REPORT ABOUT SMART VEHICULAR SYSTEMS PROJECT

Bedei Andrea(matricola 0001126957)
Bertuccioli Giacomo Leo(matricola 0001136879)
Notaro Fabio(matricola 0001126980)

28th November 2024

Contents

Chapter 1

INTRODUCTION

1.1 The project

The following document represents the technical report about our Smart Vehicular Systems project.

Our proposed project is a variation of proposal number 12, speed limit assist, integrated with a traffic light detection system and with small corrective actions autonomously taken by the vehicle in interesting circumntances related both to speed limits and traffic lights violation.

More in detail, the goal of our project is the implementation of an integrated system composed by:

- a detector for speed limit signs that informs the driver about current limit and takes corrective actions in case of violation
- a detector for traffic lights that informs the driver about the color of next detected traffic light and takes corrective actions in case of violation of a red traffic light.

1.2 Motivations

We have chosen the development of this project for many reasons:

- attempt to reduce the road accidents caused by speeding
- attempt to reduce road accidents caused by red traffich light violation
- attempt to reduce the fines and contraventions caused by speeding and red traffic light violation.

According to recent and dramatic statistics about road security, in fact, the number of accidents caused by speeeding or traffic light violation is growing

more and more.
Only in Italy:

- "accidents with injuries in 2022: 165889, or 454 every day"[1]
- "under which circumstances: 13.7% for failure to give way and 9.3% for speeding"[1]
- "with how many social costs: 17.9 billion €, or 0.9% of Italian GDP"[1]
- "victims and injured: 204728, or 561 every day"[1]
- "compared to 2021, in 2022 there was a 9.2% increase in accidents and injuries and a 9.9% increase in victims"[1]
- "almost 4 out of 10 fines are for speeding"[1].

These dramatic data pushed us to develop an integrated system able to reduce the risk of verification of road accidents, helping the driver in being more responsible and preserving his health, following the global and technological effort in developing useful ADAS (Advanced Driver Assistance Systems) for reducing the human error and saving lifes.

Chapter 2

REQUIREMENTS

The requirements of the project can be split into two main categories: functional requirements collect what the system does, while non-functional requirements collect which relevant feature the system must have.

2.1 Functional requirements

The main functionalities of the system are:

1. speed limit detection
2. last speed limit report to the driver
3. vehicle speed reduction in case of violation of the limit in order to respect it
4. vehicle speed limitation in order to deny the potential violation of the last detected speed limit sign
5. traffic light detection
6. last traffic light report to the driver
7. vehicle stopping or slowdown (whichever is safer) when violating a red light
    - 7.1 in this situation intervention of a system to signal the danger to the driver and all surrounding actors (other drivers, pedestrians, etc.) by activating hazard warning lights
8. reporting relevant events (limit detected, traffic light color detected, etc.) on an MQTT broker.

2.2 Non-functional requirements

The main features of the system are:

9. reactiveness → having to operate in real-time scenarios, the ADAS must be able to acquire images at a high frequency and process them quickly to understand the environment around it and take the necessary actions

10. robustness → the ADAS must ensure adequate operation regardless of the weather (sun, fog, rain) and light conditions (day/night) encountered

11. configurability → it is necessary to ensure that the driver can choose between two system configurations at any time, so that he can choose his preferred configuration → the informative configuration simply returns to the driver information on the last speed limit encountered and the colour of the next traffic light, if any, while the corrective configuration allows the vehicle to carry out the corrective actions described in the functional requirements (braking, stopping, automatic speed limitation, etc.)

12. integration with steering wheel and pedals (also to facilitate the validation of the system with the laboratory simulator).

Chapter 3

DESIGN OF PROPOSED SOLUTION

3.1 Sensors used

By performing image classification and object detection tasks, the proposed ADAS uses only RGB cameras in order to percept the surrounding environment.

In particular, our ADAS exploits three different cameras, each configured with different parameters to capture images from different perspectives and for different tasks:

- there is a driver camera that shows what the driver sees → this is not used by our ADAS but it is only used because it is convenient when driving the vehicle
- there is a camera dedicated to the speed limit recognition
- there is another RGB camera dedicated to the traffic light recognition
- there is also another RGB camera used to detect if the vehicle has overcome a red traffic light.

The parameters of the RGB cameras used in our project are summarized in the table below:

CAMERA	POSITION	RESOLUTION	FPS	FoV	ROTATION (P, Y)
Driver	(1, 0, 1.5)	Full screen	40	115	(0, 0)
Limit	(1, 0, 1.8)	640X640	5	50	(5, 35)
Trafficlight	(0.3, 1, 1.2)	640X640	4	35	(10, 5)
RedOver	(-1, 0, 2)	640X640	10	120	(-5, 5)

Table 3.1: parameters of the cameras used in the system.

3.2 Cameras positioning

It is possible to visualize the positioning of our cameras (except the one dedicated to the driver) using the picture below:

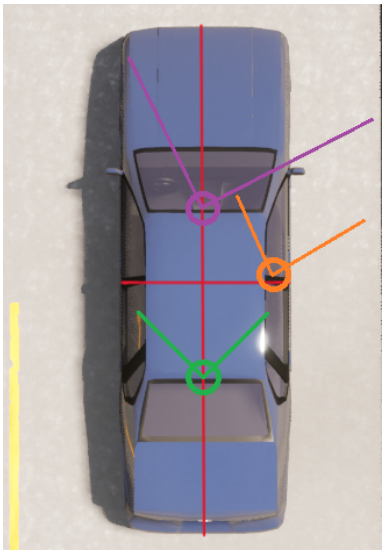

Figure 3.1: sensors positioning on the vehicle.

