Road Signs Classification

Machine Learning Course - Programming Assignment

Roberto Mastrofrancesco

10th February 2026

1 Introduction

Road sign recognition is a critical component of autonomous driving systems, enabling vehicles to understand and respond to traffic signs effectively. The objective of this report is to train different classification models to correctly classify road signs from a given dataset. Different models of varying complexity have been explored and evaluated, implementing proper feature extraction techniques where necessary.

The notebook from which the results are obtained is run on my personal computer with the following specifications: Intel Core i7-8700K CPU, 32GB RAM, and NVIDIA GeForce GTX 1080Ti GPU.

2 Dataset

The dataset comprises 1088 color images of 200x200 pixels divided into 20 distinct road sign classes. Each class contains a set of training samples and 10 test samples, with the number of training images varying across classes. To proceed with the assignment, an additional small validation set was created from the training samples, obtaining the following configuration:

- Training set: 788 images (not equally distributed across classes due to the original dataset being imbalanced).
- Validation set: 100 images, 5 per class; it has been constructed by randomly selecting 5 images from each class of the original training set.
- Test set: 200 images, 10 per class (as provided in the original dataset).

2.1 Data Augmentation

Due to the relatively small number of training samples, data augmentation techniques were applied to increase the diversity of the training data and improve model generalization. In particular, transformations able to modify the images while preserving the recognizability of the road signs, such as random low-angle rotations ($< 10°$), horizontal flips and small brightness adjustments, were implemented.

Data augmentation was also applied when extracting low-level features (section 3): although augmentation was not re-applied on the fly during training, it still increased the diversity of the training set. This design choice was made to allow consistent standardization of the extracted features; if augmentation were applied on the fly, features would differ at each epoch, making standardization inconsistent across epochs. Moreover, since low-level features were only used with relatively small models (logistic regression and MLPs with few hidden layers, section [HERE]), on-the-fly augmentation would not have provided a significant

performance benefit while being considerably more computationally expensive. For the CNN (section [HERE]), where the higher number of parameters demands stronger regularization, the standard on-the-fly augmentation procedure was followed.

Figure 1: Sample images from the training dataset, showing the effect of data augmentation.

3 Feature Extraction

To train the simpler models, four different types of low-level features were extracted from the images:

- **Color Histogram (C):** Each channel's pixel intensities are quantized into 64 bins and counted, producing a normalized histogram. For RGB images this yields $3 \times 64 = 192$ features, while for grayscale images it yields 64 features. This descriptor captures the global color distribution of the image, regardless of spatial layout.
- **Edge Direction Histogram (E):** Horizontal and vertical gradients are computed via Sobel filtering on the grayscale version of the image. The gradient angle at each pixel is quantized into 64 bins and accumulated using the gradient magnitude as weight, producing a normalized 64-dimensional descriptor. This captures the dominant edge orientations present in the image.
- **RGB Co-occurrence Matrix (R):** Each channel is quantized into 3 levels and the channels are combined into a single code per pixel ($3^3 = 27$ possible values for RGB, 3 for grayscale). Co-occurrence matrices are then computed along the horizontal and vertical directions at a distance of 10 pixels, symmetrized, and flattened into a normalized feature vector of $27^2 = 729$ features (or $3^2 = 9$ for grayscale). This descriptor captures local texture patterns by encoding spatial relationships between neighboring pixel values.
- **Histogram of Oriented Gradients (HOG, H):** Computed using the `scikit-image` implementation with 9 orientation bins, 32×32 pixel cells, 2×2 cell blocks, and power-law (square root) compression. The image is divided into a grid of cells; within each cell, gradient orientations are binned and weighted by magnitude, then normalized across overlapping blocks. This produces a descriptor that effectively encodes local shape and edge structure, yielding 900 features for the 200×200 images.

All features were extracted once from the (augmented) training images and standardized to zero mean and unit variance, with the same transformation applied to the validation and

test sets using the training statistics. The resulting feature vectors were then concatenated when multiple feature types were used together.

4 Low-Level Features Models

This sections presents the different models used for classification along with their performance metrics.

