iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78347

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

HANDIFY: A Real-Time Sign Language to Text Conversion for Empowering Differently-Abled Communication

Ms. Anjali Chauhan[1], Saumya[2], Vansh Sihania[3], Vidit Sharma[4], Sumit Chauhan[5]
Department of Computer Science and Engineering, KIET Group of Institutions, Ghaziabad, India

Abstract: ***Communication plays a vital role in everyone's life, but people who rely on sign language often struggle to interact with those who don't understand it. To address this gap, this paper introduces Handify, a real-time system designed to make communication easier between deaf individuals and hearing people. Handify converts sign language gestures into readable text, allowing smoother and more natural interaction. The system uses computer vision techniques with OpenCV and applies a Random Forest machine learning model to accurately classify hand gestures. Using a custom dataset created from American Sign Language (ASL), the model achieved an accuracy of 91.7%. This paper also explains the design of the system, the challenges faced during development, the experimental results, and how the tool can improve accessibility for people with hearing or speech disabilities. Overall, Handify contributes to inclusive technology by helping create seamless, barrier-free communication between signers and non-signers.***

Keywords: Sign Language Recognition, Computer Vision, OpenCV, Machine Learning, Random Forest, Accessibility Technology, Human-Computer Interaction

I. INTRODUCTION

Learning to communicate—whether formally or informally—is essential for social interactions, education, and career growth. According to the World Health Organization, around 466 million people worldwide live with hearing disabilities. Many of them depend on sign language as their primary way of communicating. However, because sign language is not understood by the majority of the population, it often creates a communication barrier between deaf or hard-of-hearing individuals and others.

Sign language interpreters can help bridge this gap, but they are often unavailable or too costly. This is where technology offers promising alternatives. Advances in computer vision and machine learning have made it possible to build systems that can recognize and translate sign language gestures into natural language. The Handify project aims to use these technologies to create a real-time system that converts sign language gestures into readable text. By translating signs into text instantly, Handify hopes to make communication smoother and more accessible for both sign language users and non-signers. This paper discusses the system's design, implementation, evaluation, and its potential to improve accessibility for people with diverse abilities.

II. LITERATURE REVIEW

Research on sign language recognition has expanded rapidly over the past two decades, driven by advancements in both computer vision and machine learning. Early attempts primarily relied on sensor-based systems, where users wore specialized gloves or motion-capture devices to track hand movements. Although these approaches produced reliable results, they were costly, uncomfortable, and impractical for everyday use, especially outside controlled environments [6].

With improvements in camera technology and visual processing, vision-based methods eventually became the preferred choice. These systems required only a standard camera—such as one found in laptops or smartphones—making them far more accessible. Vision-based models analyze hand shape, motion, and spatial information, reducing the need for specialized hardware.

A major early breakthrough in vision-based recognition was the system developed by Starner et al., who demonstrated real-time American Sign Language (ASL) recognition using Hidden Markov Models (HMMs). Their system achieved about 92% accuracy for 40 signs, but only under controlled lighting and background conditions [7]. Building on these foundations, later studies explored deep learning to further improve accuracy and handle continuous signing. For example, Camgoz et al. introduced neural network architectures capable of end-to-end sign language translation, although these models required substantial computational resources and large annotated datasets [8].

Further developments showed that combining spatial and temporal cues leads to more robust recognition. Researchers such as Ong and Ranganath emphasized the value of integrating both types of information to enhance performance [9]. Building on this idea, Koller et al. introduced hybrid deep-learning models using CNNs and RNNs to recognize large vocabularies of signs, demonstrating high accuracy but at the cost of requiring GPU-based training and extensive datasets [10]. Commercial systems have also emerged, including tools like SignAll, which offer real-time sign-to-text translation. However, they often depend on multi-camera setups or specialized sensors, making them expensive and difficult to deploy in low-resource settings [11].

Author / Study	Method Used	Dataset	Accuracy (%)	Limitations
Starner et al. [7]	Hidden Markov Models (HMM)	Custom ASL Dataset	92	Requires uniform lighting
Camgoz et al. [8]	Neural Networks (Deep Learning)	RWTH-PHOENIX Dataset	94	High computational cost
Ong and Ranganath [9]	Feature-based Vision Techniques	ASL Alphabet Set	88	Limited vocabulary
Koller et al. [10]	CNN + RNN Hybrid Model	Large Vocabulary Dataset	95	Requires GPU and long training time
SignAll System [11]	Multi-Camera Vision System	Proprietary Dataset	90	Expensive setup, not portable
Proposed Handify System	Random Forest + MediaPipe Landmarks	Custom ASL Alphabet Dataset	91.7	Slight drop in low-light conditions

Comparative analysis of existing sign language recognition systems and the proposed Handify model.

