International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. **11** Special Issue **03** | **2026**

A Real Time Bidirectional Sign Language Translation System for Better Communication

Shubhankar Sawant[1], Omkar Tarve[2], Shravani Kadam[3], Madhura Zagade[4]

[1,2,3] *Final Year Student, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India*
[4] *Assistant Professor, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India*

DOI: 10.5281/zenodo.20627815

ABSTRACT

Still, many people struggle to connect when one person hears well and another does not - especially across India's mix of languages and cultures. Though Indian Sign Language helps bridge gaps for those who are deaf or have trouble hearing, few outside the community can follow it. Instead of relying on human interpreters alone, technology steps in here: a tool built to spot hand movements as they happen. Using cameras, the setup grabs moving images straight away, feeding them into software tuned to detect finger positions. Before anything gets labeled, raw footage goes through cleanup stages so key points stand out clearly. Then comes guessing what each pose means - an algorithm trained on hundreds of examples makes these calls. Once recognized, signs turn directly into words appearing line by line below the screen. What shows up is plain text, matched closely to how hands shaped meaning just seconds before. Fingers and hand positions show up clearly even in dim light, thanks to MediaPipe spotting key points reliably. From those points, measurements get adjusted to a common scale before moving forward. One after another, different prediction systems learned from the data, each tested by how often they got it right, missed signs, or made errors. Whichever one handled mistakes best now runs live, catching gestures as they happen and showing words without delay. Parts of the setup fit together like building blocks, making future updates easier to plug in later. When tested, the system recognizes quickly while staying accurate, so it works well in real-world help tools. Because of how fast and correct it is, people can use it daily without delays slowing them down. What stands out is its ability to respond instantly yet reliably during live tasks. The work pushes forward how machines understand humans, especially those needing support. By focusing on speed and precision, it opens doors for fairer tech access across different users. Automation like this helps everyone join conversations more easily. Instead of complex setups, it runs smoothly on simpler hardware too. With less waiting comes better flow in user experience. One result? Technology that adapts to people rather than the reverse. Inclusion grows when systems listen right the first time

Keyword : - Indian Sign Language, Gesture Recognition, Computer Vision, MediaPipe, Machine Learning

1. INTRODUCTION

These days, more people are focusing on sign language recognition because it connects deeply with society and helps bridge communication gaps. Speaking through signs? That's how many deaf or speech-impaired folks mainly share thoughts. Yet misunderstandings pop up when signers meet those who do not know signing - schools, government offices, even chats at cafés can become tricky. Relying on human translators works sometimes, but here's the catch: they're often hard to find, cost too much, or simply aren't around when needed most.

Signs used in India carry traits unlike those seen in systems like American Sign Language. Because of this, tools built overseas need heavy changes before they work here. Movements in Indian signing come in two kinds: still ones, like letters or digits, along with flowing sequences for full thoughts. These mix location shifts with timing demands, making automatic reading tough. Each motion adds layers a machine must untangle correctly.

Computer vision keeps evolving, so spotting sign language in real time feels less out of reach now. Thanks to new tools tracking hand points, key details leap from video feeds with strong precision. Instead of static poses alone, movement flows get captured too - thanks to smart layers like CNNs plus memory-focused LSTMs working behind the scenes. Ordinary cameras do the job just fine; heavy gear sits on the sidelines. Lightweight setups run smoothly because pieces fit together without demanding extras.

One idea here moves fast - turning sign language into text as it happens. Instead of relying on existing tools, parts snap together: dots mapped on hands feed a still-gesture reader, while motion flows into another network built with layers that learn sequences. Because ready-made clips fell short, new ones were recorded by hand to fill gaps. What takes shape? A bridge where silence meets speech without slowing down. Quiet moments become words others can follow.

1.1 Problem Statement

Building a machine that understands Indian Sign Language runs into trouble because the language itself is complex - plus there are almost no big, uniform data collections to train it on. Most current tools only look at single hand shapes held still, missing how signs flow over time when people actually communicate. What happens next? They cannot catch full words or sentences properly. Some methods also rely on custom sensors you do not find everywhere, which makes broad use unlikely. Real life settings often lack such gear.

Facing changing light, messy backgrounds, or different people signing can trip up many models. When running nonstop video through these systems, delays pop up while guesses get less reliable. Some designs still struggle to turn speech into signs just as well as they turn signs into words - both inside one setup.