4.1 Logistic Regression

The first classifier is a multinomial logistic regression model. Given an input feature vector $\mathbf{x} \in \mathbb{R}^d$, the model computes class logits via an affine transformation:

$$\mathbf{z} = W\mathbf{x} + \mathbf{b}, \qquad W \in \mathbb{R}^{K \times d},\ \mathbf{b} \in \mathbb{R}^K, \tag{1}$$

where $K = 20$ is the number of classes and d is the dimensionality of the feature vector. The predicted probability for class k is obtained via the softmax function:

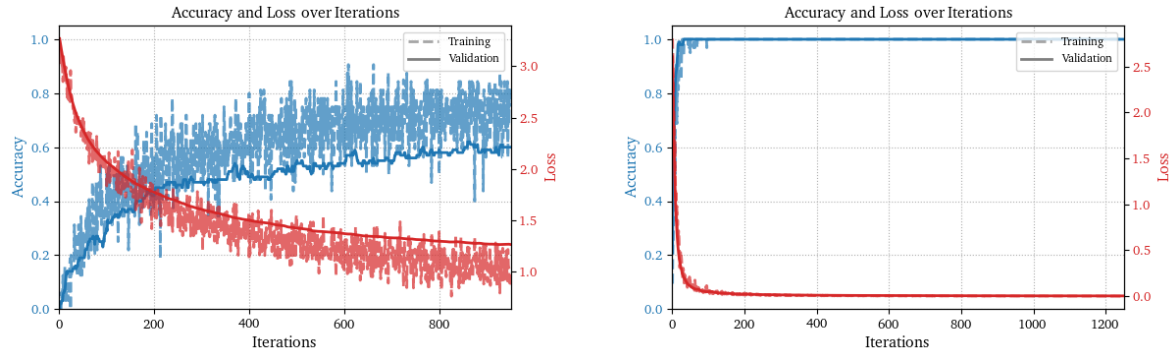
$$P(y = k \mid \mathbf{x}) = \frac{\exp(z_k)}{\sum_{j=0}^{K-1} \exp(z_j)}. \tag{2}$$

The model is trained by minimizing the cross-entropy loss over the training set:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=0}^{N-1} \log P(y = y_i \mid \mathbf{x}_i), \tag{3}$$

using the Adam optimizer with a learning rate of 10^{-3}. A maximum training of 50 epochs together with early stopping with a patience of 25 steps on the validation loss is employed to prevent overfitting. To determine which features are most informative, logistic regression was trained on all 15 possible combinations of the four feature types (C, E, R, H).

Example of training curves can be found in figure 2, where it's evident that convergence is largely achieved much earlier than the maximum number of epochs.



(a) Edge Direction Histogram.

(b) Histogram of Oriented Gradients.

Figure 2: Logistic regression training curves for different low level features.

From figure 3 it's evident that any feature combination containing HOG (H) performs significantly beter than any other combination. Also, a significant step can be noticed for the combinations containing the *Edge Direction Histogram* (E). The reason behind the performance of the HOG descriptor is in it's superior ability to capture local shape and edge information, which is crucial for distinguishing between different road signs, as can be visually observed in figure 4 (the same reasoning, to a lesser extent, applies to the E descriptor).

