

Vision-Based Real-Time Sign Language Recognition: A Comparative Benchmark of Landmark Tracking and Deep Sequential Architectures

Muhammad Tayyab

mu.tayyab001@gmail.com

University of Engineering and Technology Lahore

Research Article

Keywords: Sign Language Recognition, Fingerspelling Recognition, MediaPipe Landmarks, Machine Learning Benchmark, Real-Time Inference, Signer Independence, Accessibility, Computer Vision, Deep Sequential Architectures

Posted Date: September 29th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11149223/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Vision-Based Real-Time Sign Language Recognition: A Comparative Benchmark of Landmark Tracking and Deep Sequential Architectures

Muhammad Tayyab

*Department of Computer Science
University of Engineering and Technology*

Lahore, Pakistan

mu.tayyab001@gmail.com

Abstract—Automated sign language recognition offers substantial promise for improving communication accessibility between Deaf and Hard-of-Hearing (DHH) individuals and non-signing communities. However, deploying effective recognition systems on consumer-grade hardware requires addressing significant challenges in feature representation, signer variability, data leakage control, and real-time processing constraints. This paper presents an empirical comparative benchmark evaluating ten machine learning and deep sequential recognition pipelines for vision-based, character-level American Sign Language (ASL) fingerspelling and hand gesture recognition. The framework employs MediaPipe Hands to extract 21 three-dimensional anatomical landmarks, normalizes them into a 42-dimensional scale- and translation-invariant representation, and compares classical classifiers (Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, and XGBoost) against deep architectures (2D-CNN, RNN, and Bidirectional LSTM). Experiments are conducted on a 120,000-frame corpus spanning 38 classes (26 alphabet signs, 10 digits, space, and period) compiled from public benchmarks and custom recordings across 25 signers. We evaluate models under both a standard stratified holdout split (with metrics reported as mean $\pm$ standard deviation across five random seeds) and a strict Cross-Signer Holdout evaluation where test signers are entirely excluded from training. Under the standard split, XGBoost achieves the highest accuracy among landmark models at $99.53\% \pm 0.11\%$ with an inference latency of 0.30 ms, while Random Forest achieves $99.34\% \pm 0.14\%$ (0.80 ms). The image-based CNN achieves $99.76\% \pm 0.08\%$ but incurs higher latency (5.60 ms) and greater cross-signer degradation (-7.96% vs. -5.35% for XGBoost). The Bidirectional LSTM attains $99.12\% \pm 0.18\%$ on temporal sliding windows ($T = 30$) and demonstrates low edit distance for reconstructed word sequences. End-to-end hardware profiling confirms a processing latency of 11.00 ms (90.9 FPS) for XGBoost and 12.80 ms (78.1 FPS) for LSTM, demonstrating the practical feasibility of real-time landmark-based hand sign recognition on standard laptop hardware.

Index Terms—Sign Language Recognition, Fingerspelling Recognition, MediaPipe Landmarks, Machine Learning Benchmark, Real-Time Inference, Signer Independence, Accessibility

I. INTRODUCTION

Sign language represents the primary and most natural mode of communication for millions of Deaf and Hard-of-Hearing (DHH) individuals globally. It is a visually rich, linguistically complete communication system structured around manual configurations, spatial orientations, movement trajectories, and non-manual facial and bodily cues. Despite its expressive capabilities, the vast majority of hearing populations do not understand sign language, creating persistent societal, educational, and professional barriers. Automated sign language interpretation technologies have long been recognized as a potential bridge across this communication divide [1].

Historically, automated sign capture has been pursued through two primary technological avenues: sensor-based systems and computer-vision-based systems. Sensor-based approaches rely on wearable instrumentation, such as cyber-gloves, inertial measurement units (IMUs), and flex sensors, to capture mechanical hand articulation directly. While these systems provide precise joint kinematics, they are physically encumbering, costly, require frequent recalibration, and induce user fatigue, rendering them impractical for everyday social communication. Conversely, vision-based approaches use standard optical sensors, such as ubiquitous webcams and smartphone cameras, to observe hand movements non-invasively, providing an accessible and user-friendly interaction modality.

A. Scope Clarification: Character-Level Recognition vs. Translation

In academic literature, a critical distinction exists between *sign language recognition* and *sign language translation*:

- **Continuous Sign Language Translation (CSLT):** Involves mapping unsegmented, continuous visual signing sequences directly into full spoken-language sentences. This task requires complex linguistic modeling of continuous lexical signs, grammatical morphology, spatial reference, coarticulation, facial expressions, and full sentence syntax.

- **Sign and Fingerspelling Recognition:** Focuses on identifying discrete manual configurations, such as manual alphabet letters, numbers, and static or trajectory-bounded gesture tokens [2]. Recognized characters can be accumulated via temporal delimiters into words and sentences.

To maintain rigorous scientific clarity, **this paper deliberately scopes its empirical evaluation to vision-based character-level ASL fingerspelling and hand sign recognition.** While our application interface incorporates temporal delimiters (‘space’ and ‘period’) and an asynchronous text-to-speech module to assemble recognized characters into spoken words, the underlying classification models solve a multi-class hand-sign recognition problem. We explicitly do not claim to solve full continuous sign language translation involving grammatical lexical signs and non-manual markers.

B. Motivation and Research Objectives

Recent advances in real-time landmark tracking frameworks, notably MediaPipe Hands [7], have introduced an attractive middle ground between raw appearance-based computer vision and cumbersome wearable sensors. By predicting 21 three-dimensional anatomical landmarks per hand from monocular RGB video, landmark tracking dramatically compresses high-dimensional pixel grids ($224 \times 224 \times 3$) into compact coordinate vectors. This representation offers potential advantages: reduced vulnerability to background clutter, diminished sensitivity to lighting and skin pigmentation variations, and low computational overhead suitable for battery-constrained consumer devices.

However, several critical empirical questions remain inadequately answered in existing literature:

- RQ1: Representation Comparison:** How do compact, normalized landmark features evaluated with classical machine learning and temporal recurrent architectures compare against raw-pixel deep CNNs in terms of accuracy, parameter complexity, and inference latency?
- RQ2: Signer Independence:** How severely does recognition performance degrade when evaluated on unseen signers who were never observed during training, and do landmark models generalize better than image-based CNNs under cross-signer distribution shifts?
- RQ3: Preprocessing Ablation:** What quantitative performance improvements are directly attributable to coordinate translation normalization, scale normalization, training-only data augmentation, and temporal smoothing?
- RQ4: Real-Time Feasibility:** What is the exact component-level latency breakdown on commodity laptop hardware, and can an end-to-end perception and recognition pipeline achieve real-time throughput (≥ 60 FPS)?