So here it starts - building a smart sign language tool using just cameras. This setup focuses on reading both still poses and moving signs quickly. What matters most? Getting predictions right, almost every time. Speed counts too, with responses that lag less than a blink. It adapts when lighting shifts or backgrounds change. Communication flows two ways, smooth and natural. No extra gadgets needed. Everything runs inside the model. Efficiency stays high, even under pressure. Accuracy never drops off. That quiet moment when understanding clicks? That is what shapes the whole effort.

1.2 Objectives

One goal drives this work: building a live tool that sees Indian Sign Language and turns it into words fast. Aiming sharp at precision, the method leans on cameras plus neural networks to capture gestures. Speed matters just as much as correctness when bridging talk between signers and those who do not know signing. Real people in real settings must be able to use it without hassle. Vision-based models train heavily so mistakes drop low over time. Making access smooth sits core to how the tech moves forward. Deployment outside labs tests whether it holds up under daily pressure.

- To build a system using computer vision techniques that can recognize sign language gestures from live video input with good accuracy
- To create a translation setup that works in both directions, meaning the system should convert signs into text and also generate sign output when text is entered.
- To make sure the system works smoothly in real-time so users can communicate without noticeable delay during recognition and translation.
- To develop an interface that is simple and easy to understand so that both sign language users and general users can operate it comfortably.
- To include a feedback option where users can confirm or correct the predicted result, and store this information for future reference and possible improvements.

Cost savings come first when the new setup steps in, bringing scale without strain while moving smoothly into daily use. Efficiency shows up quietly, built right into how messages travel from one point to another. Inclusion gets stronger as fewer barriers stand in the way of connection. Human helpers are still around but needed less often now, their role lightened by smart design choices.

2. SYSTEM ARCHITECTURE AND DESIGN

The proposed system is designed using a modular and bidirectional structure to support real-time sign language translation. It works in two directions: sign-to-text and text-to-sign, so users can both understand and generate sign communication. The development started with collecting different types of data, including static images and dynamic gesture videos, from online sources and custom recordings to make the dataset more realistic and varied. After collecting the data, preprocessing was done to resize images, extract frames from videos, detect hand landmarks using MediaPipe, and clean unnecessary or noisy samples so that the models could learn properly.

The training phase was handled separately for static and dynamic gestures since both require different approaches. Static signs were trained using hand landmark coordinates, which helped the system recognize letters and numbers based on finger positions. For dynamic gestures, we used a CNN–LSTM model because simple image-based methods were not enough to capture hand movement properly. Since dynamic signs involve motion over multiple frames, the model needed to learn how gestures change step by step. In the case of dynamic gestures, we used a CNN–LSTM model because recognizing moving signs is different from recognizing a single image. A normal image-based approach was not enough since the hand keeps

changing position over time. What shows up in every frame gets figured out by the CNN, while the LSTM takes care of tracking how things move between frames. This setup allows gesture recognition to run smoother, leaving room for upgrades down the line.

