Object Detection in Urban Video Scenes Using Deep Learning Models Based on the YOLO Architecture

Delmário dos Santos Gomes Galvão
Electrical and Computer Engineering
Federal University of Bahia
Salvador-BA, Brazil
https://orcid.org/0000-0002-0738-2473

Luciano dos Santos Gomes
Electrical and Computer Engineering
Federal University of Bahia
Salvador-BA, Brazil
https://orcid.org/0009-0005-1983-7200

André Gustavo Scolari Conceição
Electrical and Computer Engineering
Federal University of Bahia
Salvador-BA, Brazil
https://orcid.org/0000-0001-8124-6253

Eduardo Furtado de Simas Filho
Electrical and Computer Engineering
Federal University of Bahia
Salvador-BA, Brazil
https://orcid.org/0000-0001-8707-785X

***Abstract*—In autonomous navigation systems, real-time environmental perception is essential for safe and efficient operation. Object detection enables agents to identify obstacles, interpret traffic, and adapt routes within very short time intervals. However, dynamic and unstructured environments introduce challenges such as illumination changes, occlusions, and object overlap, which compromise detection accuracy. Addressing these challenges, this paper presents a comparative analysis between deep learning-based models derived from the YOLO architecture, applied to object detection in urban scenarios. A customized dataset derived from COCO was used, including seven relevant classes (person, car, truck, bus, bicycle, dog, and motorcycle). Both models were trained under identical configurations — 50 epochs, 640×640 resolution, and GPU-based processing — and evaluated using real-world video footage from Salvador, Brazil. The results indicate that YOLOv8n achieved higher accuracy, with an Average Precision (AP) of 0.713, compared to 0.699 for YOLOv5n. However, YOLOv5n demonstrated faster inference, operating at 7.81 FPS versus 7.04 FPS for YOLOv8n. These findings highlight the trade-off between accuracy and speed, guiding the selection of appropriate models for computer vision applications.**

***Index Terms*—YOLO, Object Detection, Computer Vision, Deep Learning, COCO Dataset.**

I. Introduction

Object detection is one of the most relevant tasks in computer vision, with direct applications in areas such as surveillance, urban mobility, public safety, autonomous vehicles, and real-time intelligent systems. In this context, deep convolutional models from the YOLO (You Only Look Once) architecture, initially proposed in [3], stand out for their efficient balance between accuracy and inference speed, and are widely used in applications that require fast response and low computational cost.

Although YOLO detectors deliver real-time throughput, sustained accuracy in long, dense urban videos remains difficult. Recent surveys of the YOLO family [1], [2] list occlusion, motion blur and illumination variation as persistent failure sources, especially when object scale changes rapidly across consecutive frames. While Varghese and Sambath [4] report notable gains for YOLOv8 in short aerial clips, the behaviour of lightweight variants over extended, street-level footage has not been fully characterised.

Since its first version, YOLO has evolved significantly. Improvements introduced in YOLOv3 [13], YOLOv5, and more recently in YOLOv8, have brought important advances, such as the use of Concatenate 2 feature maps layers — designed to make the model lighter and faster while maintaining deep feature extraction capabilities — automatic optimizers, and enhanced anchoring mechanisms [19]. YOLOv5, for instance, has become widely adopted in both commercial and academic applications due to its balance between performance and computational efficiency. YOLOv5 has been extensively applied in intelligent transportation systems. In the work [12], an algorithm was developed for the real-time detection of helmets and license plates, with the aim of improving road safety. By combining advanced computer vision techniques with machine learning algorithms, the proposed system was able to effectively detect helmet use and automatically extract license plate information from video frames in real time. With the introduction of YOLOv8, several critical applications have begun to explore its robustness. Recent studies have demonstrated its superior performance in urban aerial image detection tasks. For example, in [11], the architecture was used to detect vehicles in drone-captured images, achieving higher precision and recall metrics than the previous YOLO versions, even under conditions of object overlap and scale variation.

Recent literature highlights the significance of object detection advancements and discusses major challenges such as variability in object scales, occlusions, and illumination changes. The reference [1] provided a comprehensive analysis of these challenges within the YOLO family. Their work identifies crucial improvements across YOLO's architectural successors, detailing strategies that helped mitigate typical issues encountered in real-world scenarios, such as handling scale variations and partial object occlusions. Such insights underscore why YOLO models have become preferred choices for time-sensitive applications, including urban mobility and autonomous systems.

In parallel, alternative object detection approaches, such as Faster R-CNN [7], continue to influence the development of detection models by introducing key innovations. The reference [9] initially proposed R-CNN architectures with hierarchical features to achieve high accuracy in object detection tasks, establishing a performance baseline for subsequent models. Faster R-CNN improved upon these concepts by incorporating Region Proposal Networks (RPNs), significantly speeding up object detection pipelines without sacrificing accuracy. Such two-stage detection models remain influential benchmarks for evaluating YOLO-based architectures.

