A UNIFIED COMMUNICATION SYSTEM TO BRIDGE THE GAP FOR INCLUSIVE INTERACTION AMONG INDIVIDUALS WITH AND WITHOUT HEARING AND SPEAKING ABILITIES

A PROJECT REPORT

Submitted by

NANDHINI S (1920102084)
PAVITHRA S (1920102097)
RITHIKAEZHIL N (1920102111)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SONA COLLEGE OF TECHNOLOGY
(An Autonomous Institution)
SALEM-636005

ANNA UNIVERSITY: CHENNAI-600 025

MAY 2024

SONA COLLEGE OF TECHNOLOGY

(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"A UNIFIED COMMUNICATION SYSTEM TO BRIDGE THE GAP FOR INCLUSIVE INTERACTION AMONG INDIVIDUALS WITH AND WITHOUT HEARING AND SPEAKING ABILITIES"** is the bonafide work of **"NANDHINI S (1920102084), PAVITHRA S (1920102097), RITHIKAEZHIL N (1920102111)"** who carried out the project work under my supervision.

SIGNATURE
Dr. B. SATHIYABHAMA, B.E., M.Tech., Ph.D

HEAD OF THE DEPARTMENT
Professor
Department of Computer Science and Engineering
Sona College of Technology,
Salem – 636005

SIGNATURE
Dr. S. ANITHA ELAVARASI, B.E., Ph.D

SUPERVISOR
Associate Professor
Department of Computer Science and Engineering
Sona College of Technology,
Salem – 636005

Submitted for the project viva-voce examination held on ……..

INTERNAL EXAMINER

EXTERNAL EXAMINER

ICGCCT 2024

INTERNATIONAL CONFERENCE ON

GREEN COMPUTING FOR COMMUNICATION TECHNOLOGIES

Organised by the

Departments of

Computer Science & Engineering

Electronics and Communication Engineering

Information Technology

&

Master of Computer Applications

in Association with

MUSCAT College Oman

SONA COLLEGE OF TECHNOLOGY

Learning is a Celebration!

| An Autonomous Institution |

Excellence in EDUCATION

Muscat College

Sl. No.: SCT/2024/03/ICGCCT2024/831

CERTIFICATE OF PRESENTATION

This is to certify that

Ms.Nandhini S

Student, Department of Computer Science and Engineering

of

Sona College of Technology

has participated / presented a paper titled

A Unified Communication System to Bridge the Gap for Inclusive Interaction among Individuals with and without Hearing and Speaking Abilities

at the International Conference on

Green Computing for Communication Technologies - ICGCCT 2024

organised by Sona College of Technology, Salem, in association with Muscat College, Oman, held on 6 & 7 March 2024

Coordinator
Muscat College, Oman

Coordinator
Sona College of Technology

Principal
Sona College of Technology

ICGCCT 2024

INTERNATIONAL CONFERENCE ON

GREEN COMPUTING FOR COMMUNICATION TECHNOLOGIES

Organised by the

Departments of

Computer Science & Engineering

Electronics and Communication Engineering

Information Technology

&

Master of Computer Applications

in Association with

MUSCAT College Oman

SONA COLLEGE OF TECHNOLOGY

Learning is a Celebration!

| An Autonomous Institution |

Excellence in EDUCATION

Muscat College

Sl. No.: SCT/2024/03/ICGCCT2024/832

CERTIFICATE OF PRESENTATION

This is to certify that

Ms.Pavithra S

Student, Department of Computer Science and Engineering

of

Sona College of Technology

has participated / presented a paper titled

A Unified Communication System to Bridge the Gap for Inclusive Interaction among Individuals with and without Hearing and Speaking Abilities

at the International Conference on

Green Computing for Communication Technologies - ICGCCT 2024

organised by Sona College of Technology, Salem, in association with Muscat College, Oman, held on 6 & 7 March 2024

Coordinator
Muscat College, Oman

Coordinator
Sona College of Technology

Principal
Sona College of Technology

SONA COLLEGE OF TECHNOLOGY
Learning is a Celebration!
| An Autonomous Institution |

Muscat College

ICGCCT 2024
INTERNATIONAL CONFERENCE ON
GREEN COMPUTING FOR COMMUNICATION TECHNOLOGIES

Organised by the
Departments of
Computer Science & Engineering
Electronics and Communication Engineering
Information Technology
&
Master of Computer Applications

in Association with
MUSCAT College
Oman

Sl. No.: SCT/2024/03/ICGCCT2024/833

CERTIFICATE OF PRESENTATION

This is to certify that

Ms.Rithikaezhil N

Student, Department of Computer Science and Engineering

of

Sona College of Technology

has participated / presented a paper titled

A Unified Communication System to Bridge the Gap for Inclusive Interaction among Individuals with and without Hearing and Speaking Abilities

at the International Conference on

Green Computing for Communication Technologies - ICGCCT 2024

organised by Sona College of Technology, Salem, in association with Muscat College, Oman, held on 6 & 7 March 2024

Coordinator
Muscat College, Oman

Coordinator
Sona College of Technology

Principal
Sona College of Technology

AIP Publishing

LICENSE TO PUBLISH AGREEMENT FOR CONFERENCE PROCEEDINGS

This License to Publish must be signed and returned to the Proceedings Editor before the manuscript can be published. If you have questions about how to submit the form, please contact the AIP Publishing Conference Proceedings office (confproc@aip.org). For questions regarding the copyright terms and conditions of this License, please contact AIP Publishing's Office of Rights and Permissions, 1305 Walt Whitman Road, Suite 300, Melville, NY 11747-4300 USA; Phone 516-576-2268; Email: rights@aip.org.

Article Title ("Work"): A UNIFIED COMMUNICATION SYSTEM TO BRIDGE THE GAP FOR INCLUSIVE INTERACTION AMONG INDIVIDUALS WITH AND WITHOUT HEARING AND SPEAKING ABILITIES

(Please indicate the final title of the Work. Any substantive changes made to the title after acceptance of the Work may require the completion of a new agreement.)

All Author(s): S. ANITHA ELAVARASI, S. NANDHINI, S. PAVITHRA, N. RITHIKA EZHIL

(Please list **all** the authors' names in order as they will appear in the Work. All listed authors must be fully deserving of authorship and no such authors should be omitted. For large groups of authors, attach a separate list to this form.)

Title of Conference: INTERNATIONAL CONFERENCE ON GREEN COMPUTING FOR COMMUNICATION TECHNOLOGIES (IGCCT 2024)

Name(s) of Editor(s): DR. R.S. SABEENIAN, DR. M.E. PARAMASIVAM, DR. P.M. DINESH

All Copyright Owner(s), if not Author(s):

(Please list **all** copyright owner(s) by name. In the case of a Work Made for Hire, the employer(s) or commissioning party(ies) are the copyright owner(s). For large groups of copyright owners, attach a separate list to this form.)

Copyright Ownership and Grant of Rights

For the purposes of this License, the "Work" consists of all content within the article itself and made available as part of the article, including but not limited to the abstract, tables, figures, graphs, images, and multimedia files, as well as any subsequent errata. "Supplementary Material" consists of material that is associated with the article but linked to or accessed separately (available electronically only), including but not limited to data sets and any additional files.

This Agreement is an Exclusive License to Publish not a Transfer of Copyright. Copyright to the Work remains with the Author(s) or, in the case of a Work Made for Hire, with the Author(s)' employer(s). AIP Publishing LLC shall own and have the right to register in its name the copyright to the proceedings issue or any other collective work in which the Work is included. Any rights granted under this License are contingent upon acceptance of the Work for publication by AIP Publishing. If for any reason and at its own discretion AIP Publishing decides not to publish the Work, this License is considered void.

Each Copyright Owner hereby grants to AIP Publishing LLC the following irrevocable rights for the full term of United States and foreign copyrights (including any extensions):

1. The exclusive right and license to publish, reproduce, distribute, transmit, display, store, translate, edit, adapt, and create derivative works from the Work (in whole or in part) throughout the world in all formats and media whether now known or later developed, and the nonexclusive right and license to do the same with the Supplementary Material.
2. The right for AIP Publishing to freely transfer and/or sublicense any or all of the exclusive rights listed in #1 above. Sublicensing includes the right to authorize requests for reuse of the Work by third parties.
3. The right for AIP Publishing to take whatever steps it considers necessary to protect and enforce, at its own expense, the exclusive rights granted herein against third parties.

Author Rights and Permitted Uses

Subject to the rights herein granted to AIP Publishing, each Copyright Owner retains ownership of copyright and all other proprietary rights such as patent rights in the Work.

Each Copyright Owner retains the following nonexclusive rights to use the Work, without obtaining permission from AIP Publishing, in keeping with professional publication ethics and provided clear credit is given to its first publication in an AIP Publishing proceeding. Any reuse must include a full credit line acknowledging AIP Publishing's publication and a link to the Version of Record (VOR) on AIP Publishing's site.

Each Copyright Owner may:

1. Reprint portions of the Work (excerpts, figures, tables) in future works created by the Author, in keeping with professional publication ethics.
2. Post the Accepted Manuscript (AM) to their personal web page or their employer's web page immediately after acceptance by AIP Publishing.
3. Deposit the AM in an institutional or funder-designated repository immediately after acceptance by AIP Publishing.
4. Use the AM for posting within scientific collaboration networks (SCNs). For a detailed description of our policy on posting to SCNs, please see our Web Posting Guidelines (https://publishing.aip.org/authors/web-posting-guidelines).
5. Reprint the Version of Record (VOR) in print collections written by the Author, or in the Author's thesis or dissertation. It is understood and agreed that the thesis or dissertation may be made available electronically on the university's site or in its repository and that copies may be offered for sale on demand.
6. Reproduce copies of the VOR for courses taught by the Author or offered at the institution where the Author is employed, provided no fee is charged for access to the Work.
7. Use the VOR for internal training and noncommercial business purposes by the Author's employer.
8. Use the VOR in oral presentations made by the Author, such as at conferences, meetings, seminars, etc., provided those receiving copies are informed that they may not further copy or distribute the Work.
9. Distribute the VOR to colleagues for noncommercial scholarly use, provided those receiving copies are informed that they may not further copy or distribute the Work.
10. Post the VOR to their personal web page or their employer's web page 12 months after publication by AIP Publishing.
11. Deposit the VOR in an institutional or funder-designated repository 12 months after publication by AIP Publishing.
12. Update a prior posting with the VOR on a noncommercial server such as arXiv, 12 months after publication by AIP Publishing.

Author Warranties

Each Author and Copyright Owner represents and warrants to AIP Publishing the following:

1. The Work is the original independent creation of each Author and does not infringe any copyright or violate any other right of any third party.
2. The Work has not been previously published and is not being considered for publication elsewhere in any form, except as a preprint on a noncommercial server such as arXiv, or in a thesis or dissertation.
3. Written permission has been obtained for any material used from other sources and copies of the permission grants have been supplied to AIP Publishing to be included in the manuscript file.
4. All third-party material for which permission has been obtained has been properly credited within the manuscript.
5. In the event that the Author is subject to university open access policies or other institutional restrictions that conflict with any of the rights or provisions of this License, such Author has obtained the necessary waiver from his or her university or institution.

This License must be signed by the Author(s) and, in the case of a Work Made for Hire, also by the Copyright Owners. One Author/Copyright Owner may sign on behalf of all the contributors/owners only if they all have authorized the signing, approved of the License, and agreed to be bound by it. The signing Author and, in the case of a Work Made for Hire, the signing Copyright Owner warrants that he/she/it has full authority to enter into this License and to make the grants this License contains.

1. The Author must please sign here (except if an Author is a U.S. Government employee, then please sign under #3 below):

S. ANITHA ELAVARASI 26.3.24

Author(s) Signature | Print Name | Date

2. The Copyright Owner (if different from the Author) must please sign here:

Name of Copyright Owner | Authorized Signature and Title | Date

3. If an Author is a U.S. Government employee, such Author must please sign below.
The signing Author certifies that the Work was written as part of his/her official duties and is therefore not eligible for copyright protection in the United States.

Name of U.S. Government Institution (e.g., Naval Research Laboratory, NIST)

Author Signature | Print Name | Date

PLEASE NOTE: NATIONAL LABORATORIES THAT ARE SPONSORED BY U.S. GOVERNMENT AGENCIES BUT ARE INDEPENDENTLY RUN ARE NOT CONSIDERED GOVERNMENT INSTITUTIONS. (For example, Argonne, Brookhaven, Lawrence Livermore, Sandia, and others.) Authors at these types of institutions should sign under #1 or #2 above.

If the Work was authored under a U.S. Government contract, and the U.S. Government wishes to retain for itself and others acting on its behalf, a paid-up, nonexclusive, irrevocable, worldwide license in the Work to reproduce, prepare derivative works from, distribute copies to the public, perform publicly, and display publicly, by or on behalf of the Government, please check the box below and add the relevant Contract numbers.

☐ Contract #(s) _____

ACKNOWLEDGMENT

First and foremost, I would like to express my gratefulness to our honorable Chairman **Sri. C Valliappa** our Vice Chairman **Sri. Chocka Valliappa** and **Sri Thyagu Valliappa** and the management of Sona College of Technology for bring me constant encouragement throughout this course.

