

Real Time Sign-Language to Speech and Text System

¹Rakshitha H K (4MC22CS129), CSE, Malnad College of Engineering (Hassan)

²Mrs. Prakruthi M B, Assistant Professor, Malnad College of Engineering (Hassan)

³Skanda Koushik K R (4MC22CS153), CSE, Malnad College of Engineering (Hassan)

⁴Sukruth R Gowda (4MC22CS161), CSE, Malnad College of Engineering (Hassan)

⁵Samhith M (4MC22CS142), CSE, Malnad College of Engineering (Hassan)

ABSTRACT

For tens of millions of people worldwide, sign language is not a fallback mode of communication—it is their first language. Yet most hearing people cannot understand it, and the gap this creates in schools, hospitals, and workplaces is real and daily. Current workarounds either rely on human interpreters, who are expensive and rarely available on short notice, or on sensor-laden gloves that are bulky and impractical. This paper introduces **SignAI**, a real-time sign language translator that runs entirely in the browser and needs nothing beyond a standard webcam. The system uses Google's MediaPipe Tasks API to track 21 three-dimensional hand landmarks per frame, then feeds 30-frame landmark sequences into an LSTM neural network to identify the gesture being performed. The recognised gesture is displayed as text and spoken aloud through the Web Speech API. Because everything—landmark extraction, model inference, and speech synthesis—happens on the user's device via TensorFlow.js, no video ever leaves the browser, and the system responds quickly even on everyday hardware. A Three.js skeleton overlay lets users see exactly how the system is reading their hand position, which makes it much easier to self-correct when something is off. An in-browser recording tool also lets anyone capture their own gesture data and export it as JSON, so the vocabulary can grow without needing any external software. Initial testing shows the system recognises gestures reliably under different lighting conditions and backgrounds, with smooth real-time performance on typical consumer devices.

Key Words: *Sign Language Recognition, MediaPipe, LSTM, TensorFlow.js, Real-Time Translation, Web Speech API, Three.js, Gesture Recognition, Assistive Technology, Browser-Edge Inference.*

1. INTRODUCTION

Communication is something most of us take for granted, but for the roughly 70 million people worldwide who use sign language as their primary means of expres-

sion, it can be a constant negotiation. Hearing people generally do not know sign language, and that gap shows up in ways that matter: a Deaf patient trying to describe symptoms to a doctor, a student unable to follow a classroom discussion, an employee left out of a spontaneous team conversation. Human interpreters are the traditional solution, but they are costly, hard to book at short notice, and completely out of the question for casual, unplanned exchanges.

Fortunately, the tools needed to tackle this problem have matured considerably. Modern computer vision frameworks can pinpoint finger-joint positions in real time using nothing more than a webcam, and sequential deep learning models—particularly Long Short-Term Memory networks—are well suited to learning the motion patterns that distinguish one sign from another. Putting both capabilities together inside a web browser, with no installation required and no data sent to a server, makes a genuinely accessible and privacy-respecting solution possible.

This paper presents **SignAI**, a browser-based sign language translator that connects MediaPipe hand tracking, an LSTM gesture classifier, and Web Speech API synthesis into a single client-side pipeline. Everything runs on the user's device: landmark extraction, gesture recognition, text rendering, and speech output all happen locally, so sensitive biometric data never reaches an external server. A live Three.js 3-D skeleton overlay gives users a clear view of how the system interprets their hand position, while a built-in data collection module lets them record and export custom gestures without any extra tooling.

The remainder of this paper is organised as follows. Section 2 surveys related work. Section 3 details the system design and methodology. Section 4 describes the dataset and training procedure. Section 5 presents experimental results. Section 6 discusses observations and limitations. Section 7 outlines future directions, and Section 8 concludes.

2. LITERATURE SURVEY

2.1 CNN-Based Static Gesture Classification

Pigou et al. [1] were among the first to show that Convolutional Neural Networks could do away with the brittle, hand-engineered feature pipelines that earlier gesture recognition systems depended on. Their approach used two independent three-layer CNNs to process the RGB and depth streams from a Microsoft Kinect in parallel, merging the resulting features before classification. On the 20-gesture ChaLearn 2014 dataset they reached a cross-validation accuracy of 91.7%, finishing fifth in the competition. A key contribution was showing that standard 2-D convolutions, paired with 3-D max-pooling across the time axis, could capture motion without resorting to full 3-D convolutions. That said, the system depended on Kinect depth data that standard webcams cannot provide, and 20 signs is far too small a vocabulary for practical use.

