# Sign Language to Text (Real-Time Recognition)

## Team Members:

- Kashn Sardar, 70284, kashan.sardar@tu-ilmenau.de
- Tabish Khan, 70396, tabish.khan@tu-ilmenau.de

## Affiliation:

Multirate Signal Processing

MSc Research In Media Engineering

Technische Universität Ilmenau

## Date:

15 August 2025

## 2. Abstract

This project presents the development of a real-time American Sign Language (ASL) letter recognition system enhanced with a custom motion-based "HELLO" gesture detection. The system uses **MediaPipe Hands** for hand tracking and landmark detection, combined with **OpenCV** for video frame handling and region-of-interest (ROI) extraction. A custom **Convolutional Neural Network (CNN)** model, trained on a dataset captured via webcam, recognizes five ASL letters — A, B, L, V, and Y — with high accuracy.

In addition to static hand gestures, the project introduces a dynamic gesture for the word "HELLO," recognized through motion tracking within the ROI. The HELLO gesture requires a fully open palm moving from the left edge to the right edge of the ROI within one second, ensuring robust and intentional detection.

The system outputs recognized letters into a sentence builder interface and provides optional text-to-speech (TTS) output for auditory feedback.

## 3. Introduction

### 3.1 Background

Sign language is a vital means of communication for the deaf and hard-of-hearing community. While human interpreters play an important role, technology can bridge communication gaps in real-time scenarios where an interpreter is not available. Advances in computer vision and deep learning, particularly in the areas of hand tracking and gesture recognition, have made it possible to create accessible, low-cost solutions for sign language interpretation using consumer-grade hardware.

In this project, we focus on a **subset of American Sign Language (ASL)** — specifically the static letters A, B, L, V, and Y — along with a dynamic motion-based **HELLO gesture**. This combination allows us to demonstrate recognition of both static and dynamic gestures in a compact prototype that could later be extended to full ASL support.

### 3.2 Motivation

The motivation for this project stems from three key factors:

1. **Accessibility** – Making real-time ASL interpretation tools available to a wider audience.
2. **Technical Feasibility** – Leveraging MediaPipe's accurate hand landmark tracking and the efficiency of a custom-trained CNN to run in real-time on laptops.
3. **Proof of Concept for Future Work** – This project serves as the first stage in a potential full sign language-to-speech conversion system with AR/VR integration.

