

Environment aware detection of traffic signs on low power edge devices using lightweight YOLO model

Gopika SAGASRAM*, Kamesh SEKARAN, Maria Kalavathy GNANAMANI

Department of Information Technology, Easwari Engineering College, Chennai, Tamil Nadu, India

gopikasagasram@gmail.com (*Corresponding author), kameshsk2020@gmail.com,
maria_kalavathy@yahoo.co.in

Abstract: Intelligent transportation systems (ITS) rely on the real-time detection of traffic signs to support both autonomous vehicles and driver-assistance systems. Unfortunately, due to limited computing power, memory, and environmental conditions (such as low light, glare, rain, and fog), developing an accurate detection system that runs on edge devices remains highly challenging. To address these issues, this research presents an environment-aware framework that leverages a YOLOv8 model (a version of YOLO optimized for edge devices) combined with adaptive preprocessing; therefore, it can adjust, in real time, both preprocessing functions and detection parameters using information from environmental conditions to ensure accurate detection of traffic signs on CPU-only devices. Additionally, the pre-trained weights of the YOLOv8 model are adjusted based on a carefully curated dataset, and the resulting model can be quantized for use with ONNX. The proposed system achieves a mean Average Precision (mAP) of 91.0%, a precision of 96.1%, and a recall of 92.3% at up to 22 frames per second on low-resource platforms. This work contributes an environment-aware framework that fuses adaptive environmental condition analysis with YOLOv8 to achieve accurate traffic sign detection in real time.

Keywords: Traffic sign detection, YOLOv8, Intelligent transportation systems, Edge deployment, Real-time detection.

1. Introduction

The rapid acceleration of robotic transportation systems and robotic vehicles creates a need for precise, quick and real-time identification of traffic signs. Traffic signs tell drivers how to move safely and react to different road conditions, yet accurate identification in the real world remains challenging: variations in illumination, weather, and visual interference cause traditional vision systems to fail, and current deep-learning models, while accurate under controlled conditions, require processing power and memory well beyond what most edge devices can offer.

These challenges highlight the need for detection models that are resource-efficient and resilient to environmental change. YOLO and Fast R-CNN achieve high accuracy on benchmark datasets but do not meet the timing requirements of low-power edge devices, and neither adapts to environmental conditions; elsewhere in ITS, deep reinforcement learning has been applied to adaptive traffic signal control (Wang et al., 2021), underscoring the broader role of adaptive, learning-based methods in this domain. This work therefore proposes an environment-aware framework that pairs a lightweight, efficient YOLO model with adaptive image preprocessing, tailoring detection to the environment to improve feature saliency, suppress noise, and adjust detection parameters in real time. Incoming video frames are first assessed for brightness, contrast, and noise characteristics; adaptive preprocessing techniques, including CLAHE for contrast enhancement, gamma correction, glare reduction, and Gaussian filtering are then applied so that traffic signs remain prominent, and the purified frames are passed to a lightweight YOLO model trained on examples spanning diverse lighting, weather, and occlusion conditions. Feature fusion across scales, component pruning, and model compression allow the model to detect small features and run on boards such as the Raspberry Pi and NVIDIA Jetson, while detection thresholds and non-maximum suppression are tuned dynamically to the environment.

Adjusting confidence thresholds with visibility reduces false positives while preserving recall, and dynamic tuning together with the lightweight architecture enables real-time performance

on CPU-only edge hardware. The framework's modular design supports integration with autonomous vehicle systems, driver-assistance platforms, and roadside monitoring, addressing both environmental variation and edge-device constraints in a single solution that is more reliable, faster, and more energy-efficient than traditional traffic-sign detection approaches.

The remainder of this paper is structured as follows. Section 2 presents the related work. Section 3 outlines the system architecture, with specific focus on its operationality. Section 4 presents the results and discussions. Finally, Section 5 concludes the paper with the main findings and recommendations.

2. Literature survey

Recent research has advanced traffic sign detection for small, diverse, and environmentally degraded signs through scale-aware feature fusion: Chu et al. (2023) proposed TRD-YOLO (coordinate attention and a BiFPN neck in YOLOv5) for small-object mAP under low-light and rainy conditions; Chen et al. (2022) used a modified CSPDarknet backbone with multi-scale aggregation at over 60 FPS; Li et al. (2022) improved Faster R-CNN for autonomous-driving contexts; and Liu et al. (2021) proposed TSingNet, a scale-aware, context-rich feature-learning framework for signs in the wild.

Other work has targeted text-bearing signs and region-specific conditions: Khalid et al. (2024) combined YOLOv5s, MSER-based preprocessing, OCR, and NLP correction for multilingual, weather-degraded imagery, while Megalingam et al. (2023) developed an optimized YOLOv4-tiny model for Indian traffic signs with synthetic weather augmentation, achieving near real-time performance on Jetson Nano but with reduced accuracy in low-light conditions, motivating further robustness work.

