Yolov8-Based Vehicle Detection and Traffic Density Estimation for Adaptive Traffic Signal Applications

Muhammad Fakhrulradzi Razali[1], Ahmad Kamalrulzaman Othman[2], Azlan Abdul Aziz[3]*, and Yusnita Sokman[2]

[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Shah Alam, Bangunan Al-Khawarizmi, 40450 Shah Alam, Selangor Darul Ehsan, Malaysia

[2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Johor, Kampus Segamat, 85000 Segamat, Johor, Malaysia

[3]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Melaka, Kampus Jasin, 77300 Merlimau, Melaka, Malaysia,

DOI: https://doi.org/10.51584/IJRIAS.2026.11080078

Received: 22 August 2026; Accepted: 27 August 2026; Published: 07 September 2026

ABSTRACT

Fixed-time traffic signals cannot respond to short-term changes in vehicle demand and may allocate green time inefficiently at urban intersections. This study developed and evaluated a YOLOv8-based vehicle-detection and traffic-density estimation prototype for adaptive signal applications. Two public traffic-intersection datasets supplied 3,300 images, of which 2,932 were retained after quality screening. Images were resized to 640 × 640 pixels, annotated with vehicle bounding boxes, and augmented through ±25% brightness and saturation adjustments. YOLOv8 was trained for 50 epochs using 70:30, 80:20, and 90:10 training-testing splits and was compared with Faster R-CNN under the 70:30 configuration. YOLOv8 achieved mAP@0.5 values of 0.989–0.992 and mAP@0.5:0.95 values of 0.962–0.965. Under the direct comparison, YOLOv8 obtained 0.989 mAP@0.5, 0.962 mAP@0.5:0.95, and 0.980 recall, while requiring 1.9 ms per image compared with 28.4 ms for Faster R-CNN. The detector was integrated into a web prototype that displayed vehicle counts, estimated density, and signal duration. The results support YOLOv8 as an efficient traffic-monitoring component, although density-classification accuracy and traffic-flow improvement require separate validation.

Keywords: YOLOv8, vehicle detection, traffic density estimation, adaptive traffic signal, computer vision.

INTRODUCTION

Rapid urbanisation and population growth have increased the number of vehicles operating within urban road networks, placing substantial pressure on existing traffic-management infrastructure. Although road networks have expanded, this expansion has not fully addressed transportation demand in densely populated cities (Kanungo et al., 2014). Traffic congestion remains particularly severe at signalised intersections where conventional traffic lights operate according to predetermined timing cycles. These fixed-time systems cannot respond effectively to traffic-volume fluctuations across different periods of the day, causing vehicles to remain stationary even when other approaches to an intersection are lightly occupied or empty (Siripatana et al., 2021). Prolonged waiting contributes to longer travel times and unnecessary fuel consumption. It also increases vehicle emissions because engines continue operating while vehicles remain idle. Such inefficiencies affect transportation productivity and urban environmental quality. An intelligent traffic-control system that responds to real-time road conditions is therefore important for improving traffic flow and supporting more sustainable urban mobility.

Previous research has investigated image processing, optimisation, distributed computing, and reinforcement learning for adaptive traffic control. Camera-based systems estimate road density directly from intersection imagery (Biradar et al., 2022), while learning and optimisation methods adjust phase decisions from traffic-state

information. Reported approaches include distributed processing for connected vehicles (Agafonov, 2022), fuzzy inference with deep reinforcement learning (Kumar et al., 2021), Q-learning controllers (Bouderba & Moussa, 2019), genetic optimisation of waiting time (Sankaranarayanan et al., 2024), cloud-supported vehicular control (Gaouar & Lehsaini, 2021), fog-computing architectures (Minh et al., 2018), and fuzzy fog-cloud control based on road density (Septyanto et al., 2021). Real-time adaptive systems and deep Q-learning methods further demonstrate the value of responsive phase decisions (Shankaran & Rajendran, 2021; Xu et al., 2022; Tigga et al., 2022). Vision-based detection is attractive because existing cameras can provide direct observations. YOLO models perform localisation and classification in one inference pipeline and have been applied to vehicle detection and adaptive traffic applications (Zaatouri & Ezzedine, 2018; Mou et al., 2023; Sirphy & Thanga Revathi, 2023). YOLOv8 combines feature extraction, multi-scale aggregation, and a detection head suitable for rapid object localisation (Talukdar et al., 2023; Ultralytics, 2024).