Meanwhile, EfficientDet [10] and customized YOLO architectures have demonstrated superior performance in specialized scenarios. For example, EfficientDet introduced innovative strategies like the weighted bi-directional feature pyramid network (BiFPN) and compound scaling methods, delivering top-tier accuracy on challenging datasets such as COCO, with significantly reduced computational demands. Additionally, [8] successfully employed an improved YOLO detection network to identify surface defects in industrial steel strips, showcasing YOLO's adaptability to niche industrial applications requiring real-time performance under highly specific conditions.

According to a comprehensive review [2], YOLO models have progressively introduced significant improvements, including anchor-free detection heads, decoupled classification and localization tasks, and modern backbone architectures. These developments have notably improved the models' robustness to object occlusion, motion blur, and varying illumination conditions. Specifically, YOLOv8 has integrated advanced strategies such as dynamic convolution operations, attention mechanisms, and novel data augmentation techniques, significantly enhancing generalization capabilities in diverse and challenging environments.

Recent YOLO variants have also extended their applicability to specialized domains. The reference [5], for example, demonstrated the robustness of an improved YOLOv5 model in precision agriculture, effectively detecting fruit targets in orchards despite significant occlusions and varying lighting conditions. Similarly, YOLOv8 has shown superior performance in aerial object detection scenarios, achieving higher precision and faster convergence rates compared to previous versions, especially when dealing with object overlap and scale variability [4].

Current literature also lacks a fair, side-by-side comparison of the nano versions of YOLO. Diwan *et al.* [1] benchmark YOLOv3–v5 only on still images, leaving temporal consistency untested. Varghese and Sambath [4] evaluate YOLOv8 in isolation, using a training pipeline that differs from earlier YOLO releases. No study to date fixes hardware and hyperparameters while applying a common multi-class urban dataset to measure both detection accuracy and video-level inference speed for YOLOv5n and YOLOv8n. The present work fills this gap.

These advancements highlight the importance of systematically comparing and benchmarking newer YOLO architectures, such as YOLOv5 and YOLOv8, to better understand their strengths and limitations. Evaluating their trade-offs in terms of computational efficiency, detection accuracy, and inference time is essential for selecting suitable models for real-time computer vision applications.

The tight coupling between perception and motion-planning has long been recognized in robotics research. Foundational texts on probabilistic robotics [14] and sampling–based planning algorithms [15] stress that an autonomous agent's decisions are only as reliable as its environment model. In practice, mapping pipelines now leverage rich visual cues: visual–LiDAR odometry such as V-LOAM [16] shows that embedding object detections into the state estimator markedly reduces pose drift over long traversals. A recent SLAM survey further argues that next-generation systems must jointly reason about geometry and semantics - treating detected cars, cyclists, and pedestrians as dynamic and prediction-worthy entities - to achieve the robustness required in dense urban scenes [17].

This work conducts an experimental comparison between YOLOv5n and YOLOv8n for object detection in real urban video scenarios. A filtered subset of the COCO (Common Objects in Context) dataset [6], comprising seven relevant object classes, was used for training. Both models were trained with identical hardware and parameter configurations to ensure a fair and unbiased evaluation. The implementation code for this study is available online[1].

II. Methodology

The methodology was divided into seven main stages, ranging from the selection of classes in the COCO dataset to the comparison of the results obtained using YOLOv5n and YOLOv8n. Fig. 1 presents the overall diagram of the process, including model preparation, training, and analysis.

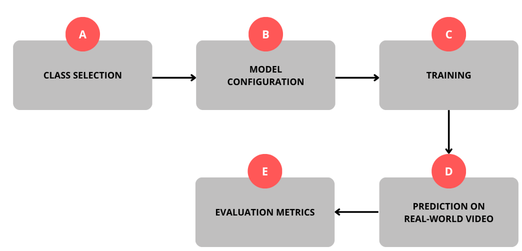


Fig. 1. Main stages of the model development process.

[1] https://github.com/delmario-galvao/yolo-video-detection

A. Class Selection

The selection of object classes used in this study was based on a preliminary analysis of the recorded video. Throughout the route, it was observed that the main obstacles and elements present in the urban scenario included people, vehicles, and animals. The COCO 2017 dataset contains over 118,000 images annotated with more than 80 different classes, covering a wide variety of objects and contexts. However, this extensive volume and diversity can pose a considerable challenge for local training, especially on machines with limited computational resources. Taking into account this limitation, seven classes from the COCO dataset were selected to best represent the elements found in the video: person, car, truck, bus, bicycle, motorcycle, and dog. This selection aimed to ensure greater consistency between the captured scenario and the detected classes, as well as to enable a realistic evaluation of the models' performance in urban traffic and shared-space conditions. Fig. 2 shows the number of instances per class in the filtered subset of the COCO dataset.