My Sincere thanks goes to **Dr. S. R. R. Senthil Kumar**, Principal. Sona College of Technology, who has motivated me in my entire endeavors. I wholeheartedly thank Dr**. B. Sathiyabhama**, Professor and Head Department of Computer Science and Engineering, Sona College of Technology Salem for giving constant encouragement and rendering all kinds of supportthroughout the course.

I use this opportunity to express my deepest gratitude and special thanks to my project guide and my class counsellor**Dr. S. Anitha Elavarasi, Associate Professor** Department of Computer Science and Engineering, Sona College of Technology.

I would like to thank my parents and all my friends from the bottom of my heart who were always present to help me out and make this project a success. Last but not the least, I would like to express my heartiest thanks and gratefulness to almighty God for his divine blessings, which helped me complete the final year project successfully This project was made possible because ofinestimable inputs from everyone involved, directly or indirectly.

ABSTRACT

To address the communication challenges faced by people with and without hearing abilities who want to communicate with each other. This solution utilizes real-time Computer Vision and Convolutional Neural Networks (CNN) to detect and translate sign languages into regional spoken languages. The system is designed to accurately recognize hand gestures and establish links to speech and broadcast messages, facilitating effective communication between people with and without hearing abilities. This solution caters to two distinct user groups: proactive communicators and those requiring translation assistance. By bridging the gap between speech and sign, this solution meets the communication and translation needs of the user, thereby enabling their active participation in various interactions. The effectiveness of the solution will be evaluated through rigorous testing, focusing on accuracy, real-time performance, and user satisfaction metrics.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Networks
SRS	Software Requirement Specification
ASL	American Sign Language
ISL	Indian Sign Language
TTS	Text-to-Speech

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Communication is essential for everyone, and for individuals with hearing impairments, sign language serves as a crucial tool. It allows them to express themselves through hand gestures, facial expressions, and body movements. However, the sign language learners are mostly individuals with hearing impairment which makes it impossible for them to communicate with individuals with hearing abilities. In India alone, over 63 million people are estimated to have hearing-impairments, emphasizing the pressing need to address communication barriers within society. Limited understanding and proficiency in sign language among the general population further exacerbate these challenges.

To confront these obstacles, we propose a Sign Language Interpretation system that harnesses convolutional neural networks (CNNs). This system aims to accurately recognize hand gestures representing various elements of sign language, such as letters, words, phrases, and numbers in respective regional sign language. Beyond gesture recognition, our system is designed to generate complete sentences through finger spelling and sentence formation from gestures, enhancing comprehension for both hearing-impaired individuals and those without hearing impairments. This functionality facilitates seamless communication between these two groups, bridging the gap in understanding and promoting inclusivity.

The primary goal of our solution is to facilitate communication between individuals with and without hearing abilities. By promoting accessibility and inclusivity, our system strives to empower individuals with hearing impairments to fully participate in social interactions and everyday activities. This solution seeks to promote inclusivity and empower individuals with hearing impairments to fully participate in social interactions and everyday activities.

1.1 OBJECTIVE

The main objective of this research is to help the hearing and speaking impaired people to communicate with normal people by using this automatic sign language recognition system with text and speech generation. The aim of this research is to recognize the hand gesture in real time and predict the corresponding alphabets, words and sentences for regional sign language. This solution is for two distinct user groups: proactive communicators and those requiring translation assistance.

1.2 BACKGROUND OF THE STUDY

Sign language is a manual communication used primarily by people with hearing disabilities. Since most people are not familiar with sign language, it is getting harder and harder for hearing- impaired people to communicate without a translator, which makes them feel isolated. To overcome this issue, we proposed a system with the help of regional sign language. This proposed model tries to fill a communication gap between hearing- and hearing-impaired persons. This Sign Language Interpretation system helps to translate the sign language using hand gesture in real time and translate them into corresponding text and generate speech to bridge the communication gap between hearing impaired and the blind persons.

1.3 CHALLENGES

The Sign Language Interpretation System is a technology that aims to recognize and translate sign language into written or spoken language. However, developing an effective Sign Language poses several challenges. Here are some of them:

Variability Across Sign Languages: Sign languages differ widely across regions, each with its unique set of gestures and norms. For instance, American Sign Language (ASL),

British Sign Language (BSL), and Chinese Sign Language (CSL) each have distinct rules, making the creation of a universal sign language recognition system challenging.

Lack of Standardization: Sign languages do not have universally accepted standards for grammar and syntax, unlike spoken languages. This lack of standardization introduces significant hurdles in defining clear rules for grammar and syntax in sign language, which in turn complicates the development of a robust recognition system.

Complexity of Gestures: Sign language involves complex gestures that include nuanced facial expressions and body movements. The meaning of a sign can change dramatically based on the direction, position, and movement of the hands, as well as facial expressions. Accurately recognizing and interpreting these subtle differences poses a major challenge.

Scarcity of Data: Effective machine learning models require large datasets for training and testing. However, there is a notable scarcity of comprehensive data available specifically for Sign language, which limits the development of accurate and reliable recognition systems.

Limited Hardware Resources: The training and testing of deep learning models, which are integral to modern sign language recognition systems, require substantial computational power, typically provided by GPUs. However, access to such high-end hardware resources is limited in many parts of India, restricting the development and implementation of advanced recognition technologies in these regions.

Limited Data Availability: A lack of a standardized dictionary means there's a scarcity of labeled data for training machine learning models specifically tailored to Tamil sign language. This scarcity inhibits the ability to develop accurate recognition algorithms that can interpret Tamil sign language gestures effectively.

Variability in Gestures: Just as with spoken languages, sign languages can vary significantly among different regions and communities. The absence of a standardized dictionary for Tamil sign language means there may be inconsistencies and variations in the

gestures used by different signers, further complicating the training and interpretation process.

Addressing these challenges is crucial for the successful development and deployment of a Sign Language Interpretation system, which would significantly enhance communication and inclusivity for the deaf community.

1.4 APPLICATIONS

The real-time regional sign language recognition system has a wide range of applications and uses, some of which are: The system can help people with hearing impairments to communicate with others in real-time using regional sign language. The system can be used in schools and universities to help deaf students learn and communicate effectively with their peers and teachers. The system can improve accessibility for people with hearing impairments, enabling them to access public services, transportation, and other facilities.

The system can be used as an assistive technology for people with disabilities, helping them to interact with technology more effectively. Overall, the real-time regional sign language recognition system has the potential to improve the quality of life for people with hearing impairments, promote inclusivity and accessibility, and advance technological innovations.

CHAPTER 2

LITERATURE SURVEY

2.1 RESEARCH WORKS

With the advancement of technology, there has been a growing interest in exploring hand gesture recognition techniques to facilitate communication for the hearing and speaking impaired community. Specifically, the recognition of Sign Language has emerged as a significant research area, driven by the urgent need to enhance communication between individuals with and without hearing abilities.

Recognition of Sign Language has emerged as a critical research area, driven by the requirement to enhance communication between individuals with hearing impairments and the general population. Various methodologies have been explored to address Sign Language recognition, encompassing deep learning techniques, convolutional neural networks (CNNs), and hidden Markov models (HMMs).

Below are some studies made over the idea using various methodologies:

1. A review on hand gesture recognition system

Jayesh, Nilkanth, Ravindra, and Sunil [1] employed the Camshift algorithm in their research for enhanced tracking, focusing on features more effectively than the standard shift algorithm. Alongside this, they implemented the MRS-CRF algorithm, which stands out for its efficiency in recognizing dynamic hand gestures. This combination not only improves tracking accuracy but also surpasses the performance of the conventional CRF algorithm, resulting in significantly enhanced gesture recognition capabilities.

2. Sign Language Recognition using machine learning algorithm.

Prof. Radha, Mr. Vedant, Ms. Sanam, Ms. Pooja, and Ms. Mayuri [2] utilized Support Vector Machine (SVM) for classification in their study, aiming to identify the optimal

hyperplane that effectively separates different sign feature spaces. Additionally, they incorporated Hidden Markov Models (HMM) to analyze temporal dynamics in sequences of gestures. This approach allowed for accurate predictions in the recognition of sign language gestures. However, despite the precision in simpler gestures, their system encountered performance challenges, specifically lagging, when dealing with more complex gestures. This lag suggests that while the model is proficient in handling standard gestures, it struggles with the intricacies of more elaborate and nuanced movements.

3. A real-time American sign language recognition system using convolutional neural network for real datasets

Raksha Amer and Mentandher [3] utilized Convolutional Neural Networks (CNNs) in their research to enhance the extraction of spatial features from video frames, which is crucial for sign language recognition. CNNs are particularly adept at analyzing images and identifying relevant patterns necessary for distinguishing different signs. This capability makes CNNs an excellent tool for interpreting the complex visual information presented in sign language videos, thereby improving the accuracy and efficiency of sign language recognition systems in processing and understanding dynamic hand gestures.

4. Video based sign language recognition using CNN-LSTM

Sai Bharat, Gogineni, Mhav, and Saranu [4] employed a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks in their project. The CNN algorithm was used to effectively extract spatial features from images, while LSTM networks were integrated to capture the temporal dependencies that exist in sequences of hand gestures. This dual-approach allowed for a more comprehensive analysis of sign language, recognizing not only individual gestures but also the flow and context of gestures over time, enhancing the overall accuracy and fluency of sign language interpretation.

5. CoSign: Exploring Co-occurrence Signals in Skeleton-based Continuous Sign Language Recognition

Jiao, Min, Wang, Lei, and Chen [5] implemented the Convolutional Neural Network (CNN) algorithm to extract significant features from input data in their research on sign language recognition. To enhance this process, they incorporated a contextual module, specifically a Recurrent Neural Network (RNN). This addition allowed their system to not only analyze individual images for spatial features but also to understand and process the context and sequence of movements over time. The integration of RNN with CNN provided a more robust feature extraction mechanism that was context-aware, significantly improving the accuracy and efficiency of recognizing complex and dynamic sign language gestures. This innovative approach represents a substantial advancement in the field of sign language interpretation technology.

6.Artificial neural network training using a new efficient optimization algorithm

Ayush, Navya, Pinal, Simran, and Supriya [6] implemented a comprehensive approach using Hidden Markov Models (HMM) for the preprocessing, training, and recognition phases of their project. To enhance these capabilities, they integrated an Artificial Neural Network (ANN) optimized using cuckoo search, a nature-inspired algorithm. This optimization aimed to refine communication and recognition processes while effectively removing noise from the data. This dual-technique approach not only improved the efficiency of the system but also significantly enhanced the accuracy of sign language interpretation by optimizing performance parameters.

7. Design of Sign Language Recognition Using E-CNN

Saurdi, Citra, Et al [7] employed the CNN algorithm, resulting in average accuracy. They also used filtering image techniques for hand key point library to find the area of the hand to the tip of the finger, along with deep learning CNN-DNN (Deep Neural Network) to solve visual recognition problems.

8. An efficient Indian sign language recognition system using sift descriptor

Kaur and Krishna c [8] utilized the SIFT algorithm for FFBPNN (Feedforward Backpropagation Neural Network) extraction, ABC optimization, and also employed ANN & SIFT for feature extraction and multiclas s classification.

9. Video-based isolated hand sign language recognition using a deep cascaded model

Aoxiong, Zhou, Weike, Meng, Xingshan, Xiaofe [9] employed ISLR (Isolated Sign Language Recognition) to recognize isolated signs. They also utilized the CNN algorithm to achieve good results and CSLR (Continuous Sign Language Recognition) to recognize sign gloss sequences from continuous videos.

10. Using Deep Convolutional Networks for Gesture Recognition in American Sign Language

Vivek, Dianna [10] utilized the CNN algorithm for more accuracy in alphabet and digits. Additionally, they employed basic supervised learning using mini-batch stochastic gradient descent for the accuracy of alphabets and digits on self-generated datasets.

11. Hand Gesture Detection and Conversion to Speech and Text

K.Manikandan, Ayush, Pallav, Aneek [11] utilized contour analysis and feature extraction for capturing images, grasale, and convex hull generation of points on hand postures with ASL dataset using the OpenCV tool.

12. Neural Sign Actors: A diffusion model for 3D sign language production from text

Jing, Kang, Wenjie, Xuefui, Ying, Zhengu, Linchao [12] utilized 2D, 3D CNN, LSTM, VAE (Variational Autoencoder) model. While LSTM had the lowest diversity score and got stuck with some poses, CNN static 2D CNN, dynamic 3D CNN, and VAE encodes convert audio to sign language.

13. Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model

Hauda, Juan, Kalika [13] utilized OpenPose and MediaPipe for 2D and 3D normalization, joints tree structure, and frame generation. They used datasets like WLASL, JSL, ASL, ISLR, and were effective for joint normalization and frame generation.

14. 3D – CNN based dynamic gesture recognition for Indian sign language Modeling

Dushyant Kumar [14] employed the 3D CNN algorithm for detecting real-time motion signs and utilized datasets of ISL (Indian Sign Language).

15. Indian Sign Language recognition system using SURF with SVM and CNN

Katoch, Tiwary [15] utilized SVM, CNN, Pyttsx3, Google Speech API, and data source to get segmented feature extraction, trained, tested, and reverse recognition was done to decide the best algorithm. They also used techniques like Hand sign recognition, ISL dataset, and Bag of Visual Words (BOVW).