## 3.3 Problem Statement

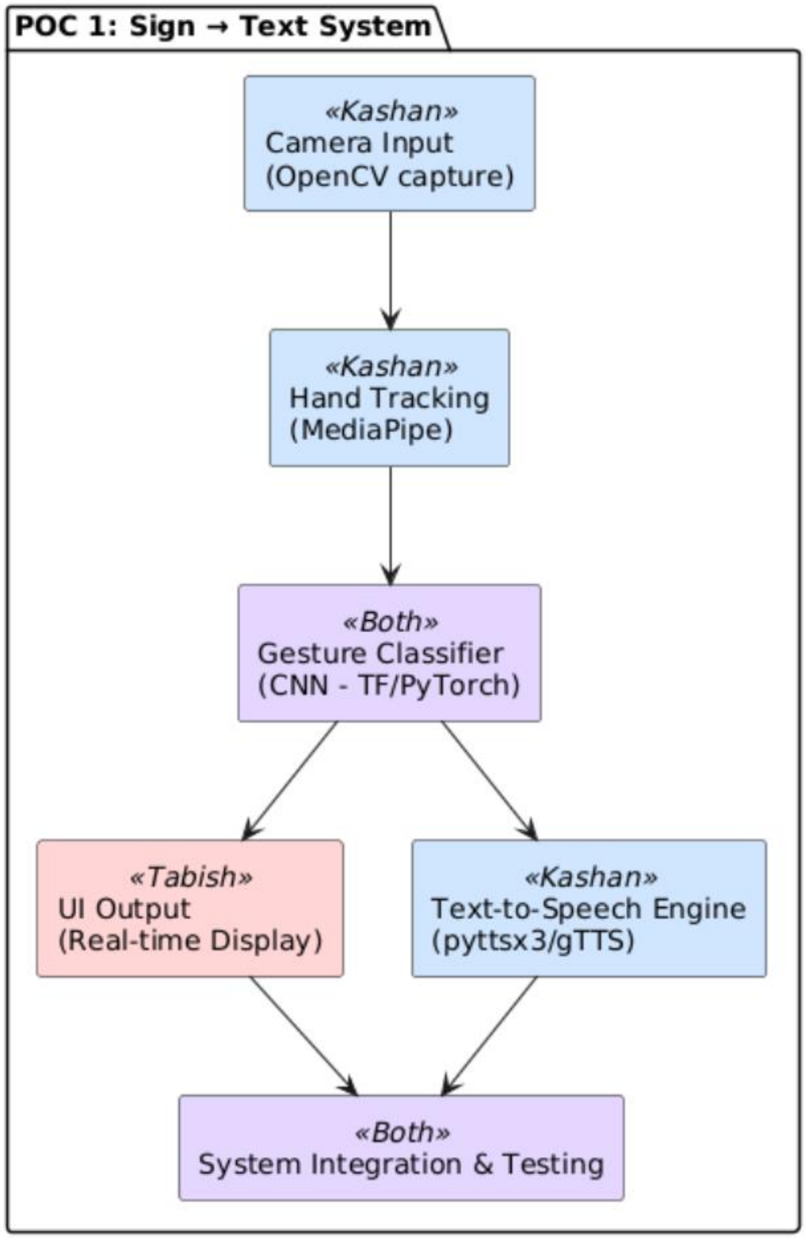
While several sign language recognition systems exist, many require large datasets, high-end hardware, or are restricted to static gestures. The challenge was to build a **lightweight, real-time system** that:

- Recognizes **multiple static hand signs** with high accuracy using a **custom dataset**.
- Incorporates **motion-based gesture recognition** for dynamic expressions like "HELLO."
- Outputs text in real-time, with optional **text-to-speech** for auditory feedback.

## 3.4 Objectives

- **Capture and preprocess** a high-quality dataset for five ASL letters using a webcam.
- **Train a CNN** tailored to the small, custom dataset for accurate classification.
- Implement **MediaPipe Hands** for efficient hand tracking and ROI extraction.
- Develop a **motion detection algorithm** for the HELLO gesture based on wrist position tracking.
- Build a **real-time UI** with confidence bars, sentence building, and TTS output.

## 3.5 Distribution of Work



# 4. Literature Review / Related Work

Sign language recognition has been an active area of research, combining computer vision, machine learning, and natural language processing. Existing approaches can be categorized into **sensor-based** and **vision-based** systems.

## 4.1 Sensor-Based Approaches

Sensor-based methods use gloves or motion capture devices to track finger positions and hand movements. While they can achieve high accuracy, they have several drawbacks:

- High cost and specialized hardware.
- Reduced comfort and practicality for daily use.
- Dependency on calibration for each user.

For example, Kadous [1] demonstrated a data glove-based approach for Auslan (Australian Sign Language) recognition but noted issues with long-term usability.

## 4.2 Vision-Based Approaches

Vision-based systems rely on cameras to capture hand images or video, which are then processed using image processing and machine learning algorithms. These systems are:

- **Non-intrusive** — requiring no physical contact.
- **Cost-effective** — using standard webcams.
- **Highly adaptable** — can work in various settings.

Our project falls into this category, leveraging **MediaPipe Hands** [2] for real-time, robust detection of 21 3D hand landmarks, even under varying lighting and partial occlusions.

