International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Gesture2Speech: AI-Powered Sign Language Interpretation System Using MediaPipe, Random Forest and YOLOv11n

Hrittik M[1], Jayanth K[2], Mahesh Kumar G[3], Mithun N[4], Hemalatha M[5]
[1,2,3,4,5]Department of Computer Science and Engineering, Don Bosco Institute of Technology, Visvesvaraya Technological University, Bengaluru, India

---

Abstract - Sign language is the primary mode of communication for many people with hearing and speech impairments, yet the lack of real-time interpretation tools limits their interaction with non-signers. This paper presents Gesture2Speech, a real-time, web-based sign language interpretation system that recognizes alphabets and word gestures and converts them into text and speech. The proposed pipeline captures webcam frames in the browser, streams them over a WebSocket to a FastAPI backend, and applies two complementary recognition paths: MediaPipe hand landmarks with a Random Forest classifier for alphabets, and a YOLOv11n detector trained on 58 classes (26 letters and 32 word gestures) for words. Temporal stabilization based on consecutive-frame agreement and sliding-window majority voting suppresses frame-level noise. In the video-call mode, confirmed words are assembled into grammatical sentences by a large language model (LLM) and spoken through text-to-speech, while a reverse Text-to-Sign module plays real ASL clips from the WLASL vocabulary. On held-out validation data, the Random Forest reached 94.2% accuracy for ASL letters, and YOLOv11n reached 93.4% mAP@0.5, 78.6% mAP@0.5:0.95, 87.5% precision and 91.2% recall. The results indicate that the system is a practical baseline for bidirectional, real-time sign language communication.

Key Words: sign language recognition, MediaPipe, Random Forest, YOLOv11n, real-time detection, text-to-speech, text-to-sign, WebRTC, assistive technology.

1. INTRODUCTION

Sign language is a vital mode of communication for people with hearing and speech impairments. However, most people in everyday situations do not understand it, and human interpreters are not always available. The lack of real-time interpretation tools therefore limits the interaction of the deaf and hard-of-hearing community with the general population in settings such as education, healthcare, shopping, and online meetings.

Computer vision has made hand-gesture recognition practical on ordinary hardware. Google's MediaPipe Hands provides on-device, real-time tracking of 21 hand landmarks from a single camera [1]. Such landmarks can be classified with lightweight machine-learning models such as Random Forest [2], whereas single-stage object detectors from the YOLO family [3, 4] can locate and classify a whole gesture in one forward pass. Large datasets such as WLASL [5] have also made word-level American Sign Language (ASL) video available for research and for sign generation.

Despite this progress, many existing systems address only one part of the problem. They focus on isolated alphabets, are evaluated offline, and are not packaged into an application that a non-expert can use. They also support only one direction of communication, from signs to text, and rarely close the loop for the hearing person who wishes to reply in sign.

This work presents Gesture2Speech, a unified web application that connects these pieces. It recognizes alphabets with MediaPipe landmarks and a Random Forest classifier and recognizes 32 common word gestures with a YOLOv11n detector. A FastAPI backend processes browser frames received over WebSocket, and temporal stabilizers decide when a prediction is committed. In the video-call mode, confirmed words are organized into a sentence by an LLM and spoken aloud, using WebRTC for the call [6]. A reverse Text-to-Sign module converts typed text into ASL video.

The main contributions of this work are:

1) A dual-path recognizer that combines landmark-based alphabet classification with YOLOv11n word detection;
2) Debouncing strategies, namely a consecutive-frame streak for words and a sliding-window majority vote for letters, that stabilize live predictions;
3) A human-in-the-loop video-call pipeline that turns detected words into a spoken sentence;
4) A Text-to-Sign module built on a deterministic WLASL clip lookup; and
5) Evaluation with accuracy, mAP, precision and recall.

2. RELATED WORK

Recent research on sign language recognition spans object detectors, transformers, and hybrid sequence models. Alsharif *et al.* [7] combined YOLOv11 with MediaPipe hand tracking for ASL interpretation and reported a mean average precision of 98.2% with real-time inference. Their system, however, is largely limited to the alphabet and struggles to separate visually similar signs. Rastogi *et al.* [8] proposed YOLOv10-ST for Indian Sign Language, which adds Swin Transformer blocks and Mish activation to YOLOv10 so that both local and global spatial dependencies are captured. The dynamic-video portion of their data is small, which restricts how far the results generalize.