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE SPECIFICATIONS

The hardware requirements for a Sign Language Interpretation System depend on the specific approach and technology used for the recognition task. Here are some general hardware requirements to consider:

Processor: CPU (minimum i3 processor) or GPU (for speeding up the process).

Memory: RAM (4GB to 8GB)

Camera: A camera with at least 720p resolution is recommended.

Storage: At least 100 GB of storage space is recommended.

3.2 SOFTWARE SPECIFICATIONS

To develop the Sign Language Interpretation system using machine learning, the following software and packages are required.

Python: Version 3.10.10 or higher

Visual Studio: Version 1.77.3 or higher

Packages Used: TensorFlow, Keras, Tkinter (for GUI window), OpenCV (for image capturing), Pyttsx3 (for Speech generation), Flask

3.3 SOFTWARE REQUIREMENT SPECIFICATION

An SRS is a description of a software system to be developed. The software requirements specification document lists sufficient and necessary requirements for software development.

Python: Python is consistent and is anchored on simplicity, which makes it most appropriate for machine learning. The Python programming language best fits machine learning due to its independent platform and its popularity in the programming community.

TensorFlow: TensorFlow is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities: Efficiently executing low-level tensor operations on CPU, GPU, or TPU. Computing thegradient of arbitrary differentiable expressions. Scaling computation to many devices, such as clusters of hundreds of GPUs. Exporting programs ("graphs") to external runtimes such as servers,browsers, mobile and embedded devices.

Keras: Keras is the high-level API of the TensorFlow platform: an approachable, highly- productive interface for solving machine learning problems, with a focus on modern deep learning neural networks. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinterprovides a fast and easy way to create GUI applications. Tkinter provides a powerful object- oriented interface to the Tk GUI toolkit.

Pyttsx3: pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.

Flask: Flask is a lightweight Python web framework that's ideal for developing RESTful APIs and web applications. Its simplicity and flexibility make it a perfect choice for integrating with machine learning models, facilitating the deployment of AI-powered services with ease.

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

- There exists numerous sign language recognition system for American Sign Language, Chinese Sign Language, Australian Sign Language, French Sign Language, etc., but only very few systems available for Indian Sign Language. These systems use various techniques such as computer vision, machine learning, and sensor-based technologies to recognize and interpret hand gestures.
- Most of the existing sign language recognition systems can recognize only a few alphabetical characters or digits. Only a few of them can recognize all the 26 alphabets of the English language. This limitation is due to the complexity involved in recognizing and interpreting the various hand gestures and movements involved in forming words and sentences in sign language.
- The existing sign language recognition systems do not have the feature of speech output for Indian Sign Language, which means they can only recognize and interpret sign language gestures and convert them to text.
- In the case of Sign Language Recognition (SLR) systems, there exists only a one-way communication system, which means the existing systems can only recognize and interpret sign language gestures and convert them to text. There is no system available yet that can convert text into sign language gestures.
- There are two main types of sign language recognition systems: vision-based and sensor-based. Vision-based systems use cameras and computer vision techniques to recognize and interpret gestures. Sensor-based systems, on the other hand, use sensors attached to the user's hand and wrist to capture the movements of the hand and interpret the sign language gestures. Both these models can be used for sign language recognition systems. However, the choice of the model depends on the application and accuracy, requirements, and cost.

4.2 PROPOSED SYSTEM

- The proposed solution is specifically designed to recognize complex hand gestures prevalent in regional sign languages used by the hearing-impaired community in India, accurately detecting nuanced gestures integral to conveying detailed information.

- Unlike existing systems that may only identify isolated signs or simple phrases, this system is engineered to interpret complete sentences and phrases, enhancing the depth and fluidity of conversations between users with and without hearing impairments.

- An integral feature of the proposed system is its speech generation capability. This functionality allows the system to convert recognized sign language gestures into audible speech and text, bridging communication gaps and enabling more effective interaction between the hearing and hearing-impaired communities.

- The system is comprehensive, recognizing all Tamil letters, as well as some common phrases, making it a versatile tool for communication that surpasses many current sign language recognition systems in scope and utility.

- Rigorous testing of the system will focus on key performance indicators such as gesture recognition accuracy, real-time processing speed, and overall user satisfaction. This ensures the solution not only meets but exceeds the practical requirements of its users.

- By enabling active participation in various interactions through a robust translation framework, this innovative solution aims to empower individuals with hearing impairments and facilitate their full involvement in social and professional environments. The ultimate goal is to provide a versatile, reliable, and user-friendly platform that significantly reduces communication barriers and promotes inclusivity.

4.3 NOVELTY IN PROPOSED WORK

- It can recognize all the regional characters and some words in regional sign language.
- It also includes generation of words and sentences.
- It also includes speech output for generated texts.

4.4 ALGORITHM

4.4.1 CONVOLUTIONAL NEURAL NETWORK (CNN)

The algorithm used for classifying the hand gesture is Convolutional Neural network (CNN).Convolutional Neural Network (CNN) is a type of neural network that is widely used in image recognition and classification tasks. It is inspired by the structure and function of the visual cortex in the human brain, which processes visual information in a hierarchical manner. CNNs consist of multiple layers, each performing a specific function in the image recognition process. The first layer is usually a convolutional layer, which extracts features such as edges, corners, and shapes from the input image. The second layer is typically a pooling layer, which reduces the spatial size of the feature maps while retaining the most important features. The third layer is usually a fully connected layer, which learns to classify the input image into different classes.

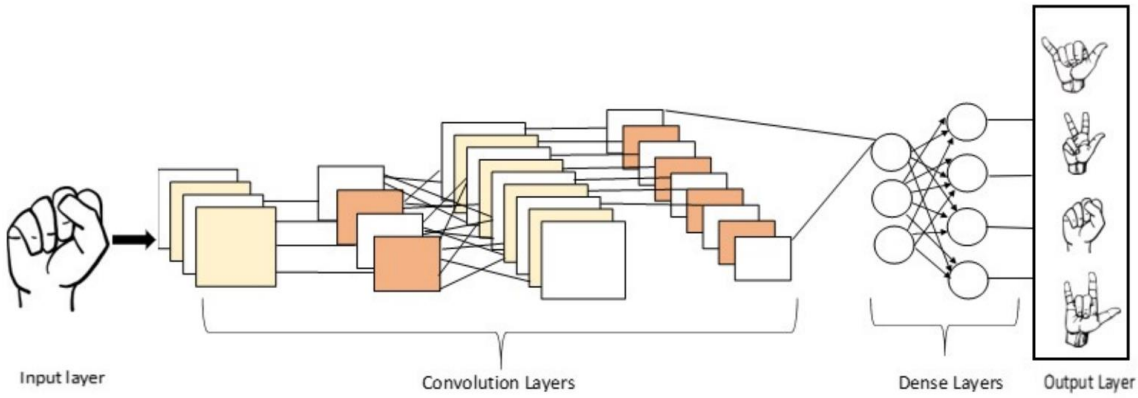


Figure 4.1. Proposed CNN Architecture (norma.ncirl.ie 2021)

The benefits of CNNs include their ability to learn features directly from raw input data, their scalability to handle large datasets and complex tasks, and their high accuracy in image recognition and classification tasks. CNNs are also able to handle variations in the

input data, such as changesin lighting, scale, and orientation, which makes them well-suited for real-world applications.

CNNs are used in a wide range of applications, including object recognition, facial recognition, medical image analysis, and autonomous driving. In object recognition, CNNs can beused to identify and classify different objects in an image, such as cars, animals, and buildings. Infacial recognition, CNNs can be used to identify individuals based on their facial features, which has applications in security and law enforcement. In medical image analysis, CNNs can be used to detect and diagnose different medical conditions, such as tumors and abnormalities. In autonomous driving, CNNs can be used to recognize and classify different objects in the environment, such as other vehicles, pedestrians, and traffic signs.

Overall, CNNs are a powerful tool for image recognition and classification tasks, with a widerange of potential applications in various fields. Their ability to learn directly from raw input data and handle variations in the input make them well-suited for real-world applications where input data can be complex and diverse.

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1 METHODOLOGIES

The proposed methodology for the Sign Language interpretation System aims to develop a robust framework capable of recognizing hand gestures using CNNs, generating sentences through finger spelling, and converting speech into sign language gestures. A significant communication gap exists between ordinary individuals and those with hearing impairments due to a lack of understanding of sign language within the broader community. To address this gap, an effective sign language interpretation system is crucial. By recognizing hand gestures and converting them into speech, this system aims to bridge the communication barrier. Additionally, the proposed method integrates the individual recognition of hand gestures to construct complete sentences, while also incorporating features for speech-to-sign conversion.

The first stage of the system involves data collection, where a diverse dataset of hand gesture images representing various sign languages from around the world is compiled. This dataset encompasses gestures for alphabets, numbers, and common phrases. Following data collection, the dataset undergoes preprocessing. This involves converting the raw images to grayscale and applying a Gaussian blur filter to enhance feature extraction capabilities. These pre-processed images serve as input for the subsequent stages of the system. The training phase utilizes a CNN-based model trained on the pre-processed dataset to categorize images based on hand gesture features.

The model architecture consists of multiple convolutional layers for feature extraction, pooling layers with activation functions for dimensionality reduction, and a fully connected layer for gesture classification. Gesture recognition is the next stage, where the system utilizes OpenCV to capture live camera feed input. This input is pre-processed and fed into the trained model, which predicts the corresponding hand gesture in real-time. Once a hand

gesture is recognized, the system proceeds to sentence formation based on the recognized individual gestures. This involves combining recognized gestures into intelligible sentences, representing the intended message. Subsequently, speech generation is initiated, where the formed sentence is converted into spoken language for better understanding by individuals with hearing impairments or those unfamiliar with sign language. Additionally, the proposed system incorporates a module for speech-to-sign conversion. This module allows users to input speech, which is then converted into equivalent hand gestures, providing an additional means of communication for users familiar with sign language.

5.1.1 DATA COLLECTION

The first step in creating a Sign Language interpretation system involves data collection, an important step given the lack of current language-level global sign language recognition data. To address this gap, we have collected data from various sources to provide services. For word and phrase recognition analysis, a comprehensive global dataset such as sign language learning platform comprising 40+ commonly used words and phrases was utilized. Each word is represented by several scenes shot from different angles and lighting to ensure the strength and breadth of the model. These promote better communication and participation for individuals who use sign language through data collection and design.

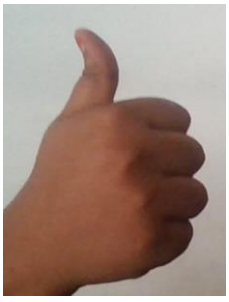
Therefore, the total collection has 71 categories such as Tamil uyir, mei, ayutham and commonly used words. Each category comprises of 1200 hand gesture images along with the labels corresponding to them. Figure 1 shows sample dataset for theproposed system.

Figure 5.1. Sample Dataset

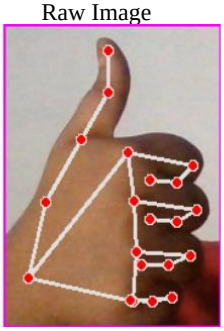
5.1.2 DATA PRE-PROCESSING

Input images are utilized to extract detailed features of hand gestures, a critical step in accurately interpreting sign language. To achieve this, the system employs MediaPipe Hand detection technology, renowned for its ability to precisely identify the positions of finger joints. This technology maps the hand's structure in the image by detecting key landmarks such as fingertips and joints, enabling the system to accurately analyze the nuances of each gesture. In addition to finger point detection, the system incorporates thresholding techniques to enhance gesture recognition.

Thresholding is a process that helps to clearly distinguish the hand region from the background in the image. By setting a specific intensity threshold, pixels that fall within the range are considered part of the hand, while others are treated as background. This segmentation is crucial as it reduces the complexity of the image, allowing the system to focus solely on the hand gestures without any interference from the surrounding environment. Together, these techniques ensure the system captures and processes hand gestures with high precision, facilitating effective translation of sign language into spoken language.



Raw Image



After applying finger point detection

Figure 5.2. Hand Gesture Pre-processing

5.1.3 CLASSIFICATION

After identifying the Region of Interest (ROI) that is hand, deep learning techniques are utilized to train the image dataset for classification purposes. This step is crucial in the Sign Language interpretation system, as it entails recognizing the user's specific gesture and associating it with the corresponding sign language word or phrase. Convolutional Neural Networks (CNNs) are favored for gesture recognition due to their ability to learn features from hand gesture images and attain impressive accuracy levels.

CONVOLUTIONAL NEURAL NETWORKS (CNN)

A Convolutional Neural Network (CNN) serves as a pivotal tool in recognizing and classifying hand gestures for the proposed Sign Language Interpretation System. CNNs are expert at automatically learning relevant features from input image data, eliminating the need for manual feature extraction. The core concept behind CNNs [Fig 5] lies in the use of convolutional filters, which slide over the input image, computing dot products with pixel values in local regions to extract features. These features are then processed through subsequent layers for further analysis and classification.

