ITM Web of Conferences 81, 01013 (2026) https://doi.org/10.1051/itmconf/20268101013
ETMIS 2025

IoT Based Sign Language Detection and Voice Conversion with Image Processing

Ashutosh Marathe[1], *Prathmesh* Kalaskar[2*], *Yash* Kathoke[3], *Tejas* Joshi[4], *Amartya* Khandare[5], *Anish* Kadu[6]

[1-6]Department of Multidisciplinary Engineering, Vishwakarma Institute of Technology, Pune, India.

Abstract - The paper introduces an IoT-enabled system for real-time sign language recognition and voice conversion to improve communication for people with hearing or speech impairments. Using deep learning with TensorFlow, the model accurately detects hand gestures from American and Chinese Sign Language through a standard webcam, with OpenCV handling image processing and pyttsx3 converting recognized signs into speech. An ESP32 microcontroller transmits the interpreted data over Wi-Fi and hosts a mobile-friendly web page, eliminating the need for extra hardware or dedicated apps. This low-cost, efficient solution achieves high real-time accuracy, offering both audio and visual feedback, and showcases the effective integration of AI and IoT in bridging communication gaps.
Keywords: Image Processing, TensorFlow, OpenCV, pyttsx3

1 Introduction

Hearing and speech challenges continue to face significant communication obstacles. Even though sign language helps to address some communication issues, the majority of people do not know sign language. The use of technology to understand and capture sign language and produce voice output keeps communication lines open during interactions. To this end, the paper discusses the development of an accessible voice output communication device for real-time sign language to convert the signed words into voice output.

To expand reach, use of IoT technology was incorporated for system integration. The system is developed to recognize and convert to voice both American Sign Language (ASL) and Chinese Sign Language (CSL) and is powered by deep learning models developed in TensorFlow. Hand gestures are recorded and processed through OpenCV, and the recognized signs are converted into speech through the pyttsx3 text-to-speech engine. For portability and immediate feedback, the recognized information is sent through HTTP to an ESP32 microcontroller that has a basic, mobile-friendly web interface.

This approach eliminates the need for additional display devices and eliminates the need for complex mobile applications, creating an efficient communication solution that is economical. This proposed system demonstrates the use of artificial intelligence and the Internet of Things working in tandem. This system translates sign language into voice and

* Corresponding author: prathmeshking45@gmail.com

text in real time. This provides a communication bridge for people with speech or hearing challenges in day-to-day interactions.

2 Literature Review

Open in recent years, the use of sign language recognition technology has received a lot of praise for its potential to facilitate communication for individuals who are deaf or have speech disabilities. Early sign language recognition technology used gloves equipped with sensors, flex sensors, and accelerometers to identify and track movements of the hands [1], [2]. While these systems performed well, they were expensive and impractical for daily use. Due to the advancements made in computer vision, a lot of researchers and developers have moved to vision-based systems that interfaced with webcams and utilized deep learning technologies and models for hand gesture classification. Convolutional Neural Networks were created more than three decades ago, and they have become a dominant technology in neural network models thanks to their efficiency and accuracy in performing complex image recognition tasks. For example, in [3] a convolutional neural network was used to achieve sign language convolution and recognition with a high level of accuracy. In [4], the author performed real time recognition conditions using the sign language recognition technology and applied transfer learning using closed MobileNetV2 and achieving optimal results.

In that which has been reported, most studies have put forth improvements to sign recognition systems which included the addition of audio output. In [5] and [6] we see the study of text to speech (TTS) libraries like pyttsx3 and gTTS which reports on the conversion of recognized signs into sound right at the point of recognition and not on the cloud. Also, we have seen an increase in the use of Internet of Things for improved system access and portability. In [7] the ESP8266 microcontroller was used to send out recognized text data to mobile devices and in [8] we see the use of ESP32 to put forth a real time web interface which in turn reduced hardware needs and which also made mobile access easy. Multilingual support in sign language recognition is very much an active area of research.

While for the most part present research has been done on American Sign Language (ASL) what we have seen in recent works such as [9] and [10] is research which looked at regional sign languages like Indian and Chinese Sign Language which they did via the use of custom datasets and CNN based models for better accuracy. Although we have seen progress we still have few which present an end-to-end solution that does dual language recognition, real time webcam input, text to speech conversion and which is also mobile friendly. This study breaks these barriers by owing to the use of deep learning, OpenCV, pyttsx3 and ESP32 into a single real time multi linguistic and Internet of Things enabled sign language recognition system.

