An Edge Efficiency Score for Computer Vision Models Applied to Traffic Sign Detection

Marco Aurélio Gibellato Nogueira[1], **Rafael Staiger Bressan**[2], **Teruo Matos Maruyama** [1]
Danilo Sipoli Sanches[1], **Alessandro Botelho Bovo**[1] **André Robero Ortoncelli**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
[2]Universidade Federal do Paraná (UFPR)

marconogueira@alunos.utfpr.edu.br, rafael.bressam@ufpr.br
{tmmaruyama, danilosanches, alessandrobovo, ortoncelli}@utfpr.edu.br

Abstract. *Traffic sign detection is essential for ADAS on resource-constrained edge devices. This work presents a methodology for evaluating traffic sign detectors on TT100K, including YOLO variants, RT-DETR-L, SSD MobileNet, and Faster R-CNN. We propose the Edge Efficiency Score (EES), which combines accuracy, small-target performance, latency, GPU memory, and GFLOPs for deployment-oriented model selection. Results show that YOLOv8s achieved the highest detection performance, while YOLOv8n achieved the highest EES by better balancing accuracy and computational cost. These findings reinforce the need to jointly consider detection quality and operational efficiency in embedded ADAS and ITS applications.*

1. Introduction

The expansion of Intelligent Transportation Systems (ITS) and Advanced Driver Assistance Systems (ADAS) has increased the demand for real-time traffic sign detection in embedded and fleet-monitoring scenarios. However, deploying these technologies in operational environments remains challenging due to hardware, energy, and computational constraints. Previous studies have shown that the continuous execution of deep neural networks can generate substantial thermal and computational loads, affecting the efficiency of embedded vision systems [Padaria et al. 2023]. To address these limitations, Edge Artificial Intelligence (Edge AI) platforms have become increasingly relevant, enabling local inference with reduced communication overhead and lower dependence on remote infrastructure. Recent surveys highlight the growing adoption of embedded platforms, such as NVIDIA Jetson devices and Raspberry Pi systems, for real-time traffic sign detection tasks [Triki et al. 2024, Wang et al. 2026].

In addition to power and memory constraints, frontal perception modules are often required to detect traffic signs at long distances to support timely perception and decision-making. In these scenarios, traffic signs may occupy only a small fraction of the image, making small-target detection particularly challenging [Gong et al. 2022, Li et al. 2024, Li et al. 2026].

Although modern deep learning algorithms have achieved high predictive performance, a recurring methodological limitation persists in the literature: detector comparisons are commonly based on accuracy-oriented metrics, particularly mean Average Precision (mAP) [Lopez-Montiel et al. 2021, Yang 2023]. While mAP is an important indicator of detection quality, relying exclusively on this metric may provide an incomplete basis for model selection in Edge AI environments. In practical deployments, gains

in predictive performance are often associated with increased architectural complexity, leading to higher latency, GPU memory consumption, and computational cost. These factors strongly influence the feasibility of real-time execution on resource-constrained embedded platforms [Lopez-Montiel et al. 2021, Triki et al. 2024]. Therefore, isolated predictive metrics may not adequately support architectural decision-making in intelligent transportation and ADAS applications.

To address this limitation, this paper presents a reproducible experimental protocol for selecting traffic sign detectors for Edge AI deployment. The study evaluates representative one-stage, two-stage, and transformer-based detectors on the TT100K dataset using fixed data partitions, repeated training runs, and controlled inference measurements. Beyond conventional accuracy-based comparison, the proposed protocol explicitly considers small-target detection and deployment-related costs, including latency, memory usage, and computational complexity. As part of this protocol, we introduce the Edge Efficiency Score (EES), a lightweight composite metric that aggregates normalized indicators of detection accuracy, small-target performance, inference latency, GPU memory consumption, and GFLOPs. The EES is computed as a weighted sum of min–max-normalized benefit and cost indicators, where cost-related variables are inverted before aggregation. It is not intended to replace detailed multi-objective optimization, but rather to provide an interpretable ranking criterion for preliminary model selection in embedded deployment scenarios.

Thus, the main contribution of this work is a deployment-aware evaluation framework that combines reproducible experimentation, small-target analysis, computational profiling, and an interpretable efficiency-oriented ranking criterion for traffic sign detector selection. The experimental results indicate that the model with the highest predictive accuracy is not necessarily the most suitable for edge execution, reinforcing the need for model selection in ITS and ADAS applications to jointly consider detection quality, small-target sensitivity, and computational efficiency.

