camAI: Bridging Communication Through AI and Camera-Based Sign Language Recognition

Chris Immanuel Arcasa
Department of Computer Engineering and Mechatronics
Mindanao State University – Iligan Institute of Technology
Iligan City, Philippines
chrisimmanuel.arcasa@g.msuit.edu.ph

***Abstract*—The communication barrier between hearing and deaf communities in the Philippines remains a significant challenge with more than 121,000 individuals relying on Filipino Sign Language (FSL). This paper presents a system for FSL recognition designed to run on everyday computers. The system comprises two complementary modules, a static alphabet recognizer and a dynamic gesture interpreter. The static module uses MediaPipe for hand landmark extraction and a sequential neural network, achieving 99.72% accuracy on 11,700 images. The dynamic module employs statistical feature engineering and a Random Forest classifier, attaining 95–96% accuracy with only 18–20 samples per class. Both modules run at more than 30 FPS on CPU-only systems, demonstrating feasibility for real-world deployment. Future work will expand the vocabulary and improve cross-signer generalization.**

***Index Terms*—Sign language recognition, computer vision, real-time systems, feature extraction, machine learning, accessibility.**

I. Introduction

Sign languages serve as the primary means of communication for deaf and hard-of-hearing individuals worldwide. In the Philippines, Filipino Sign Language (FSL) was officially recognized in 2018 [1], yet a significant communication gap persists due to limited public proficiency [2]. Automated sign language recognition has advanced through computer vision and machine learning but research has largely focused on data-rich languages like American Sign Language (ASL) [3]. This leaves FSL understudied, with scarce annotated datasets and few tailored solutions.

Existing vision-based systems often rely on deep learning models that require large datasets and substantial computational resources [4]. Sensor-based approaches offer high accuracy but depend on specialized hardware, limiting accessibility [5]. For FSL, most prior work addresses only static alphabets or depends on controlled environments [6]. There remains a clear need for a unified, data-efficient, and hardware-accessible FSL recognition system.

This paper presents *camAI*, a dual-module framework for FSL recognition using a standard camera. As shown in the Graphical Abstract (Fig. 1), the proposed workflow addresses the lack of FSL datasets and the high computational demands of deep learning by not performing raw pixel processing. Instead, it utilizes geometric landmark extraction to create a lightweight and CPU-optimized inference engine suitable for resource-constrained issues. The main contributions of this paper are as follows:

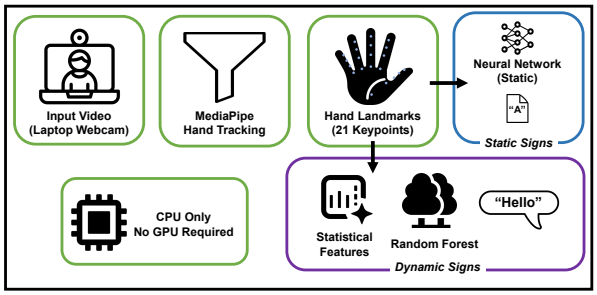


Fig. 1: Graphical abstract of the camAI system architecture.

1) We propose a unified framework for FSL recognition that handles both static alphabets and dynamic gestures, designed to meet the language needs and limited resources.
2) We introduce a novel feature engineering approach for dynamic gestures that extracts 634 statistical features from temporal landmark sequences, enabling high accuracy with minimal training data using a Random Forest classifier.
3) We demonstrate that both modules operate efficiently on consumer-grade CPUs at over 30 FPS, eliminating GPU dependency and enhancing accessibility for resource-limited settings.

The rest of the paper is organized as follows: Section II reviews related work in vision-based and sensor-based sign language recognition. Section III details the proposed methodology, including dataset preparation, feature extraction, and model architectures. Section IV presents experimental results and discussion. Finally, Section V concludes the paper and suggests future research directions.

II. Related Work

A. Vision-Based Approaches

Vision-based sign language recognition has evolved from manually engineered features to deep learning models. Early methods used geometric descriptors and optical flow [7], [8].