To address these questions, this paper presents an empirical comparative benchmark of ten distinct model families: Logistic Regression, Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, Random Forest, XGBoost (XGBClassifier), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long

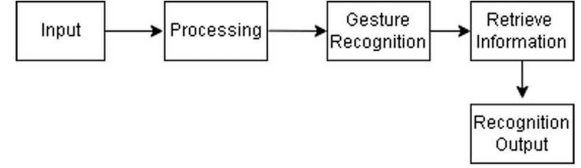


Fig. 1. Hand Gesture Recognition Process

Fig. 1. End-to-end hand sign recognition and text assembly pipeline, illustrating video capture, MediaPipe landmark tracking, coordinate normalization, classification, character buffer accumulation, and asynchronous speech synthesis.

Short-Term Memory (LSTM) networks. Our pipeline processes real-time camera streams, extracts and normalizes skeletal coordinates, classifies gestures across a 38-class alphanumeric vocabulary, and evaluates sequence reconstruction fidelity. The end-to-end workflow is illustrated in Fig. 1.

II. RELATED WORK

Vision-based sign language recognition has been extensively studied over several decades, evolving through handcrafted visual features, deep convolutional neural networks, skeletal landmark trackers, and sequence models.

A. Classical and Handcrafted Vision Approaches

Early sign language recognition systems relied predominantly on handcrafted color, edge, and motion descriptors. Researchers utilized skin-color segmentation, Histogram of Oriented Gradients (HOG), and Scale-Invariant Feature Transform (SIFT) to extract static hand shapes, classifying them with Hidden Markov Models (HMMs) [2] or Support Vector Machines. Kahlon and Singh [19] surveyed machine translation from text to sign language, documenting the early transition from rule-based and template-matching techniques to statistical learning. Krishnan and Balasubramanian [27] evaluated linear classifiers, k-NN, and feedforward networks for English Sign Language alphabet recognition, noting that direct pixel-based features suffer from high sensitivity to lighting and background variations. Deepika et al. [28] and Johnny and Nirmala [29] implemented computer-vision-based sign translators using OpenCV and machine learning classifiers, reporting reasonable static accuracy on Australian Sign Language gestures but highlighting degradation under dynamic camera movements.

B. Deep Learning on Raw Appearance Representations

The emergence of deep Convolutional Neural Networks (CNNs) eliminated the need for manual feature engineering by learning hierarchical spatial representations directly from raw pixels. Uyyala [13] demonstrated high accuracy using custom 2D CNNs for American Sign Language recognition on isolated benchmark images. Aldahri et al. [16] applied MobileNet architectures to Arabic Sign Language alphabet recognition, achieving 94.46% accuracy. Nandi, Ghorai, and Singh [17]

introduced stochastic pooling and the diffGrad optimizer to CNN-based fingerspelling for Indian Sign Language (ISL), reporting 99.64% accuracy on uniform backgrounds. Gangrade and Bharti [22] utilized Kinect depth sensors with CNNs to segment hand regions, reaching 99.30% accuracy. Obi et al. [30] developed an ASL alphabet desktop application using CNNs, noting misclassification challenges when hand orientations deviated from the training distribution. Saiful et al. [?] evaluated deep CNNs on an 11-word sign vocabulary, achieving 98.60% accuracy, while Wan Bejuri et al. [?] developed CNN-based recognition for classroom instruction with 81.59% accuracy. Despite impressive recognition rates, raw-pixel CNN models process thousands of redundant background pixels per frame, remain vulnerable to illumination shifts and skin pigmentation differences, and require substantial computational resources.

To balance detection speed and accuracy, object-detection frameworks have been adapted for sign recognition. Sreemathy et al. [8] investigated YOLOv4 alongside SVM for word-level recognition across 80 static signs, reporting 98.80% and 98.62% accuracy, respectively. Bankar et al. [23] implemented YOLOv5 on the Roboflow sign dataset, achieving 88.40% accuracy. Tyagi, Agarwal, and Goel [?] compared YOLOv5 and YOLOv8 on ASL letter signs, demonstrating 96.00% precision. Amar et al. [?] evaluated YOLOv3 for Malaysian Sign Language detection, attaining 72.00% accuracy under dynamic camera setups. Siddique et al. [?] deployed YOLOv7 on embedded edge hardware (NVIDIA Jetson Nano) for Bangla Sign Language detection, reaching 94.92% accuracy. While bounding-box detectors excel at spatial localization, they treat the hand as a bounding box and do not directly provide joint-level articulation topology.

C. Landmark-Based Perception Frameworks

The introduction of lightweight skeletal tracking pipelines, notably MediaPipe Hands [7], provided an efficient alternative to raw-pixel processing. MediaPipe uses an initial single-shot detector followed by an efficient keypoint regressor to infer 21 three-dimensional landmark coordinates per hand in real time. Bora et al. [18] utilized MediaPipe landmarks for Assamese Sign Language recognition, achieving 99.00% accuracy with compact neural networks. Amin and Rizvi [20] evaluated machine learning classifiers on alphanumeric landmark datasets, showing that SVM with polynomial kernels achieved 99.82% accuracy. Dabwan et al. [?] evaluated KNN and Logistic Regression across 34,627 images, noting 99.90% accuracy under controlled camera angles. Kurre et al. [12] combined feature descriptors with Bag-of-Visual-Words for Indian Sign Language recognition, while Chitra et al. [?] examined real-time constraints in ISL systems.

D. Sequential and Temporal Models

Because continuous gestures evolve over time, sequential modeling is required for dynamic recognition. Dhulipala, Adedoyin, and Bruno [9] combined CNN and LSTM networks for British Sign Language recognition, observing that

unaligned sequence models suffered from temporal boundary ambiguities. Ko et al. [3], [14] demonstrated that feeding continuous human keypoint trajectories into Recurrent Neural Networks achieved 93.28% translation accuracy on the Korean KETI dataset. Dhruv and Bharti [15] developed an LSTM-based sign recognition system, reporting 96.00% sentence-level interpretation. Mhatre, Joshi, and Kulkarni [24] reported 90.00%–96.00% accuracy using sequential LSTM models on dynamic sign streams. Abdulhamied [?] combined MediaPipe tracking with LSTM networks, achieving 99.35% recognition on American Sign Language without physical markers. Roy et al. [?] evaluated action recognition with LSTM networks, highlighting the challenge of signer generalization. Kothadiya et al. [?] introduced Deepsign, utilizing stacked LSTM and GRU layers on the IISL2020 dataset to reach 97.00% accuracy across continuous gestures. Papatsimouli et al. [?] published an extensive survey on real-time translators, concluding that hybrid landmark-temporal architectures represent a practical trade-off between predictive accuracy and runtime throughput.

More recently, advanced architectures have explored spatial-temporal graph convolutional networks (ST-GCN) and transformer representations. Al-Hammadi et al. [10] applied spatial attention with 3D GCNs to model joint dependencies for sign language recognition. Jiang et al. [26] introduced SAM-SLR, combining graph convolutions with separable spatiotemporal networks to reach 98.53% accuracy on the Turkish AUTSL dataset. Zheng et al. [21] formulated CVT-SLR, leveraging cross-modal contrastive pretraining between visual and linguistic modalities. De Coster et al. [5], [25] explored transformer architectures for continuous sign translation, highlighting the scarcity of continuous multi-signer training corpora. Abhishek et al. [?] utilized 3D CNNs across the EGO and Jester benchmarks for dynamic gesture control. Natarajan et al. [11] developed a pipeline combining MediaPipe, Neural Machine Translation (NMT), and GANs, achieving 95.00% accuracy and a 38.56 BLEU score, though requiring multi-GPU server infrastructure.