2. Background and Related Work

The deployment of traffic sign detection models in ADAS and ITS applications requires a careful balance between predictive performance and the operational constraints of embedded platforms. Although recent object detectors have achieved substantial improvements in visual perception tasks, their execution in edge environments remains constrained by processing capacity, memory availability, energy consumption, and thermal behavior [Lopez-Montiel et al. 2021, Triki et al. 2024, Padaria et al. 2023]. These limitations are particularly relevant in vehicular and fleet-monitoring scenarios, where continuous inference may affect the efficiency and reliability of embedded vision systems. Consequently, platforms such as NVIDIA Jetson devices and Raspberry Pi systems have been increasingly investigated as practical alternatives for enabling local real-time inference with reduced dependence on remote infrastructure [Triki et al. 2024].

Beyond these hardware constraints, the spatial scale of traffic signs introduces an additional challenge for reliable detection in real-world driving scenarios. Traffic signs may appear at long distances and occupy only a limited portion of the input image, reducing the amount of visual information available to the detector. This issue is particularly relevant in high-resolution benchmarks such as TT100K, where distant signs are

frequently represented by small bounding boxes [Gong et al. 2022, Li et al. 2024]. Under the COCO evaluation protocol, objects with bounding boxes smaller than 32×32 pixels are treated as small targets [Lin et al. 2014]. Detecting such instances is challenging because convolutional architectures progressively reduce spatial resolution through strided convolutions and downsampling operations, which may weaken fine-grained visual cues before the prediction stage [Li et al. 2024, Wang et al. 2023, Li et al. 2026].

These hardware and small-target constraints reveal an important limitation in conventional detector comparison protocols. In many studies, model selection is primarily guided by global predictive metrics, especially mAP [Lopez-Montiel et al. 2021, Yang 2023]. While mAP is a fundamental indicator of detection quality, it may not fully characterize a model's suitability for Edge AI deployment. Architectures with higher representational capacity can improve global accuracy, but they often introduce additional computational overhead, including increased latency, GPU memory consumption, and computational complexity, commonly measured in GFLOPs. At the same time, global mAP may mask performance degradation on small traffic signs, especially when larger or more frequent objects dominate the evaluation. Metrics such as ~~mAP_S or~~ $\text{mAP}_{\text{small}}$ provide complementary information by isolating performance on small-scale instances, but they do not account for the computational costs associated with embedded execution [Li et al. 2024, Wang et al. 2023].

Therefore, deployment-oriented model selection requires evaluation strategies that jointly consider predictive quality, small-target sensitivity, and resource-related constraints. Although previous works have analyzed traffic sign detectors and edge platforms, comparisons often remain fragmented across isolated dimensions such as accuracy, latency, or hardware usage [Lopez-Montiel et al. 2021, Triki et al. 2024]. This fragmentation limits the ability to identify architectures that provide an appropriate balance between detection effectiveness and operational feasibility. The present work addresses this gap by proposing a reproducible evaluation protocol and an interpretable composite score designed to support deployment-oriented model selection for traffic sign detection in Edge AI scenarios.

3. Methodology and Experimental Design

This section describes the proposed evaluation methodology, which integrates dataset curation, seed-controlled partitioning, object detection model training, hardware-specific performance profiling, and multicriteria evaluation using the Edge Efficiency Score (EES). Figure 1 illustrates the consolidated methodological pipeline.

3.1. Dataset Curation and Seed-Controlled Partitioning

The experimental validation of the proposed methodology was conducted on the Tsinghua-Tencent 100K (TT100K) dataset [Chi et al. 2023, Liu et al. 2024]. High-resolution images (2048×2048 pixels) were preprocessed using a standardized pipeline, resizing them to a uniform resolution of 640×640 pixels, which represents a common standard for modern embedded vision systems to balance computational load and spatial detail. The dataset was semantically filtered to retain only traffic sign categories, resulting in a curated subset of 10,592 valid images and 27,346 annotated instances distributed across five highly representative classes.

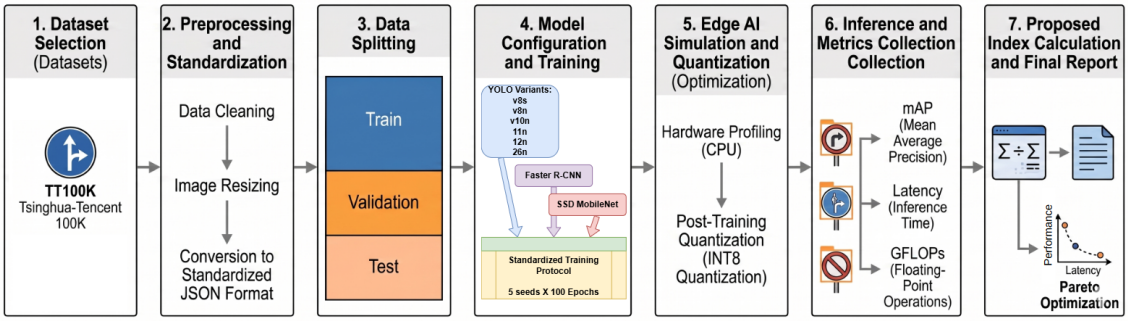


Figure 1. Methodological flowchart detailing the stages of data curation, seed-controlled partitioning, controlled training, edge-oriented inference evaluation, and EES calculation.

