SignLLM: Bridging Communication Gaps through Real-Time Sign Language Detection Using AI

Ishan R. Mukundhan (220463) Anukalp Rai (22082) Bhavya Sharma (220297)
Anwesha Dwivedi (220190) Norah Srivastava (220736)

Department of Design
Indian Institute of Technology Kanpur, India

November 9, 2025

Abstract

Sign language is one of the oldest and most natural forms of language for communication, but since most people do not know sign language and interpreters are very difficult to come by, we have come up with a real time method using neural networks for fingerspelling based on American sign language. In our method, the hand is first passed through a filter and after the filter is applied the hand is passed through a classifier which predicts the class of the hand gestures. Our method provides 95.7% accuracy for the 26 letters of the alphabet.

Keywords: Sign language recognition, computer vision, Convolutional Neural Network (CNN), real-time detection, fingerspelling, OpenCV, Keras, TensorFlow.

1 Introduction

1.1 Background and Problem Definition

Language forms the cornerstone of human interaction. For individuals with hearing or speech impairments (D&M), sign language provides a complete linguistic system enabling expression and connection. Gestures are non-verbally exchanged messages understood with vision. Sign language is a visual language that uses gestures instead of sound, combining hand-shapes, orientation, movement, facial expressions, and lip-patterns. Contrary to popular belief, sign language is not international and varies by region.

Sign language has three major components: **Fingerspelling** (spelling words letter by letter), **Word level sign vocabulary** (used for the majority of communication), and **Non-manual features** (facial expressions, body position).

Minimizing the verbal exchange gap between D&M and non-D&M people is crucial. A hand gesture recognition system offers an opportunity for deaf people to communicate with vocal humans without an interpreter. This project focuses on a system for the automated conversion of ASL into textual content.

1.2 Motivation and Research Significance

A language barrier exists between normal people and D&M people because sign language structure is different from normal text. If a common interface exists to convert sign language to text, gestures can be easily understood by non-D&M people. Research has been made for a vision-based interface system where D&M people can enjoy communication without knowing each other's language. The aim is to develop a user-friendly Human Computer Interface (HCI) where the computer understands human sign language.

2 Literature Review

Prior research in hand gesture recognition follows basic steps: data acquisition, data pre-processing, feature extraction, and gesture classification.

Data can be acquired via **sensory devices** (like data gloves) which are precise but expensive and not user-friendly, or a **vision-based approach** using a webcam. Vision-based methods are natural and low-cost, but face challenges with hand variability, skin-color, viewpoints, and scales.

For pre-processing, approaches include combining color thresholding with background subtraction or using filters like Gaussian Blur. While some use instrumented gloves to reduce computation time, this project found skin color segmentation to be unreliable due to lighting conditions. To improve accuracy for a large number of similar symbols, this project opted for a stable, single-color background rather than segmentation.

For gesture classification, Hidden Markov Models (HMM) are used for dynamic gestures. Naïve Bayes Classifiers have been used for static gesture recognition based on geometric invariants. Others have used CNNs by first applying a skin model and binary threshold, achieving 95% accuracy on 7 gestures.

3 Research Aim and Objectives

3.1 Aim

To develop a user-friendly Human Computer Interface (HCI) where the computer understands human sign language, specifically for converting American Sign Language (ASL) fingerspelling gestures into textual content in real-time.

3.2 Objectives

- To create a custom dataset for ASL fingerspelling gestures by capturing raw images from a webcam.
- To develop and train a real-time sign detection model using a Convolutional Neural Network (CNN).
- To implement a two-layer algorithm to verify and improve prediction accuracy, especially for symbols that are similar to each other.
- To integrate the gesture recognition model with sentence formation logic and an auto-correct feature to build complete words and sentences.

4 Methodology

The system uses a vision-based approach with bare hands, eliminating the need for artificial devices.

4.1 Dataset Preparation

As no suitable raw image datasets were found, a custom dataset was created. Using OpenCV, we captured frames from a webcam. A Region of Interest (ROI) was defined in the frame. We captured approximately 800 training images and 200 testing images for each ASL symbol. A Gaussian Blur filter was applied to extract features.

4.2 Model Configuration

The core of the system is a Convolutional Neural Network (CNN).

- **1st Conv Layer:** Input image (128x128) is processed with 32 filters (3x3), resulting in a 126x126 image.
- **1st Pooling Layer:** Max pooling (2x2) downsamples the image to 63x63.
- **2nd Conv Layer:** Processed with 32 filters (3x3), resulting in a 60x60 image.
- **2nd Pooling Layer:** Max pooling (2x2) downsamples to 30x30.
- **1st Densely Connected Layer:** 128 neurons. A dropout layer of 0.5 is used to avoid overfitting.
- **2nd Densely Connected Layer:** 96 neurons.
- **Final Layer:** Neurons equal to the number of classes (alphabets + blank symbol).

The **ReLU** activation function is used in each layer to add non-linearity. The **Adam optimizer** is used for updating the model.

4.3 System Integration and Real-Time Detection

The system uses a two-layer algorithm approach for prediction.

- **Layer 1:** The processed image is passed to the main CNN model. If a letter is detected for more than 50 frames, it is printed and added to the word. A blank symbol is used for spaces.
- **Layer 2:** To handle misclassifications among similar gestures (e.g., {D, R, U}, {S, M, N}), three additional classifiers were trained specifically for these subsets.

For sentence formation, if a letter's detection count exceeds 50 and no other letter is close, it is added to the string. A detected blank space predicts the end of a word.

4.4 Interface and Design Principles

The system includes an **AutoCorrect Feature**. A Python library, 'Hunspell suggest', is used to suggest correct alternatives for misspelled words. This assists users in forming complex words and reduces spelling mistakes.

5 System Architecture

The system architecture follows an iterative classification pipeline. Input from the user is first preprocessed (Gaussian Blur and threshold). This input is passed through the primary CNN. A prediction is made. This predicted value is then passed through a second layer of specialized CNNs for verification, especially for similar gestures. If the output is confident, it is added to the output word.

6 Implementation Details

The core software and hardware components used in this project are listed in Table 1. The system was built in Python, using Keras with a TensorFlow backend for the neural network, and OpenCV for real-time computer vision tasks.

Table 1: Tools and Frameworks Used

Component	Tool/Framework
Programming	Python
ML Framework	TensorFlow
NN Library	Keras
Computer Vision	OpenCV
Auto-Correct	Hunspell_suggest
Hardware	Standard Webcam

7 Evaluation Framework

Input images (RGB) were converted to grayscale, and Gaussian blur was applied to remove noise. An adaptive threshold was used to extract the hand from the background, and images were resized to 128x128.

The final prediction layer uses the **SoftMax** function to normalize the output between 0 and 1, estimating the probability for each class. To train the network, **cross-entropy** was used as the performance measurement. This function is positive when values differ from the label and zero when they are equal. We optimized the cross-entropy by minimizing it as close to zero as possible using the **Adam Optimizer**.

8 Reflection

Several challenges were faced during the project. The first issue was the dataset; we could not find an existing raw, square image dataset matching our requirements, so we built our own.

The second issue was selecting the right filter for feature extraction. We experimented with binary threshold, Canny edge detection, and Gaussian blur, ultimately settling on the Gaussian Blur Filter.

Further issues were faced regarding the model's accuracy in earlier phases. This was improved by increasing the input image size and improving the dataset.

9 Results and Discussion

We achieved an accuracy of 95.8% using only Layer 1 of the algorithm. By using the combination of Layer 1 and Layer 2, we achieved a final accuracy of **98.0%** on our dataset.

This accuracy is favorable compared to other research. For example, some studies achieved a 2.5% error rate using Kinect, while others achieved a 10.90% error rate with an HMM classifier. Our model does not use background subtraction, which some other models do, so accuracies may vary if this is implemented.

A significant advantage of this project is its accessibility. Most referenced projects use Kinect devices, which are not readily available and are expensive. Our model uses a normal laptop webcam, making it a great plus point.

10 Conclusion

This report details a functional, real-time, vision-based American Sign Language recognition system for ASL alphabets. We achieved a final accuracy of 98.0% on our dataset. We improved our prediction by implementing two layers of algorithms, which verify and predict symbols that are similar to each other. This provides the ability to detect almost all symbols, provided they are shown properly with no background noise and adequate lighting.

10.1 Future Work

Several directions for future enhancement have been identified:

- Achieve higher accuracy with complex backgrounds by implementing background subtraction algorithms.

- Improve pre-processing to predict gestures in low-light conditions.
- Build the project as a web or mobile application for user convenience.
- Extend the project to work for other native sign languages.
- Enhance the system from a fingerspelling translator to recognize contextual signs (objects, verbs) using Natural Language Processing (NLP).