Sign2Text: Accessible Communication Platform - AI Model for Especially-abled people.

Software Design Specification

Version 1.0
09 September 2024

Pranav Patni
(BTECH/25002/21)
&
Namit Goyal
(BTECH/25025/21)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BIRLA INSTITUTE OF TECHNOLOGY, MESRA
JAIPUR CAMPUS, JAIPUR
SP – 2024**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide a comprehensive software design specification for the **Sign2Text: Accessible Communication Platform**. This platform helps individuals with hearing and speech impairments communicate effectively by translating sign language into text and speech in real time. The document outlines the system's architecture, component design, user interface, database structures, and testing strategies.

1.2 Scope

This SDS covers the detailed design aspects of the system, focusing on:

1. The architectural components that support gesture recognition, text conversion, and speech output.
2. Detailed subsystem design, including gesture recognition (CNN-based), text-to-speech (TTS), and user profile management.
3. API design for internal and external communication.
4. Data storage models and their interactions with the system components.

1.3 Overview

The Sign2Text platform is designed to process real-time video input to detect hand gestures, convert them into corresponding text, and further transform the text into speech. The system relies on deep learning models like Convolutional Neural Networks (CNN) for gesture recognition, alongside Text-to-Speech (TTS) technology for generating voice output.

2. System Architecture

2.1 High-Level Architecture

The system is composed of three main modules:

1. **Gesture Recognition Module**: Captures video input and uses CNN for gesture classification.
2. **Text and Speech Conversion Module**: Converts recognized gestures into text and uses a TTS engine for voice output.
3. **User Interface Module**: Manages interaction with users, displays recognized text, and provides options for speech playback.

These modules interact through RESTful APIs, with the gesture recognition system serving as the core processor of input and the user interface as the point of interaction.

2.2 Subsystem Architecture

1. **Gesture Recognition Subsystem**: Captures video input, preprocesses the frames, and passes them through a CNN model to classify gestures.
2. **Text and Speech Conversion Subsystem**: Converts gestures into text using a pre-defined gesture-to-text mapping model and leverages the TTS engine for spoken output.
3. **User Interface Subsystem**: Provides an intuitive interface for users to interact with the platform. It includes real-time gesture feedback, text output, and TTS control options.

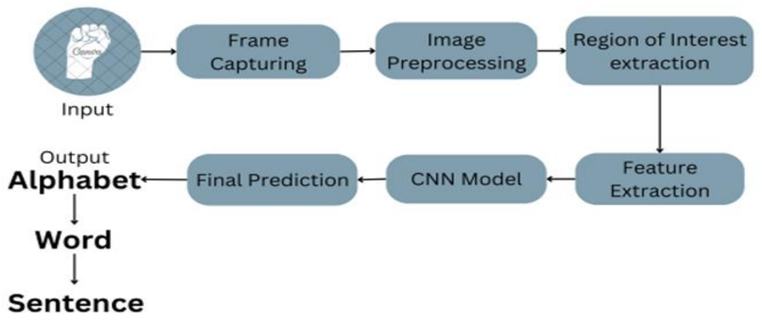

Fig. 1 System Architecture Diagram

3. Detailed Design

3.1 Gesture Recognition Subsystem

The **Gesture Recognition Subsystem** processes input video frames, identifies hand gestures using a trained CNN model, and generates corresponding text. The subsystem design includes:

1. **Frame Capture**: Video frames are captured at 30 FPS from the user's camera.
2. **Preprocessing**: Frames are resized, normalized, and converted to grayscale to reduce processing overhead.
3. **CNN Model**: A pre-trained CNN model classifies gestures into predefined classes corresponding to American Sign Language (ASL).
4. **Feedback Loop**: If the gesture is not recognized, the user can provide feedback to add a custom gesture.

3.2 Text and Speech Conversion Subsystem

Once gestures are recognized, the **Text and Speech Conversion Subsystem** maps the gesture class to the corresponding text and converts it into speech:

1. **Text Conversion**: The recognized gesture is mapped to a specific word or phrase.

2. **TTS Engine**: The text is converted into speech using the Google Cloud Text-to-Speech API or a similar service, with options for language selection and voice customization.

