Indian Sign Language Detection

Aditya Shinde[1], Shivam Dhawale[2], Vedant Lokhande[3], Chinmay Rawool[4], Prof. Pooja Pawale[5]
Department of Computer Science and Engineering, MIT Art, Design and Technology University[1,2,3,4]
Professor at MIT ADT University[5]

Abstract - The communication gap remains one of the most significant barriers between individuals with hearing and speech impairments and the broader society. This project addresses this challenge by developing a real-time Indian Sign Language (ISL) detection system that leverages computer vision and machine learning techniques. By capturing hand gestures from video input, the system translates these movements into text or speech, enabling effective communication between ISL users and those unfamiliar with the language. Additionally, the system incorporates text-to-speech functionality, ensuring a seamless and humanized interaction experience.

The proposed model utilizes Convolutional Neural Networks (CNNs) for image processing and gesture recognition, trained on a comprehensive dataset of ISL gestures. The framework employs preprocessing, feature extraction, and classification algorithms to accurately identify static and dynamic gestures. The system is designed to focus on the nuances of ISL, providing accurate recognition of gestures in real time while offering multilingual support.

This initiative aspires to create an inclusive environment by empowering the hearing-impaired community and promoting better integration within society. By using cost-effective techniques, the project ensures scalability and practicality for everyday applications, making communication more efficient and inclusive.

Keywords: Indian Sign Language (ISL), Gesture Recognition, Convolutional Neural Networks (CNNs), Real-time Communication

1.Introduction

Sign language is an essential means of communication for individuals who are Deaf and Hard of Hearing (DHH), relying on a variety of gestures that encompass hand shapes, movements, and orientations. In India, Indian Sign Language (ISL) serves as the primary communication method for those with hearing and speech impairments. ISL consists of 26 distinct hand poses, which include letters,

enabling users to convey complex thoughts and messages effectively.

However, many people struggle to understand ISL gestures, leading to significant communication barriers between the DHH community and those who do not know sign language. Traditional solutions, such as sign language interpreters, can be costly and are not always readily available. This gap underscores the urgent need for an automated system capable of recognizing and translating ISL gestures in real-time, particularly in public settings like banks, hospitals, and post offices.

The objective of this research is to create a reliable system that can translate ISL gestures into spoken language, facilitating smoother interactions between DHH individuals and the hearing population. Our approach involves using a camera to capture hand gestures, which are then processed on a server to achieve fast and accurate recognition. Initially, our focus is on single-handed gestures representing only alphabets, with plans to extend the system to accommodate two-handed gestures in the future. No digits or facial expressions will be detected.

While gesture recognition has been widely explored for American Sign Language (ASL), research on ISL remains limited. Our project aims to fill this gap by leveraging advanced computer vision and machine learning algorithms, avoiding the use of expensive technologies like gloves or Kinect systems. Ultimately, this initiative aims to empower the DHH community in India and promote inclusivity by enhancing communication between all individuals.

The following sections of this paper will cover related work in sign language recognition, the techniques used in our system, experimental results, the development of a real-time translation Android application, and potential future directions for ISL translation research. Through this work, we aspire to create a transformative tool that alleviates communication challenges faced by the DHH community.

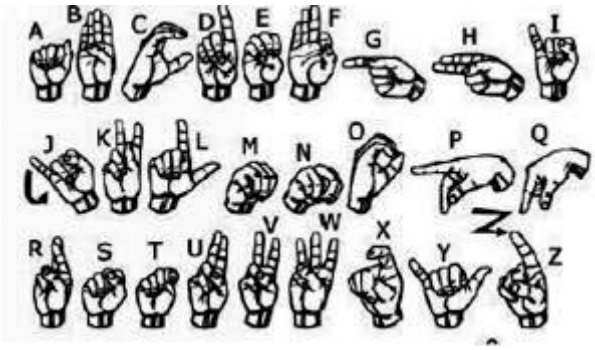

Figure 1: Indian sign language dataset([12]Amrutha, C.U. & Davis, Nithya & Samrutha, K.S. & Shilpa, N.S. & Chunkath, Job. (2016))

2. Related work

Research on sign language recognition has evolved significantly over time. Early methods primarily relied on sensor-based technologies, such as gloves equipped with sensors, to track hand movements. However, these systems were often too expensive and impractical for everyday use.