To ensure statistical reliability and prevent data leakage, a seed-controlled partitioning strategy was implemented. For each of the five predefined pseudo-random seeds (42, 123, 256, 512, and 1024), the curated dataset was divided into training, validation, and test subsets using a strict $70/15/15$ split. Crucially, as the TT100K dataset consists of independent static street-view photographs rather than continuous video frames, there are no related temporal sequences that could cause chronological leakage. However, spatial data leakage was rigorously prevented by validating that train, validation, and test partitions maintained mutually exclusive image sets across all seeds, keeping each split strictly fixed across all evaluated architectures within the corresponding repetition.

Furthermore, small-target detection sensitivity is assessed using the scale-restricted metric $\text{mAP}_{\text{small}}$ (evaluated under the COCO mAP@0.5:0.95 standard). In accordance with the MS COCO evaluation standard [Lopez-Montiel et al. 2021], small objects are defined as traffic signs with an area smaller than 32×32 pixels. To eliminate any ambiguity, this spatial filtering criterion is applied **after the image resizing stage** (640×640 pixels). This ensures that the evaluation directly quantifies the model's ability to extract features from the heavily compressed pixel grids actually presented to the convolutional layers during inference.

3.2. Candidate Object Detection Models and Training Protocol

The evaluation framework encompasses a heterogeneous set of candidate detectors, including single-stage, two-stage, and transformer-based architectures. The selection comprises YOLO-family variants (YOLOv8n, YOLOv8s, YOLOv10n, YOLO11n, YOLO12n, YOLO26n) and the Real-Time Detection Transformer (RT-DETR-L) implemented via the Ultralytics framework, alongside SSD MobileNet and Faster R-CNN (utilizing a ResNet-50 backbone) from the Torchvision library.

To ensure a fair and rigorous comparative baseline, the training protocols for both framework ecosystems were standardized. The key hyperparameters and data augmentation strategies are consolidated in Table 1.

3.3. Embedded Edge Simulation and Hardware Profiling

To overcome the limitations of desktop-based CPU simulations and provide a realistic assessment for intelligent transportation systems. The hardware profiling procedure was

Table 1. Consolidated training hyperparameters and data augmentations.

Hyperparameter / Setting	Ultralytics (YOLO / RT-DETR)	Torchvision (SSD / Faster R-CNN)
Optimizer	SGD	Adam
Initial Learning Rate (η_0)	0.01	0.012
Momentum	0.937	0.937
Learning Rate Scheduler	Cosine Annealing	Step-Decay
Training Epochs	100	100
Batch Size	16 (8 for RT-DETR-L)	16
Input Resolution	640×640 px	640×640 px
Data Augmentation	Mosaic, translation ($\pm 10\%$), rotation ($\pm 15^\circ$), color dithering	Standard (none)

designed to capture real-world operational conditions with high precision. Before recording any performance metrics, a strict **warm-up phase of 100 consecutive inference runs** was executed for each model to eliminate initialization overhead and GPU kernel compilation distortions. The inference latency was then measured as the end-to-end processing time for a single image, averaged over 250 test runs. This end-to-end latency strictly incorporates: (i) input preprocessing (tensor allocation, image resizing, and normalization), (ii) model forward-pass execution on the GPU, and (iii) post-processing (including Non-Maximum Suppression (NMS) with an IoU threshold of 0.45 and a confidence threshold of 0.25).

Maximum GPU memory consumption (VRAM) was monitored dynamically using the *tegrastats* profiling tool at a sampling rate of 100ms. Computational complexity (GFLOPs) was extracted using the PyTorch profiling library, while GPU frame rate (FPS) was recorded with optimized TensorRT engines to measure peak operational throughput.