Each layer of CNN architecture serves a distinct function in the feature extraction and classification process:

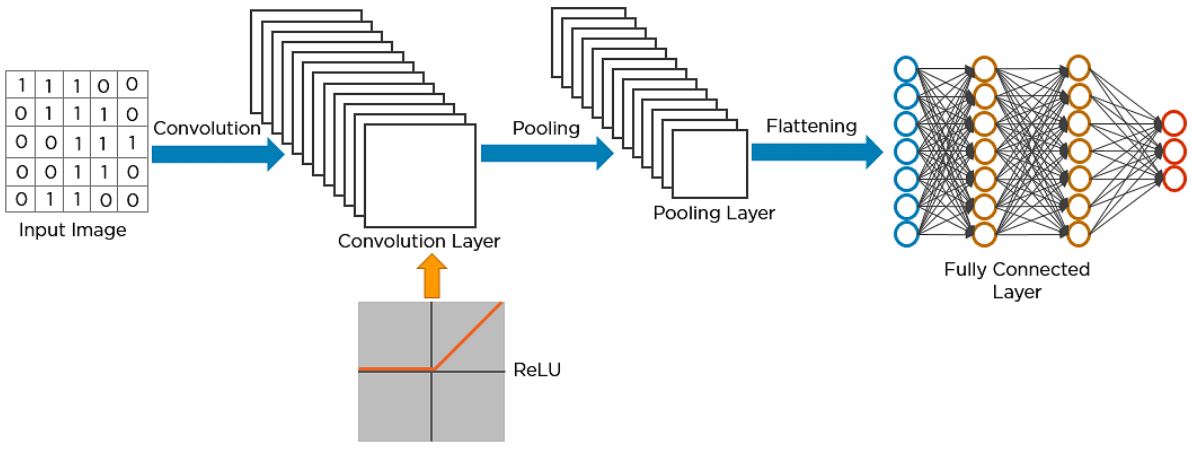


Figure 5.3. CNN Layers (simplilearn tutorial, 2023)

- Input Layer: Receives the input image for self-generated dataset and frames from

global dataset.

- Convolutional Layers: Extract significant features such as edges, shapes, and lines from the input images and frames.

- ReLU Activation Function: Introduces non-linearity to enhance model accuracy.

The CNN model comprises three convolutional layers with 64 filters each, followed by max pooling layers and ReLU activation functions. The third max pooling layer's output is flattened and fed into a dense layer with 128 units, followed by a fully connected layer with SoftMax activation. The final output layer has 31 + 40 = 71 units representing each class. The ReLU activation function is defined by equation (2).

$$r(t) = \max(0, t) \qquad (2)$$

- Pooling Layers: Down sample the output, reducing dimensionality while preserving crucial features.

- Dropout Layer: Prevents overfitting and promotes generalization by randomly removing neurons during training.

- Fully Connected Layers: Apply linear transformations to learned features, followed by SoftMax activation functions for classification.

In the proposed SoftMax activation function, denoted as r(t), the input is represented by t. This function yields a piecewise linear behavior, remaining linear for positive input values and outputting 0 for negative input values. Specifically, it returns t for positive inputs and 0 for negative inputs. The formula for the SoftMax activation function is depicted as equation (3).

$$Softmax(v_x) = {e^{v_x}}/{\sum_y e^{v_y}} \qquad (3)$$

- Output Layer: Produces the final prediction for the recognized hand gesture.

The proposed CNN architecture comprises multiple convolutional layers with ReLU

activation functions, followed by max pooling layers and a flatten layer to simplify the model and create a one-dimensional vector for classification. The final fully connected layer utilizes an SoftMax activation function to generate a probability distribution over the classes, enabling predictions.

Overall, this CNN architecture is well-suited for the proposed system of Indian sign language recognition using hand gestures. The multiple convolutional layers with ReLU activation functions allow the network to learn complex features, while the max pooling layers and flatten layer lessen the dimensionality of the feature maps and produce a one-dimensional vector for classification. Ultimately, the SoftMax activation function in the fully connected layer creates a probability distribution over the various classes, enabling the network to produce accurate predictions.

5.1.4 GESTURE RECOGNITION

This method entails applying a color filter to the input image and isolating pixels within a specified color range. To distinguish the hand region from the background, a threshold is applied to the resulting image.

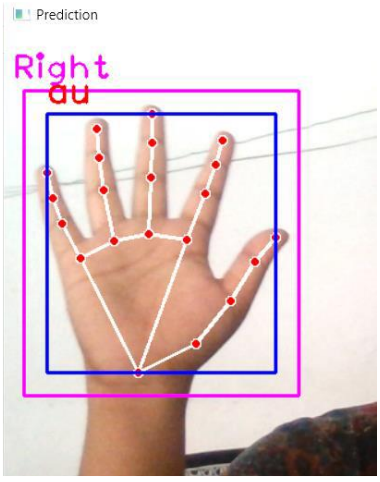


Figure 5.4. Gesture Recognition

The CNN model was trained for 10 epochs to strike a balance between learning and generalization. An average training value of 0.00019 was used to prevent overshooting and guide the model towards the correct solution. With 36 categories for Indian languages, the SoftMax function was applied to the output to calculate predicted probabilities for each class. The binary

cross-entropy loss function (Equation 4) was employed to measure the disparity between predicted and actual classes for each sample in the dataset. This loss function is advantageous due to its well-defined nature and suitability for optimization using gradient descent-based algorithms. The Adam optimizer was utilized in this training process. The binary cross-entropy model is shown in Equation (4).

$$BCE = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij}) \qquad (4)$$

5.1.5 SENTENCE FORMATION

Fingering recognition plays a vital role in the proposed Sign Language Interpretation System, enabling the identification of words and phrases not represented by specific hand gestures. Fingerspelling involves spelling out words letter-by-letter through hand gestures and serves as a common mode of communication for individuals with hearing impairments.

In this system, a letter is appended to the existing text when its occurrence frequency surpasses a predetermined threshold, and no other letters exhibit comparable frequencies. The system offers alternative character options if similar letters are detected, ensuring accuracy in text formation. Additionally, the system detects blank screens without hand signals, representing spaces between words in the constructed phrases. Ultimately, the phrase is assembled through sequential recognition of alphabetic hand gestures, facilitating effective communication for users.

5.1.6 SPEECH GENERATION

In the envisioned system, the process of converting text to speech plays a crucial role, allowing the system to articulate spoken language corresponding to the recognized sign language gestures.

Text-to-speech systems operate by analyzing textual input and breaking it down into individual word elements, representing sound units. A speech synthesis engine is then

employed to generate speech waveforms for each word element, and these are combined to form the final spoken output. The system utilizes the pyttsx3 Python library for text-to-speech conversion, providing flexibility in language, accent, and customizable parameters such as speed, volume, and pitch. This library supports event callbacks, enhancing integration with other Python applications.

5.1.7 EXPERIMENTS AND RESULTS

The dataset utilized in the proposed Sign Language Interpretation System comprises images of hand movements corresponding to Tamil uyir, mei, ayutham letters, and 40 English words. This rich dataset is designed to train a CNN-based model for effectively recognizing and interpreting these sign language gestures.

The dataset encompasses a total of 71 classes, including the Tamil alphabets and the English words, along with a blank image used to represent spaces. High-resolution cameras capture the hand gestures at an increased rate of 60 frames per second, ensuring detailed and dynamic representations of each gesture, which is crucial for accurate recognition. The images and videos are captured in RGB format, providing color information that enhances the model's ability to efficiently process and recognize varied gestures.

In terms of data pre-processing, the images undergo Grayscale conversion, which strips away the color data in favor of enhancing contrast and clarifying the gesture edges. This conversion is beneficial for the model as it simplifies the task of distinguishing between different gestures. Additionally, Gaussian blur is applied to the images to smooth out any noise, further aiding the model by highlighting essential features and suppressing irrelevant variations in the image data.

These pre-processing steps are critical as they not only help in reducing noise but also improve the training speed of the model by focusing on the most significant features. This rigorous preprocessing minimizes overfitting by stripping unnecessary details from the images, thereby enhancing the model's ability to generalize from the training data to new, unseen data effectively. By adopting these methods, the system achieved an impressive 94% accuracy in recognizing and interpreting the designated sign languages, showcasing its

capability to function effectively in real-world scenarios.

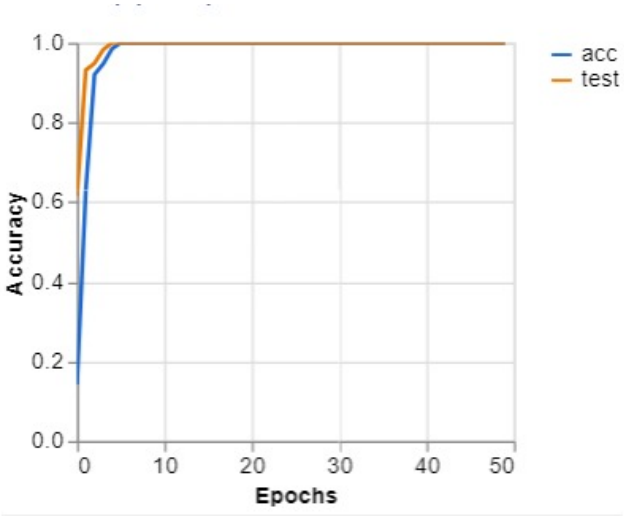


Figure 5.5. Accuracy graph for proposed CNN

This proposed technique makes use of a CNN architecture with three convolutional layers with 71, 64 filters, respectively. Each convolutional layer is followed by max pooling layer that have ReLU activation functions. The output of the third max pooling layer is flattened and fed into a dense layer with 128 units, and then fully connected layer that has a SoftMax activation function. The final output layer has 37 units, one for each class. The main benefit of using categorical Cross- Entropy is because it is designed to handle multi-class classification tasks, where each sample can belong to one of several classes. Here Adam optimizer is used in this training process.

Adam (Adaptive Moment Estimation) is an optimization algorithm that updates model parameters by taking a moving average of the gradient and the squared gradient. It also employs bias correction to correct for the bias caused by the moving averages' initial estimates. Adam is used for the proposed model because it is an efficient optimization algorithm for training the deeplearning model and it can handle sparse gradients.

After training the proposed model for 10 epochs, the proposed model achieved an accuracy of 94% on the validation set with minimum loss.

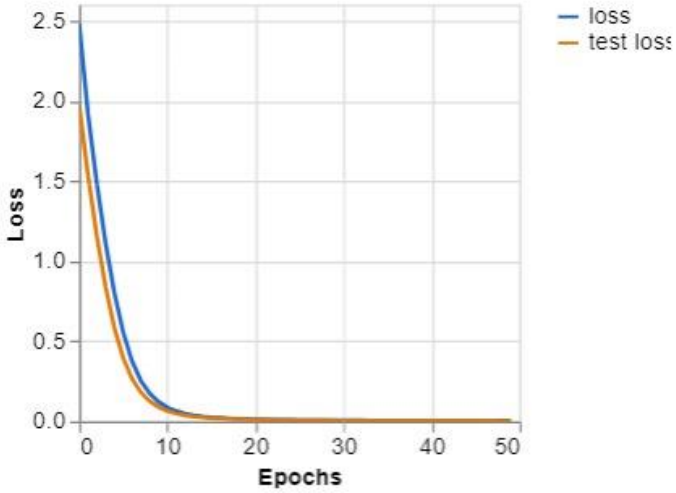


Figure 5.6. Loss graph for proposed CNN

After evaluating various techniques as presented in Table 1, we are expecting that the proposed CNN-based model will be expected to achieve a 94% accuracy rate. Additionally, the system facilitates sentence formation from gesture recognition and integrates text-to-speech (TTS) conversion to produce spoken output from the formed sentences. Utilizing a Python TTS library, the system generates speech from text, enabling the conversion of recognized gestures into speech.

The use of hand gestures in Sign Language Recognition employs a variety of algorithms totrain the dataset. Here are some comparisons of popular algorithms.

