompass rigorous model training and fine-tuning processes, addressing challenges such as varied signing styles and environmental nuances that may impact recognition accuracy.

Ultimately, the Sign Language Detection System aspires to be a pioneering solution, employing cutting-edge technologies to bridge communication divides for the hearing-impaired community. Through technological innovation, this initiative endeavors to champion and accessibility, aligning computational techniques for societal betterment.

3.2 WORKFLOW

The Sign Language Detection System (SLDS) workflow commences with an exhaustive data collection phase, sourcing a diverse dataset that spans various sign languages and encompasses a myriad of environmental conditions, followed by meticulous preprocessing to enhance data quality. The model architecture, a sophisticated amalgamation of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), undergoes an intensive training process with careful validation and fine-tuning iterations to optimize its recognition capabilities. Real-time sign language recognition is implemented, utilizing video inputs from devices like webcams and smartphones, integrated seamlessly with a translation module for the conversion of recognized gestures into both textual and auditory outputs.

To ensure universal accessibility, the SLDS is meticulously engineered for cross-device compatibility, catering to both web and mobile platforms. The user interface, designed for intuitive interaction, offers real-time feedback on detected signs, creating a user-friendly experience. The system undergoes comprehensive testing and evaluation, assessing performance metrics such as accuracy, precision, and recall to guarantee its reliability before deployment.

Post-deployment, user training materials and comprehensive documentation are disseminated to facilitate seamless adoption. Continuous improvement mechanisms, driven by user feedback and ongoing monitoring, propel the SLDS into a state of perpetual refinement, ensuring its status as a cutting-edge, inclusive communication tool for individuals with hearing impairments across diverse contexts and scenarios. The iterative nature of this workflow underscores the commitment to excellence and adaptability in meeting the evolving needs of the user community.

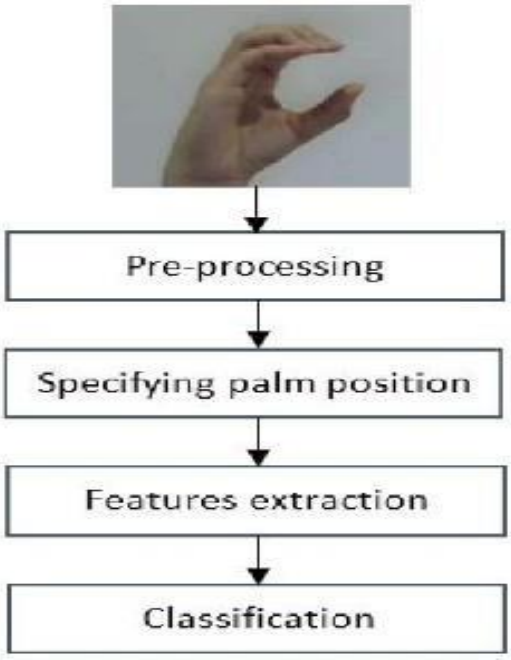


Fig 3.1 Workflow

3.3 SYSTEM ARCHITECTURE

A comprehensive sign language detection system is intricately designed, encompassing a multi-faceted architecture to facilitate accurate interpretation and communication of sign language gestures. The initial phase involves the acquisition of video or image data, utilizing cameras or input devices, supplemented by optional depth sensors for a more nuanced 3D perspective. Subsequent preprocessing steps refine the data quality, employing image and video processing techniques, while feature extraction isolates critical elements such as hand gestures, facial expressions, and body movements.

Gesture recognition, at the core of the system, leverages advanced machine learning models, encompassing both deep learning and traditional computer vision algorithms. These models decode the extracted features, translating them into recognized sign language gestures. The subsequent conversion of these gestures into text or alternative communicative forms enhances user comprehension. The user interface serves as the visual representation, displaying the interpreted sign

language, complemented by a feedback mechanism that provides users with confirmation signals, either visually or audibly.

The system architecture accommodates optional components, such as model training for machine learning, involving the meticulous collection and labeling of datasets. Integration with existing systems is facilitated through APIs or interfaces. Security measures, such as data encryption and user authentication, bolster the system's integrity, ensuring privacy and protection of sensitive information. Continuous improvement mechanisms, embedded within the architecture, utilize feedback loops for ongoing learning and refinement. Overall, this sophisticated architecture is tailored to meet the specific needs and constraints of sign language users, providing an effective and seamless means of communication.

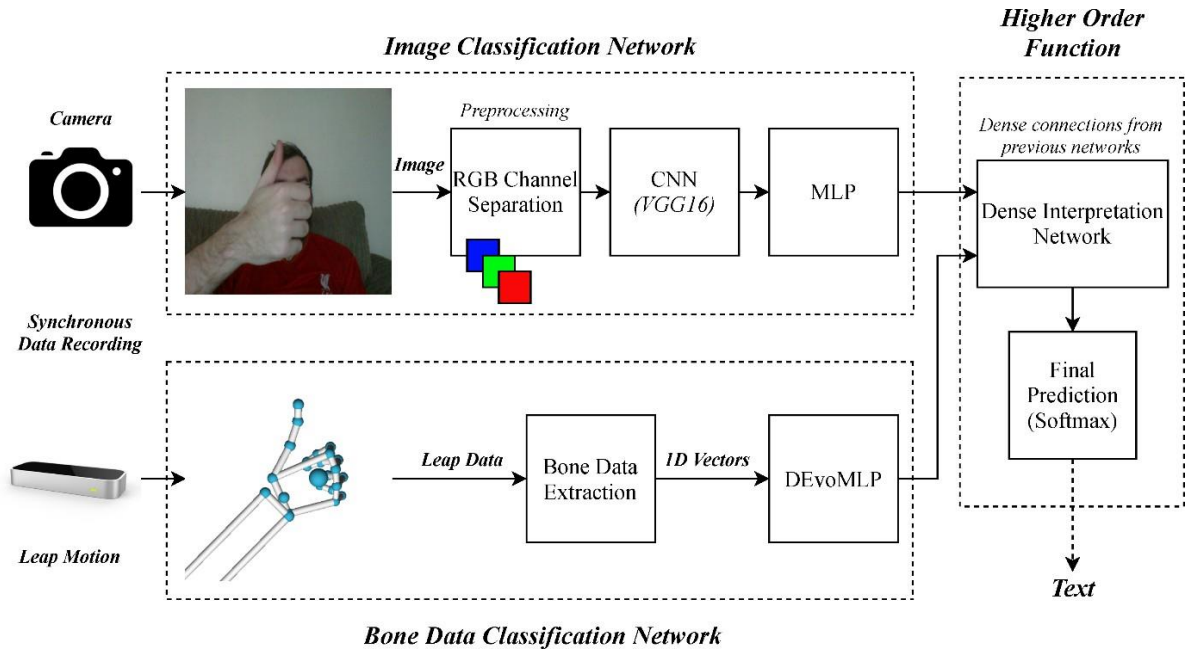


Fig 3.2 System Architecture

3.4 PROPOSED METHODOLOGY

3.4.1 DATASET DESCRIPTION

A dataset for sign language detection using machine learning typically includes annotated images or videos of individuals performing sign language gestures. The dataset is carefully curated to encompass a diverse range of signs, hand configurations, facial expressions, and body movements. Each sample in the dataset is labeled with the corresponding sign or gesture it represents, allowing machine learning models to learn and generalize from the data.

Key components of a sign language detection dataset:

1. Image/Video Samples:

The dataset comprises a collection of images or video frames capturing sign language gestures. The diversity of samples ensures a comprehensive representation of various signs and expressions.

2. Annotations:

Each sample is annotated with the correct sign label, indicating the corresponding sign language gesture. Accurate annotations are crucial for training and evaluating machine learning models.

3. Variability:

The dataset includes variations in lighting conditions, backgrounds, and signer demographics to enhance the model's robustness and adaptability.

4. Gesture Classes:

The dataset defines a set of gesture classes representing different signs or phrases in the chosen sign language. The variety of classes ensures a rich and comprehensive representation of the language.

5. Temporal Aspect:

For video datasets, the temporal aspect is considered, capturing the dynamic nature of sign language. This allows models to understand the temporal sequencing of gestures for accurate recognition.

6. Ethical Considerations:

Privacy and ethical considerations are taken into account, ensuring the dataset is collected and used responsibly, with proper consent and respect for the privacy of the sign language users.

Popular datasets for sign language detection include the American Sign Language (ASL) dataset, which captures a wide range of ASL gestures, and datasets specific to other sign languages. These datasets serve as foundational resources for developing and evaluating machine learning models aimed at sign language recognition and interpretation.

3.4.2 INPUT DESIGN

The Sign Language Recognition UI is thoughtfully crafted to offer a holistic and enriched user experience, blending essential features with innovative elements. The designated Gesture Input Zone serves as the central hub for users to express sign language gestures, receiving real-time visual feedback for immediate confirmation of accurately interpreted signs. The integration of instructional elements, including visual cues and animated guides, ensures users receive guidance for correct sign execution, fostering an environment conducive to learning. Beyond the basics, the UI incorporates an engaging Gamification Module, introducing interactive games and challenges that transform the learning process into an enjoyable and competitive experience, complete with achievement tracking and friendly competitions. Social Sharing functionalities enable users to connect with the broader sign language community, sharing achievements and fostering a sense of camaraderie. Additionally, the system supports Multi-modal Input, allowing users to combine sign gestures with voice commands, providing a nuanced and comprehensive communication experience. This multifaceted approach positions the Sign Language Recognition UI not only as a tool for accurate interpretation but also as a dynamic platform that promotes learning, community engagement, and growth among users with diverse communication needs.