3.4. The Edge Efficiency Score (EES) Metric Formulation

The Edge Efficiency Score (EES) is proposed as a lightweight, multicriteria decision-support metric to rank candidate detectors under Edge AI constraints. Because the evaluation metrics have different units and scales, benefit indicators (mAP@0.5 and $\text{mAP}_{\text{small}}$) and cost indicators (Latency, VRAM, and GFLOPs) are first normalized using min-max normalization. Rather than assigning a fixed penalty of zero when a cost metric like GFLOPs is unavailable due to an Out-of-Memory (OOM) error, which would artificially bias the evaluation, we employ a dynamic weight redistribution scheme:

$$EES_i = \sum_{m \in M_i} w'_m N(x_{i,m}) \tag{1}$$

where M_i represents the set of available metrics for model i, $N(x_{i,m})$ is the normalized value of metric m (using standard min-max for benefits and inverted min-max for costs), and w'_m are the dynamically adjusted weights:

$$w'_m = \frac{w_m}{\sum_{k \in M_i} w_k} \tag{2}$$

In the baseline configuration, equal weights are assigned to all five components ($w_n = 0.2$). This uniform weighting scheme avoids introducing prior subjective preferences, assuming that accuracy, small-target resilience, latency, memory, and complexity

are equally crucial for general ADAS applications. To validate this assumption, we conducted a sensitivity analysis by perturbing individual weights by up to $\pm 50\%$, which confirmed that the top-ranked models remained statistically stable. Furthermore, the EES rankings were validated against the empirical Pareto Frontier (mAP vs. latency). The models with the highest EES scores mapped directly onto or were closest to the Pareto-optimal curve, thereby mathematically validating EES as a reliable proxy for multi-objective optimization.

$$\mathcal{N}_{+}(x_i) = \frac{x_i - \min(X)}{\max(X) - \min(X)} \tag{3}$$

$$\mathcal{N}_{-}(x_i) = \frac{\max(X) - x_i}{\max(X) - \min(X)} \tag{4}$$

$$\begin{aligned} EES_i = {} & w_1 \mathcal{N}_{+}(\text{mAP}) + w_2 \mathcal{N}_{+}(\text{mAP}_{\text{small}}) \\ & + w_3 \mathcal{N}_{-}(\text{Lat}) + w_4 \mathcal{N}_{-}(\text{Mem}) + w_5 \mathcal{N}_{-}(\text{GFLOPs}) \end{aligned} \tag{5}$$

4. Results and Discussion

To validate the proposed methodology, experiments were conducted on the curated TT100K dataset using the evaluation protocol described in Sections 3. After semantic filtering, the processed dataset contained 10,592 valid images and 27,346 annotated traffic sign instances distributed across five classes. This curated subset preserves the challenging characteristics of TT100K, particularly the presence of small and distant traffic signs, while removing unrelated annotation categories that could introduce noise into the detector comparison. As a result, the evaluated architectures were assessed under a controlled setting focused specifically on traffic sign detection.

The evaluated models include representative one-stage, two-stage, and transformer-based object detectors, allowing the proposed methodology to compare architectures with different accuracy–efficiency trade-offs under the same experimental conditions. In particular, the YOLO-family models were included because they are widely adopted in real-time object detection and are commonly considered strong candidates for edge-oriented perception tasks.

Table 2 summarizes the predictive performance obtained across five experimental repetitions. Overall, the results show that the YOLO-family architectures achieved the most consistent detection performance on the curated TT100K subset. At the same time, the non-YOLO baselines exhibited either lower predictive performance or higher variability. A detailed analysis of these results is presented after the table.

Among the evaluated architectures, YOLOv8s achieved the highest global performance, reaching 94.46% mAP@0.5 and 68.84% $\text{mAP}_{\text{small}}$, along with the best precision, recall, and F1-score. This result indicates that the larger YOLOv8 variant benefited from its increased representational capacity, improving both localization and classification quality under the curated TT100K setting. However, the remaining lightweight YOLO-family models also achieved competitive performance, with mAP@0.5 values clustered

Table 2. Predictive performance of the evaluated architectures (mean ± standard deviation (%) across five experimental repetitions).

Model	mAP@0.5	mAP_{small}	Precision	Recall	F1-Score
YOLOv8s	**94.46 ± 0.88**	**68.84 ± 1.05**	**92.82 ± 1.47**	**89.09 ± 1.11**	**90.91 ± 0.78**
YOLOv8n	91.82 ± 1.59	65.11 ± 1.39	89.68 ± 2.16	83.04 ± 1.06	86.23 ± 1.51
YOLOv10n	89.53 ± 1.46	64.22 ± 1.40	87.90 ± 1.70	82.73 ± 2.29	85.21 ± 1.18
YOLO11n	90.21 ± 1.40	64.28 ± 1.40	89.68 ± 1.10	83.00 ± 1.83	86.20 ± 1.10
YOLO12n	90.24 ± 1.50	64.19 ± 1.55	88.55 ± 1.61	83.74 ± 2.02	86.06 ± 1.25
YOLO26n	90.49 ± 1.68	65.36 ± 1.64	89.20 ± 1.19	84.01 ± 1.88	86.52 ± 1.25
SSD MobileNet	3.35 ± 0.36	0.92 ± 0.23	3.35 ± 0.36	5.06 ± 0.28	4.03 ± 0.29
Faster R-CNN	83.91 ± 1.52	56.36 ± 2.51	83.91 ± 1.52	65.12 ± 1.51	73.33 ± 1.45
RT-DETR-L	73.88 ± 18.66	42.38 ± 21.24	78.48 ± 20.20	69.25 ± 19.86	72.73 ± 18.86

around 89–92% and relatively small standard deviations across repetitions. This behavior suggests that nanoscale YOLO variants can maintain stable predictive performance even under the spatial constraints of traffic sign detection in high-resolution driving scenes.