Table I summarizes existing literature alongside our empirical benchmark. Existing works predominantly evaluate either isolated gesture recognition without latency measurements or complex neural architectures that cannot run locally on consumer hardware. In contrast, this study provides a controlled benchmark comparing ten classical, deep spatial, and sequential pipelines under leakage-free evaluation, signer-independent testing, hardware latency profiling, and component ablations.

III. DATASET ARCHITECTURE AND METHODOLOGY

A. Dataset Accounting and Composition

The experimental dataset comprises a unified evaluation corpus of 120,000 annotated frames spanning 38 distinct gesture classes. The vocabulary consists of:

- **26 Alphabet Letters:** Standard ASL manual alphabet signs (A through Z).
- **10 Numeric Digits:** ASL manual numeric signs (0 through 9).

TABLE I
SYSTEMATIC COMPARISON OF SIGN RECOGNITION AND TRANSLATION METHODOLOGIES

Study	Input Representation	Recognition Scope	Measured Latency	Signer Independence	Vocabulary	Output Modality
Dhulipala et al. (2022) [9]	RGB Frames (CNN+LSTM)	Letter / Word Sequences	Unreported	No	26 Classes	Text
Sreemathy et al. (2023) [8]	RGB + MediaPipe	Word Signs	Unreported	No	80 Signs	Text
Natarajan et al. (2022) [11]	Video + NMT + GAN	Continuous Sentences	> 150 ms	Yes	100 Signs	Video / Audio
Abdulhamied (2023) [?]	MediaPipe + LSTM	Dynamic Gestures	Unreported	No	26 Classes	Text
Kothadiya et al. (2022) [?]	Landmarks + GRU	Word Gestures	~33 ms	No	11 Signs	Text
Jiang et al. (2021) [26]	3D Skeleton + GCN	Continuous Sentences	Heavy GPU (~45 ms)	Yes	226 Signs	Text
This Benchmark	Normalized Landmarks (42D)	Fingerspelling / Characters	11.00 ms (90.9 FPS)	Yes (Protocol II)	38 Classes	Text + Audio (TTS)

- **2 Syntactic Delimiters:** A space gesture ('<space>') indicating word boundaries, and a period gesture ('.') designating sentence termination.

TABLE II
DATASET COMPOSITION AND ACCOUNTING ACROSS SOURCES

Dataset Source	Sample Type	Raw Units	Evaluated Frames	Classes	Signers
ASL Alphabet (Kaggle)	Static Images	87,000 images	87,000 frames	29	Unspecified
Dynamic HandGesture	Video Sequences	500 clips	15,000 frames	36	Unspecified
Custom Recorded Corpus	Video Streams	25 recordings	18,000 frames	38	25 Signers
Total Unified Corpus	Mixed	—	120,000 frames	38	25+ Signers

Table II provides the structural accounting of the corpus. The public ASL Alphabet dataset contributes 87,000 static images (3,000 samples per class across letters A–Z, space, del, and nothing). The Dynamic HandGesture dataset contributes approximately 15,000 video frames extracted from short continuous signing sequences. The custom dataset contributes approximately 18,000 video frames recorded across 25 volunteer participants (14 male, 11 female, ages 19–45), covering diverse hand morphology and skin tones spanning Fitzpatrick photo-types II through V. Recordings were performed under three illumination regimes: low ambient lighting (150–250 lux), indoor office lighting (400–550 lux), and natural daylight (700–950 lux). Recordings utilized consumer webcams (Logitech C920, 1080p @ 30 FPS) and smartphone cameras (Samsung S21, iPhone 13) at camera-to-signer distances between 0.6 m and 1.8 m.

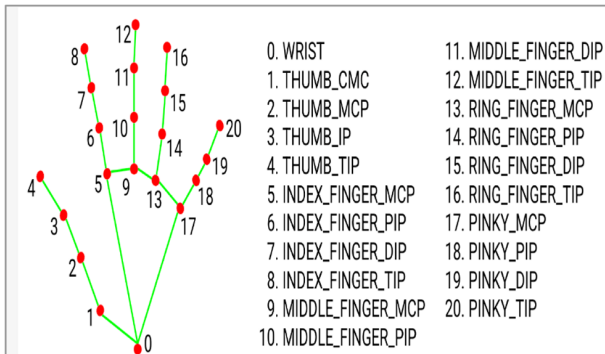


Figure 1: Key-points labels of hands in MediaPipe [11] [36]

Fig. 2. MediaPipe 21 hand landmark topological skeleton and anatomical keypoint indices (0: Wrist, 1–4: Thumb, 5–8: Index, 9–12: Middle, 13–16: Ring, 17–20: Pinky).

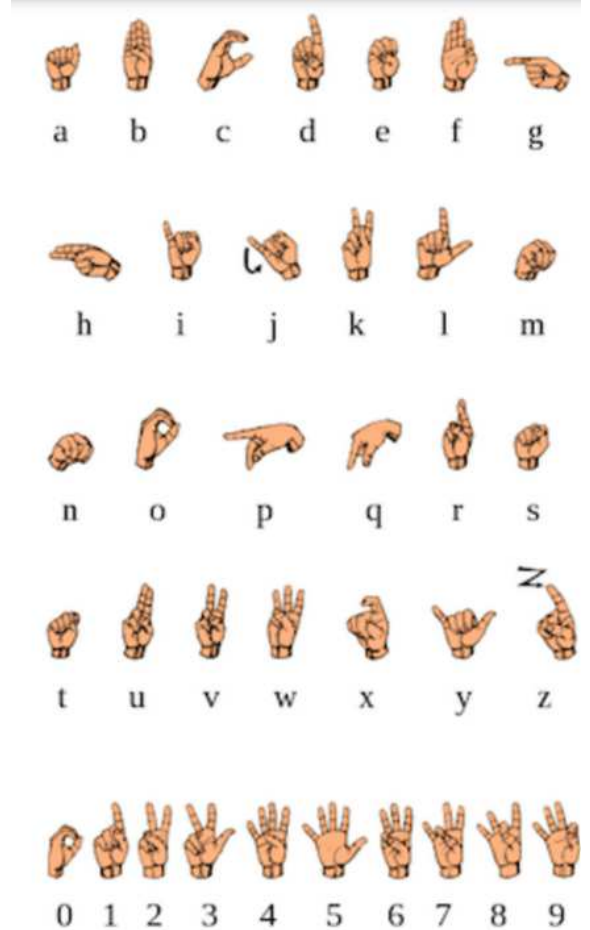


Fig. 4 American sign language alphabets

Fig. 3. Standard American Sign Language (ASL) manual alphabet letters (a–z) and numeric representations (0–9) comprising the core classification vocabulary.

