ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-1/370-376
A.Shivani *et. al.*, / International Journal of Engineering & Science Research

Full Length Article

Yolov10-Driven Enhanced Vehicle Detection In Low-Light On-Board Environments

P.Jayaraju[1],A.Shivani[2],G.Pavani[3],J.Akshaya[4]
[1]Assistant Professor ; Department Of Information Technology , Guru Nanak Institutions Technical Campus, Hyderabad, India.
[2,3,4]B.Tech Students; Department Of Information Technology , Guru Nanak Institutions Technical Campus, Hyderabad, India.
Mail Id; adepushivani116@gmail.com[2]
Article Received 26-02-2026, Accepted 24-03-2026

Abstract
Accurate detection of vehicles under low-light conditions remains a significant challenge in intelligent transportation systems and autonomous driving applications. Vision-based onboard systems frequently experience performance degradation due to poor illumination, motion-induced blur, headlight glare, and increased image noise. This paper proposes a real-time vehicle detection framework based on YOLOv10, specifically designed for low-light onboard environments.The proposed approach incorporates image enhancement techniques, including adaptive histogram equalization, denoising, and contrast optimization, to improve visual quality prior to detection. The enhanced images are then processed using a fine-tuned YOLOv10 model trained on a diverse dataset containing nighttime and low-visibility driving scenarios. This enables improved generalization and robustness under challenging lighting conditions.YOLOv10's optimized architecture provides a balance between accuracy and computational efficiency, making it suitable for deployment on resource-constrained embedded systems. Experimental results demonstrate that the proposed framework outperforms baseline models such as YOLOv8 and Faster R-CNN in terms of mean Average Precision (mAP), inference speed, and reduction of false detections.
Furthermore, the system shows strong resilience against common low-light challenges, including shadow interference, artificial light glare, and environmental noise. These results indicate that the proposed method is a reliable and efficient solution for real-time vehicle detection in nighttime and low-illumination driving scenarios.

Keywords:
Low-Light Vehicle Detection, YOLOv10, Object Detection, Computer Vision, Intelligent Transportation Systems, Autonomous Driving, Image Enhancement, Adaptive Histogram Equalization, Denoising, Contrast Optimization, Deep Learning, Real-Time Detection, Embedded Systems, Nighttime Driving, Mean Average Precision (mAP), Inference Speed, YOLOv8 Comparison, Faster R-CNN, Robust Detection, Vision-Based Systems.

Introduction

The rapid evolution of intelligent transportation systems (ITS) and autonomous driving technologies has created a growing demand for reliable vehicle detection under varying environmental conditions. Vision-based detection plays a fundamental role in applications such as traffic surveillance, collision avoidance, lane assistance, and autonomous navigation. Despite significant progress in deep learning-based object detection, achieving consistent performance in low-light and nighttime conditions remains a challenging problem.Low-light environments introduce multiple visual distortions, including inadequate illumination, motion blur, glare from headlights, shadow interference, and increased sensor noise. These factors degrade image quality and adversely impact the performance of conventional detection algorithms, often resulting in missed detections and higher false alarm rates. For onboard vehicle systems, where real-time decision-making is critical, maintaining both accuracy and speed is essential for ensuring operational safety.Recent developments in one-stage object detection models, particularly within the YOLO (You Only Look Once) family, have demonstrated strong capabilities in real-time detection tasks. Among these, YOLOv10 represents a notable advancement, offering improved architectural efficiency, enhanced feature extraction, and reduced inference latency. These characteristics make it well-suited for deployment in embedded and resource-limited onboard systems.This study presents an enhanced vehicle detection framework based on YOLOv10,

specifically designed for low-light onboard environments. The proposed system integrates image enhancement techniques, including adaptive histogram equalization, contrast adjustment, and noise suppression, to improve input quality before detection. Additionally, the YOLOv10 model is fine-tuned using a diverse dataset of nighttime and low-visibility driving scenarios to improve robustness. The overall objective is to achieve high detection accuracy while maintaining real-time performance, making the system suitable for practical ITS and autonomous driving applications.

Scope of the Project

This project focuses on the development and evaluation of a vehicle detection system capable of operating effectively under low-light and nighttime conditions. The system is built using the YOLOv10 architecture, enhanced with preprocessing techniques to improve detection reliability in visually degraded environments.

The scope includes designing a complete detection pipeline that incorporates image enhancement methods such as histogram equalization, denoising, and contrast optimization. The YOLOv10 model is trained and fine-tuned on datasets containing diverse low-illumination scenarios to ensure adaptability to real-world conditions.

Performance evaluation is carried out using standard metrics, including mean Average Precision (mAP), inference time, and false-positive rate. The proposed system is also compared with baseline models such as YOLOv8 and Faster R-CNN to validate its effectiveness.