3.3 User Interface Subsystem

The **User Interface Subsystem** is responsible for real-time interaction with users, providing:

1. **Video Feed**: A live video stream from the camera for gesture input.
2. **Text Display**: The recognized text is displayed in real-time.
3. **Speech Playback**: Users can replay the text as speech and choose different voice options.
4. **Settings Management**: Includes controls for language preferences, custom gesture input, and user profile management.

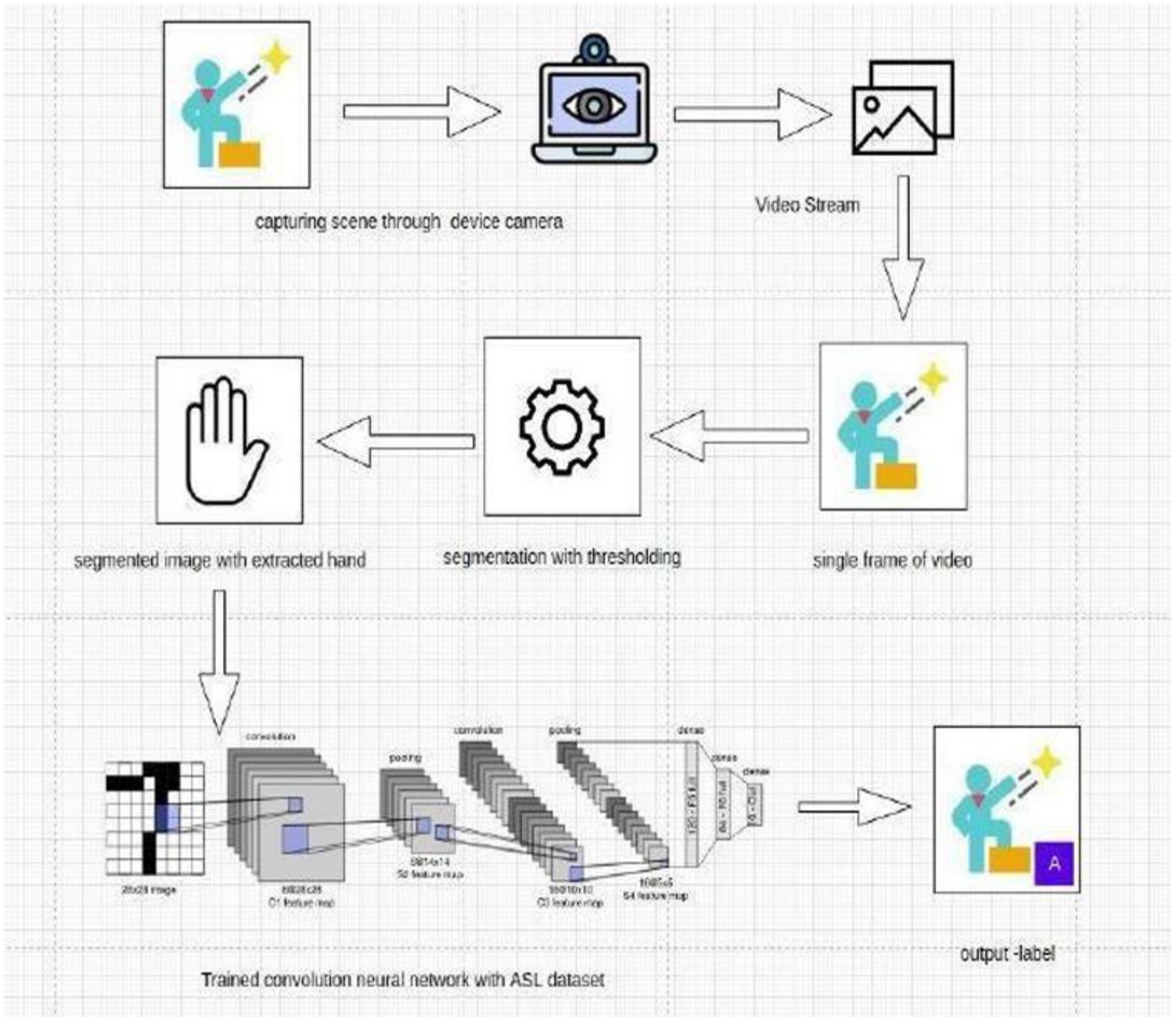
capturing scene through device camera
Video Stream
segmented image with extracted hand
segmentation with thresholding
single frame of video
Trained convolution neural network with ASL dataset
output -label
A

Fig. 2 Flow of the data in the model

4. Database Design

4.1 Data Models

1. **User Profile**: Stores user preferences such as language settings, custom gestures, and text-to-speech options.
2. **Gesture Data**: Stores predefined and custom gestures mapped to corresponding text or phrases.

