DL Applications Term Poster

Few-Shot Learning on Yolov8 for traffic sign detection

Tirdod Behbehani
Alejandro Delgado
Alex Malo

Contents

June 20, 2025

1 Introduction

Modern self-driving cars need fast and accurate object detection, especially for traffic signs, which are small, varied, and crucial for safe navigation. While deep learning has outperformed traditional heuristic-based methods, many models sacrifice speed for accuracy.

CNN-based detectors like Faster R-CNN and RetinaNet are accurate but slow, whereas the YOLO family prioritizes real-time performance. YOLOv8, launched in 2023, offers speed and advanced features like anchor-free detection and Focal Loss, but its effectiveness in low-data traffic sign detection remains underexplored.

This project evaluates YOLOv8's performance across varying data sizes, starting with a few-shot setup (at least four examples per class). We conduct error analysis, scale the dataset, and track improvements to assess how well YOLOv8 learns with minimal supervision. Finally, we test the model on real-world dash cam footage from four YouTube videos.

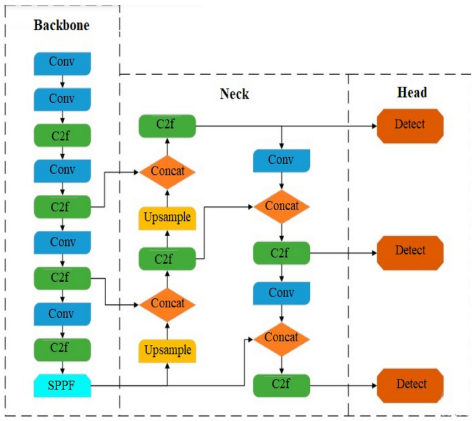

Figure 1: Diagram of the YOLOv8 architecture

2 Literature Review

It is important to clarify that our work is limited to the identification of traffic signs and objects within dashcam-style images. In contrast, real-world autonomous vehicles integrate object detection modules like ours into broader decision-making systems powered by deep reinforcement learning (RL) and traditional machine learning approaches. As discussed in [2] and [5], RL algorithms excel at learning through trial-and-error interaction with the environment, requiring no manual labels and achieving strong safety performance in realistic driving scenarios.

Although the most recent version of the YOLO object detection family is YOLOv11, as described in [4], we opted to use YOLOv8 in this project. This choice was motivated by several factors: (1) YOLOv8 is more suitable for training on resource-constrained hardware, such as our 4GB VRAM GPU, and (2) it benefits from mature, well-documented training pipelines, making it easier to implement and reproduce results consistently.

3 Dataset

This dataset, extracted from Kaggle [1], is formed by a wide variety of pictures containing images of 10 different types of traffic signs. These pictures range in lighting, angles, distance, and quality. Additionally, many images come in a dash-cam format, perfectly adapted to autonomous vehicles.