Transformer-based models target temporal structure. Shahin and Ismail [9] introduced ADAT, an adaptive transformer that uses LogSparse self-attention and an adaptive gating mechanism to model short- and long-range dependencies in sign language translation, although its dual-stream design can be up to 12.1% slower than simplified encoder-only baselines. Alkhorafi *et al.* [10] fused appearance (Video Swin) and pose (Image Swin) streams in an ensemble for word-level recognition and obtained state-of-the-art results on large

benchmarks, at a computational cost that grows with the gloss set.

Pose-based approaches reduce this cost. Alyami *et al.* [11] paired MediaPipe hand and face keypoints with a Transformer for isolated Arabic sign recognition, achieving a lightweight model suitable for mobile use, but one whose performance depends on the quality of the pose estimator and which handles only isolated signs. Kaliyaperumal and Gopalan [12] combined BERT with an LSTM for word-level Indian Sign Language recognition and reached 95% accuracy, although accuracy drops for complex and dynamic gestures. Zhang and Jiang [13] reviewed deep-learning techniques including CNNs and recurrent networks (LSTM, GRU) and summarized recent trends and datasets.

These studies show three recurring gaps: coverage is often limited to alphabets or isolated words, evaluation rarely extends to live use such as video calls, and few systems provide a text-to-sign direction. Gesture2Speech addresses these gaps with a lightweight, deployable pipeline that supports alphabets, words, sentences, and both directions of communication.

3. PROPOSED SYSTEM

A. System Objective

Gesture2Speech interprets hand gestures captured by an ordinary webcam and presents them as text and speech, and it lets a hearing user reply with sign video. The system is built from fast, lightweight components so that it runs in real time on commodity hardware. Its objective is not to claim universal sign language understanding, but to provide a measurable, deployable pipeline for a fixed vocabulary of 26 letters and 32 common word gestures.

B. Processing Pipeline

The system consists of four independent pipelines served by one backend, as shown in Fig. 1: alphabet recognition, word detection, sign-to-speech video call, and text-to-sign. The browser opens the camera at 640×480 and, every 200 ms (about 5 frames per second), copies one frame to a 320-pixel-wide canvas, encodes it as JPEG at quality 0.4, and sends the base64 string over a persistent WebSocket. A persistent connection is used because one HTTP request per frame would add several hundred milliseconds of overhead.

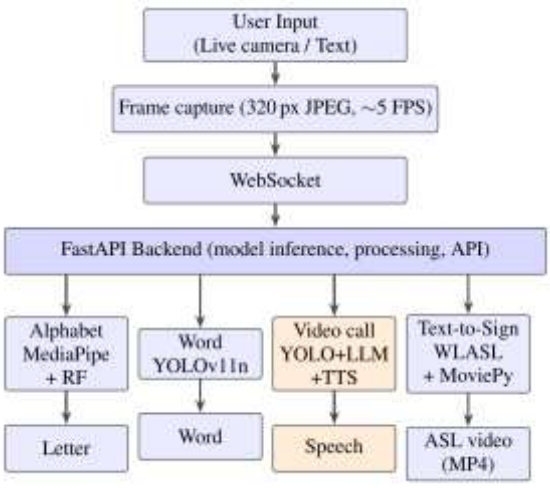


Fig -1: Overall architecture of Gesture2Speech.

On the backend, each frame is decoded with OpenCV and passed to the appropriate model. Inference runs in a worker thread pool so that blocking model calls never stall the event loop, and an undecodable frame simply returns an empty result instead of closing the socket. Prediction results are returned as JSON containing the top symbol, its confidence, and the top three candidates.

C. Alphabet Recognition

For alphabets, MediaPipe HandLandmarker [1, 22] detects a hand and returns 21 keypoints (x, y, z), giving a 63-dimensional vector. Raw coordinates depend on where the hand appears in the frame and how large it is, so each landmark p_i is normalized with respect to the wrist landmark p_0:

$$\hat{p}_i = \frac{p_i - p_0}{\max_j \|p_j - p_0\|} \quad (1)$$

Subtracting the wrist makes the features position-invariant, and dividing by the maximum wrist distance makes them size-invariant. Without this step the model fails on webcam data, because the hand lies in a different position than in the training images.