- Violet circle refers to the camera dedicated to the speed limit recognition
- Orange circle refers to the camera dedicated to the traffic light recognition
- Green circle refers to the camera dedicated to the detection of overcoming a red traffic light.

It is important to notice that even if the standard estabilish that the center of a vehicle is the center of its rear axle, in CARLA simulator the center of a vehicle is instead located in the center of the vehicle.

3.3 Cameras views

In this section there are pictures that show the field of view of each camera:

Figure 3.2: on the right there is the view of the camera dedicated to speed limits (note its limited field of view and its orientation to the right).

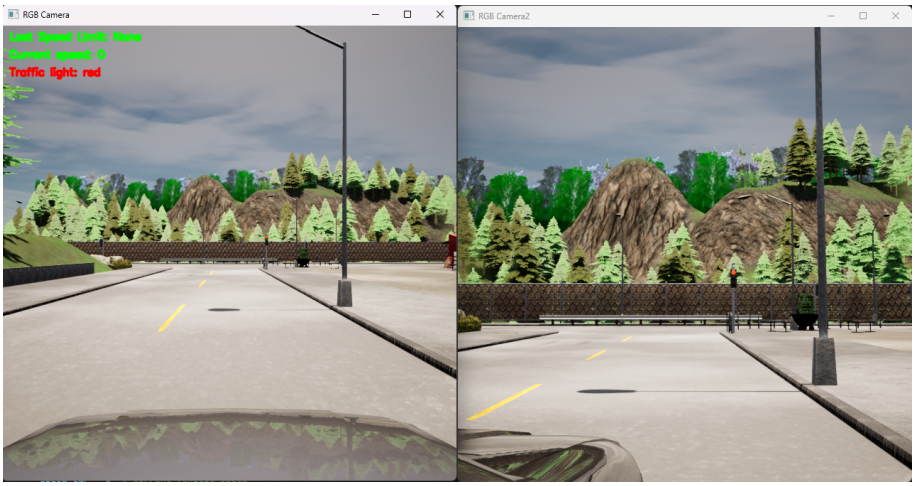

Figure 3.3: on the right there is the view of the camera dedicated to traffic light recognition (note its limited field of view and its orientation to the front-right).

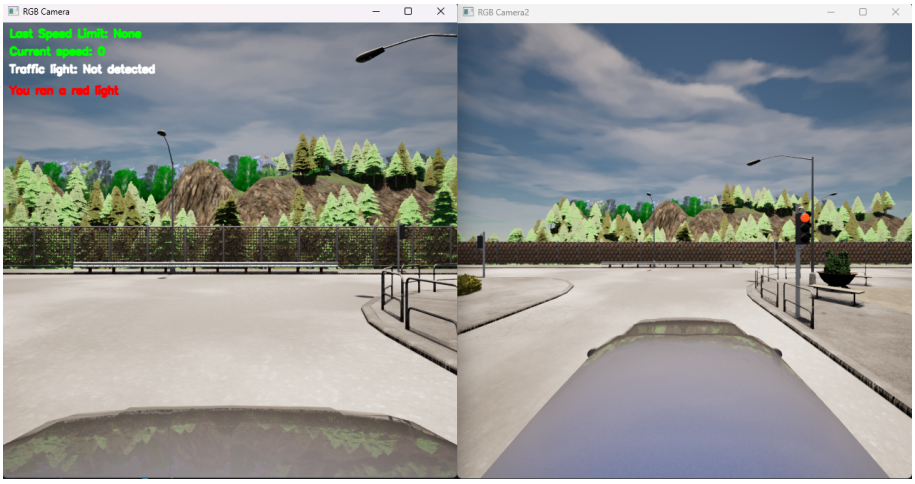

Figure 3.4: on the right there is the view of the camera dedicated to brake if the vehicle surpassed a red traffic light (note its wide field of view and its orientation to the right).

3.3.1 Recognition distances

With the camera parameters set up as specified in the previous section, the ADAS is able to recognize the target at multiple distances:

- if the target is a speed limit, the camera can recognize the sign when the vehicle reaches its proximity
- if the target is a traffic light, the recognition distance basically depends on two main factors, the traffic light color and the environmental brightness →

	GREEN	YELLOW	RED
SUNNY DAYLIGHT	28m	16m	18m
FOGGY, RAINY NIGHT	22m	10m	18m

→ please note that these distances has been measured with the vehicle in a stopped state.

3.4 Sensors purpose

As suggested during the course, in this section are reported the block diagrams of our ADAS, as a visual representation that helps understanding its behavior:

Figure 3.5: block diagram related to speed limit recognition task.

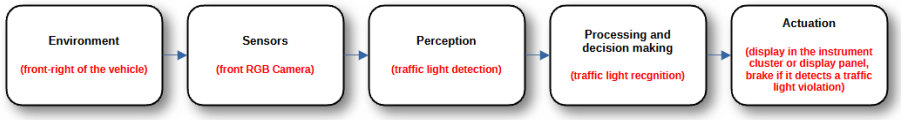

Figure 3.6: block diagram related to traffic light recognition task.