Despite these developments, current traffic-management systems continue to face limitations in adapting to unpredictable and uneven traffic demand. Fixed-time controllers use predefined schedules that remain unchanged even when traffic conditions vary substantially, which can create long queues on heavily occupied roads while allocating unnecessary green-light time to lightly occupied approaches (Biradar et al., 2022). Existing studies have demonstrated the potential of intelligent control algorithms, but many approaches depend on simulations, specialised communication infrastructure, or computationally expensive models. A further gap concerns the integration of real-time vehicle detection with a functional traffic-density classification and signal-timing prototype. Object-detection accuracy alone does not demonstrate that a system can support practical traffic management. The selected model must also process video frames quickly enough for real-time operation and provide vehicle counts that can be translated into meaningful traffic-density levels. Therefore, a need remains for a computationally efficient traffic-monitoring approach that can detect vehicles accurately, estimate lane density, and provide input for dynamic traffic-signal timing through an accessible prototype.

This study addresses the implementation gap by developing and evaluating a YOLOv8 vehicle detector, comparing it with Faster R-CNN, and integrating the trained model into a web-based traffic-monitoring prototype. Traffic images were cleaned, resized, annotated, and augmented before model training. YOLOv8 was evaluated under 70:30, 80:20, and 90:10 training-testing splits using mean average precision, recall, training duration, and per-image inference speed. The prototype processed traffic video, counted detected vehicles, displayed a density level, and generated a signal-duration estimate. The quantitative evaluation therefore concerns vehicle detection and computational performance. Density classification and signal timing are treated as prototype functions because their accuracy and traffic-flow effects were not measured independently.

MATERIALS AND METHODS

Dataset Acquisition

Traffic-intersection datasets were collected to train and evaluate the YOLOv8 model. The study used publicly available image and video data obtained from online dataset repositories, including Kaggle and Hugging Face. The datasets were selected because they contained traffic-intersection scenes and vehicles observed under different levels of congestion.

The datasets explicitly identified in the research methodology were:

1. The Multiview Traffic Intersection Dataset available through Kaggle.

2. The City Intersection Computer Vision Dataset available through Kaggle.

The collected data consisted of raw traffic-intersection images or video frames intended to represent vehicle activity within road junctions. These data formed the basis for training the object-detection model to recognise vehicles and support subsequent traffic-density classification.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VIII August 2026

The data-collection process focused on obtaining scenes in which vehicles could be identified at intersections. The dataset was intended to include varying levels of traffic congestion so that the trained model could learn to recognise vehicles under different traffic-density conditions.

Data Preprocessing

The raw datasets were pre-processed before model training to improve data consistency and suitability for YOLOv8. The preprocessing workflow involved data cleaning, image resizing, image annotation, and the preparation of the required dataset structure.

Data cleaning was conducted to remove redundant, noisy, or unsuitable samples. Images that could negatively affect the training process were excluded to improve the overall quality of the dataset. The remaining images were resized according to the input requirements of the YOLOv8 model, ensuring that all samples had a consistent spatial resolution during model training.

Image annotation was then performed to identify the objects relevant to the study. Vehicles appearing in the traffic scenes were marked so that the model could learn their locations and associated object labels. The annotation process produced bounding-box information that isolated vehicle objects from the surrounding road environment.

Roboflow was used to facilitate the preprocessing and annotation workflow. The software supported the preparation of the dataset for model training by organising the images and their corresponding annotations into a format compatible with the selected detection model. The final output of this phase was a cleaned, resized, annotated, and pre-processed dataset suitable for YOLOv8 training and evaluation.

The preprocessing workflow used bounding-box annotations to identify vehicle objects. Because segmentation masks were not documented, the study was treated as an object-detection task rather than a semantic or instance-segmentation task.

YOLOv8 Model Development

The pre-processed traffic-intersection dataset was used to train and fine-tune a YOLOv8 object-detection model. The purpose of the model was to detect vehicles within traffic scenes and produce vehicle information that could subsequently be used to estimate traffic density.

Python was used as the primary programming language for model development. The implementation relied on the YOLOv8 framework and supporting software libraries. The training procedure followed the available YOLOv8 documentation, while research papers, academic journals, technical resources, and relevant GitHub repositories were consulted to guide model implementation and configuration.