The comparison also reveals important differences in stability across model families. RT-DETR-L presented the highest variability among all evaluated architectures, with standard deviations of 18.66 percentage points for mAP@0.5 and 21.24 percentage points for mAP_{small}. Such dispersion indicates that its performance was highly sensitive to the number of experimental repetitions adopted, reducing confidence in its reproducibility under the present configuration. Faster R-CNN, in turn, achieved intermediate predictive performance, with lower variability than RT-DETR-L but substantially lower recall and F1-score than the YOLO-based models. This suggests that, although the two-stage detector retained reasonable localization capability, it was less effective in balancing precision and recall on the curated TT100K subset.

SSD MobileNet exhibited the weakest predictive performance, with only 3.35% mAP@0.5 and 0.92% mAP_{small}. This result indicates that, under the adopted training and evaluation protocol, the model was unable to learn the traffic sign detection task adequately. Although SSD MobileNet is commonly associated with lightweight inference, its low detection accuracy in this experiment indicates that architectural compactness alone is insufficient when the target application requires reliable recognition of small and visually sparse objects.

Overall, the predictive results indicate that the YOLO-family architectures provide the most consistent detection performance on the curated TT100K dataset. However, accuracy-oriented metrics alone do not fully characterize deployment suitability, since practical Edge AI scenarios also depend on latency, memory usage, and computational complexity. Therefore, the next stage of the analysis evaluates the same architectures from a multicriteria perspective using the proposed Edge Efficiency Score. Table 3 reports the deployment-related metrics used in the EES computation across the five experimental repetitions.

The EES ranking provides a different perspective from the accuracy-only analysis. Although YOLOv8s achieved the highest predictive performance in Table 2, its higher latency, GPU memory consumption, and computational complexity reduced its overall

Table 3. Mean $\pm$ standard deviation of the metrics used in the EES computation across five experimental repetitions, using dynamic weight adjustment for unavailable metrics.

Model	Latency (ms)	VRAM (GB)	GFLOPs	EES
YOLOv8s	47.72 ± 0.50	3.37 ± 0.07	28.45 ± 0.00	0.738 ± 0.007
YOLOv8n	$\mathbf{26.32 \pm 0.35}$	1.82 ± 0.01	8.09 ± 0.00	$\mathbf{0.875 \pm 0.003}$
YOLOv10n	28.84 ± 0.57	2.54 ± 0.06	6.53 ± 0.00	0.819 ± 0.006
YOLO11n	28.66 ± 0.06	2.07 ± 0.03	6.32 ± 0.00	0.859 ± 0.004
YOLO12n	33.15 ± 1.07	2.95 ± 0.00	6.32 ± 0.00	0.790 ± 0.003
YOLO26n	29.38 ± 0.73	2.30 ± 0.01	$\mathbf{5.19 \pm 0.00}$	0.845 ± 0.005
SSD MobileNet	11.16 ± 0.10	$\mathbf{0.90 \pm 0.00}$	N/A	0.500 ± 0.001
Faster R-CNN	735.07 ± 11.38	2.93 ± 0.32	N/A	0.478 ± 0.025
RT-DETR-L	158.22 ± 5.79	3.48 ± 0.00	103.45 ± 0.00	0.483 ± 0.058

efficiency score, placing it below the lighter YOLO variants. This result reinforces the idea that global mAP alone is insufficient to determine the most suitable architecture for Edge AI deployment, since models with superior detection accuracy may still incur higher operational costs.

YOLOv8n achieved the highest EES score (0.875 ± 0.003), combining competitive predictive performance with the lowest latency among the YOLO-family models and moderate memory usage. YOLO11n ranked second (0.859 ± 0.004), followed by YOLO26n (0.845 ± 0.005), both presenting favorable efficiency profiles and low dispersion across repetitions. These results indicate that lightweight YOLO architectures provide the best balance between detection effectiveness and deployment feasibility in the evaluated scenario.