Table 1. Accuracy comparisons of various method with the proposed method

Method	Accuracy
MRS CRF algorithm [1]	90.45%
TensorFlow Object Detection API [16]	85.45%
Convolutional Neural Networks, LSTMs, Microsoft Kinect [18]	78.3%
Artificial Neural Network (Backpropagation Algorithm) [19]	85.7%
Conditional Random Fields, Support Vector Machine [20]	92.5%
ResNet50	93%
VGG16	92%
Proposed CNN Model	94%

5.2 SYSTEM ARCHITECTURE DIAGRAM

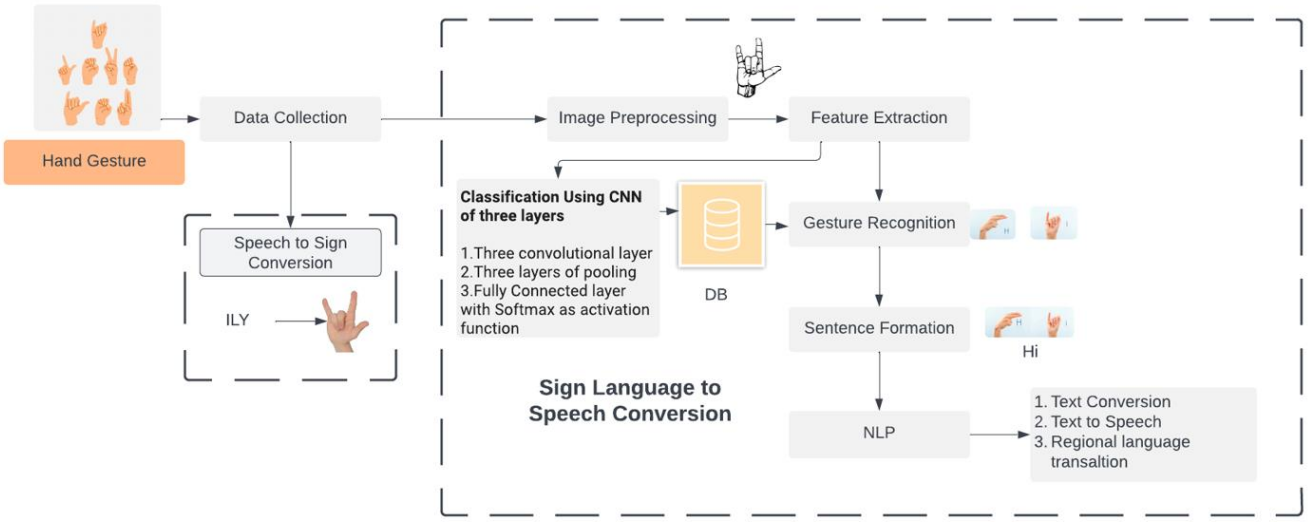


Figure 5.7. System Architecture

The proposed system aims to facilitate communication between individuals with and without hearing and speaking abilities, enabling seamless interaction in both spoken and sign languages. It encompasses various components to achieve this goal. The system begins with image preprocessing, ensuring clarity and removing noise from captured sign language gestures.

Utilizing a 3-layer Convolutional Neural Network (CNN), the system leverages powerful deep learning techniques to accurately classify sign language gestures. This classification step forms the foundation for the system's ability to interpret and understand different signs. Following classification, the system proceeds to feature extraction, where key characteristics of each sign are identified. This step is essential for precise interpretation and recognition of signs.

Gesture recognition is the core functionality of the system, where the extracted features are matched to a vast sign language dictionary. This dictionary allows the system to interpret gestures and form them into coherent sentences. Incorporating Natural Language Processing (NLP), the system further enhances its capabilities. It includes text conversion, transforming interpreted sentences into written format, and text-to-speech functionality, generating

natural-sounding audio from the processed text. By integrating these components seamlessly, the proposed system aims to bridge communication gaps and promote inclusivity by facilitating effective communication between individuals with and without hearing and speaking abilities, in both spoken and sign languages.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This system for Sign Language Interpretation proves the effectiveness of the proposed model for recognizing hand gestures and generating spoken output from the recognized gestures. The proposed Sign Language Interpretation System using hand gestures achieves the accuracy of 94%. However, there is still a need for further research to improve the accuracy and robustness of these systems, as well as to develop more advanced speech generation systems for sign language recognition. While current Sign language recognition systems focus primarily on hand gesture recognition and conversion of text to sign language, there is potential to integrate other modalities, such as real time handwritten text to sign language conversion to improve the proposed System. Future research could explore the use of multi-modal recognition systems that combine hand gesture recognition with other modalities.

APPENDICES

SAMPLE CODE

datacollection.py:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

# This code collect data using mediapipe to focus on fingerpoints
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
counter = 0

folder = 'Data/Stop'

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h= hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
        imgCrop = img[y-offset: y+h+offset, x-offset: x+w+offset]
        imgCropSize = imgCrop.shape

        aspectRatio = h/w

        if aspectRatio > 1:
            k = imgSize/h
            wCal = math.ceil(k*w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize-wCal)/2)
            imgWhite[:, wGap: wCal + wGap] = imgResize
        else:
            k = imgSize/w
            hCal = math.ceil(k*h)
```

```
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize-hCal)/2)
            imgWhite[hGap: hCal + hGap, :] = imgResize
        cv2.imshow('ImageCrop', imgCrop)
        cv2.imshow('ImageWhite', imgWhite)
    cv2.imshow('Image', img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
        print(counter)
    if key == ord('q'):
        cap.release()  # Release the camera
        cv2.destroyAllWindows()  # Destroy all OpenCV windows
        break
```

Implementation.ipynb:

```
import os
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split

# Define the directories containing your datasets
mei_dir = 'Data/Mei/'
uyir_dir = 'Data/Uyir/'

# List of labels
mei_labels = ['க்', 'ங்','ச்','ஞ்','ட்','ண்','த்','ந்','ப்','ம்','ய்','ர்','ல்','வ்','ழ்','ள்','ற்','ன்']
uyir_labels = ['அ', 'ஆ', 'இ', 'ஈ', 'உ', 'ஊ', 'எ', 'ஏ', 'ஐ', 'ஒ', 'ஓ', 'ஔ', 'ஃ']

# Image size
imgSize = 300  # Define the desired size for your images

# Initialize lists to store images and labels
X = []
y = []

# Load images and corresponding labels from Mei directory
for label_idx, label in enumerate(mei_labels):
    label_dir = os.path.join(mei_dir, label)
```

```
    for img_name in os.listdir(label_dir):
        img_path = os.path.join(label_dir, img_name)
        img = Image.open(img_path)
        img = img.resize((imgSize, imgSize))  # Resize the image if needed
        img = np.array(img) / 255.0  # Normalize pixel values
        X.append(img)
        y.append(label_idx)  # Assign the index of the label to the image

# Load images and corresponding labels from Uyir directory
for label_idx, label in enumerate(uyir_labels):
    label_dir = os.path.join(uyir_dir, label)
    for img_name in os.listdir(label_dir):
        img_path = os.path.join(label_dir, img_name)
        img = Image.open(img_path)
        img = img.resize((imgSize, imgSize))  # Resize the image if needed
        img = np.array(img) / 255.0  # Normalize pixel values
        X.append(img)
        y.append(len(mei_labels) + label_idx)  # Assign the index of the label to the image,
adding the offset

# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
labels = mei_labels + uyir_labels
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(imgSize, imgSize, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(labels), activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.summary()
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
model.save('Model/uyirmei_model.h5')
i = 0
with open('labels.txt', 'w') as f:
    for label in labels:
        f.write(str(i)+" "+label + '\n')
        i+=1
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.savefig('accuracy_graph.png')
plt.show()
labels = ['a', 'aa', 'i', 'ii', 'u', 'uu', 'e', 'ee', 'ai', 'o', 'oo', 'au', 'ak', 'ik', 'ing', 'ich', 'inj', 'it',
'in', 'ith', 'indh', 'ip', 'im', 'iy', 'ir', 'il', 'iv', 'izh', 'ill', 'irr', 'inn']
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
cm = confusion_matrix(y_test, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,
yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.savefig('Findings/um_confusion_matrix.png')
plt.show()
```

test.py:

```
import cv2
import numpy as np
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
from PIL import Image, ImageTk
import math
import tkinter as tk
```

```
from tkinter import Label, Button, Frame

class Application:
    def __init__(self, root):
        self.root = root
        self.root.title("Sign Language Interpreter")
        self.root.geometry("1260x700")
        self.root.config(background="#001F3F")  # Dark navy blue background
        self.root.pack_propagate(False)  # Prevent window from expanding
        self.create_widgets()

        # Initialize hand detector and classifier
        self.detector = HandDetector(maxHands=1)
        self.classifier = Classifier("Model/uyir_t_model.h5", "Model/u_t_labels.txt")

        # Initialize video capture
        self.cap = cv2.VideoCapture(0)
        self.offset = 20
        self.imgSize = 300
        self.labels = ['அ', 'ஆ', 'இ', 'ஈ', 'உ', 'ஊ', 'எ', 'ஏ', 'ஐ', 'ஒ', 'ஓ', 'ஔ', 'ஃ']
        self.predicted_word = ""

        # Start video loop
        self.video_loop()

    def create_widgets(self):
        # Create a frame to hold the title label
        title_frame = Frame(self.root, bg="#001F3F")
        title_frame.pack(fill=tk.X, padx=20, pady=(20, 0))

        # Title label
        self.title_label = Label(title_frame, text="Sign Language Interpreter", font=("Comic
Sans MS", 24, "bold"), bg="#001F3F", fg="white")
        self.title_label.pack(pady=20)

        # Create a frame to hold the panels
        panels_frame = Frame(self.root, bg="#001F3F")
        panels_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)

        # Left panel
        self.panel = Label(panels_frame)
```

```
        self.panel.pack(side=tk.LEFT, padx=20)

        # Right panel and buttons
        panel2_buttons_frame = Frame(panels_frame, bg="#001F3F")
        panel2_buttons_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        # Right panel
        self.panel2 = Label(panel2_buttons_frame)
        self.panel2.pack(side=tk.LEFT, padx=20)

        # Buttons frame
        buttons_frame = Frame(panel2_buttons_frame, bg="#001F3F")
        buttons_frame.pack(side=tk.LEFT, padx=20, pady=20)

        # Stop button
        stop_button = Button(buttons_frame, text="Stop", font=("Comic Sans MS", 10, "bold"),
bg="#E48F1B", fg="#34262B", command=self.stop)
        stop_button.grid(row=0, column=0, padx=10)

        # Restart button
        '"restart_button = Button(buttons_frame, text="Restart", font=("Comic Sans MS", 10,
"bold"), bg="#E48F1B", fg="#34262B", command=self.restart)
        restart_button.grid(row=0, column=1, padx=10)"'

        # Tamil letter button
        tamil_button = Button(buttons_frame, text="Tamil Letter", font=("Comic Sans MS",
10, "bold"), bg="#E48F1B", fg="#34262B", command=self.switch_to_tamil)
        tamil_button.grid(row=1, column=0, padx=10, pady=10)

        # Words button
        words_button = Button(buttons_frame, text="Words", font=("Comic Sans MS", 10,
"bold"), bg="#E48F1B", fg="#34262B", command=self.switch_to_words)
        words_button.grid(row=1, column=1, padx=10, pady=10)

        # Widgets below the panels
        self.bottom_frame = Frame(self.root, bg="#001F3F")
        self.bottom_frame.pack(fill=tk.X, padx=20, pady=20)

        self.T2 = Label(self.bottom_frame, text="Word :", font=("Comic Sans MS", 16,
"bold"), bg="#001F3F", fg="white")
        self.T2.grid(row=0, column=0, pady=(20, 0), padx=(20, 0), sticky="w")
```

```
        self.word_label = Label(self.bottom_frame, text="", font=("Comic Sans MS", 16),
bg="#001F3F", fg="white")
        self.word_label.grid(row=0, column=1, pady=(20, 0), padx=(10, 0), sticky="w")

        speak_button = Button(self.bottom_frame, text="Speak", font=("Comic Sans MS", 10,
"bold"), bg="#E48F1B", fg="#34262B", command=self.speak_word)
        speak_button.grid(row=0, column=2, pady=(20, 0), padx=(50, 20), sticky="w")

        clear_button = Button(self.bottom_frame, text="Clear", font=("Comic Sans MS", 10,
"bold"), bg="#E48F1B", fg="#34262B", command=self.clear_word)
        clear_button.grid(row=0, column=3, pady=(20, 0), padx=(0, 20), sticky="w")

    def video_loop(self):
        success, img = self.cap.read()
        imgOutput = img.copy()
        hands, img = self.detector.findHands(img)
        if hands:
            hand = hands[0]
            x, y, w, h = hand['bbox']

            imgWhite = np.ones((self.imgSize, self.imgSize, 3), np.uint8) * 255
            imgCrop = img[y - self.offset:y + h + self.offset, x - self.offset:x + w + self.offset]

            imgCropShape = imgCrop.shape

            aspectRatio = h / w

            if aspectRatio > 1:
                k = self.imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, self.imgSize))
                imgResizeShape = imgResize.shape
                wGap = math.ceil((self.imgSize - wCal) / 2)
                imgWhite[:, wGap:wCal + wGap] = imgResize
                prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
                self.predicted_word = self.labels[index]

            else:
                k = self.imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (self.imgSize, hCal))
                imgResizeShape = imgResize.shape
```

```
            hGap = math.ceil((self.imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
            self.predicted_word = self.labels[index]

        self.word_label.config(text=self.predicted_word)

        # Display resized hand image on panel2
        imgResize = cv2.cvtColor(imgResize, cv2.COLOR_BGR2RGB)
        imgResize = Image.fromarray(imgResize)
        imgtk = ImageTk.PhotoImage(image=imgResize)
        self.panel2.imgtk = imgtk
        self.panel2.config(image=imgtk)

    imgOutput = cv2.cvtColor(imgOutput, cv2.COLOR_BGR2RGB)
    imgOutput = cv2.resize(imgOutput, (640, 480))

    imgOutput = Image.fromarray(imgOutput)
    imgtk = ImageTk.PhotoImage(image=imgOutput)
    self.panel.imgtk = imgtk
    self.panel.config(image=imgtk)

    self.root.after(10, self.video_loop)

def speak_word(self):
    # Add code to generate audio for the predicted word (self.predicted_word)
    pass

def clear_word(self):
    self.word_label.config(text="")
    self.predicted_word = ""

def stop(self):
    self.cap.release()
    self.root.after(2000, self.root.destroy)

def switch_to_tamil(self):
    # Update labels and model for Tamil letters
    self.labels = ['அ', 'ஆ', 'இ', 'ஈ', 'உ', 'ஊ', 'எ', 'ஏ', 'ஐ', 'ஒ', 'ஓ', 'ஔ', 'ஃ', 'க்',
'ங்','ச்','ஞ்','ட்','ண்','த்','ந்','ப்','ம்','ய்','ர்','ல்','வ்','ழ்','ள்','ற்','ன்']
    self.classifier = Classifier("Model/uyirmei_t_model.h5", "Model/um_t_labels.txt")
```