During training, the model learned to associate the annotated image regions with vehicle objects. The model was fine-tuned using the prepared traffic dataset to improve its ability to detect vehicles under different traffic conditions. The expected output of the development phase was a functional YOLOv8 model capable of locating vehicles and providing detection results that could be used to calculate traffic-density information.

Prototype Architecture and Development

Following model development, a prototype architecture was designed to integrate the trained YOLOv8 model into a smart traffic-light control system. The architecture was intended to receive traffic-scene input, process the input using the trained neural network, detect vehicles, determine traffic density, and provide information for traffic-signal control.

The prototype design considered the input data, neural-network processing component, traffic-detection requirements, and system outputs. The trained model served as the main computer-vision component. Traffic images or video frames were provided as input to the model, after which detected vehicles were used to estimate the traffic condition at an intersection.

The prototype pipeline comprised traffic-video acquisition, frame transfer to YOLOv8, vehicle localisation, extraction of vehicle counts, assignment of a traffic-density level, generation of signal-timing information, and presentation through a web interface. The interface was implemented using Python, HTML, React.js, CSS, and JavaScript, while generated records were stored for CSV export.

The study did not report architectural modifications to the YOLOv8 backbone, neck, detection head, feature-fusion mechanism, or loss functions. The model was therefore evaluated as a trained or fine-tuned YOLOv8 implementation.

Model Evaluation

The trained YOLOv8 model was evaluated using data separated from the training dataset. The use of a separate testing dataset was intended to reduce evaluation bias and assess how well the model generalised to images that were not used during parameter optimisation.

Model performance was evaluated primarily through mean average precision. Two detection metrics were identified:

- **mAP@0.5**, which evaluates detections at an intersection-over-union threshold of 0.5.
- **mAP@0.5:0.95**, which evaluates mean average precision across intersection-over-union thresholds ranging from 0.5 to 0.95.

These metrics were used to assess the model's ability to recognise and localise vehicles accurately. The first metric provides a less restrictive assessment based on a single overlap threshold, whereas the second evaluates detection quality across a broader and more demanding range of overlap thresholds.

System Testing

System testing was conducted after the trained model had been integrated into the prototype. A traffic-intersection video feed was used to create an interactive test environment in which the complete system could be assessed.

The system-testing procedure focused on three areas:

1. The precision of traffic-density classification.
2. The response speed of the system during traffic-signal optimisation.
3. The effectiveness of the traffic-management operation.

Detection accuracy, processing speed, and system response time were identified as the principal performance indicators. Detection accuracy represented the ability of the model to recognise vehicles correctly. Processing speed represented the time required to process image or video input. System response time represented the time required for the prototype to convert the detection output into a traffic-management response.

The testing phase was intended to establish whether the YOLOv8-based prototype could operate sufficiently quickly and accurately for real-time traffic monitoring. The resulting assessment was also intended to identify system strengths, limitations, and potential areas for further improvement.

RESULTS

Dataset Preparation

An initial dataset of 3,300 traffic-intersection images was collected from Kaggle. Following data-quality screening, 368 images were removed because they were considered poor-quality samples. The final dataset

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VIII August 2026

therefore contained 2,932 images, representing 88.84% of the original collection. The excluded images represented 11.16% of the collected data.

Table 1. Dataset quality distribution

Category	Number of images	Percentage
Retained images	2,932	88.84%
Removed poor-quality images	368	11.16%
Total	3,300	100.00%

The retained images were resized to 640 × 640 pixels to maintain consistent input dimensions during model training. Auto-orientation was also applied using Roboflow to standardise the image orientation. The preprocessing procedure reduced variation in image size and prepared the dataset for model training.

Brightness and saturation augmentation of ±25% was applied to account for variations in sunlight exposure and camera colour quality. The augmentation was applied only to the training subset after each dataset split. Consequently, the number of augmented training images differed among the 70:30, 80:20, and 90:10 experiments.

Table 2. Dataset distribution after augmentation

Training-testing split	Original training images	Augmented training images	Total training images	Test images
70:30	1,557	1,557	3,114	1,375
80:20	1,923	1,923	3,846	1,009
90:10	2,353	2,353	4,706	579

The dataset sizes reported in Table 2 indicate that the experiments were not evaluated using an identical test set. The 90:10 configuration used the largest training set but the smallest test set, whereas the 70:30 configuration used the smallest training set and the largest test set.