The system is intended for real-time deployment on embedded and onboard platforms, emphasizing computational efficiency and speed. However, the scope is limited to vehicle detection and does not extend to tracking, behavior prediction, or autonomous decision-making. The framework can serve as a foundation for future advancements in intelligent driving systems.

Objectives

The main objective of this project is to design and implement a vehicle detection system that performs reliably in low-light and nighttime conditions while maintaining real-time efficiency.

Specific objectives include:

- Developing a robust detection framework using YOLOv10
- Enhancing image quality through preprocessing techniques such as contrast enhancement and noise reduction
- Improving detection accuracy under challenging lighting conditions
- Reducing false positives caused by glare, reflections, and noise
- Ensuring efficient computation for deployment on embedded systems
- Achieving a balance between detection performance and processing speed

Existing System

The existing vehicle detection system is primarily based on YOLOv8, a modern object detection model known for its speed and accuracy in standard lighting environments. YOLOv8 utilizes an anchor-free detection mechanism, decoupled prediction heads, and an improved backbone network, enabling efficient detection in real-time applications.

While YOLOv8 performs well under normal conditions, its effectiveness decreases significantly in low-light environments. Poor illumination, glare from headlights, motion blur, and noise reduce image clarity, leading to inaccurate predictions. The model may incorrectly classify reflections or light sources as vehicles and fail to detect small or partially occluded objects.

Although YOLOv8 incorporates multi-scale feature fusion, it struggles to extract fine details in degraded visual conditions. Additionally, lightweight variants improve speed but sacrifice accuracy, whereas larger models require higher computational resources, making them less suitable for onboard systems.

These limitations highlight the need for an improved detection approach that can maintain accuracy and efficiency in challenging lighting scenarios.

Proposed System

The proposed system introduces an advanced vehicle detection framework designed for low-light onboard environments using YOLOv10. The architecture incorporates improved feature extraction, optimized convolutional operations, and multi-scale feature integration to enhance detection performance under challenging conditions.

To address image degradation, preprocessing techniques such as adaptive histogram equalization, contrast enhancement, and noise filtering are applied before detection. These steps improve visibility and highlight important features in low-light images.

The YOLOv10 model is fine-tuned using a dataset containing diverse nighttime scenarios, including urban roads, highways, glare conditions, and shadowed environments. This enables the model to learn robust representations and improve detection of small, distant, and partially occluded vehicles.

The system is designed for real-time inference with low computational overhead, making it suitable for embedded platforms used in intelligent transportation and autonomous driving systems.

PROJECT DESCRIPTION

This project aims to develop a robust and efficient vehicle detection system capable of operating effectively in low-light and nighttime driving

conditions. Such environments present significant challenges for vision-based systems used in intelligent transportation and autonomous vehicles. Factors including insufficient illumination, motion blur, glare from headlights, reduced contrast, and sensor noise often degrade image quality, thereby limiting the performance of conventional object detection models in real-world onboard applications.To overcome these challenges, the proposed system utilizes the YOLOv10 object detection framework, which features an optimized architecture designed for improved feature extraction and efficient multi-scale representation. In addition, an image enhancement stage is incorporated prior to detection. This stage includes techniques such as adaptive histogram equalization, contrast adjustment, and noise filtering to improve image clarity and highlight important visual features in low-light conditions.The YOLOv10 model is further refined using a dataset composed of nighttime driving scenes, low-illumination urban environments, and scenarios affected by glare and shadow interference. This training approach enables the model to learn robust feature representations and improves its ability to detect vehicles that are small, distant, or partially occluded.The system is designed to support real-time inference on resource-limited onboard platforms, ensuring practical applicability in intelligent transportation systems. Experimental analysis indicates that the proposed approach provides improved detection accuracy, faster inference, and reduced false detections when compared with baseline methods such as YOLOv8 and Faster R-CNN. Overall, the project offers a scalable solution for reliable vehicle detection in challenging lighting environments, contributing to enhanced road safety and situational awareness.

Methodology

Techniques and Algorithms

YOLOv8 (You Only Look Once – Version 8)

The existing approach for vehicle detection primarily relies on YOLOv8, a modern deep learning-based object detection model. YOLOv8 employs an anchor-free architecture along with separate classification and regression branches, enabling efficient and accurate detection under standard lighting conditions. Its feature extraction backbone and multi-scale detection capability allow it to perform well in real-time applications.

In typical implementations, images captured from onboard cameras are directly processed by the YOLOv8 model with minimal preprocessing. While this approach is effective in well-lit environments, its performance declines significantly under low-light conditions. Issues such as reduced visibility, motion blur, glare, and noise interfere with feature extraction, resulting in missed detections and increased false positives.

Additionally, deployment challenges arise in resource-constrained environments. Smaller variants of YOLOv8 provide faster inference but at the cost of reduced accuracy, whereas larger models require substantial computational resources, limiting their practicality for onboard systems. These drawbacks highlight the limitations of existing techniques in handling low-light detection scenarios.

