

A PROJECT REPORT ON SIGN LANGUAGE TO TELUGU SPEECH TRANSLATOR

A Major Project Submitted to
Jawaharlal Nehru Technological University, Kakinada
in Partial fulfillment of Requirements for the Award of the Degree of
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING (AI & ML)



Submitted By

Ms. HANNAH HAVILAH (20KT1A4263)

Under the Esteemed Guidance of

Mrs. CH.B.V. DURGA, M.Tech, (Ph.D)

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

**POTTISRIRAMULUCHALAVADIMALLIKHARJUNARAO COLLEGE
OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)**

**(Approved by AICTE New Delhi, Affiliated to JNTU-Kakinada)
KOTHAPET, VIJAYAWADA-520001, A.P**

2020-2024

**POTTI SRIRAMULU CHALAVADI MALLIKHARJUNA RAO
COLLEGE OF ENGINEERING & TECHNOLOGY
KOTHAPET, VIJAYAWADA-520001.**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)



CERTIFICATE

This is to certify that the project work entitled “**SIGN LANGUAGE TO TELUGU SPEECH TRANSLATOR**” is a bonafide work carried out by **Hannah Havilah (20KT1A4263)**. Fulfillment for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE & ENGINEERING (AI&ML)** of Jawaharlal Nehru Technological University, Kakinada during the year **2023-2024**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the above degree.

Project Guide

Head of the Department

External Examiner

ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who helped and supported and suggested us in every step. We are glad for having the support of our principal **Dr. J. Lakshmi Narayana** who inspired us with his words filled with dedication and discipline towards work. We express our gratitude towards **Mrs. N. V. Maha Lakshmi, HOD of AIML** for extending her support through technical and motivation classes which had been the major source to carrying out our project. We are very much thankful to **CH. B. V. Durga, Associate Professor**, Guide of our project for guiding and correcting various documents of ours with attention and care. She has taken the pain to go through the project and make necessary corrections as and when needed. Finally, we thank one and all who directly and indirectly helped us to complete our project successfully.

Project Associates

Ms. SAKAM RANI (20KT1A4247)

Mr. YARLAGADDA RAKESH (21KT5A4206)

Ms. HANNAH HAVILAH (20KT1A4263)

DECLARATION

This is to declare that the project entitled “**SIGN LANGUAGE TO TELUGU SPEECH TRANSLATOR**” submitted by us in the partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering (AI & ML)** in **Potti Sriramulu Chalavadi Mallikharjuna Rao College of Engineering and Technology**, is bonafide record of project work carried out by us under the guidance of **CH.B.V.Durga, Associate Professor**. As per our knowledge, the work has not been submitted to any other institute or universities for any other degree.

Project Associates

Ms. SAKAM RANI (20KT1A4247)

Mr. YARLAGADDA RAKESH (21KT5A4206)

Ms. HANNAH HAVILAH (20KT1A4263)

ABSTRACT

ABSTRACT

The sign language to speech project is to develop a system that translates gestures from sign language into spoken Telugu using the Random Forest algorithm. It can be challenging for non-speakers to use sign language, which is the primary form of communication for those who are dumb or mute. Our intention in creating this technology is to bridge the communication gap between non-sign language users and sign language users. Using a machine learning technology called the Random Forest algorithm, sign language motions will be properly recognized and interpreted. As a result, the system will generate Telugu speech output, which indicates that our model will generate a Telugu voice. This will enable speakers and sign language users like dumb or unable to speak people to communicate in real time. This project aims to promote inclusivity and accessibility for people who cannot speak for themselves. An existing models doesn't produce a proper results. So, we are overcome those drawbacks to gives a efficient model. Using the Random Forest algorithm, Our model translate sign language to speech has reached an astonishing accuracy rate of 99%. The project aims to improve sign language to speech accuracy by integrating gesture recognition and speech synthesis. Using machine learning algorithms like Random Forest, the project aims to provide real-time translation and seamless communication for individuals with dumb or unable to speak, potentially exceeding current accuracy standards.

KEYWORDS: American Sign Language (ASL), Speech Synthesis, Telugu Language, Communication, Machine Learning, Gesture Recognition.

CONTENTS

1. Introduction	1-7
1.1 Brief Overview of project	1
1.1.1 Scope	1
1.1.2 Purpose	1
1.1.3 Objective of study	2
1.1.4 Literature Review	2
1.2 Problem Statement	6
1.3 Proposed System	7
2. System Analysis	8-12
2.1 System Study	
2.1.1 Feasibility Study	8
2.1.1.1 Operational Feasibility	8
2.1.1.2 Technical Feasibility	8
2.1.1.3 Behavioural Feasibility	8
2.1.1.4 Financial and Economic Feasibility	9
2.2 System Requirements	
2.2.1 Functional Requirements	9
2.2.2 Non-Functional Requirements	9
2.3 System Requirement specification	
2.3.1 Hardware Requirements	10
2.3.2 Software Requirements	10
2.3.3 Required Libraries	11
2.4.1 Proposed Methodology Name	11

2.4.2 Traditional Approaches	11
3. System Design	13-17
3.1 About system design	13
3.1.1 Initialize design definition	13
3.1.2 Establish design characteristics	14
3.2 System Architecture	15
3.3 Data Flow Diagrams/ UML Diagrams	16
3.4 Datasets (If any)	17
4. System Implementation	18-33
4.1 System setup	18
4.2 Code	19
4.3 Results	32
5. Testing	34-35
5.1 Performance Metrics	34
5.2 Validating the Test Cases	35
6. Conclusion	36
7. Future Work	37
8. References	38
9. Bibliography	39
10. Appendix	40-41
10.1 Python Introduction	40
10.2 Machine Learning	41

LIST OF FIGURES

S.NO	FIGURE NO	NAME OF THE FIGURE	PAGE NO
1	1	Architecture	15
2	2	Data Flow Diagram	16
3	3	Example Of ASL Image	17
4	4	Performance Metrics Results	32
5	5	Actual And Predicted Results	33
6	6	Final Output Of Interface	33
7	7	Sign Language To Speech Conversion	36

LIST OF TABLES

S.NO	TABLE NO	NAME OF THE TABLE	PAGE NO
1	1	Reference Paper Details	4
2	2	Validation Of Test Cases	35

SIGN LANGUAGE TO TELUGU SPEECH TRANSLATOR

INTRODUCTION

1. INTRODUCTION

1.1 BRIEF OVERVIEW OF THE PROJECT

The goal of the sign language to speech project is to create a revolutionary technology that will enable smooth communication between Telugu speakers and sign language users. The idea for the project originated from the realization that people who are dumb or mute/hard of hearing, especially in the Telugu-speaking community, encounter communication obstacles. Through the utilization of cutting-edge technologies and machine learning algorithms, the project aims to close this disparity and improve inclusivity for every person, regardless of communication skills.

1.1.1 Scope:

The project's declared boundaries are included in the scope of the sign language to speech initiative. This covers its goals, deliverables, and limitations. The project's specific goal is to put in place a reliable system that can translate spoken Telugu into sign language movements, with an emphasis on real-time communication. It will entail creating algorithms for speech synthesis and gesture recognition and combining these features into an intuitive user interface.

1.1.2 Purpose:

Problem Statement: The project aims to tackle the particular issue of communication barriers that affect members of the Telugu-speaking community who rely on sign language. These barriers impede social interaction and effective communication, which in turn restricts opportunities for inclusion and participation.

Objectives: The project's objectives are to:

- Provide a system that can translate sign language gestures into Telugu speech with accuracy.
- Allow Telugu speakers and sign language users to communicate in real time.
- Encourage accessibility and inclusivity for people who are dumb or mute/hard to hear, especially in the Telugu-speaking community.

Advantages: Improved social engagement and communication for sign language users. enhanced

information and service accessibility for the Telugu-speaking community. Greater engagement and inclusivity in a range of contexts, such as social situations, work, and education.

1.1.3 Objective of the study:

Creating a reliable and user-friendly system to translate sign language motions into spoken Telugu is the main goal of the sign language to speech project. This entails using speech synthesis methods to produce Telugu speech output and sophisticated machine learning algorithms, like the Random Forest algorithm, for gesture identification. The study also intends to assess the effectiveness of the system that has been established and determine how it can enhance accessibility and communication for people who use sign language.

1.1.4 Literature Review:

[1] Akshatha Rani states that the system, The American Sign Language (ASL) dataset, hand tracking methods, and artificial neural network (ANN) architecture are used by the system to identify the alphabet in sign language and translate it into voice. With a 74% accuracy rate, the technology also provides text-to-speech conversion for blind users. By bridging the communication gap between deaf-mute people and others, the suggested approach improves inclusion and accessibility through technology.