Sawant and Kumbhar [1][11] proposed a real-time system for recognizing sign language using Principal Component Analysis (PCA). Their dataset comprised 260 images representing 26 different signs. They employed Otsu's method for segmentation, morphological filtering to reduce noise, and computed the principal components for gesture recognition. By calculating the Euclidean distance between gestures, the system projected the recognized gestures onto a gesture space using the eigenvector matrix, outputting the corresponding text and voice.

Suriya M. et al. [2] developed a real-time recognition system that uses Linear Discriminant Analysis (LDA) for feature extraction and K-Nearest Neighbors (KNN) for classification. Their system evaluated similarity measures like Euclidean distance and Cosine similarity, which helped in accurately identifying gestures based on these metrics.

Kodandaram et al. [3] introduced a system that used Convolutional Neural Networks (CNNs) for hand gesture recognition. Their approach converted the recognized gestures into English text, which could then be translated into speech, showcasing CNNs' effectiveness in image-based recognition.

Sharma, Pal, and Sahoo [4] focused on recognizing Indian Sign Language (ISL) numerals. Their work enabled communication in public spaces, like banks or transportation hubs, for people with hearing or speech impairments, making it a practical solution that specifically addressed the needs of ISL users.

Ekbote and Joshi [5] extended this research by designing a system for recognizing ISL digits (0–9) using Artificial Neural Networks (ANN) and Support Vector Machines (SVM). Their work marked a step forward in automating ISL recognition, a field that has not been as thoroughly researched as American Sign Language (ASL).

Recent advancements in deep learning have led to more sophisticated systems. Kothadiya et al. [6] presented a deep learning-based model, leveraging CNNs to detect and recognize words from hand gestures. Their work demonstrates the potential of deep learning to improve the accuracy of sign language recognition.

Gupta [7] designed a model that uses CNNs, along with contour extraction and morphological operations, to predict emojis based on hand gestures. This system trained on a dataset of 11 hand gestures, achieved high recognition accuracy.

Mo [8] explored the main challenges in gesture recognition, including key features such as shape, position, and geometric properties. He also highlighted the role of different color spaces, such as RGB and HSV, in human skin detection, while discussing popular gesture recognition methods like Hidden Markov Models, Neural Networks, and Template Matching.

Lee et al. [9] introduced a novel approach for sign language recognition using Wi-Fi signals. Their system, powered by a two-stream Convolutional Neural Network (CNN), combined spatial and motion data for accurate gesture recognition, achieving impressive results on the SignFi dataset.

Antad et al. [10] developed a multi-language sign translation platform using deep learning and CNNs. Their system supported both ISL and ASL, providing translations into several Indian regional languages. This cross-lingual approach made it easier for people to communicate across different sign languages.

Rajam et al. [11] designed a real-time ISL recognition system aimed at helping individuals with speech and hearing impairments communicate in public spaces, without the need for interpreters.

3. Proposed methodology

A. Data Collection:

We utilized the **OpenCV** library to capture real-time video input through a webcam. Frames were extracted and saved into different alphabetic categories ('A' to 'Z') based on user input, storing them into pre-defined directories (Image/A/, Image/B/, etc.). A bounding box was drawn to define the Region of Interest (ROI), capturing hand gestures for each letter. Each gesture was stored as a sequence of frames.

B. Preprocessing:

Each captured frame was resized to a standard dimension and converted to RGB using **OpenCV**. A pre-trained **MediaPipe Hands** model was used to detect and extract key landmarks from each frame. These landmarks were saved as NumPy arrays for further processing.