Proposed Technique

YOLOv10 (You Only Look Once – Version 10)

The proposed method introduces an enhanced vehicle detection approach based on the YOLOv10 architecture, specifically designed for low-light and nighttime environments. YOLOv10 incorporates an optimized backbone, efficient convolutional structures, and improved feature fusion strategies to achieve better detection accuracy under challenging visual conditions.

A key feature of the proposed approach is the inclusion of an image preprocessing stage prior to detection. Input images are first enhanced using adaptive histogram equalization to improve contrast, followed by noise reduction and glare suppression techniques. These operations enhance the visibility of important structural details, enabling more effective feature extraction.

The enhanced images are then processed using a YOLOv10 model that has been fine-tuned on a dataset containing diverse low-light and nighttime driving scenarios. This allows the model to learn robust representations and improve its ability to detect vehicles under varying conditions.

YOLOv10 utilizes an anchor-free detection strategy along with improved label assignment mechanisms, reducing computational complexity while maintaining high localization accuracy. Its multi-scale feature extraction capability enables the detection of vehicles of different sizes, including small and partially occluded objects.

Due to its lightweight and efficient design, the proposed technique supports real-time inference on embedded platforms, making it suitable for deployment in intelligent transportation systems and autonomous driving applications.

REQUIREMENTS ENGINEERING

Requirements engineering defines the functional and non-functional specifications necessary for developing a reliable vehicle detection system. It establishes a clear understanding of system expectations, operational constraints, and performance goals before implementation.

In this project, the requirements are formulated to support the development of a real-time vehicle detection framework capable of operating effectively under low-light conditions. The system is designed to ensure high detection accuracy, minimal processing delay, and efficient resource utilization. The defined requirements serve as the

foundation for system design, implementation, and evaluation.

Hardware Requirements

The hardware configuration provides the computational resources required for system execution, model training, and real-time inference. The specifications are selected to ensure smooth processing while maintaining cost efficiency.

Minimum Hardware Requirements:

- **Processor** : Dual-Core Processor (or higher)
- **RAM** : 4 GB or above
- **Storage** : 250 GB Hard Disk

These requirements are sufficient for basic implementation and testing. However, higher configurations (e.g., GPU-enabled systems) can significantly improve training speed and detection performance.

Software Requirements

The software environment defines the tools and platforms required for system development, execution, and testing. The system is implemented using widely adopted and flexible technologies to ensure scalability and ease of development.

Software Specifications:

- **Operating System** : Windows 7/8/10 or later
- **Programming Language** : Python
- **Development Environment** : Jupyter Notebook / Spyder
- **Libraries and Frameworks** : OpenCV, NumPy, TensorFlow/PyTorch, YOLOv10

These tools provide strong support for image processing, deep learning model development, and real-time system integration.

Functional Requirements

Functional requirements describe the core operations that the system must perform.

The system is designed to:

- Allow authorized users to access the application through a secure interface
- Accept image input either from onboard cameras or manual upload
- Perform image preprocessing to enhance visibility in low-light conditions
- Detect vehicles using the YOLOv10 model
- Generate bounding boxes and confidence scores for detected vehicles
- Display detection results visually to the user
- Store processed outputs for future reference and analysis

These functions collectively ensure accurate and efficient vehicle detection under challenging environmental conditions.

DESIGN ENGINEERING

Design engineering plays a crucial role in transforming system requirements into a structured representation that guides implementation. It provides a clear blueprint of system components, their interactions, and the overall architecture. In this project, Unified Modeling Language (UML) diagrams are utilized to visually represent system functionality, data flow, and component relationships. The design focuses on modularity, scalability, and real-time processing capability to ensure efficient vehicle detection in low-light environments. Each module is designed to integrate seamlessly, enabling smooth data flow and reliable system performance.

UML Diagram

The system design is represented using various UML diagrams that collectively describe both structural and behavioral aspects of the application. The use case diagram illustrates the interaction between the user and the system, where the user acts as the primary actor performing tasks such as logging in, uploading images, initiating detection, and viewing results. This diagram helps define the functional scope of the system and the roles involved in its operation.

The class diagram represents the static structure of the system by defining key classes, their attributes, and methods. Important classes include the user module, image processing unit, detection model, and result management component. These classes are interconnected to facilitate efficient data handling, processing, and output generation. Complementing this, the object diagram provides a runtime perspective by showing how instances of these classes interact during execution, particularly illustrating the flow of image data through preprocessing, detection, and result generation stages.

The state diagram describes the different operational states of the system, including user authentication, image upload, preprocessing, detection, and result display. It highlights how the system transitions from one state to another based on user actions and internal processing steps. Similarly, the activity diagram outlines the workflow of the system in a step-by-step manner, beginning with image input, followed by validation, preprocessing, detection, and final output generation. It also captures decision points and parallel processes within the system.