The 63 normalized values are classified by a Random Forest [2] of 300 trees with balanced class weights, trained on an 80/20 stratified split. Balanced weights compensate for class imbalance in the data, where one letter had only 3 samples against 66 for another. The same landmark space is used for five sign languages (ASL, BISINDO, DGS, LIBRAS and TİD), with one forest per language selected at connection time. The probabilities from `predict_proba` are sorted to give the top-3 letters.

D. Word Gesture Recognition

Words are recognized with YOLOv11n, the nano variant of the YOLO family with about 2.6 million parameters, chosen to keep latency low [3, 4]. In a single forward pass at 416 × 416 the detector locates the gesture with a bounding box and assigns one of 58 classes: letters A–Z (class ids 0–25) and 32 word gestures (ids 26–57). Detections are sorted by confidence and de-duplicated by class id to give the top three, and ids of 26 or above are treated as words. A confidence threshold of 0.25 is used on the detector page.

E. Temporal Stabilization

Frame-level predictions flicker, so two stabilizers are applied. For words, a prediction is committed only after the same word appears in four consecutive frames with confidence of at least 0.50, and the display then holds for 1.2 s so that the signed word remains readable. For letters, a sliding window of the last eight predictions is kept, and a letter is committed only if at least six of the eight agree, followed by a 1.5 s cooldown that prevents the same letter from repeating. Both rules are summarized in Algorithm 1.

Algorithm -1: Temporal stabilization of predictions

Word path (per frame with word w, confidence c)
1: if c ≥ 0.50 and w equals the previous word then k ← k + 1
2: else k ← 1 if c ≥ 0.50, otherwise k ← 0
3: if k = 4 then commit w and hold the display for 1.2 s
Letter path (buffer B of the last 8 predictions)
1: append the new letter to B, dropping the oldest entry
2: if some letter occurs at least 6 times in B and no cooldown is active
3: commit that letter and ignore input for 1.5 s

F. Sign-to-Speech Video Call

In the video-call mode, the signer's browser streams frames to the backend, which runs the same YOLOv11n detector but keeps word detections only and ignores letters. Unlike the detector page, no confidence threshold is applied on the server: every per-frame detection is streamed live, and the browser removes consecutive duplicates and shows the words as editable chips. Real sentences contain natural pauses, and an aggressive automatic threshold was found to silently drop words, so the decision to commit is left to the user, who can remove a chip, clear the list, or press Done.

When the user confirms, the word list is sent to an LLM through the OpenRouter chat-completions interface, with a fallback chain of five free models. The prompt asks the model to act as a grammar engine, for example turning the tokens "I LoveYou" and "Juice" into "I love juice." A low temperature of 0.3 and a limit of 120 tokens keep the output short and stable. Because free models sometimes reply with their reasoning, a parser keeps only the last line of the response and removes surrounding quotes. The sentence is shown as a subtitle, spoken with the browser Web Speech API, and relayed to the viewer over the peer data channel.

Media is carried by WebRTC through PeerJS, which provides hosted signaling and a direct peer-to-peer audio, video and data channel [6]. The backend therefore sees only detection frames, and audio, video and subtitles never pass through the server.

G. Text-to-Sign Module

The reverse direction lets a hearing user answer in sign. The typed text is lower-cased and split with the regular expression `[a-zA-Z']+`, so punctuation is removed and no heavy language processing is needed. Each token is looked up in a WLASL-derived vocabulary [5]: an in-vocabulary word maps to a real ASL clip, whereas an out-of-vocabulary word is rendered as a text segment. MoviePy then resizes every clip to 320×240 at 25 fps, because the source clips differ in resolution and frame rate, and joins them with `concatenate_videoclips` into a single H.264 MP4 without audio. Text segments use adaptive font sizing, starting at 96 px and shrinking in 4-px steps until the word fits within 85% of the frame width. If a clip fails to load, its word is shown as text, so a word is never silently dropped. This module is a deterministic lookup rather than a learned model, and its output is a guaranteed rendering rather than a translation.