Model Configuration

The YOLOv8 model was trained using an input resolution of 640 × 640 pixels and a batch size of 16. The reported default learning rate was 0.01, with a momentum value of 0.937 and weight decay of 0.0005. All three experiments were conducted for 50 epochs using a local Python environment with GPU support and the Ultralytics libraries.

Table 3. Default YOLOv8 configuration

Parameter	Value
Model	YOLOv8
Batch size	16
Image width	640 pixels

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VIII August 2026

Image height	640 pixels
Momentum	0.937
Weight decay	0.0005
Learning rate	0.01
Epochs	50

Three experiments were reported. Experiment A compared YOLOv8 with Faster R-CNN using a 70:30 split. Experiment B evaluated YOLOv8 using an 80:20 split, while Experiment C evaluated YOLOv8 using a 90:10 split.

Experiment A: YOLOv8 and Faster R-CNN Comparison

Experiment A compared the detection performance and computational efficiency of YOLOv8 and Faster R-CNN. Both models were trained using the 70:30 dataset split for 50 epochs. The implementation used a local GPU-enabled Python environment. Ultralytics libraries were used for YOLOv8, while PyCOCO-related libraries were used for Faster R-CNN evaluation.

Table 4. Performance comparison between YOLOv8 and Faster R-CNN

Metric	**YOLOv8**	**Faster R-CNN**
mAP@0.5	0.989	0.967
mAP@0.5:0.95	0.962	0.886
Recall	0.980	0.921
Training time	40 minutes	1 hour 12 minutes
Inference time per image	1.9 ms	28.4 ms

YOLOv8 produced higher detection performance than Faster R-CNN across all reported accuracy metrics. Its mAP@0.5 exceeded that of Faster R-CNN by 0.022, while its mAP@0.5:0.95 was higher by 0.076. YOLOv8 also achieved a recall of 0.980 compared with 0.921 for Faster R-CNN.

The most substantial difference occurred in computational performance. YOLOv8 required approximately 40 minutes for training, compared with 72 minutes for Faster R-CNN. The reported YOLOv8 inference time was 1.9 milliseconds per image, whereas Faster R-CNN required 28.4 milliseconds per image. Based on these values, YOLOv8 processed an image approximately 14.9 times faster than Faster R-CNN.

Experiment B: YOLOv8 with an 80:20 Split

Experiment B evaluated YOLOv8 using an 80:20 training-testing split. The model was trained for 50 epochs using the same local GPU-enabled Python environment and Ultralytics libraries.

The training curves showed downward trends in bounding-box loss, classification loss, and distribution focal loss. The declining bounding-box loss indicated that the model progressively improved its ability to estimate vehicle locations. The reduction in classification loss indicated improved recognition of the annotated class, while the downward distribution focal loss suggested improved prediction of bounding-box position and size.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VIII August 2026

Table 5. YOLOv8 performance using the 80:20 split

Metric	Result
mAP@0.5	0.991
mAP@0.5:0.95	0.963
Recall	0.967
Training time	43 minutes

The model achieved an mAP@0.5 of 0.991 and an mAP@0.5:0.95 of 0.963. Recall reached 0.967, and the training process required approximately 43 minutes. The reported metric curves indicated that precision and recall remained above approximately 0.97 during the later stages of training.

Experiment C: YOLOv8 with a 90:10 Split

Experiment C evaluated YOLOv8 using a 90:10 training-testing split. The model was trained for 50 epochs using the same general hardware and software environment as Experiment B.

The training curves showed consistent reductions in bounding-box loss, classification loss, and distribution focal loss. The converging loss curves indicated that the model learned the training data effectively. The curves did not indicate underfitting, although no formal overfitting analysis or comparison between training and validation losses was reported.

Table 6. YOLOv8 performance using the 90:10 split

Metric	Result
mAP@0.5	0.992
mAP@0.5:0.95	0.965
Recall	0.982
Training time	50 minutes

Experiment C achieved the highest reported values among the three YOLOv8 experiments. Its mAP@0.5 reached 0.992, while its mAP@0.5:0.95 reached 0.965. The model also achieved a recall of 0.982. Training required approximately 50 minutes, which was longer than the training time reported for Experiments A and B.