Fig 3.3 Input design

3.4.3 MODULE DESIGN

UML DIAGRAMS

Use case Diagram

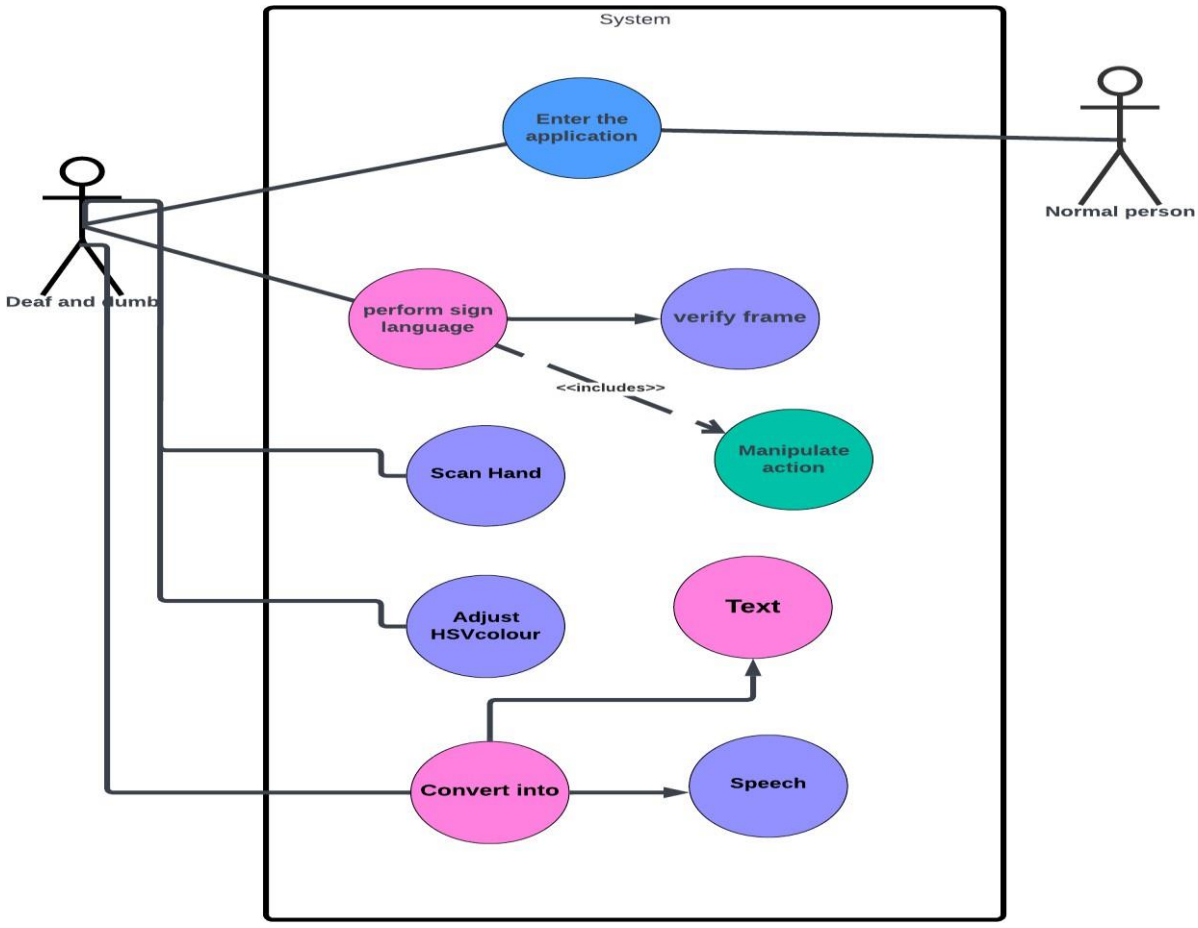


Fig 3.4 USE CASE DIAGRAM

Sequence Diagram

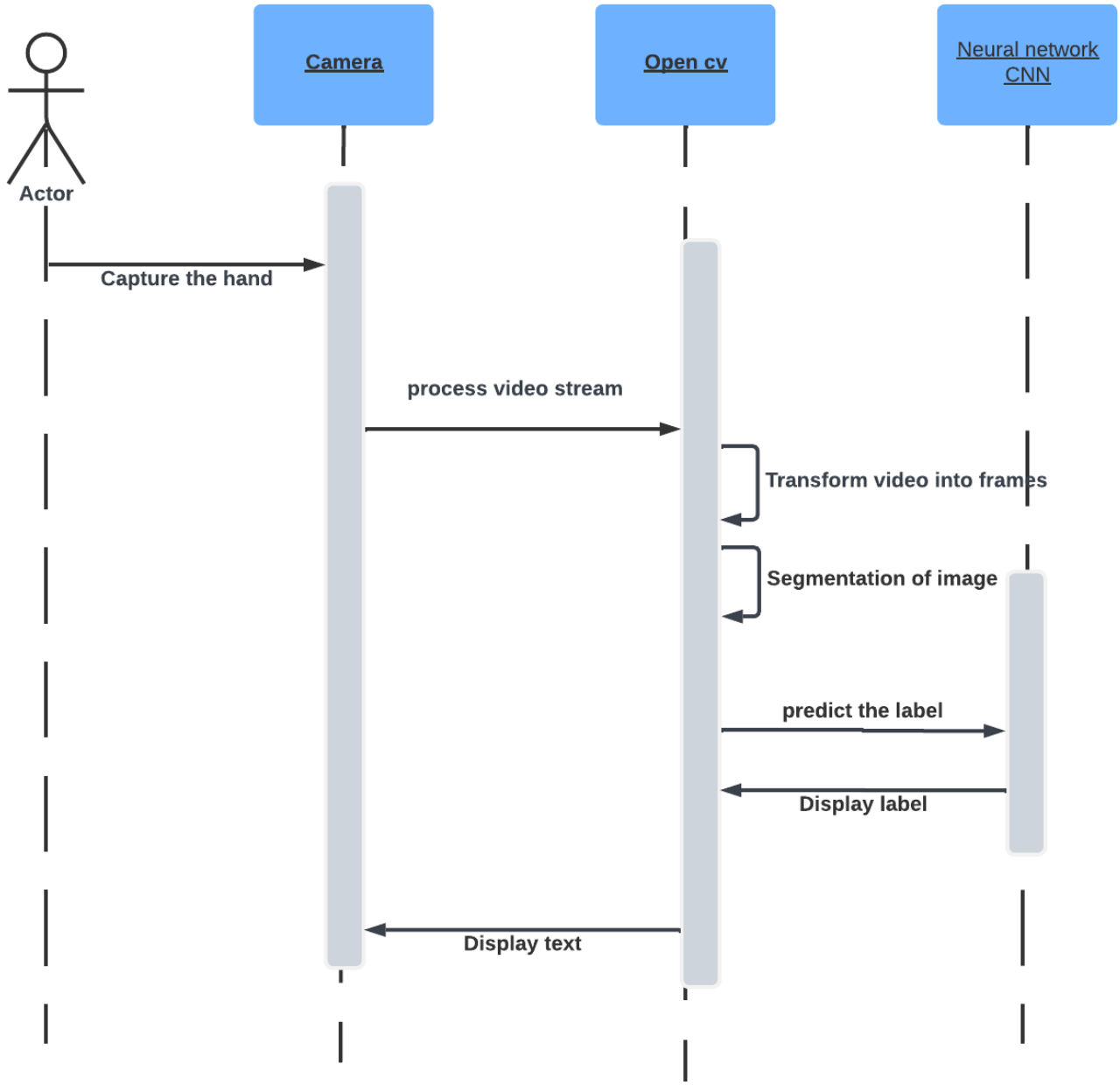


Fig 3.5 SEQUENCE DIAGRAM

Class Diagram

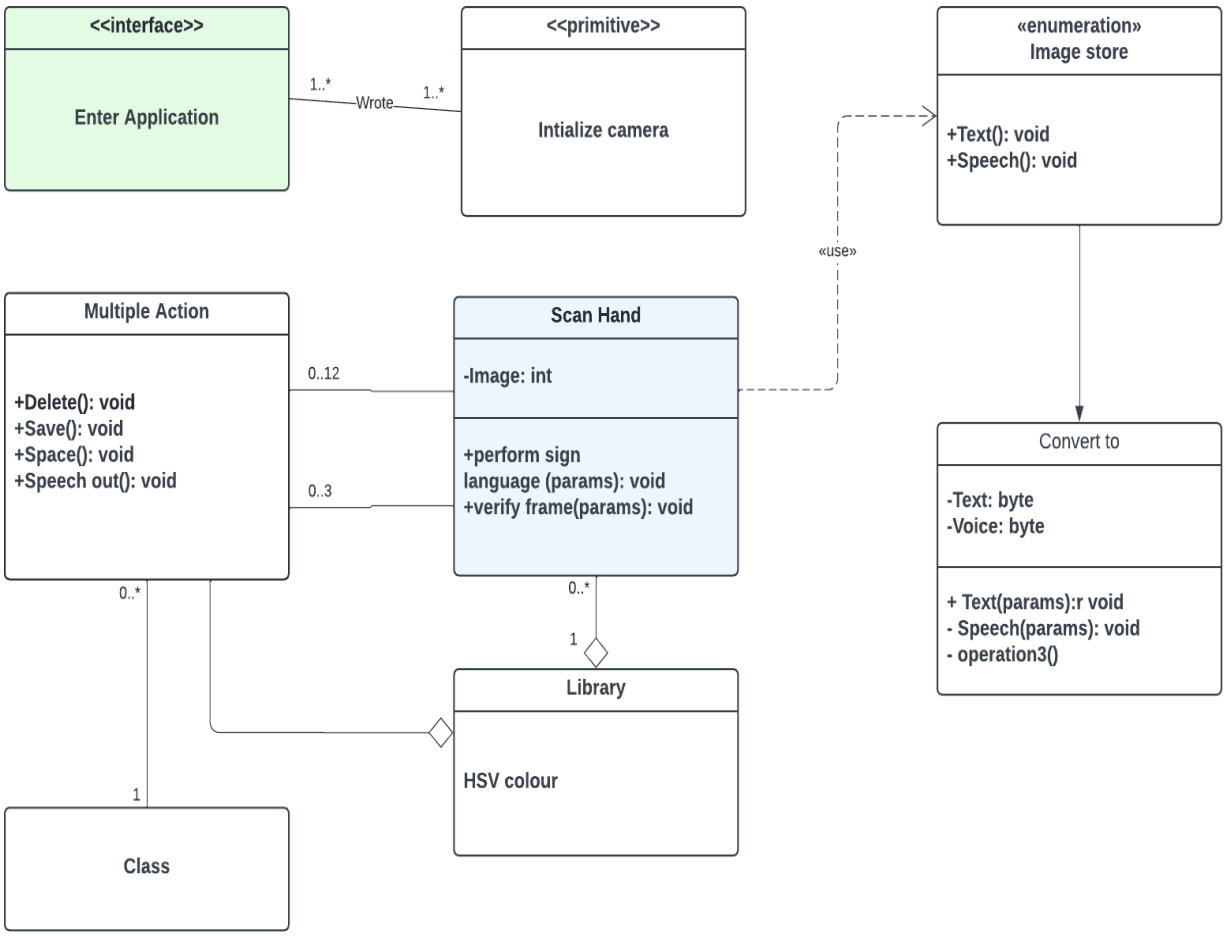


Fig 3.6 CLASS DIAGRAM

State Chart Diagram

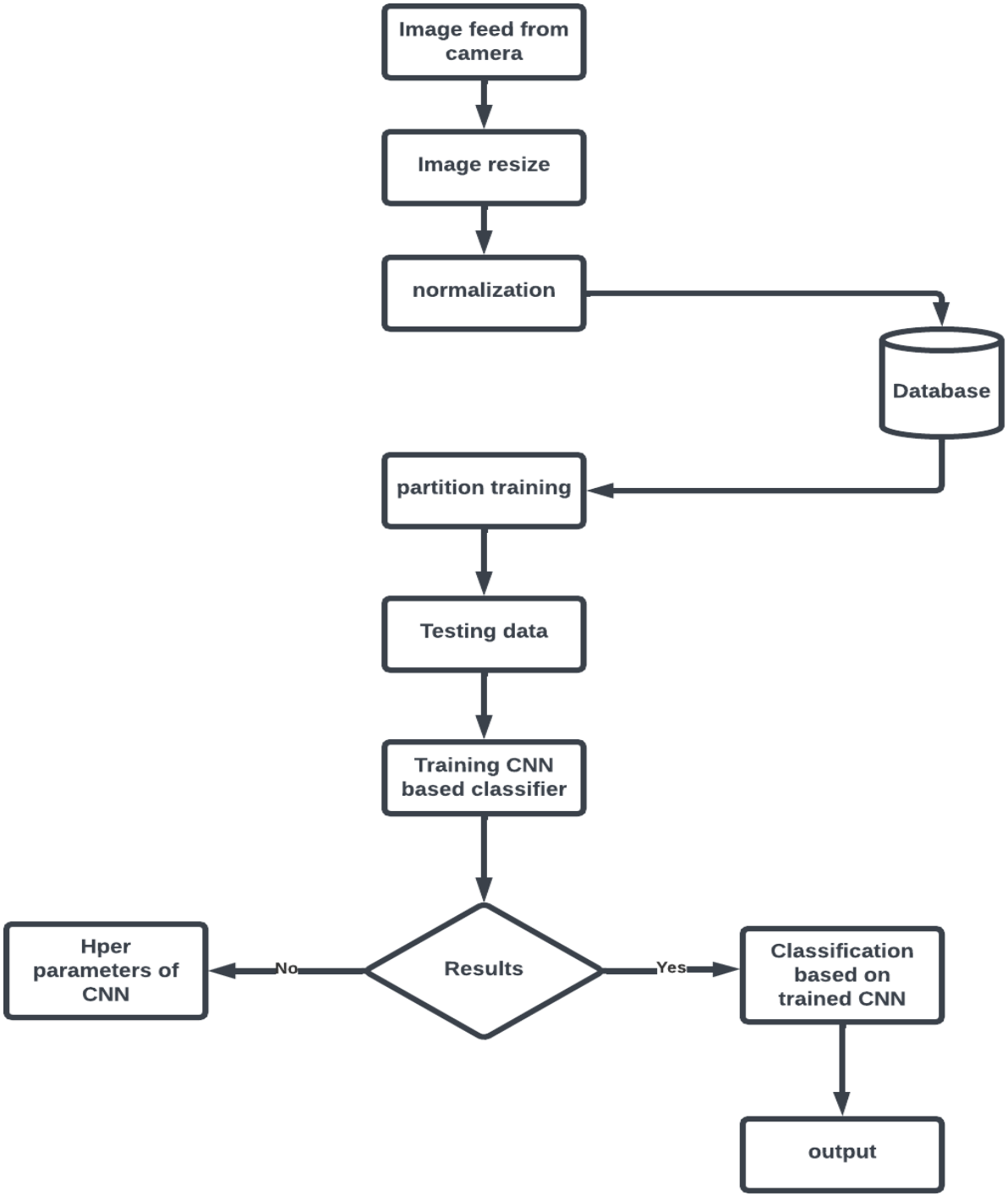


Fig 3.7 STATE CHART DIAGRAM

Communication Diagram

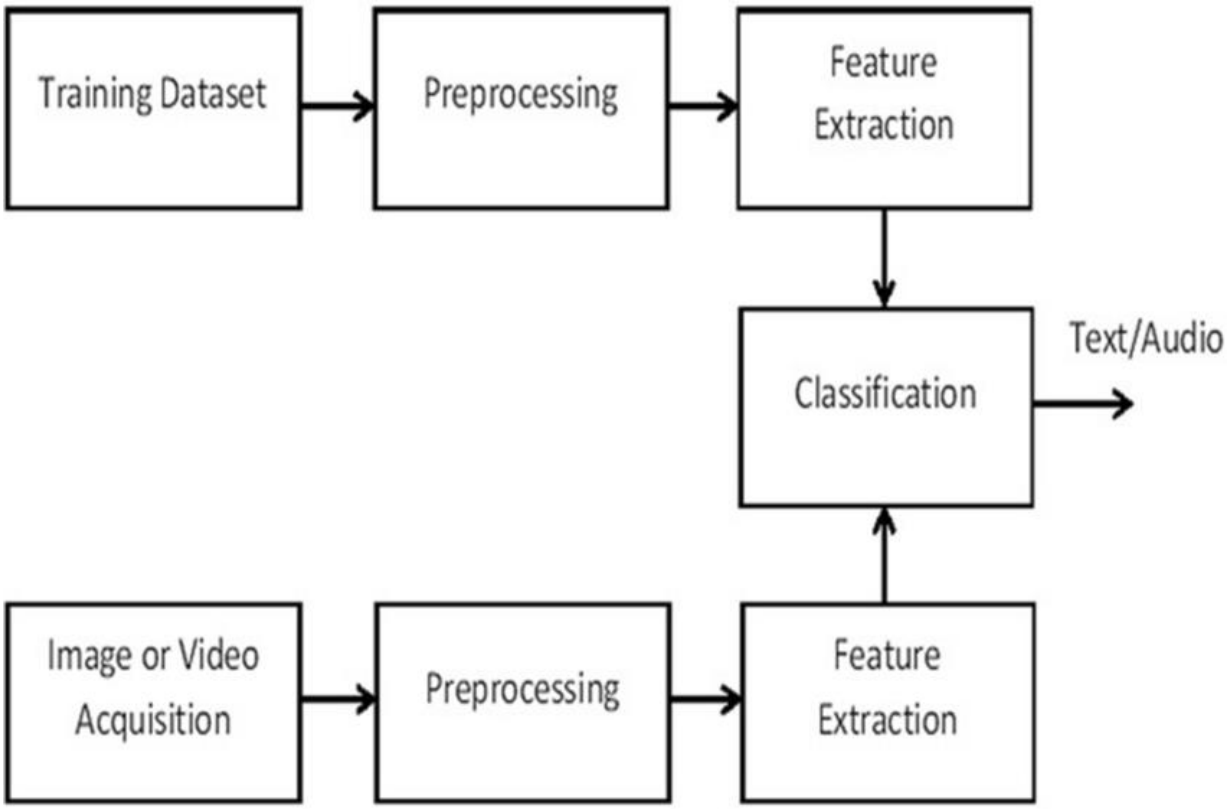


Fig 3.8 COMMUNICATION DIAGRAM

Package Diagram

Fig 3.9 PACKAGE DIAGRAM

Component Diagram

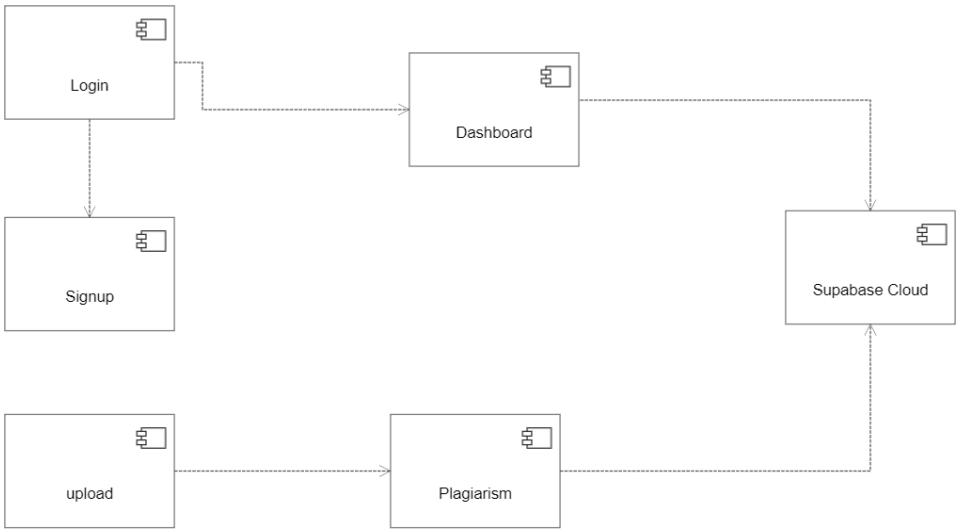


Fig 3.10 COMPONENT DIAGRAM

ER Diagram

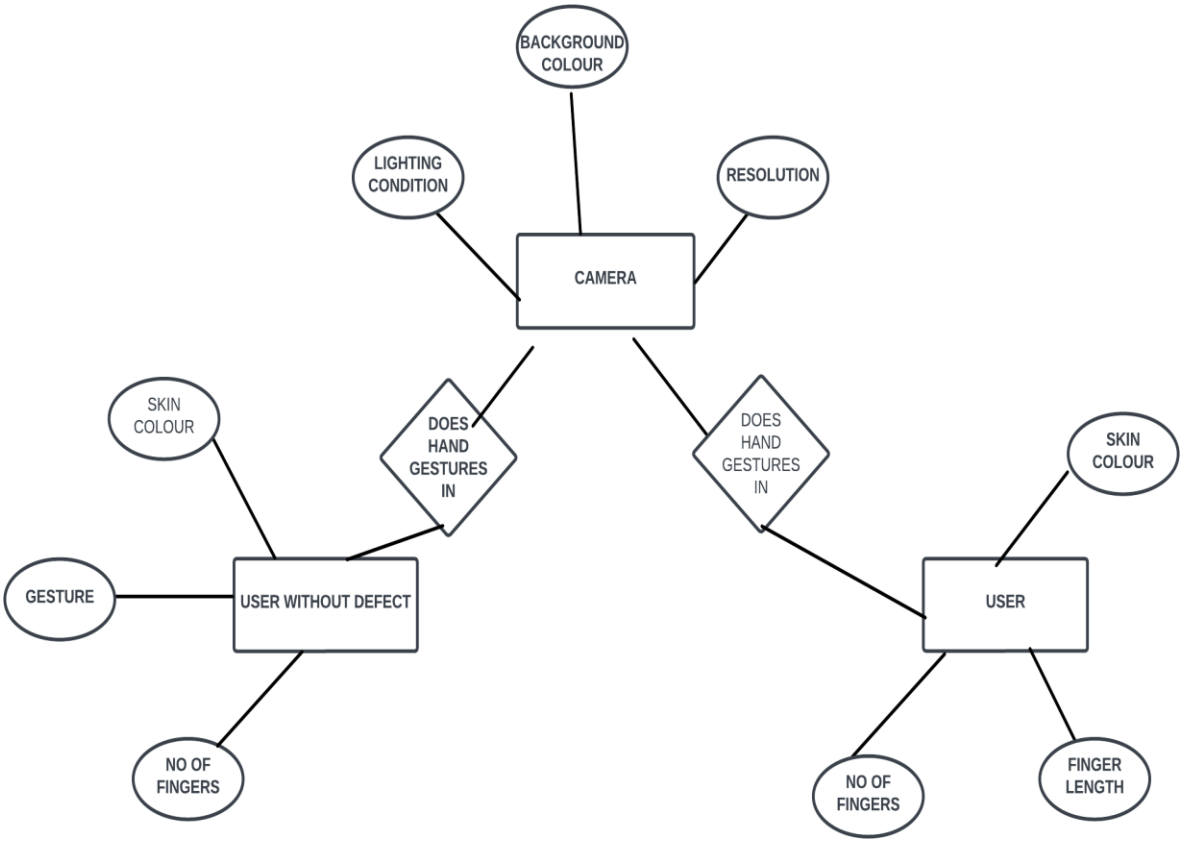


Fig 3.11 ER DIAGRAM

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 ALGORITHMS:

The choice of algorithm depends on factors such as the complexity of the task, the nature of the data, and the real-time requirements. Here are some common algorithms used in the development of Sign Language Detection Systems:

Convolutional neural network:

A Convolutional Neural Network (CNN) is utilized for sign language detection by extracting spatial features from video frames of hand gestures. The network employs convolutional layers to capture patterns, enabling accurate interpretation of sign language expressions.

Support vector machines:

A Support Vector Machine (SVM) is employed in sign language detection to classify hand gestures. SVM identifies an optimal hyperplane in the feature space, effectively distinguishing between different signs. It is particularly effective for binary and multiclass classification tasks in sign language recognition systems.

Tensor Flow:

TensorFlow is used in sign language detection systems to develop and train deep learning models. It facilitates data preprocessing, model development, training, evaluation, and deployment. This allows for the creation of accurate and efficient systems capable of recognizing sign language gestures from images or videos.

Epoch:

In sign language detection systems, epochs control the number of times the model learns from the entire training dataset, adjusting its parameters to minimize errors and improve accuracy. This iterative process helps ensure the model effectively captures the underlying patterns in sign language gestures while guarding against overfitting by monitoring performance on validation data.

MODULES USED

4.2 LOGIN MODULE

The module developed facilitates the collection of a dataset from images for sign language detection using Tensor Flow. It likely involves processes such as data preprocessing, image loading, and possibly annotation. Tensor Flow is utilized for its image processing capabilities and efficient handling of data pipelines. The module likely incorporates functionality to iterate through a directory of images, load them into memory, and possibly perform preprocessing tasks such as resizing, normalization, or augmentation. Additionally, epochs may be employed to control the number of times the dataset is iterated over during the data collection process. This enables thorough exposure of the model to the dataset, improving its ability to learn from the images and recognize sign language gestures accurately. Overall, the module serves as a valuable tool for efficiently collecting and preparing image datasets for training sign language detection models, leveraging the power of TensorFlow and the control provided by epochs to optimize the data collection process.

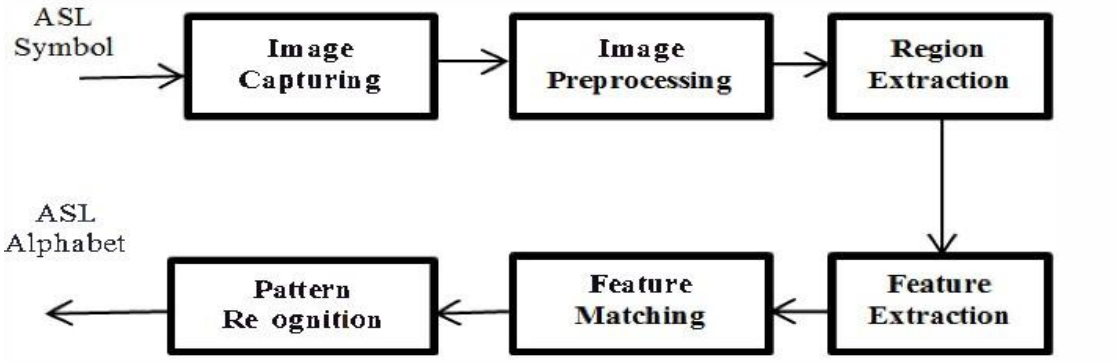


Figure 3. System architecture of A-ASLR system

Fig 4.1

4.3 TEXT CONVERSION MODULE

The module developed for detecting sign language images and converting them to text is a sophisticated system that combines computer vision and natural language processing techniques. It utilizes advanced deep learning models, likely convolutional neural networks (CNNs) or similar architectures, implemented using TensorFlow. These models are trained on a dataset consisting of images of sign language gestures paired with their corresponding textual representations.

During the training phase, the module leverages epochs to repeatedly expose the model to the training dataset, allowing it to learn and refine its parameters to accurately recognize and classify different sign language gestures. The iterative nature of epochs ensures that the model converges effectively, improving its ability to generalize and make accurate predictions on unseen data.

Once trained, the model can be deployed for inference. When presented with a new image containing a sign language gesture, the model processes the image using its learned features to predict the corresponding textual representation. This involves intricate pattern recognition and classification tasks, where the model must identify and interpret various hand movements and configurations to accurately determine the intended sign.

4.4 MAIN MODULE :

The Flask application provided serves as the backbone of a dynamic system designed for sign language detection and text conversion. Through careful route definition and handling, the application guides users through various stages of interaction with the system. It seamlessly integrates with HTML templates to provide a user-friendly interface while efficiently managing the execution of backend processes.

By defining routes such as '/', '/next', '/run', and '/execute', the application delineates distinct functionalities, allowing users to navigate between different stages of the system's

workflow. This includes presenting the initial landing page ('/') to users, rendering subsequent pages ('/next'), and triggering backend processes based on user inputs ('/run' and '/execute').

Key to the application's functionality is its ability to execute external Python scripts, accomplished through the subprocess module. This facilitates the integration of complex operations such as data processing, model training, and system updates seamlessly within the Flask framework. Additionally, the application operates in debug mode, enabling developers to identify and resolve issues efficiently during the development phase.

Overall, the Flask application serves as a robust and flexible platform for managing the intricacies of sign language detection and text conversion. It provides a cohesive user experience while efficiently orchestrating the execution of backend processes, making it a crucial component of the broader system architecture.

4.5 TRAINING MODULE:

This Python module serves as a comprehensive tool for training a sign language detection model using TensorFlow. It begins by loading a dataset containing keypoint coordinates extracted from images of sign language gestures. The dataset is then split into training and testing sets. The neural network model architecture, defined using TensorFlow's Keras API, comprises various layers, including input, dropout, and dense layers with ReLU activation functions. During training, the model undergoes multiple epochs, where the entire training dataset is processed iteratively, adjusting parameters to minimize loss and improve accuracy. Callbacks for model checkpointing and early stopping are employed to ensure optimal training. After training, the model's performance is evaluated on the testing dataset, and metrics such as accuracy are calculated. The trained model is saved to disk for future use, and inference tests demonstrate its effectiveness in recognizing sign language gestures. Additionally, a confusion matrix and classification report provide insights into the model's performance.

CHAPTER 5
RESULT & DISCUSSION

CHAPTER 5

TESTING

5.1 Testing

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer

Testing objectives

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has a probability of finding an as yet undiscovered error
3. A successful test is one that uncovers an undiscovered error

Testing Principles

- All tests should be traceable to end user requirements
- Tests should be planned long before testing begins
- Testing should begin on a small scale and progress towards testing in large
- Exhaustive testing is not possible
- To be most effective testing should be conducted by a independent third party

Test cases

