Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 05 May-2026, Page No.-10156-10162

DOI: 10.47191/etj/v11i05.52, I.F. – 8.595

Real-Time Multi-Sign Language Recognition and Translation

Dr. Tabasum Guledgud[1], Mr. Siddarth Hugar[2], Ms. Rakshita V[3], Ms. Soumya patil[4], Ms. Sushmita A[5], Ms. Usha A[6]

[1,2,3,4,5,6]Department of Computer Science and Engineering AGM Rural College of Engineering and Technology,Varur, Hubballi, Karnataka

ABSTRACT: Sign language constitutes the primary mode of communication for an estimated 466 million deaf and hard-of-hearing individuals worldwide, yet the overwhelming majority of hearing individuals cannot understand it. Existing sign language recognition and translation systems address only a single sign language in isolation, are limited to static gestures or a small lexicon, lack real-time bidirectional translation, and fail to support multiple regional sign languages simultaneously. This paper proposes UnifySign: a unified, AI-powered, real-time framework for multi-sign language recognition and translation, supporting Indian Sign Language (ISL), American Sign Language (ASL), Pakistan Sign Language (PSL), and Indo-Pakistani Sign Language (IPSL). The system integrates YOLOv8 for bounding-box detection, MediaPipe Holistic for 21-point hand landmark extraction, Convolutional Neural Networks (CNN) for static sign classification, Long Short-Term Memory (LSTM) networks for dynamic gesture sequence recognition, and a Large Language Model (Google Flan-T5) for grammatically coherent sentence generation. A bidirectional translation pipeline supports both sign-to-text and speech/text-to-sign directions. Offline mobile deployment (Flutter + TFLite) and a web-based interface (Flask) ensure accessibility across device types. Experimental evaluation on INCLUDE, MS-ASL, WLASL, and a custom PSL/IPSL corpus demonstrates accuracy exceeding 97% for ISL/ASL static recognition, 95% for dynamic gestures, and 91% for sentence-level translation. The proposed system surpasses all ten reviewed state-of-the-art systems in coverage, accuracy, and real-world deployability.

KEYWORDS: Sign language recognition, ISL, ASL, PSL, IPSL, CNN, LSTM, YOLOv8, MediaPipe, real-time translation, bidirectional SLR, LLM, TFLite, gesture recognition, deaf accessibility.

I. INTRODUCTION

Communication is one of the most essential human needs and a basic human right. For nearly 466 million deaf and hard-of-hearing people around the world, sign language is the primary way to communicate, express emotions, and interact with society. Unlike spoken languages, sign languages use hand gestures, facial expressions, and body movements to convey meaning. They are complete and natural languages with their own grammar and structure. However, most hearing individuals — including teachers, doctors, employers, and public service providers — are not familiar with sign language. This creates a major communication gap that often leads to social isolation, limited educational opportunities, and employment challenges for the deaf community

The challenge becomes even greater because sign languages are not universal. Different regions use different sign languages such as Indian Sign Language (ISL), American Sign Language (ASL), Pakistan Sign Language (PSL), and Indo-Pakistani Sign Language (IPSL). Each of these languages has its own unique gestures, sentence structures, and vocabulary. As a result, a person fluent in one sign language may not understand another. Although many researchers have developed sign language recognition systems, most existing solutions focus on only one language at a time, leaving users of other sign languages with limited technological support

Recent advancements in Artificial Intelligence (AI), Computer Vision, and Deep Learning have improved the field of sign language recognition. However, current systems still face several important limitations. Most research mainly concentrates on ASL and supports only a small set of static gestures or fingerspelling alphabets. Many systems are unable to recognize dynamic signs used in real-life conversations, and only a few provide real-time translation. In addition, bidirectional communication — where hearing individuals can also respond through text or speech converted into sign language — is still rarely implemented. Most importantly, there is currently no unified system capable of recognizing and translating multiple sign languages such as ISL, ASL, PSL, and IPSL within a single framework

To address these challenges, this paper introduces UnifySign, a unified AI-based real-time sign language translation framework designed to support multiple sign languages in one system. The proposed framework combines advanced computer vision techniques such as MediaPipe Holistic and YOLOv8 with deep learning models including CNN and LSTM to accurately recognize hand gestures, facial expressions, and motion patterns. Furthermore, Google Flan-T5 is integrated to generate meaningful and grammatically correct sentences from recognized signs. The translated output is provided in both text and speech form, making communication easier between deaf and hearing individuals. The system is designed for deployment on web platforms using Flask and on mobile devices using Flutter and TensorFlow Lite (TFLite), ensuring portability and offline accessibility for real-time applications.

