

Real-Time Sign Language Translator Using MediaPipe and Bidirectional LSTM

Avanthika Sudev¹, Hana Al Haris², Adithyan Eratt Shelly³

¹Dept. of CSE – AI & ML, JAIN (Deemed-to-be University), Bangalore

²Dept. of CSE – AI & ML, JAIN (Deemed-to-be University), Bangalore

³Dept. of CSE – AI & ML, JAIN (Deemed-to-be University), Bangalore

Abstract – Gesture-based communication is indispensable for millions of individuals who are deaf or hard-of-hearing, yet automated tools that can interpret these gestures affordably and accurately in live settings remain limited. Motivated by this gap, the authors developed a camera-only ASL translation system wherein a MediaPipe skeleton tracker, a BiLSTM sequence classifier, and a rule-assisted sentence constructor work as a unified pipeline. Each captured frame yields a 63-value descriptor derived from 21 three-dimensional hand keypoints; wrist anchoring and inter-joint distance scaling ensure that descriptor values stay meaningful regardless of where the signer sits or how large their hands appear on screen. A sliding prediction buffer fed into a majority-vote gate trims erroneous word triggers by 73% while adding only 267 ms of delay. Evaluated on fifty ASL word classes, the network scores 91.4% top-1 accuracy and keeps the full processing loop at 28–30 frames per second without any GPU involvement.

Key Words: *ASL Recognition, Gesture Translation, BiLSTM, MediaPipe, Hand Landmarks, NLP Sentence Formation, Assistive Technology, Text-to-Speech.*

1. INTRODUCTION

Hearing loss affects a substantial segment of the global population, with the WHO estimating over 430 million individuals in this category worldwide [1]. For a large proportion of this group, visual gesture languages serve as the native tongue, and ASL in particular holds this role for several hundred thousand to two million individuals across North America. What makes ASL a complete linguistic system — with its own phonology, morphology, and syntax entirely separate from English — also makes it mutually unintelligible to the vast majority of the hearing world.

Interpreter shortages, high service costs, and geographic coverage gaps collectively mean that many interactions between deaf signers and non-signing hearing individuals take place without language support at all. Technology-assisted translation therefore carries real practical stakes: healthcare consultations, job interviews, and emergency communications are among the contexts where the inability to bridge this language gap has serious consequences.

Prior automated solutions have generally traded one type of accessibility for another. Depth-sensor setups achieve strong recognition but demand hardware costing hundreds of dollars. Fingerspelling classifiers run on ordinary cameras but cover only a fraction of real ASL vocabulary. Sentence-level output is rarely present, and CPU-speed real-time performance is rarer still. A solution that avoids all of these compromises simultaneously has not been widely reported.

To address this, the current work builds a four-component pipeline: a MediaPipe-powered joint extractor, a normalization stage that strips signer-specific and camera-specific variation from the raw coordinates, a stacked

BiLSTM network trained on publicly available ASL datasets, and an NLP module that turns a sequence of recognized words into fluent sentences accompanied by synthesized audio.

2. LITERATURE REVIEW

Early machine-learning approaches to SLR leaned heavily on manually designed feature extractors. Dreuw et al. [12] extracted histogram-based gradient features alongside motion-field descriptors and fed the resulting sequences into probabilistic state-machine models. Accuracy was acceptable within narrow, controlled experimental setups, but performance degraded sharply whenever recording conditions shifted. The fragility traced back to the assumption baked into hand-designed features: that the visual cues important for one recording scenario would remain important in a different one — an assumption that rarely holds in practice.

A data-driven alternative emerged when Koller et al. [3] showed that a convolutional architecture trained on large automatically collected datasets could exceed prior feature-engineering baselines by meaningful margins. The key insight was allowing the network to discover what visual patterns mattered, rather than specifying them by hand. However, that work treated each frame independently, so temporal relationships between consecutive frames — the very information that distinguishes one motion path from another — were invisible to the model.

For the skeletal input representation, the anchor reference in this area is the MediaPipe Hands system documented by Zhang et al. [2]. Its two-module design — a coarse region proposal step followed by a fine-grained keypoint regressor — delivers sub-15-pixel accuracy on 21 hand joints at roughly 30 frames per second on consumer hardware without any depth sensor. This combination of accuracy, speed, and zero hardware cost made MediaPipe the obvious choice as the perception front-end.