3.5 Computer vision algorithm for speed limit detection

The detection algorithm uses a state-of-the-art YOLO[2] neural network for object detection.
The model was trained in two stages to optimize its performance for detecting speed limit signs in different scenarios.

3.5.1 Training process

- **First training** → the model was initially trained using a specific dataset where speed limit signs were mostly in the foreground (this ensured high accuracy in isolated conditions)
- **Second training** → a more comprehensive and wide dataset, found on Roboflow, was used for further training to enhance the model's ability to detect signs in complex backgrounds.

3.5.2 Dataset details

The dataset used for training the speed limit recognition model was sourced from Roboflow and is publicly available at `https://universe.roboflow.`

com/keehakkee/self-driving-cars-lfjou-dgdu1. In the following table we summirized the dataset details:

number of images	4969
annotation format	YOLOv8
image pre-processing	Resize to 640x640 (Stretch)

Table 3.2: dataset specifications.

3.5.3 Validation results

The following table summarizes the performance metrics of the YOLOv8 model for traffic sign detection.
The metrics include precision, recall and mean Average Precision averaged over IoU thresholds from 50% to 95% (mAP50-95) for all classes and individual traffic sign categories:

CLASS	**IMAGES**	**INSTANCES**	**PRECISION**	**RECALL**	**mAP50-95**
All	801	714	0.959	0.967	0.894
Limit 20	801	56	0.986	0.982	0.881
Limit 30	801	74	0.905	1.000	0.924
Limit 40	801	55	0.951	0.982	0.877
Limit 50	801	71	0.985	0.915	0.870
Limit 60	801	76	0.960	0.949	0.886
Limit 70	801	78	0.962	0.972	0.911
Limit 80	801	56	0.967	1.000	0.875
Limit 90	801	38	0.941	0.895	0.827
Limit 100	801	52	0.954	1.000	0.914
Limit 110	801	17	0.940	0.917	0.901
Limit 120	801	60	0.970	1.000	0.923
Stop	801	81	0.983	0.988	0.933

Table 3.3: validation results obtained by the speed limit signs detection model.

3.5.4 Performance summary

The YOLOv8 model shows strong performance for speed limit sign detection, achieving high precision, recall, and mAP scores:

- among speed limit signs, the **Speed Limit 20** and **Speed Limit 30** classes achieved the highest recall and mAP50-95 values
- The **Speed Limit 90** class shows relatively lower performance, with an mAP50-95 of 0.827.

Results from YOLO training

The training and validation scores can be deepened with:

- **precision-confidence curve** → the precision-confidence curve indicates consistent precision across different confidence thresholds for all classes

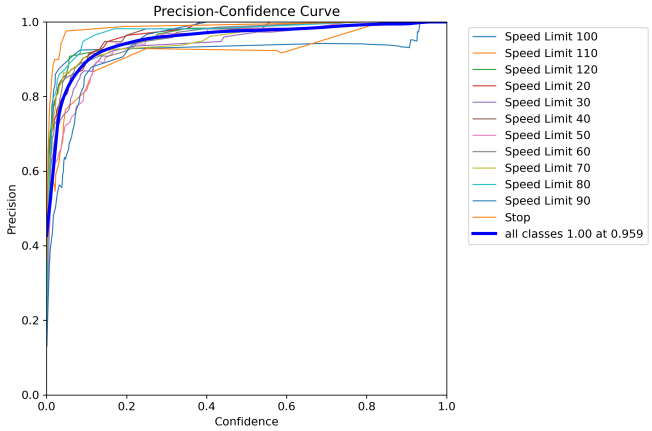

Figure 3.7: precision-confidence curve for YOLO training phase.

- **training and validation metrics** → the loss curves and mAP metrics confirm proper convergence during training

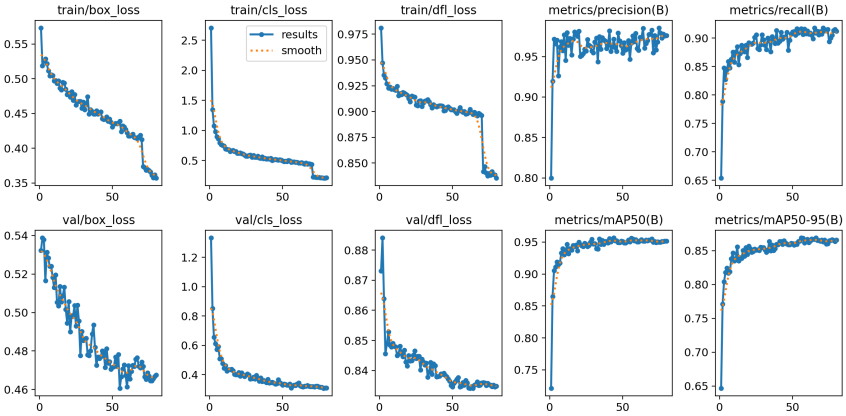

Figure 3.8: training and validation losses and precisions reached.