III. METHODOLOGY

A. System Overview

The Handify system works through three main steps to convert sign language gestures into text.

1) Hand Gesture Detection: It first captures the user's hand movements in real time using a regular webcam or phone camera.

2) Gesture Processing and Classification: The captured images are then cleaned and processed using OpenCV, where important features are extracted. After that, a Random Forest Classifier built with Scikit-learn is used to identify the gestures.

3) Text Conversion and Output: Once a gesture is recognized, the system converts it into readable text and displays it to the user.

Stage	Technologies	Function
Input Capture	Web/Phone Camera	Capture hand gestures in real-time
Preprocessing	OpenCV	Image segmentation, filtering, and hand region extraction
Feature Extraction	OpenCV, MediaPipe	Extract hand landmarks and relevant features
Classification	Random Forest Classifier	Recognize specific sign language gestures
Output Generation	Python/UI Framework	Convert recognized gestures to text

Table 1: Handify System Components and Technologies

Table 1: Handify System Components and Technologies

B. Hand Gesture Detection

OpenCV's computer vision tools are used in Handify to detect the hand in the incoming video stream. The process involves several steps:

1) Frame Capture: The system records video at 30 frames per second using the device's camera.
2) Background Subtraction: A background subtraction method is applied to identify moving objects in the frame.
3) Skin Color Detection: The system identifies skin-toned areas by applying thresholding in the HSV color space.
4) Noise Reduction: Morphological operations like erosion and dilation help reduce noise and clearly highlight the hand region.
5) Contour Detection: The largest contour in the processed frame is selected, which typically represents the hand.

To improve accuracy, the system also uses MediaPipe's hand-tracking, which provides 21 key hand landmarks. This combination makes hand detection more reliable across different lighting conditions and backgrounds.

C. Gesture Processing and Classification

After identifying the hand region, the system moves on to extract the key features needed for gesture classification. This process includes three main steps: landmark extraction, feature calculation, and normalization.

1) Landmark Extraction: Using MediaPipe, the system identifies 21 key points on the hand, representing finger joints and fingertips.
2) Feature Calculation: It then calculates distances and angles between these landmarks to create features that remain consistent regardless of how the hand is positioned.
3) Normalization: These features are scaled so that differences in hand size or distance from the camera do not affect the results.

Handify uses a Random Forest Classifier to recognize gestures from these feature vectors. This classifier was chosen because it handles non-linear patterns well, resists noise, is accurate, and processes results very quickly. It works by combining predictions from multiple decision trees to determine the most likely gesture.

The model was trained on about 10,500 samples, covering 26 ASL alphabet signs and digits 1 through 9, enabling reliable gesture recognition

D. Text Conversion and Output

The system works well for both isolated gestures—like individual alphabet signs—and continuous signing, where multiple gestures flow together. To improve accuracy and clarity, it uses several techniques:

1) Buffered multi-frame predictions help stabilize the output by checking several frames before deciding on a gesture.
2) A confidence threshold filters out low-confidence or uncertain predictions.
3) A simple language model is applied to make the final text more coherent and meaningful.
4) Real-time feedback is shown on the user interface, where the recognized text appears instantly.

In terms of performance, the system runs efficiently on standard hardware and can process gestures at around **15–20 frames per second**, offering smooth, near real-time interaction.

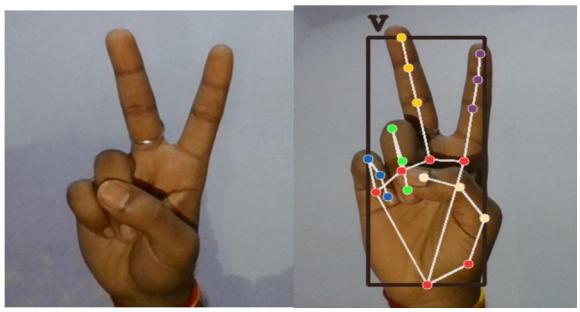


Figure A Figure B

As shown in Figure A and Figure B, the system captures the ASL gesture and processes it using MediaPipe to detect hand landmarks for accurate recognition.