On the question of how to model gesture sequences, Pigou et al. [9] ran a systematic comparison of recurrent architectures using skeletal data and found that bidirectional LSTM networks consistently outscored their unidirectional equivalents, with the margin ranging from two to four percentage points. The advantage is architectural: a bidirectional pass allows the model to incorporate information from the second half of a gesture when classifying its midpoint, which unidirectional models cannot do.

Training data diversity emerged as a separate but equally critical variable. The WLASL benchmark introduced by Li et al. [5] brought together over 21,000 video instances spanning 2,000 ASL word categories contributed by 119 unique signers, making signer-independent evaluation possible at scale for the first time. Joze and Koller [6]

assembled a complementary resource in MS-ASL by harvesting naturalistic signing from social media and educational platforms, producing 25,513 clips spread across 1,000 categories. Using both corpora simultaneously during training exposes a model to a far wider range of individual signing styles than either dataset alone.

3. METHODOLOGY

A. System Architecture

The overall design separates concerns across three independent layers. A sensing layer handles video acquisition through OpenCV at 640×480 pixels and nominally 30 FPS, passes each frame to MediaPipe for skeleton detection, and routes the resulting joint coordinates through a normalization block to produce a fixed-length input vector. A classification layer accumulates these vectors into a 30-frame rolling window, runs the BiLSTM model on the buffer at every new frame, and applies a voting gate that prevents noisy intermediate predictions from reaching the output stage. An interface layer collects confirmed word tokens, constructs sentences with phrase-level NLP assistance, passes each utterance to a TTS synthesizer running in a parallel thread, and serves an annotated live video stream through a lightweight web application.

B. Hand Landmark Detection and Normalization

MediaPipe Hands runs in continuous tracking mode rather than static-image mode, which means that after an initial palm detection, subsequent frames use the previous crop region as a starting point. This amortizes the cost of palm detection across frames and maintains real-time throughput. Configuration uses a single-hand limit, a detection confidence floor of 0.70, and a tracking confidence floor of 0.50. When the tracker successfully locates a hand, it returns a 21-by-3 array of floating-point coordinates in normalized image space; when no hand is present, the system substitutes a zero vector to preserve buffer dimensionality.

Raw MediaPipe output conflates hand identity with camera geometry: the same sign performed further from the camera looks different from the same sign performed up close, and the same sign performed in the upper left of the frame looks different from one performed in the lower right. Both variations are removed through a sequential transform. All 21 joint positions are first shifted so that the wrist joint lands at the coordinate origin. The inter-joint span from wrist to middle-finger knuckle is then computed and used to divide every coordinate in the shifted array. After this two-step process the descriptor captures finger configuration and motion geometry without encoding irrelevant positional or scale information. The final step concatenates the 21 normalized three-dimensional positions into a single vector of length 63.

C. Network Architecture and Training Strategy

The sequence classifier stacks two bidirectional LSTM blocks. The first processes all 30 time steps and passes its full output sequence to the second; the second collapses the sequence to a single fixed-length state vector. Each LSTM block is followed by a 25% dropout layer and a batch normalization layer. A fully connected layer with 128 neurons and ReLU activation, regularized with an L2 penalty of 0.001, maps the state vector toward the output space; a 35% dropout layer precedes the final softmax classification head. Across 50 output classes, this design produces

approximately 380,000 learnable parameters and a serialized file of roughly 1.5 MB, both small enough to allow sub-second load times and CPU-only inference.

Weight updates use AdamW with a starting learning rate of 0.001 and a decoupled weight decay of 0.0001. Because ASL datasets are inherently imbalanced — common words accumulate far more training clips than rare ones — class-proportional weights computed via scikit-learn are supplied to the loss function so that minority classes receive proportionally higher gradient signal. A checkpoint callback preserves the snapshot with the highest validation accuracy; early stopping with a patience of ten epochs and full weight restoration terminates training if progress stalls; and a plateau-based scheduler halves the learning rate after five stagnant epochs. On the fifty-class WLASL training split, the stopping criterion triggered at epoch 47.

D. Prediction Smoothing and Sentence Output

Frame-level classifier outputs are inherently noisy: gestures that involve hand motion will pass through transitional configurations that the model has never seen labeled, and confidence values during these transitions fluctuate rapidly. The voting module addresses this by storing the eight most recent (class label, confidence score) pairs in a circular buffer. A word is forwarded to the output stage only when three simultaneous criteria are satisfied: the most frequently predicted class holds a majority in the buffer, the average confidence for that class exceeds 0.75, and at least one full second has elapsed since the previous word output. The elapsed-time criterion serves two purposes — it filters transition noise and prevents a single held sign from generating duplicate entries.