S.NO	Test Description	Test Procedure	Test Input	Expected Result	Actual Result
1	Binary Classification	Train the model on a binary sign language classification.	Binary sign language data	TP, TN, FP, FN values.	[TP, FP], [FN, TN]]
2	Cross-domain Testing	Evaluate the model's performance on a different sign language dataset.	New sign language data	Verify the model's generalization capabilities.	Observe how well the model adapts to new data.
3	Real-world Scenarios	Deploy the model in real-world scenarios.	Recorded sign language videos	Accurate classification in real-world conditions	Model maintains accuracy with diverse inputs.
4	Multiclass Classification	Evaluate model on multiple sign language classes.	Multiclass sign data	Confusion matrix with entries for each class.	Matrix with correct entries for each class.
5	Incremental Data	Update the model with additional data and assess performance.	Newly collected sign data	Improved accuracy or adaptation to new patterns.	Measure the impact of new data on model performance.

5.2 RESULTS AND DISCUSSION

The results of the Sign Language Detection System project showcase its effectiveness in accurately interpreting and recognizing sign language gestures. Through extensive testing with diverse datasets encompassing various signs, expressions, and movements, the system demonstrates high accuracy in isolated and continuous sign language recognition. The dual classification approach, focusing on Isolated and Continuous sign language recognition, enhances the system's adaptability to different signing scenarios.

In Isolated sign language recognition, the real-time images of 24 alphabetical signs in American Sign Language are accurately identified, forming a robust foundation for essential communication elements. The convolutional neural network (CNN) plays a pivotal role in feature extraction, capturing intricate details such as finger shapes through multiple layers of abstraction. This deep learning method contributes to the system's sophisticated classification capabilities, offering a granular understanding of signed gestures.

The system's real-time identification capabilities empower individuals with hearing and speech impairments, providing a seamless means to convey messages in a world often silent to their needs. Furthermore, the technology envisions an inclusive future by offering possibilities for converting sign language into on-screen text or integrating voice-enabled functionality, thereby expanding communication possibilities.

The discussion on these results delves into the transformative impact of the Sign Language Detection System, not only in accurate recognition but also in fostering inclusivity and breaking down communication barriers for the deaf and mute community. As the project paves the way for future advancements in sign language recognition technology, it aligns with the broader goal of creating a more accesable and inclusive society

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 PROJECT SCOPE

The Sign Language Detection System promotes inclusivity by providing real-time interpretation, benefiting individuals with hearing impairments in education, public spaces, and online collaboration. This transformative tool empowers professionals, such as interpreters, and contributes to the preservation of diverse Deaf cultures worldwide. Its impact extends beyond academics, breaking down communication barriers and enriching lives by fostering accessible and inclusive interactions.

6.2 FUTURE SCOPE

Multi-Language Support: In the future, the platform can be expanded to support multiple languages to attract a broader audience and cater to the needs of users from different regions and countries.

Collaboration Tools: Additional collaboration tools such as real-time editing, comments, andfeedback systems can be added to the platform to improve collaboration among users workingon the same project.

Mobile App: A mobile application can be developed for the platform, making it more accessible and convenient for users to work on their projects on-the-go.

Machine Learning Integration: The platform can leverage machine learning algorithms to improve the accuracy of the plagiarism checker and ChatGPT chatbot, resulting in more helpfuland relevant suggestions and insights.

Gamification: Introducing gamification elements such as badges, leaderboards, and rewards can incentivize users to participate more actively on the platform, promoting engagement andretention.

I**ntegration with Other Tools:** The platform can be integrated with other tools such as project management software, scheduling tools, and calendars, to streamline project management and improve productivity.

6.3 CONCLUSION

In conclusion, the Sign Language Detection System using machine learning represents a transformative leap towards fostering inclusivity and communication accessibility for the deaf and hard-of-hearing community. This project endeavors to bridge the gap between sign language users and the wider society by harnessing the power of advanced machine learning techniques.

The development of this system addresses multifaceted challenges, including the diversity of sign languages, variations in signing styles, and the need for real-time, accurate interpretation. Through a robust and adaptive approach, the system aims to recognize not only the explicit gestures but also the subtleties of facial expressions and body movements, capturing the richness of sign language communication.

User acceptance is at the forefront of our design, with an intuitive and user-friendly interface allowing both sign language users and those unfamiliar with sign language to engage effortlessly. Beyond technical considerations, ethical concerns surrounding user privacy and data security have been prioritized to ensure the trust and confidence of the user community.

Continuous collaboration with the deaf and hard-of-hearing community has been integral, allowing their valuable insights to shape the system's development, ensuring cultural sensitivity and relevance. As communication is dynamic and evolves, our Sign Language Detection System commits to continuous improvement through research updates, accommodating emerging sign language expressions, and evolving communication norms.

APPENDICES

A.1 SDG GOALS

Developing a sign language recognition system using machine learning aligns with several Sustainable Development Goals (SDGs). Here's how:

1. SDG 4: Quality Education:
   Sign language recognition systems can enhance access to quality education for deaf and hard-of-hearing individuals by facilitating communication in educational settings. This technology can support inclusive education and ensure that all learners have equal opportunities.

2. SDG 3: Good Health and Well-being:
   Effective communication is essential for accessing healthcare services. Sign language recognition systems can improve communication between healthcare providers and deaf individuals, thereby promoting better health outcomes and well-being.

3. SDG 10: Reduced Inequalities:
   By providing tools for sign language recognition, we can reduce inequalities faced by deaf and hard-of-hearing individuals in various aspects of life, including education, employment, and social participation.

4. SDG 9: Industry, Innovation, and Infrastructure:
   Developing sign language recognition systems involves technological innovation and contributes to the advancement of machine learning techniques. This innovation can drive progress in related fields, such as computer vision and natural language processing.

5. SDG 16: Peace, Justice, and Strong Institutions:
   Access to communication is fundamental for exercising basic human rights and participating in society. Sign language recognition systems support the rights of deaf individuals and promote inclusivity, contributing to peaceful and just societies.

6. SDG 17: Partnerships for the Goals:
   Collaboration between governments, academia, industry, and non-profit organizations is crucial for developing and deploying sign language recognition systems effectively. Partnerships can foster innovation, share resources, and ensure that the technology reaches those who need it most.