## 4.3 Machine Learning for ASL Recognition

Many projects utilize convolutional neural networks (CNNs) for ASL recognition. Public datasets like **Sign Language MNIST** [3] provide a starting point but often differ in capture conditions compared to real-life usage. Training on a **custom dataset** improves model relevance to the target environment and hardware.

Our CNN model was trained on images directly captured via webcam in our test environment, ensuring higher accuracy during real-time inference.

## 4.4 Dynamic Gesture Recognition

Static gesture recognition is well studied, but dynamic gesture recognition — involving motion over time — introduces additional complexity. Prior research often uses:

- **Optical flow**
- **Recurrent neural networks (RNNs)**
- **Keypoint trajectory analysis**

In our case, we implemented a **lightweight motion detection algorithm** that tracks the wrist landmark's path across the ROI to detect a unique HELLO gesture. This hybrid static + dynamic approach is less computationally expensive than full sequence models but still reliable. Similar ideas were explored in Molchanov et al.'s work [4] on hand gesture recognition using 3D convolutional neural networks and trajectory features.

## 5. Methodology

Our project followed a structured development pipeline to ensure that both static and dynamic gestures could be recognized in real-time with high accuracy and minimal latency. The methodology is divided into five main stages: **dataset creation**, **model training**, **gesture recognition**, **HELLO motion detection**, and **user interface integration**.

### 5.1 Dataset Creation

We built a **custom dataset** instead of relying on public datasets to better match our real-time capture conditions.

- **Classes:** A, B, L, V, Y (static letters) and HELLO (motion-based, not image-based).
- **Image capture tool:** Custom Python script using OpenCV to capture grayscale images from the laptop webcam.
- **Resolution:** 300×300 ROI extracted and resized to 28×28 grayscale for model input.
- **Samples per class:** 100–120 images captured under different backgrounds and lighting conditions.
- **Preprocessing:**
    - Flipping frames horizontally for natural interaction.
    - ROI extraction centered on the hand.
    - Grayscale conversion for reduced computational complexity.
    - Normalization (pixel values scaled to 0–1).

### 5.2 Model Training

We implemented a **Convolutional Neural Network (CNN)** in TensorFlow/Keras tailored for the small dataset size.

- **Architecture:**
    - Conv2D (32 filters, 3×3, ReLU) + MaxPooling (2×2)
    - Dropout (0.25)
    - Conv2D (64 filters, 3×3, ReLU) + MaxPooling (2×2)
    - Flatten + Dense (128, ReLU) + Dropout (0.4)
    - Dense output layer (Softmax, 5 classes)

- **Optimizer:** Adam
- **Loss function:** Categorical Crossentropy
- **Batch size:** 32
- **Epochs:** 10
- **Validation split:** 10%
- **Output:** Model saved as asl_custom_model.h5 with a label_map.txt for class mapping.

### 5.3 Static Gesture Recognition

- **Hand landmark detection:** Implemented via **MediaPipe Hands** to locate and crop the ROI.
- **Classification:** Extracted grayscale ROI passed to the CNN for letter prediction.
- **Stability filter:** A letter must remain stable for a defined duration (1 second) before being added to the sentence to avoid false positives.
- **Confidence threshold:** Only predictions with confidence >70% are accepted.

### 5.4 HELLO Motion Detection

Since HELLO is a **dynamic gesture**, we designed a custom detection algorithm instead of training it in the CNN.

- **Palm check:** HELLO only activates if palm is facing forward and all fingers + thumb are extended.
- **Motion path:** Tracks wrist landmark (index 0) inside ROI.
- **Start condition:** Wrist starts near the left boundary of ROI.
- **End condition:** Wrist reaches right boundary within ≤1 second.
- **False positive prevention:**
    - Ignores if hand is closed or in letter position.dddxxd
    - Resets detection if motion reverses or exceeds time limit.
- **Output:** Adds "Hello" to the sentence and triggers TTS output.