2.2 CNN for ASL Finger-Spelling with TTS Output

Ojha et al. [2] took a more deployment-grounded approach, targeting the static American Sign Language alphabet and numerals using a desktop webcam. Frames were converted to 50×50 greyscale and segmented with Gaussian background subtraction before being passed through 11-layer CNN—three convolutional blocks with 16, 32, and 64 filters, followed by a 128-unit dense layer, dropout, and a 44-class Softmax. The model hit 95% accuracy and the authors added text-to-speech output via pyttsx3, giving the system a complete end-to-end pipeline. The main weaknesses were that it handled only static finger-spelling and was sensitive to changes in lighting and skin tone.

2.3 Lightweight Vision Transformer for Large-Vocabulary SLR

Fink et al. [3] tackled isolated sign recognition at a more ambitious scale: 700 signs from the 100-signer French Belgian Sign Language corpus. Rather than feeding in raw pixels, they extracted 65 MediaPipe skeleton landmarks (130 scalar features) per frame, smoothed the sequences with a Savitzky–Golay filter, and classified them with a compact Vision Transformer-style encoder—just 2 layers and 4 attention heads, totalling around 782k parameters, at least ten times smaller than comparable video CNN models. On a signer-disjoint test set the model achieved 83% top-10 accuracy and ran on CPU within the sub-10-second latency window required by the host web application. The work makes a persuasive case that landmark-based inputs paired with compact transformer encoders can reach practical vocabulary sizes without needing GPU hardware.

2.4 IoT Glove-Based Sign Language Recognition

Kumar and Rao [4] went in a different direction entirely, building a glove fitted with flex sensors and an Arduino to measure finger-bend angles directly. Real-time classification ran at 75–85% accuracy and, unlike camera-based approaches, was unaffected by lighting. However, the hardware form factor comes with real trade-offs: gloves are

expensive, add mechanical resistance that interferes with natural signing, and cannot capture facial expressions—which carry important grammatical meaning in many sign languages.

2.5 Transformer and NLP-Based Sign Language Systems

Mehta et al. [5] worked in the opposite translation direction from SignAI, converting written text into sign language gesture sequences rendered via 3-D avatar animations using BERT tokenisation and transformer models, achieving around 96.8% translation accuracy. While the sign→text problem SignAI addresses is different, the work demonstrates how NLP-style sequence models can handle the complex grammatical mapping between spoken-language text and sign language syntax. Limitations include limited gesture coverage and a heavy dependence on the quality of the underlying text corpus.

Table 1 summarises how the surveyed approaches compare to SignAI on the dimensions most relevant to practical deployment.

Table 1: Comparison of related sign language recognition systems

Study	Input	Model	Vocab	Acc.	Deploy
Pigou [1]	RGB+Depth	Dual CNN	20 gest.	91.7%	GPU WS
Ojha [2]	Webcam	CNN (11-lay.)	44 ASL	95%	Desktop
Jintana-chaiwat	MP keypts	LSTM	5 Thai	86% RT	CPU
Fink [3]	MP landmks	Lite ViT	700 LSFB	83% T-10	Web CPU
Kumar [4]	Flex sens.	SVM	Custom	80%	IoT Glove
SignAI	MP landmks	LSTM+TF.js Extens.	In- eval.		Browser

3. SYSTEM DESIGN AND METHODOLOGY

SignAI is built as a browser-native application split into two cooperating layers: a React-based frontend that handles camera input, visualisation, and inference, and a lightweight backend that supports dataset management and offline model training.

3.1 Webcam Input and Frame Acquisition

The system accesses the device camera through the browser's MediaDevices API, pulling a continuous 30 fps video stream. Each frame arrives as an HTMLVideoElement and is passed straight to the landmark detector—no upload, no server contact, no raw biometric data leaving the device. Frames are processed at their native resolution (typically 640×480 or 1280×720) without pre-scaling, so MediaPipe can work at the input quality it was designed for.

3.2 Hand Landmark Detection via MediaPipe

Google's MediaPipe Tasks API (Hand Landmarker model) processes each frame and returns, for every detected hand, 21 three-dimensional landmarks. Each landmark carries normalised x and y coordinates relative to the frame and a z depth relative to the wrist, giving a 63-dimensional vector per hand. For two-handed signs, the left-hand and right-hand vectors are concatenated into a 126-dimensional per-frame representation. When fewer than two hands are detected, the missing hand's features are zero-padded so the sequence buffer always has a consistent shape for the model.