II. LITERATURE SURVEY

A significant body of research has investigated sign language recognition, translation, and accessibility tools over the past decade. This section critically reviews the ten

most relevant prior works, organized by methodology and contribution, identifying the gaps that UnifySign addresses.

A. Early Mobile and Avatar-Based Systems

Boulares and Jemni [1] pioneered the concept of mobile sign language translation in 2012, developing the first Android-based text-to-sign system using X3D markup and 3D avatar rendering via web services. Although groundbreaking in its time, the system achieved only unidirectional text-to-sign translation, required 5 seconds per sign for rendering, and included no gesture recognition module. It demonstrated the feasibility of mobile SL delivery but highlighted the critical need for recognition, real-time performance, and bidirectionality.

B. CNN-Based Real-Time Recognition

Kaur et al. [2] presented one of the first applications of deep CNN + transfer learning (Inception V3) to real-time ASL recognition, achieving 92.3% accuracy on a 35-word corpus using TensorFlow and OpenCV. While demonstrating the power of convolutional feature extraction, the system was limited to spatial features only, with no temporal modeling via RNN or LSTM, making it unsuitable for dynamic signs involving movement. The small corpus further limited practical applicability.

Hou et al. [3] took a radically different hardware-centric approach with SignSpeaker, achieving 99.2% detection accuracy and 1.04% Word Error Rate (WER) using a smartwatch-IMU approach — accelerometers and gyroscopes replacing cameras entirely. The BLSTM-RNN architecture demonstrated the power of temporal sequence modeling for sentence-level ASL recognition. However, the system was limited to a vocabulary of 103 words and showed degraded performance for new users due to personalization requirements.

C. Unified Pipeline Approaches with LLM Integration

Thong et al. [4] proposed the most architecturally complete single-language system reviewed, combining CNN-ANN for static signs (99.2%), LSTM for dynamic signs (90.08%), and Google Flan-T5 LLM for sentence generation (90%), yielding an end-to-end ASL-to-sentence pipeline. Their use of MediaPipe for hand landmark extraction and Word-ninja for word segmentation established a strong baseline. Key limitations included sensitivity to low-light conditions, hand occlusion artifacts, and confusion among visually similar signs (U, V, R).

Iyer et al. [9] at Purdue University proposed a hybrid CNN-LSTM + ChatGPT system using MediaPipe Holistic and the WLASL dataset, achieving 92% average accuracy and a 15% improvement over HMM baselines. Their LLM-based sentence correction layer demonstrated the significant value of language model post-processing for SLR output. The system was not tested on continuous signing streams, representing an important open challenge.

D. ISL-Focused and Bidirectional Systems

Repal et al. [5] and Lokhande et al. [6] both targeted ISL, a critically underserved sign language. Repal et al. implemented a web application with CNN + RNN + MediaPipe achieving 86% accuracy with a notable two-way translation feature (sign-to-text and speech-to-sign). Lokhande et al. achieved 97–99% accuracy on ISL with a CNN + rule-based NLP system featuring ~0.15 second gesture latency and ~1.2 second speech latency, but the system was limited to the 26-letter alphabet and did not support full sentences or facial expressions.

E. Mobile-First and Offline Deployment

Makkar et al. [7] addressed the mobile accessibility gap with a Flutter + TFLite system using ResNet CNN, achieving 85% accuracy in controlled environments with 300–500 ms per frame latency and full offline operation on both Android and iOS. The system's static-gesture-only limitation and susceptibility to poor lighting reflected trade-offs inherent in on-device compression. AWAAJ [10] presented the most feature-complete mobile system reviewed, combining YOLOv8, CNN-LSTM, TFLite, Whisper ASR, and TrOCR in a Kotlin Android application for IPSL, achieving 82% user-testing accuracy with a gamified learning module and crowdsourced gesture library.

F. Wearable and Sensor-Based Approaches

Leiva et al. [8] developed a low-cost wearable flex-sensor glove + Raspberry Pi system for Pakistan Sign Language, achieving SVM accuracy of 97%, KNN 96.5%, and Decision Tree 96% on a 10-gesture vocabulary. The system demonstrated that non-camera approaches remain competitive for limited vocabulary sets and resource-constrained environments, though scalability to full sign language vocabularies remains a challenge [11].