Edge deployment and real-world adaptability have also been emphasized. Güney et al. (2022) demonstrated YOLOv4-tiny with TensorRT/FP16 optimization at over 30 FPS on embedded GPUs; Liang et al. (2022) improved Sparse R-CNN for small and occluded signs; and Li et al. (2023) enhanced YOLOv5 with multi-scale attention feature fusion. Wang et al. (2023) proposed a vehicle-mounted, multi-branch detector robust to fog, rain, and low light; Lai et al. (2023) and Shi et al. (2023) contributed STC-YOLO and SC-YOLO for complex environments; and lightweight models such as YOLO-TS (Liu & Luo, 2024), Sign-YOLO (Mahadshetti et al., 2024), and improved YOLOv7-tiny (Cao et al., 2024) balance accuracy with minimal computational cost.

Benchmarking and ensemble strategies have further supported the field: Zhang et al. (2022) introduced the CCTSDB 2021 benchmark, and Wang & Wang (2024) proposed a stacking ensemble for improved robustness. Surveys by Flores-Calero et al. (2024) and Zhu & Yan (2022) converge on three directions, namely, multi-scale/attention-based small-object detection, localization- and text-aware optimization, and deployment-oriented edge engineering, all of which motivate the present work, which advances lightweight, environment-adaptive traffic sign detection for practical edge deployment.

3. Proposed methodology

The system framework, depicted in Figure 1, is constructed to facilitate real-time, reliable traffic sign recognition across varied environmental conditions. It combines flexible image preprocessing, efficient object detection, adjustable parameter management, and performance tracking to ensure precise and effective functionality on edge computing devices. The component-based structure enables individual optimization of each element while preserving the system's overall responsiveness.

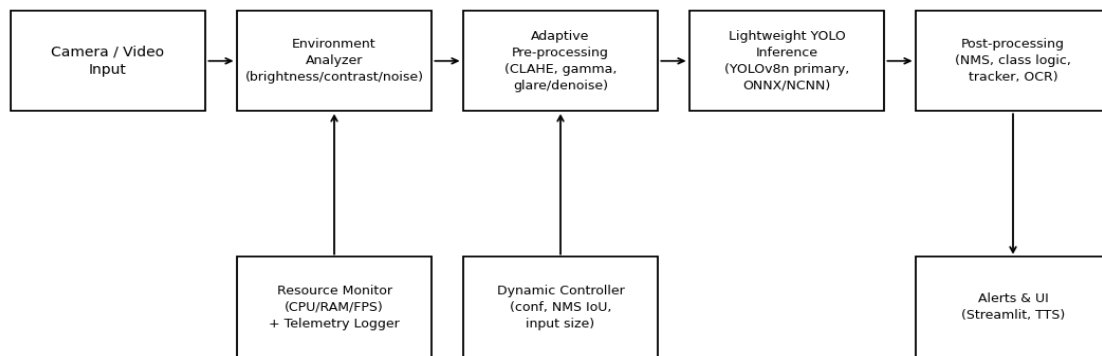


Figure 1. System architecture (Source: Authors' own work)

3.1. Camera input

The first stage collects raw video frames from vehicle-mounted cameras, roadside equipment, or recorded feeds, supporting flexible deployment while preserving image quality.

3.2. Environment analyzer

Reliable detection across environmental conditions requires first evaluating scene lighting and clarity, which is performed in real time using pixel brightness statistics for fast, lightweight categorization.

3.2.1. Mean brightness analysis

The mean brightness of an image is computed to quantify the overall illumination level. As shown in Figure 2, the input image is presented in three stages: the original RGB frame, its grayscale conversion, and a pixel-quantized representation for clearer visualization of intensity values. In the grayscale domain, each pixel intensity reflects the perceived brightness of the scene.

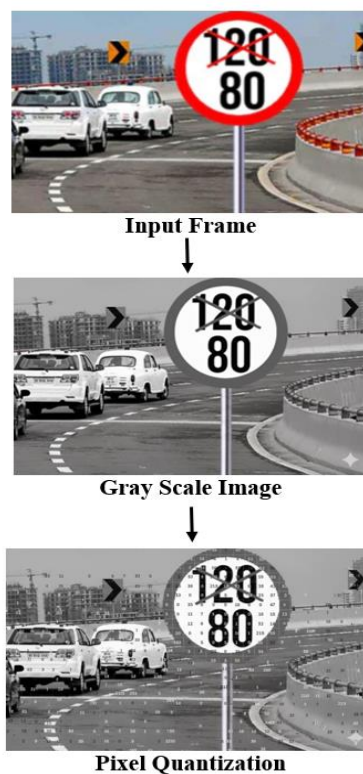


Figure 2. Pixel quantization for mean bright analysis and contrast analysis (Source: Authors' own work)

Let $I(i,j)$ denote the intensity of the pixel at position (i,j) in a grayscale frame of dimensions $H \times W$. The mean brightness μ is given in Equation (1):

$$\mu = \frac{1}{H * W} \sum_{i=1}^H \sum_{j=1}^W I(i,j) \quad (1)$$

In general, the computed value of μ enables classification of the scene into the following: low mean brightness ($\mu < 50$) indicating night-time or low-light conditions; high mean brightness ($\mu > 180$) corresponding to overexposed or glare-affected frames; and intermediate values ($50 \leq \mu \leq 180$) indicating normal daylight conditions.