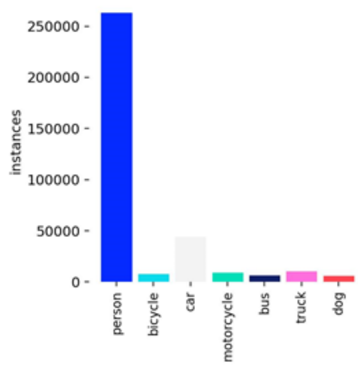


Fig. 2. Number of instances per class in the filtered subset of the COCO dataset.

This procedure significantly reduced the number of images processed during training, decreasing the execution time without compromising the representativeness of the classes in the urban scenario studied. The original format of the COCO dataset annotation files was automatically converted using a Python script to a format compatible with the YOLO training pipeline. This process ensured that the models could be trained directly on the converted data without additional manual adjustments.

B. Model Configuration

To ensure a fair comparison between the YOLOv5n and YOLOv8n models, identical training configurations were adopted for both architectures, except for their internal structures, which are inherent to each model. The specific variants selected for this study were YOLOv5n and YOLOv8n, both belonging to the nano family, which prioritize lightweight operation and inference speed, thus being better suited for training and execution in local environments with limited computational resources. The primary training parameters adopted for both models are detailed in the Tab. I.

TABLE I
TRAINING CONFIGURATIONS USED FOR THE MODELS

Parameter	Value
Base model	yolov5n.pt / yolov8n.pt
Number of epochs	50
Image size	640 $\times$ 640 pixels
Batch size	16
Optimizer	SGD (default)
Device	GPU (NVIDIA RTX 4060)
Number of classes	7

In addition to these parameters, the same data.yaml configuration file was used, containing the paths to the training/validation images and the names of the seven selected classes. Both models underwent training procedures in which performance metrics were evaluated at each epoch, facilitating the selection of the best result. The uniformity of these configurations ensures that any observed performance differences between the models can be attributed with greater confidence to the underlying architecture of each YOLO version, rather than divergent execution parameters. Although other hyperparameter combinations were tested, the configuration in Tab. I yielded the best convergence and accuracy results and was therefore used in the final evaluation.

C. Training

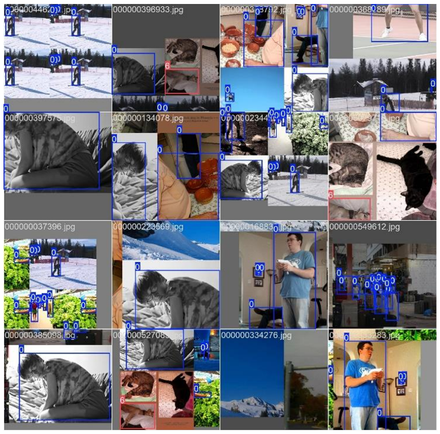
The training of the YOLOv5n and YOLOv8n models was carried out locally on a workstation equipped with an NVIDIA RTX 4060 GPU (8 GB VRAM), running Windows 10, Python 3.13, and PyTorch 2.6.0 with CUDA support. The experiments were conducted separately for each YOLO version, using the previously filtered data converted to the YOLO format.

Fig. 3 shows visual examples obtained during the training of the YOLOv8n and YOLOv5n models, respectively. Each image illustrates how the models learned to recognize the target objects over the epochs, highlighting the seven selected classes using bounding boxes. A wide variety of urban scenarios are observable, with the presence of people, vehicles, and animals reflecting the real world challenges encountered in dynamic environments.

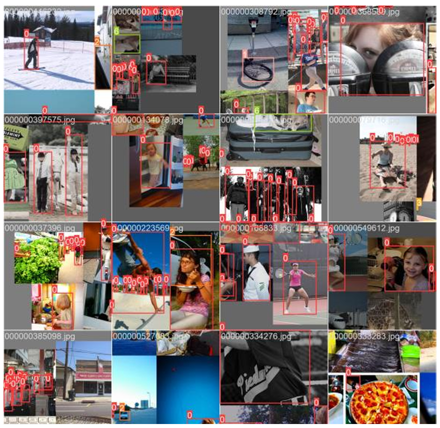
These visualizations offer practical insight into how the models interpret their surroundings and serve as a starting point for understanding the behavioral differences between YOLO versions during the learning process.

