

A Real Time Research Project/ Societal Related Project Report
On
Real Time Sign Interpretation System
Submitted in partial fulfillment of the requirements for the award of the
Bachelor of Technology
In
Department of Computer Science and Engineering

By

Pathuri Krishnama Chary	22241A050D
Makkala Vinod	22241A055B
Siddoju Nagavarshith	22241A050E
Dondapati Ravi Teja	22241A05Z7

Under the Esteemed guidance of

Dr. B.Srinivasa Rao
Professor



Department of Computer Science and Engineering
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)
Bachupally, Kukatpally, Hyderabad, Telangana, India, 500090
2023-2024



GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

CERTIFICATE

This is to certify that the Real Time Research Project/ Societal Related Project entitled “**Real Time Sign Interpretation System**” is submitted by **Pathuri Krishnama Chary (22241A050D)**, **Makkala Vinod (22241A055B)**, **Siddoju Nagavarshith (22241A050E)**, **Dondapati Ravi Teja (22241A05Z7)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2023-2024**.

INTERNAL GUIDE
Dr. B.SRINIVASA RAO
Professor

HEAD OF THE DEPARTMENT
Dr. B. SANKARA BABU
Professor and HOD

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **Dr. B.Srinivasa Rao, Professor**, Department of CSE for his support in the completion of our project report.

We wish to express our honest and sincere thanks to **Ms. KVSL Harika** for coordinating in conducting the project reviews, **Dr.B. Sankara Babu, HOD**, Department of CSE for providing resources, and to the principal **Dr. J. Praveen** for providing the facilities to complete our Real Time Research Project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Pathuri Krishnama Chary 22241A050D

Makkala Vinod 22241A055B

Siddoju Nagavarshith 22241A050E

Dondapati Ravi Teja 22241A05Z7

DECLARATION

We here by declare that the Real Time Research Project entitled “**Real Time Sign Interpretation System**” is the work done during the period from **2023- 2024** and is submitted in the fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad)**. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Pathuri Krishnama Chary 22241A050D

Makkala Vinod 22241A055B

Siddoju Nagavarshith 22241A050E

Dondapati Ravi Teja 22241A05Z7

	Table of Contents	
Chapter	Title	Page No.
	Abstract	1
1	Introduction	2
2	System Requirements	6
	2.1 Software Requirements	6
	2.2 Hardware Requirements	6
	2.3 Data Set	7
3	Literature Survey	8
4	Proposed Model, Modules Description, and UML Diagrams	16
	4.1 Modules	17
	4.2 UML Diagrams	20
5	Implementation, Experimental Results & Test Cases	26
6	Conclusion and Future Scope	43
7	References	45
	Appendix i) Snapshot of the Result	47

LIST OF FIGURES		
Fig. No.	Title	Page No.
4.3.1	System Architecture	20
4.3.2	Use Case Diagram	21
4.3.3	Sequence Diagram	22
4.3.4	Class Diagram	23
4.3.5	Component Diagram	24

ABSTRACT

This project tackles the challenge of communication barriers between deaf and mute individuals and the general population. We propose a sign language recognition system using ensemble deep learning, a powerful approach combining two techniques. Convolutional Neural Networks (CNNs) excel at image analysis, automatically detecting key landmarks within hand gestures – a crucial step for recognition. Random Forest Classifiers (RFCs) then analyze these patterns to accurately interpret the entire sign. By combining CNNs and RFCs, the model achieves improved accuracy and robustness, handling variations in hand postures and lighting conditions.

The impact extends beyond research. The trained model interprets hand gestures captured in real-time using cameras or sensors, translating them into text for seamless communication. Designed for widespread deployment across platforms (mobile, desktop), this technology fosters inclusivity and empowers deaf/mute individuals to engage more effectively in daily activities. Benefits extend further, aiding educators, healthcare professionals, and emergency responders in communication and interaction with deaf/mute individuals.

Prioritizing real-world usability, the model focuses on landmark detection for flexibility in hand posture and lighting conditions. Additionally, this focus allows for potential offline functionality, making sign recognition possible even in areas with limited internet access. This feature expands the system's reach and impact, contributing to a more inclusive and connected world.

This project doesn't just break down communication barriers; it has the potential to empower entire communities. Imagine a classroom where a deaf student can fully participate in discussions, their hand gestures translated into text in real-time. Educators can utilize this technology to create a more inclusive learning environment, fostering deeper understanding and engagement for all students. Similarly, healthcare professionals can leverage the system to ensure clear communication during consultations. Deaf patients can accurately express their concerns and receive proper medical care, leading to improved healthcare outcomes. In emergency situations, first responders can utilize the technology to understand the needs of deaf individuals, potentially saving lives through faster and more effective responses.

Chapter 1

INTRODUCTION

1.1 EXISTING SYSTEM:

Existing sign language recognition systems employ a variety of techniques and datasets to achieve their goals. Analyzing these approaches reveals some interesting insights.

Firstly, the datasets used encompass various sign languages and data sources. Some systems focus on static signs captured in images (MNIST, custom datasets), while others explore real-time capture using webcams and YouTube videos. The complexity of the datasets also varies, with some focusing on single signs like those found in the MNIST dataset and others utilizing more comprehensive collections like COCO and ImageNet. However, a potential limitation is the reliance on creating custom datasets, which might restrict the generalizability of the system.

Deep learning techniques, particularly Convolutional Neural Networks (CNNs), dominate the field for feature extraction and classification. This is likely because CNNs excel at image analysis, a crucial step in sign recognition. Additionally, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) are finding applications in handling the sequential nature of signs, where the order of hand movements matters. It's important to note that traditional machine learning techniques like Support Vector Machines (SVMs) and Bag-of-Visual-Words (BoVW) models are still present alongside deep learning. This suggests that established techniques continue to play a role in sign language recognition, even with the rise of deep learning.

Some projects explore combining multiple models or data sources, highlighting efforts to improve performance through integration and fusion. However, a key challenge seems to be balancing accuracy with efficiency. While some systems prioritize high accuracy, they might require substantial processing power, limiting their usability on resource-constrained devices. Additionally, limited datasets or models that struggle with variations in signing styles suggest challenges in adapting to real-world scenarios. Finally, the reliance on large, manually labeled datasets for training highlights a potential barrier for some projects.

The data showcases the ongoing development in sign language recognition. There's a mix of established and emerging techniques being used, but limitations in efficiency, adaptability, and data dependence highlight the need for further advancements in this field.

1.1.1 Limitations in the Existing System:

Existing sign language recognition systems, while promising, face hurdles that limit their real-world application. One major challenge is computational complexity. Some systems require significant processing power, making them unsuitable for mobile devices or environments with limited resources. This can hinder their widespread adoption.

Another limitation is accuracy for complex signs. Current solutions might struggle with intricate hand movements or subtle variations in gestures. This can lead to misinterpretations and hinder effective communication.

Accessibility is another concern. Some projects rely on specialized equipment like motion tracking gloves or high-resolution cameras. This dependence on specific hardware makes them less accessible for everyday use by the general population.

Even beyond hardware, some existing solutions might have high memory usage and overall system overhead. This can limit their scalability and deployment on devices with limited resources.

Furthermore, current projects might have a narrow scope, focusing on a specific set of signs. Alternatively, they may require trade-offs between accuracy and efficiency. A system with high accuracy might be computationally expensive, while a faster system might compromise accuracy for complex signs.

Integration challenges also exist. Integrating existing systems with other applications or platforms can be difficult. Additionally, dependence on vast amounts of high-quality training data can pose a challenge for some projects.

Finally, current solutions might be inflexible. They might struggle with recognizing new signs that haven't been included in the training data. Additionally, some systems might not adapt well to variations in sign execution or background noise, hindering their performance in real-world scenarios with diverse signing styles and environments.

These limitations highlight the need for more efficient, accessible, and adaptable sign language recognition systems. The ideal system would balance accuracy, resource efficiency, and real-world usability to bridge the communication gap for deaf and mute individuals.

1.2 Proposed System:

This project proposes a novel sign language recognition system using ensemble deep learning

to bridge the communication gap between deaf and mute individuals and the general population. The core technology leverages two powerful techniques: Convolutional Neural Networks (CNNs) act as the foundation, automatically detecting key features (landmarks) from captured hand gestures. Random Forest Classifiers (RFCs) then analyze the patterns within these landmarks to interpret the entire sign gesture. This two-step approach using CNNs and RFCs ensures both high accuracy and robustness in recognizing signs even with variations in hand postures and lighting conditions.

