Road signs and traffic lights detection

Student Lorenzo Vigorelli
Prof Dr. Pietro Zanuttigh

General Description of the Project

The goal of this project is to perform object detection on street objects, with a particular focus on two categories: road signals (emphasis on speed limits) and traffic lights.
For the road signs, the main objective is to detect and classify them into different types, focusing primarily on speed limit signs. Once detected, the challenge is to extract the speed limit value from the sign using Optical Character Recognition (OCR) methods.
The goal for traffic lights is to detect them and classify them according to their color or shape. This includes detecting traffic lights in various forms (e.g., vertical, horizontal) and classifying them into red, yellow, or green states.

Main Approach

The primary method used to achieve these objectives is the **YOLO (You Only Look Once)** algorithm, which is chosen for its speed and efficiency, as it processes the entire image in a single pass. YOLO is also well-documented and has a variety of open-source implementations, which makes it an attractive option. However, one limitation of YOLO is that it requires fine-tuning on specific datasets to improve its accuracy. This process can be data-intensive.

For the traffic lights and road signs, two distinct YOLO models were developed. The base pre-trained model, which was trained on the COCO dataset, was fine-tuned using two based versions of the YOLO architecture:

- **YOLOv5** with 9M parameters
- **YOLOv9** with 25M parameters

The model with more parameters (YOLOv9) provided better performance but was slower compared to YOLOv5. For the fine tuning the dataset was divided into train, validation and test in order to train the model, validate it by selecting the hyper-parameters and evaluate it by looking at images never seen during the training.

1 Traffic Lights

Two Approaches for Traffic Light Detection:

1. **Fine-tuning on a Specific Traffic Light Dataset:** This approach involved fine-tuning a YOLO model on a custom traffic light dataset. The goal was to detect traffic lights and detect the different shapes (such as vertical or horizontal lights) and colors. Yolov9 and Yolov5 were employed, both obtaining similar results. Leaving the comment of the benchmarks for a later analysis, a few comment on the models:
    - Classifying using only 4 color classes (red, yellow, green, off) performs greatly.
    - Classifying using 10 classes, color (red, green, yellow, off) and the different shapes (straight, left, right) actually gives bad results. But this is due, to the insufficient number of images for the different shapes (2(b)) To have better results a different dataset should be employed.

2. **Base Model Trained on COCO Dataset:** For scenarios where computational resources were limited, the base YOLO model pre-trained on the COCO dataset was used. This model could detect general traffic lights (although less accurate) and later classify the color by extracting bounding boxes and applying a color-based mask to identify the red, yellow, or green light. In this case only the color based classification was performed

Results and Performance

The fine-tuned model provided significantly higher performance in detecting traffic lights. The base model with bounding box, while still impressive, showed a reduction in performance A general problem was in distinguishing between traffic lights with similar visual features (e.g., red and yellow).

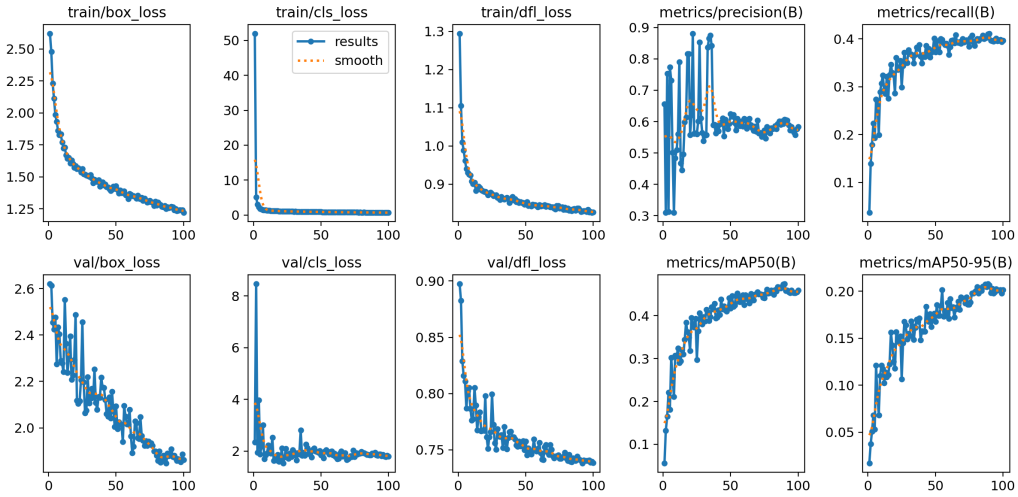
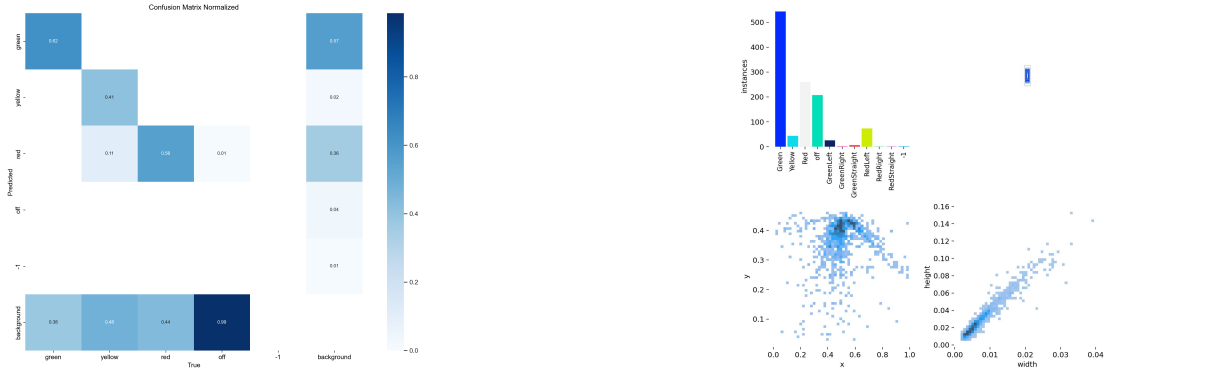