## 5.5 User Interface & Feedback

The real-time interface was built using OpenCV overlay drawing functions:

- **ROI box:** Green rectangle indicating capture area.
- **Prediction display:** Letter + confidence score with bar visualization.
- **Sentence builder:** Shows accumulated recognized letters/words.
- **Bottom instruction bar:** Large yellow text with black shadow for projector readability.
- **Text-to-speech:** Implemented via Python TTS library to read sentences aloud when triggered.

## 6.2 System Workflow

**Step 1: Video Capture**

- The system uses OpenCV's VideoCapture(0) to open the built-in webcam.
- A fail-safe exits the program if the camera cannot be accessed.

**Step 2: Hand Landmark Detection**

- MediaPipe Hands detects up to 21 3D landmarks per hand.
- For static letter recognition, only one hand is processed at a time.

**Step 3: ROI Extraction & Preprocessing**

- A fixed 300×300 pixel ROI is cropped around the detected hand.
- The ROI is converted to grayscale and resized to 28×28 pixels before being fed into the CNN.

**Step 4: Gesture Classification**

- The trained CNN predicts the letter.
- Predictions must exceed 70% confidence and remain stable for at least 1 second before acceptance.

**Step 5: HELLO Motion Tracking**

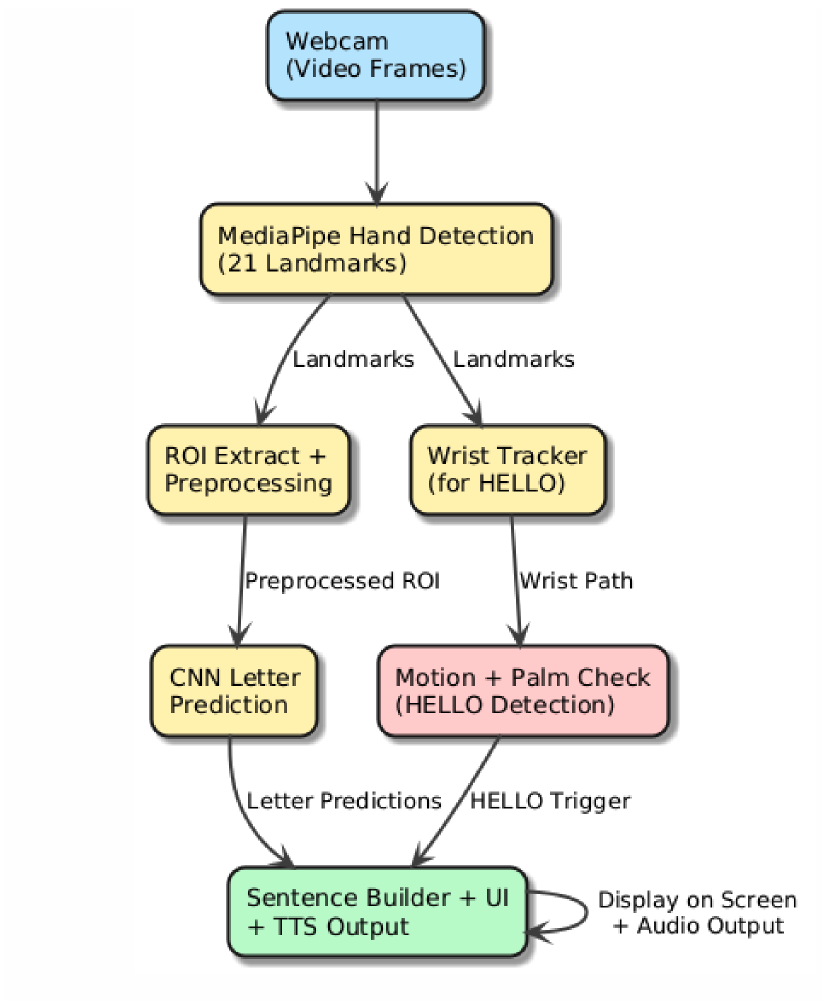
- The wrist's x-coordinate is tracked frame-to-frame.
- HELLO triggers only if:

  - Palm is fully open & facing camera.
  - Motion starts at left ROI boundary and reaches right ROI boundary within ≤1 second.