Networks (CNNs) act as the foundation, automatically detecting key features (landmarks) from captured hand gestures. Random Forest Classifiers (RFCs) then analyze the patterns within these landmarks to interpret the entire sign gesture. This two-step approach using CNNs and RFCs ensures both high accuracy and robustness in recognizing signs even with variations in hand postures and lighting conditions.

The proposed system goes beyond academic research and aims for real-world impact. The trained model interprets hand gestures captured in real-time using cameras or sensors, translating them into text for seamless communication. Designed for deployment across various platforms (mobile, desktop), this technology fosters inclusivity by removing communication barriers. Deaf/mute individuals are empowered to participate more actively in daily activities and social interactions. Additionally, the model's focus on landmark detection allows for potential offline functionality, making sign recognition possible even in areas with limited internet access. This feature expands the system's reach and impact, empowering entire communities through inclusive education, improved healthcare, and enhanced emergency response.

1.2.1 Advantages over Existing System:

This proposed sign language recognition system offers several advantages over existing solutions by leveraging ensemble deep learning with a focus on real-world usability. Here's a breakdown of the key benefits:

Enhanced Accuracy and Performance: By combining the strengths of Convolutional Neural Networks (CNNs) for image analysis and Random Forest Classifiers (RFCs) for pattern recognition, the system achieves a high degree of accuracy in sign detection. This is particularly beneficial for complex gestures or signs that might be challenging for traditional methods.

Reduced Complexity and Resource Requirements: The proposed system prioritizes landmark detection, potentially leading to a simpler underlying model. This translates to lower computational demands during operation. Unlike existing solutions that might rely on specialized hardware, this system could potentially function on devices with lower processing power, making it more accessible.

Increased Efficiency: The combination of a simpler model and lower resource requirements leads to an overall more efficient system. This translates to faster processing times and potentially lower power consumption, making it suitable for real-time applications on mobile devices.

Reduced Model Overhead: A less complex model might require less memory to run, leading to a lower overall system overhead. This can be crucial for deployment on resource- constrained devices.

Improved Adaptability: The focus on landmark detection allows for greater flexibility in recognizing variations in hand postures and lighting conditions. Additionally, the system might be easier to modify to recognize entirely new signs compared to existing, inflexible projects. This adaptability is essential for the system to remain effective in diverse real-world scenarios.

Reduced Reliance on Manual Data Construction (potential): Depending on the specific implementation, the system might leverage pre-trained models or require less extensive manual data collection for training. This can significantly reduce development time and effort.

Chapter 2

System Requirements

2.1 System Requirements:

2.1.1 Deep Learning Framework (TensorFlow or similar):

TensorFlow is a popular open-source framework developed by Google for building, training, and deploying machine learning models.

It offers a wide range of tools and functionalities specifically designed for deep learning applications.

In this project, TensorFlow would be used to create the ensemble model combining CNNs and RFCs.

2.1.2 Programming Language: Python.

Python is the preferred language for development due to its readability, extensive libraries, and large developer community.

Libraries like OpenCV would be used for computer vision tasks like image processing and landmark detection from hand gestures.

2.1.3 Operating System: Windows, macOS, or Linux.

The choice of operating system depends on developer preference and the target platform (mobile vs desktop). Most deep learning frameworks, including TensorFlow, are compatible with all three major operating systems.

2.2 Hardware Requirements:

2.2.1 Camera or Sensor:

The system captures hand gestures using a camera or sensor. This could be a webcam on a computer, a smartphone camera for portability, or even a specialized depth sensor for enhanced accuracy, especially in varying lighting conditions. Depth sensors provide more detailed information about the distance of objects in the camera's view, potentially improving sign recognition.

2.2.2 Processor:

Real-time processing of video requires a mid-range or higher processing power unit (CPU or GPU). The specific processing power depends on the target platform (desktop, laptop, or

mobile device). Devices with more processing power can handle complex models and real-time video processing more efficiently.

Memory (RAM):

The amount of RAM needed varies depending on the model's complexity. Generally, 4GB or higher is recommended for smooth operation. More RAM allows the system to handle complex calculations and store data during processing.

Storage (HDD/SSD):

Enough storage space (typically a few gigabytes) is required to house the trained model and the application itself. Solid State Drives (SSDs) offer faster read/write speeds compared to traditional Hard Disk Drives (HDDs), potentially improving system performance.

2.3 Data Set:

The data set used for this project consists of hand gesture images belonging to 15 classes. These images were collected using a camera or sensor and processed using the MediaPipe library for landmark detection. Instead of storing the raw images, we extracted the landmark coordinates from each image, resulting in a custom data set of landmark coordinates.

Each data sample in the data set represents the landmarks detected in a hand gesture image. The landmark coordinates were normalized and stored as numpy arrays, resulting in a data set of shape (N, 42), where N is the number of samples and 42 represents the x and y coordinates of 21 landmarks detected in a hand gesture.

The corresponding labels for the hand gesture images were also stored, indicating the class each hand gesture belongs to. These labels were encoded as integers ranging from 0 to 14, representing the 15 classes: 'One', 'Two', 'Three', 'Super', 'Four', 'Hello', 'Ok', 'Smile', 'Love', 'Victory', 'Help', 'Let's Go', 'Call', 'Danger(help)' and 'Please'.

The data set was split into training and testing subsets using the `train_test_split` function from the `sklearn.model_selection` module. The training subset was used to train machine learning models, while the testing subset was used to evaluate the performance of the trained models.

Chapter 3

Literature Survey

Automatic Recognition of Indian Sign of Alphabets ,Digits Robin

- **S. Katoch ,V.Singh**
- **29 March 2022**

There are two sets inside the dataset. Eighty percent of the total data is the training set, while the remaining twenty percent is utilized for testing. SVM and CNN, the two classifiers, have both demonstrated excellent accuracy on the images.

SVM reported a 99.14% accuracy rate on the test set. A total accuracy of 99% is shown by the computed values of precision and recall for alphabets and digits identified using SVM. The accuracy by class. In the most recent epoch, an overall accuracy of 94% on the training set was recorded using CNN, while an accuracy of more than 99% was observed during testing. There are 50 epochs in all. Our model is trained using the softmax function as the activation function and a categorical cross entropy loss function.

Interrupt Local and Global Contexts Ferdaous

- **Renjie**
- **2nd January 2022**

CNN performance on image classification tasks differed based on model architecture, dataset size, preprocessing methods, and training scheme. In general, CNNs shown remarkable accuracy on benchmark datasets like as ImageNet, CIFAR-10, and MNIST.

CNNs achieved accuracies ranging from 70% to 90% after switching to CIFAR-10. This dataset, which consists of 32x32 color photos from ten classes, is more complex than others, but CNNs performed well by taking use of their capacity to collect hierarchical characteristics. When it came to ImageNet, CNNs demonstrated impressive progress, with top-5 accuracies over 90%. While ImageNet presented a formidable challenge with millions of high-resolution photos in thousands of categories, cutting-edge CNN architectures like as

ResNet, Inception, and EfficientNet routinely outperformed human performance in several areas.

A Framework for Interpretable Automatic Evaluation

- **Fajrir Kato, Timothy Badain**
- **27 November 2020**

We provide the FFCI evaluation framework, which is built on four key components: inter-sentential coherence, faithfulness, focus, and coverage. We have demonstrated that the most reliable metrics for assessing fidelity are BERTScore (roberta-base), focus and coverage are BERTScore (gpt2-xl), and inter-sentential coherence is NSP-score. Overall, we find that ROUGE has improved contemporary summarization systems, although its fine-grained interpretability is lacking.

Modern abstractive summarizing systems over CNN/DailyMail have significantly improved on emphasis from the LSTM-based 27 Koto, Baldwin, & Lau seq2seq, according to FFCI, with coverage not much better than Lead3 until more recent systems. Finally, even though our study of assessment techniques from earlier work served as the foundation for the design of FFCI, we think there are certain more factors that need to be taken into consideration.

Real Time Sign Language Detection

- **Aman Pathak, Avinash Kumar, Gunjan Chigh**
- **26th December 2021**

The primary goal of a sign language detecting system is to give normal people and the dumb a practical means of communicating through hand gestures. We can infer from the model's conclusion that, in controlled light and intensity conditions, the suggested system can produce reliable results.

Additionally, adding new gestures is simple, and increasing the number of photos captured at various angles and frame rates will improve the model's accuracy. Therefore, by expanding the dataset, the model may be readily expanded on a wide scale. The model has several limitations, including uncontrolled background and low light intensity, which reduce the detection accuracy of the model. As a result, our next tasks will be to fix these issues and expand the dataset for more beautiful.