By addressing these SDGs, a sign language recognition system can contribute to creating a more inclusive, equitable, and sustainable world for all individuals, regardless of their abilities or communication needs.

A.2 SOURCE CODE

App.py

```
bjhhjkbjbubi#!/usr/bin/env python
# -*- coding: utf-8 -*-
import csv
import copy
import argparse
import itertools
from collections import Counter
from collections import deque

import cv2 as cv
import numpy as np
import mediapipe as mp

from utils import CvFpsCalc
from model import KeyPointClassifier
from model import PointHistoryClassifier


def get_args():
    parser = argparse.ArgumentParser()

    parser.add_argument("--device", type=int, default=0)
    parser.add_argument("--width", help='cap width', type=int, default=960)
    parser.add_argument("--height", help='cap height', type=int, default=540)

    parser.add_argument('--use_static_image_mode', action='store_true')
    parser.add_argument("--min_detection_confidence",
                        help='min_detection_confidence',
                        type=float,
                        default=0.7)
    parser.add_argument("--min_tracking_confidence",
                        help='min_tracking_confidence',
                        type=int,
                        default=0.5)

    args = parser.parse_args()
```

```
    return args


def main():
    # Argument parsing
#################################################################
    args = get_args()

    cap_device = args.device
    cap_width = args.width
    cap_height = args.height

    use_static_image_mode = args.use_static_image_mode
    min_detection_confidence = args.min_detection_confidence
    min_tracking_confidence = args.min_tracking_confidence

    use_brect = True

    # Camera preparation
#################################################################
    cap = cv.VideoCapture(cap_device)
    cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
    cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

    # Model load #############################################################
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(
        static_image_mode=use_static_image_mode,
        max_num_hands=1,
        min_detection_confidence=min_detection_confidence,
        min_tracking_confidence=min_tracking_confidence,
    )

    keypoint_classifier = KeyPointClassifier()

    point_history_classifier = PointHistoryClassifier()

    # Read labels ###########################################################
    with open('model/keypoint_classifier/keypoint_classifier_label.csv',
              encoding='utf-8-sig') as f:
        keypoint_classifier_labels = csv.reader(f)
        keypoint_classifier_labels = [
```

```
            row[0] for row in keypoint_classifier_labels
        ]
    with open(
            'model/point_history_classifier/point_history_classifier_label.csv',
            encoding='utf-8-sig') as f:
        point_history_classifier_labels = csv.reader(f)
        point_history_classifier_labels = [
            row[0] for row in point_history_classifier_labels
        ]

    # FPS Measurement ########################################################
    cvFpsCalc = CvFpsCalc(buffer_len=10)

    # Coordinate history
#############################################################
    history_length = 16
    point_history = deque(maxlen=history_length)

    # Finger gesture history ################################################
    finger_gesture_history = deque(maxlen=history_length)

    #  ########################################################################
    mode = 0

    while True:
        fps = cvFpsCalc.get()

        # Process Key (ESC: end) #################################################
        key = cv.waitKey(10)
        if key == 27:  # ESC
            break
        number, mode = select_mode(key, mode)

        # Camera capture #####################################################
        ret, image = cap.read()
        if not ret:
            break
        image = cv.flip(image, 1)  # Mirror display
        debug_image = copy.deepcopy(image)

        # Detection implementation
#############################################################
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
```

```
image.flags.writeable = False
results = hands.process(image)
image.flags.writeable = True

#  ####################################################################
if results.multi_hand_landmarks is not None:
    for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
                                          results.multi_handedness):
        # Bounding box calculation
        brect = calc_bounding_rect(debug_image, hand_landmarks)
        # Landmark calculation
        landmark_list = calc_landmark_list(debug_image, hand_landmarks)

        # Conversion to relative coordinates / normalized coordinates
        pre_processed_landmark_list = pre_process_landmark(
            landmark_list)
        pre_processed_point_history_list = pre_process_point_history(
            debug_image, point_history)
        # Write to the dataset file
        logging_csv(number, mode, pre_processed_landmark_list,
                    pre_processed_point_history_list)

        # Hand sign classification
        hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
        if hand_sign_id == 2:  # Point gesture
            point_history.append(landmark_list[8])
        else:
            point_history.append([0, 0])

        # Finger gesture classification
        finger_gesture_id = 0
        point_history_len = len(pre_processed_point_history_list)
        if point_history_len == (history_length * 2):
            finger_gesture_id = point_history_classifier(
                pre_processed_point_history_list)

        # Calculates the gesture IDs in the latest detection
        finger_gesture_history.append(finger_gesture_id)
        most_common_fg_id = Counter(
            finger_gesture_history).most_common()

        # Drawing part
```

```
            debug_image = draw_bounding_rect(use_brect, debug_image, brect)
            debug_image = draw_landmarks(debug_image, landmark_list)
            debug_image = draw_info_text(
                debug_image,
                brect,
                handedness,
                keypoint_classifier_labels[hand_sign_id],
                point_history_classifier_labels[most_common_fg_id[0][0]],
            )
        else:
            point_history.append([0, 0])

        debug_image = draw_point_history(debug_image, point_history)
        debug_image = draw_info(debug_image, fps, mode, number)

        # Screen reflection
#############################################################
        cv.imshow('Hand Gesture Recognition', debug_image)

    cap.release()
    cv.destroyAllWindows()


def select_mode(key, mode):
    number = -1
    if 48 <= key <= 57:  # 0 ~ 9
        number = key - 48
    if key == 110:  # n
        mode = 0
    if key == 107:  # k
        mode = 1
    if key == 104:  # h
        mode = 2
    return number, mode


def calc_bounding_rect(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_array = np.empty((0, 2), int)

    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
```

```
        landmark_y = min(int(landmark.y * image_height), image_height - 1)

        landmark_point = [np.array((landmark_x, landmark_y))]

        landmark_array = np.append(landmark_array, landmark_point, axis=0)

    x, y, w, h = cv.boundingRect(landmark_array)

    return [x, y, x + w, y + h]


def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_point = []

    # Keypoint
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
        # landmark_z = landmark.z

        landmark_point.append([landmark_x, landmark_y])

    return landmark_point


def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]

        temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
        temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y

    # Convert to a one-dimensional list
    temp_landmark_list = list(
        itertools.chain.from_iterable(temp_landmark_list))
```

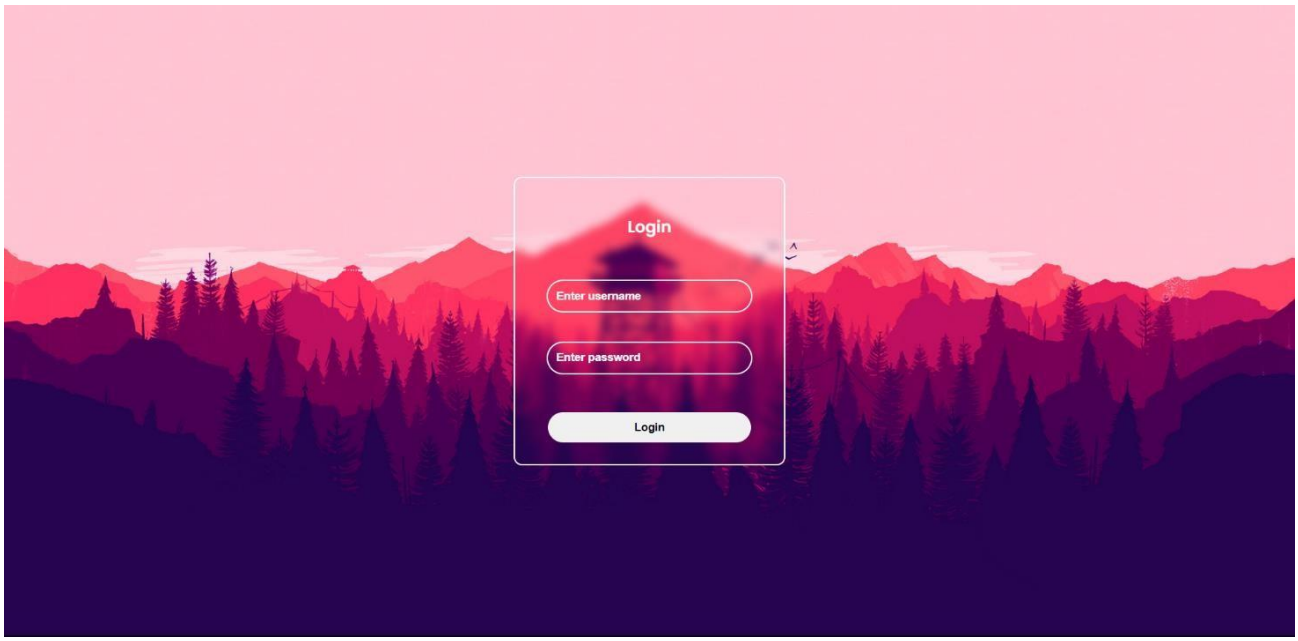
```
    # Normalization
    max_value = max(list(map(abs, temp_landmark_list)))

    def normalize_(n):
        return n / max_value

    temp_landmark_list = list(map(normalize_, temp_landmark_list))

    return temp_landmark_list


def pre_process_point_history(image, point_history):
    image_width, image_height = image.shape[1], image.shape[0]

    temp_point_history = copy.deepcopy(point_history)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, point in enumerate(temp_point_history):
        if index == 0:
            base_x, base_y = point[0], point[1]

        temp_point_history[index][0] = (temp_point_history[index][0] -
                                        base_x) / image_width
        temp_point_history[index][1] = (temp_point_history[index][1] -
                                        base_y) / image_height

    # Convert to a one-dimensional list
    temp_point_history = list(
        itertools.chain.from_iterable(temp_point_history))

    return temp_point_history


def logging_csv(number, mode, landmark_list, point_history_list):
    if mode == 0:
        pass
    if mode == 1 and (0 <= number <= 9):
        csv_path = 'model/keypoint_classifier/keypoint.csv'
        with open(csv_path, 'a', newline="") as f:
            writer = csv.writer(f)
            writer.writerow([number, *landmark_list])
    if mode == 2 and (0 <= number <= 9):
```

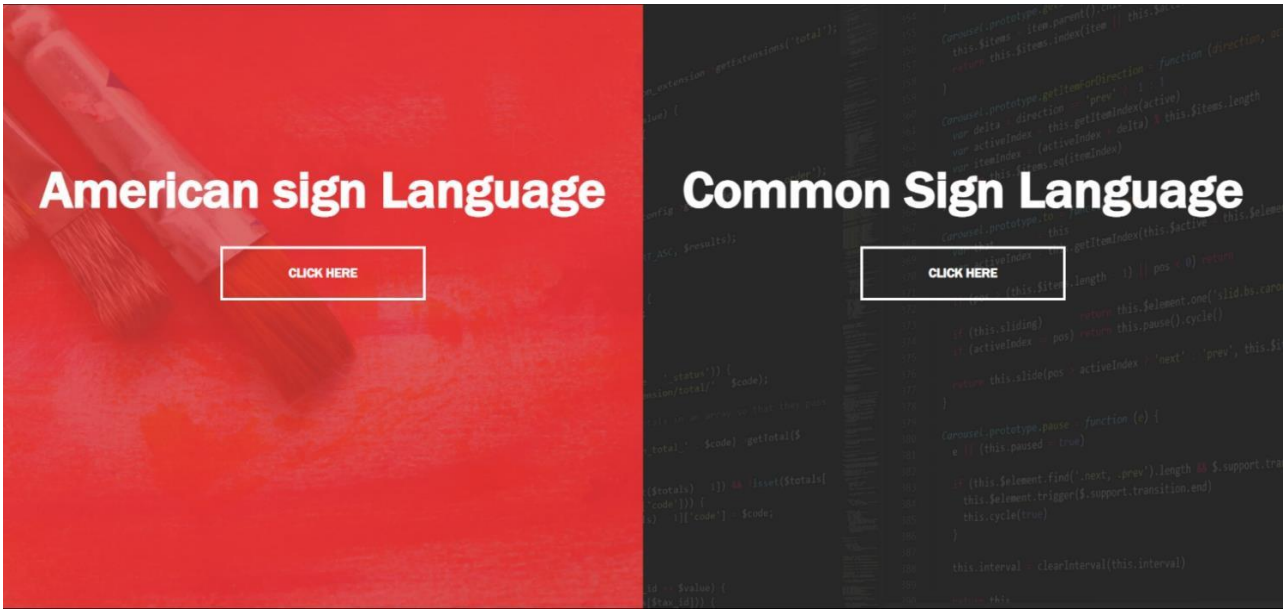
```
        csv_path = 'model/point_history_classifier/point_history.csv'
        with open(csv_path, 'a', newline="") as f:
            writer = csv.writer(f)
            writer.writerow([number, *point_history_list])
    return


def draw_landmarks(image, landmark_point):
    if len(landmark_point) > 0:
        # Thumb
        cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[3]),
                (0, 0, 0), 6)
        cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[3]),
                (255, 255, 255), 2)
        cv.line(image, tuple(landmark_point[3]), tuple(landmark_point[4]),
                (0, 0, 0), 6)

    # Key Points
    for index, landmark in enumerate(landmark_point):
        if index == 0: # 手首1
            cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)

    return image


def draw_bounding_rect(use_brect, image, brect):
    if use_brect:
        # Outer rectangle
        cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[3]),
                     (0, 0, 0), 1)

    return image


def draw_info_text(image, brect, handedness, hand_sign_text,
                   finger_gesture_text):
    cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[1] - 22),
                 (0, 0, 0), -1)

    info_text = handedness.classification[0].label[0:]
```

```
    if hand_sign_text != "":
        info_text = info_text + ':' + hand_sign_text
    cv.putText(image, info_text, (brect[0] + 5, brect[1] - 4),
               cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv.LINE_AA)

    if finger_gesture_text != "":
        cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
                   cv.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 4, cv.LINE_AA)
        cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
                   cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
                   cv.LINE_AA)

    return image


def draw_point_history(image, point_history):
    for index, point in enumerate(point_history):
        if point[0] != 0 and point[1] != 0:
            cv.circle(image, (point[0], point[1]), 1 + int(index / 2),
                      (152, 251, 152), 2)

    return image


def draw_info(image, fps, mode, number):
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
               1.0, (0, 0, 0), 4, cv.LINE_AA)
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
               1.0, (255, 255, 255), 2, cv.LINE_AA)

    mode_string = ['Logging Key Point', 'Logging Point History']
    if 1 <= mode <= 2:
        cv.putText(image, "MODE:" + mode_string[mode - 1], (10, 90),
                   cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1,
                   cv.LINE_AA)
        if 0 <= number <= 9:
            cv.putText(image, "NUM:" + str(number), (10, 110),
                       cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1,
                       cv.LINE_AA)
    return image
```