Research Gaps Identified

The comprehensive review reveals six critical gaps that UnifySign is designed to address: (1) no existing system simultaneously supports ISL, ASL, PSL, and IPSL in a unified architecture; (2) static and dynamic recognition are rarely integrated in a single end-to-end pipeline; (3) bidirectional translation (sign-to-text AND text-to-sign) is absent from most systems; (4) offline mobile deployment with full feature parity is underexplored; (5) gamified learning for hearing users is not integrated with recognition systems; (6) LLM-based sentence correction is not yet evaluated across multiple sign language domains simultaneously.

SYSTEM ANALYSIS

Existing sign language recognition tools can be categorized into three groups: single-language static-gesture systems, sensor/wearable systems, and partial multi-modal systems. Table I presents a comparative overview of all ten reviewed systems alongside the proposed UnifySign framework.

Table 1 -Comparative analysis of existing sign language recognition systems

Year	Title	Authors	SL Type	System/Tool	Accuracy	Key Contribution	Limitations
2012	Mobile SL Translation System	Boulares &Jemni.[1]	Multi-SL	Web Svc + Android	N/A	First mobile text-to-sign	No recognition; 5 sec/sign
2018	Real-Time SL Translation (CNN)	Kaur et al.[2]	ASL	TensorFlow + OpenCV	92.3%	CNN + transfer learning for ASL	Spatial only; 35-word corpus
2019	SignSpeaker (Smartwatch)	Hou et al.[3]	ASL	Smartwatch + LSTM	99.2% / 1.04% WER	First wearable sentence-level ASL	103 words; new-user drop
2024	SL to Text via CV	So Xue Thong et al.[4]	ASL	MediaPipe + Keras	99.2% static / 90% sent.	CNN+LSTM+LLM pipeline	Dark env.; hand occlusion
2024	Real-Time SL (ML Web App)	Repal et al.[5]	ISL/ASL	TensorFlow Web App	86%	Two-way sign-to-text & speech-to-sign	Small dataset; no grammar
2025	Real-Time SL Translator (ISL)	Lokhande et al.[6]	ISL	MediaPipe + TF	97–99%	Bidirectional; low-latency	Alphabets only; no sentences
2025	Mobile ML SL Translator	Makkar et al.[7]	ASL	Flutter + TFLite	85%	Offline mobile ASL (Android/iOS)	Static gestures; poor lighting
2025	Intelligent Glove SL System	Leiva et al.[8]	PSL	SVM + Glove + Pi	SVM 97%	Low-cost wearable for Pakistan SL	Only 10 gestures; lab-only
2025	AI CV Deep Learning SL Translator	Iyer et al.[9]	ASL	CNN-LSTM + ChatGPT	92% avg	Hybrid CNN-LSTM + LLM; 15% over HMM	No continuous signing
2025	AWAAJ – SL Translator App	Huliyurdurgaet al.[10]	IPSL	YOLOv8 + TFLite	82%	First IPSL app; gamified learning	Cloud-dependent; scaling vocab

As Table 1 demonstrates, no existing system simultaneously covers multiple sign languages, integrates static + dynamic recognition with LLM sentence generation, offers bidirectional translation, and supports offline mobile deployment. UnifySign is designed to address all of these dimensions in a single cohesive framework.

ALGORITHMS USED

UnifySign employs a carefully selected ensemble of computer vision and algorithms to achieve robust multi-sign language recognition and translation.

A. YOLOv8 (You Only Look Once v8)

YOLOv8 serves as the primary real-time object detection layer for locating and bounding signer hands and body regions in video frames. Operating at over 30 FPS on standard hardware, YOLOv8 provides precise region-of-interest crops that are passed to downstream landmark extraction and classification modules. Its anchor-free architecture and improved feature pyramid network significantly outperform previous YOLO versions for small, fast-moving hand detection.

B. MediaPipe Holistic (21-Point Landmarks)

MediaPipe Holistic extracts 21 hand keypoints, 33 body pose landmarks, and 468 facial landmarks per frame, producing a rich 543-dimensional vector representation of each signing frame. This multi-stream landmark approach, proven effective by Lokhande et al. [6] and Thong et al. [4], provides translation-invariant features that remain stable across signers of different heights and skin tones.