To construct the Precision-Confidence Curve, the model's detection results are evaluated across a range of confidence thresholds $c \in [0, 1]$. For each confidence level, only detections with confidence scores greater than or equal to the threshold are considered. As shown in (1), precision measures the proportion of true positive detections among all positive predictions made by the model.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$



(a)



(b)

Fig. 3. Images from the COCO dataset annotated with the selected classes after conversion to the (a) YOLOv8n and (b) YOLOv5n formats.

where, $TP(c)$ is the number of true positives at confidence threshold c and $FP(c)$ is the number of false positives at confidence threshold c.

This calculation is performed iteratively for multiple thresholds (e.g., $c = 0.0, 0.01, \ldots, 1.0$), producing a curve that illustrates how model precision varies with increasing confidence requirements. Separate curves are generated for each object class, as well as for the aggregate detection performance across all classes.

D. Prediction on Real-World Video

After training was completed, each model was used to perform predictions on the same video. This video was selected for its representation of a dynamic urban environment, with frequent occurrences of the selected object classes. The objective of the prediction task was to evaluate the models' performance under real-world conditions, where lighting variation, motion, and occlusion challenge detection algorithms.

During execution, the video frames were processed individually, and a new version of the video was automatically generated containing bounding boxes and visual labels over the detected objects. For both models, the output videos were saved with distinct filenames to facilitate subsequent comparison.

The output results were stored in the default directories: `runs/detect/predict` for YOLOv8n and `runs/detect/exp` for YOLOv5n. The generated video outputs were later used in the result comparison.

E. Evaluation Metrics

The main evaluation metrics considered were:

- **Precision** TP/(TP + FP) – fraction of correct positive detections.
- **Recall** TP/(TP + FN) – fraction of objects correctly detected.
- **mAP@0.5** – mean average precision at IoU = 0.5.
- **mAP@0.5–0.95** – mean AP averaged over IoU 0.50–0.95 (step 0.05).
- **Inference speed** – average frames per second (FPS).
- **Model size** – weight file size in megabytes.

III. Results and Discussion

A. Confusion-Matrix Analysis

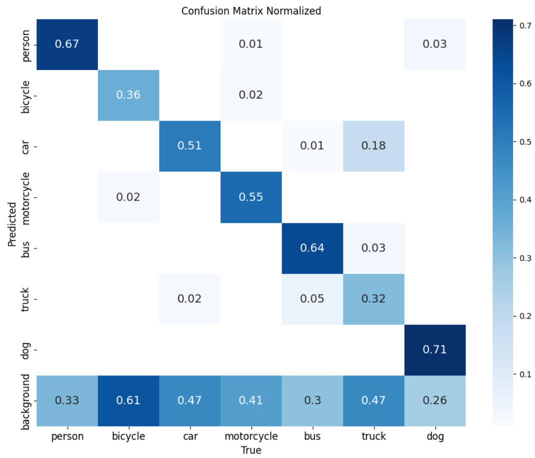
Fig. 4 shows the normalized confusion matrices for YOLOv8n and YOLOv5n. These matrices represent the proportion of correct and incorrect predictions per class, allowing an assessment of each model's performance in urban detection tasks.

According to Fig. 4(a), YOLOv8 performed well across most classes. The "person" class achieved a 67% accuracy rate, while "bus" (64%), "dog" (71%), and "motorcycle" (55%) also showed solid performance. However, some confusion is noticeable between visually similar classes such as "car" and "truck," along with misclassifications as background, particularly for fewer objects like "bicycle" and "bus". Fig. 4(b) shows YOLOv5n's matrix, with overall performance comparable to YOLOv8n. The "person" class again achieved 67%, and similar values were observed for "car" (52%) and "motorcycle" (54%). The main difference lies in slightly higher confusion with the background, especially for "bicycle" and "dog." Nevertheless, YOLOv5n showed consistent predictions and stable behavior across the validation set.

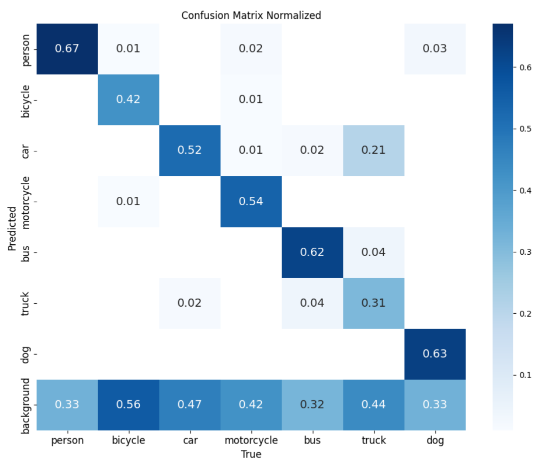
B. Precision–Confidence Curve

Fig. 5 illustrates the precision versus confidence curves for the YOLOv8n and YOLOv5n models. These curves show how the average precision varies as the confidence threshold increases, allowing an evaluation of prediction robustness under different certainty levels.