[2] Aishwarya Ramesh states that, Understanding which features to extract from static photos is the first step in designing a suitable model. The improved invariance with changes in lighting and shadows is the outcome of this normalization. HOG descriptors are used as the image features for the machine learning algorithm Support Vector Machine (SVM). Therefore, SVM is used to train our model. In this experiment, three distinct array parameters are utilized for SVM, and the outcomes of each are compared. The Detection Method, Kernel, and Dimensionality reduction type are the three array parameters.

[3] Amrutha K states that, The common means of communication for people who are speech and hearing-impaired is sign language. Users can communicate more easily because of the region-specific sign language division. The hearing impaired and others with speech impairments typically rely on human translators because a greater portion of society does not understand sign language. It may not always be feasible to use a human interpreter at a reasonable price or

availability. An automated system that can read, understand, and translate sign language into comprehensible form would be the greatest replacement. The communication gap that exists among people in society would be lessened by this translator. For a continuous and fluid sign, the SLR needs be trained with a large amount of sign language data and its syntax.

[4] Salma A. Essam El-D in states that, In order to communicate in daily life, those who are deaf or mute greatly benefit from Sign Language (SL). Instead of using sound patterns to communicate, one might use hand gestures, such as American Sign Language (ASL) or any other SL. In SL, body part movement, orientation, and defined shapes are all done at the same time. The first issue is that most healthy individuals comprehend sign languages very little to nothing. For those who are deaf or mute, effective communication is therefore viewed as a difficulty and barrier in their daily life. When a deaf person converses on one side using sign language (SL) that he is accustomed to and comfortable with, the system converts that SL into sound and pictures that the person can understand.

[5] Ayush Pandey states that, The experiment shows how CNN can be used to solve computer vision difficulties; it can translate sign language with a 95% accuracy in finger typing. By creating datasets and training CNN, it can be expanded to support additional sign languages. Eliminating the need for interpreters, keeping an eye out for distorted greyscales, and achieving precise prediction while wearing gloves are the major goals.

[6] Ashok Kumar Sahoo states that, The study recommend staking pictures of hand posture, showing text, and improving ISL digit identification with a graphical user interface. Users can add their indications, and the system can forecast outcomes with 100% accuracy. Future studies could investigate classifiers, feature extraction strategies, and combination to build a comprehensive ISL recognition system with 100% real-time interpretation accuracy of ISL signs as the goal.

[7] Bayan Mohammed Saleh states that, For non-speakers in particular, slide interpretation and speech interpretation are crucial to sign language communication. Inaccurate hand segmentation and gesture prediction can result from dim lighting. Inaccuracies can also result from in accurate peripherals. Sign language technology development is essential for productivity, professional advancement, and improving social interactions. People who are deaf or dumb will benefit from this breakthrough.

[8] Narayana Dharapaneni states that, The history, composition, and significance of American

Sign Language (ASL) among the deaf community are covered in this document, with a focus on the language's rich cultural heritage. In order to foster inclusivity and understanding, it draws attention to the importance of interpreters and promotes ASL education and awareness.

[9] AdithyaV.states that, In order to reduce computational load and uncover properties that differentiate hand postures, deep learning has been applied to the task of hand posture identification from raw pictures. Experiments on publicly available datasets proved the efficacy of the suggested CNN architecture by showcasing improved recognition performance in terms of accuracy, precision, and recall.

[10] Ankita Saxena states that, The project's main goal is to employ Principal Component Analysis to create a sign language recognition system that will allow hearing- impaired people to communicate. The system uses an android device or webcam to record live video frames ,which it then uses to match static hand motions with a database to generate text or speech commands. With successful recognition rates of about 90%according to test results, it is a useful tool for bridging the communication gap between the general public and deaf and mute people.

Table1: Reference Papers Details

S. N O	Author	Title	Dataset	Algorithm	Merits	Demerits	Accuracy
1	Akshatha Rani	Sign Language to Text-Speech Translator UsingML	American Sign Language (ASL) dataset	SVM, CNN.	Converts sign text to speech, Aiding Communication for deaf and blind	It is less accurate	74%
2	Aishwarya Ramesh	Real-time Conversion of Sign Language toTextand Speech.	OpenCV	RBF, PCA	Development of an Android app for real-time ASL conversion.	Less optimization due to underfitting	N/A

3	Amrutha K	ML Based Sign Language Recognition	ASL Dataset	KNN	Vision based isolated hand gesture detection and recognition in the model	Less Accuracy	65%
4	Salma Essam El-Din, Mohamed El-Ghany	Sign Language Interpreter System	OpenCV	SLR, ASL, ArSL	higher recognition dynamic accuracy than ASL	System struggles with gesture-s due to Speed and accuracy	88%
5	Ayush Pandey	Sign Language to Text and Speech Translation	OpenCV	CNN	Real-time Needs to N/A sign language translation using CNN for deaf community	Mainly focusing on single gesture transform	95%
6	Ashok Kumar Sahoo	Indian Sign Language Recognition Using Machine Learning	Sign database with 5000 images, 500 for each numeral sign	KNN	Recognition of ISL immobile numeric signs using classifiers	Less Efficient	N/A
7	Bayan Mohammed Saleh	D-Talk: Sign Language Recognition System for People With Disability	Sign Language Image Dataset	SLR	Utilizes machine learning and image processing for communi	Less Accuracy	60%

		Using Machine Learning and Image Processing			-cation enhancement		
8	Narayana Dharapaneni	American Sign Language Using instance based segmentation	OpenCV	CNN	The study selection highlights the Benefits Of using AI in healthcare	Less Efficient	N/A
9	AdithyaV	A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition	NUS hand posture dataset	CNN	More Optimize	Overfitting	94%
10	Ankita Saxena, Deepak Kumar Jain, Ananya Singhal	Sign Language Recognition Using Principal Component Analysis	Database of 10 sign gestures from Indian sign language	PCA	PCA used for extracting Useful data, Dimension reduction	Performance May decrease due to varying lighting conditions and background noise.	Recognition rate of gestures: 70 - 80%.

1.2 PROBLEM STATEMENT

The goal of the sign language to speech project is to enable efficient communication and social interaction by bridging the gap between Telugu speakers and sign language users. The project's main goal is to create a dependable, user-friendly system that can reliably decipher sign language movements and instantly produce meaningful Telugu speech. All users should be able to utilize the system, regardless of their level of Telugu and sign language expertise.

By tackling these issues, the intention is to improve the quality of life, foster social inclusion, and increase accessibility for people with communication problems.

1.3 PROPOSED SYSTEM

The suggested method for creating a sign language to speech system takes a methodical approach with the goal of efficiently converting sign language motions into spoken Telugu. The first step in the process is to compile a dataset of pictures that show different sign language gestures. The Media Pipe library is then used to extract hand coordinates from these photos, allowing for the accurate localization of hand movements inside the gestures. The hand coordinates that were retrieved are then preprocessed and arranged in a dictionary style, along with class labels that indicate the appropriate sign language movements. A machine learning model, like a Random Forest Classifier, is trained on this carefully selected dataset in order to determine sign language motions with accuracy.

The trained model's performance is assessed on a validation set in order to verify its effectiveness, with an emphasis on attaining a high degree of accuracy. When the trained model reaches the target accuracy criterion, it is stored in a pickle file for later use. In the future, the technology is used in real-time to record live video streams using OpenCV, enabling the trained model to anticipate sign language movements instantly. Characters are taken out of gestures and joined together to create meaningful words or sentences.

The recognized text is translated using the Googletrans library from English to Telugu or another target language to improve the system's accessibility. The translation is then rendered into voice by use of the GTTS (Google Text-to-voice) library, which translates the text into the target language, in this case Telugu, and interprets the movements using sign language.

This comprehensive approach efficiently bridges the gap between spoken and sign language for individuals with speaking problem, facilitating their communication and encouraging more inclusivity and accessibility in communication.

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

An important step in the development process is system analysis, which focuses on analyzing and assessing the system in order to improve or change its functionality. It entails dissecting the issue into more manageable parts for in-depth analysis, which eventually results in the identification of possible fixes. Analysis is essential to the system development process since it helps identify problems, offer ideas, and ultimately lead to the delivery of a computerized solution.