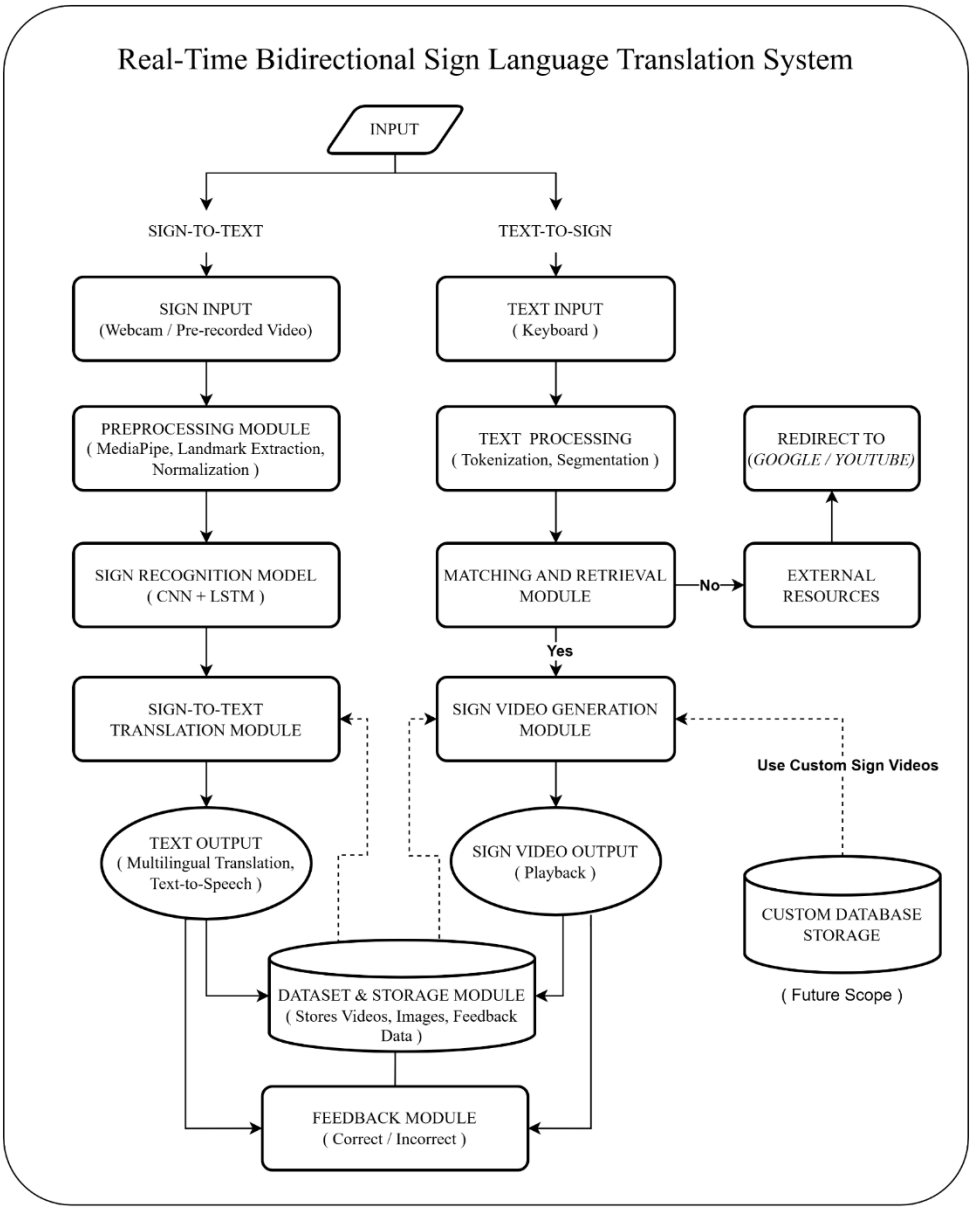

Fig -2.1: Bidirectional Sign Language Translation System Architecture

2.1 Data Collection

A good sign language recognition system relies heavily on how clear, varied, and evenly spread the data is. This study pulled material from open web resources along with personally filmed signing clips to cover both still poses and moving gestures. The mix helps systems learn more natural variations in real-world use.

The dataset was organized into the following categories:

1. **Static gestures:** Alphabets and numbers (image-based)
2. **Dynamic gestures:** Words (video-based)
3. **Dynamic gestures:** Sentences (video-based)

A single image shows each letter and number, pulled from open databases where over a thousand examples exist for every symbol, capturing shifts in angle, size, or lighting. Videos of full sentences arrived separately through web sources - each phrase appears at least six times in motion clips.

One big hurdle that came up while gathering data was not having enough word-by-word sign language examples. Because of this gap, a new set of videos focused on individual words had to be made from scratch. Filming took place at Z.P. Full Primary School Lavgan No.1, where cooperation played a key role.[12] Mr. Suresh Ganapati Aagle helped everything move smoothly by arranging both space and people

involved. From this teamwork came access to willing participants, along with a setup built for recording natural sign movements. Each word ended up with around five video clips taken. Lighting stayed the same across sessions. The backdrop didn't change either.

2.2 Data Preprocessing

Preprocessing steps included:

1. Image resizing and pixel normalization
2. Video frame extraction at a fixed frame rate
3. Removal of noisy, blurred, and redundant frames
4. Hand landmark extraction using MediaPipe
5. Temporal sequence alignment for dynamic gestures
6. Label encoding and dataset organization

The final dataset was divided into training, validation, and testing sets to ensure reliable evaluation.

2.3 Model training

Training started by splitting static signs from moving ones, since they behave differently. What mattered most came through practice - spotting key details fast. Each model learned patterns on its own time, shaped by how hands actually move. Recognition sharpened slowly, built step by step without shortcuts. Speed grew alongside precision, neither leading nor lagging behind.

Static Gesture Model:

For static gesture recognition, hand landmark coordinates were extracted using the MediaPipe framework. Each detected hand produced structured keypoints representing finger joints and palm positions. These numerical landmark values were used as input features for supervised classification. By relying on joint coordinates instead of raw images, the model reduced sensitivity to background noise, lighting variations, and camera inconsistencies.