In Fig. 5(b), corresponding to YOLOv8n, the overall curve is smoother and higher, reaching the maximum precision (1.00) at a higher threshold around 0.986. This indicates that the model maintained high performance even when requiring greater confidence for detection validation. The "person," "car", and "bus" classes stood out, maintaining high precision throughout the entire confidence range, while the "truck" class again showed a more oscillating performance. For YOLOv5n (Fig. 5(a)), precision increases consistently with confidence,

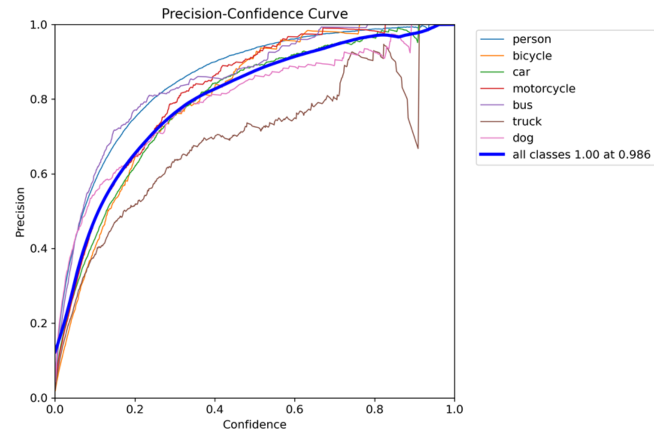


(a)

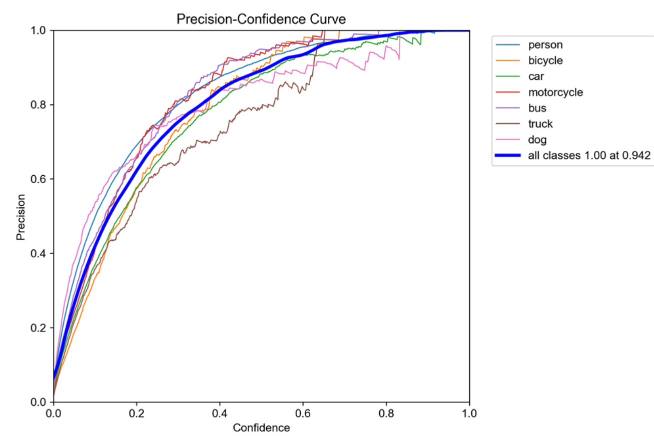


(b)

Fig. 4. Normalized confusion matrices showing per-class accuracy rates (in%) for (a) YOLOv8n and (b) YOLOv5n. These visualizations compare predicted and actual classes on the validation set.



(a)



(b)

Fig. 5. Per-class precision curves as a function of confidence threshold for (a) YOLOv5n and (b) YOLOv8n. These plots demonstrate the stability and behavior of predictions across varying confidence levels.

reaching a peak of 1.00 at approximately 0.942. The "bus" and "motorcycle" classes showed strong performance, whereas the "truck" class displayed less stability at higher thresholds, with a more irregular curve.

C. Training Dynamics

The comparison between the results of YOLOv8n and YOLOv5n (Fig. 6) also highlights performance improvements resulting from architectural evolution. Both models exhibited stable convergence, with a continuous reduction in loss over 50 training epochs.

Although YOLOv5n achieved lower absolute box loss values, YOLOv8n outperformed it in all evaluation metrics, including mAP and recall. This discrepancy is likely due to differences in loss function formulations between the models rather than inferior performance.

The gains observed in YOLOv8n reflect structural enhancements that improve model expressiveness and robustness, especially under complex scenarios, without significantly increasing computational cost.

Despite its slightly lower performance, YOLOv5n remains a robust option, offering shorter training time (see Tab. II) and satisfactory generalization ability. Therefore, the choice between models should consider the trade-off between accuracy and computational efficiency, depending on the specific requirements of the application. In scenarios where inference time is critical, YOLOv5n may be preferable, while in contexts demanding higher detection precision YOLOv8n proves to be more suitable.

TABLE II
TOTAL TRAINING TIME AND AVERAGE TIME PER EPOCH FOR EACH MODEL

Model	**Total Time**	**Time per Epoch**
YOLOv5n	7h15min	∼8–9 minutes
YOLOv8n	8h21min	∼9–10 minutes

The difference in training time is due to structural differences and internal optimizations between the versions, although both models were trained with identical parameters on the same machine.