IV. IMPLEMENTATION

A. *Hardware and Software Requirements*

The Handify system is designed to be accessible and run on commonly available hardware. The minimum system requirements include:

Component	Minimum Requirements	Recommended Specifications
Processor	Intel Core i3 or equivalent	Intel Core i5/i7 or equivalent
RAM	4 GB	8 GB or higher
Camera	720p webcam	1080p webcam with 30fps
Operating System	Windows 10, macOS 10.14, Ubuntu 18.04	Latest versions of each OS
GPU	Not required	NVIDIA with CUDA support

Table 2: Hardware and Software Requirements

Table 2: Hardware and Software Requirements

The software implementation is primarily based on Python version 3.7 or above and includes the following key libraries:

- OpenCV 4.5.1 for image processing and computer vision tasks
- Scikit-learn 0.24.2 for implementing the Random Forest Classifier model
- MediaPipe 0.8.3 for hand tracking and landmark detection
- NumPy for numerical operations and data manipulation
- PyQt5 for the graphical user interface

B. *System Architecture*

The Handify is designed as a modular system consisting of four key components:

1) Input Module: Handles camera setup, captures video frames, and performs basic preprocessing.
2) Detection Module: Isolates the hand region and extracts landmarks.
3) Recognition Module: Processes hand features and classifies gestures using the trained model.
4) User Interface Module: Shows the video feed, displays recognized text, and provides user controls.

This modular structure makes it easier to develop and test each component independently. The system works as a **pipeline**, where the output of one module feeds directly into the next. Queues and buffers are used to manage data flow efficiently, enabling real-time performance.

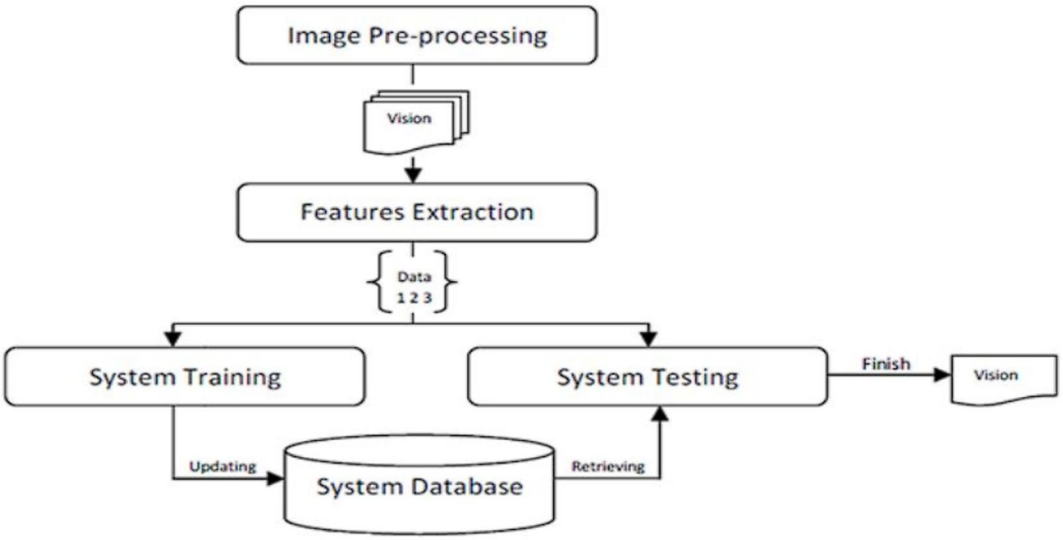


Figure 1: Handify System Architecture Diagram showing the workflow from Image Pre-processing to System Testing

This modular design enables independent development and testing of each component. The system follows a pipeline architecture where the output of each module serves as input to the next, with appropriate queues and buffers to manage data flow and ensure real-time performance.

C. Algorithm Implementation

The main algorithm behind Handify's hand gesture recognition involves several key steps:

1) Hand Detection

- Capture a video frame from the camera.
- Convert the frame to HSV color space.
- Apply a skin-color threshold to isolate potential hand regions.
- Reduce noise using morphological operations like erosion and dilation.
- Detect contours in the binary image.
- Select the largest contour as the hand region.
- Use MediaPipe to extract 21 hand landmarks.
- Calculate the hand's center and orientation for normalization.