Algorithm 1 camAI System Framework

Input: Raw video stream $\mathbf{V}$
Output: Recognized static letters L and dynamic phrases P
Initialize MediaPipe Hands model $\mathcal{M}$
Initialize Static Model f_{θ_s} and Dynamic Model f_{θ_d}
Initialize frame buffer $\mathcal{B} \leftarrow \emptyset$
while stream is active **do**
  Capture frame I_t from $\mathbf{V}$
  $\mathbf{L}_t \leftarrow \mathcal{M}(I_t)$ {Extract 21 landmarks}
  if $\mathbf{L}_t \neq \emptyset$ **then**
    $\mathbf{x}_t \leftarrow \text{normalize}(\mathbf{L}_t)$ {63D vector}
    $l_t \leftarrow f_{\theta_s}(\mathbf{x}_t)$ {Static classification}
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{x}_t\}$
    if $|\mathcal{B}| = 120$ **then** {4 seconds at 30 FPS}
      $\mathbf{X} \leftarrow \text{aggregate}(\mathcal{B})$
      $\mathbf{F} \leftarrow \phi(\mathbf{X})$ {Extract 634 features}
      $p \leftarrow f_{\theta_d}(\mathbf{F})$ {Dynamic classification}
      Output phrase p
      $\mathcal{B} \leftarrow \emptyset$
    Display letter l_t
=0

Convolutional Neural Networks (CNNs) later dominated static gesture recognition [9], [10]. For dynamic gestures, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks became widespread [11], [12]. More recent works employ 3D CNNs and two-stream architectures [13], [14]. MediaPipe has gained popularity for real-time hand tracking due to its efficiency [15]. However, these methods typically require large datasets, which are scarce for FSL.

B. Sensor-Based Approaches

Sensor-based systems often use data gloves, inertial measurement units (IMUs), or depth cameras to capture precise hand and body movements [16], [17]. These approaches achieve high accuracy and are robust to visual variations [18], [19]. However, they rely on specialized hardware, increasing cost and reducing practicality for everyday use [20], [21]. For FSL, limited studies have explored sensor-based solutions, and none offer a complete, affordable system.

III. Methodology

A. System Framework

The camAI framework processes sign language input through two parallel pipelines that share a common feature extraction front-end. Let the input video stream be denoted as a sequence of frames $\mathbf{V} = \{I_1, I_2, ..., I_T\}$, where $I_t \in \mathbb{R}^{H \times W \times 3}$ is the t-th RGB frame. The system operates as described in Algorithm 1.

The system processes each frame through MediaPipe Hands model $\mathcal{M}$, which outputs 21 hand landmarks $\mathbf{L}_t = \{(x_i, y_i, z_i)\}_{i=1}^{21}$. These coordinates are normalized to the range $[0, 1]$ relative to the image dimensions:

TABLE I: Dataset Specifications for FSL Recognition

Dataset	Type	Classes	Samples	Source
Static FSL Alphabet	Images	26 (A-Z)	11,700	Multiple contributors
Dynamic Greetings	Videos	6	120	Tupal FSL-105
Dynamic Conversational	Videos	5	100	Tupal FSL-105

Algorithm 2 Unified Hand Landmark Preprocessing

procedure PREPROCESS(I)
  $I_{\text{RGB}} \leftarrow \text{BGR2RGB}(I)$
  $\mathbf{L} \leftarrow \mathcal{M}(I_{\text{RGB}})$
  if $\mathbf{L} \neq \emptyset$ **then**
    $\mathbf{L}_{\text{norm}} \leftarrow \text{Normalize}(\mathbf{L})$
    return $\mathbf{L}_{\text{norm}}$
  else
    return $\emptyset$
=0

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \quad y'_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}, \quad z'_i = \frac{z_i}{d_{\text{ref}}} \tag{1}$$

where d_{ref} is a reference depth constant. This yields a normalized feature vector $\mathbf{x}_t \in \mathbb{R}^{63}$ per frame.