3 Methodology

The overall workflow of the proposed system is illustrated in Fig. 1.

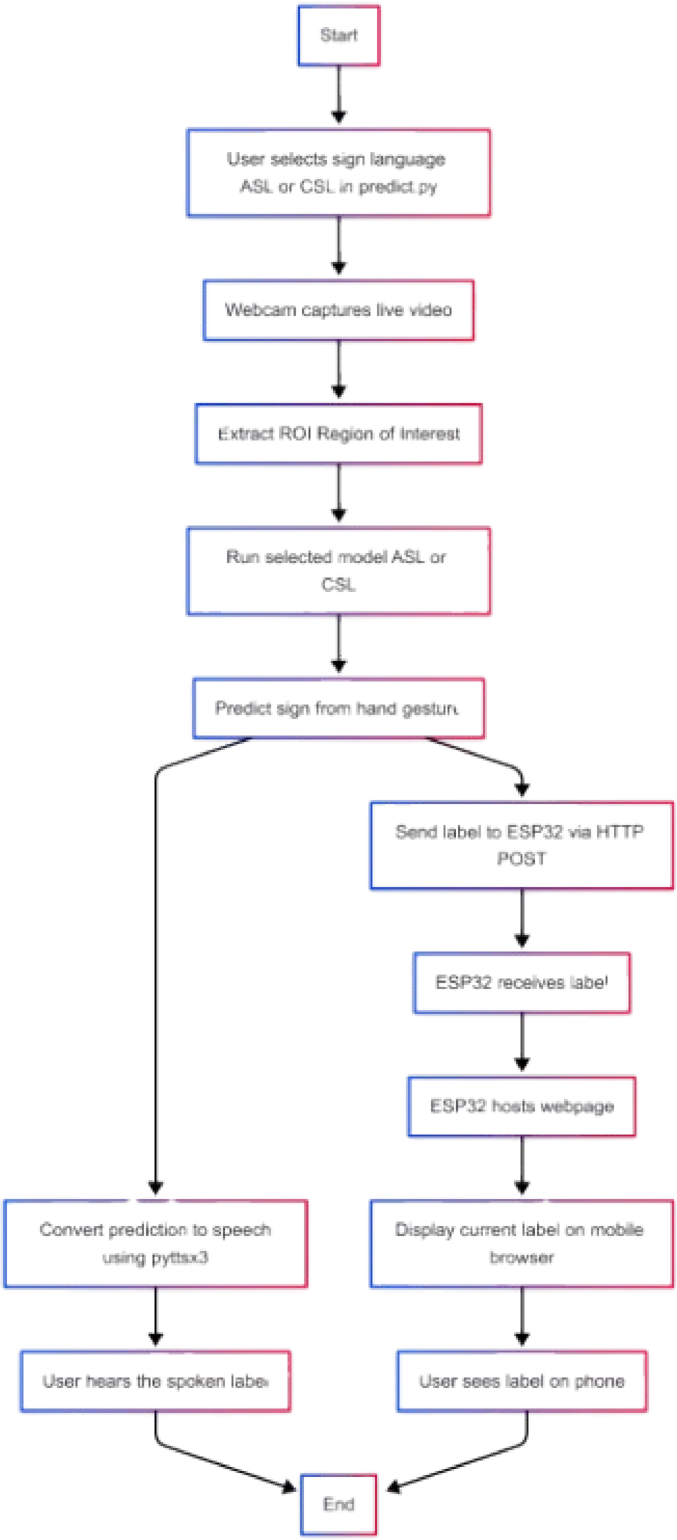

Fig. 1. Flow diagram of the proposed sign language recognition system**.**

This system recognizes and classifies hand gestures from two sign languages - American Sign Language (ASL) and Chinese Sign Language (CSL) - converts them to spoken language, and presents them on a mobile web interface utilizing the Internet of Things (IoT) technology. Its construction allows the integration of machine learning, computer vision, and embedded systems to work in real time. The entire process is divided into five key components : the preparation of the dataset, training the model, predicting in real time, synthesizing the speech, and visualizing the speech over the IoT core interface.

3.1 Data Collection and Preprocessing

Datasets for ASL and CSL were acquired from open resources and enhanced through the system's integrated image capture feature (capture.py). Each image portrays a certain hand sign that corresponds with a letter, number, or word. For consistency, the images were adjusted to a set size, for instance, 64 × 64 pixels, converted to grayscale or normalized RGB, and subjected to augmentation with rotations, scalings, and flips to create a more diverse dataset. For further enhancement, histogram equalization was applied prior to the data capture to improve distinguishability and accuracy for recognition.