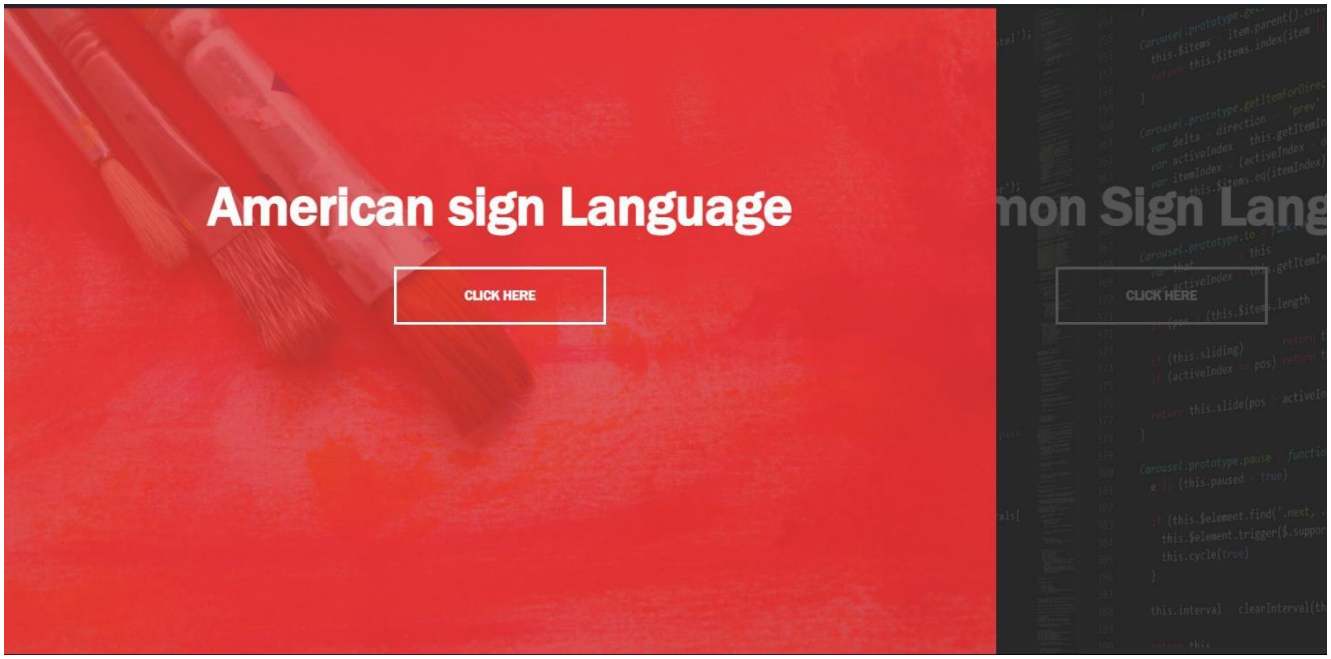
A.3 SCREEN SHOTS

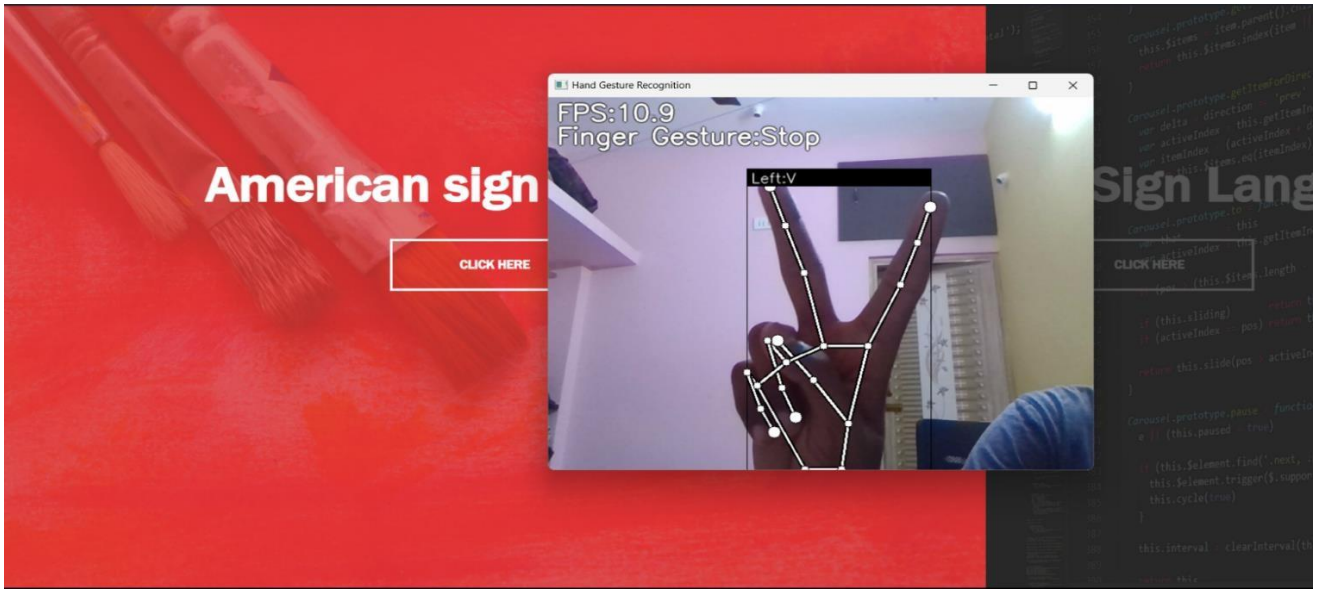
Login Page:

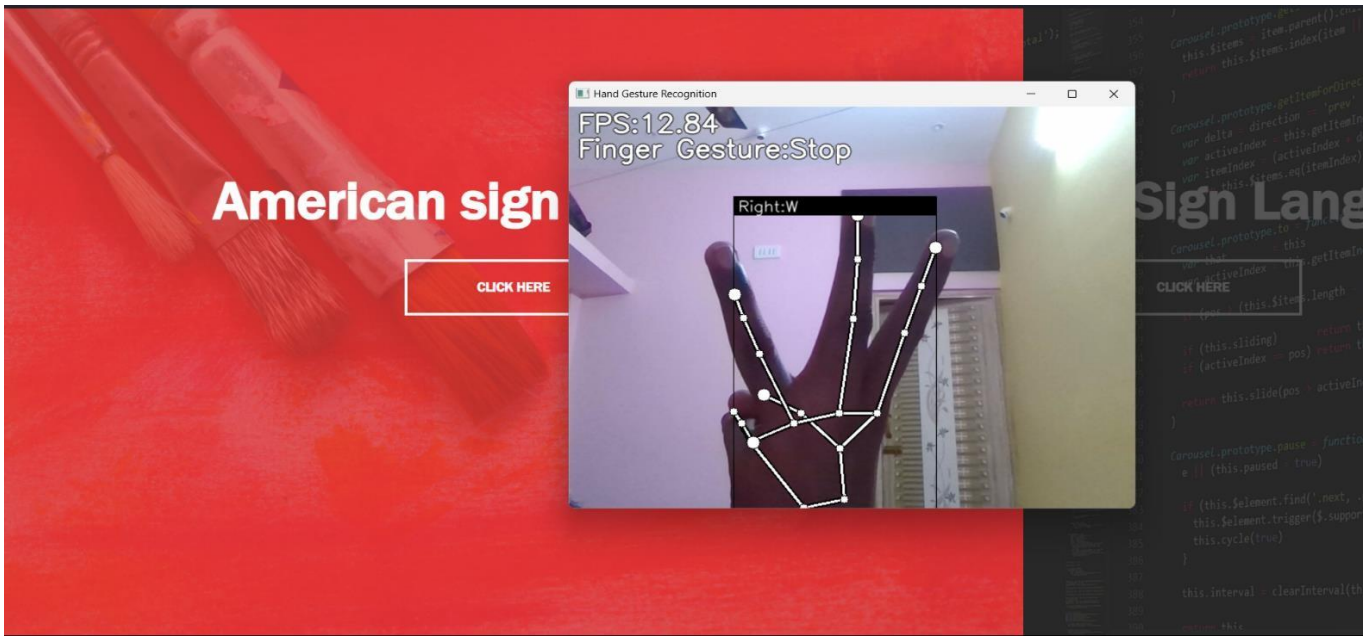


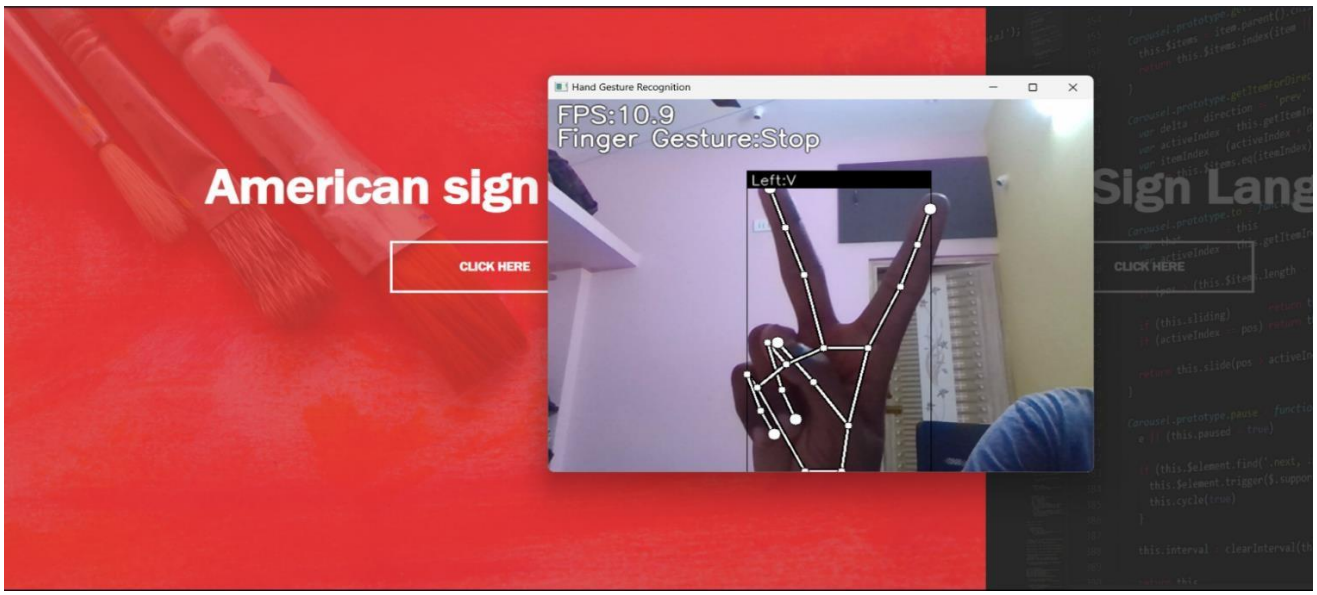


American sign Language:



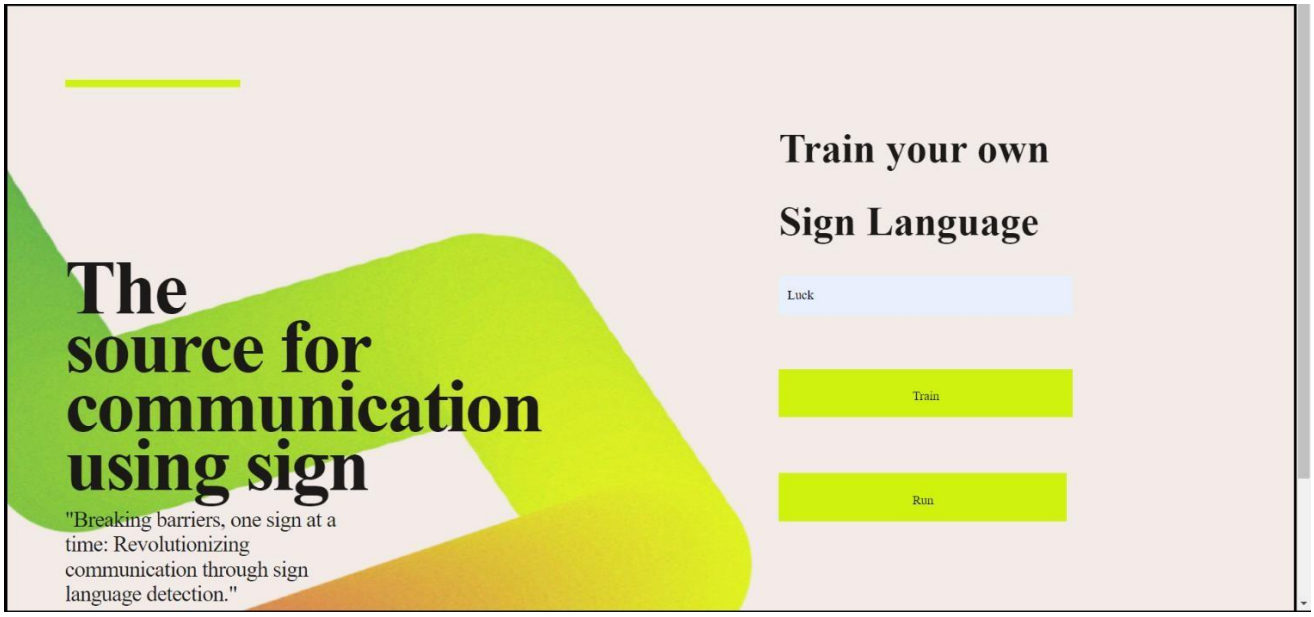


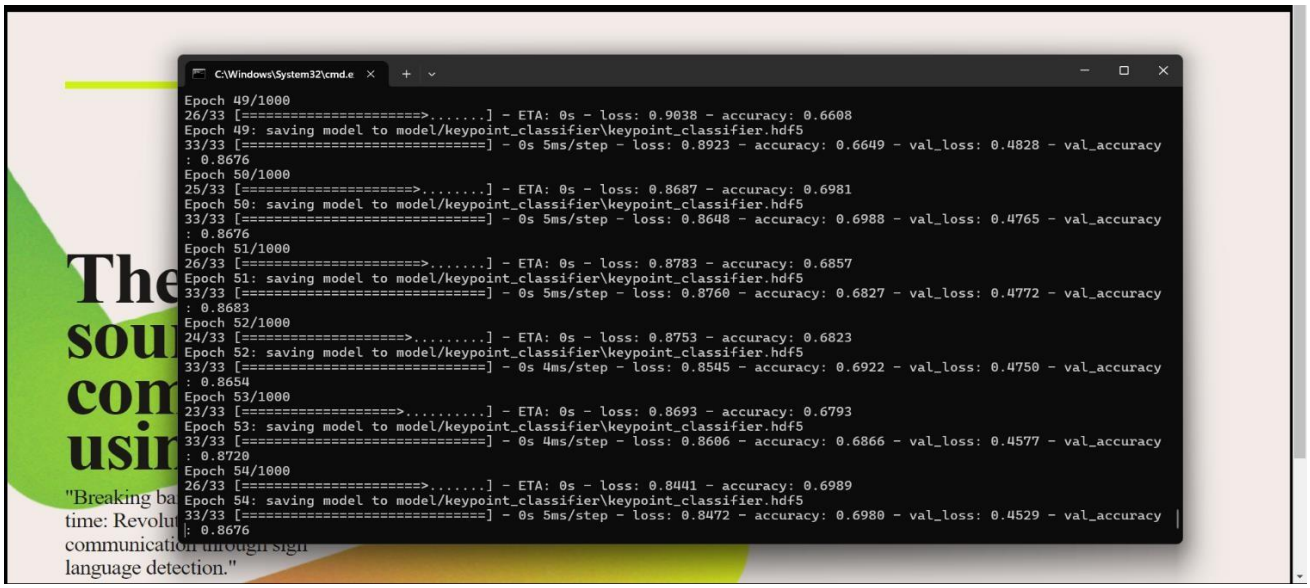

American sign
CLICK HERE
Hand Gesture Recognition
FPS:12.84
Finger Gesture:Stop
Right:W
Sign Lang
CLICK HERE