Comparison Across Dataset Splits

The three YOLOv8 experiments produced closely grouped detection results. The mAP@0.5 values ranged from 0.989 to 0.992, while mAP@0.5:0.95 ranged from 0.962 to 0.965.

Table 7. Overall YOLOv8 results

Experiment	Training-testing split	mAP@0.5	mAP@0.5:0.95
A	70:30	0.989	0.962
B	80:20	0.991	0.963
C	90:10	0.992	0.965

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VIII August 2026

Increasing the proportion of training data was associated with a small increase in both reported mAP measures. Compared with Experiment A, Experiment C increased mAP@0.5 by 0.003 and mAP@0.5:0.95 by 0.003. These differences correspond to increases of 0.3 percentage points for both metrics.

Although Experiment C achieved the highest numerical results, its test set contained only 579 images, compared with 1,009 images in Experiment B and 1,375 images in Experiment A. The results were therefore obtained from different test-set sizes and potentially different test-set compositions.

Prototype Implementation

The trained YOLOv8 model was integrated into a web-based traffic-monitoring prototype. The interface was developed using HTML, CSS, and JavaScript. The prototype dashboard displayed a traffic video feed, the number of detected vehicles within individual lanes, the estimated traffic-density level, and the corresponding traffic-light timer.

The prototype also stored the generated traffic data in a database. The recorded information could be exported in comma-separated value format for subsequent analysis. The dashboard therefore demonstrated the integration of vehicle detection, vehicle counting, density estimation, timer generation, data storage, and visual presentation within a single application.

The dashboard confirmed that the trained model was integrated into a functional interface. Quantitative results were not reported for density-classification accuracy, signal-timing accuracy, end-to-end latency, database performance, or traffic-flow improvement.

DISCUSSION

Detection Performance

The results indicate that YOLOv8 detected vehicles with high reported accuracy across all three dataset splits. Every experiment achieved an mAP@0.5 above 0.989 and an mAP@0.5:0.95 above 0.962. The relatively small difference between the two mAP measures suggests that the model maintained high localisation performance even when evaluated across stricter intersection-over-union thresholds.

The mAP@0.5:0.95 values are particularly relevant because this metric evaluates detection performance across multiple localisation thresholds. The values between 0.962 and 0.965 indicate that the model did not merely identify the presence of vehicles but also produced bounding boxes that closely overlapped with the annotated vehicle locations.

Recall values were also high. YOLOv8 achieved recall values of 0.980 in Experiment A, 0.967 in Experiment B, and 0.982 in Experiment C. These results suggest that the model missed relatively few annotated vehicles in the evaluated images. However, precision values were not reported in the tables, even though the discussion refers to precision values above 0.97. A complete assessment would require the exact precision and F1-score values.

The unusually high detection results require cautious interpretation. A model achieving approximately 99% mAP may reflect a well-defined single-class detection task, visually consistent images, limited environmental diversity, or substantial similarity between training and test samples. The study did not report whether images from the same video sequence, intersection, or camera were separated at the source level before dataset splitting. If adjacent or visually similar frames appeared in both training and testing sets, the reported values could overestimate generalisation to new intersections or unseen traffic conditions.

Comparison with Faster R-CNN

YOLOv8 outperformed Faster R-CNN in both detection accuracy and computational speed. The difference in mAP@0.5 was modest, but the difference in mAP@0.5:0.95 was more substantial. YOLOv8 achieved an

mAP@0.5:0.95 of 0.962, compared with 0.886 for Faster R-CNN. This result suggests that YOLOv8 produced more accurate vehicle localisation across stricter overlap thresholds.

The inference-time comparison provides stronger evidence for selecting YOLOv8 for real-time traffic monitoring. YOLOv8 required 1.9 milliseconds per image, while Faster R-CNN required 28.4 milliseconds. This difference is consistent with the architectural distinction between one-stage and two-stage detectors. YOLOv8 performs object localisation and classification within a single detection pipeline. Faster R-CNN first generates region proposals and then classifies the proposed regions, increasing computational cost.

The reported YOLOv8 inference time corresponds theoretically to more than 500 frames per second if preprocessing, data transfer, video decoding, visualisation, and application overhead are excluded. This theoretical rate should not be interpreted as the end-to-end frame rate of the deployed prototype because the reported value represents model inference only. Video decoding, preprocessing, data transfer, visualisation, and application logic would add processing overhead.