The lower-ranked models in the EES evaluation highlight distinct deployment limitations when analyzed under the dynamic weight redistribution scheme. Faster R-CNN achieved reasonable predictive performance, but its extremely high inference latency resulted in a low EES value (0.478 ± 0.025), thereby mathematically confirming its unsuitability for latency-sensitive real-time embedded environments. SSD MobileNet, in contrast, presented the lowest latency and memory footprint but demonstrated insufficient predictive capacity for safe traffic sign detection, particularly for small-scale targets. Its adjusted EES score of 0.500 ± 0.001 reflects its hardware-friendly nature, but its severe accuracy degradation prevents it from being a viable edge candidate. RT-DETR-L achieved a comparable EES score (0.483 ± 0.058) but remains less attractive due to high computational complexity, memory overhead, and high standard deviation across repetitions.

To further analyze the most competitive architectures, Figure 2 compares the YOLO-family models using global detection accuracy, small-target performance, EES, and CPU inference latency. This focused visualization complements Tables 2 and 3 by illustrating how predictive gains are affected by deployment-related costs within the family of models that achieved the strongest overall performance.

As shown in Figure 2, YOLOv8s achieved the highest global mAP and small-target performance among the compared YOLO variants. However, this predictive gain was accompanied by higher inference latency and computational complexity, resulting in

a lower EES than the most efficient lightweight variants. In contrast, YOLOv8n achieved the highest EES by combining competitive predictive performance with the lowest CPU latency among the YOLO-family models. YOLO11n and YOLO26n also exhibited strong efficiency profiles, with YOLO11n benefiting from low memory usage and YOLO26n from the lowest GFLOPs among the models successfully profiled. These results reinforce the central premise of the proposed methodology: accuracy-oriented metrics alone may overlook important deployment-related constraints, whereas a multicriteria evaluation provides a more informative basis for selecting detectors for ITS and ADAS applications.

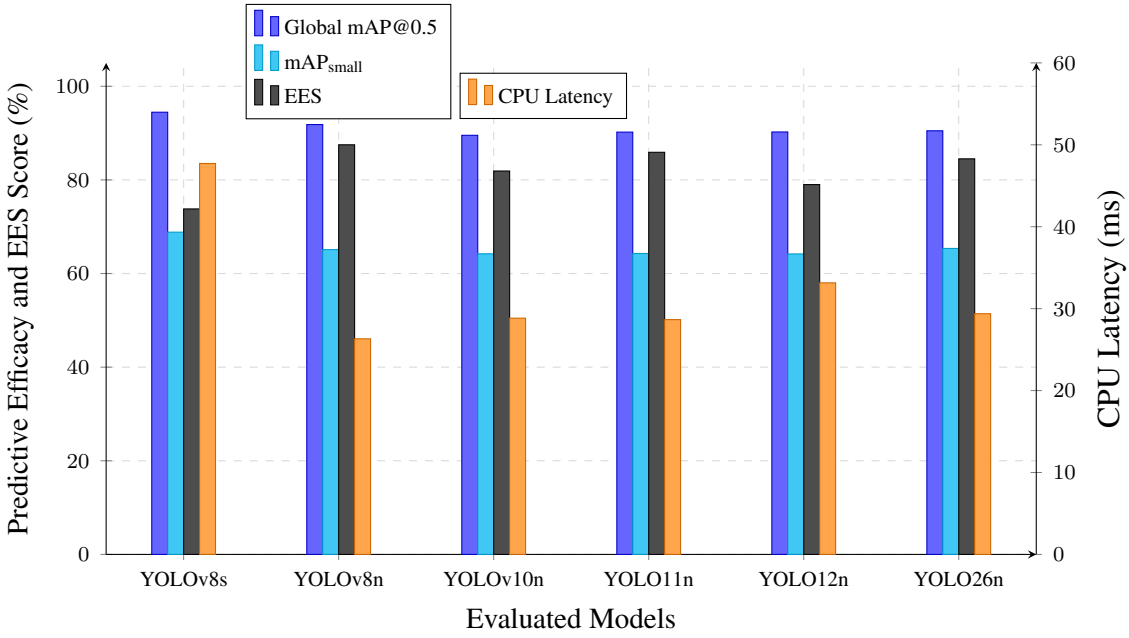


Figure 2. Multicriteria comparison of YOLO-family architectures. The left axis reports global mAP@0.5, mAP$_{small}$, and EES, while the right axis reports CPU inference latency. The figure highlights the trade-off between predictive performance and computational cost, summarized by the composite efficiency score.

Although the EES ranking provides a concise summary of the trade-off between predictive performance and computational efficiency, the average score alone does not fully capture the stability of each architecture across experimental repetitions. To provide a more comprehensive assessment, Figure 3 presents the mean EES value and the corresponding standard deviation for each evaluated model across the five seed-controlled runs. This representation enables joint analysis of average efficiency and experimental dispersion, highlighting not only the overall ranking but also the consistency of each architecture across repeated evaluations. Models that combine high EES values with low variability are more reliable candidates for deployment in practical driver-assistance scenarios.