This classification informs the choice of adaptive preprocessing techniques for low-light frames and over/underexposed images.

3.2.2. Contrast analysis

The intensity variations (contrast) across an image can be measured quantitatively. Contrast is the variation in pixel intensity across an image; low contrast means that the image was captured under conditions with limited visibility (e.g. fog, haze and rain) and high contrast means that the image was captured under conditions with high visibility (e.g. bright sunlight and shadow). When measuring the contrast of a grayscale image of size $H \times W$ with $I(i,j)$ being the pixel intensity and μ being the mean brightness of the image, the contrast is defined using the standard deviation of the pixel intensities:

$$\sigma = \sqrt{\frac{1}{H * W} \sum_{i=1}^H \sum_{j=1}^W (I(i,j) - \mu)^2} \quad (2)$$

Low contrast ($\sigma < 20$) indicates foggy, rainy, or low-visibility conditions; intermediate contrast ($20 \leq \sigma \leq 80$) indicates normal daylight scenes; and high contrast ($\sigma > 80$) indicates strong shadows, direct sunlight, or glare.

3.3. Adaptive preprocessing pipeline

To produce standardized frames for the detection network under varying lighting, an adaptive preprocessing pipeline uses the estimated mean (μ) and standard deviation (σ) of each incoming frame to select the appropriate technique:

Low-light ($\mu < 50$): For incoming frames that have been classified as low-light, i.e., $\mu < 50$, contrast-limited adaptive histogram equalization (CLAHE) was first applied to improve the local contrast of these frames by re-allocating pixels to different intensities levels within a small area of the image. For example, if a pixel at (x, y) had a value of xxx , the CLAHE equalized intensity value for that pixel would be calculated according to the following equation:

$$I'(x) = \frac{CDF(x) - CDF_{\min}}{N - CDF_{\min}} \cdot (L - 1) \quad (3)$$

where $CDF(x)$ is the cumulative distribution function (CDF) of the histogram; N = the number of pixels within the local region; and L = the number of grey levels available within the frame. The application of Equation (3) increases the intensity levels of the faint edges of the traffic signs while minimising the amount of noise added to the image.

The global brightness of the input image is then adjusted by using the gamma function, as represented in Equation (4). A visual example of how traffic signs would be displayed when viewed in a low-light frame, compared to a gamma-corrected frame, can be found in Figure 3.

$$I'(x) = 255 \cdot \left(\frac{I(x)}{255} \right)^\gamma \quad (4)$$

where $I(x)$ = original pixel intensity and γ = gamma factor; when $\gamma < 1$, the image will appear brighter, while when $\gamma > 1$, the overexposed input will appear darker. As a result of the use of Equation (4), an even distribution of the overall intensity across the entire frame is ensured as a complementary to the CLAHE method.



Figure 3. Example of gamma correction. Left: Low-light road scene with traffic sign. Right: Enhanced visibility after applying gamma correction (Source: Authors' own work)

The transformation brightens dark pixels when $\gamma < 1$ and compresses bright regions when $\gamma > 1$. Table 1 demonstrates the effect of gamma correction on sample pixel intensities. For $\gamma = 0.5$, the values are lifted significantly, making dim regions more visible, whereas for $\gamma = 2.0$, the values are compressed, which is beneficial for suppressing overexposure. This example highlights how the choice of γ allows the system to adaptively enhance traffic sign visibility under different illumination conditions.

Table 1. Effect of gamma correction on sample pixel intensities

Input $I(x)$	Output $I'(x)(\gamma=0.5)$	Output $I'(x)(\gamma=2.0)$
30	87	4
80	143	25
150	196	88
220	237	190

Glare/Overexposure ($\mu > 180$): To limit intensity clipping from intense light sources (direct sunlight, headlights), highlight suppression caps very bright pixels, then redistributes intensities (normalization) so that detail hidden in overexposed regions becomes visible again, as shown in Equation (5) and illustrated in Figure 4, where the range $[40, 220]$ is stretched to $[0, 255]$.

$$I'(x) = \frac{I(x) - I_{\min}}{I_{\max} - I_{\min}} \cdot (L - 1) \quad (5)$$

where $I_{\min}$ and $I_{\max}$ are the minimum and maximum intensity values in the frame, and L is the dynamic range.