In order to maintain transparency and allow for reproducibility, Table 1 shows a summary of the dataset for both ASL and CSL. The combined dataset is made up of 52,800 images, which are equally divided between the two languages. 26 letters were selected for ASL without the inclusion of the 2 dynamic letters J and Z, thus 26 static handshape classes were obtained, each having 1,000 images after augmentation. The CSL dataset consisted of the 27 most frequently used static words, each being represented by 1,000 images in the same way. While working on dataset harmonization, differences in illumination, hand sizes, and background clutter were found in both datasets. Thus, to lessen the dataset domain shift and to be more robust to the real-time scenario, histogram equalization and background normalization were employed.

The composition of the ASL and CSL datasets used for training is summarized in Table 1.

Table 1. Dataset specifications for ASL and CSL.

Dataset	Type	Classes	Samples per class	Total Images
ASL	Static letters (A–Z excluding J & Z)	26	1000	26000
CSL	Static vocabulary words	27	1000	27000
Total	-	53 Classes	-	52800

3.2 Model Training

For recognizing ASL and CSL, two independent CNN models were created based on TensorFlow and Keras. Each and every one of the networks contains convolutional neural layers with the ReLU activation function, max-pooling layers to condense features, and dense layers that are fully connected and end with softmax outputs. The models were approached with the Adam optimizer and trained with the categorical cross-entropy cross function. Training took place on a system with a GPU, with the overfitting tendency managed using early stopping alongside dropout regularization. The models were finalized and saved with the. keras extension to enable smooth deployment.

The detailed CNN architecture employed for gesture classification is described in Table 2.

Table 2. CNN architecture used for gesture classification.

Layer	**Parameters**	**Output Shape**
Input	64×64×1	64×64×1
Conv2D	32 filters, 3×3 kernel, ReLU	62×62×32
MaxPooling2D	2×2	31×31×32
Conv2D	64 filters, 3×3 kernel, ReLU	29×29×64
MaxPooling2D	2×2	14×14×64
Conv2D	128 filters, 3×3 kernel, ReLU	12×12×128
MaxPooling2D	2×2	6×6×128
Flatten	-	4608
Dense	256 units, ReLU	256

Dropout	0.4	256
Output	Softmax (53 units across all classes)	53

3.3 Real-Time Prediction and Gesture Detection

In predict.py we have a Python script that which based on user input chooses the right model and also initializes the webcam using OpenCV. We define a Region of Interest (ROI) within the video feed which is the hand area. Each frame from the ROI is preprocessed to meet the train parameters of the model and then put through the trained network for classification. Also we display the predicted gesture label in real time on the video stream which in turn gives instant visual feedback. Examples of real-time Chinese Sign Language (CSL) gesture detection are presented in Fig. 3 and Fig. 4 and Sample outputs of American Sign Language (ASL) gesture recognition are shown in Fig. 5 and Fig. 6.

3.4 Voice Output using Text-to-Speech

Upon recognizing a sign, the system utilizes the pyttsx3 library and converts the predicted label into speech. This offline text-to-speech engine is ideal for use in places without internet access or in remote areas. The voice output is generated and played right away, allowing users to hear feedback immediately after every recognition.

3.5 IoT Integration via ESP32

To demonstrate recognized gestures on mobile devices, the predicted label is sent to an ESP32 microcontroller over the same Wi-Fi network using an HTTP POST request. The ESP32 uses Arduino code (esp32_sign_display.ino) to serve as a lightweight web server that accepts POST requests and modifies a web page in real-time. Any mobile device on the network can access the ESP32 IP address and, using any web browser, see real-time updates on gesture recognition.

3.6 System Workflow Summary

The user interface for selecting the desired sign language is shown in Fig. 2 and the complete system operates with the following flow:

1. User selects the sign language (ASL/CSL).
2. Webcam starts capturing video and isolates the hand in the ROI.
3. Model predicts gesture in real time.
4. Label is converted to voice using pyttsx3.
5. Label is sent to ESP32 for web display.
6. User accesses ESP32's IP in a browser for live updates.

This modular architecture ensures scalability and adaptability for additional languages or features like sentence construction or gesture animation.