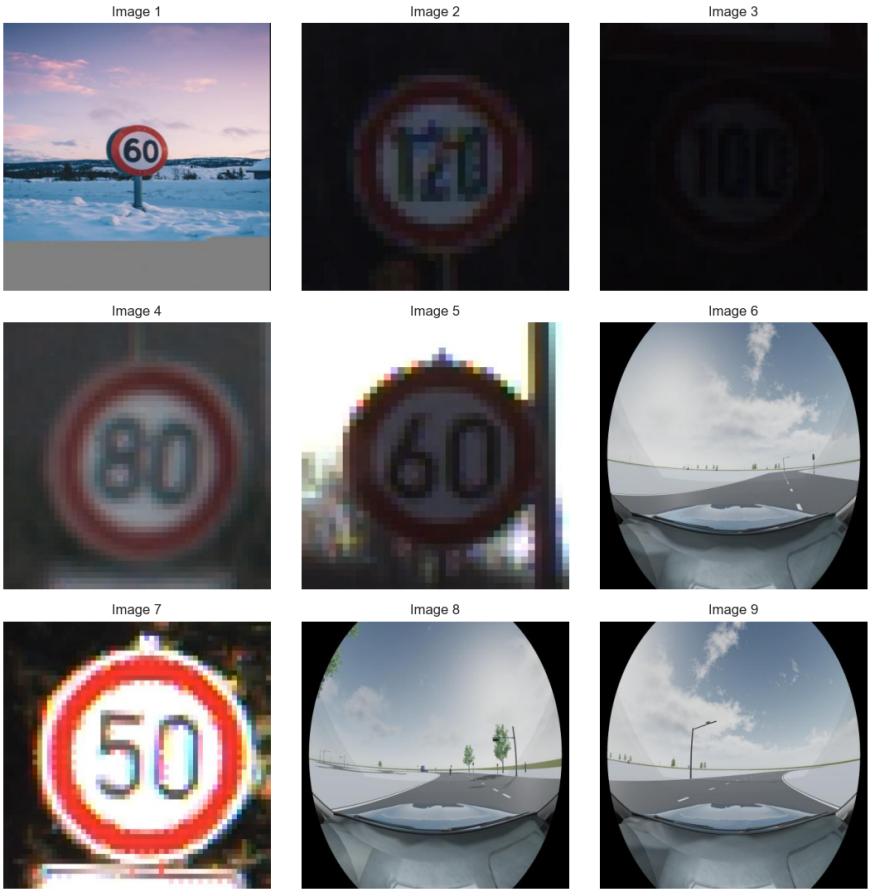

Figure 2: Random unlabeled image sample of the dataset

The initial data analysis explores how the dataset is formatted, how labels are set, and the class distribution. Image labels are object detection boxes, and each image has at least one, but can have multiple. The bounding box annotations follow the YOLO format, using normalized coordinates for center point, width, and height. Each image contains at least one labeled sign, with some containing multiple instances of the same or different classes.

The dataset has **3530 images in the training set**, 801 in validation and 638 in the test set. Images have dimensions of 416x416 and 3 channels.

As shown in Figure 3, the dataset has a highly imbalanced class distribution.

Below are a few considerations regarding the class distribution:

- Given the extreme class imbalance of Speed Limit 10, we decide to drop it. We will however keep Speed Limit 110 as it is a significantly imbalanced class which can perhaps help deliver a valuable Error Analysis later.

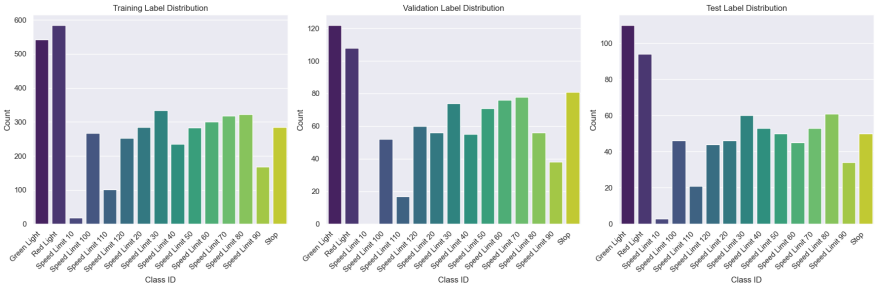

Figure 3: Class distribution across train, validation and test set

- We also examine the different relations between confidence and F1 score for each class, potentially adapting class-specific confidence thresholds at inference time, to further account for the class imbalance.

After dropping speed limit, we are left with **14 classes**: *Green Light, Red Light, Stop, Speed Limit 20, Speed Limit 30, Speed Limit 40, Speed Limit 50, Speed Limit 60, Speed Limit 70, Speed Limit 80, Speed Limit 90, Speed Limit 100, Speed Limit 110, Speed Limit 120.*

To effectively evaluate the results of fine tuning `YOLOv8` with different dataset sizes, we design a sampling pipeline that randomly selects images keeping the same class distribution for all percentages. The smallest dataset is the Few-Shot which uses a minimum of 4 examples for the rarest label, and a maximum of 20 for the most common one. the total dataset size for the Few-Shot training is $\approx 6\%$ of the full dataset.

4 Methodology

We used an NVIDIA RTX 3050 Ti GPU with 4GB of VRAM for both training and inference. Due to hardware constraints, the batch size was set to 4. All data splits were trained for 25 epochs, except for the final split (100% of the dataset), which was trained for 50 epochs to maximize performance, as it was used for predictions on dash cam video clips.

For training, we relied on YOLOv8's default composite loss function, which includes standard cross-entropy for classification, along with box regression loss and distribution focal loss (DFL). No custom loss functions were introduced. Additionally, we did not freeze any model layers — the entire network was fine-tuned during training. The optimizer used was AdamW, and no data augmentation techniques were applied, again due to hardware limitations.

Since YOLOv8 isn't a Zero-Shot detector, it must be trained on a fixed label set. Our model, pretrained on COCO 2017, includes general classes (car, person, etc.) but no traffic signs. Labels are treated as integers, so the model can't generalize to unseen classes at test time.