Figure 2 illustrates the complete architecture. The static pipeline processes individual frames, while the dynamic pipeline accumulates 120 frames (4 seconds) for phrase recognition. The landmark extraction stage provides structural consistency to scale, rotation, and lighting variations.

B. Dataset Collection and Preprocessing

Table I summarizes the datasets used for training and evaluation. The static alphabet dataset contains 450 images per letter with variations in hand size, skin tone, lighting, and background. The dynamic dataset comprises two subsets from Tupal FSL-105: Set A (greetings) and Set B (conversational phrases), each with 18-20 samples per class recorded by four native FSL signers.

Figure 3 shows representative samples from both datasets.

1) Unified Data Preprocessing Pipeline: Let $I \in \mathbb{R}^{H \times W \times 3}$ denote an input image. The preprocessing algorithm is formalized in Algorithm 2. The MediaPipe model $\mathcal{M}$ extracts landmarks $\mathbf{L}$ when a hand is detected. For dynamic sequences, we accumulate frames into temporal sequences $\mathbf{X} \in \mathbb{R}^{T \times 63}$, where $T = 120$ frames.

C. Data Partitioning and Preparation

Stratified sampling ensured proportional class representation across splits, as shown in Table II. The static dataset used a simple 80-20 split, while dynamic datasets employed 5-fold stratified cross-validation for robust evaluation given limited samples.

D. Model Architectures and Training

1) Static Alphabet Recognition: Figure 4 illustrates the sequential neural network architecture for static recognition.

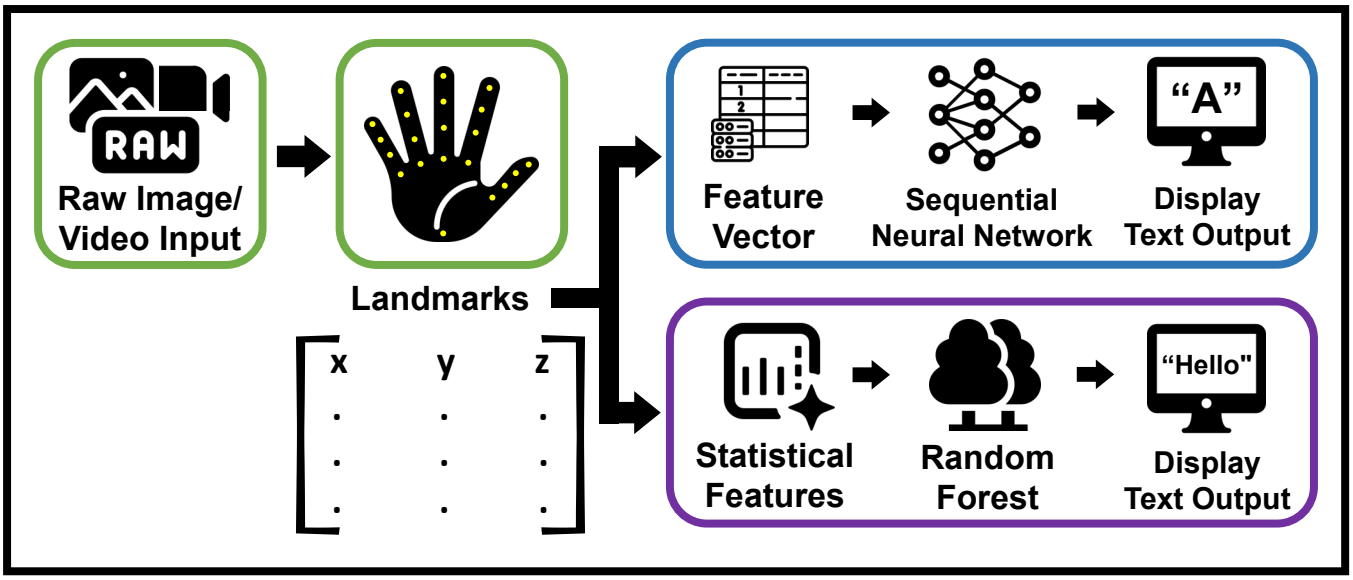
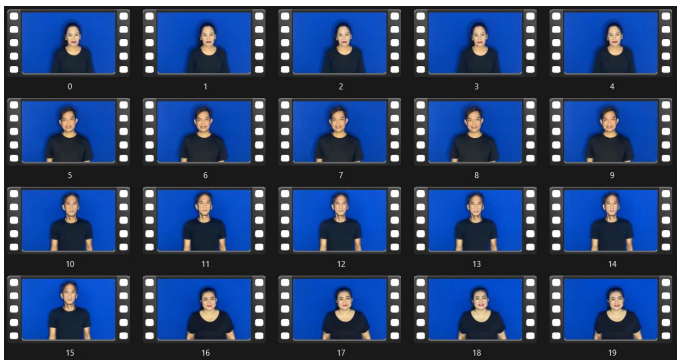