Sign Language Recognition Using Convolutional Neural Network

- D.Shiva Roopam,S.Yogesh

- 23 February 2022

Real-time video feedback was used to evaluate the device, and the outcomes were totaled. Either way, the success of the SL identification system is flawless. The maximum precision on all the signs is 90%. This indicates that the machine can comprehend a variety of signals. The suggested technique accurately predicts certain popular words and signs in a variety of illumination and speed scenarios.

The process of accurately masking the photos involves providing a range of values that have the ability to dynamically detect human hands. CNN is used by the suggested system to learn and classify images. Finely retrieved and utilized are more informative elements from the photos for training and classification. .. To provide an accurate result, a total of 1750 static photos are used for each indication throughout the training process. Ultimately, the output of the identified sign is shown both as text and as speech. The algorithm can identify 125 words, even those that start with an alphabet.

Predicting Employability of Congolese Information Technology

- Heritier Nsenge Mpia, Simon Nyaga Mwendia

-2nd October 2022

Using contextual factors to predict the employability of Congolese information technology graduates toward sustainableEmployability for use in nations like the Democratic Republic of the Congo.According to the study's findings, there are five contextual elements that can predict an IT graduate's employability in the Democratic Republic of the Congo.

The employability of IT graduates in the DRC is significantly predicted by three of these five characteristics. These variables include relationship, academic, and sociopolitical aspects. They have a significant impact on employability since, according

to Cron-bach's Alpha test, their reliabilities were 0.82, 0.75, and 0.63, respectively. Two criteria were chosen to have a modest impact on these graduates' employability. These two had a dependability score of 0.52, which was for parental financial stability, and 0.53, which was for strategy.

Consequently, this study offers variables that reflect the reality and forecast employability in unstable developing nations.

A Dynamic Interaction Graph CNN based on learnable object detection

-Yuhaojin

-4th April 2024

Current object detection techniques typically produce redundant and nearly duplicate findings when applied to dense candidates. In order to increase the object detection accuracy, we therefore suggest a dynamic interaction GCN module in the DIGCN that handles relational modeling and dynamic interaction on the proposal boxes and proposal features.

Furthermore, in order to reduce the number of hand-designed object candidates and avoid complex tasks like object candidate design and many-to-one label assignment, we introduce a learnable proposal method with a sparse set of learned object proposals. This approach

also somewhat reduces the complexity of the object detection model. On the difficult COCO dataset, DIGCN shows accuracy and runtime performance comparable to the established and extensively optimized detector baselines.

A Hybrid Forecasting Model Based on CNN and Informer for Short-Term Wind Power

-Hai-Kun, WangKe , Yi Cheng

-24 January 2022

This paper proposes a composite network that is composed of a convolutional neural network and Informer and that uses this model to improve the prediction accuracy of wind power due to the instability and intermittency of wind power generation in a complex environment and to better obtain the historical wind power data.

For verification, historical wind power statistics from a Chinese wind farm are used and compared with Informer, LSTM, RNN, and DeepAR. The following is a list of this paper's specific contributions:

Using a convolution neural network, the original historical wind power data are split into many

time scales, and additional time series features are recovered. Historical wind power data can be more effectively utilized with this strategy.

A Static and Dynamic Arabic Sign Language Recognition System Based on Machine and Deep Learning Models

-Essam Hisham, Sherine Nagy Saleh

-06 December 2022

This work presents a comparative study of many machine learning classifier models for forest fire prediction using two distinct datasets. The processing of the suggested system is dependent on temperature, humidity, wind, oxygen content, and humidity. Numerous machine learning classification algorithms, including decision trees, k neighbors, random forests, logistic regression, support vector classifiers, and decision trees, were used in this work.

K-fold cross validation techniques and hyperparameter tuning were used to optimize the model. With 96.88% accuracy, the system determined that Support Vector Machine was the most effective method for the forest fire dataset. With 90.24% accuracy, the Random Forest approach proved to be highly effective for the Cortez and Morais dataset

A transformer model for boundary detection in continuous sign language

-Razieh Rastgoo¹ , Kourosh Kiani¹ , Sergio Escalera

- 22 March 2024

In recent years, academics have focused a great deal of emphasis on Sign Language Recognition (SLR), especially in the complicated field of Continuous Sign Language Recognition (CSLR), which is more complex than Isolated Sign Language Recognition (ISLR). Accurately identifying the borders of isolated signs within a continuous video stream is one of the major issues in computer vision and speech recognition (CSLR).

A further obstacle to reaching optimal accuracy is the current models' reliance on manually created features. To overcome these obstacles, we provide a novel strategy based on a

Transformer-based paradigm. In contrast to conventional models, our method aims to improve accuracy without requiring hand-drawn features. The ISLR and CSLR both use the Transformer model..

Fast and Lightweight Vision-Language Model for Adversarial Traffic Sign Detection

-Furkan Mumcu Yasin Yilmaz

-5th June 2024

Numerous machine learning (ML)-powered attacks have been proposed against autonomous vehicles and their subsystems. Particularly extensively evaluated in a variety of adversarial ML attack scenarios, road sign recognition models have shown themselves to be susceptible. The suggested defense strategy works with any current traffic sign recognition system and is based on a unique, quick, lightweight, and salable vision-language model (VLM).

approach reliably identifies a wide range of assaults against distinct target models with very low false positive rates and high true positive rates. Our suggested detector obtains an average AUC of 0.94 when evaluated against four cutting-edge assaults that target four widely used action recognition models. Compared to the state-of-the-art protection technique suggested for generic picture assault detection, which produces an average AUC of 0.75, our result delivers a 25.3% improvement

ARPES detection of superconducting gap sign in unconventional superconductors

- Qiang Gao¹, Jin Mo Bok, Ping A

- 7 May 2024

To use ARPES to identify the super-conducting gap sign. In the ARPES measurements of Bi₂212 with a known d-wave gap symmetry, the approach is well-tested. The Fermi momentum shift, the Bogoli- Ubov band hybridization, and the aberrant superconducting gap behaviors are examples of how the gap sign appears in the resulting electronic structures superconducting state.

We have shown that ARPES may be phase sensitive and predictable in determining the pairing symmetry, which is important for comprehending the superconductivity mechanism of unconventional superconductors, in addition to detecting the superconducting gap size. able to estimate how much of the burned forest will go. But applying image processing can increase accuracy.

Sign Language Word Detection Using LRCNtors

- Md. Shaheenur Islam Sumon, Muttakee Bin Ali

- 06 April 2024

The most efficient form of communication for those who are deaf or hard of hearing is sign language. Because understanding sign language requires specialized training, those without impairments cannot effectively communicate with them. This study's primary goal is to create a method for simplifying the deep learning model for sign language recognition by using the 30 terms that are most commonly used in daily speech.

The dataset was created using thirty terms from American Sign Language (ASL) that were custom-processed video sequences with five individuals and fifty sample films per class. The Pose dataset and the Raw video dataset are two different datasets whose results we present and assess. The Long-term Recurrent Convolutional Network (LRCN) technique was used to train the dataset. Ultimately, a test accuracy of 93.66% for the posture dataset and 92.66% for the raw dataset was obtained.

Sign Language Detection Using Action Recognition LSTM Deep Learning Model

-Purushotam Kr. Singh

-07 May 2024

For the Deaf and Hard of Hearing (DHH) community, sign language is an essential communication tool, yet computational systems still struggle to recognize it. Using a Long Short-Term Memory (LSTM) deep learning model and action recognition principles, this research study proposes a novel technique to sign language detection. By utilizing the

sequential and temporal dynamics of sign language movements, the LSTM model is trained to precisely recognize and categorize signs in video footage. Key frame features are extracted from video sequences by the suggested approach and sent into the LSTM network. We train and assess the model using an extensive dataset that includes a of sign language.

Sign Language Detection and Translation using Smart Glove

- **Abhay Chopde, Ansh Magon, Shreyas Bhatkar**
- **10 April 2022**

For people who are deaf or hard of hearing, sign language communication is crucial. But the communication gap between people who use sign language and people who do not understand it makes it difficult for people to engage and participate effectively in many aspects of life. This research project is developing a smart glove-based system for real-time sign language translation and detection in order to address this urgent challenge. One of the project's objectives is to develop and build a prototype smart glove with an MPU-6050 and five flex sensors.

These sensors capture hand and finger movements that are very accurate and necessary for sign language.

Chapter 4

Proposed Approach, Module Description and UML Diagrams

4.1 Proposed Approach:

This project aims to bridge communication barriers between deaf/mute individuals and the general population by developing a robust sign language recognition system. By leveraging an ensemble deep learning approach, the system combines Convolutional Neural Networks (CNNs) and Random Forest Classifiers (RFCs) to enhance accuracy and robustness.