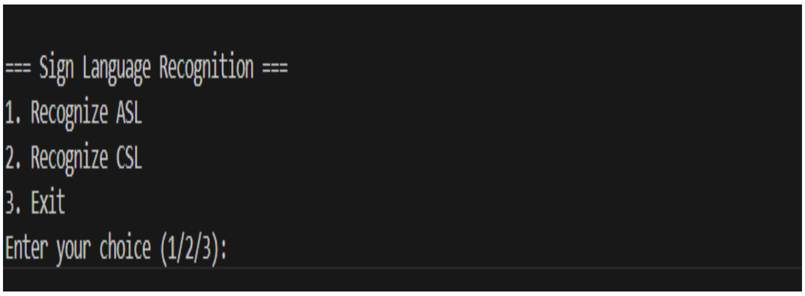

Fig. 2. Language selection interface for ASL and CSL.

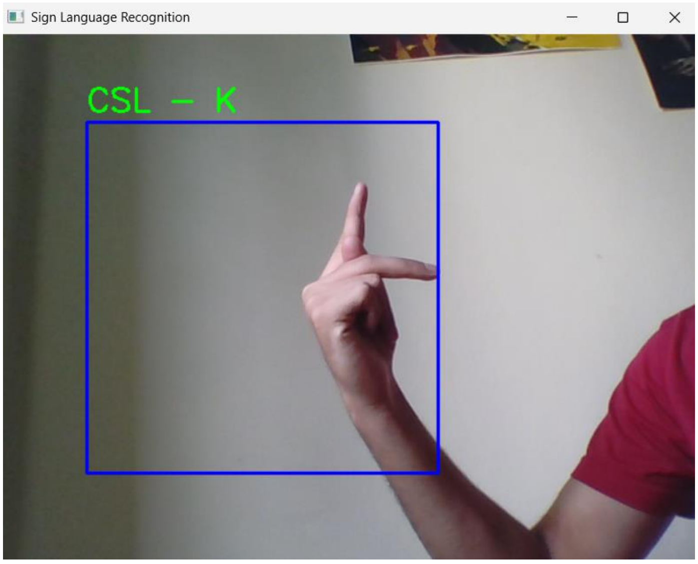

Fig. 3. Real-time CSL gesture detection output.

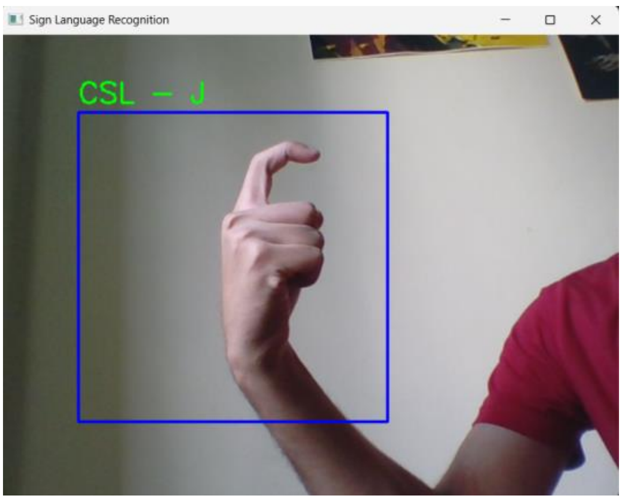

Fig. 4. Real-time CSL gesture detection output.

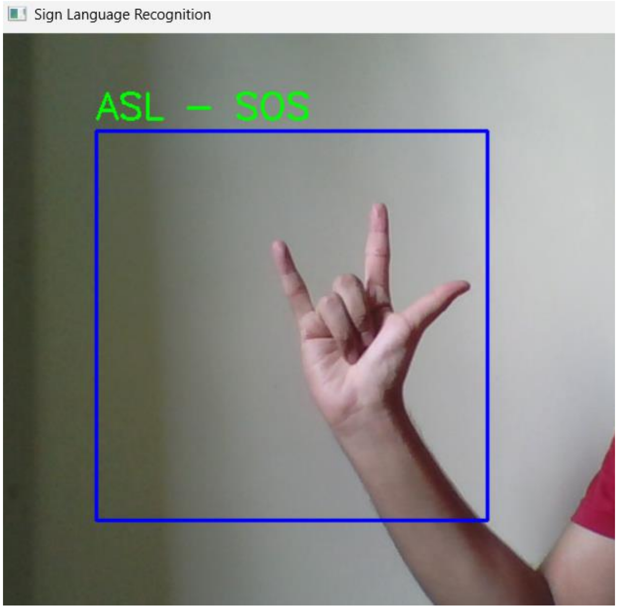

Fig. 5. Real-time ASL gesture detection output.