During training, the classifier learned to associate specific landmark patterns with corresponding alphabets and numerical signs. Since the model focused only on skeletal hand structure, it achieved stable performance across different recording environments. Minor misclassifications occurred in cases of rapid hand movement or partial occlusion; however, overall convergence was consistent, and classification accuracy improved progressively across epochs.

Dynamic Gesture Model:

Dynamic gesture recognition was implemented using a hybrid Convolutional Neural Network (CNN)[8] and Long Short-Term Memory (LSTM)[9] architecture to effectively capture both spatial and temporal features.

The CNN layers were responsible for extracting spatial features from individual video frames. Each frame was processed through convolutional filters that identified edges, shapes, and hand patterns. Through multiple convolution and pooling layers, hierarchical feature representations were formed. These layers automatically learned discriminative visual patterns without requiring manual feature engineering.

However, spatial features alone are insufficient for dynamic sign recognition because gestures evolve over time. To address this, the extracted frame-level features were sequentially fed into LSTM layers. The LSTM network modeled temporal dependencies by maintaining memory states across time steps. This enabled the system to understand motion continuity, frame transitions, and sequential variations in hand positions.

The LSTM units selectively retained important temporal information while discarding irrelevant variations, allowing the model to differentiate between gestures that appear similar in individual frames but differ in motion sequence. By combining CNN-based spatial encoding with LSTM-based temporal modeling, the architecture achieved strong performance in recognizing both word-level and sentence-level gestures.

To ensure stable learning, sequences were normalized to fixed lengths, and training was conducted over multiple epochs with validation monitoring. The integrated CNN–LSTM model demonstrated improved generalization capability and consistent real-time inference performance.

2.4 Sign-to-Text Pipeline

➢ Sign Input

From a webcam, live footage streams in. Or it takes saved clips showing hand movements. Each piece moves on to be checked, one frame at a time. Processing begins right after the frames arrive.

➢ Sign Gesture Processing

From each video, individual images get pulled out first. After that, they go through a cleanup step to prepare them. A tool called MediaPipe spots hands and picks out key points on them. These cleaned snapshots then undergo scaling so sizes match up. Later, they line up in order to study movement over time.

➢ Sign Recognition Model

A single part that spots things works in two ways. One way looks at pieces separately, while another checks them together. Each method adds something different to how it sees. Together they form the full system for

noticing stuff

- **Static Gestures:**

Twenty-one points on a hand get mapped by MediaPipe[14]. From there, a model tuned to recognize patterns takes those scaled positions - matching them to letters or digits.

- **Dynamic Gestures:**

A CNN-LSTM architecture is applied

1. Frames pass through CNN[8] layers, pulling out visual details step by step. Each layer highlights different shapes or edges, building a layered view over time
2. Time steps connect through LSTM[9] units, shaping how past inputs influence what comes next. Frames link together when the network learns patterns over moments in sequence.
3. Last bits decide class through weighted outputs turned into probabilities. Softmax shapes those scores so they add up right
4. Fingers form shapes while movement adds meaning, together allowing precise detection. The system sees still poses at the same time as it tracks dynamic signs.

➢ Sign-to-Text Translation

A single guess at what hand motion was made gets turned into words through a set list of matching terms. Each sign has its own phrase waiting in that collection.

➢ Text Output

1. A screen shows what the system has processed. On top of that, it appears right where you can see it
2. File storage works for the resulting text. Saving what comes out is possible.
3. One moment it's in English, then shifts into Hindi, flows next into Marathi, slips suddenly into Tamil - each version lives its own life across India's tongues.
4. Pressing the voice button triggers spoken words using a TTS component. Words come out loud when the system reads text aloud after that tap. Sound forms instantly, built from written input by hidden processing. A small program translates letters into natural-sounding speech each time it runs. Activation happens only once someone hits the correct icon on screen.

2.5 Text-to-Sign Pipeline

➢ Text Input Module

○ Starting at the keyboard, someone types a phrase or single word. Once entered, that message moves into software ready to shift languages. Translation kicks off in whatever tongue was typed. From there, words begin changing form - heading toward visual expression. Signs take shape based on what appeared in text. Each language feeds the same signing output path.

➢ Text Processing Module

○ Starting off, the raw text gets split into pieces through tokenization, then cleaned up by normalizing forms. After that comes breaking words apart where needed so each part can be clearly recognized. These processed chunks turn into organized units ready for matching inside a database. Each unit lines up correctly when finding corresponding signs later on.

➢ Word Matching and Dataset Lookup Module

○ From the stored signs, every word after processing gets checked. When a match occurs at the word level, the system looks to see if a video already exists.