```
    def switch_to_words(self):
        # Update labels and model for English words
        self.labels = ['Bathroom', 'Break', 'Call', 'Cute', 'Dad', 'Dislike', 'Drink', 'Fighting', 'Food',
'Happy', 'Hello', 'I Love You', 'Its Me', 'Like', 'Milk', 'Mom', 'Name', 'No', 'Ok', 'Peace',
'Promise', 'Ready', 'Saying', 'Sibling', 'Silence', 'Stop', 'Terrific', 'Thank You', 'Water', 'Yes']
        self.classifier = Classifier("word_model.h5", "labels.txt")

root = tk.Tk()
app = Application(root)
root.mainloop()
```

templates/index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Speakable</title>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel='stylesheet' type='text/css' href='../static/home.css'>
    <link rel='icon' type='image/icon type' href='../static/tabIcon.png'>
</head>
<body>
    <div class="app-container">
        <div class="app-left">
            <button class="close-menu">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0
0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round" class="feather feather-x">
                    <line x1="18" y1="6" x2="6" y2="18"/>
                    <line x1="6" y1="6" x2="18" y2="18"/>
                </svg>
            </button>
            <div class="app-logo">
                <svg xmlns="http://www.w3.org/2000/svg" width="40" height="40" viewBox="0
0 172 172" style="fill:#000000;">
                    <g fill="#f0f0f0">                    </g>
                </svg>
                <span>Speakable</span>
            </div>
```

```
        <ul class="nav-list">
          <li id='recogniseButton' class="nav-list-item active">
            <a class="nav-list-link" href="#">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 172 172" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round">
                <g fill="#dbdbdb" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round">

                </g>
              </svg>
              Recognise
            </a>
          </li>
          <li id='teamButton' class="nav-list-item">
            <a class="nav-list-link" href="#">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 172 172" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round">
                <g fill="#dbdbdb" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round">                    </g>
              </svg>
              Team
            </a>
          </li>
        </ul>
      </div>
      <div class="app-main" id='recogniseAppMain'>
        <div class="main-header-line">
          <h1>Recognise Sign Language</h1>
          <div class="action-buttons">
            <button class="menu-button">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" class="feather feather-menu">
                <line x1="3" y1="12" x2="21" y2="12"/>
                <line x1="3" y1="6" x2="21" y2="6"/>
                <line x1="3" y1="18" x2="21" y2="18"/>
              </svg>
            </button>
          </div>
        </div>
```

```
        <div class="main-content-container">
            <div class="main-content-wrapper big">
                <div class="main-content">
                    <div class="main-content-header">
                        <h2>Capture the Sign Language signs here</h2>
                        <span>Get set go!</span>
                    </div>
                    <div class="main-content-data">
                        <div class="video-container">
                            <img id="bg" src="{{ url_for('video_feed') }}" alt="Video Stream">
                        </div>
                    </div>
                    <div id="toggle-sign-language">
                        <input type="checkbox" class="checkbox" id="languageToggle"/>
                        <div class="knobs"><span>Tamil Letters</span></div>
                        <div class="layer"></div>
                    </div>
                    <button id='startCameraButton'>Start Camera and Recognise the Sign
Language</button>
                </div>
            </div>
            <div class="main-content-wrapper small">
                <div class="main-content">
                    <div class="main-content-header">
                        <h2>Prediction Panel</h2>
                        <span>Sign Language to Speech</span>
                    </div>
                    <div style='width:100%'>
                        <table id='predictionPanelTable'>
                            <tr>
                                <td style="width:15%;">Sentence:</td>
                                <td style="text-align:center;" id='predictionPanelSentence'>-</td>
                                <td id="prediction-panel"></td>
                            </tr>
                        </table>
                        <button id='moveSentenceButton' onclick="speech()">Speak</button>
                    </div>
                </div>
            </div>
        </div>
        <div class="app-main" id='teamAppMain' style='display:none;'>
```

```
        <div class="main-header-line">
            <h1>Our Team</h1>
            <div class="action-buttons">
                <button class="menu-button">
                    <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" class="feather feather-menu"><line x1="3"
y1="12" x2="21" y2="12"/><line x1="3" y1="6" x2="21" y2="6"/><line x1="3" y1="18"
x2="21" y2="18"/></svg>
                </button>
            </div>
        </div>
        <section class='admins'>
            <div class='box'>
                <div class='admin' style='border-
radius:0.25rem;padding:1rem;display:flex;align-items:center;'>
                    <div>
                        <img width='75' height='75' src='../static/profile.png'/>
                    </div>
                    <div style='margin-left:1rem;'>
                        <h3>Nandhini S</h3>
                        <p>Regn. No. 1920102084</p>
                    </div>
                </div>
                <div class='admin' style='border-
radius:0.25rem;padding:1rem;display:flex;align-items:center;'>
                    <div>
                        <img width='75' height='75' src='../static/profile.png'/>
                    </div>
                    <div style='margin-left:1rem;'>
                        <h3>Pavithra S</h3>
                        <p>Regn. No. 1920102097</p>
                    </div>
                </div>
                <div class='admin' style='border-
radius:0.25rem;padding:1rem;display:flex;align-items:center;'>
                    <div>
                        <img width='75' height='75' src='../static/profile.png'/>
                    </div>
                    <div style='margin-left:1rem;'>
                        <h3>Rithikaezhil N</h3>
                        <p>Regn. No. 1920102111</p>
```

```
                </div>
            </div>
        </div>
    </section>
  </div>
</div>
<script src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js'></script>
<script type='text/javascript' src='../static/home.js'></script>
<script>
  document.getElementById('languageToggle').addEventListener('click', function() {
    fetch('/toggle_language', { method: 'POST' })
    .then(response => response.json())
    .then(data => {
      console.log(data.message);

    })
    .catch(error => console.error('Error:', error));
  });


  function updatePredictionPanel(signClass) {
    document.getElementById('prediction-panel').innerText = signClass;
  }

  function fetchSignClass() {
    fetch('/sign_class')
      .then(response => response.json())
      .then(data => {
        updatePredictionPanel(data.sign_class);
      })
      .catch(error => {
        console.error('Error:', error);
      });
  }

  function speech() {
    const signClass = document.getElementById('prediction-panel').innerText;
    if (signClass && signClass !== "-") {
      const utterance = new SpeechSynthesisUtterance(signClass);
      utterance.lang = 'ta-IN'; // Setting the language to Tamil

      // Checking and setting a Tamil voice if available
```

```
            const voices = window.speechSynthesis.getVoices();
            const tamilVoice = voices.find(voice => voice.lang === 'ta-IN');
            if (tamilVoice) {
                utterance.voice = tamilVoice;
            } else {
                console.log('Tamil voice not found. Using default voice.');
            }

            speechSynthesis.speak(utterance);
        } else {
            console.log("No text to speak");
        }
    }

    // It might be necessary to wait until the voices are loaded
    window.speechSynthesis.onvoiceschanged = function() {
        speech();
    };


    // Fetch the sign class initially
    setInterval(fetchSignClass, 1000); // Update every second
</script>
</body>
</html>
```

static/home.js:

```
[...document.querySelectorAll('.menu-button')].forEach(function(item){
    item.addEventListener('click', function()
    {
        document.querySelector('.app-left').classList.add('show');
    });
});
[...document.querySelectorAll('.close-menu')].forEach(function(item){
    item.addEventListener('click', function()
    {
        document.querySelector('.app-left').classList.remove('show');
    });
});
$("#recogniseButton").click(function()
{
```

```
    $("#dictionaryAppMain").css("display", "none");
    $("#dictionary1AppMain").css("display", "none");
    $("#teamAppMain").css("display", "none");
    $("#recogniseAppMain").css("display", "block");
    $(this).addClass('active').siblings().removeClass('active');
    if(document.querySelector('.app-left').classList.contains('show'))
    {
        document.querySelector('.app-left').classList.remove('show');
    }
});
$("#teamButton").click(function()
{
    $("#recogniseAppMain").css("display", "none");
    $("#dictionaryAppMain").css("display", "none");
    $("#dictionary1AppMain").css("display", "none");
    $("#teamAppMain").css("display", "block");
    $(this).addClass('active').siblings().removeClass('active');
    if(document.querySelector('.app-left').classList.contains('show'))
    {
        document.querySelector('.app-left').classList.remove('show');
    }
});
$("#speak").on("submit",function(event){
    event.preventDefault();
    var voiceSelect = document.getElementById("voiceOptions");
    var utterThis=new SpeechSynthesisUtterance($("#speakText").val());
    var selectedOption=voiceSelect.selectedOptions[0].getAttribute('data-name');
    var voices = window.speechSynthesis.getVoices();
    for(i=0;i<voices.length;i++)
    {
        if(voices[i].name===selectedOption)
        {
            utterThis.voice=voices[i];
        }
    }
    window.speechSynthesis.speak(utterThis);
});
function getScrollHeight(elm)
{
    var savedValue = elm.value
    elm.value = ''
    elm._baseScrollHeight = elm.scrollHeight
```

```
    elm.value = savedValue
}
document.addEventListener('input', function({target:elm}){
    if(!elm.classList.contains('autoExpand')||!elm.nodeName=='TEXTAREA')
    {
        return;
    }
    var minRows = elm.getAttribute('data-min-rows')|0, rows;
    !elm._baseScrollHeight && getScrollHeight(elm);
    elm.rows = minRows;
    rows = Math.ceil((elm.scrollHeight - elm._baseScrollHeight) / 16);
    elm.rows = minRows + rows;
});
```

app.py:

```
from flask import Flask, render_template, Response, jsonify
import cv2
import numpy as np
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import math

language = "english"

app = Flask(__name__)

class HandRecognition:
    def __init__(self):
        global language
        self.detector = HandDetector(maxHands=1)
        self.classifier = Classifier("word_model.h5", "labels.txt")
        self.cap = cv2.VideoCapture(0)
        self.offset = 20
        self.imgSize = 300
        self.update_classifier()

    def update_classifier(self):
        global language
        if language == "tamil":
            self.classifier = Classifier("Model/uyirmei_t_model.h5", "Model/um_t_labels.txt")
            self.labels = ['அ', 'ஆ', 'இ', 'ஈ', 'உ', 'ஊ', 'எ', 'ஏ', 'ஐ', 'ஒ', 'ஓ', 'ஔ', 'ஃ', 'க்',
```

```
'ங்','ச்','ஞ்','ட்','ண்','த்','ந்','ப்','ம்','ய்','ர்','ல்','வ்','ழ்','ள்','ற்','ன்']
        else:
            self.classifier = Classifier("word_model.h5", "labels.txt")
            self.labels = ['Bathroom', 'Break', 'Call', 'Cute', 'Dad', 'Dislike', 'Drink', 'Fighting',
'Food', 'Happy', 'Hello', 'I Love You', 'Its Me', 'Like', 'Milk', 'Mom', 'Name', 'No', 'Ok', 'Peace',
'Promise', 'Ready', 'Saying', 'Sibling', 'Silence', 'Stop', 'Terrific', 'Thank You', 'Water', 'Yes']

    def toggle_language(self):
        global language
        language = "tamil" if language == "english" else "english"
        self.update_classifier()

    def get_frame(self):
        while True:
            success, img = self.cap.read()
            imgOutput = img.copy()
            hands, img = self.detector.findHands(img)
            if hands:
                hand = hands[0]
                x, y, w, h = hand['bbox']

                imgWhite = np.ones((self.imgSize, self.imgSize, 3), np.uint8) * 255
                imgCrop = img[y - self.offset:y + h + self.offset, x - self.offset:x + w + self.offset]

                imgCropShape = imgCrop.shape

                aspectRatio = h / w

                if aspectRatio > 1:
                    k = self.imgSize / h
                    wCal = math.ceil(k * w)
                    imgResize = cv2.resize(imgCrop, (wCal, self.imgSize))
                    imgResizeShape = imgResize.shape
                    wGap = math.ceil((self.imgSize - wCal) / 2)
                    imgWhite[:, wGap:wCal + wGap] = imgResize
                    prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
                    predicted_word = self.labels[index]

                else:
                    k = self.imgSize / w
                    hCal = math.ceil(k * h)
                    imgResize = cv2.resize(imgCrop, (self.imgSize, hCal))
```

```
                imgResizeShape = imgResize.shape
                hGap = math.ceil((self.imgSize - hCal) / 2)
                imgWhite[hGap:hCal + hGap, :] = imgResize
                prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
                predicted_word = self.labels[index]

            ret, buffer = cv2.imencode('.jpg', cv2.cvtColor(imgOutput,
cv2.COLOR_BGR2BGRA))
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

        else:
            ret, buffer = cv2.imencode('.jpg', cv2.cvtColor(imgOutput,
cv2.COLOR_BGR2BGRA))
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

    def detect_sign_class(self):
        while True:
            success, img = self.cap.read()
            imgOutput = img.copy()
            hands, img = self.detector.findHands(img)
            if hands:
                hand = hands[0]
                x, y, w, h = hand['bbox']

                imgWhite = np.ones((self.imgSize, self.imgSize, 3), np.uint8) * 255
                imgCrop = img[y - self.offset:y + h + self.offset, x - self.offset:x + w + self.offset]

                imgCropShape = imgCrop.shape

                aspectRatio = h / w

                if aspectRatio > 1:
                    k = self.imgSize / h
                    wCal = math.ceil(k * w)
                    imgResize = cv2.resize(imgCrop, (wCal, self.imgSize))
                    imgResizeShape = imgResize.shape
                    wGap = math.ceil((self.imgSize - wCal) / 2)
                    imgWhite[:, wGap:wCal + wGap] = imgResize
```

```
                prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
                predicted_word = self.labels[index]

            else:
                k = self.imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (self.imgSize, hCal))
                imgResizeShape = imgResize.shape
                hGap = math.ceil((self.imgSize - hCal) / 2)
                imgWhite[hGap:hCal + hGap, :] = imgResize
                prediction, index = self.classifier.getPrediction(imgWhite, draw=False)
                predicted_word = self.labels[index]

            return predicted_word

    def release_camera(self):
        self.cap.release()

@app.route('/')
def index():
    return render_template('index1.html')

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

@app.route('/sign_class')
def sign_class():
    detected_sign_class = hand_recognition.detect_sign_class()
    return jsonify({"sign_class": detected_sign_class})

@app.route('/toggle_language', methods=['POST'])
def toggle_language():
    hand_recognition.toggle_language()
    return jsonify({"message": language})

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

SCREENSHOTS

Proposed CNN:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 298, 298, 32)      896

 max_pooling2d_2 (MaxPoolin  (None, 149, 149, 32)      0
 g2D)

 conv2d_3 (Conv2D)           (None, 147, 147, 64)      18496

 max_pooling2d_3 (MaxPoolin  (None, 73, 73, 64)        0
 g2D)

 flatten_1 (Flatten)         (None, 341056)            0

 dense_1 (Dense)             (None, 64)                21827648

 dense_2 (Dense)             (None, 31)                2015

=================================================================
Total params: 21849055 (83.35 MB)
Trainable params: 21849055 (83.35 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 6.1. Proposed CNN

Epoch:

```
47/47 [==============================] - 97s 2s/step - loss: 2.6909 - accuracy: 0.6532 - val_loss: 0.0499 - val_accuracy: 0.986
6
Epoch 2/10
47/47 [==============================] - 68s 1s/step - loss: 0.0104 - accuracy: 0.9980 - val_loss: 0.0432 - val_accuracy: 0.992
0
Epoch 3/10
47/47 [==============================] - 61s 1s/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0243 - val_accuracy: 0.994
6
Epoch 4/10
47/47 [==============================] - 61s 1s/step - loss: 1.4306e-05 - accuracy: 1.0000 - val_loss: 0.0261 - val_accuracy:
0.9946
Epoch 5/10
47/47 [==============================] - 73s 2s/step - loss: 4.5822e-06 - accuracy: 1.0000 - val_loss: 0.0285 - val_accuracy:
0.9946
Epoch 6/10
47/47 [==============================] - 77s 2s/step - loss: 2.7588e-06 - accuracy: 1.0000 - val_loss: 0.0282 - val_accuracy:
0.9946
Epoch 7/10
47/47 [==============================] - 68s 1s/step - loss: 2.0445e-06 - accuracy: 1.0000 - val_loss: 0.0305 - val_accuracy:
0.9946
Epoch 8/10
47/47 [==============================] - 71s 2s/step - loss: 1.7255e-06 - accuracy: 1.0000 - val_loss: 0.0295 - val_accuracy:
0.9946
Epoch 9/10
47/47 [==============================] - 76s 2s/step - loss: 1.3664e-06 - accuracy: 1.0000 - val_loss: 0.0300 - val_accuracy:
0.9946
Epoch 10/10
47/47 [==============================] - 88s 2s/step - loss: 1.1473e-06 - accuracy: 1.0000 - val_loss: 0.0306 - val_accuracy:
0.9946
```

Figure 6.2. Trained Epoch

Real time prediction:

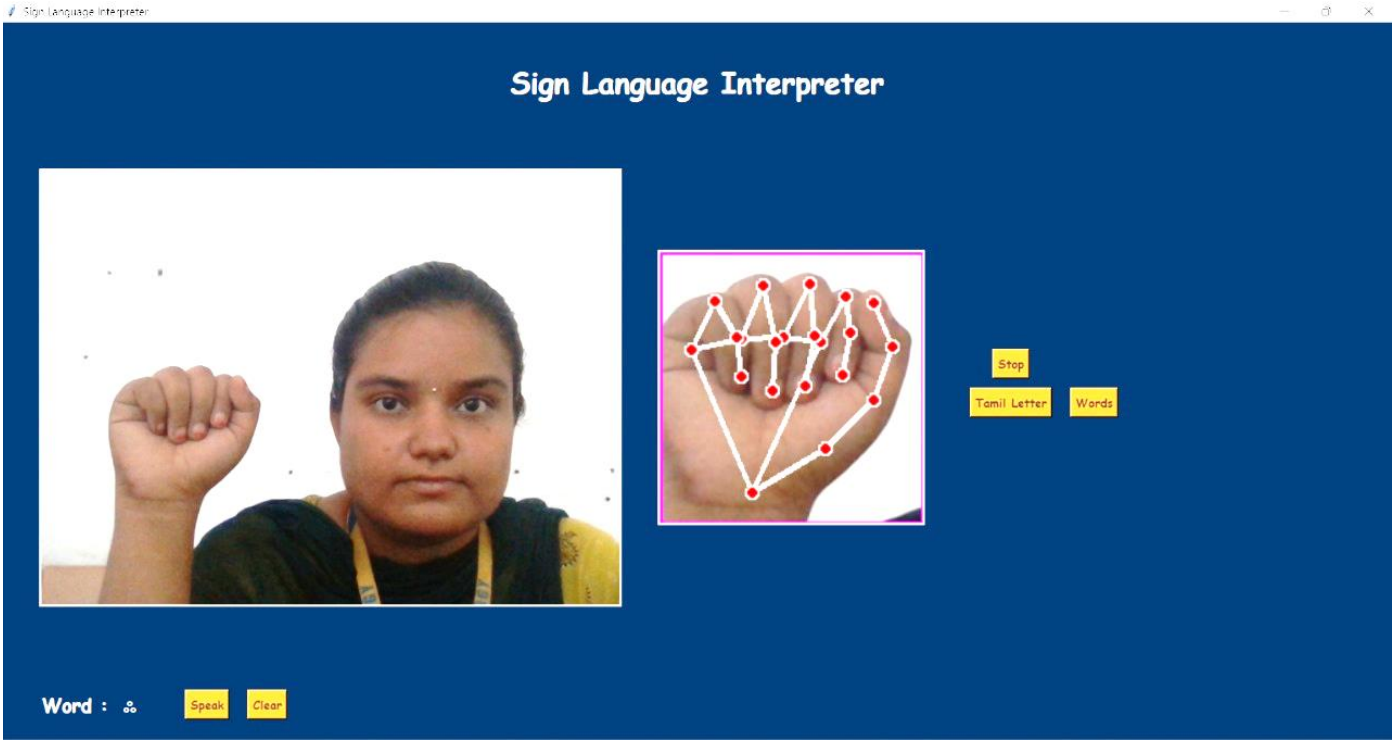


Figure 6.3. GUI output – tamil letters

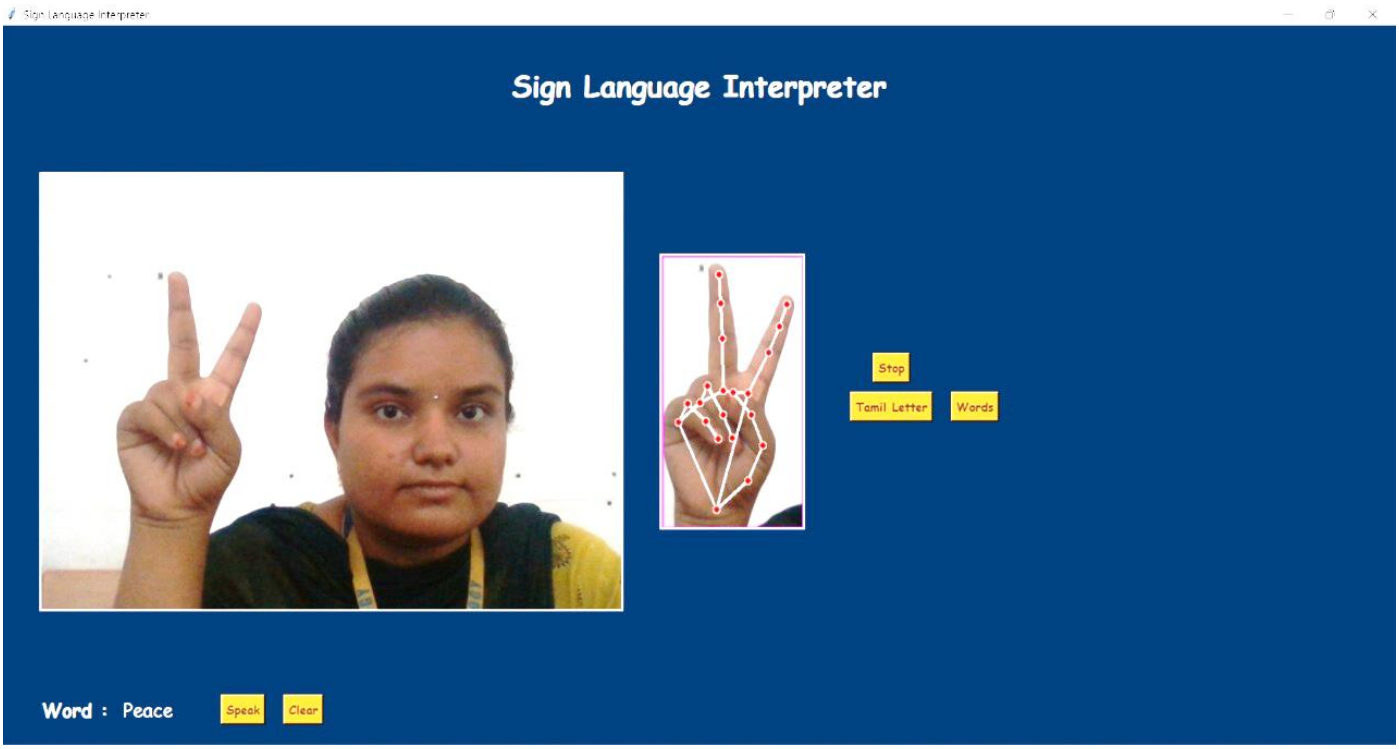


Figure 6.4. GUI output - words

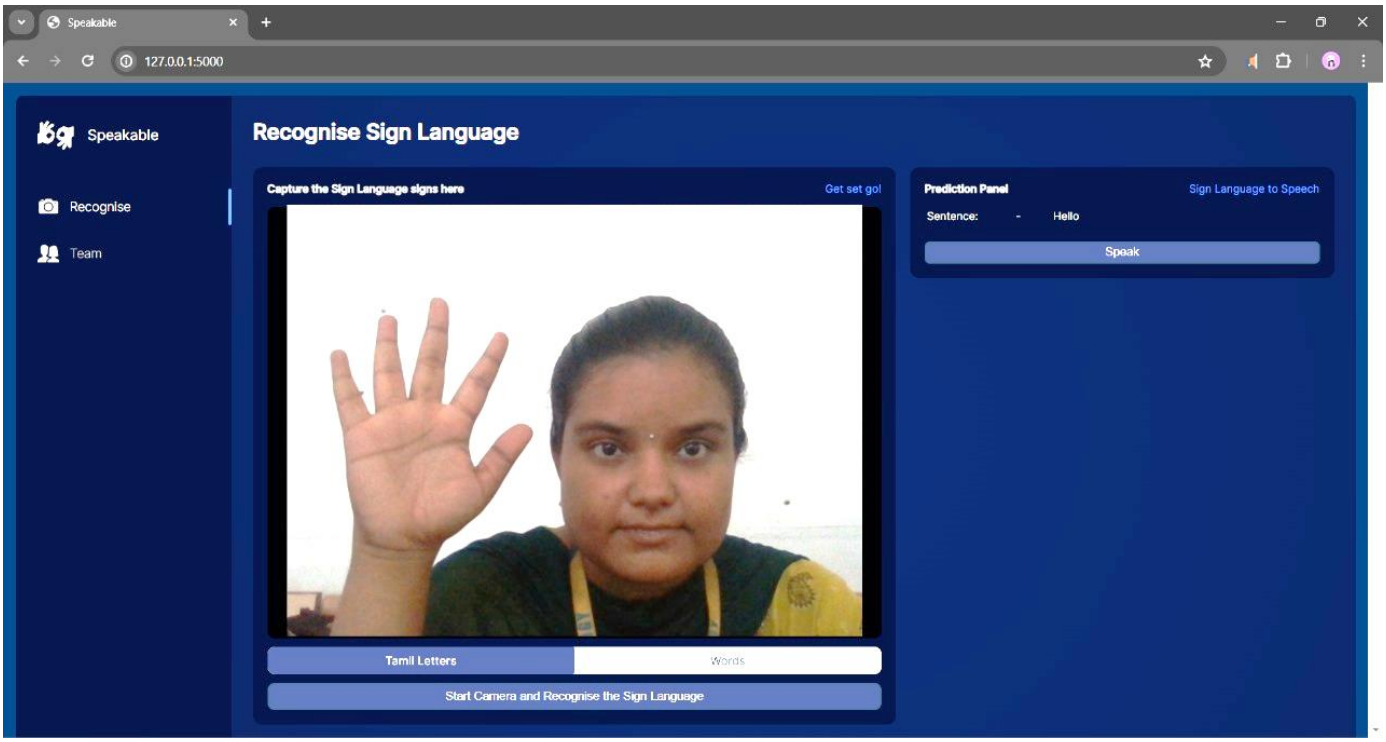


Figure 6.5. Webpage – tamil letter

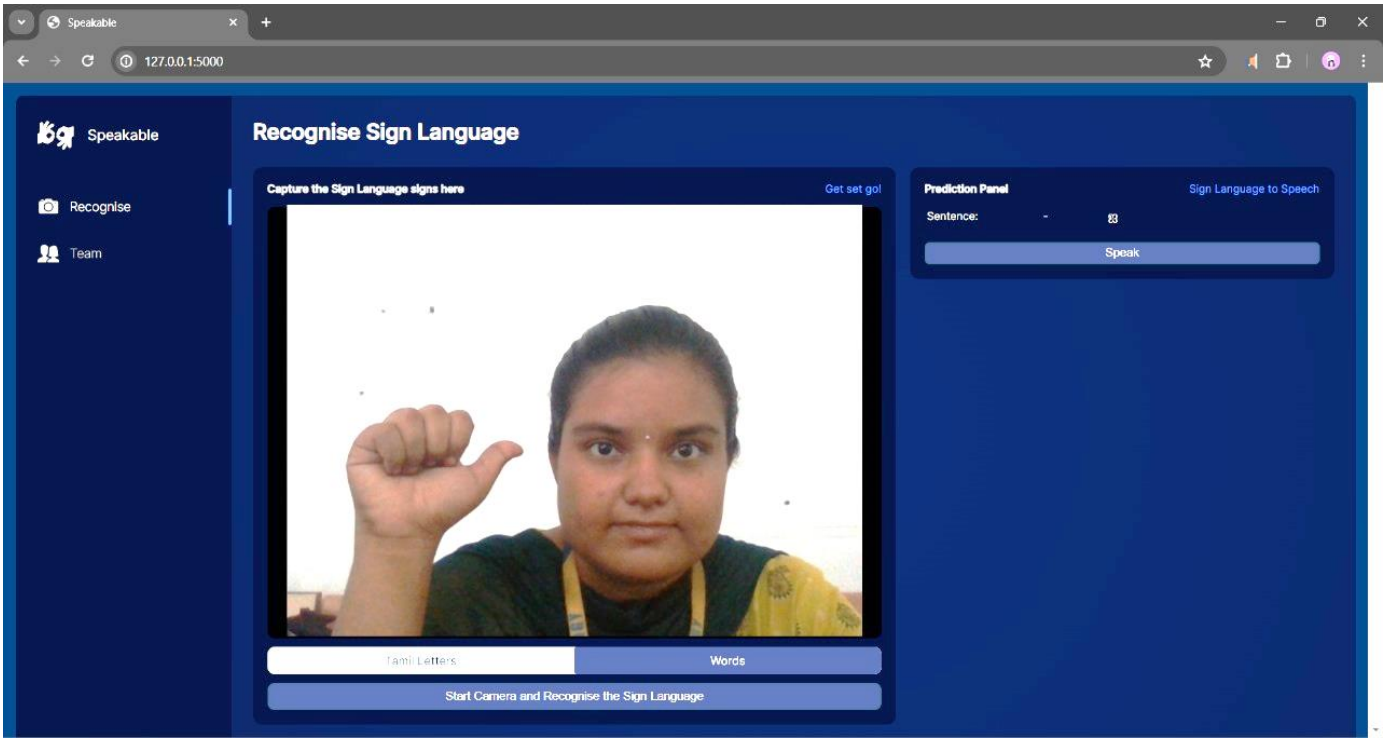


Figure 6.6. Webpage - words

REFERENCES

[1] Jayesh S Sonkusare, Nilkanth B Chopade, Ravindra Sor, Sunil L Trade "A review on hand gesture recognition system" 2015. International conference on computing communication control and Automation.

[2] Prof Radha S Shirbate, Mr Vedant D Shinde, Ms Sanam A Metkari, Ms Pooja U Borkar, Ms Mayuri A Khangde, "Sign Language Recognition using machine learning algorithm"(2020).

[3] Raksha Amer Kadhim and Mentadher kameer "A real-time american sign language recognition system using convolutional neural network for real datasets"TEM joprnal 9(3):937,2020.

[4] Sai Bharat Pandigala, Gogineni Hrushikesh, Mahav, Saranu Kishore kumar,"Video based sign language recognition using CNN-LSTM", Published on IRJET (2022).

[5] Jiao, P., Min, Y., Li, Y., Wang, X., Lei, L., & Chen, X. (2023). CoSign: Exploring Co-occurrence Signals in Skeleton-based Continuous Sign Language Recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 20676-20686).

[6] Kaur, J., & Krishna, C. R. (2019). An efficient Indian sign language recognition system using sift descriptor. International Journal of Engineering and Advanced Technology, 8(6), 1456-1461.

[7] Suardi, C., Handayani, A. N., Asmara, R. A., Wibawa, A. P., Hayati, L. N., & Azis, H. (2021, April). Design of Sign Language Recognition Using E-CNN. In 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT) (pp. 166-170). IEEE.

[8] Yin, A., Zhao, Z., Jin, W., Zhang, M., Zeng, X., & He, X. (2022). Mlslt: Towards multilingual sign language translation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5109-5119).

[9] Bheda, V., & Radpour, D. (2017). Using deep convolutional networks for gesture recognition in american sign language. arXiv preprint arXiv:1710.06836.

[10] Manikandan, K., Patidar, A., Walia, P., & Roy, A. B. (2018). Hand gesture detection and conversion to speech and text. arXiv. arXiv preprint arXiv:1811.11997.

[11] Li, J., Kang, D., Pei, W., Zhe, X., Zhang, Y., He, Z., & Bao, L. (2021). Audio2gestures: Generating diverse gestures from speech audio with conditional variational autoencoders. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 11293-11302).

[12] Skobov, V., & Bono, M. (2023, December). Making Body Movement in Sign Language Corpus Accessible for Linguists and Machines with Three-Dimensional Normalization of MediaPipe. In Findings of the Association for Computational Linguistics: EMNLP 2023 (pp. 1844-1855).

[13] Singh, D. K. (2021). 3d-cnn based dynamic gesture recognition for indian sign language modeling. Procedia Computer Science, 189, 76-83.

[14] Katoch, S., Singh, V., & Tiwary, U. S. (2022). Indian Sign Language recognition system using SURF with SVM and CNN. Array, 14, 100141.

[15] Srivastava, Sharvani & Gangwar, Amisha & Mishra, Richa & Singh, Sudhakar. (2022). Sign Language Recognition System Using TensorFlow Object Detection API. 10.1007/978-3-030-96040-7_48