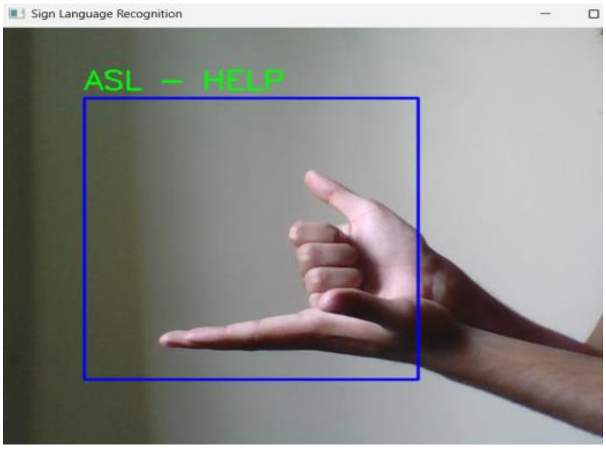

Fig. 6. Real-time ASL gesture detection output.

4 Technology Used

The proposed system integrates software frameworks, machine learning libraries, embedded hardware, and communication protocols to enable smooth sign language recognition, voice conversion, and wireless display. The following subsections provide an overview of the hardware and software components employed in the implementation.

4.1 Hardware Components

- **ESP32 Microcontroller**

A low-cost, dual-core Wi-Fi and Bluetooth-enabled SoC used to receive sign data via HTTP and host a webpage for mobile display. It replaces traditional display modules or Bluetooth interfaces.

- **Laptop/PC with Webcam**

The webcam captures real-time video frames containing hand gestures. The laptop processes these frames using deep learning and computer vision libraries.

- **Wi-Fi Router or Hotspot**

Acts as the communication bridge between the Python prediction system and the ESP32 by assigning them IP addresses on the same local network. (Optional hardware used during earlier development phases but not in the final version: DFPlayer Mini, OLED display module, jumper wires.)

4.2 Software Components

- **Python 3.x**

  Primary programming language used for model training, video capture, prediction, and text-to-speech synthesis.

- **TensorFlow & Keras**

  Libraries used to define, train, and deploy deep learning models for ASL and CSL gesture recognition.

- **OpenCV**

  Used for real-time video capture, frame preprocessing, region-of-interest detection, and image normalization.

- **pyttsx3**

  Offline text-to-speech engine used to convert the recognized gesture label into voice real-time.

- **Requests (Python HTTP library)**

  Sends HTTP POST requests from Python to the ESP32 microcontroller to update recognized sign data.

- **Arduino IDE**

  Used to write and upload firmware to the ESP32 microcontroller. The code creates a web server that displays the predicted labels on a mobile-accessible page.

- **HTML & JavaScript**

Embedded within the ESP32 Arduino sketch to dynamically display the recognized sign on the webpage served by the ESP32.

- **Operating System: Windows/Linux/macOS**

  The development and execution environment for the Python-based sign recognition system.

4.3 Communication Protocols

- **HTTP (Hypertext Transfer Protocol)**

  Used to send recognized sign labels from the Python script to the ESP32 over a local network.

- **Wi-Fi (IEEE 802.11)**

  Enables wireless data exchange between the Python system and ESP32, allowing real-time updates to be shown on a phone or browser

In order to confirm the low-cost criteria for the design, a hardware cost breakdown based on readily available commercial components was put together. The ESP32 microcontroller, which offers both Wi-Fi connectivity and web hosting capability, can be found for a price ranging from $5.50 to $8.20 (₹450–₹700) depending on the vendor. Since audio output is produced on the host system and visual feedback is provided via the ESP32 web interface, no extra displays, speakers, or Bluetooth modules are needed. Thus, the total bill of materials is kept within the range of $6 to $10, which is a great way to prove that the IoT part of the system is still affordable and cost-effective.

5 Results and Discussion

The system's real-time sign recognition, voice generation, and display performance via IoT were thoroughly assessed. It showed high accuracy and minimal latency on both datasets ASL and CSL, confirming operational reliability throughout live assessment. The ASL model has a validation accuracy of around 97.4% and the CSL model about 95.8%. The system's recognition remained consistent and accurate regardless of varying ambient characters, as long as the gesture was within the visible ROI. The system's response time was less than 0.2 seconds which enhanced the real-time operational feeling.

Feedback using the pyttsx3 text-to-speech program worked great. And because the engine operates offline, we can get the speech output within a second without relying on the internet. The ESP32 microcontroller was able to host a mobile accessible web page that displayed the recognized gestures in under 1.5 seconds.