4.1.1 Convolutional Neural Networks (CNNs):

Objective: Detect and identify key landmarks in hand gestures from images.

Architecture:

Input Layer: Accepts images of hand gestures. **Convolutional Layers:** Extract features using filters.

Pooling Layers: Reduce dimensionality while retaining important features.

Fully Connected Layers: Classify features into landmark coordinates.

Output Layer: Provides key landmark coordinates (e.g., positions of fingers and hand orientation).

4.1.2 Random Forest Classifiers (RFCs):

Objective: Classify detected landmarks into specific sign language symbols. **Structure:**

Input: Key landmark coordinates from the CNN.

Decision Trees: Multiple trees analyze subsets of the landmark data.

Ensemble Method: Aggregates predictions from all trees to finalize classification. **Output**

Layer: Predicted sign language symbol.

4.1.3 Model Integration:

Pipeline:

Image Capture: Use cameras or sensors to capture real-time hand gesture images. Landmark

Detection (CNN): Process images to extract key landmarks.

Sign Classification (RFC): Use landmark coordinates to predict the sign language symbol.

Output: Display the recognized sign language symbol to the user in real-time.

4.2 Modules

4.2.1 Data Collection and Preprocessing

Description:

This module is responsible for collecting raw data (images) using a webcam and preprocessing them to extract relevant features (hand landmarks) using the MediaPipe library. Each frame captured by the webcam undergoes processing to detect and extract specific hand landmarks, which are essential for subsequent stages in the system.

Implementation Details:

Utilizes OpenCV for capturing frames from the webcam. Implements MediaPipe Hands library for hand landmark detection. Preprocesses the captured images to normalize and extract hand landmarks. Stores the processed data in a suitable format for further use (e.g., pickle files).

Purpose:

To gather and preprocess raw data efficiently, ensuring that extracted features are suitable for training machine learning models.

4.2.2 CNN Model

Description:

This module involves the implementation and training of a Convolutional Neural Network (CNN) for sign language recognition. The CNN architecture is designed to process the extracted hand landmarks as inputs and predict the corresponding sign language gestures. Training involves optimizing the CNN model using labeled data, validating its performance,

and saving the trained model for real-time interpretation.

Implementation Details:

Uses TensorFlow and Keras for building the CNN architecture.

Defines layers such as Conv2D, MaxPooling2D, Flatten, and Dense.

Compiles the model with appropriate loss function, optimizer, and evaluation metrics. Trains the CNN model on preprocessed data and evaluates its accuracy.

Purpose:

To leverage deep learning techniques for effective pattern recognition in sign language gestures, providing robust and accurate predictions.

4.2.3 RFC Model

Description:

This module focuses on implementing and training a Random Forest Classifier (RFC) for sign language recognition. Unlike the CNN model, the RFC operates on extracted

representations derived from hand landmarks. It utilizes ensemble learning techniques to build a collection of decision trees, each contributing to the final classification outcome.

Implementation Details:

Utilizes OpenCV for capturing frames from the webcam. Implements MediaPipe Hands library for hand landmark detection. Preprocesses the captured images to normalize and extract hand landmarks. Stores the processed data in a suitable format for further use (e.g., pickle files).

Purpose: To gather and preprocess raw data efficiently, ensuring that extracted features are suitable for training machine learning models.

4.2.4 Feature Extraction

Description:

This module extracts and prepares features from raw data (hand landmarks) for input into machine learning models. It involves techniques to transform and represent raw data into a format suitable for model training and prediction. Feature extraction ensures that relevant information from hand gestures is effectively captured and utilized by the models.

Implementation Details:

Converts raw hand landmarks into feature vectors. Standardizes and preprocesses feature data for consistency. Implements techniques like normalization, scaling, or dimensionality reduction if necessary. Provides extracted features as input to both CNN and RFC models. Purpose: To enhance model performance by optimizing feature representation, ensuring effective utilization of input data for accurate predictions.

4.2.5 Ensembling Technique

Description:

This module integrates predictions from both the CNN and RFC models using an ensembling technique. Ensembling combines multiple models to achieve better predictive performance than individual models alone. Techniques such as averaging or voting may be employed to reconcile predictions from different models and produce a final output.

Implementation Details:

Combines predictions from trained CNN and RFC models. Applies simple averaging or more sophisticated ensemble methods. Determines the final sign language gesture prediction based on combined model outputs. Evaluates ensemble performance and compares it with individual model results. Purpose: To leverage the strengths of multiple models, enhancing overall prediction accuracy and robustness in real-time sign interpretation.

4.2.6 Evaluation and Metrics

Description:

This module evaluates the performance of the implemented system, focusing on metrics and criteria to assess model effectiveness and efficiency. It involves quantitative analysis of model predictions against ground truth data, calculating metrics such as accuracy, precision, recall, and F1-score. Evaluation helps validate the system's capability to accurately interpret sign language gestures under various conditions.

Implementation Details:

Computes evaluation metrics using test data sets. Compares performance metrics between CNN, RFC, and ensemble models. Analyzes strengths and weaknesses based on evaluation results. Provides insights into model reliability and real-world applicability.

Purpose: To objectively measure and validate the effectiveness of the implemented sign language

recognition system, guiding improvements and optimizations

4.3 UML (Unified Modeled Language) Diagrams

4.3.1 System Architecture

These features are aggregated to create a comprehensive feature representation for each gesture. The aggregated feature vector is fed into an RFC, which consists of multiple decision trees that vote to classify the gesture based on the combined CNN features. In real-time processing, input is captured via a camera, preprocessed, and passed through the trained CNNs for feature extraction and aggregation before the RFC performs the final gesture classification. The system is evaluated using metrics like accuracy, precision, recall, and F1-score, with hyperparameter tuning and cross-validation ensuring robustness.

Finally, the trained model is deployed in a production environment, integrated into an API, and made accessible through a user-friendly interface for end-users. Classified by the RFC, leading to the interpretation and display of the recognized gesture through a translation module.

Administratively, the system allows for database management and user account maintenance. This architecture ensures robust performance, leveraging advanced machine learning techniques to enhance accessibility and communication for individuals using sign language.

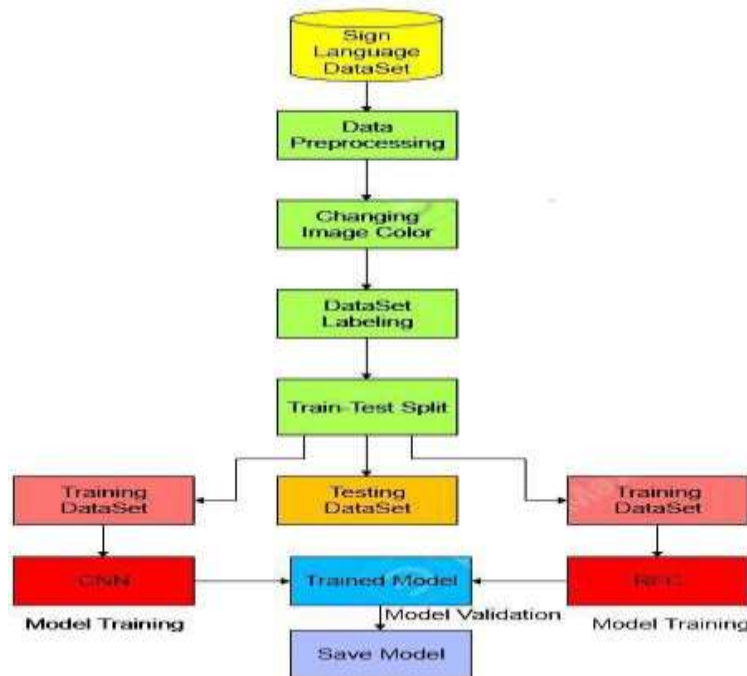


Fig 4.3.1 System Architecture Diagram

4.3.2 Use Case Diagram

A use case diagram for a sign language detection system using an ensemble of CNNs and RFC would feature the main actors, including the User and the Administrator, and their interactions with the system's functionalities. The User captures a gesture via a camera, which is then preprocessed, and features are extracted using CNNs. These features are aggregated and classified by the RFC to determine the gesture, and the translated gesture is displayed to the User. The Administrator manages user accounts and updates the gesture database. The diagram, enclosed within the system boundary, visually represents these interactions and functionalities, showcasing the flow from capturing a gesture to displaying its translation, along with the administrative capabilities for maintaining the system.