4. DATASET AND EXPERIMENTAL SETUP

A. Datasets

The detector is trained on a 58-class dataset assembled from two sources: letters A–Z (class ids 0–25) from the Roboflow ASL Letters dataset [14], which is already labeled, and 32 word gestures (ids 26–57) from a public Kaggle collection of about 500 images per word [15]. The word images carry no bounding boxes, so MediaPipe was used to find each hand and write the box automatically, with 5% padding. The word classes are Alcohol, Bad, Bed, Beg, Coffee, Come, Confused, Dont like, Egg, Fries, Good Bye, Hello, Help, How Are u, I LoveYou, Icecream, Juice, Like, Mango, My Pleasure, No, Please, Sad, Soda, Sorry, Standup, Strawberry, Thanku, Toffee, Wait, Water and Welcome.

To avoid a detector that favors over-represented classes, the word classes were undersampled to about 100 images each while the alphabet classes kept their original counts. The final split contains about 4,700 training, 728 validation and 600 test images. For the Random Forest, landmark vectors were extracted from still images for five sign languages: ASL, BISINDO (Indonesian), DGS (German), LIBRAS (Brazilian) and TİD (Turkish), each with an 80/20 stratified split.

B. Training Configuration

YOLOv11n was trained for 50 epochs at an input size of 416×416 with batch size 16, mixed precision and early stopping with a patience of 15 epochs, on an NVIDIA RTX 2050 GPU with 4 GB of memory. The optimizer was selected automatically (AdamW) with an initial learning rate of 0.01. An input size of 416 was used instead of 640 so that training fits in 4 GB. Augmentation imitates webcam conditions: hue, saturation and value shifts of 0.015, 0.7 and 0.4, rotation of ±15°, scale 0.3, translation 0.1, horizontal flip 0.5 and mosaic 0.5. Vertical flipping is disabled because hands have a natural orientation and upside-down samples would teach wrong shapes. Table 1 summarizes the setup.

Table -1: Training configuration

Parameter	Value
Detector	YOLOv11n (~2.6M parameters)
Classes	58 (26 letters + 32 words)
Train / Val / Test	~4,700 / 728 / 600
Input size, batch	416×416, 16
Epochs, patience	50, 15
Optimizer, lr0	AdamW (auto), 0.01
Letter classifier	Random Forest, 300 trees
Landmark features	63 (21 points × xyz)

C. Implementation Stack

The frontend is a React application written in TypeScript and built with Vite, with pages for alphabets, word detection, video call and sign learning. The backend is a FastAPI service [16] that loads all models once at startup: the YOLOv11n weights (5.2 MB), five Random Forest classifiers built with scikit-learn [17], and the MediaPipe hand landmarker. Frames are decoded with OpenCV [18], Text-to-Sign videos are assembled with MoviePy [19], sentences are requested through OpenRouter [21], and calls use PeerJS [20] on top of WebRTC. A same-origin development proxy forwards the WebSocket and API routes to the backend. One deployment detail is worth noting: the proxy target must be the IPv4 address `127.0.0.1` rather than `localhost`, because on Windows the name can resolve to the IPv6 address while the server listens on IPv4, which makes connections fail silently.

D. Evaluation Metrics

The Random Forest is evaluated on its held-out split using accuracy, and the detector is evaluated on the validation split using precision, recall and mean average precision (mAP). With True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

Average precision is the area under the precision–recall curve of one class, and mAP averages it over the N classes:

$$AP = \int_0^1 p(r)\, dr$$

$$mAP = \frac{1}{N_c} \sum_{c=1}^{N} AP_c \tag{5}$$

mAP@0.5 counts a detection as correct when its intersection-over-union (IoU) with the ground truth is at least 0.5, and mAP@0.5:0.95 averages over IoU thresholds from 0.5 to 0.95 in steps of 0.05, which rewards tighter boxes.

5. RESULTS AND EVALUATION

A. Alphabet Recognition

The Random Forest reached 94.2% accuracy for ASL on the held-out 20% split. The other four languages used the same 63-feature landmark space, and the results are given in Table 2. The LIBRAS and TİD datasets are near-duplicate frames on the same feature space, so their perfect scores should not be read as evidence of better generalization. The German set includes the class SCH (27 classes) and the Turkish set includes special characters (28 classes).