4.2 Data Storage and Retrieval

The database uses a relational model (e.g., MySQL) to store user data, gesture mappings, and system logs. Each user profile is linked to the stored custom gestures, allowing for flexibility and personalization.

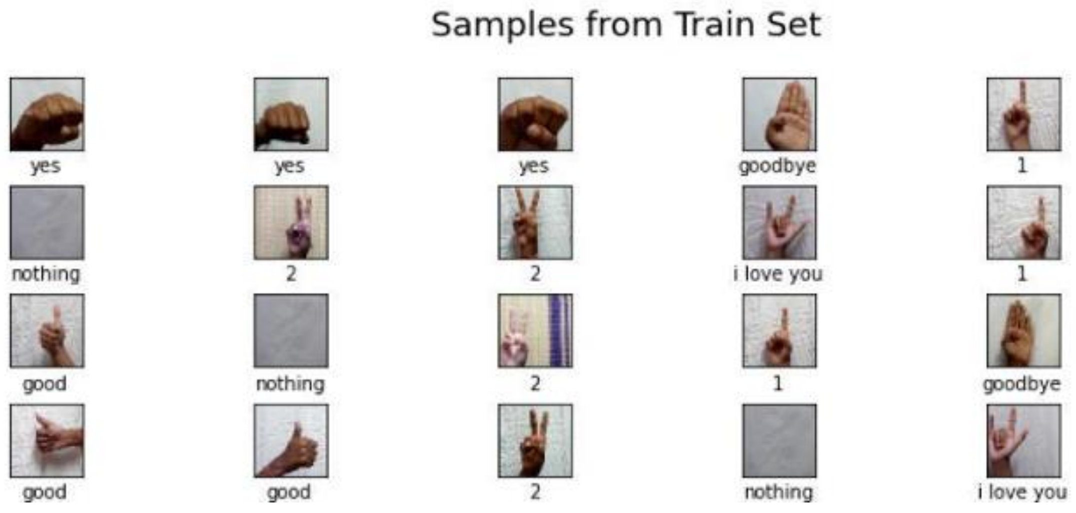

Fig. 3 Samples from Train Set

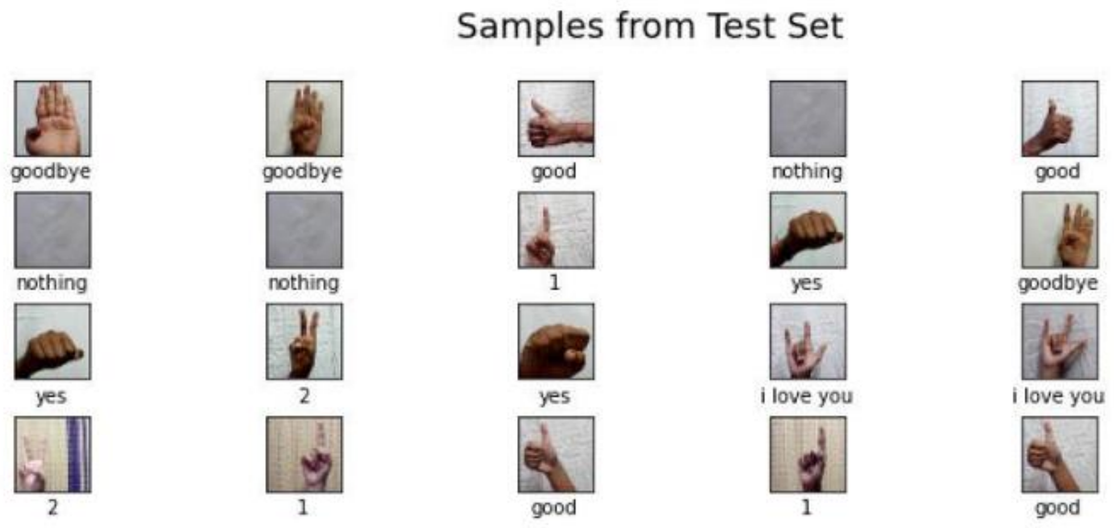

Fig. 4 Samples from Test Set

5. API Design

5.1 Internal APIs

Internal APIs manage the interaction between the gesture recognition, text conversion, and TTS modules. APIs include:

1. **Recognize Gesture**: Processes the video frame and returns recognized text.
2. **Convert Text To Speech**: Converts the recognized text into speech.

5.2 External APIs

External APIs facilitate integration with third-party services like:

1. **Google Cloud TTS API**: For converting text into speech.
2. **TensorFlow / PyTorch**: For running the CNN model to recognize gestures.

6. Component Design

6.1 CNN Gesture Recognition Module

This module processes images to detect and classify gestures. The key components include:

1. **Image Preprocessing**: Normalizes and resizes the captured frames.
2. **CNN Model**: Classifies the gesture into one of the pre-defined ASL categories.

6.2 Text-to-Speech Module

The module uses external APIs to convert recognized text into speech. Users can choose different languages and voice options for the speech output.

6.3 User Management Module

Handles user registration, login, and profile management. It also allows users to save custom gestures and preferred TTS settings.

7. Data Flow Diagrams

Data Flow Diagrams (DFDs) illustrate the interaction between components:

1. **Level 0**: High-level data flow from user input (gesture) to system output (speech).
2. **Level 1**: Detailed flows, showing preprocessing, CNN classification, text mapping, and TTS conversion.