2.1 SYSTEM STUDY

2.1.1 Feasibility Study:

It is crucial to evaluate a project's viability and potential impact on the organization before starting any new one. Studies of feasibility examine the possible results and repercussions of system development, taking into account both favorable and unfavorable effects. Economic, technological, operational, and behavioral feasibility are the four main factors that are usually assessed.

2.1.1.1 Operational Feasibility:

Operational viability evaluates how well the suggested solution can fulfill the organization's operational needs. Its main goal is to ascertain whether, once constructed and put into use, the system will perform as expected. Operational viability is guaranteed in the case of our sign language to voice project since the suggested solution sufficiently meets users' communication demands and encourages regular use.

2.1.1.2 Technical Feasibility:

Technical needs and resources required for system development and maintenance are assessed in terms of technical feasibility. The software facilities, processes, and inputs needed for system functionality are identified during this phase. Technical feasibility analysis for our project verifies that all resources required for software development and maintenance are easily accessible, making efficient use of already-existing resources.

2.1.1.3 Behavioural Feasibility:

Behavioral feasibility evaluates the possibility that users will accept and adjust to the suggested system. It recognizes that people could be resistant to change and looks at how prepared users are to move to a digital system. Comprehending the behavior and preferences of users is crucial to

guaranteeing appropriate authorization, authentication,

and security of confidential information within the company. The system has the ability to support different user levels and mitigate possible resistance to change, so enabling a seamless transition and increasing user engagement.

2.1.1.3 Financial and Economic Feasibility:

Assessing the system's cost-effectiveness and economic rationale is the main goal of the financial and economic feasibility examination. To do this, a cost-benefit analysis must be performed to ascertain the anticipated savings and advantages of the system in relation to its expenses. The purpose is to determine if the advantages outweigh the disadvantages in order to justify system design and deployment. The economic viability of our sign language to speech project is excellent. The system's implementation will save a great deal of time and money for the company because it will not require new hardware resources. This economic analysis provides a strong argument for the proposed system's adoption by highlighting its feasibility and value proposition.

2.2 SYSTEM REQUIREMENTS

The goal of the proposed sign language to voice system is to automate report generation and seamlessly provide necessary information in order to streamline manual procedures.

2.2.1 Functional Requirements:

Functional requirements, which are customized for the kind of software, anticipated users, and system environment, specify the precise functioning of a system or its subsystems. While functional system requirements offer comprehensive descriptions of system services, functional user requirements provide high-level declarations of what the system should be able to do.

2.2.2 Non-Functional Requirements:

In addition to functional requirements, non-functional requirements specify criteria that are used to assess how well a system functions overall rather than just for certain capabilities.

- **Usability:** The system places a high priority on usability, offering a straightforward and user-friendly interface to make it easier for all users to utilize. It is made to be as simple to use as possible for both inexperienced and seasoned users, saving time and reducing confusion.
- **Reliability:** The system demonstrates good dependability even in loud situations by utilizing Python platform at the receiving end and Google-developed APIs for sign language recognition.

This guarantees steady performance and strengthens the system's resilience.

- **Performance:** The system operates at a high level because to optimization strategies and high-level language development. Users receive prompt responses from it, which improves productivity and user experience.
- **Supportability:** The system can operate on a variety of hardware and software platforms because it was built to be cross-platform compatible. Because Python is used, portability is improved and it may be easily deployed in a variety of contexts.
- **Flexibility:** After deployment, additional modules and expansions can be easily integrated into the system thanks to its flexible design. This guarantees that any future improvements or changes can be applied without interfering with current features or system design.

2.3 SYSTEM REQUIREMENT SPECIFICATION

The official document that describes and defines the system-level requirements of the sign language to voice application is called the System Requirements Specification (SRS). The Project Concept Proposal, Project Business Case, and Project Charter all identify, define, and formalize the requirements needed to meet the operational and functional demands. By signing this paper, you agree that the produced system will be accepted even though it satisfies these requirements.

2.3.1 Hardware Requirements:

The following pieces of hardware are needed for the sign language to speech system:

- A web cam or camera to record sign language movements
- A computer or mobile device with enough RAM and computational power
- Optional microphone to record audio input
- For audio output, use speakers or headphones.

2.3.2 Software Requirements:

- Operating System (Windows)
- Environment for development: Vscode
- OpenCV library for video recording and image processing
- Recognition of hand gestures using Media Pipe library
- Googletrans is a text translation library.
- Google Text-to-Speech(GTTS)library for speech synthesis applications

2.3.3 Required Libraries:

The following libraries are necessary for the system:

- Random Forest Classifier
- Numpy (np)
- OpenCV is an Open Source Library for Computer Vision
- A framework for creating cross-platform machine learning pipelines is called Media Pipe.
- Googletrans: A Python Google Translate API module
- GTTS: Google Text-to-Speech API Python library

2.4 METHODOLOGIES

2.4.1 Proposed Methodology:

The proposed methodology for developing the sign language to speech system begins with gathering a dataset of sign language gestures. Hand coordinates are then extracted using the Media Pipe library and the data is preprocessed and labeled. A machine learning model, such as a Random Forest Classifier, is trained on the prepared dataset. Real-time prediction is achieved using OpenCV. Text translation is facilitated using Googletrans, followed by speech synthesis using GTTS for the final output

2.4.2 Traditional Approaches:

Prior to the development of machine learning algorithms like Random Forest, traditional methods for translating sign language to voice included:

- ✧ **Rule-based Systems:** These systems convert spoken language from sign language gestures using established grammatical structures and linguistic rules. Rules are created by hand using linguistic understanding and sign language grammar competence.
- ✧ **Handcrafted Feature Engineering:** This method uses preset methods and approaches to extract and express characteristics from sign language motions. Spoken language output is produced by mapping features like hand forms, gestures, and facial expressions to linguistic parts.
- ✧ **Template Matching:** In order to find relevant linguistic elements, template matching systems compare incoming sign language motions to predetermined templates or patterns. Nevertheless, this method might not be robust enough to handle variances in sign language phrases.
- ✧ **Hidden Markov Models (HMMs):** The temporal dependencies and variability of sign language motions have been modeled using HMMs. They may, however, be limited in their scalability by the

need for substantial training data and manual gesture classification.

- ✧ **Neural Networks (Early Architectures):** The recognition and translation of sign language has been studied using early neural network architectures, such as feed forward networks and recurrent neural networks (RNNs). However, the availability of training data and restricted computer resources may be problems for these methods.

SYSTEM DESIGN

3. SYSTEM DESIGN

3.1 ABOUT SYSTEM DESIGN

The methodical process of conceiving and organizing a system's architecture, modules, components, interfaces, and data flow is known as system design.

Purpose: The main goal of system design is to provide thorough information about the system and all of its components so that it may be implemented in a way that aligns with the architectural entities that are defined in the models and views of the system architecture.

Elements of a System:

- **Architecture:** The architecture is a conceptual model that describes the behavior, structure, and different perspectives of the system. Flowcharts are a useful tool for visually representing architecture and showing how various components relate to one another.
- **Modules:** Modules are distinct parts of the system, each in charge of carrying out a certain function. The system is made up of all of these elements, which allows for modularization for simpler development and management.
- **Components:** Within the system, components perform particular functions or collections of related functions. They consist of modules, each of which adds to the system's total performance and usefulness.
- **Interfaces:** Interfaces are the common boundaries that allow system components to communicate and engage with one another. Clearly specified interfaces make it easier for various system components to integrate and communicate with one another.
- **Data:** Managing data entails managing information flow and handling inside the system. It includes operations like gathering, storing, processing, and retrieving data, guaranteeing effective use and administration of data resources across the system.

3.1.1 Initialize design definition:

At the beginning of the design phase of the sign language to speech project, the following tasks will be carried out:

- **Plan and Identify Technologies:** The technologies that will make up and execute the system's components, as well as their physical interfaces, will be thoroughly planned. To enable precise sign language detection and voice synthesis, this involves choosing the right hardware and

software components, such as cameras, microphones, image processing libraries, and machine learning frameworks.

- **Risk Assessment for Technologies:** During the system's operation phase, a thorough risk assessment will be carried out to identify which technologies and system components are most likely to become outdated or undergo evolution. To maintain the system's long-term viability and sustainability, contingency plans will be developed to reduce any hazards. These plans will include methods for technology replacement or adaption as needed.
- **Documentation of Design Definition Strategy:** A thorough documentation of the design definition strategy will be provided, detailing the necessary systems, goods, or services that must be in place in order to carry out the design in an efficient manner. The development team will have clear goals and guidelines to follow from this documentation, which will act as a road map for the design implementation process.