Fog, Rain, or Haze ($\sigma < 20$): In low-contrast conditions caused by scattering media, Gaussian denoising is applied to suppress noise while preserving important edges. A Gaussian kernel is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (6)$$

where (x, y) are spatial distances from the kernel center and σ controls the smoothing strength;

convolving the frame with this kernel reduces high-frequency noise, after which contrast enhancement restores edge sharpness for detection.

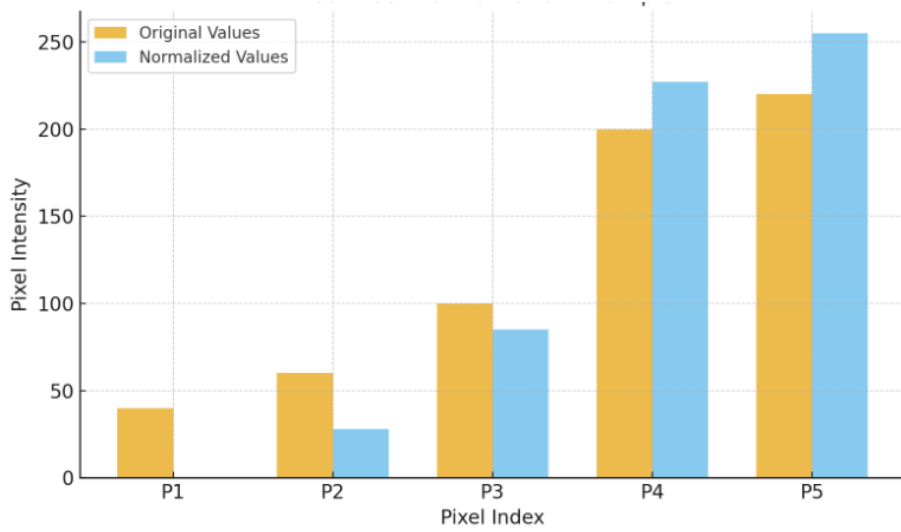


Figure 4. Original pixel intensities are stretched to the full dynamic range, improving visibility in glare-affected regions. (Source: Authors' own work)

These adaptive preprocessing steps ensure robust feature preservation and provide consistently enhanced frames for the YOLO-based detection network, thereby improving traffic sign recognition performance under varying environmental conditions.

3.4. Lightweight YOLO inference

The detection system employs a lightweight YOLO architecture optimized for edge deployment, specifically YOLOv8n, the nano-scale variant used to produce all results reported in Section 4 of this paper. The model is initialized with pre-trained weights and trained on a hand-curated dataset using supervised learning. Techniques such as network pruning and quantization reduce model size and inference time without compromising accuracy. Specialized loss functions are used to optimize detection performance. Bounding box localization is guided by IoU-based losses to address imbalance.

IoU loss measures the overlap between predicted and ground-truth boxes. Equation (7) is as follows:

$$IoU = \frac{|B_{pred} \cap B_{gt}|}{|B_{pred} \cup B_{gt}|} \quad (7)$$

GIoU extends IoU by adding a penalty for the distance between the predicted and ground-truth boxes when they do not overlap. Equation (8) is as follows:

$$GIoU = 1 - IoU + \frac{|C - (B_{pred} \cup B_{gt})|}{|C|} \quad (8)$$

where C is the smallest enclosing box covering both predicted and ground-truth boxes.

DIoU incorporates the Euclidean distance between the centers of the predicted and ground-truth boxes. Equation (9) is as follows:

$$\mathcal{L}_{DIOU} = 1 - IoU + \frac{\rho^2(b_{pred}, b_{gt})}{c^2} \quad (9)$$

where ρ is the distance between box centers and c is the diagonal of the smallest enclosing box.

CIOU combines IoU, center distance, and aspect ratio consistency to produce a more comprehensive localization loss. Equation (10) is as follows:

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b_{pred}, b_{gt})}{c^2} + \alpha v \quad (10)$$

where v accounts for aspect ratio differences and α is a weighting factor.

Input frames are adaptively preprocessed for contrast, brightness, and noise, and the YOLO network outputs bounding boxes, class labels, and confidence scores. The lightweight design, optimized architecture, and loss function strategy enable real-time detection on low-power hardware while maintaining robust performance under varying environmental conditions.

3.5. Post processing

After the YOLO model generates raw detections, the post-processing stage refines these outputs into accurate, meaningful, actionable results. Non-Maximum Suppression (NMS) first removes redundant, overlapping bounding boxes for the same object, retaining only the highest-confidence box when IoU exceeds a threshold, so that each object is counted once and false positives are reduced. Class logic then interprets detections by category, for example, triggering an alert for a stop sign or prioritizing pedestrian detections during autonomous navigation, and can filter out irrelevant classes to limit unnecessary downstream processing.