This aggregated feature vector is then passed to the RFC, which classifies the gesture based on the learned patterns. The system then displays the translated gesture to the User. Additionally,

the Administrator can manage user accounts and update the gesture database to ensure the system remains accurate and up-to-date. The use case diagram, encapsulated within the system boundary, visually demonstrates these interactions and workflows, providing a clear overview of the system's operation from capturing and processing gestures to classification and user .

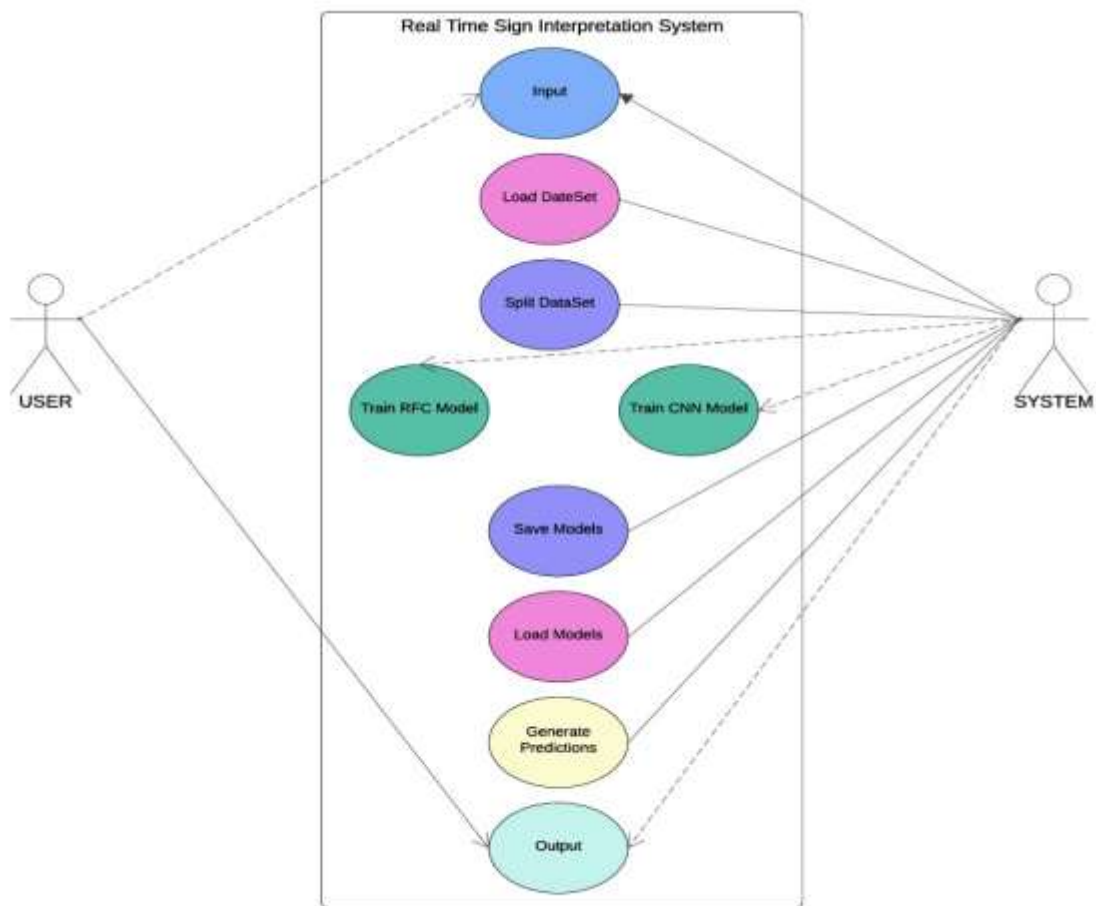


Fig 4.3.2 Use Case Diagram

4.3.3 Sequence Diagram

A sequence diagram for a sign language detection system using an ensemble of CNNs and RFC illustrates the flow of interactions over time among various components and actors involved in the system. The primary actors, the User and the Administrator, initiate interactions that trigger a series of processes within the system. When the User captures a gesture via a camera, the sequence begins with the captured data being sent to the preprocessing component,

which cleans and prepares the data. The extracted features from each CNN are aggregated into a single feature vector, which is subsequently sent to the RFC for classification. The RFC analyzes the aggregated features and determines the corresponding sign language gesture, which is then translated and displayed to the User. Concurrently, the Administrator can interact with the system to manage user accounts or update the gesture database, ensuring the system is current and functioning correctly. The sequence diagram captures these interactions in a step-by-step manner, emphasizing the temporal order and the dynamic relationships between the system components and actors, from the initial gesture capture to the final translation display and administrative tasks.

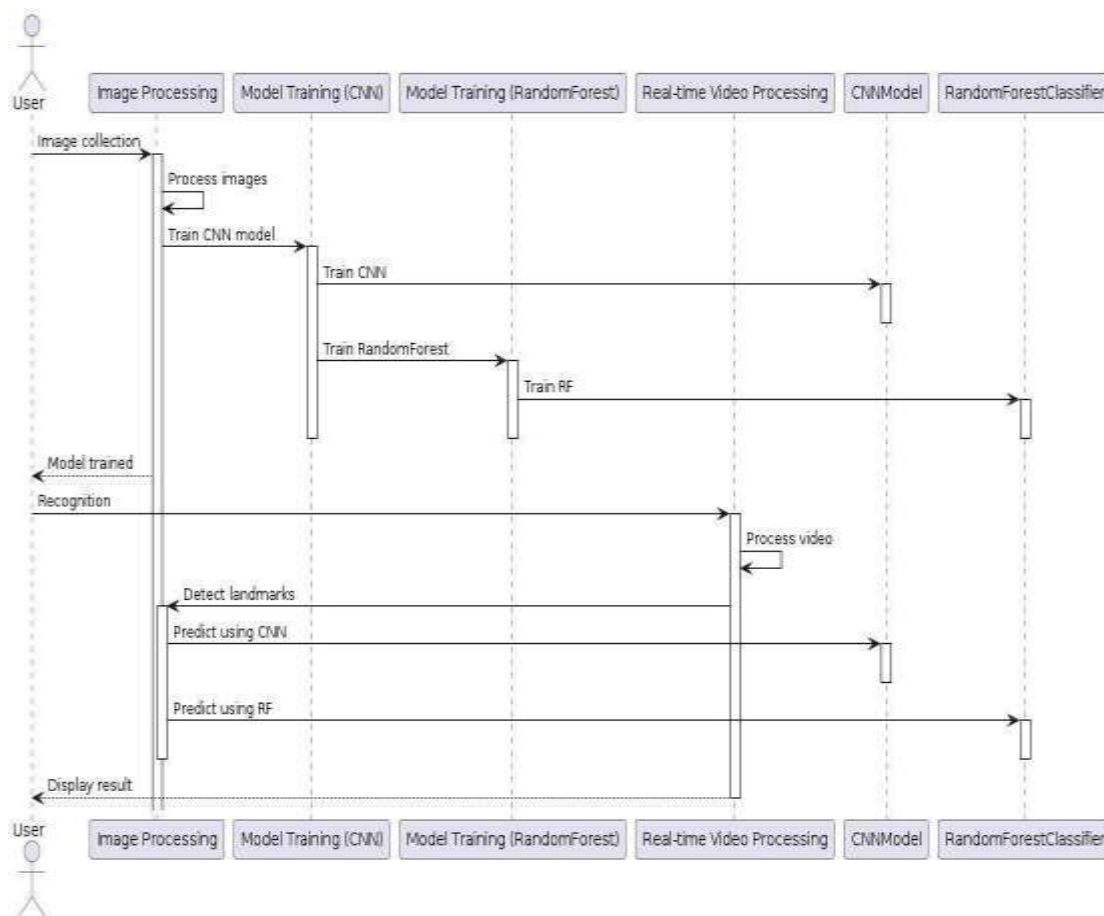


Fig 4.4.3 Sequence Diagram

4.3.4 Class Diagram

A class diagram for a sign language detection system employing an Ensemble of CNNs and RFC illustrates the static structure of the system by depicting the classes, their attributes,

methods, and the relationships between them. Key classes include User, Administrator, GestureCaptureSystem, Preprocessor, CNNModel, RFClassifier, and DatabaseManager. The User class interacts with the system to capture gestures, triggering methods in the GestureCaptureSystem. The captured data flows through the Preprocessor, which prepares it for feature extraction by multiple CNNModel classes. Each CNN model extracts specific features, which are then aggregated by the FeatureAggregator class.

The resulting feature vector is input to the RFClassifier, which classifies the gesture and sends the result to the TranslationModule for display to the User. The Administrator class manages system operations, including updating the GestureDatabase through the DatabaseManager class. Associations and dependencies between these classes illustrate how they collaborate to detect and translate sign language gestures, emphasizing the static structure and relationships within the system.