C. Model Architecture:

The model starts with an LSTM layer consisting of 64 units, designed to process input sequences with a shape of (30, 63)—representing 30 timesteps with 63 features at each step. This layer utilizes the ReLU activation function and has return sequences set to True, ensuring the output retains a 3D tensor shape of (batch_size, 30, 64) for further layers. Following this, another LSTM layer with 128 units is added, also configured with return sequences=True. It processes the output from the first layer while maintaining the sequence structure,

transforming it into (batch_size, 30, 128). The third LSTM layer, with 64 units, differs in that it sets return_sequences=False, reducing the sequence data to a 2D vector of shape (batch_size, 64). This vector is then passed to a fully connected Dense layer with 64 units and ReLU activation, producing an output of shape (batch_size, 64). Another Dense layer further reduces the dimensionality to 32 units, again with ReLU activation. The final layer, a Dense layer with softmax activation, predicts the probability distribution across the number of classes defined by actions.shape[0], resulting in an output shape of (batch_size, actions.shape[0]). This architecture effectively combines recurrent and dense layers for sequence classification tasks.

LSTMs, or Long Short-Term Memory networks, are a type of recurrent neural network (RNN) specifically designed to capture long-term dependencies in sequential data. They achieve this by incorporating a gating mechanism that regulates the flow of information within the network.

Core Components and Functions:

Forget Gate: This component decides which parts of the cell state should be removed, ensuring the model forgets irrelevant information and retains essential details for future processing.

$$ft = \sigma(Wf \cdot [ht - 1, xt] + bf)$$

ft: Forget gate output (values between 0 and 1).
Wf ,bf: Learnable weights and biases.
ht−1: Previous hidden state.
xt: Current input.
σ: Sigmoid activation function.

Input Gate: Decides which information to update in the cell state.

$$it = \sigma(Wi \cdot [ht - 1, xt] + bi)$$
$$C\sim t = \tanh(WC \cdot [ht - 1, xt] + bC)$$
A

$$Ct = ft \cdot Ct - 1 + it \cdot C\sim t$$

it : Input gate output.
C~t : Candidate cell state.
Ct : Updated cell state

Output Gate: Controls what information flows to the next hidden state.

$$ot = \sigma(Wo \cdot [ht - 1, xt] + bo)$$
$$ht = ot \cdot \tanh(Ct)$$

ot : Output gate output.
ht : Current hidden state (LSTM output).

A Dense layer establishes full connectivity between neurons, meaning every neuron in the current layer is linked to all neurons in the subsequent layer. These layers are pivotal for capturing high-level patterns and representations in the data.

The output of a Dense layer is calculated using:

$$y = \sigma(W \cdot x + b)$$

y: Output vector.
W: Weight matrix.
x: Input vector.
b: Bias vector.
σ: Activation function (e.g., ReLU, Sigmoid, or Softmax).

D.Evaluation:

Model performance was tracked using **TensorBoard** callbacks to visualize loss and accuracy metrics. The testing set was used to evaluate the final model accuracy.

4.Actual Execution and Results

A.Data Collection

Figure 2: Preview of created dataset

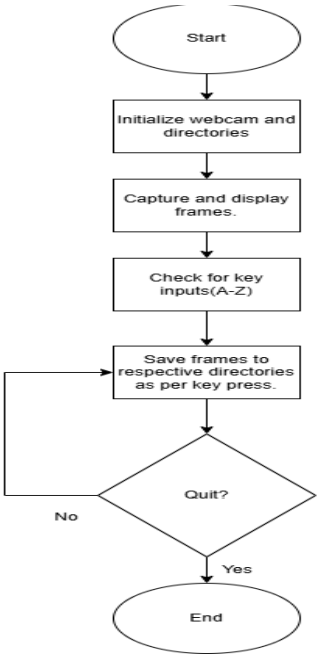

Figure 3 : Dataset creation

Setup and Initialization:

- Import Libraries: Import os for directory operations and cv2 for computer vision tasks.
- Video Capture Initialization: Initialize video capture from the default webcam using cv2.VideoCapture(0).
- Directory Path: Set the path where images for each letter (A-Z) will be saved.

