Real-Time Colombian Sign Language (LSC) Recognition using Skeletal Tracking and Deep Learning

Outils Mathématiques Avancés pour la Science des Données et la Prise de Décision Project Report

Sebastián Diaz Pabón
Institut Polytechnique des Sciences Avancées

November 30, 2025

Abstract

This project addresses the communication gap between the hearing-impaired community in Colombia and the general population by developing a real-time Colombian Sign Language (LSC) recognition system. Unlike traditional methods that rely on raw image processing (CNNs) or expensive hardware, this approach utilizes Google's MediaPipe framework for skeletal tracking to extract 21 distinct hand landmarks. These geometric coordinates serve as inputs for a lightweight Deep Neural Network (DNN) trained to classify static alphabet signs. The result is a computationally efficient system capable of running on consumer-grade hardware (standard webcams) with high accuracy, demonstrating the potential of geometry-based AI in accessibility technologies.

Contents

1 Introduction

Sign language is the primary means of communication for millions of people worldwide. In Colombia, the "Lengua de Señas Colombiana" (LSC) is the official language for the deaf community. However, the majority of the hearing population lacks knowledge of LSC, creating a significant barrier to social inclusion, education, and accessing public services.

Technological solutions to this problem often fall into two categories: invasive hardware (gloves with sensors) which are expensive and impractical for daily use, or vision-based systems requiring high computational power.

This project proposes an accessible software solution using Computer Vision and Artificial Intelligence. By leveraging **MediaPipe** for hand detection and a custom-trained neural network, we aim to translate LSC static gestures into text in real-time using a standard laptop camera.

2 Objectives

2.1 General Objective

To develop a computer vision system capable of identifying and translating static signs from the Colombian Sign Language alphabet into text in real-time.

2.2 Specific Objectives

- To construct a custom dataset of LSC gestures, ensuring variability in angles and distances.
- To implement a feature extraction pipeline using MediaPipe to convert raw images into normalized numerical coordinate vectors (Landmarks).
- To train and validate a Neural Network classifier using TensorFlow/Keras.
- To deploy the trained model in a live video application that overlays prediction results on the user's screen.

3 Theoretical Framework

3.1 Computer Vision and MediaPipe

Computer Vision enables computers to "see" and interpret visual information. A key challenge in hand gesture recognition is robustness against lighting changes and background noise.

MediaPipe Hands is a high-fidelity hand and finger tracking solution employed in this project. It infers 21 3D landmarks of a hand from a single frame. Instead of processing the entire image (thousands of pixels), we extract only the geometry of the hand.

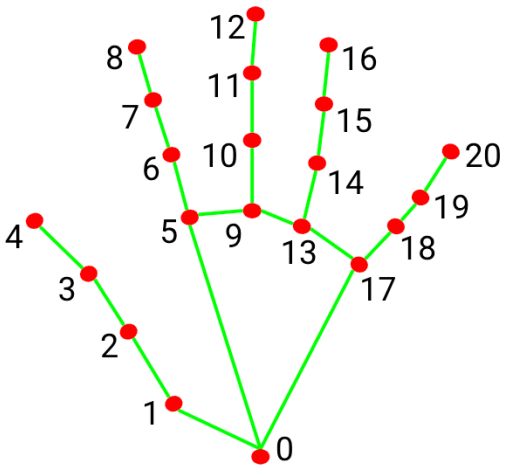

Figure 1: The 21 hand landmarks extracted by MediaPipe (Source: Google Developers).

3.2 Deep Learning Classification

The classification is performed by a Dense Neural Network (DNN). The network receives a vector of coordinates and outputs a probability distribution across possible classes (letters). The class with the highest probability is selected as the prediction.

The model architecture was implemented using the Keras Sequential API, which allows linear stacking of layers. Specifically, we designed a Multi-Layer Perceptron (MLP) or Dense Neural Network (DNN). This architecture consists of fully connected layers (Dense layers) where each neuron processes input from all neurons in the previous layer. This structure is ideal for our feature-based approach, as it effectively learns non-linear relationships between the discrete geometric coordinates (x, y, z) of the hand landmarks

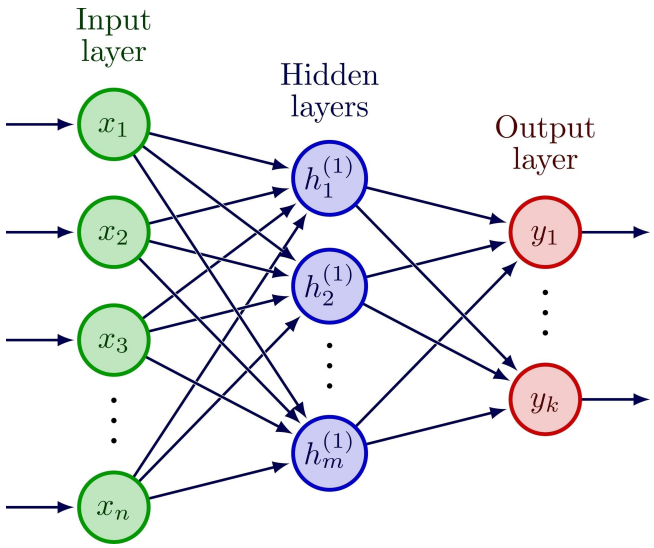

Figure 2: MLP - Multi-Layer Perceptron

3.3 Artificial Neural Networks (ANN)

The core classification engine of this project is based on an Artificial Neural Network (ANN), specifically a Multilayer Perceptron (MLP). An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer.

3.3.1 Activation Functions

To introduce non-linearity into the model, allowing it to learn complex patterns from the hand geometry, we utilize activation functions:

- **ReLU (Rectified Linear Unit):** Used in the hidden layers. It is defined as $f(x) = \max(0, x)$. ReLU is computationally efficient and helps mitigate the vanishing gradient problem, allowing for faster convergence during training.
- **Softmax:** Used in the output layer for multi-class classification. It converts the raw output logits into a probability distribution, ensuring that the sum of probabilities across all classes (LSC letters) equals 1.

3.3.2 Optimization and Loss Function

The training process involves minimizing a loss function using an optimizer.

- **Loss Function:** We employ *Sparse Categorical Crossentropy*, which measures the dissimilarity between the predicted probability distribution and the true class label.
- **Optimizer:** The *Adam* (Adaptive Moment Estimation) optimizer is used to update the network weights. Adam combines the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp, adapting the learning rate for each parameter.

4 Methodology

The project development was divided into four distinct phases:

4.1 Phase 1: Data Acquisition

A custom dataset was created manually. We captured multiple images for selected letters of the LSC alphabet. To ensure model robustness, images were taken with:

- Different distances from the camera.
- Slight variations in rotation and angle.
- Different lighting conditions.

Figure 3: Example Samples Letters

4.2 Phase 2: Preprocessing and Feature Extraction

This is the most critical step where the approach differs from standard CNNs.

1. **Hand Detection:** The image is passed to MediaPipe.
2. **Landmark Extraction:** If a hand is detected, 21 points (x, y, z) are extracted.
3. **Normalization:** Coordinates are normalized to be relative to the image dimensions (0.0 to 1.0), ensuring the hand's position on the screen does not affect the specific gesture geometry.
4. **Dataset Creation:** Instead of saving images, we saved the numerical data into a CSV file. Each row represents a sample, containing 63 columns (21 points $\times$ 3 coords) plus the label.

4.3 Phase 3: Model Architecture

We designed a Sequential Neural Network using TensorFlow/Keras with the following structure:

- **Input Layer:** Accepts 63 values (flattened landmarks).
- **Hidden Layers:** Dense layers with ReLU activation to learn non-linear relationships between finger positions.
- **Dropout Layer:** Added to prevent overfitting by randomly setting inputs to zero during training.
- **Output Layer:** A Dense layer with Softmax activation, where the number of neurons equals the number of classes (letters).

4.4 Phase 4: Real-Time Deployment

The final application captures video from the webcam using OpenCV. For every frame, it extracts landmarks, feeds them to the trained model, and displays the predicted letter and confidence score on the screen.

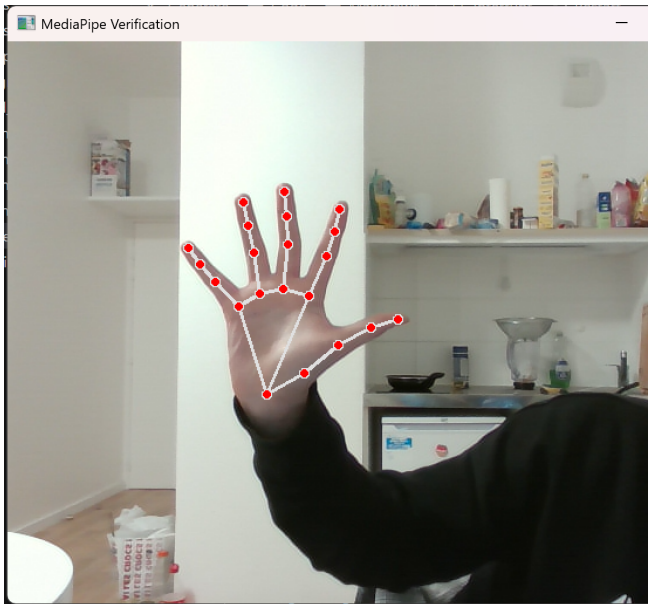

Figure 4: Environment Verification

5 Results

5.1 Model Performance

The model was trained using the custom CSV dataset.

- **Training Accuracy:** The model achieved a high accuracy rate on the training set, demonstrating its ability to learn the geometric patterns of the signs.
- **Validation:** The validation metrics confirmed that the model generalizes well to new data, provided the hand is clearly visible.

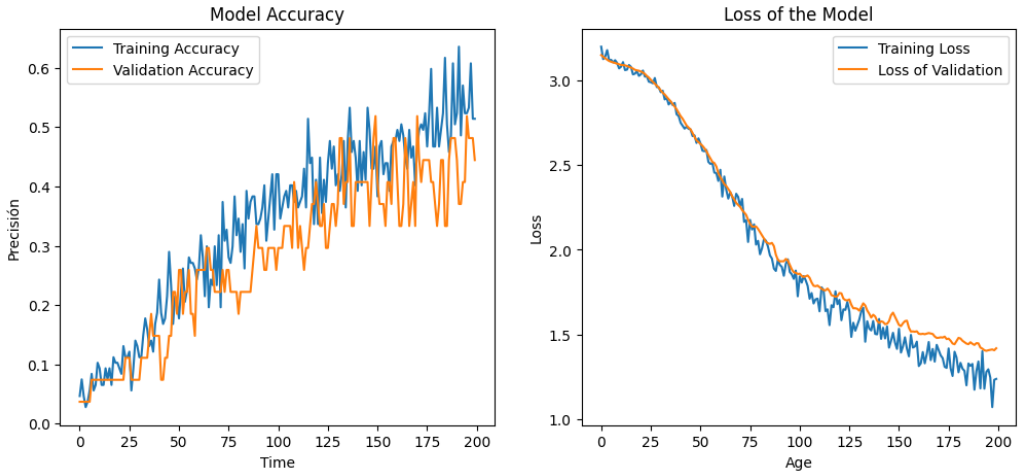

Figure 5: Model Accuracy and Perfomance

Key findings from the training metrics include:

- **Optimal Convergence:** The model achieved its highest validation performance at **epoch 194**.
- **Peak Metrics:** At this specific epoch, the system recorded a validation **accuracy of 55.56%** and a **loss value of 1.43**.
- **Overfitting Observation:** Beyond epoch 194, a divergence was observed where training accuracy continued to rise while validation accuracy fluctuated or decreased. This indicates the onset of overfitting, confirming that 200 epochs is slightly excessive for this specific dataset size.

5.2 Real-Time Testing

During live testing, the system demonstrated:

- **Low Latency:** The inference time is negligible, maintaining a high Frame Per Second (FPS) rate.
- **Robustness:** The landmark-based approach proved superior to pixel-based methods in varying lighting conditions. Since the model relies on the relative position of fingers (geometry) rather than color, shadows had minimal impact.

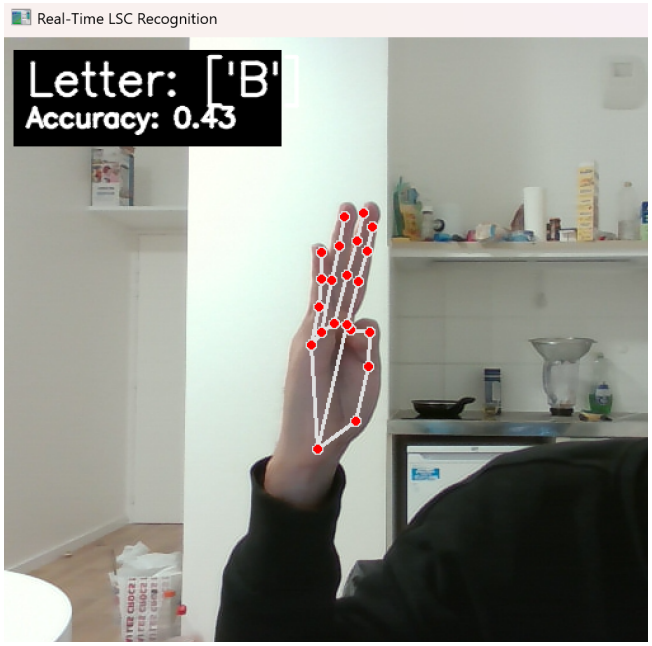

Figure 6: Real Time Validation

6 Code Availability

The complete source code, dataset generation scripts, and trained models developed for this project are available in the following GitHub repository:

Project: Colombian Sign Language LSC Recognition with AI

7 Conclusions

This project successfully demonstrated the viability of using skeletal tracking for sign language recognition. The key takeaways are:

1. **Training Optimization:** Empirical results demonstrate that more training epochs do not always equal better results. We identified that the model reached maximum generalization at **194 epochs (55.56% accuracy)**. Training beyond this point increased the computational cost and introduced overfitting.

2. **Efficiency over Brute Force:** By extracting landmarks, we reduced the input data dimensionality from millions of pixels to just 63 numbers. This allowed for faster training and real-time performance on a standard CPU without needing a dedicated GPU.

3. **Accessibility:** The solution works with a standard webcam, fulfilling the objective of creating an accessible tool.

4. **Limitations:** The current model handles static signs effectively. However, dynamic signs (like 'J' or 'Z' which require movement) are not fully supported by a single-frame static classifier.

8 Future Work

Future iterations of this project will focus on:

- Implementing Recurrent Neural Networks (LSTM) to handle dynamic gestures by analyzing sequences of frames.
- Expanding the dataset to include the full LSC alphabet and common phrases.
- Developing a mobile application to increase accessibility.