In contrast, Zero-Shot detectors use language models (e.g., BERT) to embed class names, allowing them to generalize to new labels (e.g., "blue motorcycle") by comparing visual and semantic features.

5 Results

This section contains the results analysis for all the models trained in this project. Our most important metrics are precision, recall, AP and mAP. `map@50` measures a more loose object detection, while `mAP@50-95` is stricter and more robust, penalizing more poor box fitting. Therefore, `mAP@50` is easier to optimize and to compare as a baseline, while a real life model would prioritize `mAP@50-95`.

(a) Few-Shot (6% of the data)

Table 1 shows the analytical metrics for the Few-Shot model.

Class Name	Precision	Recall	F1	AP@0.50	AP@0.50:0.95
Green Light	0.663	0.287	0.400	0.401	0.240
Red Light	0.776	0.354	0.486	0.473	0.271
Speed Limit 100	0.465	0.404	0.432	0.448	0.412
Speed Limit 110	0.316	0.235	0.270	0.270	0.244
Speed Limit 120	0.695	0.650	0.672	0.687	0.611
Speed Limit 20	0.648	0.786	0.710	0.803	0.671
Speed Limit 30	0.516	0.473	0.493	0.484	0.442
Speed Limit 40	0.619	0.582	0.600	0.648	0.551
Speed Limit 50	0.770	0.493	0.601	0.691	0.609
Speed Limit 60	0.709	0.658	0.683	0.742	0.664
Speed Limit 70	0.813	0.628	0.709	0.764	0.683
Speed Limit 80	0.468	0.393	0.427	0.456	0.388
Speed Limit 90	0.330	0.211	0.257	0.215	0.191
Stop	0.953	0.765	0.849	0.843	0.769

Table 1: Per-class performance metrics for the model with few-shot training

The largest Precision–Recall gaps are found in traffic lights, which, despite being the most common classes, perform poorly in F1 score—likely due to distant or unclear images.

Speed Limit 90 performs worst overall, with no clear reason, suggesting possible label noise. *Speed Limit 110*, though the rarest class, performs worse than other low-sample classes, implying that YOLOv8 can learn from few examples if the signal is clean.

The best results come from the *Stop* sign (95% Precision), but its 77% Recall indicates underprediction. Similarly, *Speed Limit 50* shows strong F1 but a large Precision–Recall gap, suggesting missed true positives.

We focus further analysis on these unexpected cases: *Green Light, Red Light, Speed Limit 50*, and *Speed Limit 90*.

Before doing so, let's look at a few more performance evaluators to gain a deeper insight into the results.

Results from the Confusion Matrix (figure 4)confirm some of the Hypothesis from the initial interpretation of results.

One interesting thing we notice is that most prediction errors for speed limit signs come from confusion with adjacent values (e.g., 100 vs. 110), indicating the model