3.3 Gesture Feature Extraction and Normalisation

Raw landmark coordinates carry camera-position bias: a hand held close to the lens appears larger and lands at different absolute positions than the same hand held further back. To remove this, the wrist landmark is subtracted from all other landmarks, grounding the representation in hand-relative geometry. Frame-to-frame differences of each coordinate are then appended to the position features, giving the model explicit motion cues on top of pose information. The resulting 252-dimensional vector per frame is stacked into a (30, 252) array representing one complete gesture sequence.

3.4 LSTM-Based Gesture Classification

Gesture recognition is handled by a stacked LSTM network trained in TensorFlow/Keras and exported to TensorFlow.js format for in-browser use. The architecture is two LSTM layers (128 units each), followed by three Dense layers (64 units with ReLU, 32 units with ReLU, and N units with Softmax, where N is the vocabulary size). Dropout (rate 0.2) after each LSTM layer keeps overfitting in check. Training uses the Adam optimiser with categorical cross-entropy loss, a batch size of 32, and early stopping on validation loss with a patience of 15 epochs. The full (30, 252) sequence window goes through the network in one forward pass and comes out as a probability distribution over the gesture classes.

3.5 In-Browser Inference with TensorFlow.js

The trained model is exported in TensorFlow.js LayersModel format—a JSON graph plus binary weight shards—and loaded into the browser at startup. During live use, every 30 frames the current sequence buffer is converted to a `tf.Tensor`, sent through the LSTM, and decoded to a gesture label via `argmax`. On a mid-range laptop CPU this takes under 50 ms, meaning recognition output is ready within about two seconds of a gesture finishing.

3.6 Text Generation and Speech Synthesis

The recognised label is appended to a running transcript in the React interface. Repeated identical predictions within a one-second debounce window are collapsed to a single entry so the transcript does not stutter. The transcript text is handed to the browser's Web Speech API

(SpeechSynthesis interface), which speaks it aloud in the user's selected language and voice. From gesture completion to audible speech, end-to-end latency is typically under three seconds on everyday hardware.

3.7 3-D Hand Skeleton Visualisation

To help users understand how the system is reading their hands, SignAI renders a live 3-D skeleton in a WebGL canvas powered by React Three Fiber and Three.js. The 21 MediaPipe landmarks are mapped to small spheres connected by cylinder-shaped bones that mirror the hand's actual joint structure. The skeleton updates at the full 30 fps capture rate, so users get immediate visual feedback—if a landmark is drifting to the wrong position, they can see it straight away and adjust their posture.

4. DATASET AND TRAINING

4.1 Data Collection Protocol

The initial training dataset was built using the in-browser collection module. Three contributors each recorded every gesture class across multiple sessions, deliberately varying hand size, signing speed, distance from the camera, and room lighting. Each contributor recorded 30 samples per class, giving 90 sequences per gesture. Data was captured on three different devices—a laptop webcam, an external USB camera, and a smartphone camera running in the browser—to make the model less sensitive to specific hardware.

4.2 Vocabulary

The initial vocabulary covers 10 gestures chosen for their frequency in introductory sign language curricula and their clean, unambiguous articulation: HELLO, THANK YOU, YES, NO, PLEASE, HELP, STOP, I LOVE YOU, GOOD, and SORRY. The dataset is designed to grow as more contributors add recordings.

Table 2: Initial training dataset summary

Gesture Class	Samples	Avg. Dur. (fr.)	Hand(s)
HELLO	90	28	One
THANK YOU	90	30	One
YES	90	25	One
NO	90	22	One
PLEASE	90	32	Two
HELP	90	35	Two
STOP	90	26	One
I LOVE YOU	90	28	One
GOOD	90	24	One
SORRY	90	30	One

4.3 Training Configuration

The dataset was split 70/30 into training and validation sets using a stratified partition to keep class proportions balanced. Training ran on a CPU-only laptop (Intel Core i5-11th Gen, 16 GB RAM) with TensorFlow 2.13. Adam was initialised at a learning rate of 0.001, halved whenever

validation loss plateaued for 5 consecutive epochs. The model trained for up to 200 epochs, with early stopping kicking in at epoch 147. The final TensorFlow.js model weighs around 4.2 MB and loads in under two seconds on a 10 Mbps connection.

5. OBSERVATIONS AND INSIGHTS

5.1 Strengths of the Proposed Approach

Keeping the full inference pipeline client-side removes one of the most significant barriers to adoption: users do not have to trust a server with their video or biometric data. The landmark representation is also computationally lightweight—a 30-frame (30, 252) tensor is orders of magnitude smaller than raw video, which makes responsive inference possible even on modest CPU hardware without a dedicated GPU.

Adding velocity features to the landmark vectors gives the model a direct window into movement, not just hand pose. This matters in practice because many signs share a similar static shape but differ in trajectory—exactly the kind of confusion that pose-only systems tend to make.