Figure 1: Metrics over epochs



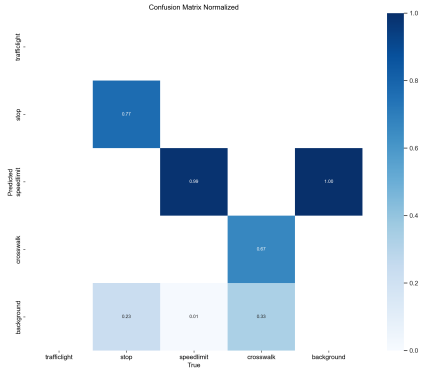
((a)) Confusion matrix on validation set (4 labels). ((b)) Distribution of the labels in training set (10 labels).

Figure 2: Classes

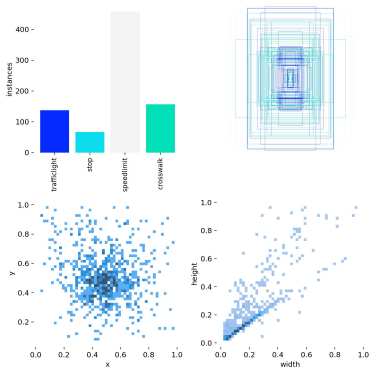
2 Road Signs

Two Approaches for Traffic signals Detection:

1. **Fine-tuning the YOLO model on road signals, especially speed limit signs:** A Kaggle dataset with 800 high-quality samples was used, which focused on only four classes of road signs. This dataset contained well-annotated images but had fewer samples compared to other datasets, which made the fine-tuning process more challenging. The classes of the dataset are 4: traffic lights, speed limit, stop and crosswalk. This model didn't perform quite well in detecting the different street signals, but it does better for speed limits, since they are quite present in the training dataset, respect to the other classes (3(b)).



((a)) Confusion matrix on validation set.



((b)) Distribution of the labels in training set

Figure 3: Classes

The bad general performance are due to the difficulty of finding a training dataset large enough and with data in the same conditions of the testing data.

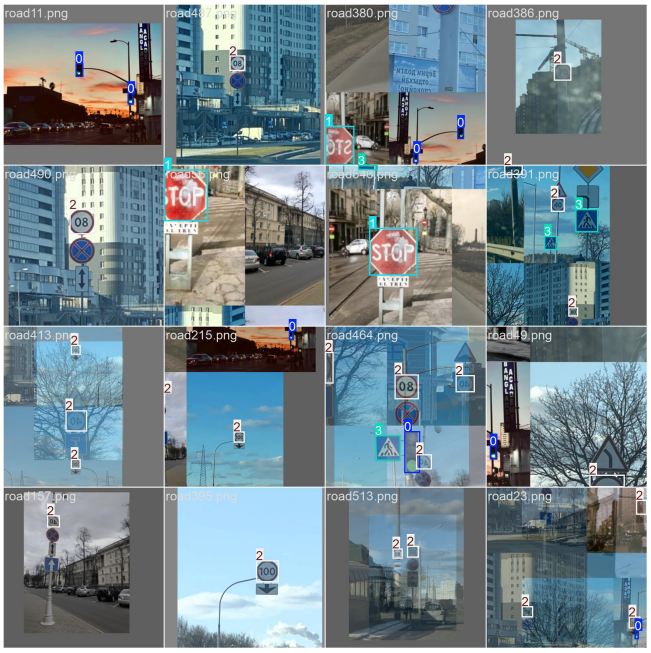


Figure 4: Results on data from the training set

For the detection of speed limits, OCR (Optical Character Recognition) methods were employed to extract the speed value from the speed limit signs. The main difficulty was in accurately detecting and isolating the speed limit numbers from the rest.

2. **Using Circle Hough transform:** It was used a circle hough transform to detect circular signals and speed limits, and as before OCR is used later to read the actual limit.