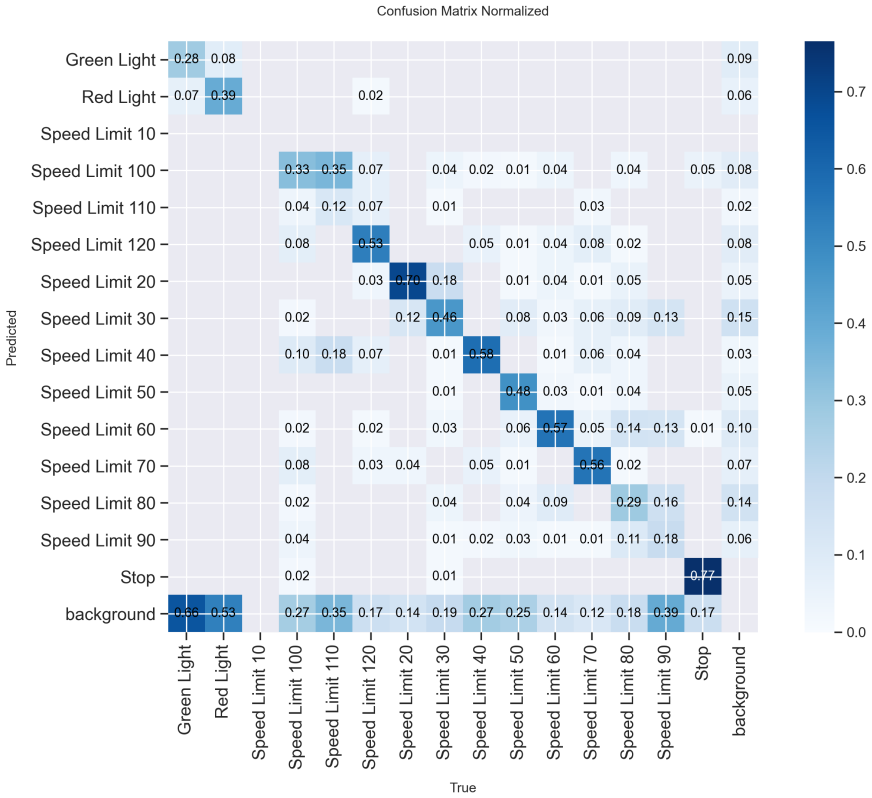

Figure 4: Normalized Confusion Matrix for Few-Shot Model

has learned general patterns but struggles with fine-grained distinctions. Traffic signs are rarely confused with speed limits, showing class-level separation, but many false negatives result from missed detections rather than misclassifications—suggesting a lower confidence threshold might improve recall, as supported by the F1 vs. confidence plot. The model also rarely produces false positives on background. Visual inspection reveals:

- Lighting has little effect, but extreme angles reduce accuracy.
- Traffic lights are better detected against clear skies than cluttered backgrounds.
- Blurry images often lead to misses, even for humans.
- Speed Limit 30 is often confused with 20, even in ideal conditions—likely due to weak training signal.

In sum, the model learns well from few examples but would benefit from more varied data. Given its high precision, Focal Loss may not be necessary at this point.

(b) Dataset Increment

Below are two plots that show the evolution for some of our most important metrics against dataset increment, as well as some overall conclusions:

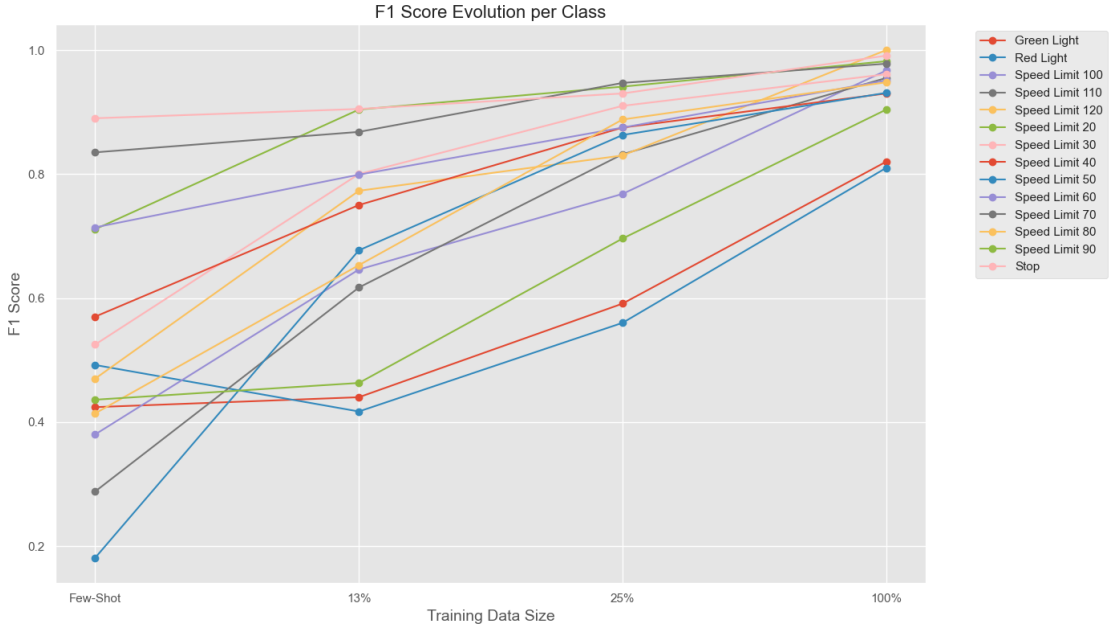

Figure 5: Evolution of F1 score (per label) against dataset increment

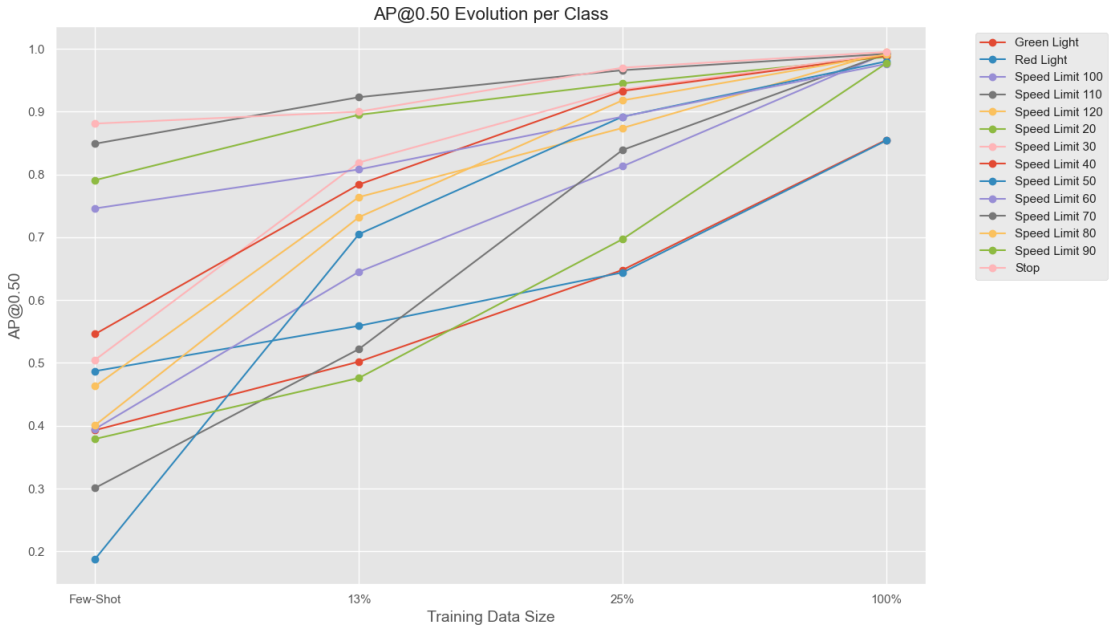

Figure 6: Evolution of AP@50 score (per label) against dataset increment

The results demonstrate that YOLOv8 is capable of learning meaningful traffic sign representations even under few-shot conditions, with several classes (e.g., Stop, Speed Limit 70) achieving high F1 and AP@0.50 scores with as little as 6% of the data. As expected, performance improves consistently across all classes as more data is added, confirming the benefit of scaling supervision. However, some classes—such as Red Light or Speed Limit 90—show slower improvement, likely due to visual similarity. Overall, while YOLOv8 is robust with minimal data, full training still yields the best results, justifying its use for real-world inference scenarios like dashcam video prediction.

(c) Final Model Performance

Below we can see the performance metrics for the full model (50 epochs):

Class Name	Precision	Recall	F1	AP@0.50	AP@0.50:0.95
Green Light	0.871	0.775	0.820	0.855	0.531
Red Light	0.857	0.769	0.810	0.854	0.530
Speed Limit 100	0.936	1.000	0.967	0.993	0.898
Speed Limit 110	0.913	1.000	0.955	0.992	0.926
Speed Limit 120	1.000	0.999	1.000	0.995	0.924
Speed Limit 20	0.983	0.982	0.982	0.987	0.871
Speed Limit 30	0.948	0.973	0.961	0.992	0.924
Speed Limit 40	0.883	0.982	0.930	0.989	0.888
Speed Limit 50	0.962	0.901	0.931	0.980	0.872
Speed Limit 60	0.972	0.930	0.951	0.976	0.887
Speed Limit 70	0.983	0.974	0.978	0.992	0.903
Speed Limit 80	0.932	0.964	0.948	0.991	0.872
Speed Limit 90	0.970	0.847	0.904	0.977	0.798
Stop	0.982	1.000	0.991	0.995	0.934

Table 2: Per-class performance metrics after training on the full dataset

As shown, the model achieves strong performance across all classes, with Green Light and Red Light being the lowest-performing categories—yet still maintaining F1 scores above 81%.

We then applied our trained model to classify four clips extracted from [3], [6], [8], and [7]. For computational convenience, these clips were grouped into two separate videos. The full evaluation videos can be accessed at the following link: `https://drive.google.com/drive/folders/1NWW1kwtiTf99BjV13quUKxWztANMQbCw?usp=sharing`.