3.12 Establish design characteristics:

The following steps will be taken in order to establish design characteristics for the sign language to speech project:

- **Define Architectural Characteristics:** In order to meet the project's objectives, key architectural attributes such as modularity, scalability, flexibility, and performance will be specified. Design characteristics related to the system's architecture will be carefully defined, ensuring that they are implementable within the planned framework.
- **Refine Interfaces:** We will identify and specify any interfaces that need to be refined as design details change or that were not fully defined throughout the System Architecture phase. In order to guarantee smooth integration and inter operability, this involves defining the communication protocols, data formats, and interaction patterns between various system components.
- **Document Design Characteristics of Each System Element:** To give readers a thorough grasp of each system element's functionality, behavior, and dependencies within the larger system architecture, its design characteristics will be precisely described and documented. To help with implementation and integration, this entails describing the precise roles, responsibilities, inputs, outputs, and interactions of each component.

3.2 SYSTEM ARCHITECTURE

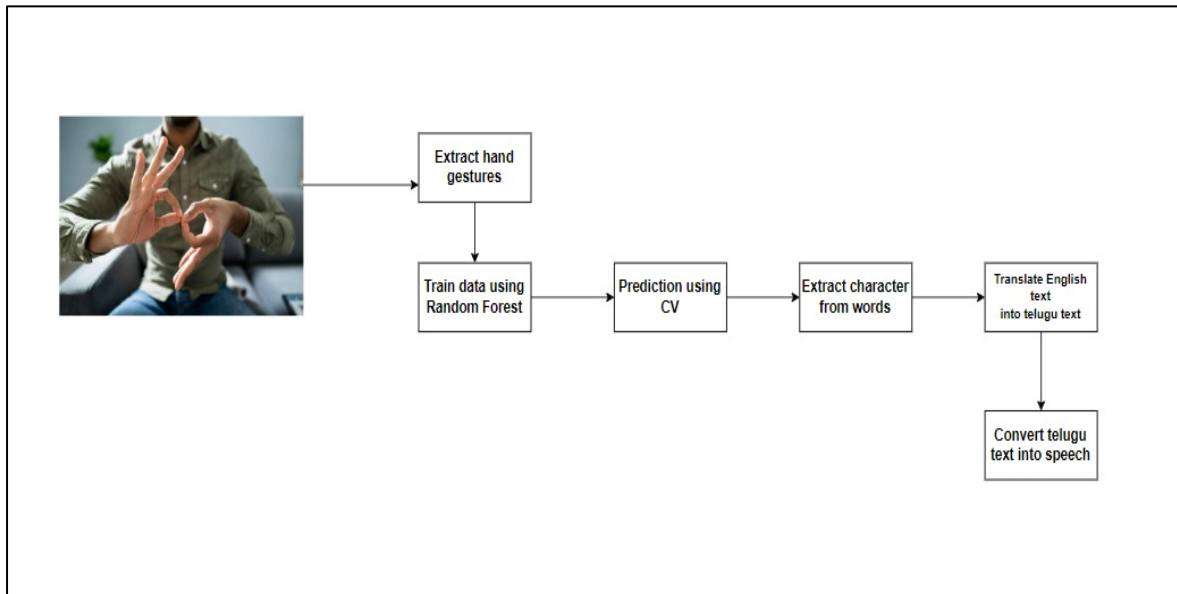


Figure 1: Architecture of sign language to speech

The sequential flow of data and system activities is depicted in the data flow diagram for the project:

- **Gathering Images:** To provide the system with input data, images showing sign language gestures are gathered.
- **Extraction of Hand Coordinates:** The fundamental elements of the sign language gestures are captured by extracting hand coordinates from the gathered photos using the Media Pipe library.
- **Saving Coordinates:** To make data storage and retrieval easier, the extracted hand coordinates are saved in a pickle file together with the dictionary-formatted class labels that correspond to them.
- **Training the Model:** The model learns and recognizes sign language motions by use of the Random Forest Classifier, which is trained using the saved hand coordinates and class labels.
- **Computing Accuracy:** The trained model's accuracy is computed to evaluate how well it recognizes sign language motions.
- **Preserving Trained Model:** To enable its utilization in real-time prediction scenarios in the future, the trained model is preserved in a pickle file.
- **Real-Time Prediction:** A trained model uses OpenCV to evaluate real-time sign language motions collected from live video feeds and predict the related text representations.
- **Character Extraction and Word Formation:** Subsequent processing is performed on the anticipated text representations in order to separate individual characters and create meaningful words or phrases.

- **Translation to Telugu:** To facilitate multilingual communication, the extracted English text representations are translated into Telugu using the Googletrans library.

Text-to-Speech Conversion: The GTTs (Google Text-to-Speech) library is used to convert the translated Telugu text into speech, producing audio output that corresponds to the identified sign language gestures.

3.3 DATAFLOWDIAGRAM

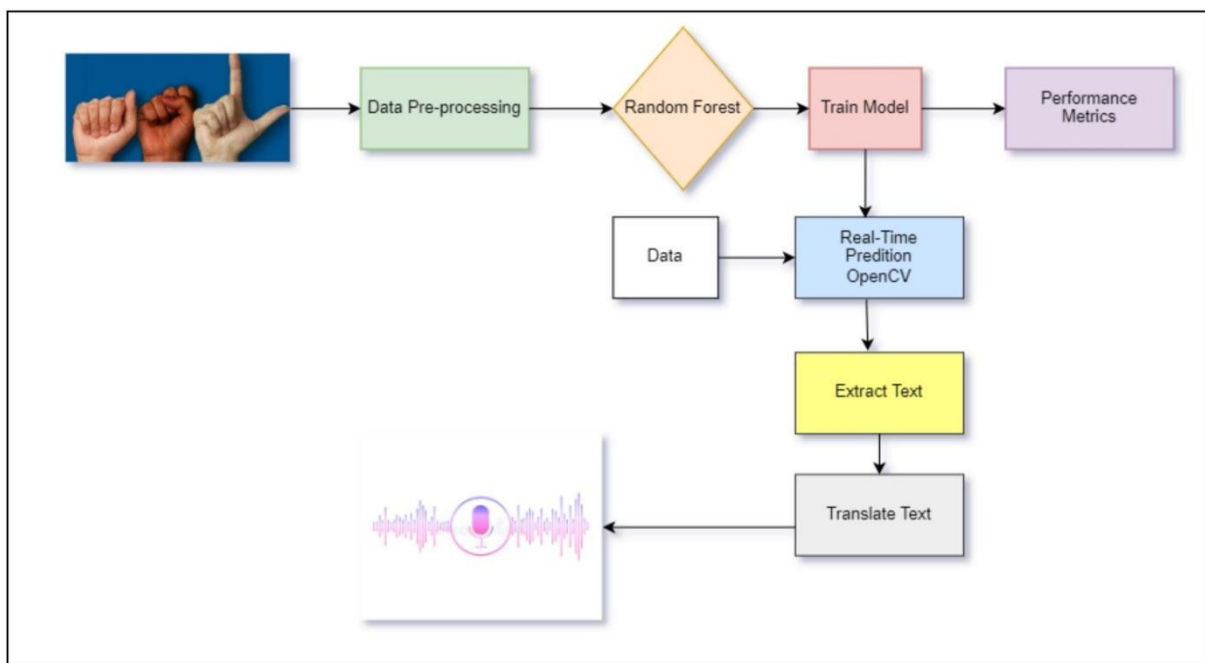


Figure 2: Flow of the sign language to speech

The following essential elements make up the sign language to speech project's system architecture:

- **American Sign Language images:** The main source of input data for the system is photos that represent gestures used in American Sign Language.
- **Data processing:** To get ready for more analysis, the input photos are preprocessed And feature extracted.
- **Algorithm:** Using processed data, the algorithm is used to train a machine learning model to identify and decipher sign language gestures.
- **Train Model:** After being trained, a machine learning model can reliably identify and categorize sign language motions into the appropriate text representations.
- **Performance Metrics:** To make sure the trained model is effective in identifying sign language movements, performance metrics are used to assess the model's efficiency and accuracy.
- **Real-Time Prediction:** The system makes use of OpenCV to predict sign language motions in real-time from live video sources.

- **Extract Text:** To facilitate additional processing and analysis, the identified sign language motions are converted into text representations.
- **Text to Speech Translation:** Lastly, text-to-speech synthesis is used to translate the translated text representations into spoken Telugu, facilitating natural conversation for people with communication impairments.