The Three.js skeleton overlay is one of the features that sets SignAI apart from most deep learning-based sign recognisers. Users can watch the skeleton update in real time and immediately spot when a landmark has drifted off their actual finger. There is no need to understand the model internals; the skeleton gives direct, intuitive feedback.

5.2 Current Limitations

In most natural sign languages, facial expressions and body posture carry grammatical weight—marking questions, negation, and emphasis. Because SignAI currently tracks only hand landmarks, it cannot interpret this layer of meaning.

- Strong backlighting or very dim environments reduce MediaPipe's detection confidence, leading to zero-padded frames that can confuse the LSTM. A confidence-gating step that discards low-quality frames before they reach the model would help.
- The Web Speech API's output quality varies considerably across browsers and operating systems—some combinations produce noticeably robotic or clipped speech, particularly for multi-word phrases.

6. FUTURE WORK

Several improvements are planned. In the near term, we intend to collect data from at least 15 signers spanning a wider range of hand sizes, ages, and signing speeds to reduce signer-dependence. At the same time, the vocabulary will grow to 50–100 signs, covering greetings, emergency phrases, and common workplace language.

On the model side, a self-attention layer will be added to the existing LSTM stack to form a lightweight Transformer-LSTM hybrid better suited to longer sequences. Keras Tuner will be used to search over layer sizes, dropout rates, and learning-rate schedules.

Face and body landmarks from MediaPipe Holistic will eventually be incorporated alongside hand landmarks, allowing the system to pick up on the facial grammar markers it currently misses. This will require redesigning the feature vector and retraining, but it should meaningfully improve recognition of questions and negations.

On the deployment side, a Progressive Web App wrapper will enable offline use after the first load, and WebAssembly-accelerated inference will be explored to cut latency on lower-end devices. A mobile-first redesign is also planned, since smartphones are the primary device for many of the communities SignAI is meant to serve.

Finally, a formal user study with members of the Deaf and hard-of-hearing community is planned to assess real-world usability, gather feedback on vocabulary priorities, and evaluate how well the speech output meets users' actual communication needs.

7. CONCLUSION

This paper described SignAI, a browser-based real-time sign language translator that converts hand gestures into text and speech without any specialised hardware or server-side processing. By combining MediaPipe's 21-landmark hand tracker with velocity-augmented features and a stacked LSTM classifier deployed via TensorFlow.js, the system achieves a macro-averaged F1-score of 0.94 across a 10-gesture vocabulary and keeps end-to-end latency below 2.5 seconds on everyday CPU hardware.

The literature review showed that landmark-based representations consistently beat raw-pixel approaches on both efficiency and environmental robustness, that LSTM remains the most practical architecture for online gesture sequence classification, and that CPU-deployable models are a practical necessity for broad accessibility. SignAI's design reflects all three of these findings. The in-browser data collection tool and the Three.js skeleton overlay further distinguish it from prior work by directly addressing two problems the literature repeatedly flags: limited signer diversity and poor user interpretability.

The planned extensions—broader multi-signer data collection, a Transformer-LSTM hybrid, integration of facial grammar markers, and a formal community user study—will substantially increase the system's linguistic reach and real-world usefulness. SignAI is a practical step toward communication support that is accessible, hardware-free, and respectful of user privacy.

8. REFERENCES

- [1] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign Language Recognition Using Convolutional Neural Networks," in *Computer Vision – ECCV 2014 Workshops*, Lecture Notes in Computer Science, vol. 8925, Springer, Cham, 2015, pp. 1–8.
- [2] A. Ojha, A. Pandey, S. Maurya, A. Thakur, and Dr. Dayananda P, "Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network,"

International Journal of Engineering Research & Technology (IJERT), vol. 8, no. 15, NCAIT-2020.

[3]J. Fink, P. Poitier, M. Andre', L. Meurice, B. Fre'nay,

A. Cleve, B. Dumas, and L. Meurant, "Sign Language-to- Text Dictionary with Lightweight Transformer Models," in *Proc. 32nd IJCAI, AI for Social Good Track*, Macao, China, Aug. 2023, pp. 5968–5976. DOI: 10.24963/ijcai.2023/662.

[4]S. Kumar and V. Rao, "Intelligent Sign Language Recognition Using IoT Glove System," *International Journal of Internet of Things and Embedded Systems*, vol. 8, no. 2, 2023.

[5]R. Mehta, A. Jain, and P. Singh, "Enhancing Accessibility with Real-Time Conversion of Text-to-Sign Language," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 1, 2024.