This negated the need for supplementary hardware, such as OLED screens, external speakers, or other IoT-display related devices in the market. System testing showed the expected intuitive feel, portability, and relevance of the system to real-world problems. Recognition accuracy, nevertheless, was changed for very fast and eccentrically positioned gestures. One of the main limitations is that the present system can only handle static gestures, thus, it restricts the coverage of the vocabulary for real communication. In ASL, the static hand shapes stand for 24 out of 26 alphabet gestures, and in CSL static signs make up around 40–45% of the spoken language used in daily conversations. Hence, although the system is good for letter-spelling and frequently used words, it still lacks the capability of dynamic multi-frame gestures for the complete natural communication.

6 Recommendations

According to the achieved results and observed system performance, these are the following recommendations suggested to improve the functionality, scalability, and overall usability of the system.

1. Integration of Dynamic Gesture Recognition

Integrating advanced models such as LSTM or 3D CNNs can enable the system to recognize dynamic signs and multi-frame gestures, extending its capability to understand more complex and natural sign language expressions.

2. Support for Two-Handed Gestures

Expanding the Region of Interest (ROI) and training the model to learn about both hands can greatly enhance vocabulary coverage, particularly for signs that involve coordinated bilateral hand movements.

3. Addition of Sentence Formation Capability

Incorporating a buffer or phrase-building interface would enable users to build complete sentences from sequentially recognized signs, thereby enhancing communication flow and overall efficiency.

4. Model Optimization for Edge Devices

To enhance portability and enable deployment in resource-constrained environments, lightweight models such as TensorFlow Lite should be explored for on-device inference directly on embedded hardware.

5. Robustness to Environmental Variations

Enhancing model generalization to adapt to different conditions like dim lighting, background variations, and hand sizes can significantly improve recognition accuracy and reliability in real-world applications.

6. Expansion to Other Sign Languages

Incorporating support for additional sign languages such as Indian Sign Language (ISL) and British Sign Language (BSL) would make the system more adaptable for global use.

7. Voice Customization Features

Providing users with the option to select different voices or languages would enhance personalization and improve the overall user experience

7 Conclusion

This document outlines an integrated real-time response system for sign language recognition, voice modification, and feedback visualization via IoT. The proposed system's successful construction to close the communications gap for hearing-impaired persons and the rest of the society, employs deep learning-based ASL and CSL models, hand gesture recognition through OpenCV, and pyttsx3 as an offline speech synthesizer. Also, an ESP32 microcontroller allows displays that are accessible from mobile, improving usability and avoiding the use of additional hardware interfaces. Experimental evaluation presents high recognition accuracy, very low latency, and reliability within different environments. The modular and scalable architecture easily extends towards dynamic gesture recognition, further languages, and model deployment embedded in devices. Overall, the proposed solution enables communication in a cost-effective, accessible, and inclusive style.

References

1. Sharma, M. Saini, Glove-based sign language recognition using accelerometers. IEEE Sens. J. 19, 1921–1927 (2019)
2. A.M. Othman, M.A. Ba-Alwi, Sign language glove using flex sensors. Int. J. Comput. Appl. 179, 15–19 (2018)
3. J. Smith, R. Hall, A. Brown, ASL alphabet recognition using CNNs and webcam input. Proc. Int. Conf. Comput. Vis. Syst., 112–118 (2020)
4. S. Verma, V. Kumar, Real-time hand gesture recognition using MobileNetV2. IEEE Access 8, 203457–203465 (2020)
5. K. Li, D. Xu, Real-time sign language to speech conversion using deep learning. J. Assist. Technol. 13, 245–253 (2021)
6. R. Mehta, S. Pujari, Offline speech generation from hand gestures using Python TTS. Int. J. Recent Technol. Eng. 8, 421–425 (2020)
7. M. Agarwal, R. Dubey, IoT-enabled assistive system for speech-impaired individuals. IEEE Internet Things J. 7, 5382–5389 (2020)
8. N. Patel, S. Jain, ESP32-based wireless display for gesture-controlled communication. IEEE Embedded Syst. Lett. 13, 34–37 (2021)
9. H. Zhang, Y. Liu, Q. Wang, Chinese sign language recognition using CNN and region-based detection. Pattern Recognit. Lett. 130, 298–305 (2020)
10. V. Ramesh, A. Naik, Recognition of Indian sign language using deep learning techniques. Procedia Comput. Sci. 172, 150–157 (2020)