3.4 DATASETS:

American Sign Language (ASL) pictures with 52 classes make up the dataset for the sign language to speech project. The alphabetic letters A–Z, the numeric digits 0–9, and extra classes denoting certain words are among these classes. The ASL vocabulary is fully covered by the carefully chosen examples of sign language motions included in the collection.

The project's goal is to use this dataset to train a machine learning model that can recognize and understand a variety of ASL gestures with accuracy. This dataset provides the basis for training the model that will enable smooth conversion of gesticulations from sign language into spoken language, improving accessibility to communication for those with communication impairments.



Figure 3: ASL example image

SYSTEM IMPLEMENTATION

4. SYSTEM IMPLEMENTATION

4.1 SYSTEM SETUP:

The system setup for the sign language to voice project include setting up the required hardware and software parts to make the solution's development and implementation. This includes:

Hardware Setup: Make sure that the machine learning model can support its training and inference duties by having PCs or servers with enough memory and processing power. Make sure that input devices, including cameras, are accessible so that sign language gestures can be recorded.

Software Installation: Set up the necessary software libraries and packages, such as but not restricted to:

1. Numpy (np)
2. The Media Pipe library is used to manually extract coordinates from images.
3. OpenCV for preprocessing and real-time video analysis.
4. Googletrans is a text translation library.
5. Telugu text to speech conversion is possible with the GTTs (Google Text-to- Speech) library.
6. The Python programming environment is utilized for both system development and execution (Vscode)

Dataset Preparation: Ensure that the images in the American Sign Language (ASL) collection are properly labeled and formatted by organizing and preprocessing the dataset. To train and evaluate the model, divide the dataset into testing, validation, and training sets.

Model Training: Using the selected machine learning methods and the prepared dataset, train the Random Forest Classifier model. Adjust the model's hyper parameters as needed, then assess its performance with suitable metrics like accuracy, precision, recall, and F1-score.

Real-Time Prediction Setup: Set up the system such that live video streams can be used to record sign language motions and predict them in real-time. Utilize the Open CV framework to interpret video input and anticipate related text representations by integrating the trained model with it.

Translation and Speech Synthesis: To translate recognized text representations from English to Telugu, implement translation capability using the Googletrans library. To translate the Telugu text into speech and produce auditory output that corresponds to the identified sign language motions, use the Gtts library.

4.2 CODE:

4.2.1 Collecting Images:

```
import os
import cv2
import random
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
DATA_DIR = './Data'
ifnot os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)
WITH_LANDMARKS_DIR = os.path.join(DATA_DIR, 'With Landmarks')
ifnot os.path.exists(WITH_LANDMARKS_DIR):
    os.makedirs(WITH_LANDMARKS_DIR)
WITHOUT_LANDMARKS_DIR = os.path.join(DATA_DIR, 'Without Landmarks')
ifnot os.path.exists(WITHOUT_LANDMARKS_DIR):
    os.makedirs(WITHOUT_LANDMARKS_DIR)
number_of_classes = 3
users_per_class = 1
cap = cv2.VideoCapture(0)
# Initialize MediaPipe Hands model
with mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=2,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
    for j in range(number_of_classes):
        class_dir_with_landmarks = os.path.join(WITH_LANDMARKS_DIR, str(j))
        class_dir_no_landmarks = os.path.join(WITHOUT_LANDMARKS_DIR, str(j))
        ifnot os.path.exists(class_dir_with_landmarks):
            os.makedirs(class_dir_with_landmarks)
        ifnot os.path.exists(class_dir_no_landmarks):
            os.makedirs(class_dir_no_landmarks)
        print('Collecting data for class {}'.format(j))
        # landmarks_count = 1
        landmarks_count = int(input("Enter the landmarks count for class {} (1 or 2): ".format(j)))
        while landmarks_count notin [1, 2]:
            print("Invalid input. Please enter 1 or 2.")
            # landmarks_count = 1
            landmarks_count = int(input("Enter the landmarks count for class {} (1 or 2): ".format(j)))
            for user in range(users_per_class):
                user_dir = os.path.join(DATA_DIR, str(j))
                print(f'Collecting data for user {user + 1} in class {j + 0}')
                user_dir_with_landmarks = class_dir_with_landmarks
                user_dir_no_landmarks = class_dir_no_landmarks
```

```
# Display present user number and capturing class number
while True:
    ret, frame = cap.read()
    user_text = f'User: {user + 1}, Class: {j}'
    cv2.putText(frame, user_text, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255,
255), 2, cv2.LINE_AA)
    cv2.imshow('frame', frame)
    if cv2.waitKey(25) == ord('q'):
        break
# Capture a random number of images between 45 and 60
image_count = random.randint(150, 175)
counter = 0
# Find the last serial number used in the class
existing_serial_numbers = [int(filename.split('.')[0]) for filename in
os.listdir(user_dir_no_landmarks)]
last_serial_number = max(existing_serial_numbers) + 1 if existing_serial_numbers else 0
while counter < image_count:
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
    cv2.waitKey(25)
    # Convert the image to RGB before processing
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Process the image with MediaPipe Hands to detect landmarks
    results = hands.process(frame_rgb)
    # Check if landmarks are detected and count matches user input
    if results.multi_hand_landmarks and len(results.multi_hand_landmarks) ==
landmarks_count:
        cv2.imwrite(os.path.join(user_dir_no_landmarks, f'{last_serial_number}.jpg'), frame)
        # Draw landmarks on the image
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(
                frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)
        cv2.imwrite(os.path.join(user_dir_with_landmarks, f'{last_serial_number}.jpg'), frame)
        last_serial_number += 1
        counter += 1
cap.release()
cv2.destroyAllWindows()
```

4.2.2 Creating Dataset:

```
import os
import pickle
import mediapipe as mp
import cv2
mp_hands = mp.solutions.hands

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
DATA_DIR = './Data/Without Landmarks'
```

```
data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []
        x_ = []
        y_ = []
        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    x_.append(x)
                    y_.append(y)
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))
            # Check the shape of the data_aux
            if len(data_aux) == 84:
                # Flatten and normalize data_aux to (42,) shape
                normalized_data = [(item - min(data_aux)) / (max(data_aux) - min(data_aux)) for item in
data_aux]
                # Pad or truncate the data to ensure it has length 42
                if len(normalized_data) < 42:
                    normalized_data.extend([0] * (42 - len(normalized_data)))
                elif len(normalized_data) > 42:
                    normalized_data = normalized_data[:42]
                data.append(normalized_data)
                labels.append(dir_)
            elif len(data_aux) == 42: # If already 42, append as it is
                data.append(data_aux)
                labels.append(dir_)
f = open('data_ASL.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

4.2.3 Training Model:

```
import pickle
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
import numpy as np
data_dict = pickle.load(open('./data_ASL.pickle', 'rb'))
data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])
x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True,
stratify=labels)
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
score = accuracy_score(y_predict, y_test)
print('{}% of samples were classified correctly !'.format(score * 100))
f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