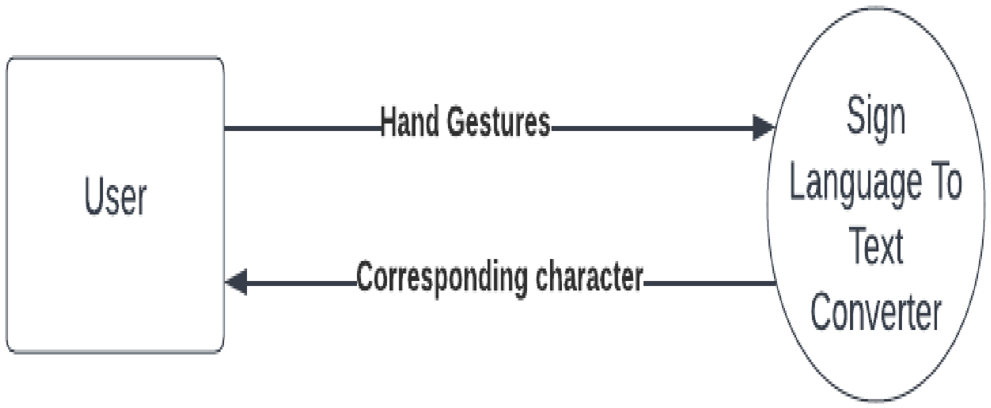

Fig. 5 DFD Level 0

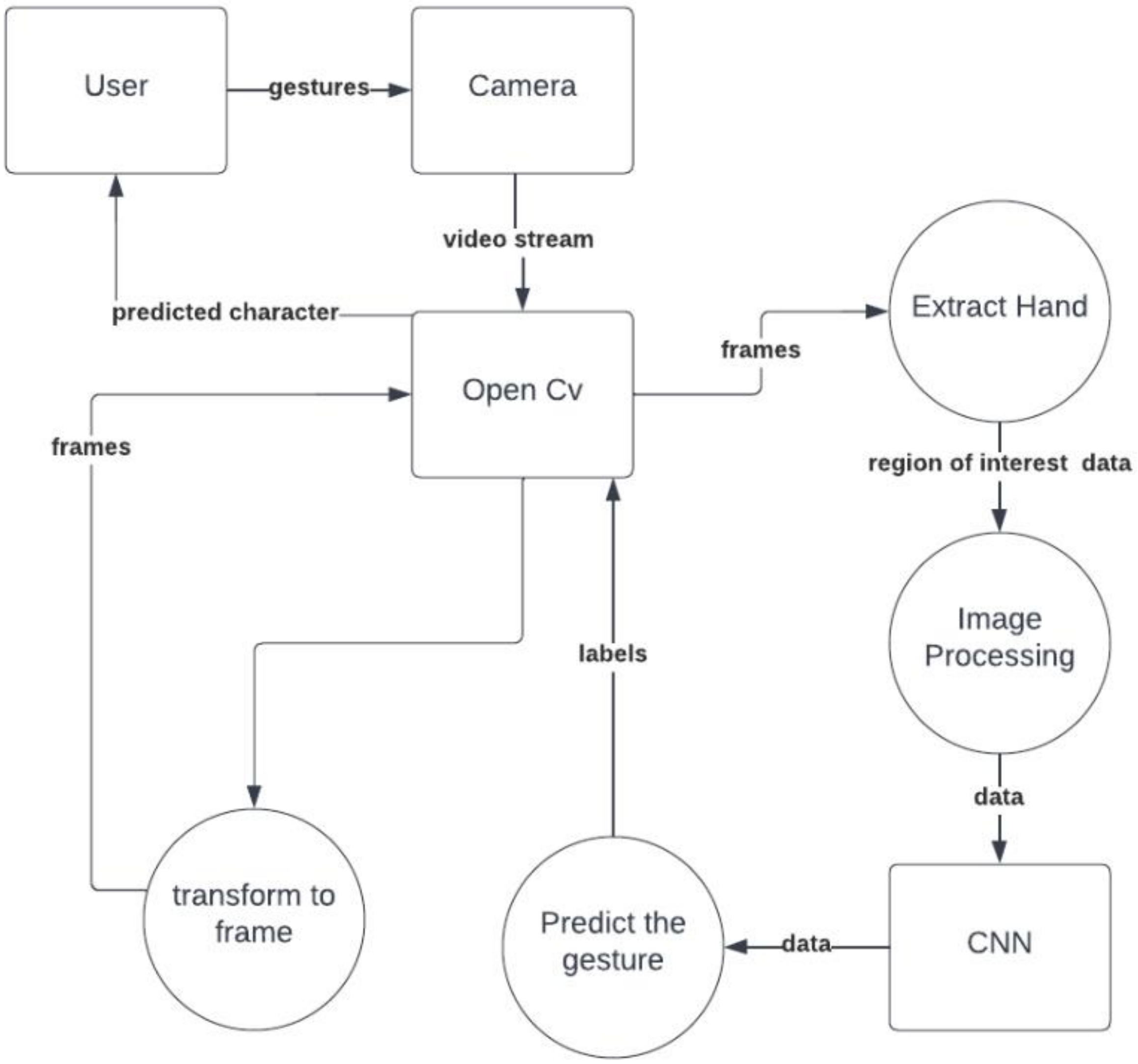

Fig. 6 DFD Level 1

8. User Interface Design

8.1 User Interaction Flow

The interaction flow includes:

1. User performs a gesture.
2. The system captures and processes the gesture in real-time.
3. Recognized text is displayed on the UI.
4. User has the option to hear the spoken version of the text.

8.2 Screen Design

Screens include:

1. **Gesture Recognition Screen**: Live video feed and real-time text display.
2. **Settings Screen**: Options for TTS preferences, language selection, and custom gesture input.

9. Testing and Validation Strategy

9.1 Unit Testing

Tests each subsystem independently to verify functionality:

1. Gesture recognition accuracy.
2. Text-to-speech conversion quality.

9.2 Integration Testing

Ensures that different subsystems work together seamlessly, focusing on interaction between gesture recognition, text conversion, and TTS.

9.3 Performance Testing

Measures the system's response time and ensures gestures are processed and converted to text and speech with minimal delay.

9.4 User Acceptance Testing

Engages users with hearing and speech impairments to test the usability, effectiveness, and accuracy of the platform in real-world scenarios.