Fig. 2: Unified system architecture for camAI. The raw input is processed through MediaPipe to extract 21 hand landmarks. For static recognition, a single frame with landmarks is fed to the sequential neural network. For dynamic recognition, a 4-second sequence of landmarks undergoes statistical feature extraction before classification by the Random Forest ensemble.

(a) Static FSL alphabet samples from the dataset



(b) Dynamic FSL gesture samples from the dataset

Fig. 3: Sample images from (a) static alphabet and (b) dynamic gesture datasets.

The model comprises three fully connected layers with ReLU activation, batch normalization, and dropout regularization:

$$\begin{aligned} \mathbf{h}_1 &= \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{h}_2 &= \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \\ \mathbf{h}_3 &= \text{ReLU}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3) \\ \mathbf{y} &= \text{Softmax}(\mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4) \end{aligned} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{63}$ is the input landmark vector, and $\mathbf{y} \in \mathbb{R}^{26}$ represents class probabilities.

Table III details the training configuration. Algorithm 3 outlines the optimization procedure.

2) Dynamic Gesture Recognition: For dynamic recognition, we extract 634 statistical features $\phi(\mathbf{X})$ from each 4-second sequence $\mathbf{X}$, including temporal means, variances, correlations, and inter-landmark distances. These features capture both spatial configurations and motion dynamics.

Table IV presents the ensemble model configurations. The Random Forest classifier with 100 decision trees proved optimal through grid search with 5-fold cross-validation.

Algorithm 4 describes the grid search procedure used to identify optimal hyperparameters. The Random Forest's ensemble nature and non-parametric characteristics made it particularly suitable for the high-dimensional feature space with limited training samples.

IV. Results and Discussion

A. Model Performance and Comparative Analysis

1) Static FSL Alphabet Recognition: The sequential neural network demonstrated exceptional performance in classifying

TABLE II: Dataset Partitioning for Model Training and Evaluation

Dataset	Total Samples	Training (80%)	Testing (20%)	Classes
Static Alphabet	10,610	8,488	2,122	26
Dynamic Greetings	120	96	24	6
Dynamic Conversational	100	80	20	5

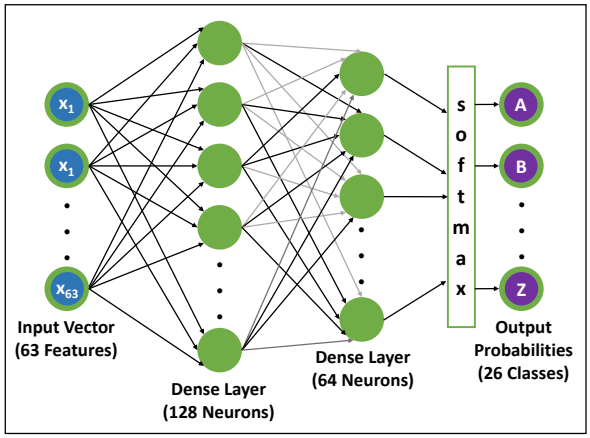


Fig. 4: Neural Network Architecture for Static FSL Recognition. The 63-dimensional input feeds into a 128-neuron dense layer with ReLU and dropout, followed by a 64-neuron dense layer. The final Softmax layer predicts 26 output classes.