4.2.4Real time Prediction:

```
from flask import Flask, render_template, Response, request, jsonify
import cv2
import numpy as np
import time
import pickle
import pygame
from gtts import gTTS
from googletrans import Translator
import mediapipe as mp
import datetime
import requests
import csv
from difflib import get_close_matches

# Load suggestions from words.csv
def load_suggestions():
    suggestions = {}
    with open('words.csv', 'r', newline='', encoding='utf-8') as file:
        reader = csv.reader(file)
        for row in reader:
            if row: # Check if the row is not empty
                word = row[0]
                first_letter = word[0].upper()
                if first_letter in suggestions:
                    if word not in suggestions[first_letter]:
                        suggestions[first_letter].append(word)
                else:
                    suggestions[first_letter] = [word]
    return suggestions
```

```
suggestion_words = load_suggestions()
# print(suggestion_words)

def find_similar_words(user_input, suggestions):
    similar_words = []
    first_letter = user_input[0].upper()
    if first_letter in suggestions:
        similar_words = get_close_matches(user_input, suggestions[first_letter],
n=3)
    return similar_words

app = Flask(__name__)

translator = Translator()
pygame.mixer.init()
model_dict = pickle.load(open('../Create Dataset/model.p', 'rb'))
model = model_dict['model']
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

labels_dict = {
    0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9',
    10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F', 16: 'G', 17: 'H', 18: 'I',
19: 'J',
    20: 'K', 21: 'L', 22: 'M', 23: 'N', 24: 'O', 25: 'P', 26: 'Q', 27: 'R', 28: 'S',
29: 'T',
    30: 'U', 31: 'V', 32: 'W', 33: 'X', 34: 'Y', 35: 'Z',
    36: 'home', 37: 'brother', 38: 'pay', 39: 'justice', 40: 'science',
    41: 'Please', 42: 'Hello', 43: 'You are Welcome', 44: 'Good Bye',
    45: 'sorry', 46: 'yes', 47: 'no', 48: 'thanks', 49: 'what',
    50: 'sad', 51: ' ', 52: 'book', 53: 'sleep'
}

prediction_started = False
last_prediction_time = 0
output = ""

@app.route('/')
def index():
    return render_template('index.html')

def generate_frames():
    global prediction_started, last_prediction_time, output
    cap = cv2.VideoCapture(0)
    while True:
```

```
data_aux = []
x_ = []
y_ = []
ret, frame = cap.read()
H, W, _ = frame.shape
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = hands.process(frame_rgb)
if results.multi_hand_landmarks and prediction_started:
    for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            frame, # image to draw
            hand_landmarks, # model output
            mp_hands.HAND_CONNECTIONS, # hand connections
            mp_drawing_styles.get_default_hand_landmarks_style(),
            mp_drawing_styles.get_default_hand_connections_style())

    for hand_landmarks in results.multi_hand_landmarks:
        for i in range(len(hand_landmarks.landmark)):
            x = hand_landmarks.landmark[i].x
            y = hand_landmarks.landmark[i].y

            x_.append(x)
            y_.append(y)

        for i in range(len(hand_landmarks.landmark)):
            x = hand_landmarks.landmark[i].x
            y = hand_landmarks.landmark[i].y
            data_aux.append(x - min(x_))
            data_aux.append(y - min(y_))

x1 = int(min(x_) * W) - 10
y1 = int(min(y_) * H) - 10

x2 = int(max(x_) * W) - 10
y2 = int(max(y_) * H) - 10

current_time = time.time()

if current_time - last_prediction_time > 4:
    if len(data_aux) == 84: # Check if data_aux has 84 features
        # Flatten and normalize data_aux to 42 features
        flattened_data = data_aux[:42] # Use the first 42 elements
        normalized_data = [(item - min(flattened_data)) /
            (max(flattened_data) - min(flattened_data))] for item in flattened_data
        prediction = model.predict([np.asarray(normalized_data)]) #
        Predict using the normalized data
    else:
```

```
        prediction = model.predict([np.asarray(data_aux)]) # Predict
using the original data_aux

        predicted_character = labels_dict[int(prediction[0])]
        output += predicted_character
        last_prediction_time = current_time

        # Send AJAX request to update web page output (new)
        data = {'output': output}
        response = requests.post('http://localhost:5000/update_output',
json=data) # Use requests library

        # print("Current Prediction:", predicted_character)
        # print("Current Output:", output)

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
        # cv2.putText(frame, output, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
1.3, (0, 0, 0), 3, cv2.LINE_AA)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')

@app.route('/start', methods=['POST'])
def start_prediction():
    global prediction_started
    prediction_started = True
    return output

@app.route('/stop', methods=['POST'])
def stop_prediction():
    global prediction_started, output
    prediction_started = False
    translation = translator.translate(output, src='en', dest='te')
    translated_text = translation.text

    # Get current date and time
    now = datetime.datetime.now()
    timestamp = now.strftime("%Y%m%d_%H%M%S") # Format: YYYYMMDD_HHMMSS

    # Save audio file with timestamp appended to its name
    audio_filename = f'output_{timestamp}.mp3'
```



```
tts = gTTS(text=translated_text, lang='te')
tts.save(audio_filename)

# Play the latest audio file
pygame.mixer.music.load(audio_filename)
pygame.mixer.music.play()

output = "" # Reset output after translation
return translated_text

@app.route('/backspace', methods=['POST'])
def backspace():
    global output
    output = output[:-1]
    return output

@app.route('/addspace', methods=['POST'])
def addspace():
    global output
    output = output+" "
    return output

@app.route('/update_output', methods=['GET'])
def update_output(): #suggestions
    global output, suggestion_words
    if output!="":
        search_word = output.split()[-1] if ' ' in output else output
        # print(search_word)
        suggestions = find_similar_words(search_word, suggestion_words)
        # print(suggestions)
    return jsonify({'output': output, 'suggestions': suggestions})

@app.route('/addsuggestion', methods=['POST'])
def addsuggestion():
    global output
    suggestion = request.json['suggestion']

    # Replace the last word in output with the clicked suggestion
    output_list = output.split()
    if output_list:
        output_list[-1] = suggestion
        output = " ".join(output_list)
    else:
        output = suggestion

    return output
```

```
if __name__ == "__main__":  
    app.run(debug=True)
```

4.2.5 Templates/index.html :

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Hand Gesture Recognition</title>  
    <!-- Bootstrap CSS -->  
    <link rel="stylesheet"  
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">  
</head>  
<body>  
    <div class="container">  
        <h1 class="mt-4">Hand Gesture Recognition</h1>  
        <div class="row">  
            <div class="col-6">  
                <div id="video-container">  
                      
                </div>  
            </div>  
            <div class="col-6">  
                <p>Predicted Output:</p>  
                <div class="row">  
                    <div class="col-12">  
                        <div class="input-group mb-3">  
                            <input type="text" class="form-control"  
id="predicted_output" readonly>  
                            <div class="input-group-append">  
                                <button class="btn btn-outline-secondary"  
type="button" onclick="backspace()">Backspace</button>  
                                <button class="btn btn-outline-secondary"  
type="button" onclick="addSpace()">Space</button>  
                            </div>  
                        </div>  
                        <div id="suggestion-buttons">  
                            <!-- Three buttons for suggestions -->  
                            <div class="row text-center">
```

```
        <div class="col-4">
            <button class="btn btn-outline-primary w-100"
id="suggestion1" onclick="addSuggestion(this.innerText)"></button>
        </div>
        <div class="col-4">
            <button class="btn btn-outline-primary w-100"
id="suggestion2" onclick="addSuggestion(this.innerText)"></button>
        </div>
        <div class="col-4">
            <button class="btn btn-outline-primary w-100"
id="suggestion3" onclick="addSuggestion(this.innerText)"></button>
        </div>
    </div>
</div>
<br><br>
<div class="row text-center">
    <div class="col-4">
        <button class="btn btn-primary mr-2 w-100"
onclick="startPrediction()">Start</button>
    </div>
    <div class="col-4">
        <button class="btn btn-danger mr-2 w-100"
onclick="stopPrediction()">Stop</button>
    </div>
</div>
</div>
<br>
<div class="col-12">
    <!-- Display translated output dynamically -->
    <div id="translated-output" class="m-5" style="display:
none;"></div><br>
    <!-- Play audio dynamically -->
    <audio id="output-audio" controls style="display:
none;"></audio>
    </div>
</div>
</div>
</div>
</div>
</div>

<!-- Bootstrap JS and jQuery (required for Bootstrap functionality) -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></scr
ipt>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></scrip
t>
```

```
<script>
    var outputElement = document.getElementById('predicted_output');
    var translatedOutputElement = document.getElementById('translated-output');
    var audioElement = document.getElementById('output-audio');

    function startPrediction() {
        outputElement.value = ""; // Clear the output element
        translatedOutputElement.style.display = "none"; // Hide translated
output
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/start");
        xhr.send();
    }

    // Function to handle output updates received via AJAX
    function updateOutput(newOutput) {
        outputElement.value = newOutput;
    }

    function addSpace() {
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/addspace");
        xhr.send();

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                updateOutput(xhr.responseText); // Add a space after backspace
            }
        }
    }

    function backspace() {
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/backspace");
        xhr.send();

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                updateOutput(xhr.responseText); // Update output after backspace
            }
        }
    }

    function stopPrediction() {
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/stop");
        xhr.send();
    }
</script>
```

```
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        var translation = xhr.responseText;
        translatedOutputElement.innerHTML = "Translated Text: " +
translation;
        translatedOutputElement.style.display = "block"; // Display
translated output
        audioElement.src = 'output.mp3';
        audioElement.style.display = "none";
        audioElement.play();
    }
}

// Function to handle updating the output dynamically
function updateOutputDynamically(output) {
    outputElement.value = output;
}

function addSuggestion(suggestion) {
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/addsuggestion");
    xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    xhr.send(JSON.stringify({suggestion: suggestion}));

    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
            updateOutput(xhr.responseText); // Update output with suggestion
text
        }
    }
}