2) Feature Extraction

- For each frame with detected hand landmarks:
    - Normalize the landmarks relative to the hand center.
    - Calculate distances between key landmarks.
    - Measure angles at finger joints.
- Combine these measurements into a feature vector containing normalized landmarks, distances, and angles.
- Scale the feature vector to ensure all features are comparable.

3) Gesture Classification

- Input the feature vector into the trained Random Forest Classifier (using Scikit-learn).
- Predict the gesture based on the majority vote of the decision trees.
- Smooth predictions across multiple frames to reduce jitter.
- If the confidence score is above a set threshold, the corresponding gesture label is returned; otherwise, it is marked as "Unknown."

For continuous sign language recognition, the system also uses a simple finite state machine to manage smooth transitions between gestures and detect the start and end of gesture sequences, ensuring accurate real-time recognition.

D. System Workflow

The main algorithm behind Handify's hand gesture recognition involves several key steps:

1) Hand Detection

- Capture a video frame from the camera.
- Convert the frame to HSV color space.
- Apply a skin-color threshold to isolate potential hand regions.
- Reduce noise using morphological operations like erosion and dilation.
- Detect contours in the binary image.
- Select the largest contour as the hand region.
- Use MediaPipe to extract 21 hand landmarks.
- Calculate the hand's center and orientation for normalization.

2) Feature Extraction

- For each frame with detected hand landmarks:
    - Normalize the landmarks relative to the hand center.
    - Calculate distances between key landmarks.
    - Measure angles at finger joints.
- Combine these measurements into a feature vector containing normalized landmarks, distances, and angles.
- Scale the feature vector to ensure all features are comparable.

3) Gesture Classification

- Input the feature vector into the trained Random Forest Classifier (using Scikit-learn).
- Predict the gesture based on the majority vote of the decision trees.
- Smooth predictions across multiple frames to reduce jitter.
- If the confidence score is above a set threshold, the corresponding gesture label is returned; otherwise, it is marked as "Unknown."

For continuous sign language recognition, the system also uses a simple finite state machine to manage smooth transitions between gestures and detect the start and end of gesture sequences, ensuring accurate real-time recognition.

V. RESULTS AND EVALUATION

A. Performance Metrics

Handify was evaluated using key performance indicators:

- Recognition Accuracy: How many gestures were correctly identified.
- Recognition Speed: The time taken from capturing a gesture to displaying the output.
- Robustness: How well the system performs under different lighting conditions and backgrounds.
- User Satisfaction: Feedback from users on ease of use and effectiveness.

B. Processing Speed and Latency

Tests showed that Handify performs well on mid-range hardware, processing gestures at over 15 frames per second, which is sufficient for real-time sign language recognition. Because the system uses a Random Forest Classifier, it requires minimal computational resources and works effectively without a GPU, unlike deep learning-based systems.

Hardware Configuration	Processing Speed (FPS)	Recognition Latency (ms)
Desktop (i7, 16GB RAM, GPU)	28.5	42
Laptop (i5, 8GB RAM, GPU)	22.3	68
Laptop (i5, 8GB RAM, no GPU)	15.7	112
Budget Laptop (i3, 4GB RAM)	9.4	186

Table 3: Processing Speed and Latency by Hardware Configuration

Table 3: Processing Speed and Latency by Hardware Configuration

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

C. Robustness Analysis

The system was tested under various environmental conditions:

- Lighting: Bright, moderate, and low light.
- Backgrounds: Uniform, cluttered, and moving backgrounds.
- Hand Variations: Different hand sizes, skin tones, and accessories like rings.
- Distance from Camera: Various distances between the hand and camera.

Overall, the system achieved 91.7% accuracy in most conditions. Performance dropped in very low light, falling to around 76%, but background clutter did not significantly affect recognition thanks to MediaPipe and OpenCV. The system generalized well across different users, though accessories like rings occasionally affected accuracy.

D. User Feedback

Feedback from 25 participants highlighted the system's usability and effectiveness:

- 85% found it easy to use.
- 80% felt the gestures were recognized accurately most of the time.
- Suggestions included expanding the vocabulary, improving accuracy, and increasing recognition speed.