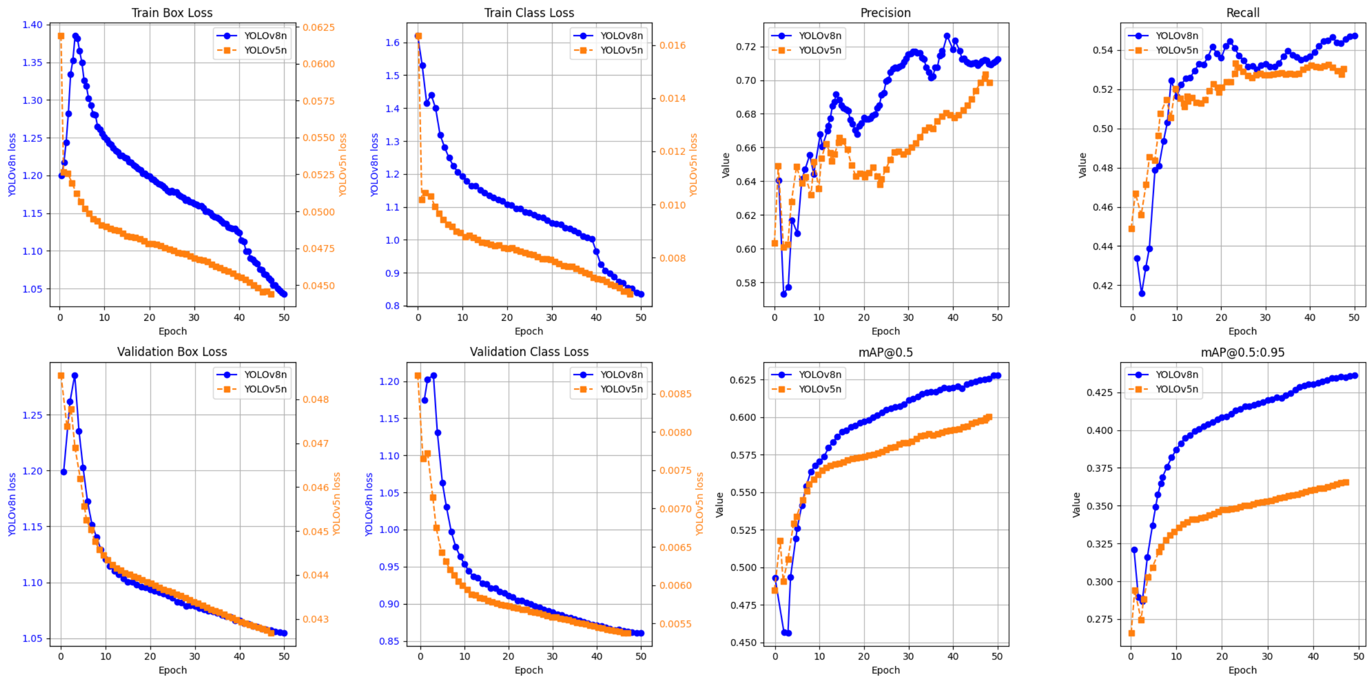


Fig. 6. Training performance of YOLOv8n and YOLOv5n. The graphs present the evolution of training and validation loss, as well as precision, recall, and mAP metrics [18] at different thresholds (mAP50 and mAP50-95).

D. YOLOv8n Results

After 50 epochs of training with the YOLOv8n model, the results were promising regarding detection performance across seven classes, as shown in Tab. III. The total training time was 8.3 hours, and the metrics demonstrated good convergence in loss and evaluation curves:

- **Training and Validation Losses**: The curves `train/box_loss`, `train/cls_loss`, and `train/dfl_loss`, along with their validation counterparts, showed steady decline over epochs, indicating stable learning.
- **Precision and Recall**: The curves `metrics/precision(B)` and `metrics/recall(B)` showed consistent growth, with final values around 0.71 for precision and 0.55 for recall.
- **mAP50**: Reached 0.629, with standout performance in the classes person (0.759), dog (0.739), and bus (0.755).
- **mAP50-95**: Achieved 0.437, which is considered satisfactory for a compact model like YOLOv8n.

TABLE III
PER-CLASS PERFORMANCE OF THE YOLOV8N MODEL

Class	Precision	Recall	mAP50	mAP50-95
person	0.802	0.660	0.759	0.528
bicycle	0.706	0.364	0.455	0.252
car	0.689	0.519	0.585	0.378
motorcycle	0.708	0.544	0.661	0.407
bus	0.810	0.663	0.755	0.618
truck	0.574	0.383	0.451	0.296
dog	0.701	0.702	0.739	0.584

E. YOLOv5n Results

The YOLOv5n model, trained using the same dataset and parameters as YOLOv8n, completed its training in approximately 7.15 hours (Tab. IV).

- **Training and Validation Losses**: The `train/box_loss`, `train/cls_loss`, and `val/box_loss` curves for YOLOv5n remain within a very low range (approximately 0.06 to 0.045), showing a mild downward trend and stable behavior throughout training.
- **Precision and Recall**: The results indicate an average precision of 0.699 and recall of 0.531, slightly lower than YOLOv8n, suggesting reduced generalization in some classes.
- **mAP50**: The final value was 0.600, below the 0.629 achieved by YOLOv8n, possibly due to reduced capacity for capturing complex patterns.
- **mAP50-95**: Reached 0.366, also slightly inferior to YOLOv8n in this more rigorous metric.