- **qualitative predictions** → examples of predictions can highlight the model's ability to accurately detect speed limit signs, including in challenging scenarios

Figure 3.9: Sample Predictions on the Validation Dataset

- **confusion matrix** → shows correct and wrong predictions, highlighting the confusion cases

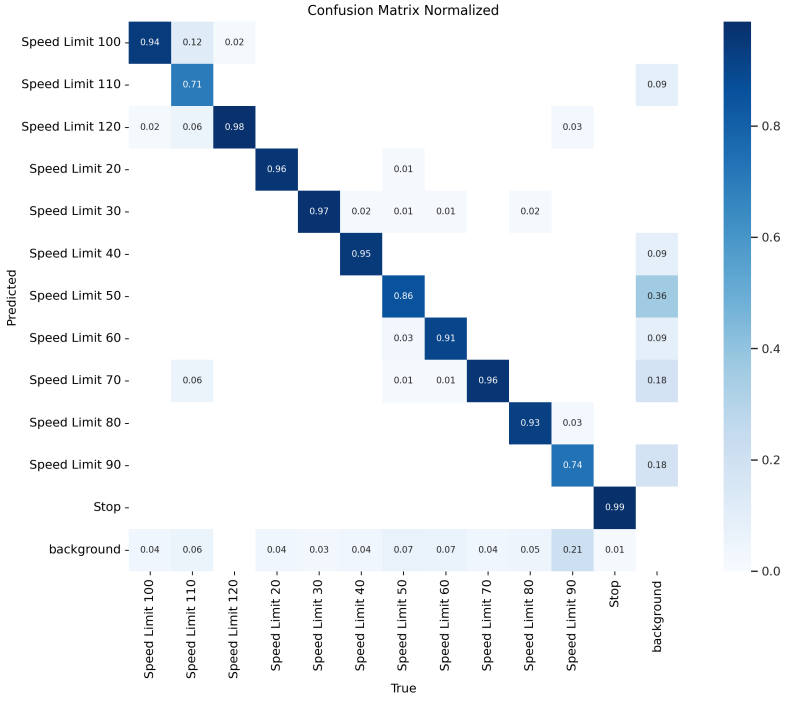

Figure 3.10: confusion matrix.

3.5.5 Speed analysis

The model exhibits efficient processing times for real-time applications (the following time has been computed exploiting a GPU):

STEP	**TIME (ms)**
preprocessing	0.2
inference	1.5
loss computation	0.0
postprocessing	0.8

Table 3.4: time taken for each step.

These results highlight the YOLOv8 model's suitability for the real-time speed limit sign detection task.

3.6 Computer vision algorithm for traffic light detection

This section outlines the computer vision approach employed for traffic light detection tasks in our ADAS system.

Also the detection algorithm utilizes YOLO neural network, specifically optimized for identifying traffic light states in various conditions.

3.6.1 Dataset details

The dataset used for training was found on Roboflow and is available at `https://universe.roboflow.com/iaagl/traffic-lights-rqds4`.

The dataset properties are summarized in the following table:

number of images	20,841
annotation format	YOLOv8
image pre-processing	auto-orientation (EXIF stripping) resize to 640x640 (stretch)
augmentation applied	random gaussian blur (0 to 0.25 pixels)
bounding box transformations	random rotation (-30° to +30°) salt and pepper noise applied to 5% of pixels

Table 3.5: dataset specifications.

3.6.2 Validation results

The following table shows the performance metrics of the trained YOLOv8 model for traffic light recognition during validation.
Metrics include precision, recall and mean Average Precision averaged over IoU thresholds from 50% to 95% (mAP50-95) for all classes and individual traffic light states:

CLASS	IMAGES	INSTANCES	PRECISION	RECALL	mAP50-95
All	1755	2568	0.943	0.945	0.747
Green	1755	1080	0.958	0.944	0.773
Red	1755	1318	0.945	0.943	0.665
Yellow	1755	170	0.926	0.947	0.803

Table 3.6: validation results obtained by traffic light detection model.

3.6.3 Performance summary

The validation of the YOLOv8 model trained for traffic light detection proves high precision and recall for all classes:

- the **green** light class shows balanced performance with a precision of 0.958 and recall of 0.944
- the **yellow** light class achieved the precision of 0.926, and highest recall of 0.947
- the **red** light class shows balanced performance with a precision of 0.945 and recall of 0.943.

Results from YOLO training

The training results of YOLO on the traffic light dataset show significant performance in detecting and classifying traffic lights.

The key results are reported in the following:

- **precision-confidence curve** $\rightarrow$ the model exhibits a high precision for all classes, achieving a maximum precision of 1.00 at a confidence threshold of 0.898

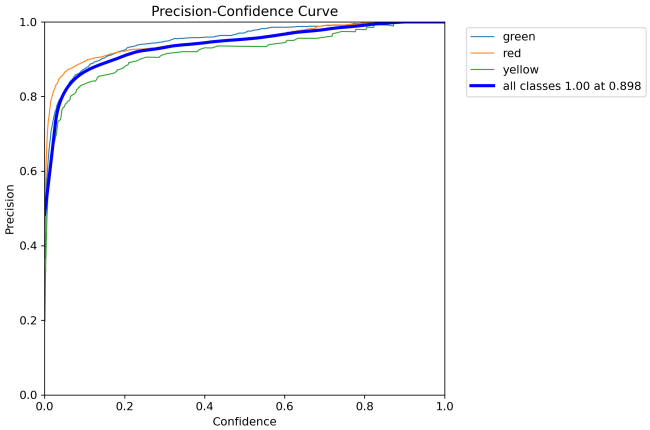

Figure 3.11: precision-confidence curve for YOLO training on traffic light recognition task.