TABLE III: Training Configuration for Static Alphabet Model

Parameter	Value
Optimizer	Adam (lr=0.001)
Loss Function	Sparse Categorical Crossentropy
Batch Size	32
Epochs	50 (early stopping patience=5)
Hidden Layers	3 (256, 128, 64 units)
Regularization	Dropout (0.3), Batch Normalization
Validation Split	20% of training data

static FSL alphabet gestures. As shown in Figure 5, the model achieved rapid convergence during training, with validation accuracy stabilizing after approximately 25 epochs.

Key performance metrics are quantified in Table V.

The minimal gap (0.13%) between training and validation performance indicates excellent generalization without overfitting, validating our architectural choices and regularization strategies. The model achieves real-time performance with inference times under 5 ms per frame, enabling seamless user interaction.

2) Dynamic FSL Gesture Recognition: The Random Forest classifier significantly outperformed alternative approaches across both gesture sets, as summarized in Table VI.

Statistical analysis confirmed Random Forest's superiority (RF vs LR: $p < 0.01$; RF vs SVM: $p < 0.001$), demonstrating that the performance advantage was not attributable to random variation. The feature engineering approach proved particularly effective in the low-data regime, achieving 95-96% accuracy with only 18-20 samples per class.

Algorithm 3 Static Model Training Procedure

Initialize parameters θ_s randomly
for epoch $= 1$ to N **do**
  for batch $(\mathbf{X}_b, \mathbf{y}_b)$ in training data **do**
    $\hat{\mathbf{y}}_b \leftarrow f_{\theta_s}(\mathbf{X}_b)$
    $\mathcal{L} \leftarrow \text{CrossEntropy}(\hat{\mathbf{y}}_b, \mathbf{y}_b)$
    $\nabla_{\theta_s} \leftarrow \text{Backpropagate}(\mathcal{L})$
    $\theta_s \leftarrow \text{AdamUpdate}(\theta_s, \nabla_{\theta_s})$
  Evaluate on validation set
  if validation loss not improving for 5 epochs **then**
    break
=0

TABLE IV: Ensemble Model Configurations for Dynamic Gesture Recognition

Model	Hyperparameters	Search Range
Random Forest	n_estimators: 100, criterion: gini	[50, 100, 200]
SVM (RBF)	C: 1.0, γ: scale	C: [0.1, 1, 10]
Logistic Regression	solver: lbfgs, penalty: l2	C: [0.1, 1, 10]

B. Real-Time System Performance and Validation

Both systems demonstrated strong performance, operating at 34 frames per second on standard CPU hardware. Figure 6 showcases the integrated system performing both recognition tasks, with subfigure (a) showing static alphabet recognition and subfigure (b) displaying dynamic gesture interpretation.

Real-world testing confirmed the system's practical utility with consistent performance across varied lighting conditions and backgrounds. The modular architecture allowed seamless switching between alphabet recognition for finger-spelling and dynamic gesture recognition for common phrases. System latency measurements showed end-to-end processing times of 29.4 ms per frame, well within the 33 ms threshold for real-time 30 FPS operation.

C. Discussion of Key Findings

1) Methodological Insights: The exceptional performance of both approaches can be attributed to strategic problem formulation. For static recognition, MediaPipe landmark extraction transformed the problem from image recognition to geometric pattern recognition, eliminating common computer vision challenges. For dynamic gestures, the feature engineering approach proved that temporal dynamics could be effectively captured through statistical representations rather than complex sequential models. This challenges the current assumption that deep learning necessarily outperforms traditional methods in data-constrained scenarios.