Object tracking (e.g., SORT) maintains detection identity across frames using motion and appearance cues, supporting trajectory and velocity estimation for traffic monitoring and autonomous driving; when a tracked detection contains text, Optical Character Recognition (OCR) extracts it to support downstream actions such as database lookups and compliance checks. Together, NMS, class logic, tracking, and OCR filter and contextualize raw detections into structured outputs for alerting, visualization, and user interaction.

3.6. UI and alerts

The UI and Alerts module concludes the pipeline, presenting a Streamlit interface with detection boxes, category names, confidence scores, and system performance data (frame rate, CPU/memory usage). Operators can adjust confidence and NMS thresholds at runtime, and Text-to-Speech notifications flag critical or security-related detections, ensuring prompt, accessible results for continuous surveillance and operational decision making.

4. Experiments and results

4.1. Data set

The dataset produced for this study was created to ensure the diversity and reliability of traffic signs when operating in real-world environments. Traffic signs were collected from publicly available resources, as well as from direct-field images. All the images included variances in the environment such as low-visibility conditions, excessive glare, foggy conditions, and partially obstructed views. All images underwent extensive annotation and processing to produce a standardized dataset suitable for both model training and evaluation/validation.

In addition to utilizing comprehensive data augmentation techniques to assist the system's ability for generalization, diverse techniques were used to enhance the dataset, including rotating images, changing the size of the images, changing the brightness and contrast of the image, adding noise to the images, and altering the perspective. The augmentation techniques reduced the overall chance of overfitting while increasing the system's robustness, resulting in an effective framework for handling challenging real-world conditions. Examples of images obtained through the data collection process are shown in Figure 5.



Figure 5. Example of dataset images (Source: Authors' own work)

The data was then cleaned, resulting in 15,860 images. We divided them into 13,300 training images, 1,632 validation images and 928 test images. The validation set was used for hyperparameter tuning and model selection during training, while the held-out test set of 928 images was reserved exclusively for the final, independent performance evaluation reported in this paper. The dataset was prepared and enlarged to cover a wide variety of traffic sign types, thus, it is suitable for training a detection system that can be implemented in real time and on the edge devices.

4.2. Evaluation methods

The performance of the proposed traffic sign detection system was assessed using mean Average Precision (mAP), Precision, Recall, F1-score, and Frames Per Second (FPS). Table 2 presents the four classification metrics, which are computed according to whether each predicted sample is correctly classified as positive or negative.

Table 2. Evaluation Metrics in Classification

TP	The number of positive samples that are correctly predicted as positive by the model.
TN	The number of negative samples that are correctly predicted as negative.
FP	The number of negative samples that are incorrectly predicted as positive.
FN	The number of positive samples that are incorrectly predicted as negative.

Precision represents the accuracy of correctly predicting positive samples. Equation (11) is as follows:

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

High precision ensures that most detected objects are true traffic signs, minimizing false alarms.

Recall represents the coverage rate of correctly predicted positive samples. Equation (12) is as follows:

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

High recall implies that the majority of traffic signs in the scene are identified.

F1-score represents the harmonic mean of precision and recall. Equation (13) is as follows:

$$F1-Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (13)$$

The F1-score balances precision and recall, providing a single indicator of detection robustness.

Average Precision (AP) represents the accuracy for a single category. It is calculated as the area under the precision–recall curve. Equation (14) is as follows:

$$AP = \int_0^1 (Precision) d(Recall) \quad (14)$$

Mean Average Precision (mAP) represents the mean of Average Precisions across all categories. Equation (15) is as follows:

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i \quad (15)$$

Frames Per Second (FPS) measures inference speed on edge devices, defined as the number of processed frames per second. High FPS ensures that the system is running in real time.

Together, these metrics comprehensively evaluate accuracy (mAP, Precision, Recall, F1-score) and efficiency (FPS), making them suitable for assessing traffic sign detection performance in diverse environmental conditions.

4.3. Performance analysis

The proposed system was evaluated on images captured under bright sun, low light, glare, fog, and rain, correctly identifying signs in 91% of cases overall. Table 3 reports precision, recall, F1-score, mAP, and inference speed for each condition, and Figure 6 shows representative detection outputs. The results indicate that the system maintains high detection reliability in real time on resource-constrained hardware across all tested conditions.



Figure 6. Example of detection outputs from the proposed environment-aware YOLO system, showing traffic signs identified with bounding boxes, class labels, and confidence scores under diverse road conditions (Source: Authors' own work)

Performance was strongest in clear daylight, and the adaptive preprocessing pipeline, including CLAHE and gamma correction for low light, highlight suppression for glare, and Gaussian denoising for fog/rain, maintained accuracy high across the remaining conditions, with only a modest drop in mAP and inference speed as conditions worsened (Table 3).