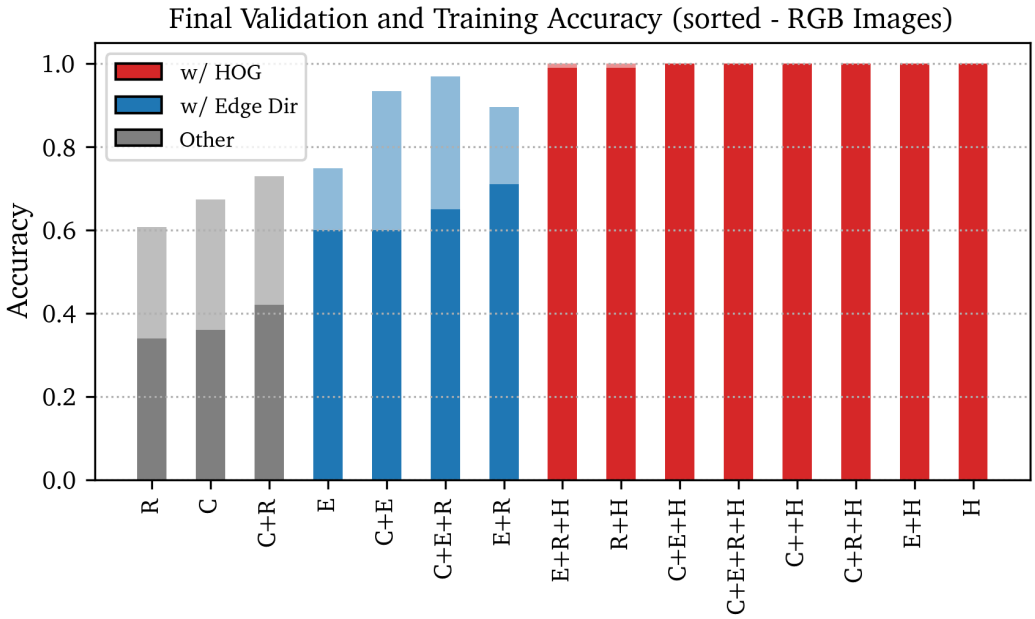


Figure 3: Summary of training (semi-transparent) and validation (solid) performance for different low-level feature combinations with logistic regression.

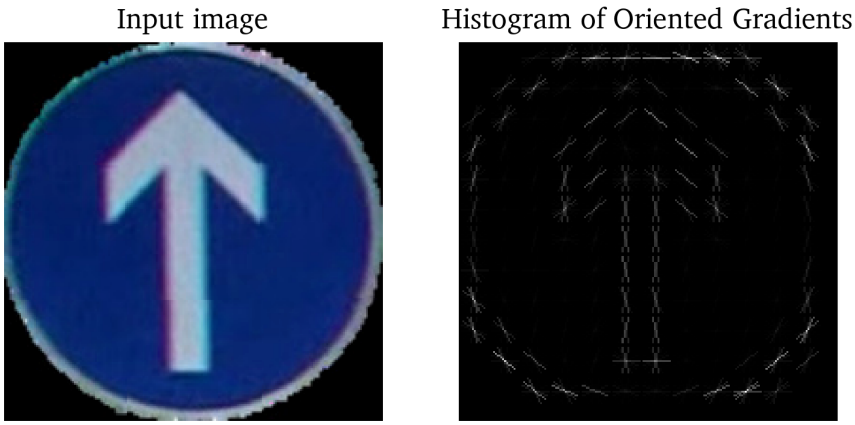


Figure 4: Visualization of the *Histogram of Oriented Gradients* for a sample image. It's evident how HOG features perfectly capture shape and edge information, crucial for road sign classification.

4.2 Feature reduction

Having identified HOG as the most efficient feature type, a further step was to evaluate whether reducing the dimensionality of the HOG features could lead to a deterioration in performance. The feature reduction procedure was split into two steps:

- **removal of low-variance features**: only features with variances above a set percentile were retained; using the top 30% variance features reduced the feature dimensionality from 900 to 270 while maintaining the same validation accuracy of 100%.
- **Principal Component Analysis (PCA)**: PCA was applied to the HOG features to further reduce dimensionality while retaining 95% of the variance. This resulted in a further reduction from 270 to 57 features, with a slight decrease in validation accuracy

to 97%.

The final performance of the logistic regression model with the reduced HOG features was evaluated on the test set, achieving an accuracy of 97.5%, confirming that the dimensionality reduction did not significantly impact the model's ability to generalize to unseen data.

4.3 Grayscale

The Logistic Regression model was then trained on grayscale images, using properly extracted features (C, E, R, H). In figure 5 the same recap of training and validation performance can be found. As expected, HOG and Edge Direction Histogram (E) performances are almost unchanged, since they capture only shape and edge information, not depending on color. On the other hand, the performance of the color histogram (C) and RGB co-occurrence matrix (R) features significantly deteriorates, since they are not able to capture any useful information from grayscale images.