Table -2: Random Forest accuracy by language

Sign language	Accuracy
ASL	94.2%
BISINDO (Indonesian)	92.6%
DGS (German)	91.5%
LIBRAS (Brazilian)	100%
TİD (Turkish)	100%

B. Word and Letter Detection

On the validation split, YOLOv11n obtained 93.4% mAP@0.5, 78.6% mAP@0.5:0.95, 87.5% precision and 91.2% recall, as shown in Table 3. The compute cost is 0.2 ms for preprocessing, 3.0 ms for inference and 2.0 ms for postprocessing per image, about 190 frames per second on the RTX 2050. The live rate of about 5 frames per second is therefore set by the 200-ms browser capture interval and not by the model. Each encoded frame is only about 5–15 KB, so the signer sees the recognized letter or word on screen within roughly 200–400 ms of making the gesture.

Table -3: YOLOv11n validation performance

Metric	Value
mAP@0.5	93.4%
mAP@0.5:0.95	78.6%
Precision	87.5%
Recall	91.2%
Latency per image	~5.2 ms

Fig. 2 shows the four pages of the web interface. The per-class results in Fig. 3 show the best classes (Good Bye, I LoveYou and Water, each 99.5% mAP@0.5) and the weakest ones.

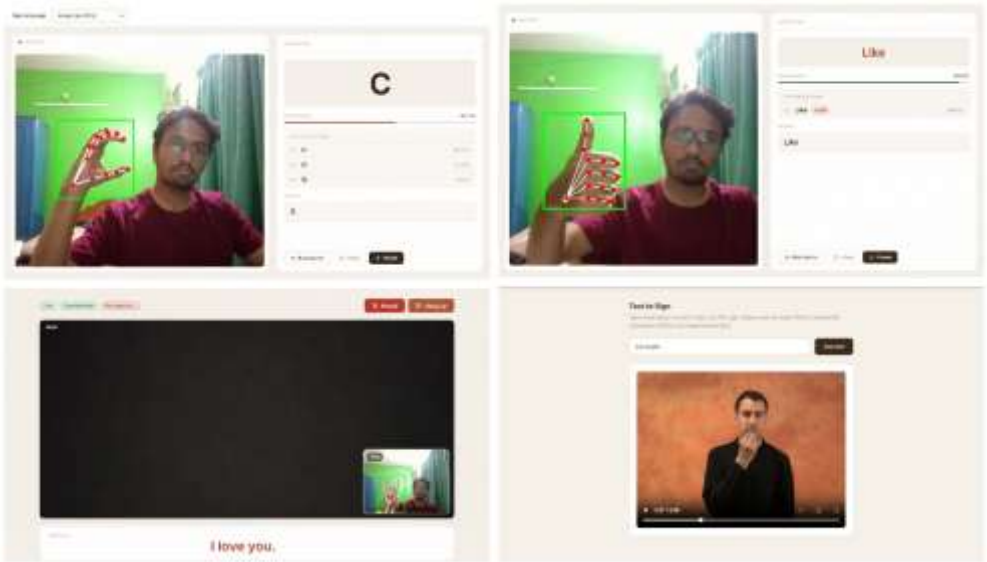


Fig -2: Gesture2Speech web interface: live alphabet recognition (top left), word detection (top right), video call with the sentence “I love you.” shown as a subtitle (bottom left), and Text-to-Sign playback of typed text (bottom right).

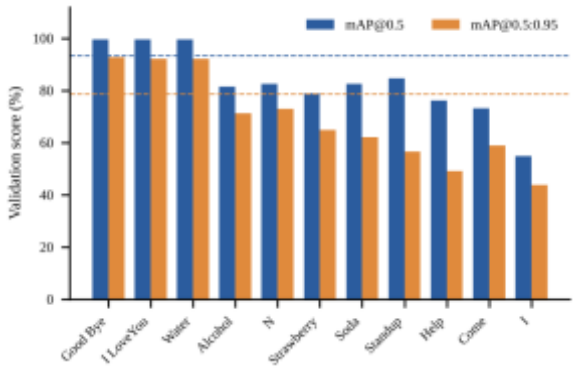


Fig -3: Per-class validation mAP for the three best and eight weakest classes. Dashed lines show the overall values.