- **training and validation metrics** → the training and validation loss curves indicate proper convergence and additionally the metrics such as mAP@0.5 and mAP@0.5-0.95 exhibit steady improvement, confirming the model's robust training process

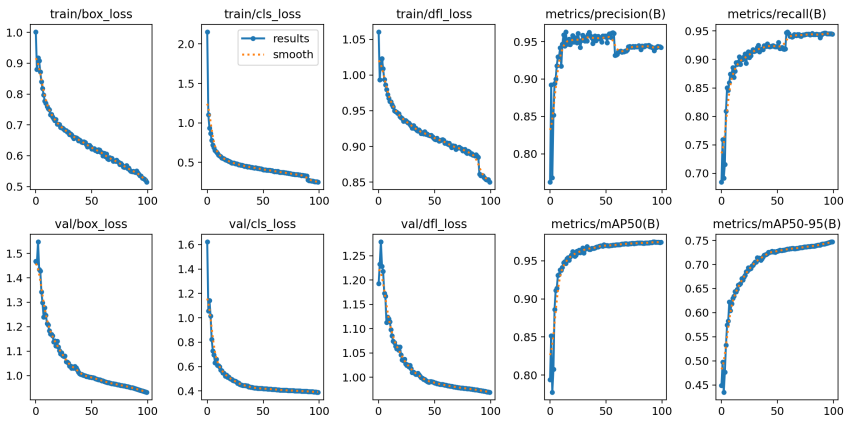

Figure 3.12: training and validation metrics from YOLO ability on traffic light recognition task.

- **qualitative predictions** → the image grid displays successful detection of traffic lights in various real-world scenarios

Figure 3.13: Example of predictions on the validation set.

- **confusion matrix** →

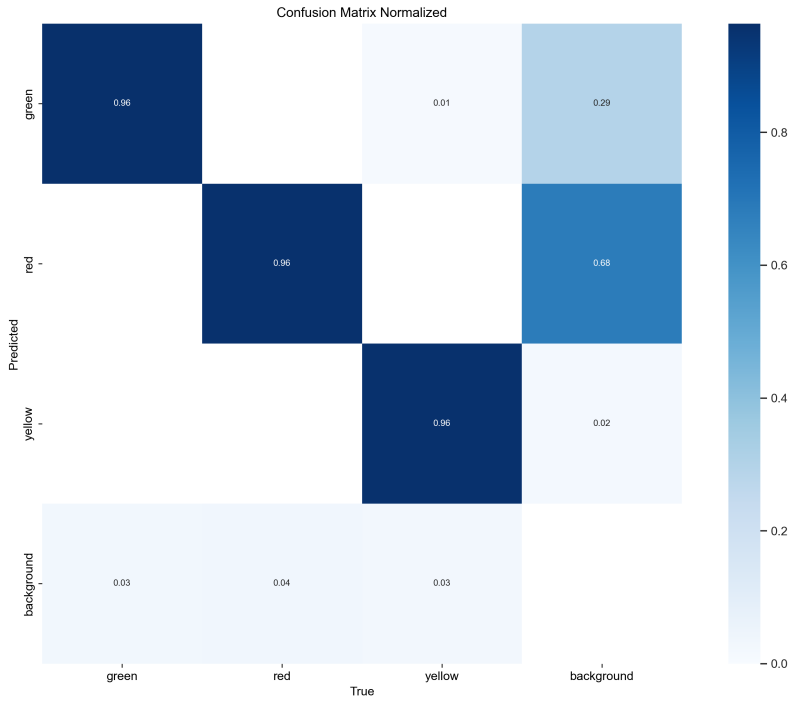

Figure 3.14: confusion matrix

3.6.4 Speed analysis

The model operates efficiently and quickly, with the following processing times with gpu per image:

STEP	TIME (ms)
preprocessing	0.2
inference	1.3
loss computation	0
postprocessing	0.9

Table 3.7: processing times for different steps.

These metrics highlight the model's suitability for real-time applications in traffic light detection.

3.7 Corrective measures

In this section, we discuss the corrective measures implemented to enhance driving safety.

The user can choose to activate/deactivate two different configurations to adapt the car's behavior and make it react to dangerous situations.

3.7.1 Speed limit compliance mode

The first mode allows the user to maintain the speed of the last detected speed limit.

When a speed sign indicating a lower limit is detected, the car brakes proportionally to its current speed.
The braking and acceleration intensity are determined using the following code logic:

```
# Calculate speed error relative to the limit
KP_THROTTLE = 0.15  # Proportional gain for acceleration
KP_BRAKE = 0.02  # Proportional gain for braking
DEAD_ZONE = 3.0  # Dead zone around the speed limit
MIN_THROTTLE = 0.2  # Minimum throttle value
MIN_BRAKE = 0.1  # Minimum brake value
error = float(last_speed_limit) - float(car_speed) + 3
if error < -DEAD_ZONE:
    throttle = 0
    brake = max(KP_BRAKE * abs(error), MIN_BRAKE)
else:
```

```
throttle = max(KP_THROTTLE * error, MIN_THROTTLE)
brake = 0
```

This code adjusts the car's speed to comply with the detected speed limit by computing the difference between the last detected speed limit and the car's current speed, with a small positive bias to allow for slight deviations. If the car's speed exceeds the speed limit by more than a predefined threshold (the dead zone), it disables acceleration (throttle) and applies braking, with the braking force increasing as the speed difference grows.
The braking force is also ensured to have a minimum value to avoid being too weak.
Then, when the car's speed is within the acceptable range or below the speed limit, the system enables acceleration, with the throttle adjusted proportionally to the difference.
A minimum throttle value ensures that the car maintains some level of acceleration when necessary.