Table 3. Performance analysis

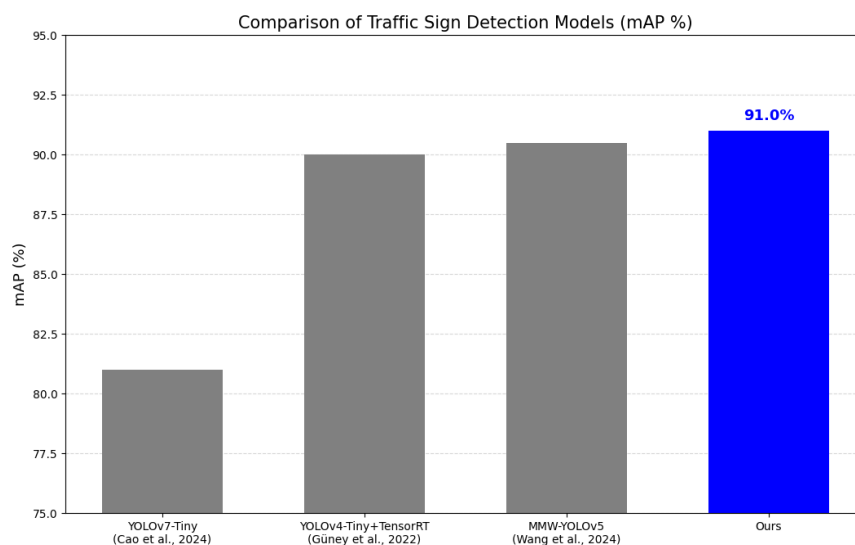
Condition	mAP (%)	Precision (%)	Recall (%)	F1-score	FPS (Edge Device)
Clear daylight	94.2	96.1	92.3	94.1	22.0
Low-light	91.0	92.4	89.5	90.9	21.5
Glare	90.3	91.2	88.7	89.9	21.2
Fog / Rain	88.5	89.1	87.9	88.5	20.8

Under glare, lightweight contrast normalization limited the accuracy loss, and under fog/rain, Gaussian denoising and contrast enhancement offset the effects of added noise and reduced visibility; both conditions still achieved mAP above 88% at above 20 FPS (Table 3), representing the lowest performance observed among the four tested conditions.

The evaluation in this study focuses on aggregate detection metrics (mAP, precision, recall, F1-score) across the four environmental conditions, computed on the held-out test set. Per-class Average Precision and variance across repeated training runs were outside the scope of the present study and are identified as directions for future work.

4.4. Results discussion

Comparative experiments were conducted to evaluate the effectiveness of the proposed Environment-Aware YOLO variant for traffic sign detection. A custom dataset of 15,860 images was used to train and evaluate the proposed model, and its performance was benchmarked against the mAP values reported in the respective published studies for several recent lightweight object detection approaches, including YOLOv7-Tiny (Cao et al., 2024), YOLOv4-Tiny accelerated with TensorRT (Güney et al., 2022), and MMW-YOLOv5 (Wang et al., 2024). These baseline models were not independently retrained on the dataset used in this study; therefore, the comparison presented in Figure 7 should be interpreted as a positioning against literature-reported performance rather than as a controlled head-to-head evaluation under identical conditions. The quantitative results are illustrated in Figure 7. The proposed method achieved a higher reported mAP than the reference values available for the selected lightweight baseline approaches. Specifically, YOLOv7-Tiny reported approximately 81% mAP, while YOLOv4-Tiny with TensorRT optimization reported around 90% mAP. MMW-YOLOv5 reported 90.5% mAP, whereas the proposed Environment-Aware YOLO model achieved 91.0% mAP on the held-out test set of this study. These results are encouraging, though a controlled re-evaluation of the baseline models on the same dataset and hardware remains necessary to draw a fully rigorous comparative conclusion.

**Figure 7.** Performance comparison (Source: Authors' own work)

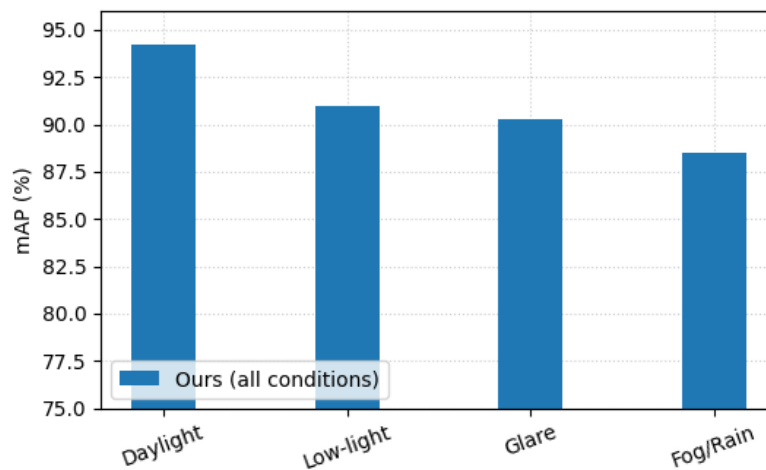


Figure 8. Performance across different environment (Source: Authors' own work)