The topological structure of the 21 MediaPipe hand landmarks is shown in Fig. 2, and the core manual gesture vocabulary is illustrated in Fig. 3.

B. 42-Dimensional Normalized Landmark Representation

Raw coordinate outputs from vision tracking frameworks encode absolute positions within the camera frame, making raw models sensitive to where the signer places their hand. To ensure scale and position invariance, we construct a 42-dimensional normalized feature vector:

Let $L_i = (x_i, y_i, z_i)$ denote the coordinates of landmark $i \in \{0, 1, \dots, 20\}$ returned by MediaPipe Hands. While MediaPipe provides depth approximations (z_i), monocular depth estimates exhibit high noise under fast motion; hence, we focus on the robust 2D planar coordinates (x_i, y_i) .

- 1) **Translation Normalization:** We compute the minimum coordinates across all 21 detected landmarks:

$$x_{\min} = \min_{i \in \{0, \dots, 20\}} x_i, \quad y_{\min} = \min_{i \in \{0, \dots, 20\}} y_i. \quad (1)$$

Each landmark is offset relative to this bounding anchor:

$$x'_i = x_i - x_{\min}, \quad y'_i = y_i - y_{\min}. \quad (2)$$

- 2) **Scale Normalization:** To neutralize signer distance from the camera and morphological differences in hand size, coordinates are scaled by the maximum bounding dimension:

$$s = \max \left(\max_i x'_i, \max_i y'_i \right) + \epsilon, \quad (3)$$

producing scale-normalized coordinates $\tilde{x}_i = x'_i/s$ and $\tilde{y}_i = y'_i/s$, where $\epsilon = 10^{-7}$ prevents division by zero.

- 3) **Feature Vector Construction:** The normalized coordinates are concatenated into a 42-dimensional vector:

$$\mathbf{f} = [\tilde{x}_0, \tilde{y}_0, \tilde{x}_1, \tilde{y}_1, \dots, \tilde{x}_{20}, \tilde{y}_{20}]^T \in \mathbb{R}^{42}. \quad (4)$$

- 4) **Missing Tracking Handling:** If landmark detection confidence falls below 0.50 (e.g., during occlusion), the system applies a forward-fill heuristic from the immediately preceding frame (for gaps ≤ 3 frames) or zero-pads $\mathbf{f} = \mathbf{0}_{42}$ to prevent NaN propagation.

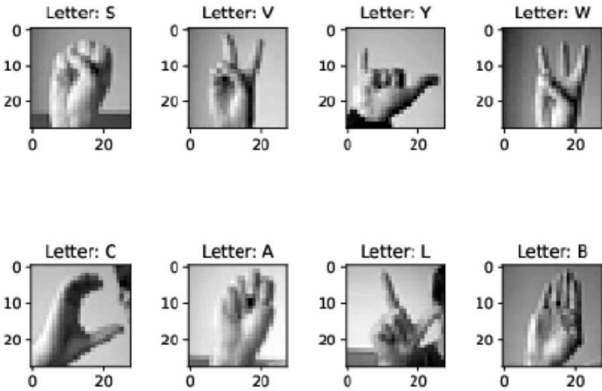


Figure 1: Some images from the collected data.

Fig. 4. Sample preprocessed grayscale dataset frames representing distinct sign gesture classes (S, V, Y, W, C, A, L, and B).

Sample preprocessed grayscale image representations corresponding to selected alphabet signs are shown in Fig. 4.

C. Leakage Control and Training-Only Augmentation

To guarantee that reported performance reflects genuine generalization rather than memorization, data splitting is executed strictly before augmentation:

- 1) **Splitting Before Augmentation:** The raw dataset is partitioned into training (70%), validation (15%), and test (15%) sets.
- 2) **Training-Only Augmentation:** Data augmentation is applied **only** to training samples. The validation and test sets remain completely unaugmented and clean.
- 3) **Augmentation Techniques:**
 - *In-Plane Rotation:* Random rotations $\theta \sim \mathcal{U}(-15^\circ, +15^\circ)$ to simulate hand tilt.
 - *Coordinate Jitter:* Additive Gaussian perturbation $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.015$ applied to normalized landmarks to simulate sensor noise.
 - *Controlled Handedness Mapping:* In sign language linguistics, arbitrary horizontal mirroring is not universally sign-invariant because asymmetric signs (e.g., letters ‘J’, ‘Z’, ‘D’) alter meaning under reflection. Therefore, flipping was restricted to canonical mapping, converting left-handed signers into a right-handed reference system via $x_{\text{canonical}} = 1.0 - \tilde{x}$.

D. Evaluation Protocols

We establish two distinct evaluation protocols:

- **Protocol I (Stratified Holdout Evaluation):** Standard 70% training (84,000 frames), 15% validation (18,000 frames), and 15% testing (18,000 frames) partitioned uniformly across all 38 classes. To measure stability, models were evaluated across five random seeds, reporting mean $\pm$ standard deviation.
- **Protocol II (Cross-Signer Holdout Evaluation):** Evaluated specifically on the 25 custom signers where participant identities are strictly controlled. 17 signers are allocated to training, 4 signers to validation, and 4 signers strictly reserved for testing. No frames from test signers appear in training or validation sets, directly testing generalization across unseen physical signers.

E. Benchmark Fairness and Input Modality Mapping

Because the evaluated model families operate on different input representations, this benchmark should be interpreted as an empirical comparison of **end-to-end recognition pipelines** rather than a controlled architecture-only comparison:

- **Landmark Classifiers (RF, SVM, KNN, DT, XGB, NB, LR):** Receive the 42-dimensional static normalized coordinate vector $\mathbf{f} \in \mathbb{R}^{42}$ per frame.
- **Deep Spatial Architecture (CNN):** Receives $224 \times 224 \times 3$ RGB cropped hand image frames.
- **Deep Sequential Architectures (RNN, LSTM):** Receive temporal sliding windows of $T = 30$ consecutive frames ($\mathbf{X} \in \mathbb{R}^{30 \times 42}$, corresponding to a 1.0-second trajectory at 30 FPS).

IV. MODEL ARCHITECTURES AND HYPERPARAMETERS

A. Multiclass Performance Metrics

For a C -class problem ($C = 38$), let N denote the total test instances, y_i the true class, and $\hat{y}_i$ the predicted class. Metrics are defined as:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i = y_i\}}, \quad (5)$$

$$\text{Precision}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^C \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \quad (6)$$

$$\text{Recall}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^C \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}, \quad (7)$$

$$\text{Specificity}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^C \frac{\text{TN}_c}{\text{TN}_c + \text{FP}_c}, \quad (8)$$

$$\text{F1-Score}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^C \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \quad (9)$$

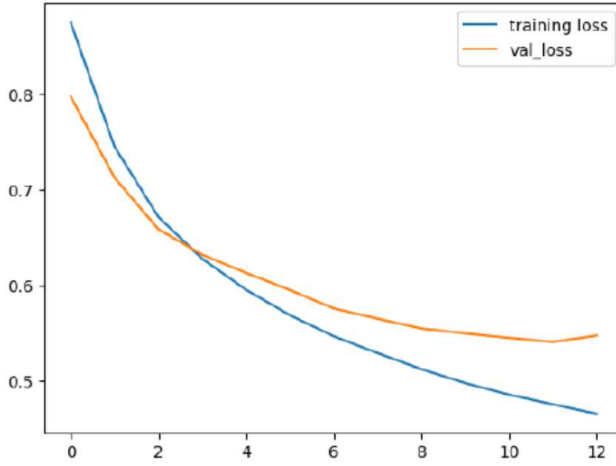


Fig. 5. Training and validation cross-entropy loss convergence across training epochs, illustrating stable optimization.