Algorithm 4 Dynamic Model Training with Grid Search

Define parameter grid Θ (see Table IV)
Initialize best score $\leftarrow 0$
for each combination $\theta \in \Theta$ **do**
  Initialize scores list $\mathcal{S} \leftarrow \emptyset$
  for fold $= 1$ to 5 **do**
    Split data into train/validation folds
    Train model f_θ on training fold
    Evaluate on validation fold: $s \leftarrow$ accuracy
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
  $\bar{s} \leftarrow \text{mean}(\mathcal{S})$
  if $\bar{s} >$ best score **then**
    $\theta_{\text{best}} \leftarrow \theta$
    best score $\leftarrow \bar{s}$
Train final model $f_{\theta_{\text{best}}}$ on full training data =0

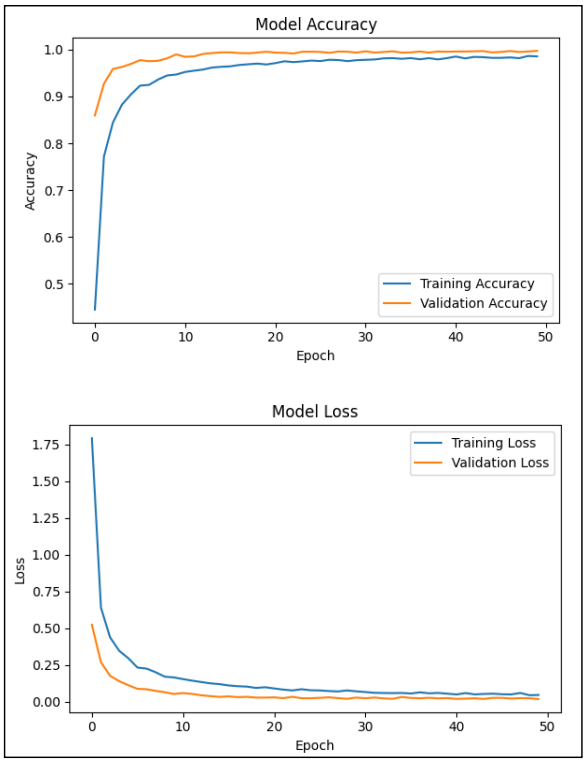


Fig. 5: Training convergence for static alphabet model showing accuracy and loss trends over 50 epochs.

2) Practical Implications and Accessibility: The systems' CPU-only operation enables deployment on low-cost hardware, significantly increasing accessibility for educational institutions and individual users in resource-constrained environments. The real-time responsiveness ensures natural, fluid interaction without noticeable delays, while consistent performance across varying conditions supports reliable real-world deployment. The technology demonstrates particular promise for educational applications, where it could facilitate communication between hearing educators and deaf students.

TABLE V: Performance Metrics for Static Alphabet Recognition

Metric	Value
Training Accuracy	0.9985
Validation Accuracy	0.9972
Training Loss	0.0087
Validation Loss	0.0093
Training Time (50 epochs)	102 seconds
Inference Time per Frame	4.2 ms

TABLE VI: Performance Comparison for Dynamic Gesture Recognition

Model	Greetings	Conversational	Cross-Val Score
Random Forest	0.960	0.950	0.904 ± 0.016
LSTM	0.520	0.650	0.605 ± 0.020
SVM (RBF)	0.400	0.300	0.439 ± 0.014

V. Conclusion

This research presents camAI, a real-time vision-based system for Filipino Sign Language recognition that establishes a framework for handling both static alphabets and dynamic conversational gestures. The static recognition module achieves 99.72% accuracy using a sequential neural network, while the dynamic recognition module achieves 95-96% accuracy with minimal training data through innovative feature engineering and ensemble learning. Both systems operate efficiently on consumer-grade CPUs at 34 FPS, eliminating GPU dependency and enhancing accessibility for resource-limited settings.