Confirmed words accumulate in a sentence buffer inside the SentenceFormer module. After each addition, the module checks the most recent word sequence against a pre-loaded phrase dictionary using token-set overlap scoring and character-level similarity via Python's difflib library. Matches that exceed a combined threshold of 0.70 replace the raw token sequence with the matched phrase, so that frequently occurring multi-word expressions are rendered correctly rather than as disconnected items. Audio output is handled by a pyttsx3-backed synthesizer living in a daemon thread; a thread-safe queue decouples synthesis requests from the video loop so that neither process blocks the other.

4. RESULTS

All experiments were carried out on a laptop equipped with a six-core Intel Core i7-10750H running at 2.6 GHz, 16 GB of DDR4 memory, and an NVIDIA GTX 1650 Mobile discrete GPU with 4 GB of video RAM, under Windows 11. The software stack comprised TensorFlow 2.17.1, MediaPipe 0.10.14, and OpenCV 4.10.0.84. Training data came from the fifty most frequently occurring sign classes in WLASL, with a per-class cap of fifty video clips that yielded approximately 2,100 usable sequences in total after MediaPipe processing. An 80/20 stratified split reserved 420 sequences as a held-out validation set.

Table -1: Normalization Ablation Study

Feature Representation	Accuracy	Gain
Raw (x,y,z) – no norm.	83.1%	—
Wrist-centered only	88.6%	+5.5%
Proposed (full norm.)	91.4%	+8.3%

The normalization ablation in Table 1 isolates the contribution of each preprocessing step. Removing all normalization drops accuracy to 83.1%, confirming that raw screen-space coordinates carry substantial extraneous variation. Applying only the wrist-centering step recovers 5.5 percentage points. Adding the scale division step recovers a further 2.8 points, bringing the validated accuracy to 91.4% — an overall gain of 8.3 points from preprocessing alone, with no change to the network. Training dynamics were stable: the final training-validation gap was 4.8 points, and the ReduceLROnPlateau callback reduced the learning rate twice (at epochs 28 and 36) before early stopping triggered at epoch 47. Live inference averaged 28.3 FPS in CPU-only mode; the MediaPipe joint detection step contributed roughly 20 ms per frame, while BiLSTM inference added approximately 5 ms. Median word-output latency from the moment of gesture completion was 0.9 seconds.

Table -2: Voting Window Size Evaluation

Window	False Emiss.	Correct	Latency
0 (none)	31	20	0 ms
4 frames	12	20	133 ms
8 frames*	9	20	267 ms
12 frames	7	18	400 ms

* Selected configuration

Table 2 sweeps four voting-buffer lengths against two outcome dimensions: spurious outputs and missed outputs. Without any smoothing, 31 spurious word tokens are generated across the 20-gesture test sequence. A four-frame window cuts this to 12 while still capturing all 20 intended gestures. At eight frames, spurious tokens fall to 9, all 20 intended gestures remain captured, and the added latency is 267 ms — a figure that falls within acceptable bounds for conversational pacing. Extending to twelve frames squeezes out two more false triggers but begins to miss genuine gestures (18 of 20), because the buffer fills with inter-gesture transition frames before the target class achieves a majority. These results confirm eight frames as the operating point that best balances reliability against responsiveness. Sentence assembly accuracy was 95% under ideal word emission and 65% under live recognition conditions, with fuzzy matching recovering three additional phrases that strict token matching missed.

Table -3: Comparison with Related Systems

System	Hardware	Accuracy	RT?
DeepASL [10]	Leap Motion	93.7%	No
Koller [3]	Std. Cam	89.2%	No
Pigou [9]	Kinect	91.5%	Yes
Jiang [11]	Cam+GPU	94.3%	No
Proposed	Webcam	91.4%	Yes

5. DISCUSSION

Table 3 frames the proposed system within the broader literature. Two prior works achieve higher absolute accuracy: DeepASL at 93.7% [10] and Jiang et al. at 94.3% [11]. Both, however, carry hardware or speed constraints that would prevent casual deployment — the former requires a proprietary near-field sensor, the latter requires GPU acceleration to operate in real time. Pigou et al. [9] is the nearest comparison point for a simultaneously real-time system, but its 91.5% accuracy figure relies on a depth camera that is no longer commercially available. The proposed method occupies the intersection of the three most

practically important constraints — camera-only, real-time, above 90% — and that intersection is, to the authors' knowledge, unoccupied by any prior published system.