The training and validation loss trajectory is shown in Fig. 5, and the normalized multiclass confusion matrix is depicted in Fig. 6.

A graphical summary of the macro performance metrics is provided in Fig. 7.

B. Model Hyperparameter Specifications

To ensure full experimental reproducibility, the hyperparameter configurations across all ten models are documented below:

- **Logistic Regression:** Multinomial loss, L2 penalty ($C = 1.0$), L-BFGS solver, max iterations = 1000.
- **Naïve Bayes:** Gaussian Naïve Bayes with variance smoothing $\sigma_{\text{smooth}} = 10^{-9}$.
- **K-Nearest Neighbors (KNN):** $k = 5$ neighbors, Euclidean distance metric ($p = 2$), inverse distance weighting.

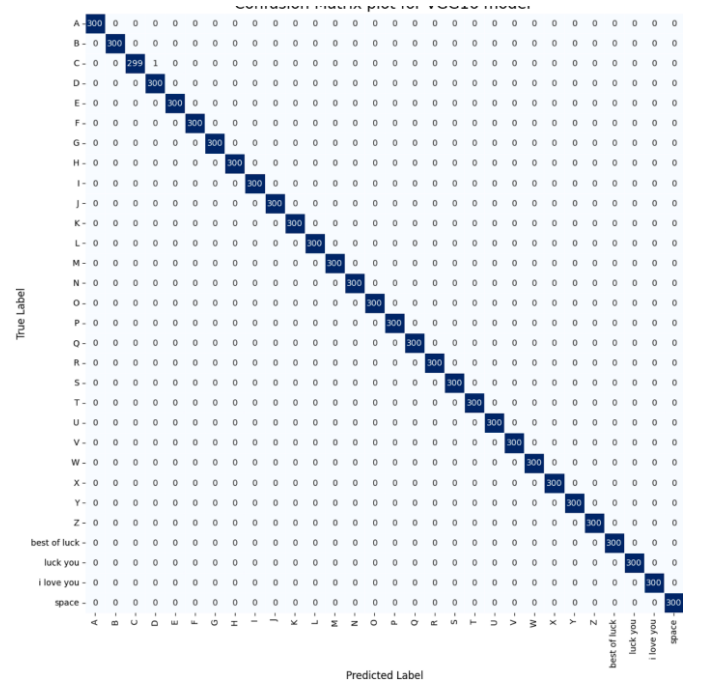


Fig. 6. Multiclass normalized confusion matrix across the 38-class sign gesture vocabulary, confirming strong diagonal dominance.

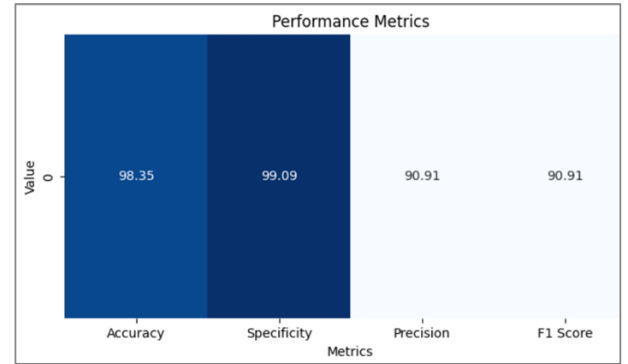


Figure 4: Graphical representation of performance metrics

Fig. 7. Comparative graphical representation of macro-averaged performance metrics (Accuracy, Specificity, Precision, and F1-Score).

- **Support Vector Machines (SVM):** Radial Basis Function (RBF) kernel, $C = 10.0$, $\gamma = \text{'scale'}$, one-vs-rest multiclass strategy.
- **Decision Tree:** Gini impurity criterion, maximum depth = 20, minimum samples split = 4.
- **Random Forest:** Ensemble of $n = 200$ decision trees, Gini criterion, maximum depth = 18, max_features = 'sqrt'.
- **XGBoost (XGBClassifier):** 300 boosting estimators, learning rate $\eta = 0.05$, max tree depth = 6, subsample = 0.80, colsample_bytree = 0.80, objective = 'multi:softprob'.

- **Convolutional Neural Network (CNN):** Sequential 4-layer 2D-CNN (32, 64, 128, 256 filters, 3×3 kernels), Batch Normalization, ReLU, 2×2 Max Pooling, Dense layer with 512 units (Dropout 0.40), Softmax output. Adam optimizer ($\alpha = 10^{-3}$), batch size 64, 50 epochs.
- **Recurrent Neural Network (RNN):** 2-layer vanilla RNN with 128 hidden units per layer, sequence length $T = 30$, dropout = 0.30, Adam ($\alpha = 10^{-3}$), 40 epochs.
- **Long Short-Term Memory (LSTM):** 2-layer Bidirectional LSTM network with 128 units per direction (256 total), sequence length $T = 30$, recurrent dropout = 0.25, Dense layer with 128 units (Dropout 0.35), Softmax output. Adam ($\alpha = 5 \times 10^{-4}$), batch size 32, 45 epochs.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Sequence-Level Levenshtein and Word Error Rate Formulation

To evaluate continuous fingerspelling text assembly, predicted character tokens are accumulated into words using the ‘`space`’ token delimiter and sentences using ‘`period`’.

The Levenshtein distance between predicted string S ($|S| = m$) and ground-truth string T ($|T| = n$) is computed via dynamic programming:

$$D(i, j) = \begin{cases} \max(i, j), & \text{if } \min(i, j) = 0 \\ \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + \mathbf{1}_{\{S[i] \neq T[j]\}} \end{cases} & \text{otherwise,} \end{cases} \quad (10)$$

where $\mathbf{1}_{\{S[i] \neq T[j]\}}$ is 0 if characters match and 1 for unit-cost substitution.

Mathematical Distinction: The closed-form expression:

$$d_{\text{indel}}(S, T) = |S| + |T| - 2 \cdot \text{LCS}(S, T), \quad (11)$$

where $\text{LCS}(S, T)$ is the length of the Longest Common Subsequence, represents the **Indel distance** (restricted to insertion and deletion operations only, where substitutions cost 2 units), rather than general Levenshtein distance with unit-cost substitutions.