3.7.2 Automatic breaking at red light mode

If it has been enabled, the second mode activates when the car passes a red traffic light and in such case:

- the system alerts the user of the violation
- if the car's speed is below 50 km/h, it applies maximum braking and activates the hazard lights.

This behavior is designed to prevent the risk of accidents and collisions in case of unintentional violations at intersections.
Also in this case the braking force depends on the speed of the car when the violation occurs, ensuring a rapid yet safe deceleration process.

3.8 Event publishing in MQTT broker

To enable real-time communication and event monitoring in our system, one of the mandatory requirements was about the use of an MQTT broker.

The code fragment in the next section shows how events are published to the broker when specific traffic-related or road-related conditions occur.

3.8.1 Implementation of MQTT event publishing

The MQTT broker is set up using HiveMQ Cloud, with a secure connection established through TLS.

A client instance is created and its authentication credentials are provided for secure access.
The connection to the broker is established on port 8883, which is the standard port for MQTT over TLS.

The core functionality is implemented in the function `sendEventToBroker(topic, message)`, which takes a topic and a message as parameters: this function attempts to publish the specified message to the given topic on the broker and if any exception occurs during this process, it is silently handled to ensure that the application continues to run smoothly.

Listing 3.1: event publishing to MQTT broker

```
BROKER = "cd027bc56d0e4f84a8cd8c7558775d7b.s1.eu.hivemq.cloud"
PORT = 8883

clientMQTT = mqtt.Client()
clientMQTT.username_pw_set("...", "...")
clientMQTT.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
clientMQTT.connect(BROKER, PORT, 60)

def sendEventToBroker(topic, message):
    try:
        clientMQTT.publish(topic, message)
    except Exception as e:
        pass
```

3.8.2 Event types and publishing logic

Events are published to the MQTT broker under specific conditions based on detections made by the traffic monitoring system.

These events include:

- **speed limit detection event** → when a speed limit sign is detected, an event is published under the topic `"speedLimit"` with a message indicating the detected speed
- **traffic light detection event** → when a traffic light is detected, an event is sent under the topic `"TrafficLight"` with a message identifying the color of traffic light detected
- **traffic light violation event** → if the system detects a red light violation, it publishes an event to the `"TrafficLightViolation"` topic, notifying about the violation.

3.8.3 Event publishing modes

Here are examples of how the system triggers the event publishing functionality for different scenarios:

- **speed limit sign detection →**

  `sendEventToBroker("speedLimit","Detected"+class_name)`

  → this publishes the detected speed limit to the broker, extracted dynamically from the detected class name

- **traffic light detection →**

  `sendEventToBroker("TrafficLight","Detected trafficlight"+class_name)`

  → this sends the color of detected traffic light as a message to the `"TrafficLight"` topic

- **red traffic light violation →**

  `sendEventToBroker("TrafficLightViolation","Red light violation")`

  → this informs the broker about a red light violation detected by the system.

An exemple of received events is shown in the following picture:

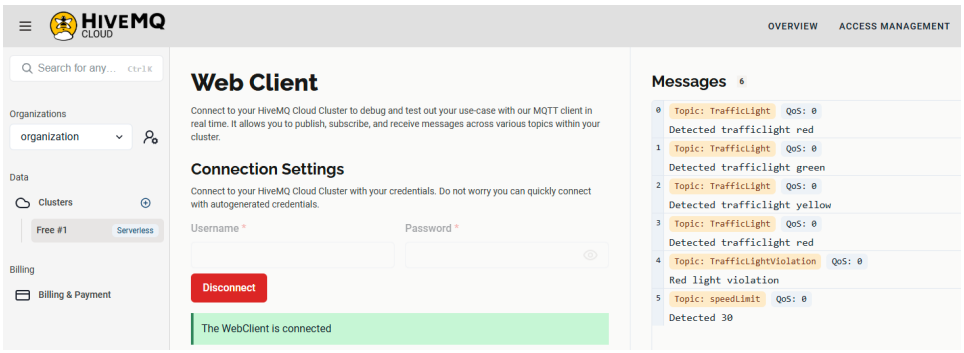

Figure 3.15: HiveMQ client.

3.8.4 Purpose of MQTT in the system

The use of MQTT ensures low-latency and lightweight communication, making it ideal for real-time event publishing in traffic monitoring and control systems.
The organized use of topics allows seamless integration with other components or external systems that subscribe to these topics for processing or visualization of traffic-related events.