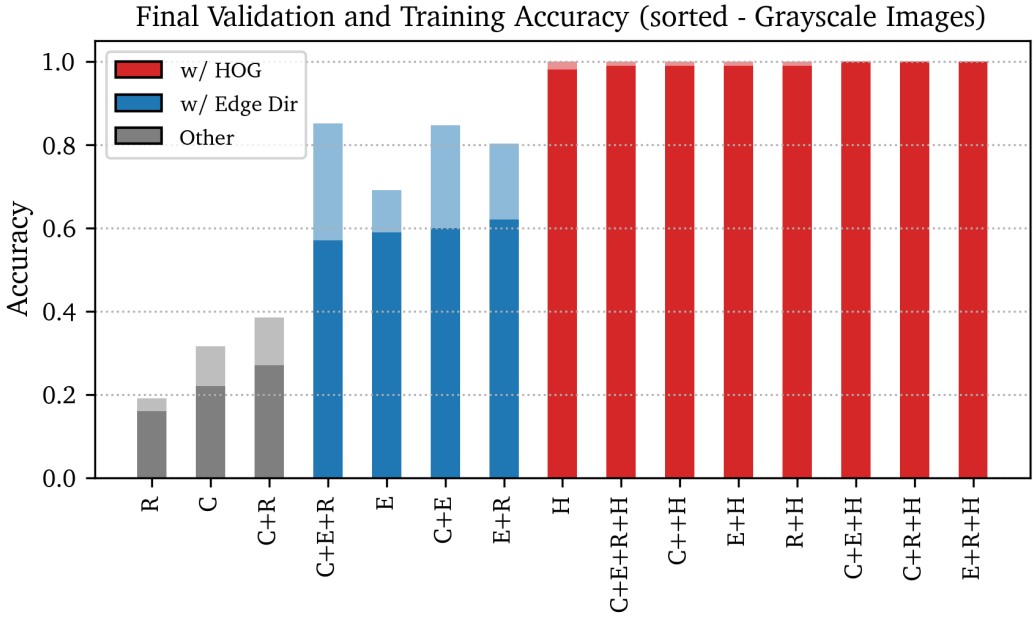


Figure 5: Summary of training (semi-transparent) and validation (solid) performance for different low-level feature combinations with logistic regression on grayscale images.

4.4 Multi-Layer Perceptron

Given that logistic regression already achieved near-perfect accuracy with HOG features (a quick test with a two-hidden-layer MLP, 50 and 30 units, on the same reduced HOG features confirmed this, yielding comparable results) the more interesting question is whether an MLP can improve classification performance on the remaining feature types (C, E, R). By introducing hidden layers with ReLU activations, the MLP can learn non-linear decision boundaries that a single linear layer cannot capture, potentially extracting more discriminative information from features like color histograms and co-occurrence matrices.

Different MLPs have been trained on C, E, R feature combinations with same max epochs and patience as the logistic regression. Adam optimizer and cross entropy loss were chosen as well. The additional peculiarities of these architectures are:

- **Feature Reduction**: a more conservative approach was followed, retaining the top 50% variance features without applying PCA. Unlike logistic regression, MLPs can learn to ignore irrelevant features through training, so aggressive dimensionality reduction is unnecessary and could discard useful patterns.
- **Hidden layer structure**: the number and width of hidden layers were determined automatically from the input dimensionality using a reduction factor of 3. Starting from the input size, each successive hidden layer is a third as wide as the previous one, and layers are added until the width falls below $2K = 40$ (twice the number of classes). This ensures a gradual bottleneck without excessive parameterization.