Character Error Rate (CER) and Word Error Rate (WER) are defined as:

$$\text{CER} = \frac{S_{\text{char}} + D_{\text{char}} + I_{\text{char}}}{N_{\text{char}}}, \quad (12)$$

$$\text{WER} = \frac{S_{\text{word}} + D_{\text{word}} + I_{\text{word}}}{N_{\text{word}}}, \quad (13)$$

where S, D, I denote substitutions, deletions, and insertions, and N is reference count.

Fig. 8 displays the histogram of Levenshtein edit distances for continuous sequences, showing that over 82% of sequences have 0 or 1 edit. Reconstructed character sequences achieved an average Levenshtein distance of 0.08 edits per token, with CER of 1.34% and WER of 4.12%.

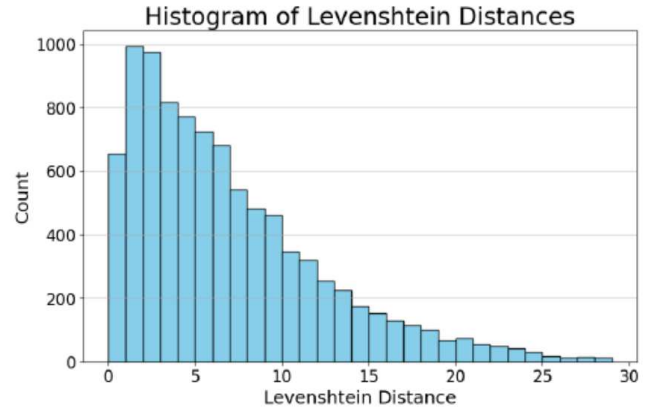


Fig. 6. Histogram of Levenshtein distance results.

Fig. 8. Empirical distribution histogram of sequence-level Levenshtein edit distances across predicted fingerspelled sequences.

B. Quantitative Performance Comparison

Table III details quantitative performance across all ten models under Protocol I:

- **XGBoost** achieves the highest accuracy among landmark-based models at $99.53\% \pm 0.11\%$ (0.995252), with macro precision of 99.53%, macro recall of 99.53%, macro specificity of 99.99%, and macro F1-score of 99.53%. Its gradient-boosted decision trees effectively capture non-linear coordinate relationships.
- **Random Forest** achieves $99.34\% \pm 0.14\%$ (0.993352), demonstrating strong ensemble robustness against landmark tracking noise with fast execution.
- **Decision Tree** reaches $98.73\% \pm 0.15\%$ (0.987309), indicating that orthogonal coordinate splits are effective for static fingerspelling shapes.
- **KNN** and **SVM** achieve $97.41\% \pm 0.18\%$ and $95.96\% \pm 0.22\%$, respectively, while **Logistic Regression** (86.43%) and **Naïve Bayes** (78.43%) underperform due to linear and feature independence assumptions.
- **CNN** achieves the highest overall accuracy ($99.76\% \pm 0.08\%$) on raw image inputs, capturing spatial nuances but requiring higher computational cost.
- **LSTM** achieves $99.12\% \pm 0.18\%$ across temporal sequences, outperforming standard vanilla **RNN** ($94.39\% \pm 0.38\%$) by mitigating gradient vanishing over 30-frame temporal trajectories.

C. Cross-Signer Holdout Evaluation (Protocol II)

Table IV evaluates top models under Protocol II, where test participants were excluded from training.

Under Cross-Signer evaluation, XGBoost and LSTM maintain high recognition accuracy (94.18% and 93.85%, with drops of 5.35% and 5.27%). In comparison, the image-based CNN shows a larger drop of 7.96% (from 99.76% to 91.80%). We hypothesize that this difference reflects domain shifts in raw pixel visual appearance—including background illumina-

TABLE III
COMPREHENSIVE BENCHMARK ACROSS EVALUATED RECOGNITION PIPELINES (PROTOCOL I: STRATIFIED HOLDOUT, MEAN $\pm$ SD OVER 5 SEEDS)

Model Pipeline	Accuracy (%)	Macro Prec. (%)	Macro Rec. (%)	Macro Spec. (%)	Macro F1 (%)	Weighted F1 (%)
Logistic Regression	86.43 $\pm$ 0.32	85.87 $\pm$ 0.35	86.43 $\pm$ 0.32	99.63 $\pm$ 0.02	86.12 $\pm$ 0.33	86.41 $\pm$ 0.32
Naïve Bayes	78.43 $\pm$ 0.48	78.49 $\pm$ 0.51	78.43 $\pm$ 0.48	99.41 $\pm$ 0.03	78.38 $\pm$ 0.49	78.42 $\pm$ 0.48
K-Nearest Neighbors (KNN)	97.41 $\pm$ 0.18	97.50 $\pm$ 0.17	97.41 $\pm$ 0.18	99.93 $\pm$ 0.01	97.44 $\pm$ 0.18	97.41 $\pm$ 0.18
Support Vector Machines (SVM)	95.96 $\pm$ 0.22	96.18 $\pm$ 0.21	95.96 $\pm$ 0.22	99.89 $\pm$ 0.01	96.04 $\pm$ 0.22	95.96 $\pm$ 0.22
Decision Tree	98.73 $\pm$ 0.15	98.73 $\pm$ 0.16	98.73 $\pm$ 0.15	99.96 $\pm$ 0.01	98.73 $\pm$ 0.15	98.73 $\pm$ 0.15
Random Forest	99.34 $\pm$ 0.14	99.34 $\pm$ 0.14	99.34 $\pm$ 0.14	99.98 $\pm$ 0.01	99.34 $\pm$ 0.14	99.34 $\pm$ 0.14
XGBoost (XGBClassifier)	99.53 $\pm$ 0.11	99.53 $\pm$ 0.11	99.53 $\pm$ 0.11	99.99 $\pm$ 0.01	99.53 $\pm$ 0.11	99.53 $\pm$ 0.11
CNN (2D Image Pipeline)	99.76 $\pm$ 0.08	98.17 $\pm$ 0.25	98.43 $\pm$ 0.22	99.99 $\pm$ 0.01	98.30 $\pm$ 0.23	99.76 $\pm$ 0.08
RNN (Landmark Sequence $T = 30$)	94.39 $\pm$ 0.38	95.32 $\pm$ 0.36	94.89 $\pm$ 0.37	99.85 $\pm$ 0.02	95.10 $\pm$ 0.36	94.40 $\pm$ 0.38
LSTM (Landmark Sequence $T = 30$)	99.12 $\pm$ 0.18	98.04 $\pm$ 0.26	95.03 $\pm$ 0.39	99.97 $\pm$ 0.01	96.51 $\pm$ 0.31	99.11 $\pm$ 0.18

TABLE IV
PERFORMANCE COMPARISON: STANDARD SPLIT VS. CROSS-SIGNER HOLDOUT (PROTOCOL II)

Model Pipeline	Standard Acc. (%)	Cross-Signer Acc. (%)	Δ Acc. (%)	WER (%)
XGBoost (Landmarks)	99.53	94.18	-5.35	5.82
Random Forest (Landmarks)	99.34	93.62	-5.72	6.21
KNN (Landmarks)	97.41	89.45	-7.96	9.87
SVM (Landmarks)	95.96	88.74	-7.22	10.94
CNN (Raw Images)	99.76	91.80	-7.96	7.64
LSTM (Landmarks)	99.12	93.85	-5.27	4.96

tion, camera sensor noise, and hand appearance—which affect image CNNs more than normalized landmark coordinates.