Chapter 4

Technologies used

4.1 YOLOv8 and the nano version

In this project, we exploited the YOLOv8 architecture, specifically the nano version (YOLOv8n).
This choice was motivated by the need for rapid predictions and efficient computation, due to the real-time and reactivity requirements required for our system.
YOLOv8n strikes a balance between model complexity and speed, making it ideal for scenarios where computational resources are limited or speed is a critical factor.

4.1.1 YOLOv8n overview

YOLOv8n is a lightweight variant of the YOLOv8 model characterized by its smaller size and faster inference times compared to larger versions, such as YOLOv8s or YOLOv8m.

The key specifications of the YOLOv8n model are summarized in the following table:

input image size	640 pixels
mAPval (50-95)	37.3%
inference speed on CPU	80.4 ms
inference speed on A100 (TensorRT)	0.99 ms
number of parameters	3.2 million
floating point operations (FLOPs)	8.7 billion

Table 4.1: specifications of YOLOv8n model.

YOLOv8n's compact design enables it to achieve significant performance while maintaining a fast processing time, as emerged by its high inference

speed on both CPUs and GPUs.

4.1.2 Pre-trained YOLOv8n on COCO dataset

For this project, the YOLOv8n model was initialized with weights pre-trained on the COCO dataset.

The COCO dataset (Common Objects in Context) is one of the most widely used datasets for object detection tasks: it contains over 200,000 labeled images with 80 different object classes, ranging from everyday items (e.g., chairs, cars) to traffic-related objects (e.g., traffic lights, stop signs).

Using a pre-trained YOLOv8n on COCO provides a significant advantage: the model has already learned general features of objects, such as edges, textures, and shapes, and this enables faster convergence and improved accuracy during training on specific datasets, as the model builds upon the robust feature representations learned from COCO.

COCO eataset overview

The COCO dataset includes:

- over **200,000 labeled images**, annotated with bounding boxes, segmentation masks, and object classes
- **80 object classes**, including both generic objects and traffic-related signs or lights, which are particularly useful for this project
- **contextual diversity** → objects appear in various poses, scales, and contexts, making the dataset highly generalizable.

Benefits of pre-training on COCO

By exploiting YOLOv8n pre-trained on COCO, we achieved:

- **faster training** due to its pre-trained weights
- **improved accuracy** as the pre-trained model already recognizes generic object features
- **Robust performance** → the model performs well even with limited training data.

4.2 Structure of YOLOv8n neural network

YOLOv8n's architecture is designed for efficiency and performance.

It includes a backbone, neck, and head:

- the backbone extracts features from the input image using a series of convolutional layers and exploiting CSP (Cross-Stage Partial) connections to reduce computation and improve feature learning
- the neck aggregates and fuses features from different levels of the backbone using a PANet (Path Aggregation Network) structure, improving localization and detection accuracy
- the head predicts bounding boxes, class probabilities, and confidence scores for the detected objects, operating at multiple scales to handle objects of different sizes.

With 3.2 million parameters and 8.7 billion FLOPs, YOLOv8n is compact yet powerful: it efficiently balances the trade-off between computational complexity and detection performance, making it ideal for real-time applications in traffic monitoring systems.

Chapter 5

TESTING, OBTAINED RESULTS AND PERFORMANCE

In order to test the robustness of our integrated ADAS, we tested its behavior in many different conditions.

Each test has been recorded and it is available in the following link: test recordings.

Each testing session lasted for about 15 minutes and had a different combination of the following parameters:

- town → town01 or town02
- weather → clear (sunny daytime) or adverse (rainy night with extreme fog)
- vehicle speed → fast or slow.

After recording all the sessions we observed the behavior of our ADAS in all the situations emerged (speed limit classification and detection, traffic light classification and detection and automatic brake at red traffic light).

In the following sections we summarized many statistical aspects.

5.1 Global accuracy

During the tests, 442 cases occurred.
In particular, the division for each task is reported in the following table:

TASK	CASES
speed limit classification	137
traffic light classification	209
brake at red traffic light	96

Table 5.1: table reporting the number of cases for each task.

Globally, considering all cases in all sessions together, our ADAS reached the following scores:

METRIC	SCORE
accuracy	0.9321 (412/442)
precision	0.92
recall	0.93

Table 5.2: table reporting global accuracy, global precision and global recall.

5.2 Accuracy based on the type of task

The following table reports the accuracy scores obtained in the different tasks:

TASK	ACCURACY
speed limit classificaion	0.9635
traffic light classification	0.976
brake at red traffic light	0.8316

Table 5.3: table reporting the accuracy scores reached in the different tasks.

As the table shows, our ADAS is highly accurate for what concerns the speed limit and traffic light classification task, but it is less accurate when it has to brake at a red traffic light.

5.3 Accuracy based on environmental parameters

While in the previous section we computed the global accuracy, in the following table we report the accuracy scores differentiated considering the environment parameters:

ENVIRONMENTAL PARAMETER	ACCURACY
town 01	0.9231
town 02	0.9393
clear weather	0.9339
adverse weather	0.9302
slow driving	0.9274
fast driving	0.934

Table 5.4: accuracy scores based on different environmental parameters.

The results show that the ADAS is pretty robust independently from the town, the weather and the driving speed, as it reaches high accuracy independently from these conditions.

In the following table we can see how the accuracy changes combining the environmental parameters:

TOWN / WEATHER	CLEAR	ADVERSE
TOWN 01	0.9293	0.9167
TOWN 02	0.9375	0.9412

Table 5.5: accuracy scores considering combinations of town and weather parameters.