C. Stabilization and Sentence Assembly

The stabilizers have no standalone metric because they operate on top of the classifier. They suppress single-frame flicker by requiring four consecutive agreeing frames for words and six of eight for letters. For sentence assembly, the LLM turned the detected tokens “I LoveYou” and “Juice” into “I love juice.” and “Help” and “Please” into “Help me, please.” in live tests. This is a qualitative result, since no formal sentence-quality metric was computed. Text-to-speech uses the browser voice and has no metric of its own.

Table 4 summarizes each feature with the method used and its measured result.

Table -4: Feature, method and measured result

Feature	Method	Result
Letters	Landmarks + RF	94.2% acc.
Words, letters	YOLOv11n	93.4% mAP@0.5
Word commit	4-frame streak	No metric
Letter commit	6-of-8 vote	No metric
Sentence	LLM chain	Qualitative
Speech	Web Speech API	No metric
Text-to-Sign	WLASL lookup	Deterministic

6. DISCUSSION

The results show that two lightweight paths can cover complementary needs. The landmark-based Random Forest is accurate on stills because the wrist-centered, scale-normalized features already remove position and size, so the forest only has to separate shapes. The YOLOv11n detector handles letters and words in one pass and generalized better on live feeds, which is why it replaced the Random Forest as the letter engine. The two numbers are not directly comparable: the forest classifies single-letter stills, whereas the detector must also localize the hand.

The failures follow a clear pattern. They concentrate in near-identical handshapes, such as Help against Come and the letter I against J, and in classes with few samples. The letter X has only one validation image and 38.5% precision, the letter I has two validation images, and the letter T has a recall of 66.7%. The gap between mAP@0.5 (93.4%) and mAP@0.5:0.95 (78.6%) shows that the boxes are less tight at strict overlap thresholds, which matters less for recognition than for localization. Precision (87.5%) is lower than recall (91.2%), so the detector tends to over-report rather than miss gestures, and the temporal stabilizers are what keep such false detections from becoming committed output.

The video-call design reflects a deliberate trade-off. An earlier automatic word buffer, with fixed frame and pause thresholds, silently dropped words because real sentences contain natural pauses. Streaming every detection and leaving the commit decision to the signer through editable chips avoids this loss at the cost of one confirmation step. Sentence assembly with a free LLM pool needed a fallback chain and response-parsing safeguards, because free models are volatile and sometimes reveal their reasoning.

Table 5 places the system against the reviewed work. Gesture2Speech does not claim higher recognition accuracy than the larger transformer or ensemble models, since it uses a much smaller vocabulary and a different dataset. Its contribution is breadth of function: one deployable application covers alphabets, word gestures, spoken sentences and a reverse Text-to-Sign path.

Table -5: Scope compared with related work

Work	Method	Reported scope
[7]	YOLOv11 + MediaPipe	ASL, mainly alphabet
[8]	YOLOv10 + Swin	Indian Sign Language
[9]	Adaptive transformer	Sign language translation
[10]	Video/Image Swin	Word-level benchmarks
[11]	Landmarks + Transformer	Isolated Arabic signs
[12]	BERT + LSTM	Word-level Indian signs
Ours	**Landmarks + RF, YOLOv11n**	**Letters, words, speech, Text-to-Sign**

The architecture also keeps data exposure small. Only compressed detection frames reach the backend, while call audio, video and subtitles travel peer to peer, and the system stores no accounts or sessions. The model is small enough, at 5.2 MB and about 5 ms per image, to run on a laptop GPU, which supports private, local deployment.

7. LIMITATIONS

The vocabulary is fixed at 26 letters and 32 words, so the system cannot interpret phrases beyond the LLM reordering of those tokens. Letters are classified but are not assembled into words during a call. The reported detector metrics are from the validation split, and no separate study of unseen signers, lighting or backgrounds was carried out. Word bounding boxes were generated automatically and some classes have very few samples. Video calls use no STUN or TURN servers and therefore work only on a local network. Sentence assembly depends on a volatile pool of free LLMs and an optional API key, and no sentence-quality metric was computed. Speech output is limited by the browser and system volume, and there is no speech-to-text. The Text-to-Sign module is a lookup over a fixed vocabulary. Finally, the system keeps no accounts or session storage.