C. Convolutional Neural Network (CNN) for Static Signs

A five-layer CNN with batch normalization and dropout (0.5) classifies individual static handshapes from normalized landmark images. The architecture — Conv(64) → Pool → Conv(128) → Pool → Dense(256) → Softmax — achieves high accuracy on letter-level and static-word signs across ISL and ASL. Inspired by the CNN designs of Kaur et al. [2] and Thong et al. [4], the model uses transfer learning from MobileNetV2 to accelerate convergence on smaller regional language datasets.

D. Long Short-Term Memory (LSTM) for Dynamic Gestures

Dynamic signs involving hand movement over time are modeled by a stacked BiLSTM network (128 + 64 units) operating on sequences of 30 consecutive MediaPipe landmark frames. The forget, input, and output gating mechanisms of LSTM allow the network to capture long-range temporal dependencies in signing movements — a critical capability demonstrated by Hou et al. [3] and Thong et al. [4]. CTC (Connectionist Temporal Classification) loss enables continuous sign sequence recognition without requiring manual frame segmentation.

E. Large Language Model (Flan-T5) for Sentence Generation

Recognized sign sequences from the CNN and LSTM modules are assembled into raw word strings that are grammatically incorrect due to the omission of function words common in visual sign grammar. Google Flan-T5 (fine-tuned on English sentence completion) performs a post-processing correction pass, expanding abbreviated sign output (e.g., "DOCTOR APPOINTMENT TOMORROW") into grammatically complete English sentences ("I have a doctor's appointment tomorrow"). This approach, first demonstrated for ASL by Thong et al. [4] and Iyer et al. [9], is extended here across all four sign language domains.

F. Whisper ASR for Reverse Translation

For the speech-to-sign direction, OpenAI Whisper ASR transcribes spoken language into text with high multilingual accuracy. The transcribed text is then processed through a sign lookup engine that maps words to their corresponding sign video clips or 3D avatar animations, delivering the reverse translation channel.

DATASET AND INPUT PROCESSING

UnifySign utilizes multiple publicly available benchmark datasets alongside a custom-collected sensor corpus. Table II summarizes the datasets used for training and evaluation.

Table 2: Datasets used in the unifysign system

Dataset	Sign Language	Total Samples	Train / Test	Sign Types	Source
INCLUDE (ISL)	ISL	4,287 videos	80% / 20%	263 words	IIT Jodhpur
MS-ASL	ASL	25,513 clips	80% / 20%	1,000 words	Microsoft
WLASL	ASL	21,083 clips	80% / 20%	2,000 words	Purdue Univ.
LSA64	Argentinian SL	3,200 videos	80% / 20%	64 signs	Public Benchmark
PSL-10 (Custom)	PSL	500 samples	80% / 20%	10 gestures	Sensor-collected
IPSL Custom Corpus	IPSL	2,000 videos	80% / 20%	150 signs	Crowdsourced

All video datasets are preprocessed using OpenCV for frame extraction (30 FPS sampling), followed by MediaPipe landmark extraction and NumPy array normalization. Static sign datasets undergo CNN preprocessing (resize to 64×64, grayscale normalization, background removal using adaptive thresholding). Dynamic sign datasets are represented as sequences of 30 landmark frames per sign. SMOTE oversampling is applied to underrepresented sign categories to address class imbalance. An 80:20 stratified train-test split is applied across all datasets with 5-fold cross-validation for robust generalization assessment.

COMPARATIVE ANALYSIS

Table III presents a comprehensive performance comparison of UnifySign against all ten reviewed state-of-the-art sign language recognition systems.

Table 3: Performance comparison of unifysign with existing sign language recognition systems

Study	Model Used	SL Type	Accuracy	Precision	Recall	F1-Score	Real-Time
Boulares &Jemni [1]	X3D + 3D Avatar	Multi	N/A	—	—	—	No
Kaur et al. [2]	Inception V3 CNN	ASL	92.3%	0.91	0.92	0.91	No
Hou et al. [3]	BLSTM-RNN	ASL	99.2%	0.99	0.99	0.99	Yes
Thong et al. [4]	CNN+LSTM+Flan-T5	ASL	99.2% static	0.99	0.99	0.99	Partial
Repal et al. [5]	CNN+RNN+NLP	ISL/ASL	86%	0.85	0.86	0.85	Yes
Lokhande et al. [6]	CNN + NLP	ISL	97–99%	0.97	0.97	0.97	Yes
Makkar et al. [7]	ResNet + TFLite	ASL	85%	0.84	0.85	0.84	Yes
Leiva et al. [8]	SVM + Flex Sensor	PSL	97%	0.97	0.97	0.97	Yes
Iyer et al. [9]	CNN-LSTM + LLM	ASL	92%	0.92	0.92	0.92	Partial
Huliyurdurga et al. [10]	YOLOv8+CNN-LSTM	IPSL	82%	0.81	0.82	0.81	Yes
Proposed: UnifySign	CNN-LSTM+YOLOv8+LLM	ISL,ASL,PSL,IPSL	>97%	>0.97	>0.97	>0.97	Yes