To assess the capability of the proposed framework under various environmental conditions, the detection performance of the proposed framework was evaluated during daytime, under low-light conditions, in the presence of glare, and under foggy and rainy conditions. Unlike Figure 7, which compares the proposed model against the baseline models, Figure 8 reports only the per-condition breakdown of the proposed model itself, since a per-condition breakdown was not available for the baseline models. As shown in Figure 8, the overall performance of the proposed framework was a Mean Average Precision (mAP) of 94.2% during the day, 91.0% during low light, 90.3% in the presence of glare, and 88.5% under foggy and rainy conditions. The results of this work indicate that the proposed framework has the ability to maintain high levels of accuracy. The reduction of accuracy from the best to worst conditions was less than 6%, which indicates real life consistent detection, regardless of the lighting or weather conditions.

The proposed method achieved a maximum precision of 96.1%, recall of 92.3%, and an F1-score of 94.1%. The inference speed was 22 frames per second (FPS) for edge devices, which was marginally slower than the GPU-optimized frameworks, such as YOLOv7-Tiny (which has a speed of 104 FPS). However, this slower speed is an acceptable trade-off for the high accuracy and ease of deployment in the real-time applications of intelligent transportation systems that are critical for both adaptabilities to changing environmental conditions and the performance of these applications. The experimental results overall validate that the environment-aware YOLO architecture effectively achieves high-quality and robust detection of traffic signs under various conditions and at the same time maintains real-time performance, as shown in Figure 7. The system's use of adaptive environmental examination, a lightweight detection structure, and optimizations for edge-specialized use enables reliable performance in realistic intelligent transportation scenarios and, as such, demonstrates significant improvement over conventional non-adaptive approaches.

5. Conclusion and future directions

The environment-aware traffic sign detection system presented here addresses environmental variability and resource constraints on embedded hardware by combining a lightweight YOLOv8n model, adaptive preprocessing, dynamic parameter calibration, and an environment analysis block. Experiments on the Raspberry Pi 4 and NVIDIA Jetson Nano achieved real-time performance above 20 FPS in CPU-only implementations within stringent memory and power budgets, supporting deployment on ADAS, autonomous navigation, and city-scale traffic surveillance, with adaptive measures improving recall of small, distant, or partly hidden signs without a substantial increase in false positives. Network pruning and quantization keep the architecture easy to extend and standardize, and the results overall suggest that environment-aware, edge-optimized detection can move traffic sign recognition from research prototypes toward reliable, energy-aware operation in dynamic real-world conditions. Future work includes multi-class object detection, contextual

attention models, on-device adaptive learning, and a more granular evaluation protocol reporting per-class Average Precision and variance across repeated training runs.

Author contributions

Conceptualization: G. S., K. S. and M. K. G.; Data curation: G. S., K. S. and M. K. G.; Project administration: G. S., K. S. and M. K. G.; Supervision: K. S. and M. K. G.; Validation: G. S. and K. S.; Writing - original draft: G. S.; Writing - review and editing: G. S., K. S. and M. K. G..

Submission received: 12 January 2026; Revised: 07 September 2026; Accepted: 16 September 2026; Published: 30 September 2026.