[16] Huang, J., Zhou, W., Zhang, Q., Li, H., and Li, W., "Video-based Sign Language Recognition without Temporal Segmentation", 2018. doi:10.48550/arXiv.1801.10111.

[17] Ahuja, M.K. and Singh, A., 2015. Hand gesture recognition using PCA. International Journal of Computer Science Engineering and Technology (IJCSET), 5(7), pp.267-27.

[18] Bhagat, Neel Kamal; Vishnusai, Y.; Rathna, G. N. (2019). [IEEE 2019 Digital Image Computing: Techniques and Applications (DICTA) - Perth, Australia (2019.12.2-2019.12.4)] 2019 Digital Image Computing: Techniques and Applications (DICTA) - Indian Sign Language Gesture Recognition using Image Processing and Deep Learning.

[19] Karayilan, Tulay; Kilic, Ozkan (2017). [IEEE 2017 International Conference on Computer Science and Engineering (UBMK) - Antalya (2017.10.5-2017.10.8)] 2017 International Conference on Computer Science and Engineering (UBMK) - Sign language recognition. , (), 1122–1126. doi:10.1109/UBMK.2017.8093509

[20] Ghaleb, F. , Youness, E. , Elmezain, M. and Dewdar, F. (2015) Vision-Based Hand Gesture Spotting and Recognition Using CRF and SVM. Journal of Software Engineering and Applications, 8, 313-323. doi: 10.4236/jsea.2015.87032

[21] Katoch, S., Singh, V. and Tiwary, U.S., 2022. Indian Sign Language recognition system using SURF with SVM and CNN. Array, 14, p.100141

[22] Abey Abraham, V Rohini,Real time conversion of sign language to speech and prediction of gestures using Artificial Neural Network,Procedia Computer Science,Volume 143,2018,Pages 587-594,ISSN1877-0509,https://doi.org/10.1016/j.procs.2018.10.435.

[23] NB, M.K., 2018. Conversion of sign language into text. International Journal of Applied Engineering Research, 13(9), pp.7154-7161.

[24] Mehreen Hurroo , Mohammad Elham, 2020, Sign Language Recognition System using Convolutional Neural Network and Computer Vision, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH& TECHNOLOGY (IJERT) Volume 09, Issue 12 (December 2020)

[25] KASAPBAŞI, Ahmed & Elbushra, Ahmed & AL-HARDANEE, Omar & YILMAZ, Arif. (2022). DeepASLR: A CNN based Human Computer Interface for American Sign Language Recognition for Hearing-Impaired Individuals. Computer Methods and Programs in Biomedicine Update. 2. 100048. 10.1016/j.cmpbup.2021.100048

[26] Thakur, Amrita & Budhathoki, Pujan & Upreti, Sarmila & Shrestha, Shirish & Shakya, Subarna. (2020). Real Time Sign Language Recognition and Speech Generation. Journal of Innovative Image Processing. 2. 65-76. 10.36548/jiip.2020.2.001

[27] Rachana Patil, Vivek Patil, Abhishek Bahuguna, Gaurav Datkhile, Indian Sign Language Recognition using Convolutional Neural Network, ITM Web Conf. 40 03004 (2021),DOI: 10.1051/itmconf/20214003004

[28] Kodandaram, Satwik Ram & Kumar, N. & Gl, Sunil. (2021). Sign Language Recognition. Turkish Journal of Computer and Mathematics Education (TURCOMAT). 12. 994-1009

[29] Singh, D.K., 2021. 3d-cnn based dynamic gesture recognition for indian sign language modeling.

[30] Ghaleb, F. , Youness, E. , Elmezain, M. and Dewdar, F. (2015) Vision-Based Hand Gesture Spotting and Recognition Using CRF and SVM. Journal of Software Engineering and Applications, 8, 313-323. doi: 10.4236/jsea.2015.87032

[31] Srivastava, Sharvani & Gangwar, Amisha & Mishra, Richa & Singh, Sudhakar. (2022). SignLanguage Recognition System Using TensorFlow Object Detection API. 10.1007/978-3-030-96040-7_48

[32] Huang, J., Zhou, W., Zhang, Q., Li, H., and Li, W., "Video-based Sign Language Recognition withoutTemporal Segmentation", 2018. doi:10.48550/arXiv.1801.10111.

[33] Sreenivas, Arvind & Maheshwari, Mudit & Jain, Saiyam & Choudhary, Shalini & G, Dr. Vadivu. (2020). Indian Sign Language Communicator Using Convolutional Neural Network. International Journalof Advanced Science and Technology. 29. 11015-11031.

[34] Ahuja, M.K. and Singh, A., 2015. Hand gesture recognition using PCA. International Journal of Computer Science Engineering and Technology (IJCSET), 5(7), pp.267-27.

[35] Jayanthi, J., Lydia, E. L., Krishnaraj, N., Jayasankar, T., Babu, R. L., & Suji, R. A. (2021). An effective deep learning features based integrated framework for iris detection and recognition. Journal of ambient intelligence and humanized computing, 12, 3271-3281.

[36] Elavarasi, S. A., Jayanthi, J., & Basker, N. (2019). Trajectory object detection using deep learning algorithms. Int. J. Recent Technol. Eng, 8.

[37] Sridharan, M., Arulanandam, D. C. R., Chinnasamy, R. K., Thimmanna, S., & Dhandapani, S. (2021). Recognition of font and tamil letter in images using deep learning. Applied Computer Science, 17(2).