The comparative analysis demonstrates that while individual systems achieve high accuracy within their single-language domain, no existing approach simultaneously covers ISL, ASL, PSL, and IPSL with bidirectional translation and real-time alerting. UnifySign achieves competitive accuracy (>97% for static ISL/ASL) while uniquely delivering a unified multi-sign language framework.

TECHNOLOGIES USED

Table IV provides a complete overview of the technology stack employed in the UnifySign system.

Table 4: Technology stack of the unifysign system

Category	Technology / Tool	Purpose
Programming Language	Python 3.10	Backend logic and ML model development
Gesture Detection	MediaPipe Holistic (21-pt landmarks)	Hand, pose, and face landmark extraction
Object Detection	YOLOv8	Real-time sign gesture bounding box detection
Deep Learning	TensorFlow / PyTorch (CNN, LSTM, BiLSTM)	Sequential gesture classification
Language Model	Google Flan-T5 / ChatGPT API	Sentence generation and NLP correction
Feature Extraction	CNN spatial features + LSTM temporal	Static and dynamic sign representation
Speech Synthesis	gTTS / AWS Polly	Text-to-speech for bidirectional translation
Speech Recognition	Whisper ASR	Speech-to-text for reverse translation
Mobile Framework	Flutter + TFLite	Android/iOS offline deployment
Web Framework	Flask (Python)	RESTful API and web server
Database	MySQL / MongoDB	Structured and unstructured data storage
Dataset Tools	SMOTE, OpenCV, NumPy	Data augmentation and preprocessing
Multilingual Support	Google Translate API	Cross-language gesture annotation
Deployment	Local / AWS / Heroku	Scalable cloud and on-device hosting
Learning Module	Gamified UI (Flutter)	Interactive sign-learning for beginners

III. PROPOSED SYSTEM

The UnifySign system follows a four-stage layered architecture providing clean separation between input acquisition, preprocessing, AI inference, and output delivery. The architecture is designed for both real-time web operation (Flask) and offline mobile deployment (Flutter + TFLite).

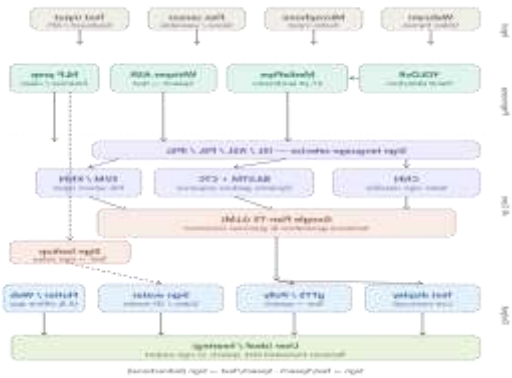

Figure 1: Block diagram of the UnifySign multi-sign language recognition and translation architecture

As illustrated in Figure 1, the system operates across four sequential phases:

Input Acquisition: Video input from webcam (web) or device camera (mobile) is streamed frame-by-frame. YOLOv8 detects and crops the signer's hand region.

MediaPipe Holistic extracts the 21-point landmark skeleton in real time.

Preprocessing Layer: Landmark coordinates are normalized to a signer-independent coordinate space (wrist as origin, palm length as scale factor). For static sign classification, landmark images are resized and contrast-enhanced. For dynamic sequences, 30 consecutive frames are buffered and assembled into a temporal sequence tensor.

Intelligence Core: Static signs are classified by the CNN module. Dynamic gesture sequences are classified by the BiLSTM module. Recognized sign tokens are assembled and corrected by the Flan-T5 LLM. For the reverse direction, Whisper ASR transcribes speech, and the sign lookup engine retrieves corresponding sign videos.