Appendix

A. Glossary

1. **AI (Artificial Intelligence):** The simulation of human intelligence in machines that are designed to think and act like humans. In the context of this project, AI is used to recognize hand gestures through deep learning models.
2. **CNN (Convolutional Neural Network):** A type of deep learning model commonly used in image and video recognition. This model will classify hand gestures for sign language recognition.
3. **ISL (Indian Sign Language):** The primary sign language used in India, which this system will initially support for gesture recognition.
4. **TTS (Text-to-Speech):** A technology that converts written text into spoken words using computer-generated voices.
5. **API (Application Programming Interface):** A set of functions and protocols that allow different software components to communicate with each other.
6. **UI (User Interface):** The visual part of the software that the user interacts with, including input forms, buttons, and displays.
7. **Real-time Processing:** The ability of the system to process inputs (like video frames) and provide outputs (like text or speech) without noticeable delay.
8. **Custom Gestures:** User-defined hand movements that the system can learn and incorporate into the gesture recognition process.
9. **Recognition Accuracy:** The percentage of correctly identified gestures by the system, which is crucial for real-time communication.9.2 Integration Testing

B. Acronyms

1. **AI**: Artificial Intelligence
2. **CNN**: Convolutional Neural Network
3. **ISL**: Indian Sign Language
4. **TTS**: Text-to-Speech
5. **API**: Application Programming Interface
6. **UI**: User Interface

D. Tools and Technologies

1. **TensorFlow/PyTorch:** Deep learning frameworks used to develop and train CNN models for gesture recognition.
2. **Google Cloud Text-to-Speech API:** Converts the recognized text into spoken words using advanced speech synthesis technologies.
3. **Python:** The primary programming language used for the development of the platform.
4. **OpenCV:** A computer vision library that will be used to process video frames and capture gestures from the camera.

5. **Flask/Django:** Web frameworks used for building the web-based platform, handling API requests, and serving the user interface.
6. **MySQL/PostgreSQL:** Relational databases used for storing user data, gestures, and custom preferences.

G. Future Enhancements

1. **Multilingual Support:** Expanding the platform to support American Sign Language (ASL), British Sign Language (BSL), and other sign languages.
2. **Mobile Application:** Development of mobile applications for Android and iOS, allowing users to access the platform from smartphones and tablets.
3. **Offline Mode:** Implementing an offline version of the platform that can function without an internet connection for gesture recognition and TTS conversion.
4. **Gesture Customization AI:** Incorporating machine learning algorithms that allow the system to self-learn and adapt to custom gestures without manual input.

H. Related Projects and Research

1. *Real-Time Sign Language Recognition Using Machine Learning* (2020): A project that explored the use of machine learning for recognizing gestures from live video feeds, forming the foundation for the gesture recognition model used in Sign2Text.
2. *AI for Accessibility*: A Microsoft initiative aimed at developing AI tools to aid especially-abled individuals, similar in mission to Sign2Text.

This appendix provides essential supporting information, references, and additional considerations to supplement the Software Design Specification (SDS) for **Sign2Text**.

References

1. American Sign Language (ASL) Dictionary: A comprehensive source for the visual representation of ASL gestures and their corresponding meanings.
2. Machine Learning Techniques for Gesture Recognition: Research papers and articles focusing on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for real-time gesture recognition.
3. Dhruv, A., & Bharti, S. (2021). Real-Time Sign Language Converter for Mute and Deaf People. 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV), 1-6. https://doi.org/10.1109/aimv53313.2021.9670928.
4. Vu, H., Hoang, T., Tran, C., & Pham, C. (2023). Sign Language Recognition with Self-Learning Fusion Model. IEEE Sensors Journal, 23, 27828-27840. https://doi.org/10.1109/JSEN.2023.3314728.
5. K, R., A, P., S, P., Sasikala, S., & Arunkumar, S. (2022). Two-Way Sign Language Conversion for Assisting Deaf-Mutes Using Neural Network. 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), 1, 642-646. https://doi.org/10.1109/ICACCS54159.2022.9785057.
6. Dhamane, S., Ainapure, A., & Dhage, S. (2023). Breaking Silence, Embracing Inclusivity: The Power of AI in Communication. 2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), 231-236. https://doi.org/10.1109/ICCCMLA58983.2023.10346765.