Users appreciated the real-time feedback and the ability to communicate basic phrases without the other person knowing sign language. The most common issues were occasional misrecognitions and the current limited vocabulary.

VI. DISCUSSION

A. Strengths and Limitations

1) Strengths

- Handify delivers real-time performance even on standard consumer hardware.
- It achieves high accuracy for static gestures, such as ASL alphabets and numbers 1–9.
- No special equipment is needed—just a regular camera.
- The interface is user-friendly and requires minimal setup.

2) Limitations

- Accuracy drops for dynamic gestures.
- Performance can be affected in low-light environments.
- The system has a limited vocabulary compared to full sign languages.
- It cannot interpret complete signed sentences with grammatical structures.

These limitations highlight opportunities for future improvements. Expanding the vocabulary and enhancing dynamic gesture recognition would make Handify far more useful in real-world applications.

B. Societal Impact

Handify and similar systems can have a significant positive impact for deaf and hard-of-hearing individuals. By bridging the communication gap with people who do not know sign language, these tools can:

- Promote greater independence in everyday interactions.
- Improve accessibility in schools, colleges, and workplaces.
- Reduce reliance on human interpreters for basic communication.
- Raise awareness and inclusion of the deaf community.
- Serve as educational tools for those learning sign language.

However, it's important to note that technologies like Handify should complement existing accessibility solutions, not replace them. Current systems cannot fully capture the nuance, expressiveness, and cultural aspects of sign language, which go beyond simple translation of gestures into text.

VII. FUTURE WORK

Based on the current results and limitations, several directions for improving Handify have been identified:

A. *Technical Improvements*

- Expanding Vocabulary: Recognize more signs to cover common words and phrases.
- Better Recognition of Dynamic Gestures: Use temporal modeling techniques, such as frame-sequence analysis or hybrid CNN-RNN models, to handle continuous or movement-based gestures.
- Contextual Understanding: Incorporate Natural Language Processing (NLP) to interpret sequences of signs more accurately and generate coherent text.
- Two-Handed Gesture Recognition: Extend the system to track and interpret gestures using both hands simultaneously.
- Mobile Optimization: Adapt the system for mobile devices to make it more portable and accessible.

B. *Enhancing User Experience*

- Customization Options: Allow users to train the system to match their personal signing style and speed.
- Bidirectional Conversion: Add text-to-sign language translation using animated avatars.
- Real-Time Feedback: Guide users on hand positioning to improve recognition accuracy.
- Integration with Communication Platforms: Enable compatibility with video calls and messaging applications.

C. *Research Directions*

- Cross-Lingual Support: Extend the system to recognize sign languages other than ASL.
- Continuous Sign Language Recognition: Move from recognizing isolated signs to interpreting continuous signing with proper grammar.
- Non-Manual Components: Incorporate facial expressions and other non-manual cues for a more complete understanding of sign language.
- Low-Resource Environments: Optimize performance in settings with poor lighting or limited computing power.