At the per-class level, the distribution of F1 scores narrows the remaining accuracy gap to a specific linguistic phenomenon. Signs that employ geometrically distinct handshapes or clear directional motion — for example, an open waving hand versus a nodding closed fist — scored between 0.95 and 0.97. Signs that share a handshape and differ only in the body region being touched, such as the “mother” and “father” pair (thumb at chin versus thumb at forehead), fell in the range 0.78–0.84. Incorporating additional spatial context — either through upper-body pose estimation or through deliberate data collection that over-represents phonologically confusable pairs — is the most direct path to closing this gap.

6. CONCLUSION

Four design goals guided this project: operate without specialized sensors, run fast enough for live conversation, cover a meaningful signing vocabulary, and produce sentence-level rather than word-level output. All four goals were met. The delivered system records 91.4% classification accuracy across fifty ASL word classes, sustains a 28–30 FPS processing rate on an unmodified consumer laptop, and outputs full sentences aloud within approximately one second of a gesture's completion. The technical foundation rests on five innovations: a coordinate normalization scheme that improved accuracy by 8.3 percentage points; a compact BiLSTM architecture with roughly 380,000 parameters that runs entirely on CPU; a voting filter that reduced false triggers by 73% at a latency cost of 267 ms; a fuzzy phrase-matching layer that raised sentence assembly accuracy to 65% under real recognition errors; and an asynchronous audio synthesis module that decouples speech output from video throughput. Beyond the performance numbers, the work illustrates that the combination of an open-source skeleton tracker, a modest recurrent network, and a lightweight language module is sufficient to deliver a usable, low-cost sign language translation tool on hardware that most households already own.

ACKNOWLEDGEMENT

The team owes a particular debt to Dr. Shivam Swaroop, whose detailed technical feedback and steady encouragement were instrumental at every stage of this project. Dr. Chandrasekar V. and Dr. Kamlesh Tiwari provided valuable institutional guidance as Program Head and Head of Department respectively. Coordination support from Dr. Rakesh Patra, Dr. Swati Gupta, and Dr. K. S. Arvind kept the project on track across its full duration. The authors also thank the wider faculty community of the Department of CSE – AIML at JAIN (Deemed-to-be University) for a research environment that made this kind of work possible.

REFERENCES

1. World Health Organization, World Report on Hearing, Geneva, Switzerland: WHO Press, 2021.
2. F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “MediaPipe Hands: On-device real-time hand tracking,” in Proc. ECCV Workshop on CV for AR/VR, Glasgow, UK, 2020.
3. O. Koller, J. Forster, and H. Ney, “Continuous sign language recognition: Towards large vocabulary statistical recognition systems,” Computer Vision and Image Understanding, vol. 141, pp. 108–125, Dec. 2015.
4. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

5. D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in Proc. IEEE WACV, 2020, pp. 1459–1469.
6. H. R. V. Joze and O. Koller, "MS-ASL: A large-scale data set and benchmark for understanding American Sign Language," in Proc. BMVC, Cardiff, UK, 2019.
7. O. M. Sincan and H. Y. Keles, "AUTSL: A large-scale multi-modal Turkish sign language dataset and baseline methods," IEEE Access, vol. 8, pp. 181340–181355, Sep. 2020.
8. A. Duarte et al., "How2Sign: A large-scale multimodal dataset for continuous American sign language," in Proc. IEEE/CVF CVPR, Nashville, TN, USA, 2021, pp. 2735–2744.
9. L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video," IJCV, vol. 126, pp. 430–439, 2018.
10. R. Cui, H. Liu, and C. Zhang, "Recurrent convolutional neural networks for continuous sign language recognition by staged optimization," in Proc. IEEE CVPR, 2017, pp. 7361–7369.
11. S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. Fu, "Skeleton aware multi-modal sign language recognition," in Proc. IEEE/CVF CVPRW, Nashville, TN, USA, 2021.
12. P. Dreuw, D. Rybach, T. Deselaers, M. Zahedi, and H. Ney, "Speech recognition techniques for a sign language recognition system," in Proc. INTERSPEECH, 2006, pp. 2513–2516.