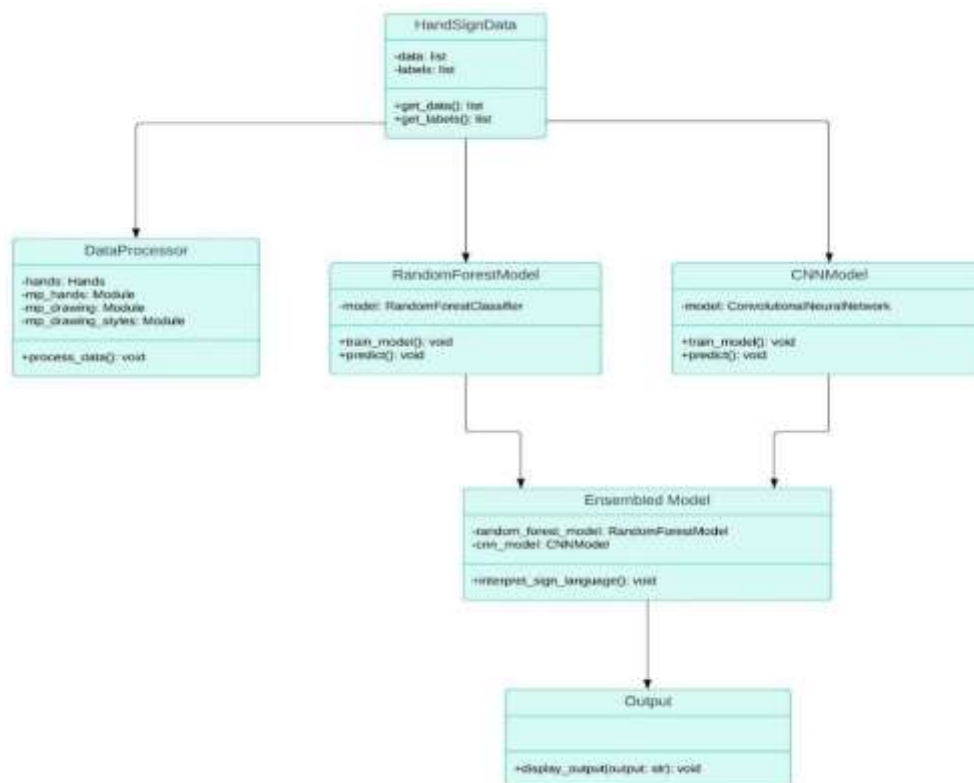


Fig 4.3.4 Class Diagram

4.3.5 Component Diagram

A component diagram for a sign language detection system utilizing an ensemble of CNNs and

RFC illustrates the physical and logical components of the system and their interactions. Key components include the User Interface, Gesture Capture Module, Preprocessing Module, CNN Ensemble, Feature Aggregation Module, RFC Classifier, Translation Module, Administrator Interface, and Database Management.

The User Interface allows interaction with the system to capture gestures, which are processed by the Gesture Capture Module and then preprocessed by the Preprocessing Module to ensure they are in a suitable format for analysis. The CNN Ensemble consists of multiple CNN models that extract features from the preprocessed data. These features are then aggregated by the Feature Aggregation Module to form a comprehensive feature representation.

The aggregated features are passed to the RFC Classifier, which uses them to classify the gesture. The result is then sent to the Translation Module for interpretation and display to the user.

The Administrator Interface allows system administrators to manage user accounts and update the Gesture Database through the Database Management component.

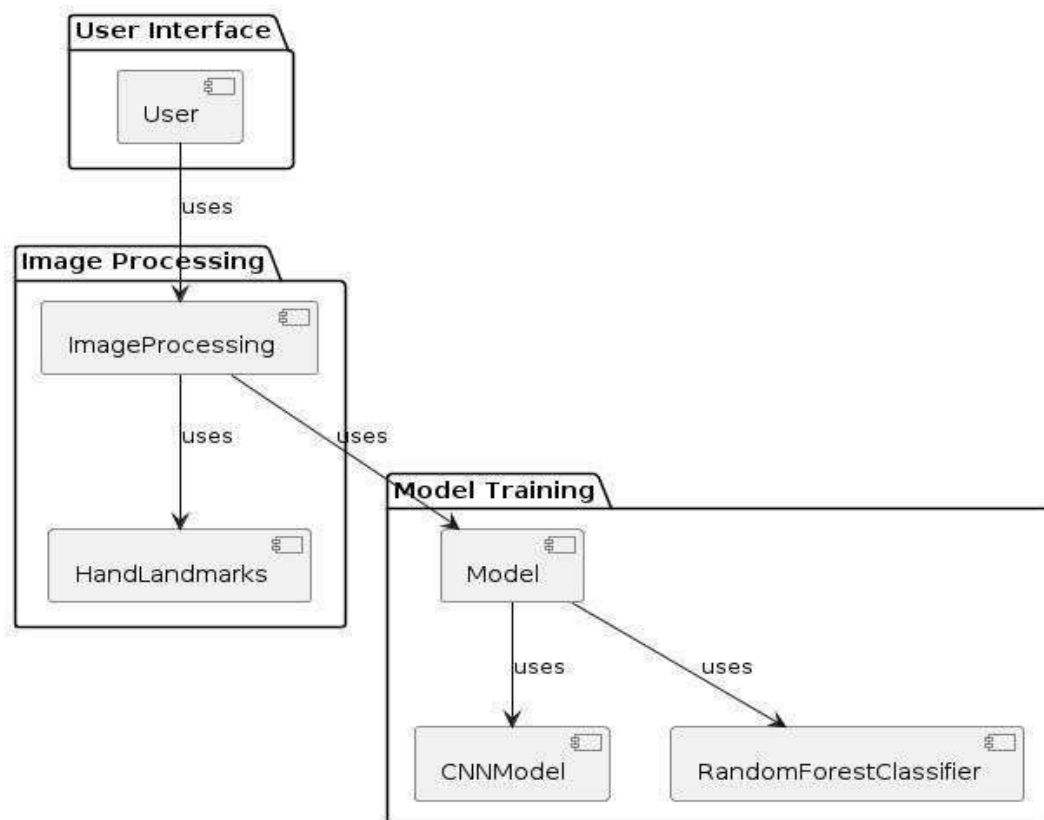


Fig 4.3.5 Component Diagram

Chapter 5

Implementation, Experimental Results and Test Cases

5.1 Implementation:

5.1.1 Collecting Images:

```
import os
import cv2
DATA_DIR = './new_data'
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)

number_of_classes = 15
dataset_size = 300

cap = cv2.VideoCapture(0)
for j in range(number_of_classes):
    if not os.path.exists(os.path.join(DATA_DIR, str(j))):
        os.makedirs(os.path.join(DATA_DIR, str(j)))

    print('Collecting data for class {}'.format(j))

    done = False
    while True:
        ret, frame = cap.read()
        cv2.putText(frame, 'Press Q', (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,
                    cv2.LINE_AA)
        cv2.imshow('frame', frame)
        if cv2.waitKey(50) == ord('q'):
            break

    counter = 0
    while counter < dataset_size:
        ret, frame = cap.read()
        cv2.imshow('frame', frame)
        cv2.waitKey(25)
        cv2.imwrite(os.path.join(DATA_DIR, str(j), '{}.jpg'.format(counter)), frame)

        counter += 1

cap.release()
cv2.destroyAllWindows()
```


5.1.2 Creating Dataset:

```
import os
import pickle

import mediapipe as mp
import cv2

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

DATA_DIR = './data_3'

data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []

        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

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

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

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

            data.append(data_aux)
            labels.append(dir_)

f = open('data_3.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

5.1.3 Training Random Forest Classifier Model

```
import pickle

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np

data_dict = pickle.load(open('./data_3.pickle', 'rb'))

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels)

model = RandomForestClassifier()

model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)

print('{}% of samples were classified correctly !'.format(score * 100))

f = open('num_model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

5.1.4 Training CNN Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import pickle
import numpy as np

with open('./data_3.pickle', 'rb') as f:
    data_dict = pickle.load(f)

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])
num_classes = len(np.unique(labels))

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels)

x_train = x_train.reshape(-1, 42, 1, 1) # 42 points (21 landmarks * 2 coordinates)
x_test = x_test.reshape(-1, 42, 1, 1)
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

cnn_model = Sequential([
    Conv2D(32, (3, 1), activation='relu', input_shape=(42, 1, 1)),
    MaxPooling2D(pool_size=(2, 1)), # Adjust pool size here
    Flatten(),
    Dense(128, activation='relu'),
    Dense(num_classes, activation='softmax')
])

cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

cnn_model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
cnn_model.save('cnn_num_model.h5')
```

5.1.5 Real Time Sign Interpretation:

```
import cv2
import mediapipe as mp
import numpy as np
from keras.models import load_model
import pickle

# Load the CNN model
cnn_model = load_model('./cnn_model.h5')

# Load the previous model
model_dict = pickle.load(open('./model2.p', 'rb'))
previous_model = model_dict['model']

cap = cv2.VideoCapture(0)

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

labels_dict = {0: 'Hello', 1: 'Super!', 2: 'Three', 3: 'Yo!', 4: 'I\'m #1 '}

while True:
    ret, frame = cap.read()

    H, W, _ = frame.shape

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(
                frame,
                hand_landmarks,
                mp_hands.HAND_CONNECTIONS,
                mp_drawing_styles.get_default_hand_landmarks_style(),
                mp_drawing_styles.get_default_hand_connections_style())
```

```

data_aux = []
x_ = []
y_ = []

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

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

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

# Convert data_aux to a format suitable for CNN input
cnn_data_input = np.expand_dims(np.array(data_aux).reshape(-1, 42, 1, 1), axis=3)

# Get predictions from both models
cnn_prediction = cnn_model.predict(cnn_data_input)
previous_prediction = previous_model.predict([np.asarray(data_aux)]).astype(np.float32)

# Ensemble predictions using simple averaging
ensemble_prediction = (cnn_prediction + previous_prediction) / 2

predicted_label = np.argmax(ensemble_prediction)

predicted_character = labels_dict[predicted_label]

# Adjust the position of the text
x1 = int(min(x_) * W) - 10
y1 = int(min(y_) * H) - 10

# Display the predicted character
cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 2, (0, 0, 255), 2,
            cv2.LINE_AA)

cv2.imshow('frame', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```


Front End Implementation:

5.1.6 App.py

```
from flask import Flask, render_template, Response, request, jsonify, stream_with_context
import cv2
import pickle
import mediapipe as mp
import numpy as np
import os
import signal

app = Flask(__name__)

model_dict = pickle.load(open('./model2.p', 'rb'))
model = model_dict['model']

cap = cv2.VideoCapture(0)

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=False, min_detection_confidence=0.3)

labels_dict = {0: 'Hello', 1: 'Super!', 2: 'Three', 3: 'Yo!', 4: 'I\'m #1'}

gesture_counter = {}
sentence = []

def generate_frames():
    global gesture_counter, sentence
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        H, W, _ = frame.shape
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(frame_rgb)
```

```

if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            frame,
            hand_landmarks,
            mp_hands.HAND_CONNECTIONS,
            mp_drawing_styles.get_default_hand_landmarks_style(),
            mp_drawing_styles.get_default_hand_connections_style())

        data_aux = []
        x_ = []
        y_ = []

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

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

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

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

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

        prediction = model.predict([np.asarray(data_aux)])
        predicted_character = labels_dict[int(prediction[0])]

```

```

        # Reset and set the gesture counter
        gesture_counter = {}
        if predicted_character in gesture_counter:
            gesture_counter[predicted_character] += 1
        else:
            gesture_counter[predicted_character] = 1

        # Display sign immediately
        sentence = [predicted_character]

        display_sentence = ' '.join(sentence)

        cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 2, (0, 0, 255), 2,
                    cv2.LINE_AA)
        cv2.putText(frame, display_sentence, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    ret, buffer = cv2.imencode('.jpg', frame)
    frame = buffer.tobytes()

    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

def stream_signs():
    global sentence
    while True:
        yield f"data: {sentence[0]}\n\n"

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

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

@app.route('/shutdown', methods=['POST'])
def shutdown():
    shutdown_server()
    return 'Server shutting down...'

@app.route('/sign_updates')
def sign_updates():
    return Response(stream_with_context(stream_signs()), content_type='text/event-stream')

def shutdown_server():
    pid = os.getpid()
    os.kill(pid, signal.SIGTERM)

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

```


5.1.7 Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Real Time Sign Interpretation System</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f0ead6; /* Sandy beach color */
      color: #333; /* Dark text color */
      margin: 0;
      padding: 0;
    }
    .container {
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff; /* White container background */
      border-radius: 10px;
      box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
    }
    .title {
      text-align: center;
      color: #1565c0; /* Deep blue title color */
      margin-bottom: 20px;
    }
    #animation-container, #video-container {
      text-align: center;
      background-color: #fff; /* White container background */
      padding: 20px;
      border-radius: 8px;
      margin-bottom: 20px;
    }
    #video-container {
      display: none; /* Initially hidden */
    }
    video, img {
      max-width: 100%;
      border-radius: 8px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
    }
  </style>
</head>
```

```

#sign-display {
    font-size: 1.5em;
    color: ■ #1565c0; /* Deep blue sign display color */
    margin-bottom: 20px;
}
#controls {
    text-align: center;
}
.btn {
    padding: 10px 20px;
    font-size: 1em;
    color: □ #fff; /* White button text color */
    background-color: ■ #4caf50; /* Light green button color */
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}
.btn:hover {
    background-color: ■ #388e3c; /* Darker green on hover */
}
</style>
</head>
<body>
<div class="container">
    <h1 class="title">Real Time Sign Interpretation System</h1>
    <div id="animation-container">
        <div>Starting Hand Gesture Recognition...</div>
    </div>
    <div id="video-container">
        
        <div id="sign-display">Detected Sign: None</div>
    </div>
    <div id="controls">
        <button id="start-button" class="btn">Start Camera</button>
        <button id="end-button" class="btn" style="display: none;">End Camera</button>
    </div>
</div>

```

```

<script>
  document.getElementById('start-button').addEventListener('click', function() {
    document.getElementById('animation-container').style.display = 'none';
    document.getElementById('video-container').style.display = 'block';
    this.style.display = 'none';
    document.getElementById('end-button').style.display = 'inline-block';
  });

  document.getElementById('end-button').addEventListener('click', function() {
    fetch('/shutdown', { method: 'POST' })
      .then(response => {
        if (response.ok) {
          document.getElementById('video-container').style.display = 'none';
          document.getElementById('animation-container').style.display = 'block';
          this.style.display = 'none';
          document.getElementById('start-button').style.display = 'inline-block';
        }
      });
  });

  function updateSignDisplay(sign) {
    document.getElementById('sign-display').textContent = 'Detected Sign: ' + sign;
  }