YOLOv8 also required less training time than Faster R-CNN. The 40-minute YOLOv8 training time was 32 minutes shorter than the 72-minute Faster R-CNN training time. Nevertheless, training-time comparisons are meaningful only when both models use comparable hardware, data loading, image resolution, batch size, pretrained weights, and optimisation settings. The available configuration details were insufficient to confirm that all such conditions were controlled.

Effect of Dataset Split Ratio

Experiment C produced the highest reported mAP and recall values. The study attributes this result to the larger proportion of images used for training. Increasing the training proportion from 70% to 90% provided the model with more examples from which to learn vehicle appearance and location patterns.

However, the improvement was small. The difference between Experiment A and Experiment C was only 0.003 for both mAP@0.5 and mAP@0.5:0.95. Such a small increase cannot establish that the 90:10 split is meaningfully superior without repeated experiments, uncertainty estimates, or statistical testing.

The comparison is also confounded by the use of different test sets. Each split changed both the quantity and composition of the evaluation data. The 90:10 experiment may have achieved higher results because it received more training images, because its smaller test set was less difficult, or because of both factors. The experiments therefore do not isolate the effect of training-set size.

A stronger experimental design would retain one independent test set and vary only the proportion of the remaining development data used for training and validation. Alternatively, repeated stratified splits or k-fold cross-validation could be used to estimate variability. Without such controls, Experiment C can be described only as producing the highest reported score, not as being conclusively the best model configuration.

Effect of Data Augmentation

Brightness and saturation augmentation were applied to simulate variations in illumination and camera colour quality. This strategy is relevant to traffic monitoring because outdoor images may differ substantially across daylight conditions and camera systems.

The augmentation procedure doubled the number of training images in every experiment, while the test sets remained unaugmented. No ablation experiment compared training with and without augmentation, so the independent contribution of augmentation to the final detection performance could not be determined.

Traffic-Density Classification and Signal Timing

The prototype demonstrates that YOLOv8 detections can be connected to a web-based interface that displays vehicle counts, density levels, and estimated signal durations. This implementation establishes technical integration between the detection model and the application interface.

The reported experiments evaluated vehicle detection rather than traffic-density classification. No results were provided for the accuracy of low-, medium-, or high-density labels. The study did not define the vehicle-count thresholds used to assign density categories or present a density-class confusion matrix, class-level precision, recall, F1 score, or comparison with manually assigned density labels.

The traffic-light timer was displayed as a prototype output, but the study did not specify the mathematical or rule-based relationship between vehicle count, density level, and signal duration. No baseline fixed-time controller was evaluated, and no measurements of average waiting time, queue length, vehicle throughput, travel time, or congestion reduction were reported.

The prototype results therefore support the feasibility of integrating detection outputs into a traffic-management dashboard. They do not yet demonstrate that the generated timer improves traffic flow or reduces congestion.

Validity of the Findings

The study provides evidence that YOLOv8 performed strongly on the selected traffic-image dataset and offered faster inference than Faster R-CNN under the reported conditions. The results also show that the trained model could be incorporated into a functional web-based prototype.

Several factors limit the strength of the conclusions:

1. The three dataset splits used different test sets, preventing a controlled comparison of training-set size.
2. The study did not report repeated runs, standard deviations, confidence intervals, or statistical significance.
3. The source-level separation of images is not described, creating a possible risk of leakage between related video frames.
4. Exact precision and F1-score values are absent.
5. Density-classification performance is not quantitatively evaluated.
6. Signal-timing effectiveness is not tested against fixed-time or adaptive baselines.
7. End-to-end prototype latency is not reported.
8. The exact YOLOv8 variant, pretrained weights, optimiser, learning-rate schedule, confidence and intersection-over-union thresholds, random seed, hardware specification, and software-library versions were not fully documented.
9. The dataset provenance by repository, geographical location, camera position, environmental condition, and vehicle class was not fully specified.
10. The density thresholds, signal-timing rules, and system-testing protocol were not documented in sufficient detail for independent replication.

Within these constraints, the most defensible finding is that YOLOv8 achieved high vehicle-detection performance on the evaluated dataset and provided a more favourable accuracy-speed balance than Faster R-CNN. Further experiments are required before concluding that the prototype improves traffic-signal efficiency or generalises reliably across different intersections, camera positions, weather conditions, and traffic environments.