Main Loop:

- Capture Frames: Continuously capture frames from the webcam using cap.read().
- Count Images: Create a dictionary to keep track of the number of images already saved in each letter's folder, ensuring new images are named sequentially.

Display Frames:

- Define ROI: Draw a rectangle on the frame to indicate the Region of Interest (ROI) using cv2.rectangle().
- Display Frames: Show both the original frame and the ROI using cv2.imshow().

Capture and Save Images:

- Keyboard Interrupts: Detect key presses (a-z) to capture the current frame.
- Save Images: Save the frame in the appropriate directory based on the key pressed. Use cv2.imwrite() to write the image file, naming it sequentially based on the count.

Cleanup:

- Release Resources: Stop video capture with cap.release().
- Close Windows: Close all OpenCV windows with cv2.destroyAllWindows().

B.Data Preprocessing

Library Imports:

- Import os for handling directories.
- Import cv2 for computer vision tasks.
- Import numpy for numerical operations.
- Import mediapipe for hand landmark detection.

Constants:

- Set DATA_PATH as the directory for saving images and data.
- Define actions as a list of uppercase letters from 'A' to 'Z'.
- Set no_sequences to 30 for the number of sequences per action.
- Set sequence_length to 30 frames per sequence.

MediaPipe Setup:

- Initialize MediaPipe Hands and drawing utilities for hand landmark detection.

Helper Functions:

- mediapipe_detection(image, model): Detects hand landmarks in an image.
- draw_styled_landmarks(image, results): Draws the detected landmarks on the image.
- extract_keypoints(results): Extracts and flattens hand landmark data into an array.

Directory Setup:

- Create directories for each action and sequence under DATA_PATH.

Data Collection Loop:

- For each action:
- Print action being collected.
- For each sequence and frame:
- Read the frame from a predefined path.
- Skip if the frame is missing.
- Detect hand landmarks and draw them on the frame.
- Display the frame with status text using cv2.putText() and cv2.imshow().

- Extract and save keypoints as .npy files.

Cleanup:

- Release video capture with cap.release() and close all OpenCV windows using cv2.destroyAllWindows().

C. Function prototype:

Imports essential libraries: cv2 for video processing, numpy for handling numerical data, and mediapipe for detecting hand landmarks. Key functions include:

- mediapipe_detection: Converts images to the proper format for MediaPipe and processes them to detect hand landmarks.
- draw_styled_landmarks: Visualizes detected hand landmarks on the image.
- extract_keypoints: Retrieves the 3D coordinates of hand keypoints and flattens them into an array.

The configuration section defines:

- DATA_PATH: The directory to save processed data.
- **actions: The set of hand gesture labels, such as 'A', 'B', and 'C'.**
- **no_sequences and sequence_length: Parameters for the number of action sequences and frames per sequence.**

D.Train Model

Layer(type)	Output shape	Param #
lstm (LSTM)	(NONE , 30 , 64)	32,768
lstm_1 (LSTM)	(NONE , 30 , 128)	98,816
lstm_2 (LSTM)	(NONE , 64)	49,408
dense (DENSE)	(NONE , 64)	4,160
dense_1 (DENSE)	(NONE , 32)	2,080
dense_2 (DENSE)	(NONE , 26)	858

Figure 4:Model architecture an paramaters

Data Preparation:

1) The code processes hand key points stored in .npy files, associating each sequence with an action label.

Model:

2) The architecture is based on LSTM (Long Short-Term Memory) layers, which are ideal for sequence prediction. The model comprises three LSTM layers followed by Dense layers and uses categorical cross-entropy loss along with accuracy metrics to classify multiple actions.

Training:

3) The dataset is split with 95% for training and 5% for testing. The model is trained for 200 epochs using a TensorBoard callback for logging.