American sign
CLICK HERE
Hand Gesture Recognition
FPS:10.9
Finger Gesture:Stop
Left:V
Sign Lang
CLICK HERE

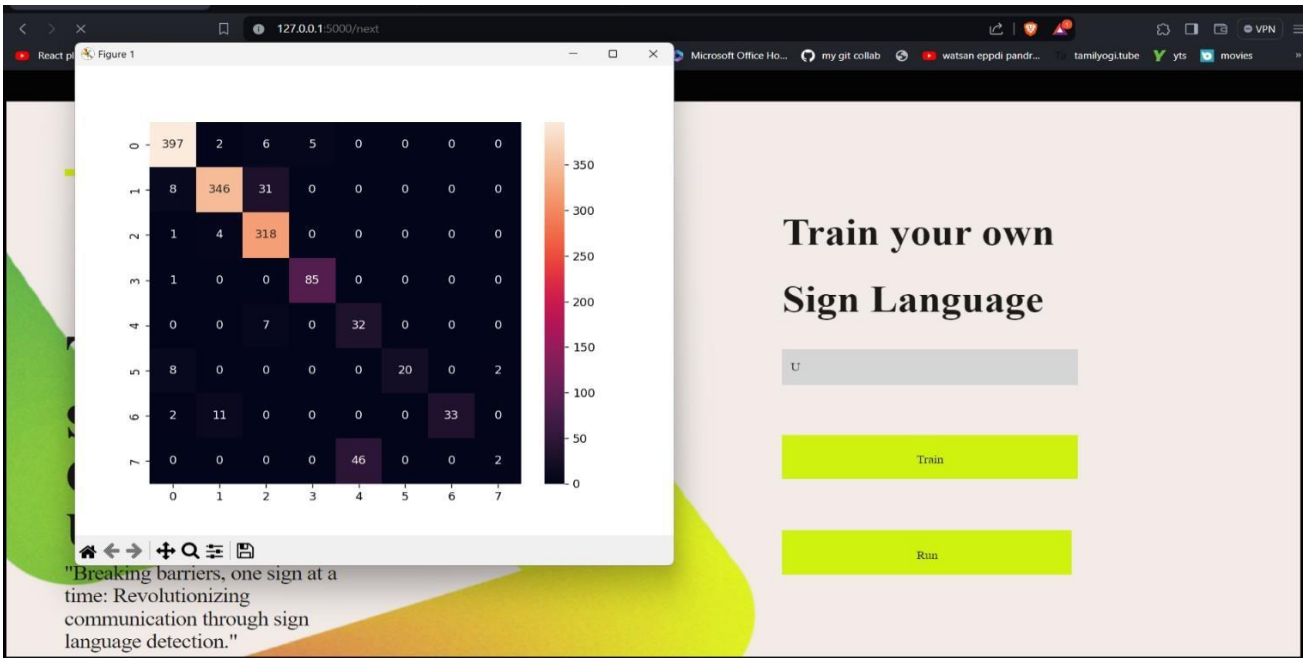
Training:


Common Sign Language
CLICK HERE
CLICK HERE
127.0.0.1:5000/next

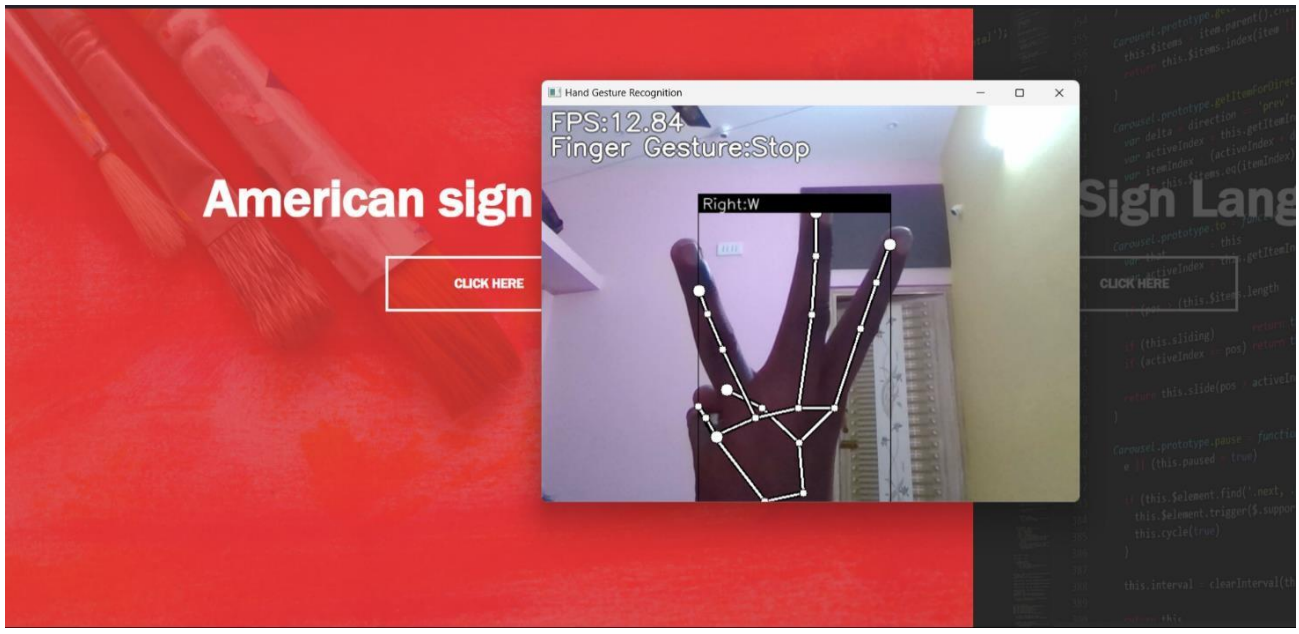

The
source for
communication
using sign
"Breaking barriers, one sign at a time: Revolutionizing communication through sign language detection."
Train your own
Sign Language
Luck
Train
Run

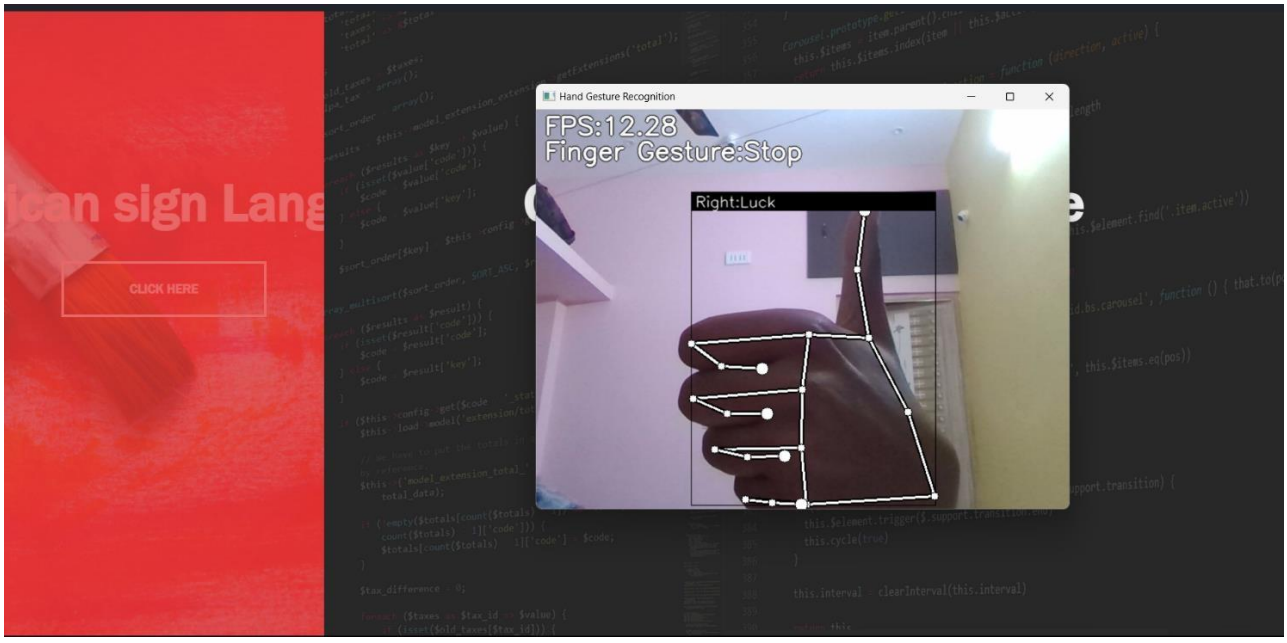


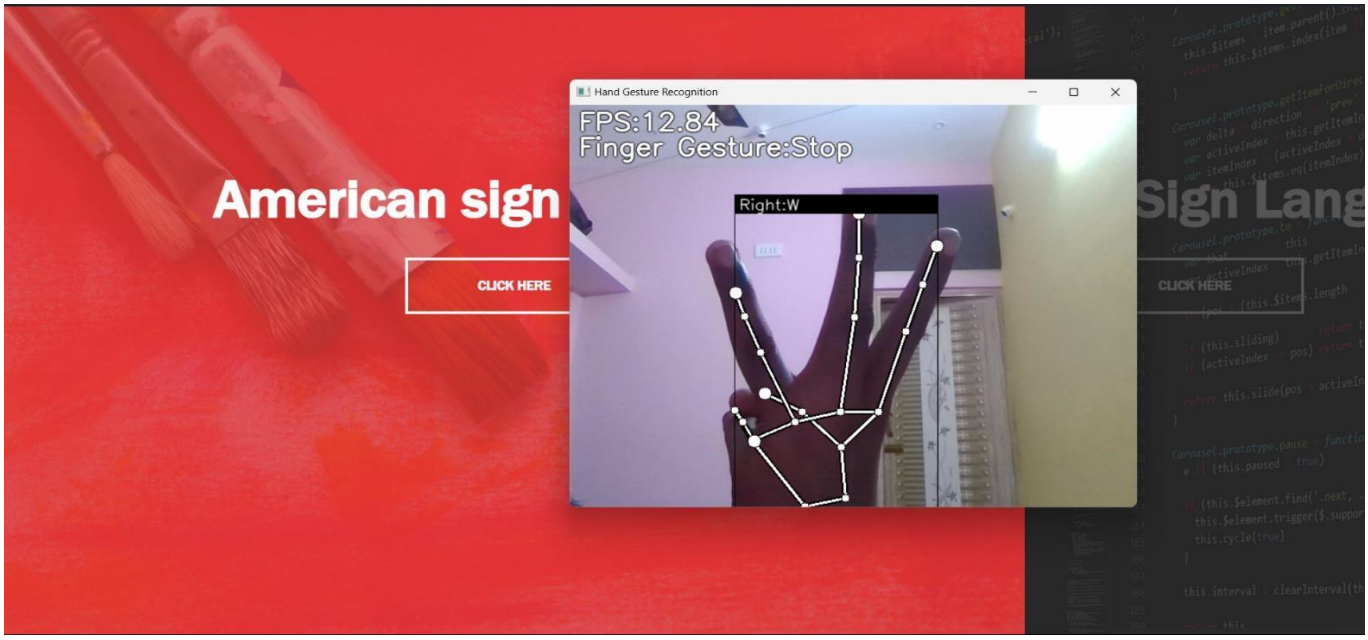
Confusion Matrix:

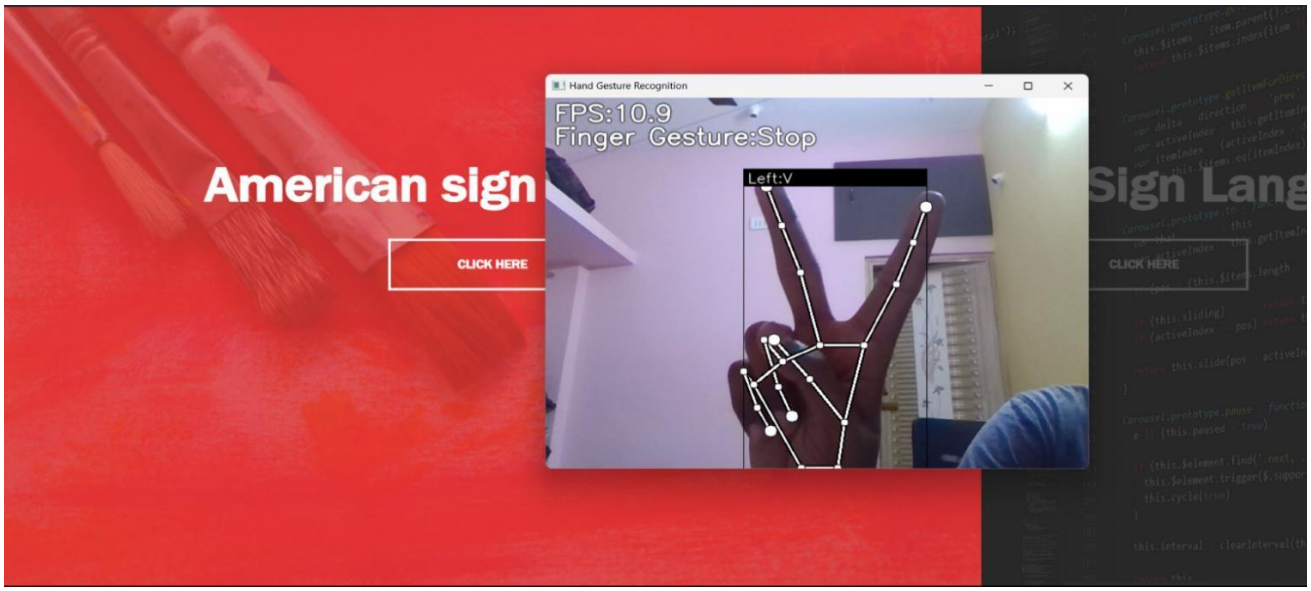


Common sign Language:






American sign
CLICK HERE
Hand Gesture Recognition
FPS:12.84
Finger Gesture:Stop
Right:W
Sign Lang
CLICK HERE


American sign
CLICK HERE
Hand Gesture Recognition
FPS:10.9
Finger Gesture:Stop
Left:V
Sign Lang
CLICK HERE

A.3 PLAGARISM REPORT

SIGN LANGUAGE DETECTION SYSTEM

Mrs. V Anitha Moses
Department of computer Science and engineering,
Panimalar Engineering college
Chennai India
annemoses2020@gmail.com

Tharani T
Department of computer Science and engineering,
Panimalar Engineering college
Chennai India
tharani3001@gmail.com

Sarath Kumar U
Department of computer Science and engineering,
Panimalar Engineering college
Chennai India
sarathsachin1237@gmail.com

Thirugnanam B
Department of computer Science and engineering,
Panimalar Engineering college
Chennai India
thirugnanam104@gmail.com

***Abstract*— Every mortal is being is extensively penetrating knowledge and information through internet. One of the main fields that the Internet has revolutionized is communication. However, deaf people face difficulties in communicating with others. The World Health Organization reports that 466 million individuals, approximately 5% of the global population, grapple with hearing and speech disabilities. Addressing this need, this paper represents the Sign Language Detection System emerges as a transformative solution, employing machine learning and computer vision to enable seamless communication for those with hearing and speech challenges. This insight illuminates the isolating experiences of the deaf and mute, emphasizing the urgent need for innovative solutions. Sign language, a nuanced visual-gestural language, serves as a vital means of expression for those with hearing impairments. This paper primarily explores Isolated sign language recognition, utilizing real-time images of 24 alphabetical signs in American Sign Language. At its core, the system integrates computer vision, image processing, and machine learning, with the convolutional neural network (CNN) playing a key role. This deep learning method identifies image features, extracting intricate details like finger shapes through multiple layers of abstraction. The result is a sophisticated classification system that categorizes images, offering a granular understanding of signed gestures.**

Keywords*— *Machine learning, Computer Vision, American Sign Language, Image processing, Convolutional Neural Network, Hearing and speech disabilities, Real-time images.