CONCLUSION

This study developed and evaluated a YOLOv8-based vehicle-detection framework integrated with a web-based traffic-density and signal-timing prototype for adaptive traffic-signal applications. Across the three evaluated

training-testing configurations, YOLOv8 achieved mAP@0.5 values of 0.989-0.992 and mAP@0.5:0.95 values of 0.962-0.965. Under the shared 70:30 configuration, YOLOv8 achieved a recall of 0.980 and an inference time of 1.9 ms per image, compared with 0.921 recall and 28.4 ms per image for Faster R-CNN. These findings support YOLOv8 as the more favourable detector for the reported prototype because it provides a strong balance between detection performance and computational efficiency. The evidence, however, supports the vehicle-detection component rather than the effectiveness of a complete adaptive traffic-control system.

The prototype demonstrates that detected vehicles can be converted into vehicle counts, traffic-density outputs, signal-duration estimates, stored records, and dashboard visualisation. Nevertheless, the low-, medium-, and high-density categories and the generated signal timings were not independently validated. A stronger evaluation should define reproducible vehicle-count or occupancy thresholds for each density class, compare the predicted density labels with manually assigned ground truth, and report a confusion matrix together with accuracy, class-level precision, recall, macro-F1, and weighted-F1. The signal-timing policy should also be compared with at least a conventional fixed-time controller, and preferably an established adaptive baseline, using traffic-oriented indicators such as average delay, waiting time, queue length, vehicle throughput, and intersection clearance time. These measurements are required before claiming that the prototype improves traffic flow or reduces congestion.

The reported detection results also require cautious interpretation with respect to generalisation. The 70:30, 80:20, and 90:10 configurations used test sets of different sizes and potentially different compositions, while source-level separation by video sequence, intersection, or camera was not documented. Consequently, the small numerical advantage of the 90:10 split cannot establish that it is conclusively superior, and visually similar frames across training and testing partitions could inflate performance. Future evaluation should therefore retain one fixed, source-separated independent test set containing unseen video sequences or intersections, ensure that adjacent frames do not span training and testing partitions, and use repeated runs with controlled random seeds or repeated cross-validation to estimate variability. Exact precision and F1-score values should be reported together with recall and mAP, with mean performance, standard deviation, and confidence intervals where repeated experiments are performed.

Real-world readiness also requires system-level validation beyond detector inference time. The reported 1.9 ms per-image value represents model inference and does not include video decoding, preprocessing, frame transfer, vehicle counting, density classification, timer generation, database operations, interface rendering, or communication with a traffic controller. End-to-end testing should therefore report effective frames per second, capture-to-output latency, CPU and GPU utilisation, memory consumption, and stability during continuous video processing. Scalability should be examined under multiple concurrent camera streams and intersections, with complete hardware and software specifications reported for reproducibility. Robustness should also be tested across different camera viewpoints, traffic volumes, vehicle scales, daytime and nighttime scenes, rain, glare, reduced visibility, shadows, and partial occlusion.

Within these constraints, the principal contribution of this study is a high-performing and computationally efficient YOLOv8 vehicle detector integrated into a functional traffic-monitoring prototype. The present results establish technical feasibility for connecting computer-vision detections with density and signal-timing outputs, but they do not yet establish validated density classification, optimal signal control, cross-intersection generalisation, or measurable traffic-flow improvement. Addressing the identified validation, reproducibility, deployment, and robustness requirements would provide the evidence needed to progress the current prototype toward a reliable adaptive traffic-monitoring and signal-control system.

REFERENCES

1. Agafonov, A. (2022). Adaptive traffic light control using a distributed processing approach. In 2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) (pp. 186–189). IEEE. https://doi.org/10.1109/SIBIRCON56155.2022.10016941
2. Biradar, V., Chandra Mohanty, S., Kumar Jana, A., & Krishna Goswami, B. (2022). Image processing based intelligent traffic lighting system. In 2022 IEEE International Women in Engineering Conference on Electrical and Computer Engineering (pp. 290–293). IEEE. https://doi.org/10.1109/WIECON-ECE57977.2022.10151401