  const eventSource = new EventSource("/sign_updates");
  eventSource.onmessage = function(event) {
    const sign = event.data;
    updateSignDisplay(sign);
  };
</script>
</body>
</html>

```

5.2 Experimental Results

Random Forest Classifier:

Accuracy: 100.00%

Precision: 1.00

Recall: 1.00

F1-score: 1.00

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	58
1	1.00	1.00	1.00	54
10	1.00	1.00	1.00	60
11	1.00	1.00	1.00	59
12	1.00	1.00	1.00	50
13	1.00	1.00	1.00	53
14	1.00	1.00	1.00	48
2	1.00	1.00	1.00	50
3	1.00	1.00	1.00	60
4	1.00	1.00	1.00	54
5	1.00	1.00	1.00	60
6	1.00	1.00	1.00	38
7	1.00	1.00	1.00	43
8	1.00	1.00	1.00	44
9	1.00	1.00	1.00	60
accuracy			1.00	791
macro avg	1.00	1.00	1.00	791
weighted avg	1.00	1.00	1.00	791

CNN Classifier:

Accuracy: 99.12%

Precision: 0.99

Recall: 0.99

F1-score: 0.99

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	58
1	1.00	1.00	1.00	54
2	1.00	1.00	1.00	50
3	1.00	1.00	1.00	60
4	1.00	1.00	1.00	54
5	1.00	1.00	1.00	60
6	0.97	0.84	0.90	38
7	1.00	0.98	0.99	43
8	1.00	1.00	1.00	44
9	1.00	1.00	1.00	60
10	1.00	1.00	1.00	60
11	1.00	1.00	1.00	59
12	0.89	1.00	0.94	50
13	1.00	1.00	1.00	53
14	1.00	1.00	1.00	48
accuracy			0.99	791
macro avg	0.99	0.99	0.99	791
weighted avg	0.99	0.99	0.99	791

Ensemble (Majority Voting):

Accuracy: 99.87%

Precision: 1.00

Recall: 1.00

F1-score: 1.00

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	58
1	1.00	1.00	1.00	54
2	1.00	1.00	1.00	50
3	1.00	1.00	1.00	60
4	1.00	1.00	1.00	54
5	1.00	1.00	1.00	60
6	0.97	1.00	0.99	38
7	1.00	0.98	0.99	43
8	1.00	1.00	1.00	44
9	1.00	1.00	1.00	60
10	1.00	1.00	1.00	60
11	1.00	1.00	1.00	59
12	1.00	1.00	1.00	50
13	1.00	1.00	1.00	53
14	1.00	1.00	1.00	48
accuracy			1.00	791
macro avg	1.00	1.00	1.00	791
weighted avg	1.00	1.00	1.00	791

Ensemble (Averaging Probabilities):

Accuracy: 100.00%

Precision: 1.00











Recall: 1.00

F1-score: 1.00

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	40
1	1.00	1.00	1.00	40
2	1.00	1.00	1.00	40
3	1.00	1.00	1.00	40
4	1.00	1.00	1.00	40
5	1.00	1.00	1.00	40
6	1.00	1.00	1.00	40
7	1.00	1.00	1.00	40
8	1.00	1.00	1.00	40
9	1.00	1.00	1.00	40
accuracy			1.00	400
macro avg	1.00	1.00	1.00	400
weighted avg	1.00	1.00	1.00	400

5.3 Test Cases:

Test Case	Prediction	Output	Status
	No Sign Detected		No
	Hello		Yes
	No Sign Detected		No
	Four		Yes
	Victory		Yes

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The development of a sign language recognition system that ensembles a Random Forest Classifier (RFC) and a Convolutional Neural Network (CNN) addresses several limitations present in existing systems. By leveraging the strengths of both models, this project achieves a balance between accuracy and computational efficiency, making it suitable for deployment on resource-constrained devices such as mobile phones. The system's design reduces the dependency on specialized hardware, enhancing its accessibility for everyday use by a broader population.

The development of a sign language recognition system that ensembles a Random Forest Classifier (RFC) and a Convolutional Neural Network (CNN) addresses several limitations present in existing systems. By leveraging the strengths of both models, this project achieves a balance between accuracy and computational efficiency, making it suitable for deployment on resource-constrained devices such as mobile phones. The system's design reduces the dependency on specialized hardware, enhancing its accessibility for everyday use by a broader population.

Through careful dataset selection and preprocessing, the system demonstrates robustness in recognizing a wide range of signs, including dynamic gestures. The integration of deep learning and traditional machine learning techniques highlights the potential of hybrid approaches in improving the performance of sign language recognition systems.

6.2 Future Scope

The future scope of this project includes several avenues for enhancement and expansion:

6.2.1 Incorporating Advanced Neural Networks:

Transformer Models: Explore the use of transformer-based models, such as Vision Transformers (ViT) and attention mechanisms, to capture complex dependencies in sign language gestures.

Graph Neural Networks (GNNs): Investigate the application of GNNs to model the relationships between different parts of the hand and body for more nuanced gesture recognition.

6.2.2 Expanding the Dataset:

Multilingual Datasets: Collect and incorporate datasets from various sign languages (e.g., British Sign Language, Indian Sign Language) to enhance the system's versatility.

Crowdsourcing Data: Utilize crowdsourcing platforms to gather diverse sign language data, ensuring the system can generalize to different signing styles and demographics.

6.2.3 Real-Time Processing and Deployment:

Edge Computing: Implement the system on edge devices to enable real-time processing with minimal latency.

Optimizing Models: Use model compression techniques such as quantization and pruning to reduce the model size and computational load without significantly compromising accuracy.

6.2.4 Improving User Interface and Experience:

6.2.4.1 Interactive Feedback: Develop interactive feedback mechanisms that help users improve their signing by providing real-time suggestions and corrections.

Augmented Reality (AR) Integration: Incorporate AR technology to overlay sign language translations onto real-world objects, enhancing the learning experience for users.

By exploring these future directions, the project can further enhance the effectiveness, adaptability, and usability of the sign language recognition system, making it a valuable tool for improving communication for deaf and mute individuals worldwide

Chapter 7

References

- [1] Abacha, A. B., & Demner-Fushman, D. (2019). On the summarization of consumer health questions. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2228–2234.
- [2] Agirre, E., Cer, D., Diab, M., & Gonzalez-Agirre, A. (2012). SemEval-2012 task 6: A pilot on semantic textual similarity. In *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), pp. 385–393, Montr´eal, Canada. Association for Computational Linguistics.
- [3] Martin D S 2003 Cognition, Education, and Deafness: Directions for Research and Instruction (Washington: Gallaudet University Press)
- [4] McInnes J M and Treffry J A 1993 Deaf-blind Infants and Children: A Developmental Guid (Toronto : University of Toronto Press)
- [5] S. C. W. Ong and S. Ranganath, —Automatic sign language analysis: A survey and the future beyond lexical meaning, IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 6, pp. 873– 891, Jun. 2005
- [6] L. Ding and A. M. Martinez, —Modelling and recognition of the linguistic components in American sign language, Image Vis. Comput., vol. 27, no. 12, pp. 1826– 1844, Nov. 2009.
- [7] Geza, W.; Ngidi, M.S.C.; Slotow, R.; Mabhaudhi, T. The Dynamics of Youth Employment and Empowerment in Agriculture and Rural Development in South Africa: A Scoping Review. Sustainability 2022,14, 5041. [CrossRef]
- [8] Posel, D.; Oyenubi, A.; Kollamparambil, U. Job loss and mental health during the COVID-19 lockdown: Evidence from South Africa. PLoS ONE 2021,16, e0249352. [CrossRef] [PubMed]
- [9] Chai, S., Xu, Z., Lai, L. L., and Wong, K. P. (2015). “An Overview on Wind Power Forecasting Methods,” in Proceedings of the 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China, July 2015 (IEEE), 765–770. doi:10.1109/ICMLC.2015.7340651

[10] Chen, M.-R., Zeng, G.-Q., Lu, K.-D., and Weng, J. (2019). A Two-Layer Nonlinear Combination Method for Short-Term Wind Speed Prediction Based on ELM, ENN, and LSTM. *IEEE Internet Things J.* 6, 6997–7010. doi:10.1109/JIOT.2019.2913176

recognition from video: A new large-scale dataset and methods comparison. In *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, pp. 1459-1469.

[11] Donahue, J., et al. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625-2634.

[12] Cui, R., Liu, H., & Zhang, C. 2017. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7361-7369.

[13] Bukhari J, Rehman M, Malik SI, Kamboh A, Salman A, “American Sign Language Translation through Sensory Glove; Signspeak”, *International Journal of u-and e-Service, Science and Technology*, 2015, vol. 8, no. 1, pp. 131-142.

[14] Mlakić D, Nikolovski S, Alibašić E, “Designing Automatic Meter Reading System Using Open Source Hardware and Software”, *International Journal of Electrical and Computer Engineering*, vol. 7, no. 6, 2017, pp. 3282-3291

[15] Masood, S., Srivastava, A., Thuwal, H. C., & Ahmad, M. 2018. Real-time sign language gesture (word) recognition from video sequences using CNN and RNN. In *Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA*, 623-632.

[16] Wei, C., Zhou, W., Pu, J., & Li, H. 2019. Deep grammatical multi-classifier for continuous sign language recognition. In *2019 IEEE Fifth International Conference on multimedia big data (BigMM)*, IEEE, pp. 435-442.

[17] Hochreiter, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02), 107-116.

[18] Battina, D. S., & Surya, L. 2021. Innovative study of an AI voice-based smart Device to assist deaf people in understanding and responding to their body language. *SSRN Electronic Journal*, 9, 816-822.

[19] Compton, S. E. 2014. American Sign Language as a heritage language. In *Handbook of heritage, community, and Native American languages in the United States*, Routledge, pp. 272-283.

Appendix

Snapshot of the Result



Real Time Sign Interpretation System



Super!

Detected Sign: Super!

End Camera

Real Time Sign Interpretation System

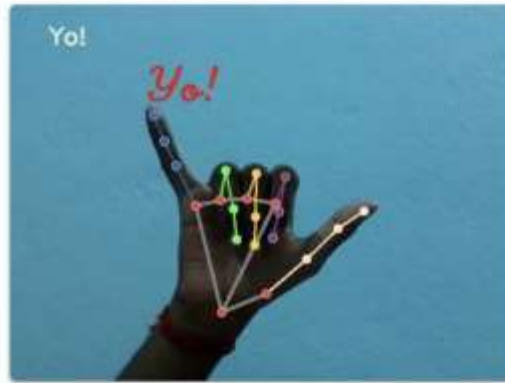


Three

Detected Sign: Three

End Camera

Real Time Sign Interpretation System



Detected Sign: Yo!

End Camera