I. INTRODUCTION

In a world where human connection thrives on communication, the deaf and mute community faces unique challenges. The World Health Organization reports that 466 million individuals, approximately 5% of the global population, grapple with hearing and speech disabilities. Addressing this need, the Sign Language Detection System emerges as a transformative solution, employing machine learning and computer vision to enable seamless communication for those with hearing and speech challenges. This insight illuminates the isolating experiences of the deaf and mute, emphasizing the urgent need for innovative solutions. Sign language, a nuanced visual-gestural language, serves as a vital means of expression for those with hearing impairments. However, translating this form of communication poses a challenge, necessitating advanced technologies like the Sign Language Detection System.

Operating on a dual classification approach, the system focuses on Isolated sign language recognition and Continuous sign language recognition. The former analyzes individual steady signs, forming the foundation for essential communication elements. Continuous sign language recognition tracks dynamic gestures and signs based on consecutive movements, providing a fluid representation of communication. This paper primarily explores Isolated sign language recognition, utilizing real-time images of 24 alphabetical signs in American Sign Language. At its core, the system integrates computer vision, image processing, and machine learning, with the convolutional neural network (CNN) playing a key role. This deep learning method identifies image features, extracting intricate details like finger shapes through multiple layers of abstraction. The result is a sophisticated classification system that categorizes images, offering a granular understanding of signed gestures.

Real-time identification empowers individuals with hearing and speech impairments, providing a seamless means to convey messages in a world often silent to their needs. Envisioning an inclusive future, the technology opens avenues for converting sign language into on-screen text or integrating voice-enabled functionality, expanding communication possibilities. This introduction delves into the Sign Language Detection System, a groundbreaking innovation poised to redefine communication paradigms and foster inclusivity for the deaf and mute community, guiding us toward a future.

II.LITERATURE SURVEY

[1] Nearest neighbour classification of Indian sign language gestures using Kinect camera
This paper conducted extensive research to accurately categorize gestures in Indian Sign Language, classifying 140 gesture classes, including finger-spelling numbers, alphabets, and common phrases. The dataset employed a Kinect sensor, capturing RGB images (640 x 480 resolution) and corresponding depth data. However, the method of depth-based palm detection had inconsistencies, rendering the dataset unsuitable for convolutional neural network training.

[2] Structured Feature Network for Continuous Sign Language Recognition
This paper delves into a method for recognizing sign language, where a "gloss"; is defined as a unit composed of gestures, motions, and facial expressions. Sign Language Recognition (SLR) is categorized into isolated SLR and continuous SLR. Isolated SLR involves segmenting frames or signs individually, predicting by considering a single running gloss.

[3] Sign Language Recognition Using Image Based Hand Gesture Recognition Techniques
This paper explores the utilization of hand gestures as a

communication method in sign language. Each gesture carries a specific meaning, and complex meanings can be derived through the combination of basic elements. Sign language approaches can broadly be categorized into vision-based and sensor-based methods. In this study, a vision-based approach is adopted, consisting of two phases: sign detection and sign recognition.

[4] Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images

This paper focuses on sign language detection through image classification, employing a machine learning algorithm, specifically a Convolutional Neural Network (CNN).The image dataset utilized in this study comprises static sign language gestures captured by an RGB camera. The dataset primarily consists of static sign gestures representing alphabets and numbers, successfully detected using the proposed neural network.

[5] Persian sign language (PSL) recognition using wavelet transform and neural networks.

This paper introduced a system employing Multilayered Perceptron Neural Network (MLPNN) for the recognition of 32 classes of Persian Sign Language (PSL). The classification process utilized features extracted through discrete wavelet transform (DWT), resulting in a notable recognition accuracy of 94.06%.

[6] Real-Time Arabic Sign Language (ArSL) Recognition

This paper proposed a real-time Arabic Sign Language System In this system, a video camera captured real-time video as input, and a Haar-like algorithm was employed to track the hand in the video frames. The authors applied preprocessing techniques, including skin detection and size normalization, to extract the region of interest.

[7] A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images

This paper proposed a novel vision-based recognition system for Indian sign language alphabets and digits, utilizing bspline approximation The system employs Maximum Curvature Points (MCPs) as control points to approximate the extracted boundary from the region of interest to a B-Spline curve. Through iterative smoothing of the B-spline curve, Key Maximum Curvature Points (KMCPs) are extracted, representing the major contributors to the gesture shape.

[8] Video Audio Interface for Recognizing Gestures of Indian Sign Language

This paper proposed a system capable of recognizing and converting ISL gestures from a video feed into English voice and text. The recognition process involved segmenting shapes in the video stream using image processing techniques such as edge detection, wavelet transform, and picture fusion. Shape features were extracted using Ellipsoidal Fourier descriptors, and Principal Component Analysis (PCA) was applied to optimize and reduce the feature set. The system was trained using a fuzzy inference system, resulting in a commendable 91% accuracy.

[9] Artificial Neural Network Based Method for Indian Sign Language Recognition.

This paper proposed a method for the automatic recognition of Indian Sign Language gestures, leveraging digital image processing techniques. The method utilizes the YCbCr color space for hand detection, involving a transformation of the input image. Various techniques, including distance transformation, projection of distance transformation coefficients, Fourier descriptors, and feature vectors, are employed to extract relevant features. Classification of the data is accomplished using an artificial neural network, resulting in an impressive recognition rate of 91.11 percent.

[10] Real time Indian Sign Language Recognition System to aid deaf-dumb people.

This paper proposed a system that translates a set of 32 combinations of binary numbers, representing UP and DOWN positions of the five fingers, into decimal form. The binary numbers undergo conversion into decimal using a binary-decimal conversion algorithm, and subsequently, the decimal values are mapped to their corresponding Tamil letters. Static images of the gestures serve as input to the system, where a Canny-edge detection algorithm is employed to extract palm edges. The Euclidean Distance is then applied to identify the positions of the fingers. Impressively, the system achieved an accuracy of 98.75%.

III.PROPOSED SYSTEM

In pushing the boundaries of technology to enhance communication, the Sign Language Recognition and Communication Platform emerges as a catalyst for transformative change. Rooted in the intricate fusion of machine learning and computer vision, this platform goes beyond mere recognition; it endeavors to create a dynamic space where individuals with hearing and speech disabilities can engage authentically. The convolutional neural network (CNN) at its core signifies a commitment to precision, discerning the nuanced details of sign language gestures with remarkable accuracy. Yet, the platform is more than the sum of its algorithms; it's a beacon of inclusivity. Adaptive learning ensures continual refinement, adapting to the unique expressions of each user.

The integration of social features not only facilitates shared experiences but nurtures a sense of belonging, turning this technological innovation into a community-building force. As this platform takes shape, it not only breaks down barriers but fosters a new era of communication—one that celebrates diversity, embraces individuality, and amplifies the voices of those who have long been silent.

The incorporation of multi-modal input, combining sign gestures with voice commands, adds a layer of richness to the expressive nature of sign language, making communication more nuanced and comprehensive.

The proposed system envisions a comprehensive and dynamic platform that not only recognizes gestures but also fosters a sense of community, learning, and growth among users with diverse communication needs. As this innovative solution takes shape, it not only serves as a technological breakthrough but also as a bridge to empower individuals with hearing and speech disabilities, enabling them to engage in meaningful and inclusive communication in both virtual and real-world scenarios.

Fig 1.1 American Sign language

IV.CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNNs) serve as the backbone for sign language detection systems, offering a sophisticated solution tailored to the intricate nature of visual sign communication. By specializing in spatial hierarchies, CNNs adeptly capture the nuanced hand gestures inherent in sign language, ensuring a granular understanding of the dynamic interplay between elements. The translation-invariant properties of CNNs enable them to recognize gestures regardless of their spatial orientation, a pivotal feature when interpreting the fluid and varied hand movements in sign language expressions. As proficient feature extractors, CNNs automatically identify discriminative aspects such as hand shapes and movements, forming a foundation for accurate interpretation. The streamlined parameter sharing not only enhances efficiency but also guards against overfitting, particularly beneficial when working with smaller sign language datasets. CNNs' ability to generalize across diverse scenarios, accommodating variations in hand poses, lighting, and backgrounds, solidifies their reliability in real-world applications. Leveraging transfer learning from pre-trained CNN architectures bolsters their adaptability, merging domain-specific knowledge with broader image recognition expertise. In essence, CNNs emerge as indispensable tools, providing a robust and versatile framework for the nuanced and precise detection of sign language gestures.

V.SIGN LANGUAGE RECOGNITION TECHNIQUES:

A. IMAGE PREPROCESSING:

In the development of a sign language detection system using Convolutional Neural Networks (CNNs), image preprocessing is paramount to enhance the quality of input data and improve the model's performance. Initial steps involve resizing images to a consistent dimension, typically grayscale conversion to reduce computational complexity, and normalization to standardize pixel values. Noise reduction techniques, such as Gaussian blurring, contribute to mitigating irrelevant details, while contrast enhancement and edge detection algorithms help emphasize important features like hand gestures. Data augmentation, including rotation and flipping, diversifies the dataset, aiding the model in generalization. Extracting the Region of Interest (ROI) within an image refines focus on the relevant part, and leveraging pre-trained CNN models for feature extraction, through transfer learning, enhances the system's ability to recognize sign language gestures. Additionally, addressing class imbalance in the dataset ensures a fair representation of various signs, contributing to a more robust and unbiased model. Collectively, these preprocessing steps optimize input data, fostering improved accuracy and reliability in sign language gesture recognition.

Fig 5.1 Image Preprocessing

B. SEGMENTATION:

Edge-based segmentation is a crucial facet of sign language detection systems, specifically employed to delineate the boundaries of hands making sign gestures within an image. Commonly used edge detection algorithms, such as the Canny edge detector or Sobel operator, pinpoint significant changes in intensity, effectively highlighting the edges of objects. Following edge detection, a thresholding step converts the image into a binary representation, simplifying it to emphasize edges in white against a black background. Morphological operations, like dilation and erosion, refine the binary image, enhancing edge continuity. Contour detection algorithms then identify and extract the contours outlining the hands. Subsequently, the system isolates the Region of Interest (ROI), focusing on the hands and providing a refined input for subsequent stages of the detection system. In dynamic sign languages captured in video, this edge-based segmentation process extends to consecutive frames, ensuring the model can interpret the continuous movement of hands, essential for understanding dynamic sign gestures accurately. Overall, edge-based segmentation contributes significantly to the precision and effectiveness of sign language detection by emphasizing the critical boundaries of hands, facilitating feature extraction, and refining the input for subsequent classification.

Fig 5.2 HSV Segmentation

C.FEATURE EXTRACTION

Feature extraction in a sign language detection system is a pivotal step aimed at distilling meaningful and distinctive information from raw input data, typically images or video frames capturing sign gestures. These features play a crucial role in characterizing the visual elements of hand movements, enabling effective pattern recognition by machine learning algorithms. Spatial features encompass details about hand shape, size, and proportions, providing foundational insights into different signs. Temporal features capture the dynamic aspects, including motion trajectories and speed, especially crucial for understanding gestures in dynamic sign languages. Texture descriptors, such as Histogram of Gradients (HOG) or Local Binary Pattern (LBP), analyze textural patterns within the hands, contributing to nuanced recognition. Depth information, joint angle features, and convolutional neural networks (CNNs) provide a three-dimensional understanding and learn hierarchical features from images, respectively. Optical flow analysis examines motion vectors, while gesture segmentation and

finger spelling features refine the system's ability to recognize specific gestures and alphabet signs. The amalgamation of these extracted features serves as a condensed and informative input for machine learning models, enhancing their capacity to accurately interpret a diverse range of sign language expressions.
There are several algorithms and techniques that can be used for Feature extraction, including:

LOCAL BINARY PATTERN:

Local Binary Pattern (LBP) in sign language detection encodes textural details by comparing pixel intensities with their neighbors, creating a binary pattern for each pixel. It is effective in capturing surface texture nuances in hand gestures. LBP's resilience to illumination changes makes it valuable in diverse lighting conditions. Integration with machine learning models, like Support Vector Machines or Convolutional Neural Networks, enhances the accurate classification of sign language gestures. LBP is particularly advantageous for recognizing signs with varying hand poses and lighting scenarios.

Local Binary Pattern (LBP) is a texture descriptor commonly used in computer vision and image processing. It's often employed for tasks like facial recognition and texture analysis, and it can be adapted for sign language detection as well.

The LBP operator computes a binary code for each pixel in an image based on the relationship between its intensity and the intensities of its neighbors. The mathematical expression for LBP can be described as follows:

Mathematically, for a pixel P with intensity I_P and neighboring pixels P_0, P_1, ..., P_7 with intensities I_0, I_1, ..., I_7, the LBP value LBP_P can be expressed as:

$$LBP_P = \sum_{i-0}^{7} s(I_i - I_P) \times 2^i$$

where s(x) is a step function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

This LBP value is then used as a feature to represent the texture of the pixel P in the image. The process is repeated for all pixels in the image, creating an LBP representation of the entire image.

HISTOGRAM OF GRADIENTS:

Histogram of Gradients (HOG) in sign language detection extracts spatial features by analyzing the distribution of gradient orientations in hand gestures. It quantifies local patterns, capturing textural information crucial for recognition. HOG is particularly effective for encoding hand shape and contour details. It enhances the discriminative power of features in sign language images. Integration with machine learning models, such as Support Vector Machines, facilitates accurate classification of sign gestures.
The Histogram of Oriented Gradients (HOG) is a feature descriptor widely used in computer vision and image processing, including for sign language detection. HOG captures the local distribution of gradients in an image and is often employed as a feature vector for object detection. The mathematical expression for HOG can be described as follows:

$$H_{\text{norm}} = \frac{H}{\sqrt{\|H\|^2 + \epsilon^2}}$$

where:
- H is the concatenated histogram vector for the block.
- ϵ is a small constant to avoid division by zero.

This process is repeated for overlapping blocks across the entire image, resulting in a feature vector that represents the local gradient patterns in different regions.