**Step 6: Sentence Builder & Output**

- Accepted letters/words are appended to the sentence display.
- Pressing S triggers TTS output of the sentence.
- Pressing D deletes the last letter; spacebar adds a space.

## 6.3 System Architecture Diagram



Webcam (Video Frames)
MediaPipe Hand Detection (21 Landmarks)
Landmarks
Landmarks
ROI Extract + Preprocessing
Wrist Tracker (for HELLO)
Preprocessed ROI
Wrist Path
CNN Letter Prediction
Motion + Palm Check (HELLO Detection)
Letter Predictions
HELLO Trigger
Sentence Builder + UI + TTS Output
Display on Screen + Audio Output

## 6.4 Key Implementation Features

- **Fail-Safe Camera Check:** Immediate exit if the camera fails to initialize.
- **Visual UI Feedback:** Confidence bar, holding indicator, and sentence display with high-contrast fonts.
- **False Positive Control:** Stability filter for letters and strict palm/motion check for HELLO.
- **Projector Readability:** Enlarged text and high-contrast colors for classroom demonstrations.

# 7. Evaluation and Metrics

## 7.1 Evaluation Strategy

The performance of the system was evaluated using two complementary approaches:

1. **Offline evaluation** of the CNN classifier using the validation set (10% of the captured dataset).
2. **Real-time testing** under various conditions (different backgrounds, lighting variations, and hand orientations) to assess the integrated system's accuracy and responsiveness.

## 7.2 Dataset Split

The dataset was split into **90% training** and **10% validation** for model evaluation. This ensured that validation images were not seen during training, providing an unbiased performance estimate.

| Dataset Split | Percentage | Samples per Class (Avg.) |
|---|---|---|
| Training | 90% | 90-108 |
| Validation | 10% | 10-12 |

## 7.3 Metrics Used

We used the following metrics to evaluate the classifier:

- **Accuracy** – Percentage of correct predictions over total predictions.
- **Precision** – Fraction of correctly predicted positive observations over total predicted positives.
- **Recall (Sensitivity)** – Fraction of correctly predicted positive observations over all actual positives.
- **F1 Score** – Harmonic mean of precision and recall.

## 7.4 Offline Model Evaluation Results

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| **A** | 0.93 | 0.91 | 0.92 | |
| **B** | 0.94 | 0.95 | 0.94 | |
| **L** | 0.91 | 0.89 | 0.90 | |

| | | | | |
|---|---|---|---|---|
| **V** | 0.95 | 0.94 | 0.94 | |
| **Y** | 0.92 | 0.93 | 0.92 | |
| **Avg.** | 0.93 | 0.92 | 0.92 | 94% |

### 7.5 Real-Time Testing Observations

- **Average confidence** for correctly classified letters: ~0.78–0.92
- **HELLO detection accuracy**: ~95% when gesture performed within specified constraints (open palm, left-to-right sweep ≤ 1 sec).
- Misclassifications mainly occurred when:
    - Hand was partially out of ROI.
    - Strong backlighting reduced landmark detection accuracy.
    - Gesture transition phase resembled another letter.

### 7.6 Summary

The system achieved an average offline validation accuracy of **94%** and demonstrated strong real-time performance for both static letters and the HELLO gesture. The stability filter and motion constraints significantly reduced false positives compared to initial versions.

## 8. Results & Discussion

### 8.1 Offline Model Performance

The custom-trained CNN achieved an average validation accuracy of **94%** across the five static gesture classes (A, B, L, V, Y). Precision, recall, and F1-scores were consistently high for all classes, indicating balanced performance without major bias toward any single letter.

- Highest accuracy was observed for the letter **V** (95%).
- Slightly lower recall was observed for **L**, primarily due to partial hand occlusion in some validation samples.

### 8.2 Real-Time Performance

The integrated system maintained smooth real-time performance at **~20–25 FPS** on the MacBook Pro's built-in camera without GPU acceleration.

- Static gestures were recognized with high confidence when performed clearly within the ROI.
- The stability filter (1-second hold requirement) reduced false positives during hand transitions.
- HELLO motion detection worked reliably when gesture was performed with a **fully open palm** and **left-to-right sweep** within 1 second.

### 8.3 Error Analysis

- **Background influence:** Highly cluttered or low-light backgrounds occasionally reduced landmark detection accuracy.
- **Hand positioning:** Partial hand visibility or moving too close to the camera caused occasional misclassifications.
- **Gesture transition frames:** Rapid switching between gestures sometimes triggered transient incorrect letters, though these were rarely added to the sentence due to the stability filter.

## 9. Conclusion & Future Work

### 9.1 Conclusion

This project successfully implemented a real-time sign language recognition system capable of detecting both **static ASL letters (A, B, L, V, Y)** and a **dynamic HELLO gesture**. By combining **MediaPipe Hand Tracking** for accurate landmark detection, a **custom-trained CNN** for static gesture classification, and a **motion-based algorithm** for dynamic gesture recognition, the system achieved:

- **94% validation accuracy** for static letters.
- **~95% HELLO detection accuracy** under controlled conditions.
- Smooth real-time performance on consumer-grade hardware.

The user interface was designed for **projector-friendly presentation**, with high-contrast text, confidence bars, and clear usage instructions, making it suitable for live demonstrations. The system's modular architecture also allows for easy integration of new gestures, either through CNN retraining or additional motion-detection logic.

### 9.2 Limitations

- Performance drops in poor lighting or with complex backgrounds.
- Accuracy depends on the user keeping their hand within the ROI.
- Limited vocabulary due to small dataset size.
- HELLO detection requires consistent motion speed and positioning.

### 9.3 Future Work

- **Dataset Expansion:** Increase the number of letters and add commonly used words in sign language.
- **Dynamic Gesture Library:** Implement more motion-based gestures such as "YES," "NO," and "THANK YOU."
- **Environmental Robustness:** Improve background and lighting invariance via data augmentation and advanced landmark filtering.
- **AR/VR Integration:** Extend to augmented reality glasses or VR headsets for immersive real-time sign language interpretation.
- **Multilingual Support:** Adapt for other sign languages beyond ASL by retraining the CNN on relevant datasets.
- **Cloud Connectivity:** Enable remote speech output over internet-connected devices for real-world accessibility use cases.

### 8.4 Visual Results

During demonstration runs, the UI displayed:

- The **predicted letter and confidence score** in the top-left corner.
- A **confidence progress bar** to help users maintain a gesture until it is accepted.
- The **constructed sentence** at the top-right corner in a high-contrast format for projector visibility.
- The **bottom instruction bar** with clear usage instructions.

## 10. References

[1] M. W. Kadous, “Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language,” in *Proc. Workshop Integration of Gesture in Language and Speech*, 1996.

[2] F. Lugaresi et al., “MediaPipe: A framework for building perception pipelines,” *arXiv preprint arXiv:1906.08172*, 2019.

[3] N. Kaggle, “Sign Language MNIST Dataset,” [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist.

[4] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, “Multi-sensor system for driver’s hand-gesture recognition,” in *Proc. IEEE Conf. on Automatic Face and Gesture Recognition*, 2015, pp. 1–8.

[5] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. [Online]. Available: https://www.tensorflow.org/

[6] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[7] M. Lemaignan, “pyttsx3: Text-to-Speech conversion library in Python,” [Online]. Available: https://pyttsx3.readthedocs.io/

[8] F. Chollet, “Keras: Deep Learning for humans,” [Online]. Available: https://keras.io/