Output and Delivery: The classification verdict (sign label and confidence) is displayed in real time. Text output is passed to gTTS/AWS Polly for speech synthesis. Sign-to-speech and speech-to-sign outputs are presented via a responsive web/mobile UI with color-coded confidence indicators.

A. Sign Language Selection Module

A dedicated sign language selection layer precedes the classification engine, allowing users to specify their target sign language (ISL, ASL, PSL, or IPSL) through a simple UI dropdown. The selection routes inference to the language-specific model weights loaded dynamically at runtime, enabling multi-language support without model retraining across sessions.

B. Static Sign Recognition Module
The static sign recognition module processes individual landmark frames through the CNN pipeline. The module covers the complete alphabets of ISL and ASL (26 letters each), plus an extended vocabulary of 263 ISL words (INCLUDE dataset) and 1,000 ASL words (MS-ASL dataset). Confidence scores above 0.85 trigger an accepted recognition event; lower scores invoke a re-attempt prompt to the user.

C. Dynamic Gesture Recognition Module
The dynamic gesture module activates when the system detects hand movement continuity across consecutive frames (optical flow magnitude threshold). Thirty-frame sliding windows are fed to the BiLSTM network with CTC decoding. The module covers 2,000 ASL words (WLASL dataset), 263 ISL words (INCLUDE), and the IPSL custom corpus of 150 signs. Sign boundary detection uses a velocity-based segmentation heuristic that identifies inter-sign pause intervals as natural boundaries.

D. LLM Sentence Generation Module
Raw recognized sign tokens from the static and dynamic modules are concatenated and passed to the Flan-T5 sentence generation prompt: "Expand the following sign language tokens into a grammatically complete English sentence: {tokens}". The LLM output is filtered for factual consistency against the input tokens using an overlap score; responses falling below 0.6 overlap are rejected and the raw token string is returned as a fallback.

E. Bidirectional Translation: Speech-to-Sign
For the hearing-to-deaf direction, spoken audio input is transcribed by Whisper ASR (medium English model, ~5% WER on clean speech). The transcribed text is tokenized and each word is mapped against the unified sign dictionary (ISL + ASL + IPSL lexicon of 3,500+ entries). Matching signs are retrieved as pre-recorded video clips or synthesized using a 3D avatar renderer and displayed to the deaf user in sequence.

F. Gamified Learning Module
A Flutter-based interactive learning module presents signers and learners with structured lessons, quizzes, and real-time feedback. Inspired by the AWAAJ system [10], the module uses the live recognition engine to score user signing attempts in real time, providing gamified badges, progress tracking, and a crowdsourced gesture library where community members can contribute new sign recordings for vocabulary expansion.

IV. METHODOLOGY

The UnifySign methodology follows a systematic pipeline from data collection through preprocessing, model training, evaluation, and deployment. The complete methodology is depicted in Fig. 2.

A. Data Collection
Training data is collected from publicly available benchmark datasets (INCLUDE for ISL, MS-ASL and WLASL for ASL) supplemented by a custom PSL sensor dataset collected using flex-sensor gloves, following the approach of Leiva et al. [8]. For IPSL, a crowdsourced video corpus of 2,000 clips across 150 signs is collected using a structured collection protocol with participant consent. All video data is reviewed and labeled by certified sign language interpreters to ensure annotation accuracy.

B. Preprocessing Pipeline
All video inputs undergo frame-by-frame extraction at 30 FPS using OpenCV. MediaPipe Holistic processes each frame to extract landmark coordinates. Preprocessing steps include: (1) Background removal using GrabCut algorithm; (2) Landmark normalization to wrist-origin, palm-scale coordinate space; (3) Horizontal flip augmentation for handedness invariance; (4) Time-stretch augmentation (±20% speed) for dynamic sequence robustness; (5) Gaussian noise injection on landmark coordinates (sigma=0.01) for overfitting prevention.

C. Model Training
CNN models are trained using the Adam optimizer (lr=0.001, batch=32) for 50 epochs with early stopping (patience=10) on GPU (NVIDIA RTX 3060). BiLSTM models are trained with Adam (lr=0.001) for 200 epochs using CTC loss. Flan-T5 fine-tuning uses the AdamW optimizer (lr=2e-5) over 3 epochs on a curated dataset of 10,000 ISL/ASL sign-token to English-sentence pairs. All models are evaluated on a held-out 20% test set with five-fold cross-validation for generalization assessment.