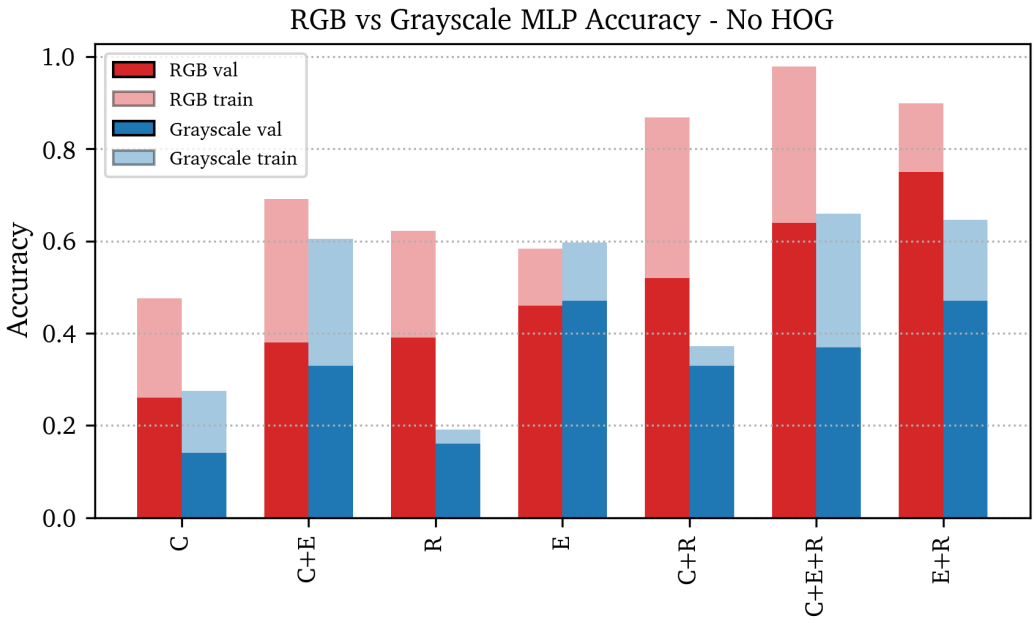


Figure 6: Comparison of performance (both training and validation) on color and grayscale images with different MLPs.

As is evident from figure 6, while some improvement can be noticed for the combination of *Edge Direction Histogram* and *RGB Co-occurrence Matrix*, the remaining combinations do not benefit significantly from the increased complexity. For grayscale images often the performance even deteriorates, likely due to the reduced amount of information contained in the features, not sufficient to train properly the larger number of parameters. A general tendency to overfit can be noticed across the board, especially for RGB images.

5 Convolutional Neural Network

Finally, a simple Convolutional Neural Network (CNN) architecture was implemented and trained on the raw images to see how it would perform compared to simpler models with low-level features and especially HOG. The architecture, summarized in table 1, consists of three convolutional blocks followed by two fully connected layers. Each convolutional block includes a convolutional layer with ReLU activation, batch normalization, and max pooling for downsampling. The final fully connected layers reduce the feature dimensionality to the number of classes (20) for classification.

The training was performed using the Adam optimizer with an initial learning rate of 10^{-3}, a batch size of 32, and the same early stopping criteria as the previous models, with an

Table 1: CNN architecture. $C = 3$ for RGB, $C = 1$ for grayscale.

Block	Layers	Output Shape
Input	—	$C \times 200 \times 200$
Conv Block 1	Conv2d(5×5, 16), BN, ReLU, MaxPool(2)	$16 \times 100 \times 100$
Conv Block 2	Conv2d(3×3, 32), BN, ReLU, MaxPool(2)	$32 \times 50 \times 50$
Conv Block 3	Conv2d(3×3, 64), BN, ReLU, MaxPool(2)	$64 \times 25 \times 25$
Flatten	Flatten	40,000
FC Layer 1	Linear(256), ReLU, Dropout(0.5)	256
FC Layer 2	Linear(K)	$K = 20$

increased patience (200 steps). A scheduler was added to reduce the learning rate when no improvement was observed for 50 steps, down to 10^{-6}. Data augmentation was applied on the fly during training to enhance generalization as explained in section 2.1.

As expected, training converged more slowly than the previous models due to the increased complexity and number of parameters (figure 7). However, the CNN achieved a validation accuracy of 99% and a test accuracy of 100%, outperforming all previous models, including those using HOG features. Similar results were obtained when training the same architecture on grayscale images, confirming the CNN's ability to learn robust features directly from the raw pixel data.