5.4 Error analysis

In this section we briefly analyze the 30 errors (on 442 cases) occurred during the testing phase:

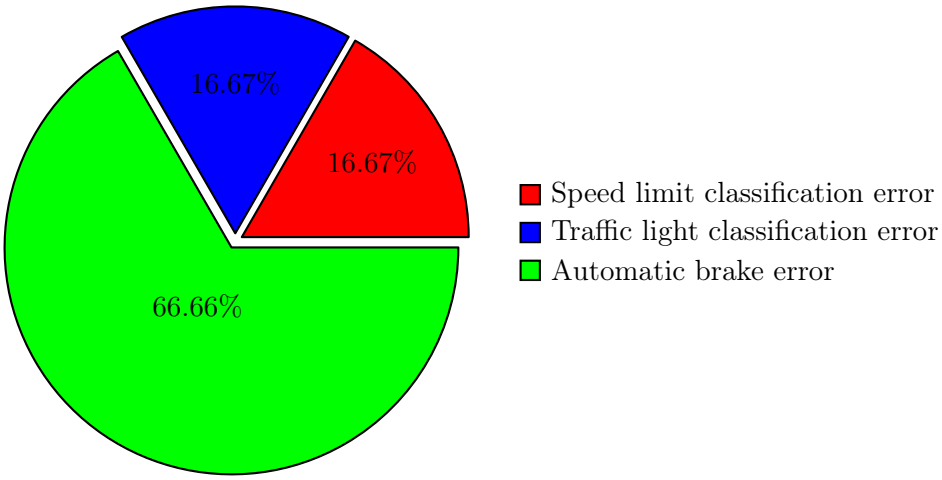

Figure 5.1: percentage of errors related to the different tasks considering the total of errors.

In the following table we explore the diffusion of the errors that can be defined as serious errors: the errors that potentially can cause a damage.

TYPE OF ERROR	% OF TOTAL ERRORS	% OF TOTAL CASES
predicted green, actual class red	0	0
predicted red,actual class green	0	0
predicted go, actual class stop	43.33%	2.44%
predicted stop, actual class go	23.33%	1.08%

Table 5.6: table that shows dangerous errors percentage with respect to the total of errors and the total of predictions done.

As the results show, the serious errors are only about the 3.5% of the total detections and classifications, so our ADAS is sufficiently correct: in fact it reaches a 97% accuracy on speed limit and traffic light classification (note that it has never confused a green and a red traffic light, as the errors reported in the table are only caused by a wrong detection of the position of the red traffic light).

Chapter 6

DEPLOYMENT

Please refer to the Readme file in the GitHub repository, where you can find the requirements and all the instructions to setup the simulator and the virtual environment.

Chapter 7

CONCLUSIONS

7.1 Limitations and strengths of the ADAS prototype

This chapter discusses the limitations and strengths of the developed Advanced Driver Assistance System (ADAS) and highlights potential areas for future improvements.

7.1.1 Limitations

While the ADAS developed in this project has shown promising results, it is important to acknowledge its limitations.

Currently, the system is designed to detect and respond only to vertical traffic lights placed on the right-hand side of the road (this reduces the functionality to scenarios where the vehicle remains strictly in the right-hand lane).
Therefore, the system is not suitable for multi-lane situations or for scenarios where traffic lights are positioned above or on the left side of the road.

7.1.2 Strengths and robustness

Despite its limitations, the ADAS prototype has proven to be highly robust across various scenarios and tests.

The system reached a remarkable performance, achieving a low error rate of just abput 6% in the behaviors tested: this level of reliability indicates that the foundational design of the ADAS is sound and capable of performing effectively in controlled environments.

Some key strengths include:

- **high detection accuracy** → as the previous chapter on performance shows, the proposed ADAS accurately detects and recognizes vertical traffic lights and speed limit signs.
- **responsiveness** → surely the ADAS respects the requirement on reacting promptly to detected signals, ensuring compliance with traffic rules and real-world scenarios.
- **simplicity and efficiency** → the lightweight design of the system ensures fast response times and reduces computational overhead.

7.1.3 Future developments

To further enhance the system and expand its applicability, several areas for future work have been identified:

- **multi-lane support** → extending the ADAS functionality to handle scenarios involving multiple lanes and different traffic light placements
- **dynamic lane management** → develop ing algorithms to take into account for possible lane changes and other dynamic behaviors of the vehicle during driving
- **integration with additional sensors** → combining the system with data from radar (for example to improve automatic brake on red light), LiDAR or GPS to improve decision-making and environmental awareness.

7.2 Closing remarks

While the developed ADAS is clearly a prototype and a simplified version of what a full-scale system would entail, it has laid a strong foundation for future advancements.
The robustness demonstrated in testing, coupled with the low error rate, highlights its potential as a base platform for more complex and feature-rich ADAS solutions.
The insights gained from this project provide a valuable starting point for further innovation in the field of autonomous and assisted driving systems.

Bibliography

[1] *Incidenti stradali / anno 2022 / versione testuale dell'infografica*, ISTAT, 2022.

[2] *You Only Look Once: Unified, Real-Time Object Detection*, Joseph Redmon and Santosh Divvala and Ross Girshick and Ali Farhadi, 2016