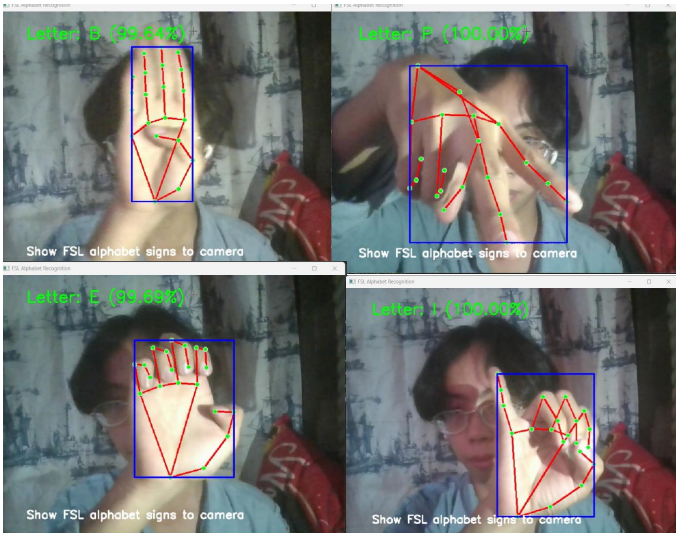
The practical significance of this work lies in its immediate applicability for bridging communication barriers in the Philippines' deaf community. Future development will focus on expanding vocabulary coverage and improving cross-signer robustness. This research contributes both a practical tool for FSL communication and a methodological framework for other under-resourced sign languages worldwide.

Acknowledgment

The author acknowledges the use of DeepSeek AI for assistance in generating boilerplate code and debugging the Python implementation of MediaPipe and TensorFlow. The AI tool was utilized to optimize the sequential neural network architecture and streamline the data preprocessing pipeline. All code and concepts were verified and validated by the author.

References

[1] P. Congress, "Republic act no. 11106: Filipino sign language act," 2018.
[2] P. S. Authority, "Disability data," 2020.
[3] R. R. *et al.*, "Sign language recognition: A deep survey," *Expert Systems with Applications*, 2021.
[4] M. W., "Deep learning for gesture recognition: A review," *Pattern Recognition*, 2024.
[5] J. J.. *et al.*, "A systematic mapping of computer vision-based sign language recognition," *IEEE Transactions on Pattern Analysis*, 2022.
[6] L. T. *et al.*, "Filipino sign language translation through transfer learning," *International Conference on Natural Language Processing and Information Retrieval*, 2025.



(a) Static alphabet inference.

(b) Dynamic gesture inference..

Fig. 6: Real-time inference system in operation: (a) static alphabet recognition and (b) dynamic gesture recognition.

[7] M. Hu, "Visual pattern recognition by moment invariants," in *IRE Transactions on Information Theory*, 1962.

[8] F. Chen *et al.*, "Hand gesture recognition using a real-time tracking method and hidden markov models," in *Image and Vision Computing*, 2003.

[9] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *NeurIPS*, 2012.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[11] A. Graves *et al.*, "Speech recognition with deep recurrent neural networks," *ICASSP*, 2013.

[12] P. Shah *et al.*, "Survey on vision based hand gesture recognition," *International Journal of Computer Sciences and Engineering*, 2019.

[13] D. Tran *et al.*, "Learning spatiotemporal features with 3d convolutional networks," *ICCV*, 2015.

[14] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *CVPR*, 2017.

[15] G. Research, "Mediapipe: A framework for building perception pipelines," 2020.

[16] J. Kim *et al.*, "Lvq-based hand gesture recognition using a data glove," *Smart Innovation, Systems and Technologies*, 2013.

[17] K. Kudrinko *et al.*, "Wearable sensor-based sign language recognition: A comprehensive review," *IEEE Reviews in Biomedical Engineering*, 2020.

[18] J. Wang *et al.*, "Dynamic hand gesture recognition using inertial sensors," *IEEE Sensors Journal*, 2016.

[19] C. Lu *et al.*, "Sign language recognition with multimodal sensors and deep learning methods," *Electronics 2023*, 2023.

[20] T. Yang *et al.*, "A survey: The sensor-based method for sign language recognition," 2023.

[21] R. Gupta *et al.*, "Deep learning based sign language recognition robust to sensor displacement," 2023.