Although YOLOv5n presented slightly lower metrics, its ability to perform fast and early detections stands out in autonomous navigation applications where response time is critical. In scenarios where the system must react within milliseconds, the model's agility can outweigh small accuracy losses, especially when integrated with additional sensor data.

TABLE IV
PER-CLASS PERFORMANCE OF THE YOLOV5N MODEL

Class	Precision	Recall	mAP50	mAP50-95
person	0.753	0.655	0.730	0.450
bicycle	0.664	0.481	0.445	0.218
car	0.648	0.518	0.565	0.326
motorcycle	0.736	0.535	0.635	0.326
bus	0.732	0.608	0.720	0.521
truck	0.617	0.354	0.441	0.263
dog	0.744	0.625	0.684	0.459

F. Comparison Between YOLOv5n and YOLOv8n Predictions in Real-World Video

With both models trained under identical conditions and evaluated on the same dataset, a comparative analysis between YOLOv5n and YOLOv8n was conducted, as shown in Tab. V. This comparison considered both quantitative aspects (metrics and timing) and qualitative aspects (videos and visual examples).

TABLE V
OVERALL COMPARISON BETWEEN YOLOV5N AND YOLOV8N MODELS

Metric	YOLOv5n	YOLOv8n
mAP50	0.600	0.629
mAP50-95	0.366	0.437
Average Precision	0.699	0.713
Average Recall	0.531	0.548
Training Time	7h15min	8h21min
Video Inference Time	7.81 FPS	7.04 FPS
Model Size	6.2 MB	6.2 MB

YOLOv8n outperformed YOLOv5n in all evaluated performance metrics, particularly in mAP50-95, which reflects stronger robustness across various overlap thresholds. This advantage can be attributed to structural improvements in YOLOv8n, including a more optimized head and backbone design. The choice between YOLOv5n and YOLOv8n should take into account not only numerical metrics but also the operational context in which navigation takes place. In dense and unpredictable urban environments, YOLOv8n may provide greater robustness in visual interpretation, while YOLOv5n favors faster responses, making it suitable for systems with computational constraints.

Fig. 7 illustrates these detections throughout the 31-minute real-world video used for visual comparison. Both models correctly identified multiple classes relevant to the urban context, highlighting their suitability for embedded computer vision systems applied to autonomous navigation in urban environments.

G. Visual Detection in Video

Both models were tested on the same video, approximately 31 minutes long and containing over 55,000 frames. This material was selected for its representation of a real urban scenario, with a strong presence of the classes used during training. To enable a clearer comparison, a side-by-side video was produced, showing the detections from both models starting from the same initial point. In the visual analysis, YOLOv8n proved more stable in its detections, even under challenging conditions such as intense motion, lighting variations, and occlusions. YOLOv5n, while faster in inference, showed minor fluctuations in more complex scenes. However, it compensated with the ability to anticipate distant or partially visible objects, an asset in applications requiring rapid responses. Fig. 8 illustrates these differences using frames extracted from the same moment in the video, processed by both models. Despite starting at the same time, a temporal mismatch is noticeable in detection progression, reflecting the models' different prediction times. In some cases, YOLOv5n even detected background vehicles earlier than YOLOv8n, highlighting its visual reach in certain contexts. A side-by-side comparison video of the YOLOv5n and YOLOv8n models is available online[2]. This video provides valuable insights that complement numerical metrics, offering a more comprehensive evaluation of detection consistency and contextual awareness. By analyzing frame-by-frame differences, it is possible to identify subtle behaviors in model predictions, such as delayed recognition, false positives, or stable object tracking, that are not always captured through aggregate statistics alone.

(a)

(b)

Fig. 7. Detections performed by (a) YOLOv8n and (b) YOLOv5n during inference on real-world video.

[2]https://www.youtube.com/watch?v=KH6e24C4HDs

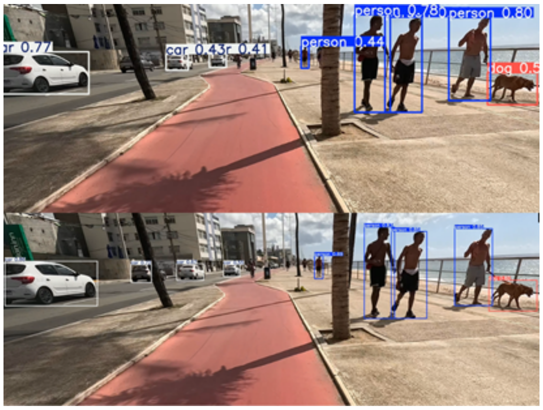


Fig. 8. Comparison between the results obtained with YOLOv8n (top, time 16:12) and YOLOv5n (bottom, time 15:40).