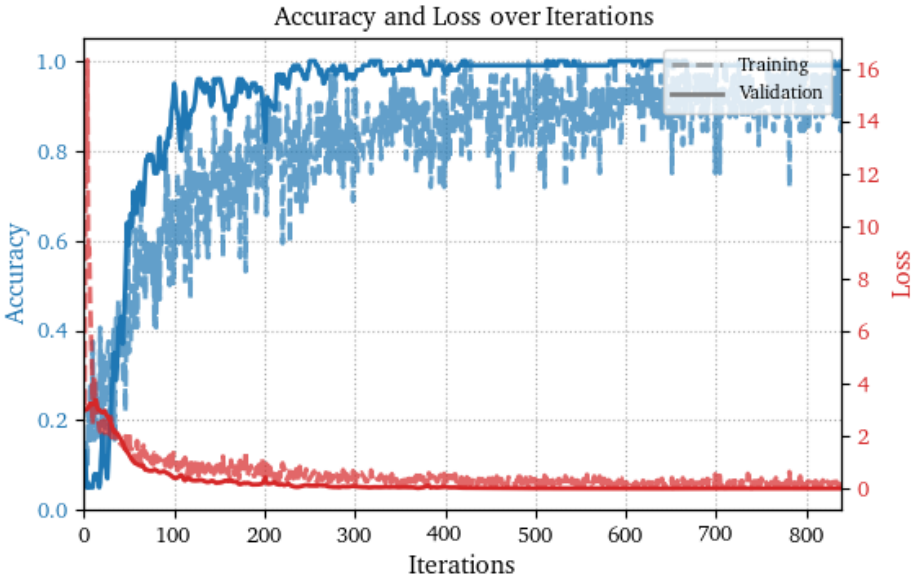


Figure 7: CNN training curves for RGB images.

5.1 Activation Maps

An interesting observation can be made by comparing grayscale and RGB activation maps of the first convolutional layer (figure 8). The RGB activation maps show a stronger focus on color-specific regions, while the grayscale maps rely more heavily on contrast and shape information. This highlights how the CNN adapts its feature extraction process based on the input data, effectively learning to utilize the most relevant information for classification.

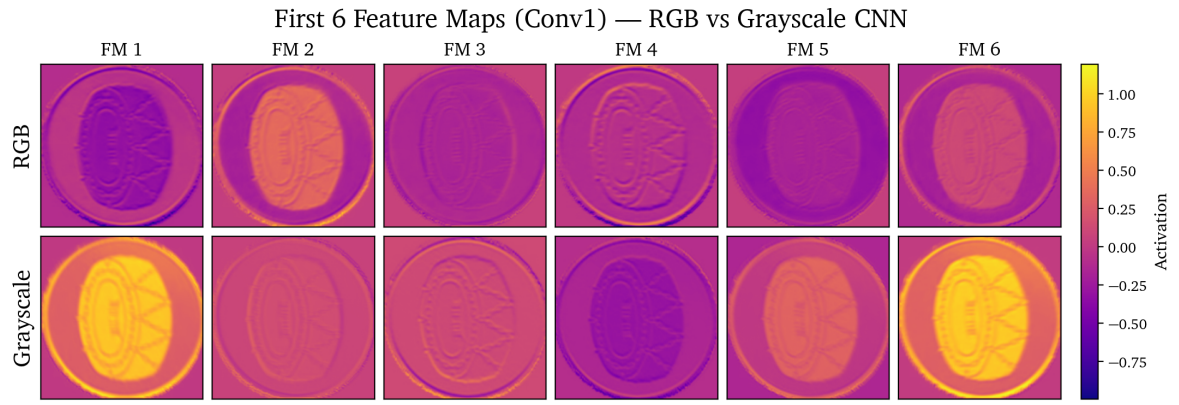


Figure 8: Comparison of first-layer activation maps for RGB and grayscale inputs. The RGB maps show stronger responses to color features, while the grayscale maps focus more on edges, shapes and contrast.

6 Conclusion

This report explored road sign classification using models of increasing complexity, from logistic regression on hand-crafted features to a convolutional neural network operating on raw pixels. The results highlight two key findings. First, among low-level features, HOG proved overwhelmingly dominant: its ability to encode local shape and edge structure allowed even a simple linear classifier to reach near-perfect accuracy, while color- and texture-based descriptors (C, R) provided limited discriminative power on their own. Introducing non-linear models (MLPs) on the non-HOG features yielded only marginal improvements and often led to overfitting, suggesting that the bottleneck lies in the features themselves rather than in model capacity. Second, the CNN achieved the best overall performance (100% test accuracy) while requiring no manual feature engineering, confirming that end-to-end learning can surpass carefully designed pipelines when sufficient model capacity and regularization are provided. A comparison between RGB and grayscale inputs further revealed that color information, while beneficial for hand-crafted descriptors like C and R, is largely redundant for shape-aware methods (HOG, E) and for the CNN, which learns to compensate through its convolutional filters.