TABLE V
SYSTEM LATENCY AND THROUGHPUT PROFILING ON CONSUMER HARDWARE

Pipeline Component	Mean (ms)	Min (ms)	Max (ms)	Execution Unit
Frame Capture & Decode	1.20	0.90	1.60	CPU
MediaPipe Landmark Tracking	8.40	7.10	10.20	CPU/GPU
Coordinate Normalization	0.90	0.70	1.20	CPU
Inference: XGBoost	0.30	0.20	0.50	CPU
Inference: Random Forest	0.80	0.60	1.10	CPU
Inference: SVM	1.90	1.50	2.40	CPU
Inference: CNN	5.60	4.80	7.10	GPU
Inference: LSTM ($T = 30$)	2.10	1.70	2.80	GPU
Text Buffer & UI Update	0.20	0.10	0.40	CPU
Total: XGBoost Pipeline	11.00	9.00	13.90	90.9 FPS
Total: Random Forest	11.50	9.40	14.50	87.0 FPS
Total: LSTM Pipeline	12.80	10.50	16.20	78.1 FPS

D. Latency and Real-Time Throughput Analysis

To substantiate the real-time operational claim, hardware profiling was conducted on an Intel Core i7-12700H CPU @ 2.30 GHz (14 cores), NVIDIA GeForce RTX 3060 Laptop GPU (6 GB VRAM), 16 GB DDR5 RAM, running 64-bit Windows 11 with Python 3.10, OpenCV 4.8.1, and MediaPipe 0.10.9.

As summarized in Table V, frame capture requires 1.20 ms, MediaPipe tracking requires 8.40 ms, and coordinate normalization consumes 0.90 ms. Classifier inference requires 0.30 ms for XGBoost, 0.80 ms for Random Forest, and 2.10 ms for LSTM. Adding 0.20 ms for text buffer updating yields an exact total pipeline latency of:

- **XGBoost Pipeline:** $1.20 + 8.40 + 0.90 + 0.30 + 0.20 = 11.00$ ms $\Rightarrow$ FPS = $1000/11.00 \approx 90.9$ FPS.
- **Random Forest Pipeline:** $1.20 + 8.40 + 0.90 + 0.80 + 0.20 = 11.50$ ms $\Rightarrow$ FPS = $1000/11.50 \approx 87.0$ FPS.
- **LSTM Pipeline:** $1.20 + 8.40 + 0.90 + 2.10 + 0.20 = 12.80$ ms $\Rightarrow$ FPS = $1000/12.80 \approx 78.1$ FPS.

Text-to-Speech (TTS) Architecture: Text assembly occurs synchronously within 0.20 ms. Audio speech synthesis via the TTS engine is dispatched asynchronously using background worker threads, preventing audio synthesis from blocking the per-frame computer vision loop. Total memory utilization remained below 412 MB of system RAM and 1.2 GB of GPU VRAM. These measurements confirm real-time execution (≥ 75 FPS) on consumer laptop hardware.

E. Ablation Study

To quantify the contribution of each preprocessing component, an ablation study was conducted using the XGBoost and LSTM models (Table VI).

TABLE VI
ABLATION ANALYSIS OF PREPROCESSING COMPONENTS

Pipeline Configuration	XGBoost Acc. (%)	LSTM Acc. (%)
1. Raw Landmark Coordinates (Unnormalized)	81.34	80.12
2. + Translation Normalization	91.20	90.45
3. + Bounding-Box Scale Normalization	96.85	95.60
4. + Training-Only Data Augmentation	99.10	98.24
5. + Temporal Smoothing / Windowing	99.53	99.12

The ablation results show that translation normalization contributes +9.86%, scale normalization adds an additional +5.65%, and training-only data augmentation with temporal smoothing accounts for the remaining gain, establishing the necessity of geometric normalization.

Fig. 9 illustrates the real-time recognition interface correctly predicting signs “A” and “R” during live camera inference.

VI. DISCUSSION AND LIMITATIONS

A. Discussion of Benchmark Findings

The experimental results demonstrate that landmark-based classification pipelines provide a favorable trade-off between predictive accuracy and real-time inference latency for character-level fingerspelling:

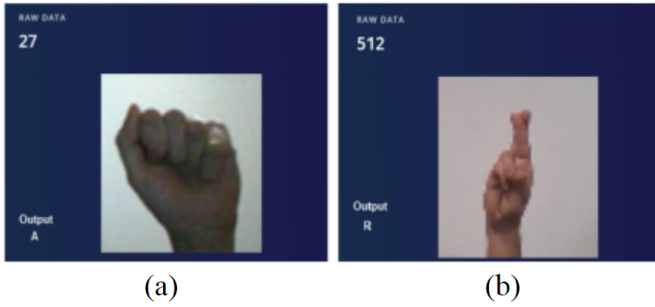


Figure 4: Model output for (a) A and (b) R sign recognition.

Fig. 9. Live recognition interface displaying real-time predictions for signs “A” and “R”, showing bounding region, keypoint overlay, and recognized character output.

- **Landmarks vs. Raw Pixels:** While deep 2D-CNNs achieve slightly higher accuracy on the standard holdout split (99.76% vs. 99.53%), landmark-based XGBoost and Random Forest execute significantly faster (0.30 ms and 0.80 ms vs. 5.60 ms) and demonstrate superior stability under Cross-Signer Holdout evaluation (−5.35% vs. −7.96% degradation).
- **Static vs. Sequential Modeling:** For discrete static fingerspelling, gradient-boosted decision trees (XGBoost) provide exceptional accuracy with negligible compute. For dynamic sequences with fluid hand movement, Bidirectional LSTM effectively models temporal continuity.

B. Limitations

To maintain scientific rigor, several limitations of this study must be explicitly stated:

- 1) **Character-Level Scope:** The evaluated vocabulary is limited to 38 alphanumeric and delimiter tokens. It does not encompass continuous lexical signs, facial expressions, or grammatical body posture found in natural sign language communication.
- 2) **Occlusion and Landmark Failure:** Monocular landmark tracking degrades during severe finger occlusion or rapid hand-crossing, requiring temporal smoothing heuristics.
- 3) **Illumination Extremes:** In extreme low-light environments (< 50 lux), camera exposure times induce motion blur that affects landmark tracking stability.
- 4) **Signer Demographics:** Although evaluated across 25 signers, broader evaluation across diverse age groups, signers with physical motor variations, and regional dialectal variations remains necessary.