➢ Decision and Handling Module

○ Beside each term found in the collection, a matching gesture clip shows up. When present, that visual response plays directly from storage.

○ When a term doesn't show up, off goes the signal to an outside tool that points toward reliable websites - like YouTube or Google - hooking users to clear video examples of the needed signs.

➢ Custom Sign Database Module

○ Now picture someone adding their own video signs for certain words. That clip gets saved right into the system's collection. It fits alongside everything else, no changes needed to the main setup. Personal touches stick around. The whole set grows richer over time because of it..

➢ Sign Video Generation and Output Module

○ Out of the fetched clips, one follows another just like the words in your sentence. This joined-up series of signs then plays on screen when it finishes loading.

2.6 Feedback Mechanism

Once the result appears, users can check accuracy through a response prompt. Feedback follows completion, offering a way to confirm if things went right. A chance to review comes after everything shows up on screen. When it finishes, there is an opening to say whether it worked. The moment output arrives, verification becomes possible through user input.

- When mistakes appear in the result:

- A freshly fixed tag might sit handy later on.

- The feedback data can be:
  - Stored in a dataset for future updates.

3. RESULTS AND EVALUATION

3.1 Performance Metrics and Evaluation

Performance of the model got checked with a fresh batch of data, one it had never viewed while learning. For scoring how well things went, these measures came into play:

1. **Accuracy** –Accuracy gives the share of right guesses compared to total tries. This measure shows up often because it tracks how frequently the model gets things right.[11]

$$\textbf{Accuracy} = \frac{\text{Count of every forecast made}}{\text{Correct Predictions Count}}$$

2. **Precision** – Precision tells you how often the model is right when it says something belongs to the positive class. When incorrect alarms carry heavy risks, this measure matters most. Think of illness detection - flagging a sickness that isn't there might lead to harmful outcomes.[11]

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Where - TP = True positive, FP = False Positive

3. **Recall –** Recall is Instead of just counting wins, it tracks whether the real yes-situations got caught. When letting one slip means big trouble, this number tells the story. Missing an actual yes hits harder than flagging a maybe as yes, so catching those matters more.[11]

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Where - TP = True positive, FN = False Negative

4. **F1-score** – It blends precision with recall using a kind of average that pulls lower values more sharply into view. When both identifying true cases correctly and catching most actual positives matter equally, this measure helps judge performance clearly. High value here signals solid results across those two goals at once. Values stretch from zero up to one, nothing beyond.[11]

$$\textbf{F1-score} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

1. Looking at wrong guesses shows mix-ups happen when signs feel too much alike. What trips up recognition often comes down to subtle overlaps in motion. Mistakes reveal patterns you might miss at first glance. When gestures blur together, errors spike. Spotting these moments helps clarify boundaries that seemed fuzzy before
2. Watch how well the model does on unseen data to avoid learning noise instead of patterns
3. Real-time inference testing to assess latency and responsiveness

Not surprisingly, the tested systems handle both still and moving hand signals well without slowing down. What stands out is how the CNN–LSTM picks up on shifts in movement over time. Meanwhile, using MediaPipe leads to sharp detection when gestures stay fixed in place.

3.2 Results Analysis

3.2.1 Comparative Analysis of Performance Metrics: One look at how these two models stack up shows differences in precision, processing demands, and data flow. Faster response times come from the static setup, thanks to fewer moving parts inside. It handles one image at a time without delay. Even so, when motion is involved, the CNN–LSTM version spots patterns more correctly. Its extra computation doesn't drag things down too much. A solid mix of speed and smarts emerges under real conditions. Precision climbs while resource use stays within reason. For actions that change over time, it holds an edge. Simplicity wins for still inputs. Where movement matters, depth in design pays off.

Table -3.2.1 : Performance Metrics Comparison

Model	Accuracy (%)	Latency (ms)	Throughput (sample/sec)	Model Size (MB)	Frames Used	Real-Time Suitability	Scalability
Static NN (csv)	99.160000	0.003000	333333.333333	0.8	1	High	High
CNN + LSTM (Raw Videos)	94.252874	1694.239792	0.590319	12.5	30	Low	Moderete

3.2.2 Comparison of Static and Dynamic Model Capabilities: One look at the model comparison shows how differently the Static Neural Network works compared to the CNN–LSTM setup. Though the first

handles still images well - picking out hand shapes clearly - it stumbles when signs unfold over time. On the flip side, the CNN–LSTM blends image analysis with pattern tracking across moments. That mix helps catch moving gestures more precisely. Because of this blend, performance stays steady even as motions change speed or form.