Some interesting findings derived by this inference were:

The model doesn't capture well distant signs, but is able to capture closer sings effectively, as we can see in the two following images (this can be due to the image size passed during inference):

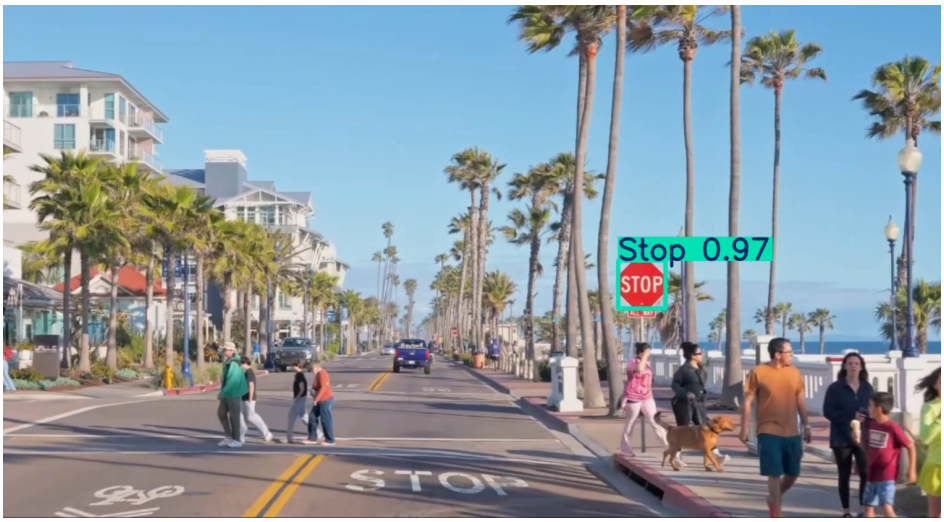

Figure 7: Correct detection of a stop sign

Figure 8: No detection of a stop sign

Also, we can see some cases of false positives given signs that the model have never seen, as for example:

Figure 9: False Positive

The model also struggles to detect signs that are distorted or viewed at unusual angles, compared to those that are perfectly aligned. For example:

Figure 10: Speed limit detection

Lastly, we evaluate the model's performance under nighttime conditions. In the following image, we observe that with the 0.4 confidence threshold used during inference, the model barely detects a nearby green light, while more distant ones go undetected.

Figure 11: Night Lights Detection

In conclusion despite being trained on only 3,000 samples (many of which differed significantly in format from the target video data) our model demonstrated strong classifying performance. While it struggled somewhat with distant traffic signs, the majority of its predictions were accurate. A promising avenue for improvement would be to use higher-resolution images during both training and inference. Currently, image downscaling is required due to hardware limitations, but addressing this constraint could enhance the model's ability to detect smaller or distant objects more effectively. Additionally, training on a larger and more diverse dataset would enable the model to learn a broader range of classes and reduce the occurrence of false positives—such as the incorrectly predicted red traffic light shown earlier.

References

[1] Parisa Karimi Darabi. *Traffic Signs Detection Using YOLOv8*. Kaggle Notebook. Accessed: 2025-06-20. Jan. 2025. URL: https://www.kaggle.com/code/pkdarabi/traffic-signs-detection-using-yolov8.

[2] Haocheng Guo et al. "Research on vehicle detection based on improved YOLOv8 network". In: *Scientific Reports* 14.1 (2024), p. 6777. DOI: 10.1038/s41598-024-55045-7.

[3] Relaxing Scenes – Driving. *Driving Southern California San Diego Coast in 8K Dolby Vision HDR – Oceanside to Pacific Beach*. YouTube. Accessed: 2025-06-20; https://www.youtube.com/watch?v=TKXHxRzvNIEt=57s. 2023.

[4] Ultralytics. *YOLOv11 Models — Ultralytics Documentation*. https://docs.ultralytics.com/es/models/yolo11/. Accessed: June 20, 2025. 2025.

[5] Fei Ye et al. "A Survey of Deep Reinforcement Learning Algorithms for Motion Planning and Control of Autonomous Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2022), pp. 7382–7400. DOI: 10.1109/TITS.2021.3105581.

[6] YouTube. *350km Drive on Italian Highways 4K | Affi, Verona to Torrita di Siena*. YouTube. Accessed: 2025-06-20; https://www.youtube.com/watch?v=V13BNyRd5o8t=10415s. 2023.

[7] YouTube. *Driving New York City 4K HDR – Midtown Manhattan Lights*. YouTube. Accessed: 2025-06-20; https://www.youtube.com/watch?v=7s4-DVIs6Pkt=3173s. 2023.

[8] YouTube. *Rome 4K – Morning Drive – Driving Downtown*. YouTube. Accessed: 2025-06-20; https://www.youtube.com/watch?v=ytiM1nMv$_xUt = 182s$. 2021.