Model Saving:

- The trained model architecture is saved as a JSON file, and its weights are saved in .h5 format

E.Real -Time Visualization

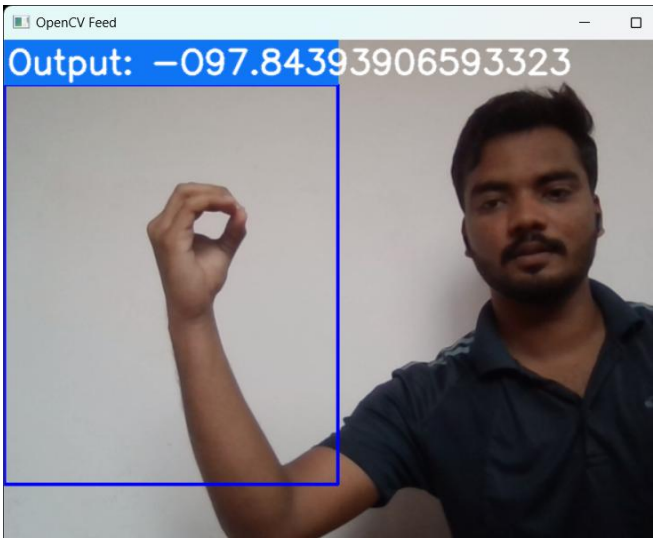

Figure 5:Output screen

Model Initialization:

A pre-trained model is loaded from a model.json file, and its associated weights are restored from a model.h5 file. This model is specifically designed to recognize hand gestures based on sequential input data.

Gesture Prediction Process:

The system captures live video frames using OpenCV and identifies hand landmarks within these frames through MediaPipe. Extracted keypoints from the landmarks are accumulated in sequences of 30 frames. Once a complete sequence is formed, it is passed through the LSTM-based model, which predicts the corresponding gesture.

Dynamic Feedback Display:

The recognized gesture is displayed on the screen, with predictions being refined based on confidence levels and the consistency of recent outputs. When the model's confidence exceeds the predefined threshold (0.8), the action is shown alongside its confidence percentage to ensure accurate feedback.

Audio Feedback Mechanism:

Users can receive audio feedback for the identified gesture. By pressing the **Enter** key, the last predicted gesture is spoken aloud using a text-to-speech engine, making the system more interactive and accessible.

Live Interface:

The program continuously processes video feed, updating the predictions and their confidence scores in real-time. Users can terminate the system by pressing **'q'**, providing a seamless and user-controlled experience.

5. Future Scope

The Indian Sign Language detection system offers substantial potential for growth and application across various domains. As technology evolves, integrating advanced features can significantly enhance its utility and accessibility. Below are the key areas for future development:

A.Advancements in Machine Learning Models

Advanced Deep Learning Techniques: Exploring cutting-edge architectures like Transformers and complex convolutional neural networks (CNNs) can improve recognition accuracy. Transfer learning from pre-trained models on large datasets can further boost performance, even with limited ISL-specific data.
Real-Time Personalization: Algorithms that adapt to individual signing styles in real time can provide a more personalized and accurate user experience.

B.Integration with Augmented and Virtual Reality

Augmented Reality (AR) Applications: Real-time visual translations or sign cue overlays via AR can enhance user interaction in educational and public settings.
Virtual Reality (VR) Environments: Immersive VR platforms can enable users to practice ISL in simulated settings, fostering engagement for both deaf and hearing individuals.

C.Mobile and Wearable Technology

Mobile Accessibility: Expanding ISL-based mobile applications will make the technology accessible, allowing seamless interactions through smartphones and tablets.
Wearables: Sensor-embedded gloves and other wearable devices can capture subtle gestures that cameras may miss, offering an alternative approach to gesture recognition.

D.Diverse Datasets for Enhanced Accuracy

Diverse Signer Representation: Incorporating data from signers across different regions, age groups, and backgrounds ensures a comprehensive dataset that captures the full variability of ISL.

Continuous Learning Capabilities: Feedback loops to collect user interaction data can enable models to evolve and adapt to changes in sign language usage.