// Polling for updates
setInterval(function() {
    fetch('/update_output')
        .then(response => response.json())
        .then(data => {
            updateOutputDynamically(data.output);
            if (data.suggestions.length > 0) {
                // Update suggestion buttons
                for (var i = 0; i < 3; i++) {
                    var suggestionBtn = document.getElementById('suggestion'
+ (i + 1));

                    if (i < data.suggestions.length) {
                        suggestionBtn.innerHTML = data.suggestions[i];
```

```
        suggestionBtn.style.display = "block";
    } else {
        suggestionBtn.style.display = "none";
    }
}
} else {
    // Hide all suggestion buttons if no suggestions
    for (var i = 0; i < 3; i++) {
        var suggestionBtn = document.getElementById('suggestion'
+ (i + 1));

        suggestionBtn.style.display = "none";
    }
}
});
}, 1000); // Update every second
</script>
</body>
</html>
```

4.3 RESULTS:

	precision	recall	f1-score	support
0	0.95	1.00	0.97	37
1	1.00	0.97	0.99	40
10	1.00	0.98	0.99	53
11	1.00	1.00	1.00	51
12	1.00	1.00	1.00	50
13	1.00	1.00	1.00	48
14	1.00	1.00	1.00	53
15	1.00	1.00	1.00	51
16	1.00	0.98	0.99	42
17	1.00	1.00	1.00	50
18	1.00	1.00	1.00	50
19	0.98	1.00	0.99	58
2	1.00	1.00	1.00	55
20	1.00	0.98	0.99	53
21	1.00	1.00	1.00	43
22	1.00	0.97	0.99	40
23	0.95	0.98	0.96	41
24	1.00	1.00	1.00	51
25	1.00	0.98	0.99	44
26	1.00	1.00	1.00	44
27	1.00	1.00	1.00	51
28	1.00	1.00	1.00	54
29	0.98	1.00	0.99	55
3	1.00	1.00	1.00	55
30	1.00	1.00	1.00	53
31	1.00	1.00	1.00	52
32	1.00	1.00	1.00	48
33	1.00	1.00	1.00	54
34	1.00	1.00	1.00	52
35	1.00	0.96	0.98	27
36	1.00	1.00	1.00	14
37	1.00	1.00	1.00	12
38	1.00	1.00	1.00	33
39	1.00	1.00	1.00	31
4	1.00	1.00	1.00	48
40	1.00	1.00	1.00	30
41	1.00	1.00	1.00	34
42	1.00	1.00	1.00	30
43	1.00	1.00	1.00	32
44	1.00	1.00	1.00	32
45	1.00	1.00	1.00	35
46	1.00	1.00	1.00	32
47	1.00	1.00	1.00	9
48	1.00	1.00	1.00	2
49	1.00	1.00	1.00	31
5	0.98	1.00	0.99	55
50	1.00	1.00	1.00	35
51	1.00	1.00	1.00	34
6	1.00	1.00	1.00	39
7	1.00	1.00	1.00	54
8	0.98	1.00	0.99	49
9	1.00	1.00	1.00	52
accuracy			1.00	2178
macro avg	1.00	1.00	1.00	2178
weighted avg	1.00	1.00	1.00	2178

Figure 4: Performance metrics results for model

The sign language to speech system, with 99% accuracy in classifying gestures, has the potential to significantly improve communication between sign language users and non-users.

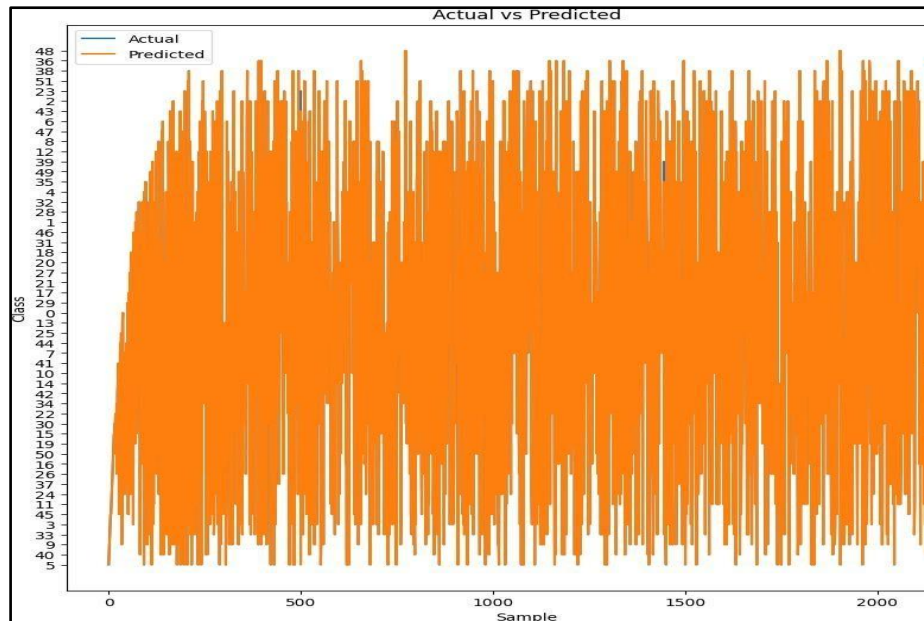


Figure 5: Actual & Predicted results

The system generates a graphical representation displaying both actual and predicted results.

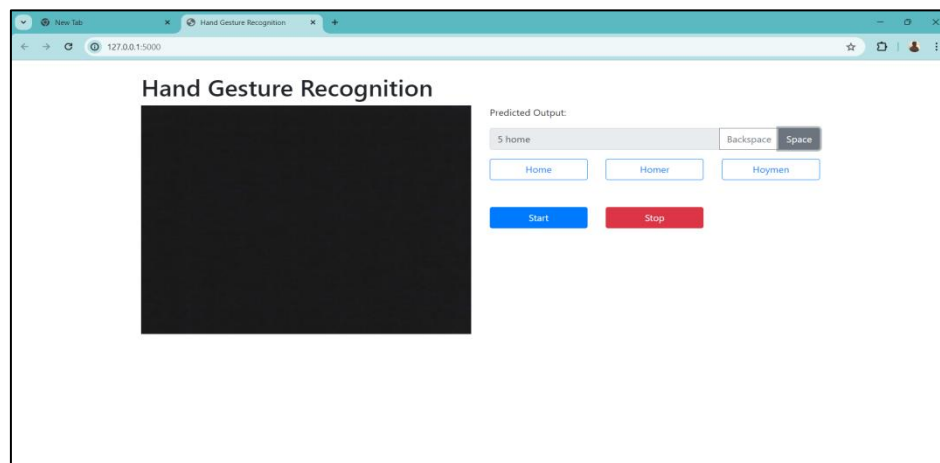


Figure 6: Final output of web interface

The system processes input sign language, accurately predicts gestures, and converts them into English text. This text is then translated into Telugu language text before being synthesized into Telugu language speech, facilitating seamless communication for users.

TESTING

5. TESTING

5.1 PERFORMANCE METRICS:

Accuracy: When assessing a model's performance, accuracy is a crucial indicator that shows the percentage of accurate predictions compared to all of the predictions made. It functions as an all-encompassing indicator of how well the model performs overall in accurately identifying occurrences across all dataset classifications.

Formula:

$$\text{Accuracy} = \text{Number of correct predictions} / \text{Total number of predictions.}$$

Error rate: The ratio of the total number of wrong forecasts to the total number of predictions made by the model is the error rate, a statistic that expresses the percentage of incorrect predictions made by a model.

Formula:

$$\text{Error Rate} = 1 - \text{Accuracy}$$

Precision: Precision is a metric that expresses how well the model predicts positive outcomes; it is the ratio of accurately identified positive cases to the total number of positive predictions. It evaluates the model's capacity to reduce false positive mistakes and guarantee that the positive occurrences that are predicted are, in fact, relevant.

Formula:

$$\text{Precision} = \text{True Positives} / (\text{False Positives} + \text{True Positives})$$

Recall: By calculating the percentage of accurately detected positive occurrences among all real positive instances in the dataset, recall, sometimes referred to as sensitivity, assesses the model's capacity to capture all positive cases.

Formula:

$$\text{Recall} = \text{True Positives} / (\text{False Negatives} + \text{True Positives})$$






F1-score: The model's performance is balanced by combining precision and recall into a single metric called the F1-score. It makes sure that both metrics are taken into account equally. It is the harmonic mean of recall and precision.

Formula:

$$\text{F1-score} = 2 \times (\text{Precision} + \text{Recall} / \text{Precision} \times \text{Recall})$$

5.2 VALIDATION OF TEST CASES:

Table2: Validation Of Test Cases

TEST CASE NUMBER	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
1		సహాయం	సహాయం	True
2		అలాగే	అలాగే	True
3		ఎ	ఎ	True
4		కాదు	కాదు	True
5		అవును	అవును	True

CONCLUSION

6. CONCLUSION

Our sign language to speech project successfully developed a system that translates sign language gestures into spoken Telugu language, facilitating seamless communication between who cannot speak or dumb people and those who speak. By leveraging the Random Forest algorithm for gesture recognition and speech synthesis, we have created a user - friendly solution that promotes inclusivity and accessibility for people who can not speak or dumb. This model has the accuracy is 99%. This project not only addresses the communication barriers but also fosters understanding and social interaction, ultimately enhancing the quality of life for this demographic.

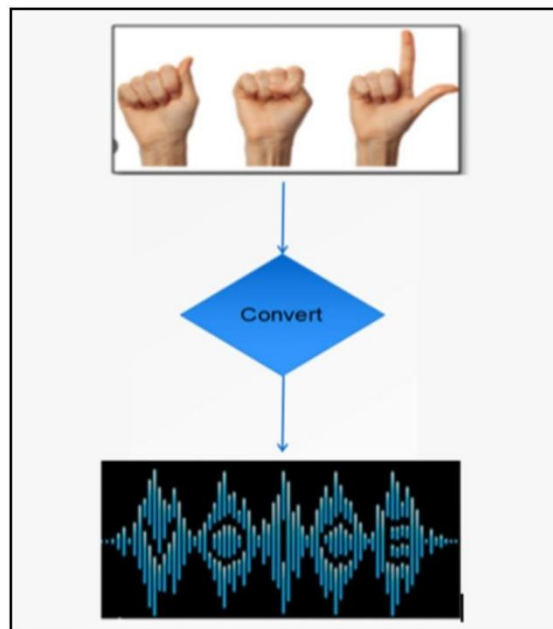


Figure 7: Converting sign language to voice

FUTUREWORK

7. FUTUREWORK

In the future, broadening the scope of the project to incorporate an intuitive mobile application or web interface would improve accessibility and usability for a larger user base. Furthermore, including more sophisticated machine learning methods like deep learning models inside Random Forest could enhance efficiency even further. Enhancing the gesture recognition and voice synthesis capabilities of the system can be accomplished by working with specialists in sign language linguistics and human-computer interaction. For those with communication impairments, the technology will continue to improve communication and quality of life through ongoing development and updates based on user feedback and technological improvements.

REFERENCES
&
BIBLIOGRAPHY

8. REFERENCES

- [1] Akshatha Rani K; Dr. N Manjanaik; Sign Language to Text-Speech Translator Using Machine Learning