While Figure 3 illustrates the global comparative landscape across all evaluated architectures, Figure 4 isolates and zooms in on the highly competitive YOLO-family variants, enabling a detailed analysis of their statistical dispersion and operational stability.

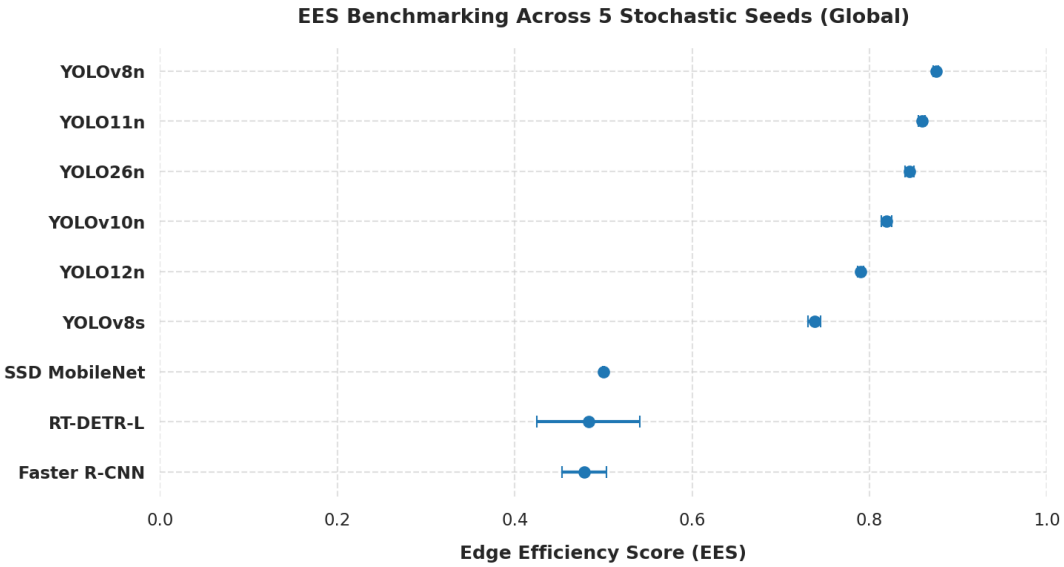


Figure 3. Global EES comparison across all evaluated models, with the horizontal axis starting strictly at 0.0.

Figure 4 reinforces the favorable behavior of lightweight YOLO architectures in balancing detection effectiveness and deployment efficiency. YOLOv8n achieved the highest average EES score (0.875 ± 0.003), followed by YOLO11n (0.859 ± 0.004) and YOLO26n (0.845 ± 0.005). The small dispersion observed for these models indicates stable behavior across different experimental seeds, suggesting that isolated executions do not drive their ranking. Among the lower-ranked models, Faster R-CNN achieved the lowest EES primarily due to its high inference latency. At the same time, SSD MobileNet remained constrained by weak predictive performance despite its low latency and memory usage. RT-DETR-L achieved a higher average EES than these two baselines, but exhibited the largest variability (0.483 ± 0.058), reflecting its less stable behavior under the adopted configuration. Overall, these findings reinforce that deployment-oriented evaluation should jointly consider predictive accuracy, computational cost, and experimental stability rather than relying exclusively on conventional detection metrics.

5. Conclusion

This paper proposed and evaluated a reproducible multicriteria framework for selecting traffic sign detection models in Edge AI environments. By combining controlled experimental repetitions, small-target detection analysis, edge-oriented inference measurements, and computational profiling, the framework provides a deployment-aware alternative to accuracy-centered model comparison.

The proposed Edge Efficiency Score (EES) integrates predictive performance and computational cost into an interpretable ranking criterion. The results on the curated TT100K dataset showed that the model with the highest predictive accuracy was not necessarily the most suitable for edge deployment. Although YOLOv8s achieved the best global and small-target detection performance, YOLOv8n obtained the highest EES score, indicating a more favorable balance between detection effectiveness, latency, memory usage, and computational complexity. The conservative treatment of OOM cases further