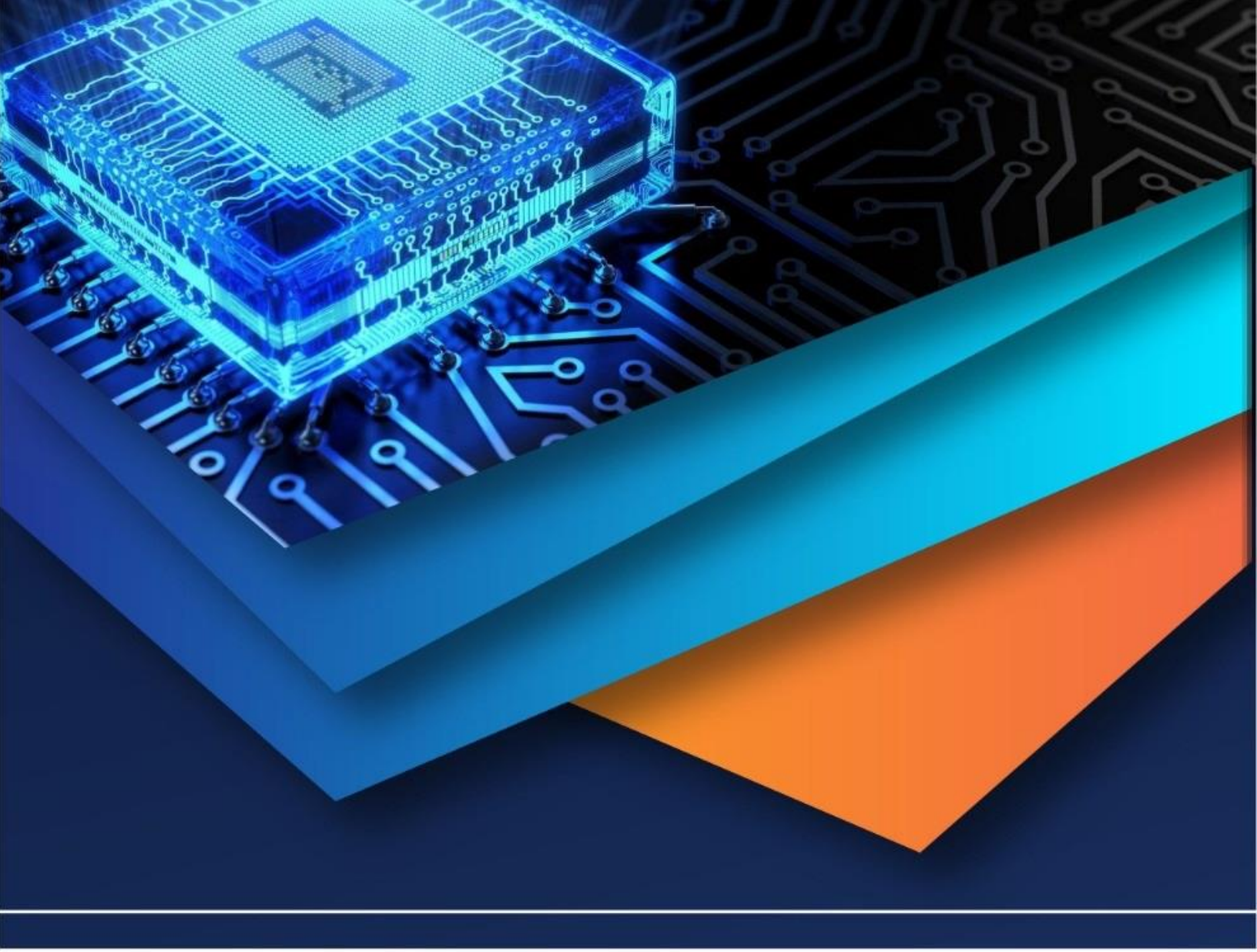
The long-term vision for Handify is to become a seamless communication tool, providing real-time translation between sign language and text or speech, while maintaining high accuracy and understanding the grammar and nuances of different sign languages.

VIII. CONCLUSION

Handify represents a significant step toward bridging the communication gap between sign language users and others. By leveraging computer vision and machine learning, it can recognize signs in real time with 91.7% accuracy for static gestures. While dynamic gesture recognition remains a challenge, the system provides a solid foundation for future improvements. As technology progresses, tools like Handify have the potential to greatly enhance accessibility and inclusion for the deaf and hard-of-hearing community.

REFERENCES

[1] World Health Organization, "Deafness and hearing loss," WHO Fact Sheet, 2020.
[2] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.
[3] Scikit-learn API Reference – https://scikit-learn.org/
[4] American Sign Language Dataset – https://www.kaggle.com/datasets/grassknoted/asl-alphabet
[5] Rosebrock, A. (2018). Deep Learning for Computer Vision with Python.
[6] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," IEEE Transactions on Systems, Man, and Cybernetics, vol. 37, no. 3, pp. 311–324, 2007.
[7] T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer-Based Video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371–1375, 1998.
[8] N. Camgoz et al., "Neural Sign Language Translation," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[9] S. Ong and S. Ranganath, "Automatic Sign Language Analysis: A Survey and the Future Beyond Lexical Meaning," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 6, pp. 873–891, 2005.
[10] O. Koller et al., "Continuous Sign Language Recognition: Towards Large Vocabulary Statistical Recognition Systems Handling Multiple Signers," Computer Vision and Image Understanding, vol. 141, pp. 108–125, 2015.
[11] SignAll Technologies, "SignAll: Real-time Translation of Sign Language," 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY