1. Project Overview

The objective of this project is to develop a specialized computer vision pipeline capable of detecting and classifying prohibitory traffic signs from real-world environmental images.

2. Technical Pipeline Stages

2.1 Data Acquisition and Preprocessing

- **Source**: Images were sourced from Google Street View to improve the training of the model.
- **Color Space Selection**: I used the **HSV (Hue, Saturation, Value)** color model for initial segmentation. Unlike RGB, HSV separates color information (Hue) from lighting (Value), allowing for more consistent red-color detection across different environments.

2.2 Multi-Strategy Detection

To address the challenges of varying luminosity, I developed four distinct detection approaches:

1. **Standard Detection**: For well-lit, clear images.
2. **Luminosity-Adjusted Detection**: For overexposed or bright images.
3. **Dark Image Detection**: Utilizes **CLAHE** (Contrast Limited Adaptive Histogram Equalization) to enhance contrast in low-light conditions before color extraction.
4. **Hough Circle Transform (Edge Detection)**: Implemented as a fallback for cases where color segmentation is unreliable (e.g., heavily faded signs or complex backgrounds), relying on the geometric circularity of the signs.

2.3 Object Refinement and Segmentation

- **Dynamic Bounding Boxes**: A custom function was developed to "tighten" detection boxes around actual red pixels, reducing background noise.
- **Vertical Signal Separation**: For stacked signals (multiple signs on one pole), the system analyzes the **aspect ratio** of the detection box. If the ratio indicates vertical stacking, the box is automatically divided into individual crops for separate classification.

2.4 Data Augmentation

To improve the robustness of the neural network, I implemented an augmentation pipeline that applies:

- **Geometric Transforms**: Random rotations (+/- 15°), scaling, and affine transformations to simulate different camera perspectives.
- **Photometric Transforms**: Random brightness and contrast adjustments to prepare the model for varying times of day.

- **Noise Filters**: Occasional bilateral blurring to simulate motion blur or low-resolution sensor data.

2.5 Classification via Transfer Learning

- **Architecture**: I chose **MobileNetV2** as the base model. This choice was driven by the model's efficiency and lightweight nature.
- **Transfer Learning Strategy**: The model was pre-trained on ImageNet. I added some layers to classify the 6 specific traffic sign categories.
- **Fine-tuning**: We initially froze the base model to train the new classifier, followed by an optional fine-tuning of the deeper layers to adapt the feature extraction to the specific pictograms of traffic signs.

3. Challenges Encountered and Resolved

Challenge	Solution
Luminosity Variation	Developed specialized detection functions for different light levels and used CLAHE for dark images.
False Positives (Cars/Walls)	Implemented **Circularity Descriptors** and white-center validation to ignore rectangular red objects.
Dataset Imbalance	Managed class weighting and targeted data augmentation for categories with fewer samples (e.g., Category F).

4. Why Not YOLO?

While YOLO is a state-of-the-art object detector, it was not selected for this specific project because:

1. **Small Dataset Performance**: YOLO often requires a very high volume of annotated data to achieve precision. My custom "Segment-then-Classify" approach allows for high accuracy with significantly fewer images.
2. **Granular Control**: This pipeline allows for the manual tuning of mathematical filters (like the Hough Transform or Circularity), providing more transparency and control over what is considered a "prohibitory signal" versus general red objects.

Code used:

Hough circles: https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

Contours, area, perimeter: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html

CLAHE: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

Bilateral filtering: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

Canny: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html