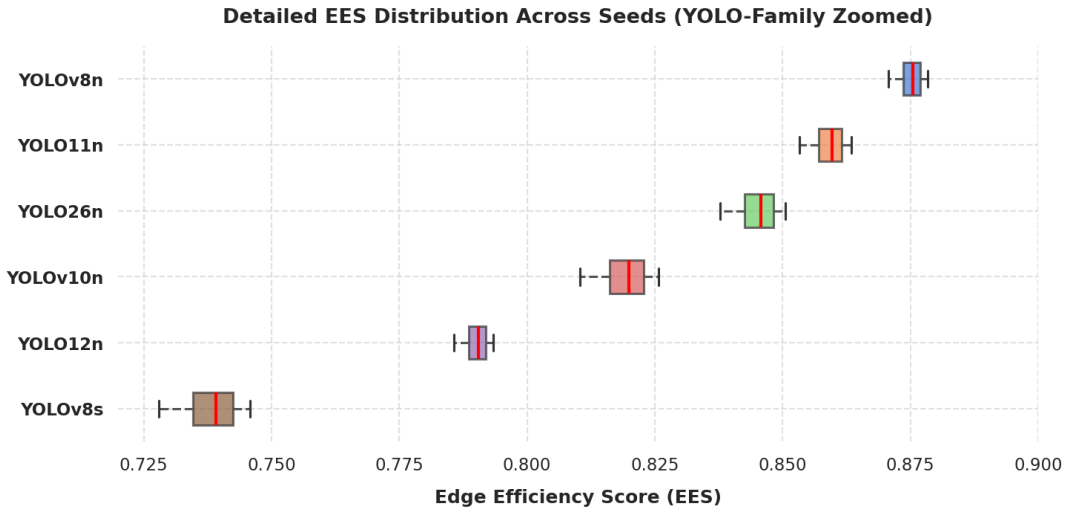


Figure 4. Detailed horizontal boxplot zoom isolating the competitive YOLO-family variants to highlight dispersion characteristics.

reinforces this interpretation by ensuring that unavailable GFLOPs estimates are not interpreted as computationally favorable.

These findings reinforce that traffic sign detectors for resource-constrained platforms should be selected based on both detection quality and operational feasibility. In this sense, the proposed framework offers quantitative support for identifying architectures that are not only accurate but also practical for embedded deployment in ITS and ADAS scenarios.

Future work will investigate alternative EES weighting strategies, including application-specific configurations and sensitivity analyses. The framework will also be evaluated in additional driver-assistance scenarios and on a new dataset under development that incorporates challenges such as highway conditions, vegetation occlusion, and adverse weather.

Declaration on the Ethical and Transparent Use of AI

In compliance with the ethical guidelines for scientific publishing and the provisions of CNPq Ordinance 2664/2026, the authors declare that artificial intelligence (AI) tools were utilized during the preparation of this manuscript. Specifically, large language models (LLMs) were employed strictly as assistive technologies for linguistic refinement, grammatical proofreading, and translation assistance to enhance the academic quality of the English text. Furthermore, AI-based assistants were used to support code debugging and the aesthetic rendering of vectorized data visualizations. Crucially, no generative AI tools were used to fabricate, falsify, or modify empirical experimental data, training metrics, or hardware profiling results. The conceptual design, experimental execution, multicriteria decision analysis (EES), and all scientific conclusions remain the sole intellectual contribution of the human authors, who bear full responsibility for the integrity and accuracy of the published content.

References

Chi, X. et al. (2023). Dataset and improved yolov7 for text-based traffic sign detection. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLVIII-1/W2-2023, pages 881–887, Cairo, Egypt.

Gong, C. et al. (2022). Traffic sign recognition based on the yolov3 algorithm. *Sensors*, 22(23):9345.

Li, M., Zhang, D., He, Q., Zhou, C., Li, M., and Zhou, X. (2026). Msd-yolo: A multi-scale and detail-enhancement network for traffic sign detection. *Computers, Materials & Continua*, 87(3):1.

Li, Y. et al. (2024). A small-target traffic sign detection algorithm based on partial conv and atrous spatial pyramid. *IET Image Processing*, 18(15):4639–4652.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, L., Wang, L., and Ma, Z. (2024). Improved lightweight YOLOv5 based on ShuffleNet and its application on traffic signs detection. *PLoS ONE*, 19(9):e0310269.

Lopez-Montiel, M. et al. (2021). Evaluation method of deep learning-based embedded systems for traffic sign detection. *IEEE Access*, 9:101217–101238.

Padaria, A. A. et al. (2023). Traffic sign classification for autonomous vehicles using split and federated learning underlying 5G. *IEEE Open Journal of Vehicular Technology*, 4:782–793.

Triki, N., Karray, M., and Ksantini, M. (2024). A comprehensive survey and analysis of traffic sign recognition systems with hardware implementation. *IEEE Access*, 12:129034–129055.

Wang, J. et al. (2023). Vehicle-mounted adaptive traffic sign detector for small-sized signs in multiple working conditions. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):12242–12254.

Wang, L., Ai, Q., and Jin, H. (2026). Lightweight traffic sign detection algorithm with noise suppression and semantic enhancement. *PloS one*, 21(1):e0340810.

Yang, J. (2023). Traffic sign detection based on classic visual recognition models. *Journal of Physics: Conference Series*, 2646(1):012024.