E.Cross-Language Support

Multilingual Sign Language Recognition: Extending recognition to include other sign languages like American Sign Language (ASL) will foster communication between different communities.
Text Translation: Utilizing natural language processing (NLP) to translate signs into multiple spoken languages will facilitate broader accessibility.

F.Educational Tools and Resources

Learning Platforms: Web-based platforms combining ISL detection with interactive learning modules can support both deaf and hearing users in acquiring sign language skills.
Teacher Training Programs: Professional development initiatives for educators can promote inclusive education by incorporating ISL detection systems in classrooms.

G.Community Engagement and Public Awareness

Awareness Initiatives: Public campaigns can educate people about ISL and its detection systems, fostering inclusivity and understanding.
NGO Partnerships: Collaborations with NGOs focused on disability rights will ensure that the technology reaches those most in need.

H.Policy Advocacy and Support

Government Partnerships: Collaborating with government agencies can aid in policy development for integrating ISL technology into education and public services.
Research Funding: Securing grants for accessibility technology can drive advancements in ISL detection systems, enabling further development and refinement.

By addressing these focus areas, the ISL detection system can evolve into a transformative tool, empowering the hearing-impaired community and bridging communication gaps across society.

6.Conclusion

The proposed real-time Indian Sign Language (ISL) detection system aims to bridge the communication gap between individuals with hearing and speech impairments and the wider society. By leveraging computer vision and machine

learning techniques, this system translates hand gestures into text or speech, facilitating effective and inclusive communication. The integration of text-to-speech functionality further humanizes interactions, enabling the system to be utilized in real-world settings such as public offices, hospitals, and banks. The project offers a cost-effective, scalable solution, promoting inclusivity and improving accessibility for the hearing-impaired community in India.

References

[1] Adithya V and Rajesh R (2020) A deep convolutional neural network approach for static hand gesture recognition. Procedia Computer Science 171: 2353–2361. DOI: 10.1016/j.procs.2020.04.255

[2] Suriya M., Sathyapriya N., Srinithi M., Yesodha V., "Survey on Real-Time Sign Language Recognition System: An LDA Approach," 2020.

[3] Satwik Ram Kodandaram, N. Pavan Kumar, Sunil Gl, "Sign Language Recognition," Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(14):994-1009, August 2021.

[4] Madhuri Sharma, Ranjna Pal, Ashok Kumar Sahoo, "Indian Sign Language Recognition using Neural Networks and kNN Classifiers," Journal of Engineering and Applied Sciences, 9(8):1255-1259, 2014.

[5] Juhi Ekbote, Mahasweta Joshi, "Indian Sign Language Recognition using ANN and SVM Classifiers," International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017.

[6] Rui Ma, Zendong Zhang, Enqing Chen, "Human Motion Gesture Recognition Based on Computer Vision", February 2021

[7] J. Gupta, "Hand Gesture Recognition for Emoji Prediction," International Journal of Research in Applied Science and Engineering Technology, 2020.

[8] S. Narayan Sawant, M. S. Kumbhar, "Real-Time Sign Language Recognition using PCA," 2014.

[9] C.-C. Lee, Z. Gao, "Sign Language Recognition Using Two-Stream Convolutional Neural Networks with Wi-Fi Signals," Applied Sciences, 10(9005), 2020.

[10] S. M. Antad, S. Chakrabarty, S. Bhat, S. Bisen, S. Jain, "Sign Language Translation Across Multiple Languages," 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, 2024.

[11] Rajam, P. Subha, G. Balakrishnan, "Real-Time Indian Sign Language Recognition System to Aid Deaf-Dumb People," IEEE 13th International Conference on Communication Technology (ICCT), IEEE, 2011.

[12] Amrutha, C.U. & Davis, Nithya & Samrutha, K.S. & Shilpa, N.S. & Chunkath, Job. (2016). Improving Language Acquisition in Sensory Deficit Individuals with Mobile Application. Procedia Technology. 24. 1068-1073. 10.1016/j.protcy.2016.05.237.