REFERENCES

- Cao, X., Xu, Y., He, J., Liu, J. & Wang, Y. (2024) A lightweight traffic sign detection method with improved YOLOv7-Tiny. *IEEE Access*. 12, 105131–105147. <https://doi.org/10.1109/ACCESS.2024.3435384>.
- Chen, J., Jia, K., Chen, W. et al. (2022) A real-time and high-precision method for small traffic-signs recognition. *Neural Computing and Applications*. 34(3), 2233–2245.
- Chu, J., Zhang, C., Yan, M., Zhang, H. & Ge, T. (2023) TRD-YOLO: A real-time, high-performance small traffic sign detection algorithm. *Sensors*. 23(8), 3871.
- Flores-Calero, M., Astudillo, C.A., Guevara, D., Maza, J., Lita, B.S., Defaz, B. & Armingol Moreno, J.M. (2024) Traffic sign detection and recognition using YOLO object detection algorithm: A systematic review, *Mathematics*. 12(2), 297.
- Güney, E., Bayilmiş, C. & Cakan, B. (2022) An implementation of real-time traffic signs and road objects detection based on mobile GPU platforms. *IEEE Access* 10, 86191–86203.
- Khalid, S., Shah, J.H., Sharif, M., Dahan, F., Saleem, R. & Masood, A. (2024) A robust intelligent system for text-based traffic signs detection and recognition in challenging weather conditions. *IEEE Access*. 12, 78261–78274. <https://doi.org/10.1109/ACCESS.2024.3401044>.
- Lai, H., Chen, L., Liu, W., Yan, Z. & Ye, S. (2023) STC-YOLO: Small object detection network for traffic signs in complex environments. *Sensors*. 23(11), 5307.
- Li, A., Sun, S., Zhang, Z., Feng, M., Wu, C. & Li, W. (2023) A multi-scale traffic object detection algorithm for road scenes based on improved YOLOv5. *Electronics*. 12(4), 878.
- Li, X., Xie, Z., Deng, X., Wu, Y. & Pi, Y. (2022) Traffic sign detection based on improved Faster R-CNN for autonomous driving. *The Journal of Supercomputing*. 78(6), 7982–8002.
- Liang, T., Bao, H., Pan, W. & Pan, F. (2022) Traffic sign detection via improved sparse R-CNN for autonomous vehicles. *Journal of Advanced Transportation*. 2022, Article ID 3825532.
- Liu, Y. & Luo, P. (2024) YOLO-TS: A lightweight YOLO model for traffic sign detection. *IEEE Access*. 12, 169013–169023. <https://doi.org/10.1109/ACCESS.2024.3498057>.
- Liu, Y., Peng, J., Xue, J.H. et al. (2021) TSingNet: Scale-aware and context-rich feature learning for traffic sign detection and recognition in the wild. *Neurocomputing*. 447, 10–22.
- Mahadshetti, R., Kim, J. & Um, T.-W. (2024) Sign-YOLO: Traffic sign detection using attention-based YOLOv7. *IEEE Access* 12. 132689–132700. <https://doi.org/10.1109/ACCESS.2024.3417023>.
- Megalingam, R.K., Thanigundala, K., Musani, S.R. et al. (2023) Indian traffic sign detection and recognition using deep learning. *International Journal of Transportation Science and Technology*. 12(3), 683–699.

Shi, Y., Li, X. & Chen, M. (2023) SC-YOLO: An object detection model for small traffic signs. *IEEE Access*. 11, 11500–11510.

Wang, J., Chen, Y., Ji, X. et al. (2023) Vehicle-mounted adaptive traffic sign detector for small-sized signs in multiple working conditions. *IEEE Transactions on Intelligent Transportation Systems*. 25(1), 710–724.

Wang, M., Wu, L., Li, J. & He, L. (2021) Traffic signal control with reinforcement learning based on region-aware cooperative strategy. *IEEE Transactions on Intelligent Transportation Systems*. 23(7), 6774–6785.

Wang, T., Zhang, J., Ren, B. & Liu, B. (2024) MMW-YOLOv5: A multi-scale enhanced traffic sign detection algorithm. *IEEE Access*. 12, 148880–148892. <https://doi.org/10.1109/ACCESS.2024.3476371>.

Wang, Y. & Wang, Q. (2024) Robust stacking ensemble model for traffic sign detection and recognition. *IEEE Access*. 12, 178941–178950. <https://doi.org/10.1109/ACCESS.2024.3504827>.

Zhang, J., Zou, X., Kuang, L.D. et al. (2022) CCTSDB 2021: A more comprehensive traffic sign detection benchmark. *Human-centric Computing and Information Sciences*, 12. <https://doi.org/10.22967/HGIS.2022.12.023>.

Zhu, Y. & Yan, W.Q. (2022) Traffic sign recognition based on deep learning. *Multimedia Tools and Applications*. 81(13), 17779–17791.



Gopika SAGASRAM is currently a final-year B.Tech student in Information Technology at Easwari Engineering College, India. Her interests include software engineering, web development, and machine learning, with recent project work spanning full-stack development (MongoDB, Express.js, React.js, Node.js) and applications of deep learning and computer vision in transportation.



Kamesh SEKARAN is currently a final-year B. Tech student in Information Technology at Easwari Engineering College, India. His interests include software engineering, web development, and machine learning, with recent project work on a student management system (HTML, Tailwind CSS, MySQL, Java) and applications of deep learning and computer vision in transportation.



Maria Kalavathy GNANAMANI received the B.E. degree in Computer Engineering from Govt. College of Engineering, Tirunelveli, the M.E degree in Computer Science and Engineering from Sathyabama Institute of Science and Technology (2002), and a Ph.D degree from Sathyabama University (2011). With 27 years of teaching experience, she is currently a Professor in the Department of Information Technology at Easwari Engineering College, Ramapuram, and a recognized Ph.D. supervisor at Anna University who has guided six Ph.D. scholars to completion. Her interests include Service-Oriented Architecture, multimedia streaming, distributed computing, information retrieval, image processing, and machine learning; she holds six patents granted in India and serves as a reviewer for IEEE conference.



This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.