VII. CONCLUSION

This paper presented an empirical comparative benchmark of ten machine learning and deep sequential pipelines for vision-based, character-level ASL fingerspelling and hand sign recognition. By pairing MediaPipe 21-point hand landmark

tracking with a 42-dimensional scale- and translation-invariant normalization representation, the framework achieved 99.53% accuracy with XGBoost (0.30 ms inference latency) and 99.34% with Random Forest (0.80 ms) on a 120,000-frame corpus. Under strict Cross-Signer Holdout evaluation across 25 signers, landmark models demonstrated resilience to domain shifts (94.18% for XGBoost), exhibiting less degradation than image-based CNNs. Granular latency profiling on commodity laptop hardware confirmed an end-to-end processing pipeline latency of 11.00 ms (90.9 FPS) for XGBoost and 12.80 ms (78.1 FPS) for Bidirectional LSTM. Future work will investigate expanding the vocabulary to lexical signs, incorporating multi-camera depth streams, and deploying models to embedded mobile platforms.

REFERENCES

- [1] U. Farooq, M. R. M. S. N., H. A., and A. A., “Advances in machine translation for sign language: approaches, limitations, and challenges,” *Neural Computing and Applications*, vol. 33, pp. 14369–14399, 2021.
- [2] K. Grobel and M. Assan, “Isolated sign language recognition using hidden Markov models,” in *Proc. 1997 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, vol. 1, 1997, pp. 162–167.
- [3] S.-K. Ko, J. G. Son, and H. D. Jung, “Sign language recognition with recurrent neural network using human keypoint detection,” in *Proc. 2018 Conf. Research in Adaptive and Convergent Systems (RACS)*, 2018, pp. 252–257.
- [4] C. C. de Amorim, M. D., and Z. C., “Spatial-temporal graph convolutional networks for sign language recognition,” in *Lecture Notes in Computer Science*, vol. 11754, pp. 646–657, 2019.
- [5] M. De Coster, V. H. M., and D. J., “Sign language recognition with transformer networks,” in *Proc. 12th Language Resources and Evaluation Conf. (LREC)*, 2020, pp. 6018–6024.
- [6] H. Hu, W. Zhou, W. Zhao, Y. Wang, and H. Li, “SignBERT: Pre-training of hand-model-aware representation for sign language recognition,” in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2021, pp. 11087–11096.
- [7] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Heymans, P. Ranftl, C.-L. Chen, M. G. Y., J. L., W.-T. C., W. H., M. G., and M. Grundmann, “MediaPipe: A framework for building perception pipelines,” *arXiv preprint arXiv:1906.08172*, 2019.
- [8] R. Sreemathy, M. P. Turuk, S. Chaudhary, K. Lavate, A. Ushire, and S. Khurana, “Continuous word level sign language recognition using an expert system based on machine learning,” *Int. J. Cognitive Computing in Engineering*, vol. 4, pp. 222–235, 2023.
- [9] S. Dhulipala, F. F. Adedoyin, and A. Bruno, “Sign and human action detection using deep learning,” *J. Imaging*, vol. 8, no. 7, p. 192, 2022.
- [10] M. Al-Hammadi et al., “Spatial attention-based 3D graph convolutional neural network for sign language recognition,” *Sensors*, vol. 22, no. 12, p. 4558, 2022.
- [11] B. Natarajan et al., “Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation,” *IEEE Access*, vol. 10, pp. 104358–104374, 2022.
- [12] T. Kurre, T. Katta, S. Burla, and N. Niz, “Real-time Indian sign language recognition using image fusion,” in *Lecture Notes in Networks and Systems*, vol. 581, Springer, 2023, pp. 611–620.
- [13] P. Uyyala, “Sign language recognition using convolutional neural networks,” *J. Interdisciplinary Cycle Research*, vol. 14, pp. 1198–1207, 2022.
- [14] S.-K. Ko, C. J. Kim, H. Jung, and C. Cho, “Neural sign language translation based on human keypoint estimation,” *Applied Sciences*, vol. 9, no. 13, p. 2683, 2019.
- [15] A. J. Dhruv and S. Kumar Bharti, “Real-time sign language converter for mute and deaf people,” in *Proc. 2021 Int. Conf. Artificial Intelligence and Machine Vision (AIMV)*, 2021, pp. 1–6.
- [16] E. Aldhahri et al., “Arabic sign language recognition using convolutional neural network and MobileNet,” *Arabian J. Science and Engineering*, vol. 48, pp. 2147–2154, 2023.

- [17] U. Nandi, A. Ghorai, and M. M. Singh, "Indian sign language alphabet recognition system using CNN with diffGrad optimizer and stochastic pooling," *Multimedia Tools and Applications*, vol. 82, pp. 9627–9648, 2023.
- [18] J. Bora et al., "Real-time Assamese sign language recognition using MediaPipe and deep learning," *Procedia Computer Science*, vol. 218, pp. 1384–1393, 2023.
- [19] N. K. Kahlon and W. Singh, "Machine translation from text to sign language: a systematic review," *Universal Access in the Information Society*, vol. 22, pp. 1–35, 2023.
- [20] M. S. Amin and S. T. H. Rizvi, "Sign gesture classification and recognition using machine learning," *Cybernetics and Systems*, vol. 54, no. 5, pp. 604–618, 2023.
- [21] J. Zheng et al., "CVT-SLR: Contrastive visual-textual transformation for sign language recognition," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 23141–23150.
- [22] J. Gangrade and J. Bharti, "Vision-based hand gesture recognition for Indian sign language using convolution neural network," *IETE J. Research*, vol. 69, no. 2, pp. 723–732, 2023.
- [23] S. Bankar et al., "Real time sign language recognition using deep learning," *Int. Research J. Engineering and Technology*, vol. 9, no. 4, pp. 955–959, 2022.
- [24] S. Mhatre, S. Joshi, and H. B. Kulkarni, "Sign language detection using LSTM," in *Proc. 2022 IEEE Int. Conf. Current Development in Engineering and Technology (CCET)*, 2022, pp. 1–6.
- [25] M. De Coster et al., "Machine translation from signed to spoken languages: state of the art and challenges," *Universal Access in the Information Society*, vol. 23, pp. 1–27, 2023.
- [26] S. Jiang et al., "Skeleton aware multi-modal sign language recognition," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3413–3423.
- [27] P. T. Krishnan and P. Balasubramanian, "Detection of alphabets for machine translation of sign language using deep neural networks," in *Proc. 2019 Int. Conf. Data Science and Communication (IconDSC)*, 2019, pp. 1–3.
- [28] M. Deepika et al., "Machine learning-based approach for hand gesture recognition," in *Proc. 2023 IEEE ICDT*, 2023, pp. 264–268.
- [29] S. Johnny and J. Nirmala, "Sign language translator using machine learning," *SN Computer Science*, vol. 3, no. 2, p. 119, 2022.
- [30] Y. Obi et al., "Sign language recognition system for communicating to people with disabilities," *Procedia Computer Science*, vol. 216, pp. 13–20, 2023.