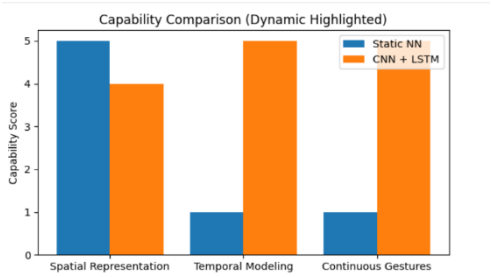

Fig -3.2.1 : Model Capability Comparison

3.2.3 Temporal Gesture Analysis : Every small shift in finger position shows up clearly when tracking movement over time. Motion follows a clear rhythm from one instant to the next. Instead of looking at single moments, the model learns how movements flow by studying what comes before and after. Patterns that seem nearly identical become distinct once timing is part of the picture.

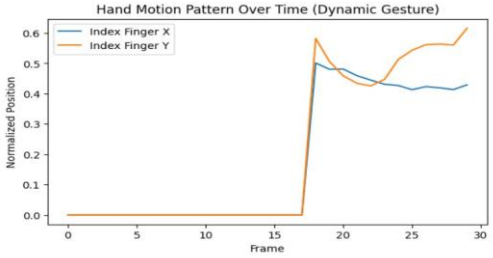

Fig -3.2.2 : Temporal Gesture Analysis

3.2.4 Real-Time Latency Analysis: Although keypoint extraction takes the longest, it's closely trailed by CNN–LSTM steps in timing. Processing splits into phases: pulling keypoints, prepping features, handling sequences, then producing results. Even so, total response time stays low enough for fluid performance. Smooth interactions hold up while translating signs on the fly. The chart tracks these delays as they happen.

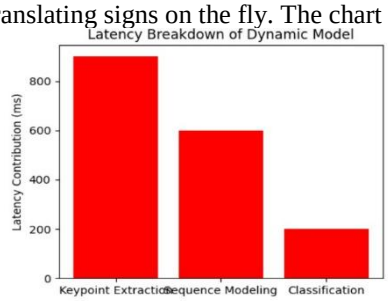

Fig -3.2.3: Real-Time Latency Analysis

3.3 Real-Time Performance

3.3.1 Step1: Right off the bat, the landing page gives a clear look at what SignAI does. This platform uses artificial intelligence to translate into sign language, something shown right up front. Features that matter most are pointed out without clutter. Moving forward, visitors find paths to both Login and Sign Up options built into the layout.

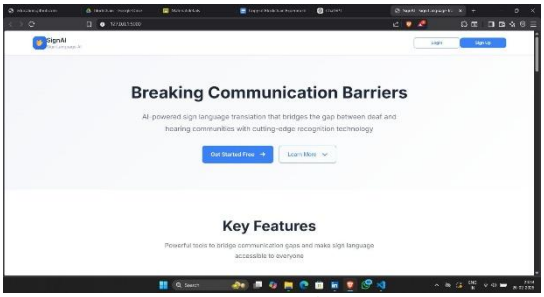

Fig -3.3.1: Landing Page of SignAI System

3.3.2 Step2: The system begins with a simple sign−up and login process where users first create an account and then log in using their credentials. Only after the details are verified are they allowed to access the main

application. This step helps keep user data secure and ensures that each session belongs to an authenticated user.

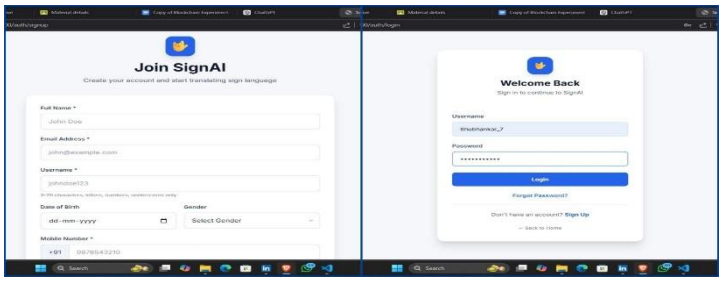

Fig -3.3.2 : User Signup and Login Interface

3.3.3 Step3: Managing your details happens here - username, email, phone number. The system keeps things tailored while locking down access

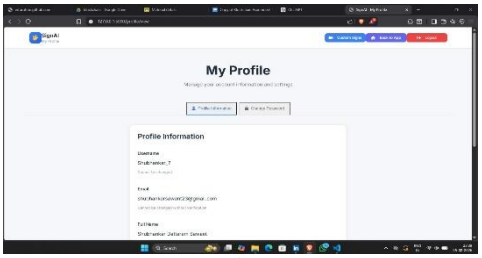

Fig -3.3.3 : User Profile Management Page

3.3.4 Step4: The setup checks it meets specific conditions. Though short ones get rejected right away. When letters mix cases, that helps. Numbers must appear somewhere inside too. Security improves because of these steps taken together.