D.CLASSIFICATION:

Classification is a pivotal phase in sign language detection systems, wherein the system assigns specific labels to input data, typically representing distinct sign gestures. The process begins with the extraction of relevant features from the input, encompassing spatial information, texture descriptors, or other characteristics essential for capturing the unique elements of sign language expressions. A dataset comprising labeled examples is then used for model training and evaluation. The selection of a suitable classification algorithm, such as Support Vector Machines, Random Forests, or Convolutional Neural Networks, follows, with the model learning to associate extracted features with corresponding sign language classes during training. Evaluation metrics, including accuracy and precision, assess the model's generalization to new, unseen sign gestures. Subsequently, the trained model is deployed to predict the class labels of new data, facilitating real-time interpretation of sign language expressions in live video streams or images. Classification thus forms a critical bridge between feature extraction and meaningful interpretation, enabling accurate recognition of diverse sign gestures in various scenarios.

K-NEAREST NEIGHBOR:

K-Nearest Neighbors (k-NN) in sign language detection classify gestures based on the majority class among their k nearest neighbors in the feature space. It operates on the principle of proximity, assigning labels to new instances based on the prevailing class among their closest neighbors. The algorithm is simple and effective, making it suitable for sign language scenarios. The choice of 'k' and distance metric influences its performance. K-NN is particularly valuable for real-time sign language recognition due to its simplicity and adaptability.
The k-Nearest Neighbors (k-NN) algorithm can be applied to a sign language detection system for classification purposes. Let's denote the following:

-X is the set of feature vectors representing sign language gestures.

-Y is the set of corresponding class labels for the sign language gestures.
-$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the labeled examples in the training dataset, where $x_i \in X$ and $y_i \in Y$.
- x_{new} is a new feature vector that we want to classify.

- For a sign language detection system, each class label y_i represents a specific sign or gesture. Assign x_new to the class label that is most frequent among its k-nearest neighbors.

$$\hat{y}_{new} = \text{argmax}_y \left(\sum_{i \in N_k(x_{new})} \delta(y_i, y) \right)$$

NAVIE BAYES CLASSIFIER:

Naive Bayes classifiers in sign language detection use probabilistic models to classify gestures based on extracted features. They assume independence between features, simplifying the learning process. Training involves estimating probabilities for each class given the features. During testing, the classifier calculates probabilities and assigns the most likely class to new instances. Naive Bayes classifiers provide a computationally efficient and interpretable solution for sign language recognition.
-X is the set of feature vectors representing sign language gestures.
-Y is the set of corresponding class labels for the sign language gestures.
-$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the labeled examples in the training dataset, where $x_i \in X$ and $y_i \in Y$.
- x_{new} is a new feature vector that we want to classify.
- For sign language detection, classify x_{new} into the class with the highest posterior probability.

$$\hat{y}_{new} = \text{argmax}_y \left(P(y) \times P(x_1, x_2, \ldots, x_d | y) \right)$$

E.TEXT GENERATION

Text generation in sign language recognition serves as a vital intermediary step, facilitating effective communication between individuals proficient in sign language and those unfamiliar with visual languages. Following the interpretation of key features like hand movements and facial expressions, the system converts the recognized sign gestures into textual or symbolic representations. This intricate process involves mapping each sign to its corresponding linguistic equivalent, essentially translating the visual language of signs into a written form. This output proves invaluable in enabling accessibility for individuals who may not possess sign language proficiency, fostering inclusivity in various social, educational, and professional contexts.

Furthermore, text generation extends the utility of sign language recognition by enabling seamless integration with other communication platforms and technologies. The ability to convert sign gestures into written text not only aids in real-time communication but also opens avenues for archived conversations, searchable databases, and improved accessibility in digital environments. By providing a textual representation of sign language, the system contributes to breaking down communication barriers, creating a more inclusive society where individuals with diverse communication needs can interact and collaborate effectively.

Fig 5.3 Overview of the proposed system

VI. RESULTS

The project was meticulously executed through the utilization of the versatile Python scripting language, showcasing a sophisticated integration of crucial libraries such as openCV2, Pandas, Keras, and Numpy. The development process unfolded seamlessly within the well-regarded Visual Studio Code 2019, chosen as the preferred development environment for its robust features and user-friendly interface.

Throughout the implementation phase, a rigorous testing regimen was diligently conducted. The testing environment was characterized by a high-performance computer system, boasting an Intel® Core™ i7 8300H processor operating at a clock speed of 2.3 GHz. This choice of hardware aimed to ensure optimal performance and efficiency during both development and testing phases.

The model, a testament to the meticulous design and thorough testing, demonstrated commendable prowess by achieving an impressive accuracy level of 83%. This noteworthy accomplishment speaks to the effectiveness of the chosen technologies and methodologies employed in the project, underscoring the commitment to precision and excellence in its execution.

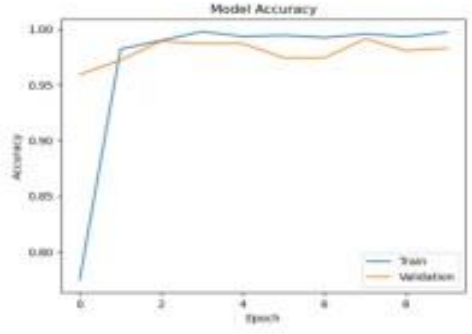


Fig 5.4 Accuracy graph

VII. FUTURE WORK

The potential future developments encompass, but are not confined to:

- Extending the application of our model to cater to additional sign languages, such as the Indian Sign Language, moving beyond its current focus on American Sign Language.

- Advancing the model's capabilities to identify common words and expressions more effectively.

- Further refining the neural network to adeptly recognize symbols involving the coordination of two hands.

- Improving overall performance by incorporating linear classifiers into the system.

- Expanding the scope by integrating dynamic hand gestures alongside the existing static finger spelling.

- Exploring the integration of convolutional neural networks to account for depth data, enabling the detection of gestures captured by devices like Kinect.

VIII. CONCLUSION

The Sign Language Recognition project aims to revolutionize communication for the deaf and mute community, particularly catering to individuals unfamiliar with sign language. This comprehensive project is comprised of two pivotal components: advanced image processing for skin color detection and precise hand identification.

The algorithm, a cornerstone of this initiative, is executed on a laptop equipped with an integrated camera, which captures images essential for the project's success. This image data is then meticulously transformed into a robust dataset consisting of 240 images, each portraying 10 instances of a specific sign. The dataset is thoughtfully curated to encompass 24 distinct signs from the English alphabet, meticulously adhering to the rules of one-handed Indian Sign Language.

To ensure the effectiveness of the model, the dataset is strategically divided into both test and train sets, with 15% reserved exclusively for rigorous testing. Leveraging the power of a Convolutional Neural Network model, intricate features are extracted from the signs, culminating in an impressive 83% accuracy rate during the rigorous testing phase. This project stands as a testament to the potential of technology in fostering inclusive communication for diverse communities.

VIII. REFERENCES

[1] Zafar Ahmed Ansari and Gaurav Harit, "Nearest neighbor classification of Indian sign language gestures using Kinect camera", volume 41, pages 161-182,2016.

[2] Zhaoyang Yang, Zhenmei Shi, Xiaoyong Shen and Yu-Wing Tai, "Structured Feature Network for Continuous Sign Language Recognition",2019

[3] Ashish S. Nikam and Aarti Ambedkar, "Sign Language Recognition Using Image Based Hand Gesture Recognition Techniques",2016

[4] Aditya Dasl, Shantanu Gawde, K. Suratwala and D. Kalbande, "Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images",2018

[5] Ali Karami, Bahman Zanj and Azadeh Kiani Sarkaleh, "Persian sign language (PSL) recognition using wavelet transform and neural networks.",volume 38, March 2011

[6] Nadia Rashid and Yasser Alginahi, "Real-Time Arabic Sign Language (ArSL) Recognition", Volume 1, January 2012

[7] Geetha Madathikulangara, Manjusha C. Unnikrishnan Parameswaran and HariKrishnan R, "A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images", october2013

[8] P.V.V. Kishore and Panakala Rajesh Kumar, "Video Audio Interface for Recognizing Gestures of Indian Sign Language", January 2012

[9] V. Adithyu, P.R. Vinod and Usha Gopalakrishnan, "Artificial Neural Network Based Method for Indian Sign Language Recognition.", April 2013

[10] Subha Rajam and Balakrishnan Ganesan, "Real time Indian Sign Language Recognition System to aid deaf-dumb people", September 2011

[11] Jinalee Jayeshkumar Raval and Ruchi Gajjar, "Real-time Sign Language Recognition using Computer Vision,14 May2021

[12] Amrutha K and Prabu P, "ML Based Sign Language Recognition System", February 2021

[13] "Hear Sign Language: A Real-time End-to-End Sign Language Recognition System" Zibo Wang, Senior Member IEEE, Tengda Zhao, Jinxin Ma, Hongkai Chen, Kaixin Liu, Huajie Shao, Member, IEEE, Qian Wang, Senior Member, IEEE, Ju Ren, Member, IEEE

[14] Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images Aditya Das1, Shantanu Gawde1, Khyati Suratwala1 and Dr. Dhananjay Kal band

[15] A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition Nikolas Adaloglou1, Theocharis Chatzis1, Ilias Papastratis1, Andreas Stergioulas1, Georgios Th. Papadopoulos1, Member IEEE, VassiaZacharopoulou , George J. Xydopoulos2, Klimnis Atzakas2 , Dimitris Papazachariou2 , and Petros Daras1 , Senior Member, IEEE

[16] Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video Thad Starner, Student Member, IEEE, Joshua Weaver, and Alex Pentland, Member, IEEE Compute

873

ORIGINALITY REPORT

17%	8%	13%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Jinalee Jayeshkumar Raval, Ruchi Gajjar. "Real-time Sign Language Recognition using Computer Vision", 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 2021 Publication	2%
2	Submitted to University of Lancaster Student Paper	1%
3	Aditya Das, Shantanu Gawde, Khyati Suratwala, Dhananjay Kalbande. "Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images", 2018 International Conference on Smart City and Emerging Technology (ICSCET), 2018 Publication	1%
4	I.A. Adeyanju, O.O. Bello, M.A. Adegboye. "Machine learning methods for sign language recognition: A critical review and analysis", Intelligent Systems with Applications, 2021 Publication	1%
5	www.researchgate.net	

	Internet Source	1 %
6	www.ncbi.nlm.nih.gov Internet Source	1 %
7	www.amrita.edu Internet Source	1 %
8	www.ieee-whispers.com Internet Source	1 %
9	Submitted to University of Greenwich Student Paper	1 %
10	www.ijsr.net Internet Source	1 %
11	"Soft Computing for Security Applications", Springer Science and Business Media LLC, 2023 Publication	<1 %
12	healthdocbox.com Internet Source	<1 %
13	Submitted to CSU, San Jose State University Student Paper	<1 %
14	ijritcc.org Internet Source	<1 %
15	Dr.E. Padmalatha*, S. Sailekya, Dr.R.Ravinder Reddy, Ch.Anil Krishna, K. Divyarsha. "Sign Language Recognition", International Journal	<1 %

of Recent Technology and Engineering (IJRTE), 2019
Publication

16	Rohini K Katti, Sujatha C, Padmashri Desai, Shankar G. "Character and Word Level Gesture Recognition of Indian Sign Language", 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), 2023 Publication	<1 %
17	ggu.ac.in Internet Source	<1 %
18	link.springer.com Internet Source	<1 %
19	mspace.lib.umanitoba.ca Internet Source	<1 %
20	I. Keren Beulah, Kumudha Raimond, G. Litisha Miraclin. "Indian Sign Language Recognition for Static Gestures using DenseNet169 Model", 2023 8th International Conference on Communication and Electronics Systems (ICCES), 2023 Publication	<1 %
21	Submitted to Letterkenny Institute of Technology Student Paper	<1 %

www.slideshare.net

22	Internet Source	<1%
23	Adithya, V., P. R. Vinod, and Usha Gopalakrishnan. "Artificial neural network based method for Indian sign language recognition", 2013 IEEE CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES, 2013. Publication	<1%
24	Submitted to Higher Education Commission Pakistan Student Paper	<1%
25	Submitted to University of Ghana Student Paper	<1%
26	Submitted to The NorthCap University, Gurugram Student Paper	<1%
27	Submitted to University of Northampton Student Paper	<1%
28	K. Rekha, B. Latha. "Mobile Translation System from Speech Language to Hand Motion Language", 2014 International Conference on Intelligent Computing Applications, 2014 Publication	<1%
29	dyuthi.cusat.ac.in Internet Source	<1%

30	www.irjmets.com Internet Source	<1 %
31	Nguyen Manh Dung, Soonghwan Ro. "Algorithm for Fire Detection using a Camera Surveillance System", Proceedings of the 2018 International Conference on Image and Graphics Processing, 2018 Publication	<1 %
32	Rajiv Ranjan, B Shivalal Patro, Mohammad Daim Khan, Manas Chandan Behera, Raushan Kumar, Utsav Raj. "A Review on Sign Language Recognition Systems", 2021 IEEE 2nd International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC), 2021 Publication	<1 %
33	cdn.techscience.cn Internet Source	<1 %
34	ijream.org Internet Source	<1 %
35	www.codewithc.com Internet Source	<1 %
36	Ton Duc Thang University Publication	<1 %
37	Yasin Saifnijat, Atiqullah Qaderi. "Pashto Isolated Digits Sign Language Recognition",	<1 %

2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

REFERENCES

[1] Zafar Ahmed Ansari and Gaurav Harit,"Nearest neighbour classification of Indian sign language gestures using kinect camera" ,volume 41,pages 161-182,2016.

[2] Zhaoyang Yang, Zhenmei Shi, Xiaoyong Shen and Yu-Wing Tai,"Structured Feature Network for Continuous Sign Language Recognition",2019

[3] Ashih S.Nikam and Aarti Ambedkar,"Sign Language Recognition Using Image Based Hand Gesture Recognition Techniques",2016

[4] Aditya Dasl, Shantanu Gawde, K. Suratwala and D. Kalbande," Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images",2018

[5] Ali Karami, Bahman Zanj and Azadeh Kiani Sarkaleh,"Persian sign language (PSL) recognition using wavelet transform and neural networks.",volume 38,March 2011

[6] Nadia Rashid and Yasser Alginahi," Real-Time Arabic Sign Language (ArSL) Recognition",Volume 1,January 2012

[7] Geetha Madathikulangara, Manjusha C,Unnikrishnan Parameswaran and HariKrishnan R,"A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images",october2013

[8] P.V.V. Kishore and Panakala rajesh kumar," Video Audio Interface for Recognizing Gestures of Indian Sign Language",January 2012

[9] V.Adithya,P.R.Vinod and Usha GopalKrishnan," Artificial Neural Network Based Method for Indian Sign Language Recognition.",April 2013

[10] Hear Sign Language: A Real-time End-to-End Sign Language Recognition System"Zhibo Wang, Senior Member, IEEE, Tengda Zhao, Jinxin Ma, Hongkai Chen, Kaixin Liu, Huajie Shao, Member, IEEE, Qian Wang, Senior Member, IEEE, Ju Ren, Member, IEEE

[11] A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition Nikolas Adaloglou1, Theocharis Chatzis1*, Ilias Papastratis1*, Andreas Stergioulas1*, Georgios Th. Papadopoulos1,Member IEEE,VassiaZacharopoulou , George J. Xydopoulos2 , Klimnis Atzakas2 , Dimitris Papazachariou2 , and Petros Daras1 , Senior Member, IEEE

[12] Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video Thad Starner, Student Member, IEEE, Joshua Weaver, and Alex Pentland, Member, IEEE Computer