3. Bouderba, S. I., & Moussa, N. (2019). Reinforcement learning using Q-learning for a traffic light controller within an intersection traffic system. In Proceedings of the International Conference on Intelligent Systems and Computer Vision. Association for Computing Machinery. https://doi.org/10.1145/3372938.3372999
4. Gaouar, N., & Lehsaini, M. (2021). A cloud computing-based intelligent traffic control system for vehicular networks. In Proceedings of the ACM International Conference. Association for Computing Machinery. https://doi.org/10.1145/3454127.3456600
5. Kanungo, A., Sharma, A., & Singla, C. (2014). Smart traffic lights switching and traffic density calculation using video processing. In 2014 Recent Advances in Engineering and Computational Sciences (pp. 1–6). IEEE. https://doi.org/10.1109/RAECS.2014.6799542
6. Kumar, N., Rahman, S. S., & Dhakad, N. (2021). Fuzzy inference-enabled deep reinforcement learning-based traffic light control for intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems, 22(8), 4919–4928. https://doi.org/10.1109/TITS.2020.2984033
7. Minh, Q. T., Tran, C. M., Le, T. A., Nguyen, B. T., Tran, T. M., & Balan, R. K. (2018). FogFly: A traffic light optimization solution based on fog computing. In UbiComp/ISWC 2018: Adjunct proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2018 ACM International Symposium on Wearable Computers (pp. 1130–1139). Association for Computing Machinery. https://doi.org/10.1145/3267305.3274169
8. Mou, A. R., Milanova, M., & Piya, S. S. (2023). Intelligent traffic control system using YOLO algorithm for traffic-congested cities. In Proceedings of the 2023 Congress in Computer Science, Computer Engineering, and Applied Computing (pp. 2014–2019). IEEE. https://doi.org/10.1109/CSCE60160.2023.00331
9. Sankaranarayanan, M., Yerramsetty, S., Kakkera, S., & Kumar, N. (2024). Adaptive genetic algorithm for reducing average waiting time for road traffic signals. In 2024 5th International Conference on Innovative Trends in Information Technology. IEEE. https://doi.org/10.1109/ICITIIT61487.2024.10580688
10. Septyanto, A. W., Rosyida, I., & Suryono, S. (2021). A fuzzy rule-based fog-cloud system for controlling traffic light duration based on road density. In 2021 6th International Conference on Informatics and Computing. IEEE. https://doi.org/10.1109/ICIC54025.2021.9632941
11. Shankaran, S., & Rajendran, L. (2021). Real-time adaptive traffic control system for smart cities. In 2021 International Conference on Computer Communication and Informatics. IEEE. https://doi.org/10.1109/ICCCI50826.2021.9402597
12. Siripatana, B., Nopchanasuphap, K., & Chuai-Aree, S. (2021). Intelligent traffic light system using image processing. In 2021 2nd SEA-STEM International Conference (pp. 14–18). IEEE. https://doi.org/10.1109/SEA-STEM53614.2021.9668057
13. Sirphy, S., & Thanga Revathi, S. (2023). Adaptive traffic control system using YOLO. In 2023 International Conference on Computer Communication and Informatics. IEEE. https://doi.org/10.1109/ICCCI56745.2023.10128619
14. Talukdar, R., Dutta, S., & Das, S. (2023). Enhancing skin disease diagnosis through convolutional neural networks and YOLOv8 object detection. In 2023 7th International Conference on Electronics, Materials Engineering and Nano-Technology. IEEE. https://doi.org/10.1109/IEMENTech60402.2023.10423482
15. Tigga, A., Hota, L., Patel, S., & Kumar, A. (2022). A deep Q-learning-based adaptive traffic light control system for urban safety. In Proceedings of the 2022 4th International Conference on Advances in Computing, Communication Control and Networking (pp. 2430–2435). IEEE. https://doi.org/10.1109/ICAC3N56670.2022.10074123
16. Ultralytics. (2024, January 15). YOLOv8 architecture: Deep dive into its architecture. YOLOv8. https://yolov8.org/yolov8-architecture/#Key_Innovations_in_YOLOv8
17. Xu, Z., Wang, P., Zhang, L., Qi, F., & Wang, G. (2022). Adaptive traffic signal control based on a dueling deep Q-learning network. In Proceedings of the ACM International Conference (pp. 297–301). Association for Computing Machinery. https://doi.org/10.1145/3584376.3584431
18. Zaatouri, K., & Ezzedine, T. (2018). A self-adaptive traffic light control system based on YOLO. In 2018 International Conference on Internet of Things, Embedded Systems and Communications (pp. 16–19). IEEE. https://doi.org/10.1109/IINTEC.2018.8695293