Fig -3.3.4 : Change Password Interface

3.3.5 Step5: This page shows the full collection of 220 signs you can learn. While browsing them ahead of time helps get familiar with the tool, seeing each one clearly makes practice easier. Getting comfortable with these symbols supports smoother translation later on.

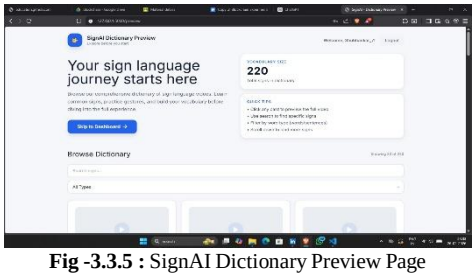

Fig -3.3.5 : SignAI Dictionary Preview Page

3.3.6 Step6: From time to time, you can look up signs using the dictionary tool - each entry sorted by how it's made and where it comes from. This setup shows how gestures are grouped, stored in clear layers behind the scenes.

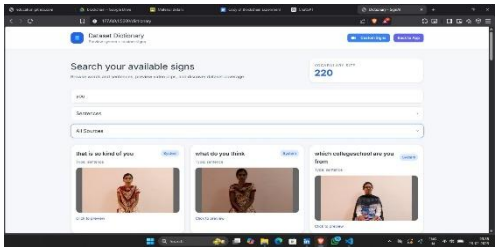

Fig -3.3.6 : Dataset Dictionary with Search and Filters

3.3.7 Step7: From incoming footage, translation appears as text on screen together with how sure the system feels about its guess. That shows it can recognize things live.

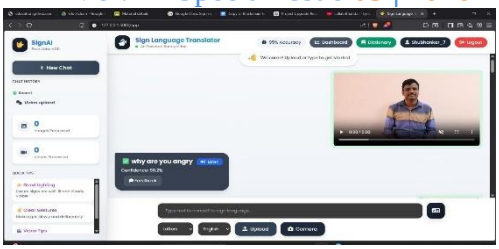

Fig -3.3.7 : Real-Time Gesture Prediction Output

3.3.8 Step8:A single motion starts it all - the camera picks up movement in real time. From each snapshot, vital points emerge after careful analysis. These points travel into a combined network of convolutional and memory-based layers. Recognition happens as patterns shift across moments, seen but never named.

Fig -3.3.8 : Real-Time Camera Capture Module

3.3.9 Step9: Starting with a choice, people pick gestures like letters or whole sentences. Because of that, sorting them gets easier and fits how signs are built.

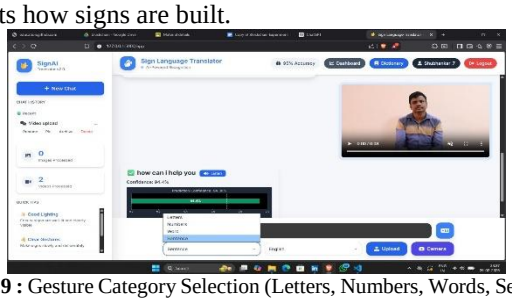

Fig -3.3.9 : Gesture Category Selection (Letters, Numbers, Words, Sentences)

3.3.10 Step10: English sits alongside Marathi, Hindi, Gujarati, Tamil, plus Telugu within the system's language options. Because of this mix, more people can access it easily no matter where they are locat

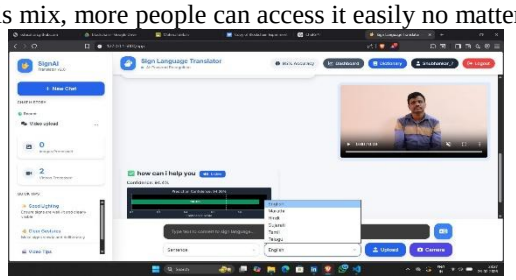

Fig -3.3.10 : Multi-Language Support Feature

4. CONCLUSIONS

A silent gap in conversation finds a solution here - using hand tracking powered by MediaPipe, the system reads **Indian Sign Language** even when offline. Instead of relying on cloud processing, it runs locally with smart tweaks in machine learning to keep delays short and results consistent. Accuracy stays strong because the model focuses only on essential movements, skipping unnecessary data. Rural areas benefit most since stable internet is not required at any stage. Performance holds steady under tough conditions, where many similar tools would struggle.