IV. Conclusions

This work conducted a head-to-head comparison of YOLOv5n and YOLOv8n for object detection in urban-video scenarios. Both models were trained on the same hardware, with an identical subset of the COCO dataset and uniform hyper-parameters, ensuring a fair baseline for evaluation.

Quantitatively, YOLOv8n outperformed YOLOv5n on the main accuracy indicators — mAP50 and mAP50-95 — while exhibiting greater stability under challenging conditions such as occlusion, rapid motion, and lighting variation. Conversely, YOLOv5n delivered higher inference speed, making it advantageous where stringent real-time or low-power constraints apply.

Qualitative side-by-side video analysis reinforced these patterns: YOLOv8n maintained consistent detections across dense scenes, whereas YOLOv5n was quicker to flag distant or partially visible objects, although with more frequent misses in complex frames.

From the perspective of autonomous navigation, these findings are particularly relevant. The ability to detect pedestrians, vehicles, bicycles, and animals with both precision and speed is essential to ensure safety and real-time decision — making by mobile agents. YOLOv8n is more suitable for scenarios requiring robustness and continuous stability, whereas YOLOv5n may be preferred for embedded platforms that demand quick response and low resource consumption.

As future work, the use of pre-trained models on the full COCO dataset is planned, applying transfer learning to focus specifically on the target classes. Additionally, a hyperparameter optimization process will be carried out for each architecture to enhance model performance in real urban scenarios.

V. Acknowledgment

The authors acknowledge the financial support provided by CNPq-Brazil (grant numbers [201532/2024-7, 442706/2023-5 e 407163/2022-0]), FNDCT-Brazil, MCTI-Brazil, and CAPES-Brazil.

References

[1] Diwan, T., Anirudh, G. & Tembhurne, J.V. Object detection using YOLO: challenges, architectural successors, datasets and applications. Multimed Tools Appl 82, 9243–9275 (2023). https://doi-org.ez10.periodicos.capes.gov.br/10.1007/s11042-022-13644-y.

[2] Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach. Learn. Knowl. Extr. 2023, 5, 1680-1716. https://doi.org/10.3390/make5040083

[3] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[4] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ADICS58448.2024.10533619.

[5] Yan, B.; Fan, P.; Lei, X.; Liu, Z.; Yang, F. A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOv5. Remote Sens. 2021, 13, 1619. https://doi.org/10.3390/rs13091619

[6] Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). "Microsoft COCO: Common Objects in Context." In European Conference on Computer Vision (ECCV). DOI: https://doi.org/10.1007/978-3-319-10602-1_48.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[8] Jiangyun Li, Zhenfeng Su, Jiahui Geng, Yixin Yin, Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network, IFAC-PapersOnLine, Volume 51, Issue 21, 2018, Pages 76-81, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2018.09.412.

[9] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

[10] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 10778-10787, doi: 10.1109/CVPR42600.2020.01079.

[11] R. Guerra, J. Macedo, & F. Carvalho, "YOLOv8 Deep Learning Model for Vehicle Detection in Aerial Images," Mathematics, vol. 12, no. 7, p. 1103, 2024. DOI: https://doi.org/10.3390/math12071103.

[12] H. Chen, W. Zhang, & C. Wu, "YOLOv5-based real-time helmet and license plate detection algorithm for intelligent transportation systems," Journal of Physics: Conference Series, vol. 2753, no. 1, p. 012084, 2024. DOI: https://doi.org/10.1088/2051-672X/ad601c.

[13] Redmon, J., & Farhadi, A. (2018). "YOLOv3: An Incremental Improvement." arXiv preprint, arXiv:1804.02767.

[14] Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*. Cambridge: MIT Press, 2005.

[15] LaValle, S. M. *Planning Algorithms*. Cambridge University Press, 2006. Available at: http://planning.cs.uiuc.edu/. Accessed: May 14, 2025.

[16] Zhang, J.; Singh, S. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015. DOI: https://doi.org/10.1109/ICRA.2015.7139807.

[17] Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. DOI: https://doi.org/10.1109/TRO.2016.2624754.

[18] Padilla, R.; Passos, W. L.; Dias, T. L.; Netto, S. L.; da Silva, E. A. B. A comparative analysis of object detection metrics with a companion open-source toolkit. *arXiv preprint*, arXiv:2006.11275, 2021. Available at: https://arxiv.org/abs/2006.11275. Accessed: May 14, 2025.

[19] Ultralytics. (2023). "YOLOv8." GitHub Repository. Disponível em: https://github.com/ultralytics/ultralytics.