- [2] Tiku, Kohsheen; Maloo, Jayshree; Ramesh, Aishwarya; R, Indra (2020). [IEEE 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) - Coimbatore, India (2020.7.15-2020.7.17)] 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) - Real-time Conversion of Sign Language to Text and Speech. , (),346–351. doi:10.1109/ICIRCA48905.2020.9182877
- [3] K Amrutha;P Prabu; (2021). ML Based Sign Language Recognition System . 2021 International Conference on Innovative Trends in Information Technology (ICITIIT), (), –. doi:10.1109/icitiit51526.2021.9399594
- [4] Salma A. Essam El-Din;Mohamed A. Abd El-Ghany; (2020). Sign Language Interpreter System: An alternative system for machine learning . 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), (), –.doi:10.1109/niles50944.2020.9257958
- [5] Ayush Pandey; Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network
- [6] Ashok Kumar Sahoo; (2021). Indian Sign Language Recognition Using Machine Learning Techniques . Macromolecular Symposia, (), –. doi:10.1002/masy.202000241
- [7] Bayan Mohammed Saleh; D-Talk: Sign Language Recognition System for People with Disability using Machine Learning and Image Processing
- [8] Narayana Darapaneni;Prasad Gandole;Sureshkumar Ramasamy;Yashraj Tambe;Anshuman Dwivedi;Anwesh Reddy Paduri;Vihan Parmar;Kirthi Krishnan Ganeshan; (2021). American Sign Language Detection Using Instance-Based Segmentation . 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), (), –. doi:10.1109/iemtronics52119.2021.9422601
- [9] V., Adithya; R., Rajesh (2020). A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition. Procedia Computer Science, 171(), 2353–2361. doi:10.1016/j.procs.2020.04.255
- [10] Saxena, Ankita; Jain, Deepak Kumar; Singhal, Ananya (2014). [IEEE 2014 International Conference on Communication Systems and Network Technologies (CSNT) - Bhopal, India (2014.04.7-2014.04.9)] 2014 Fourth International Conference on Communication Systems and Network Technologies - Sign Language Recognition Using Principal Component Analysis. , (), 810–813. doi:10.1109/CSNT.2014.168

9. BIBLIOGRAPHY

Machine Learning Libraries:

CV2 (OpenCV)

media pipe

os

Random

pickle

scikit-learn

numpy(np) time

gttspygame

googletrans

Flask

Paper References:

Google Scholar: <https://scholar.google.com/>

Sci-hub: <https://sci-hub.hkvisa.net/>

Dataset: American Sign Language Images (own dataset)

APPENDIX

10. APPENDIX

10.1 Python Introduction:

Python is a versatile, high-level programming language known for its simplicity and readability. It's widely used in various fields like web development, data analysis, machine learning, and more, thanks to its extensive libraries and frameworks.

Versatility: Python is a versatile language used in web development, data science, artificial intelligence, scientific computing, and more.

Readability: Its clean and easy-to-understand syntax makes Python suitable for beginners and experienced developers alike.

Large Community and Ecosystem: Python has a vast community of developers contributing to its ecosystem, resulting in a rich selection of libraries and frameworks.

Interpreted and Interactive: Python is an interpreted language, allowing for quick prototyping and interactive coding through tools like Jupyter Notebooks.

Strong Support for Data Analysis and Visualization: Libraries like NumPy, Pandas, and Matplotlib make Python a top choice for data analysis and visualization tasks.

Machine Learning and AI: Python's libraries like TensorFlow, PyTorch, and scikit-learn have made it a dominant force in machine learning and artificial intelligence research and development.

Web Development: Frameworks like Django and Flask enable rapid development of web applications, making Python a popular choice for web development.

Cross-Platform Compatibility: Python runs on various platforms like Windows, macOS, and Linux, ensuring code portability across different environments.

Community Support and Documentation: Python has extensive documentation and a supportive community, making it easy to find solutions to problems and learn new concepts.

Open Source: Python is open source, allowing developers to contribute to its development and customize it to suit their needs.

10.2 Machine Learning:

The goal of ML is to create models and algorithms that let computers learn from data and make judgments or predictions without needing to be explicitly programmed. ML comprises a wide range of methods, such as reinforcement learning, unsupervised learning, and supervised learning, each suited for a particular set of tasks and data. Applications for it can be found in many different fields, such as recommendation systems, natural language processing, autonomous cars, and

image and audio recognition. In order to identify patterns and insights from data, ML algorithms make use of statistical methodologies and optimization techniques. This allows systems to perform better over time and adapt to changing settings. ML is becoming more and more essential in facilitating automation, personalization, and decision-making across a broad range of sectors and domains as data availability and processing power increase.

ML Libraries:

cv2 (OpenCV): OpenCV is used to process videos in real-time and to record sign language movements from live video streams.

media pipe: The Media Pipe library is used to extract hand coordinates from pictures, making it easier to analyze gestures used in sign language.

scikit-learn: Scikit-learn is a machine learning package that is used to assess model performance and implement the Random Forest technique.

gtts: The gtts library is used to translate text to voice, allowing spoken output to be synthesized that corresponds to detected motions in sign language.

googletrans: The Googletrans library is used to provide text translation capabilities, making it easier to translate known English to Telugu text representations.

Speech Synthesis: The gtts library is used to implement speech synthesis capability. It transforms translated text representations into audible speech output. This facilitates effective communication for people with communication difficulties by generating spoken Telugu language that corresponds to recognized sign language motions.

Random Forest approach: Using extracted hand coordinates, a machine learning model that can recognize and understand sign language motions is trained using the Random Forest approach. To increase accuracy and resilience, it builds an ensemble of decision trees and combines their predictions.

Flask: Flask is a lightweight web framework for Python, designed to make it easy to build web applications quickly and with minimal boilerplate code.