A twist comes when **CNN** meets **LSTM** - showing solid results in sign language spotting using regular devices, no heavy machinery needed. What stands out? It flips words into signs just as fast as it turns gestures into written form, opening lines between hearing and Deaf worlds alike.

On top of that, speaking multiple languages comes naturally, turning written words into voice without fuss, while learning from user reactions sharpens its fit over time. Built in blocks, the setup grows easily, stays manageable, opens doors later on. Tests show steady performance when things happen live, proving useful support where it counts.

When it comes to talking across differences, this work adds something real. Tied closely to worldwide efforts such as the UN's push for better schooling and fairer societies, it quietly supports goals about learning and equality. Deaf people gain more control over daily life through these findings. At the same

time, others start noticing gaps they never saw before. Awareness spreads without force, simply by showing what change looks like up close.

Though it handles still gestures fine, motion tracking needs work along with full-sentence interpreting. Next steps include reading face cues, speaking more languages, running on phones. Better versions could reach further, helping deaf users talk in ways now out of reach.

There are still areas that can be improved. Dynamic gesture recognition can be made more accurate, and full sentence interpretation needs further development. Future work can explore facial expressions, mobile deployment, and support for more languages. With these improvements, the system has the potential to become even more useful and reach a wider group of users.

5. ACKNOWLEDGEMENT

First off, huge thanks go to Professor **Madhura Zagade** from the Information Technology department at **Finolex Academy of Management and Technology** - her steady support, sharp ideas, plus constant push made a real difference during this project's journey. On another note, deep appreciation flows toward **Mrs. Vaidehi Hindlekar**, whose role as both Technical Product Manager and industry guide brought clarity; her observations shaped how well the work holds up in actual use.

Thanks go to the teachers at **Z.P. Full Primary School Lavgan No.1** - their help while gathering information made it possible to build the unique word-by-word sign language set used here. Without their time and effort, putting together this material would have been much harder. Their role was quiet but essential throughout the process. What they offered shaped how the data turned out in ways that matter. This work stands on small actions done by real people each day. Not grand gestures, just steady presence when needed most.

Happy to thank the Department of Information Technology, FAMT - without their tools and space for work, progress would have stalled early. Help came in many forms, some clear, others quiet; classmates lent time, teachers offered guidance, each small part adding up where it mattered most.

6. REFERENCES

[1] R. Sivasankar, D. Dharshan, M. Subiksha, G. Jennifer, and R. Rasika, “Real-Time Bidirectional Sign Language Translator Through Machine Learning,” *International Journal of Engineering Research & Technology (IJERT).*

[2] H. Orovwode, I. D. Oduntan, and J. Abubakar, “Development of a Sign Language Recognition System Using Machine Learning,” in *2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD),* 2023.

[3] A. Ghorai et al., “Tsi-cnn-net: Truly Shift-Invariant Convolutional Neural Network for Indian Sign Language Recognition System,” *Pattern Analysis and Applications,* 2025.

[4] M. Alaftekin, I. Pacal, and K. Cicek, “Real-Time Sign Language Recognition Based on YOLO Algorithm,” *Neural Computing and Applications,* 2024.

[5] I. Bulugu, “Tanzanian Sign Language Recognition System for an Assistive Communication Glove Sign Tutor Based on the Inertial Sensor Fusion Control Algorithm,” *Journal of Electrical Systems and Information Technology,* 2025.

[6] S. K., A. Sree Lakshmi, Sri Geethika, and B. Kulkarni, “Indian Sign Language Character Recognition,” *IOSR Journal of Computer Engineering (IOSR-JCE),* 2020.

[7] GeeksforGeeks, “What is a Neural Network?,” Dec. 16, 2025.

[8] GeeksforGeeks, “Convolutional Neural Network (CNN) in Machine Learning,” Dec. 17, 2025.

[9] GeeksforGeeks, “What is LSTM – Long Short Term Memory?,” Dec. 23, 2025.

[10] H2O.ai, “Confusion Matrix,” *H2O Wiki.*

[11] GeeksforGeeks, “Evaluation Metrics in Machine Learning,” Oct. 29, 2025.

[12] Google Drive Dataset Repository.

[13] “Alphabet (Indian Sign Language),” *YouTube Video,* duration 6:49, Dec. 23, 2020.

[14] Google, “MediaPipe Solutions Guide,” *Google AI Edge,* accessed Feb. 25, 2026.