3 Fine-Tuning Approach and Results

Dataset Formatting for YOLO:

YOLO requires a specific format for dataset annotations, with image files and their corresponding label files stored in the following structure:

```
/dataset
    /images
        /train
        /val
        /test
    /labels
        /train
        /val
        /test
```

Each label file must be in YOLO format, which contains the class ID, center position, width and height of bounding box or each object in the image.

Converting Annotations: For datasets that are not in YOLO format, the annotations must be converted into the appropriate format. This involves calculating the center coordinates, width, and height of each bounding box relative to the image size.

Hyperparameter Tuning:

To fine-tune the YOLO model, different sets of pre-configured hyperparameters were tested. These included configurations optimized for:

- **Night Mode:** For images taken in low-light conditions.
- **Low Number of Data:** A set of parameters designed to handle small datasets with few samples.
- **Regular Mode:** A standard trade-off between performance and generalization.

The hyperparameters mainly impacted the input image's brightness, saturation, and contrast, and included augmentation techniques such as random scaling and rotation.

The best results on our training set were obtained using night mode, or by letting the model automatically selecting the hyper-parameters.

Model Evaluation:

After fine-tuning the model, various evaluation metrics were used to assess its performance, including:

- **Loss** (classification, bounding box, and object loss)
- **Precision and Recall** for each class
- **F1 Score**
- **mAP (mean Average Precision)**
- **Confusion Matrix**

The primary metric for model performance was **mAP**, as it provides a comprehensive measure of detection accuracy across all classes.

4 Conclusion

The project successfully demonstrated the application of YOLO for detecting and classifying traffic lights and road signs, with a particular focus on speed limit signs. Fine-tuning the YOLO model on specific datasets significantly improved its performance, especially in terms of accuracy and speed. However, challenges such as limited data for certain categories and the need for careful annotation and data preparation remained. More classical methods, like hough transform to detect object of certain forms, or analyzing the colors still work, even though deep learning algorithm are more robust and seem to provide better results.

Further improvements could include:

- Fine tuning the model doing a serach on the space of the parameters.
- Expanding the dataset to include more diverse traffic sign types and traffic light forms.
- Testing the model on other real-world street images to evaluate its performance in uncontrolled environments.
- Exploring more advanced techniques for dealing with occlusions and partially visible objects.

In future work, it would be beneficial to further investigate the trade-offs between model size and speed, particularly when deploying the model in real-time systems with limited computational resources.

Appendix

Model Evaluation:

After fine-tuning the model, various evaluation metrics were used to assess its performance, including:

- **Loss** (classification, bounding box, and object loss): *The loss function is used to measure how well the model's predictions match the actual values. Specifically:*
  - **Classification Loss**: Measures the error in classifying the object correctly (e.g., detecting a red traffic light as a red light).
  - **Bounding Box Loss**: Measures the error in predicting the location of the object's bounding box in the image (i.e., the position and size of the detected object).
  - **Objectness Loss**: Measures the confidence of whether a region contains an object or not.
- **Precision and Recall** for each class: *These metrics evaluate the ability of the model to correctly identify objects and avoid false positives and false negatives. Specifically:*

  - **Precision**: The ratio of correctly predicted positive observations to the total predicted positives. It measures how accurate the positive predictions are.

$$Precision = \frac{TP}{TP + FP}$$

    where **TP** is the number of true positives (correctly detected objects) and **FP** is the number of false positives (incorrectly detected objects).
  - **Recall**: The ratio of correctly predicted positive observations to all the actual positives in the dataset. It measures how well the model can identify all relevant objects.

$$Recall = \frac{TP}{TP + FN}$$

    where **FN** is the number of false negatives (missed objects).

- **F1 Score**: *The F1 score is the harmonic mean of precision and recall. It provides a balance between the two, particularly useful when the data is imbalanced (e.g., more background than objects).*

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

- **mAP (mean Average Precision)**: *mAP is one of the most important metrics for object detection. It measures the overall accuracy of the model in detecting objects across multiple classes. It is calculated as the mean of the average precision (AP) for each class, based on the Intersection over Union (IoU) threshold between the predicted and ground truth bounding boxes.*
  - **mAP@0.5**: Measures the average precision with an IoU threshold of 0.5, which is generally considered a good standard for object detection.
  - **mAP@0.5:0.95**: Measures the average precision across multiple IoU thresholds, from 0.5 to 0.95 in increments of 0.05. This provides a more detailed evaluation of the model's performance.
- **Confusion Matrix**: *The confusion matrix is a tool for evaluating the performance of the model by comparing the predicted labels with the true labels. It helps visualize misclassifications and identify areas for improvement. The matrix shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).*
  - **True Positives (TP)**: The number of correctly identified objects.
  - **False Positives (FP)**: The number of incorrectly identified objects.
  - **True Negatives (TN)**: The number of correctly identified non-objects.
  - **False Negatives (FN)**: The number of missed objects.

The primary metric for model performance was **mAP**, as it provides a comprehensive measure of detection accuracy across all classes.