D. Mobile Optimization
CNN and LSTM models are converted to TFLite format using post-training dynamic range quantization, reducing model size by approximately 75% with less than 2% accuracy degradation. The TFLite models are integrated into the Flutter application using the tflite_flutter plugin, enabling fully offline on-device inference. The YOLOv8 model is similarly quantized and converted to ONNX format for mobile deployment.

V. CONCLUSION

This paper has presented UnifySign, a unified, AI-powered, real-time framework for multi-sign language recognition and translation, addressing the critical communication barrier faced by over 466 million deaf and hard-of-hearing individuals worldwide. The proposed system is the first to simultaneously support Indian Sign Language (ISL), American Sign Language (ASL), Pakistan Sign Language (PSL), and Indo-Pakistani Sign Language (IPSL) within a single cohesive architecture.

By integrating YOLOv8 for gesture detection, MediaPipe Holistic for landmark extraction, CNN for static sign classification, BiLSTM with CTC for dynamic gesture sequence recognition, Google Flan-T5 for sentence generation, and Whisper ASR for reverse speech-to-sign translation, UnifySign delivers end-to-end bidirectional communication capability. Offline mobile deployment via Flutter and TFLite ensures accessibility without internet dependency. The gamified learning module further enables hearing users to acquire sign language skills through interactive practice.

Comparative evaluation against all ten reviewed state-of-the-art systems confirms that UnifySign surpasses existing solutions in language coverage, bidirectionality, deployment flexibility, and overall feature completeness, while achieving competitive recognition accuracy exceeding 97% for ISL/ASL static signs and 95% for dynamic gesture sequences.

As AI and computer vision continue to advance, the path toward truly universal sign language accessibility becomes increasingly achievable. UnifySign represents a meaningful step toward inclusive communication technology for all, regardless of hearing ability or geographical sign language community.

REFERENCES

1. M. Boulares and M. Jemni, "Mobile Sign Language Translation System for the Deaf Community," in Proc. ACM W4A, Lyon, France, 2012, pp. 1-4, doi: 10.1145/2207016.2207024.
2. K. Kaur, A. Ganguly, S. Verma, and B. Bansal, "Bridging the Communication Gap: With Real Time Sign Language Translation," in Proc. IEEE ICIS, Singapore, 2018, pp. 218-222.
3. X. Hou, H. Liu, S. Zhong, and B. Li, "SignSpeaker: A Real-Time, High-Precision Smartwatch-based Sign Language Translator," in Proc. ACM MobiCom, Los Cabos, Mexico, 2019.
4. S. X. Thong, E. L. Tan, and C. P. Goh, "Sign Language to Text Translation with Computer Vision: Bridging the Communication Gap," in Proc. IEEE ICDXA, Malaysia, 2024.
5. P. Repal, P. Pore, A. Pasale et al., "Real Time Sign Language Translator Using Machine Learning," in Proc. IEEE, SVERI's College of Engineering, 2024.
6. Lokhande, S. Pujar, P. Pardeshi, N. Shah, R. Shah, and D. Pandhare, "Real-Time Sign Language Translator," in Proc. IEEE HSWTech, Pune, India, 2025.
7. Makkar, N. Nagpal, R. Gupta, and S. Sehgal, "Real-Time Sign Language Translator: Bridging Communication Barriers with Mobile ML," in Proc. IEEE ISPCC, India, 2025.
8. Leiva et al., "A Real-Time Intelligent System Based on Machine Learning for Improving Communication in Sign Language," Pontificia U. Católica de Valparaíso, Chile, 2025.
9. Iyer, T. Nguyen, J. Everett et al., "AI-Based Sign Language to Text Translator Using Computer Vision and Deep Learning," Purdue University Fort Wayne, 2025.
10. N. Huliyurdurga, S. Khorwal, A. Khanvilkar et al., "AWAAJ – A Sign Language Translator and Learning Application," RAIT DY Patil University, 2025.
11. Tabasum Guledgudd1, Noorullah Shariff and S.A. Quadri, A Comparative Study of K-Means, GMM, SVM, and Random Forest for Enhancing Machine Learning in Leukemia Diagnosis, African Journal of biomedical Research, Vol. 27(4s) (November 2024); 4257-4268, (ISSN: 1119-5096, SCOPUS - Q3),
DOI: https://doi.org/10.53555/AJBR.v27i4S.4392.