8. FUTURE WORK

Several extensions follow from these limitations. First, adding STUN and TURN servers would allow calls across networks. Second, assembling recognized letters into words would support fingerspelling inside calls. Third, a larger vocabulary and continuous signing could be handled with temporal models such as the transformer approaches reviewed in Section 2. Fourth, a held-out test evaluation with unseen signers, lighting and backgrounds should be added, together with a user study involving the deaf and hard-of-hearing community. Fifth,

speech-to-text would complete the spoken-to-sign path. Finally, a quantitative evaluation of sentence quality and an on-device language model would remove the dependence on external services.

9. CONCLUSIONS

This paper presented Gesture2Speech, a real-time web system that recognizes sign language alphabets and word gestures and converts them into text and speech, with a reverse Text-to-Sign path for replies. The pipeline combines MediaPipe landmarks and a Random Forest for letters with a YOLOv11n detector for 58 classes, stabilized by consecutive-frame and sliding-window voting, and adds an LLM and text-to-speech stage for video calls. The Random Forest reached 94.2% accuracy for ASL letters, and YOLOv11n reached 93.4% mAP@0.5, 78.6% mAP@0.5:0.95, 87.5% precision and 91.2% recall on validation data at about 5 ms per image. The system is a practical baseline for bidirectional sign communication on a fixed vocabulary, and broader evaluation, a larger vocabulary and network-ready calls are the main directions for strengthening it.

ACKNOWLEDGEMENT

The authors thank their guide, Hemalatha M, Assistant Professor, Department of Computer Science and Engineering, and Don Bosco Institute of Technology, Bengaluru, for guidance and support during this project.

REFERENCES

[1] Zhang, F., et al.: MediaPipe Hands: On-device Real-time Hand Tracking. arXiv:2006.10214 (2020).
[2] Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001).
[3] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. IEEE CVPR, pp. 779–788 (2016).
[4] Ultralytics: YOLO11 Documentation (2024). Available: https://docs.ultralytics.com
[5] Li, D., Rodriguez, C., Yu, X., Li, H.: Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison. IEEE WACV, pp. 1459–1469 (2020).
[6] W3C: WebRTC 1.0: Real-Time Communication Between Browsers. W3C Recommendation (2021).
[7] Alsharif, B., et al.: American Sign Language Interpretation Using Deep Learning (2025).
[8] Rastogi, U., et al.: Indian Sign Language Recognition Using a Combination of YOLOv10 with Swin Transformer Model (2025).
[9] Shahin, N., Ismail, L.: ADAT: Time-Series-Aware Adaptive Transformer Architecture for Sign Language Translation (2025).
[10] Alkhorafi, A. A., et al.: Ensemble Transformer-Based Word-Level Sign Language Recognition with Multi-Modal Input Fusion (2026).
[11] Alyami, S., et al.: Isolated Arabic Sign Language Recognition Using a Transformer-Based Model and Landmark Keypoints (2024).
[12] Kaliyaperumal, V. G., Gopalan, P. A.: Indian Sign Language Recognition Utilizing BERT and LSTM Models (2025).
[13] Zhang, Y., Jiang, X.: Recent Advances on Deep Learning for Sign Language Recognition (2024).
[14] Roboflow Universe: ASL Letters. Dataset, version 1. Available: https://universe.roboflow.com
[15] A. (ayeshasomething): 32 Word Gestures. Kaggle dataset. Available: https://www.kaggle.com
[16] Ramírez, S.: FastAPI. Software. Available: https://fastapi.tiangolo.com
[17] Pedregosa, F., et al.: Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011).
[18] Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000).
[19] Zulko: MoviePy: Video Editing with Python. Software. Available: https://zulko.github.io/moviepy
[20] PeerJS: PeerJS: Simple Peer-to-Peer with WebRTC. Software. Available: https://peerjs.com
[21] OpenRouter: OpenRouter API Documentation. Available: https://openrouter.ai/docs
[22] Lugaresi, C., et al.: MediaPipe: A Framework for Building Perception Pipelines. arXiv:1906.08172 (2019).