The sequence diagram focuses on the time-ordered interaction between system components. It shows how messages are exchanged between the user interface, preprocessing module, detection model, and result module to complete the detection task. In contrast, the collaboration diagram emphasizes the relationships and communication between different objects, illustrating how system components work together to achieve the desired functionality.

The component diagram provides a high-level view of the system architecture by identifying major components such as the user interface, preprocessing module, detection engine, and storage system. It highlights dependencies between these components and their integration within the overall framework. The data flow diagram (DFD) further

explains how data moves through the system. The Level 0 diagram presents a general overview of system interaction with external entities, while the Level 1 diagram offers a detailed representation of internal processes such as image acquisition, preprocessing, detection, and storage.

The deployment diagram defines the physical configuration of the system, showing how software components are mapped onto hardware devices. It illustrates the execution environment, including user systems and processing units where the detection model is deployed. This ensures a clear understanding of how the system operates in real-world conditions.

DEVELOPMENT TOOLS

Python

Python is a high-level, interpreted programming language widely used for software development, data analysis, and artificial intelligence applications. It supports object-oriented, procedural, and functional programming paradigms, making it highly flexible for diverse use cases. Python is designed with an emphasis on code readability and simplicity, using clear syntax and minimalistic constructs compared to many other programming languages. This makes it particularly suitable for rapid development and prototyping, especially in complex domains such as machine learning and computer vision.

History of Python

Python was created by Guido van Rossum during the late 1980s and officially released in the early 1990s. The language was developed at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python was influenced by several earlier programming languages, including ABC, C, C++, and Modula-3, among others. Over time, Python has evolved significantly and is now maintained by a global community of developers under the supervision of the Python Software Foundation. Its open-source nature and continuous development have contributed to its widespread adoption across industries.

Importance of Python

Python plays a vital role in modern software development due to its simplicity, versatility, and efficiency. As an interpreted language, Python executes code directly without requiring a separate compilation step, which accelerates the development process. It also provides an interactive environment that allows developers to test and debug code in real time. Python's support for object-oriented programming enables modular and reusable code design, which is essential for large-scale applications. Additionally, its ease of learning makes it an ideal choice for beginners, while its extensive capabilities make it equally valuable for advanced applications such as deep learning, automation, and web development.

Libraries Used in Python

Python provides a rich ecosystem of libraries that simplify development and enhance functionality. In this project, several key libraries are utilized. NumPy is used for efficient numerical computations and handling multi-dimensional arrays. Pandas provides powerful data structures such as data frames for data manipulation and analysis. Matplotlib is employed for generating graphical visualizations and plotting results. Scikit-learn offers a range of machine learning algorithms and tools for data analysis and preprocessing.

These libraries play a crucial role in implementing the vehicle detection system by enabling efficient data handling, visualization, and model development, thereby improving overall system performance and accuracy.

Figure : NumPy, Pandas, Matplotlib, Scikit-learn

SOFTWARE TESTING

Software testing is a critical phase in the system development lifecycle, aimed at identifying errors and ensuring that the application performs as intended. It involves systematically evaluating software components to verify that they meet specified requirements and function reliably under different conditions. The primary objective of

testing is to detect faults, validate system behavior, and ensure that the final product satisfies user expectations without failure. A structured testing approach is adopted in this project to evaluate both individual modules and the integrated system, ensuring robustness and accuracy in vehicle detection under low-light conditions.

Testing Methodology

The testing process is carried out using a well-defined strategy that includes planning, execution, and validation. A comprehensive test plan is developed to evaluate both general functionality and specialized features of the system across different scenarios. The methodology ensures that the application adheres to the requirements specified in the design phase and performs consistently without errors. Quality control measures are incorporated throughout the testing process to identify defects early and ensure system stability. Each module is tested independently before integration, followed by system-level validation to confirm overall functionality.

Unit Testing

Unit testing focuses on verifying the correctness of individual components within the system. Each module is tested independently to ensure that its internal logic functions correctly and produces expected outputs for given inputs. This type of testing examines decision paths, control flow, and specific functionalities at a granular level. By identifying and resolving errors early, unit testing improves code reliability and simplifies the integration process.

Functional Testing

Functional testing evaluates whether the system performs according to the defined requirements. It verifies that all features operate as expected by testing various input conditions and validating corresponding outputs. Both valid and invalid inputs are considered to ensure proper system behavior. This testing also confirms that all functional components, including preprocessing and detection modules, interact correctly and produce accurate results.Test planning involves dividing the system into smaller units and designing appropriate testing strategies for each component. A structured test plan is created to guide the testing process, including test case design, execution procedures, and evaluation criteria. This approach ensures systematic validation of the system and facilitates the identification and correction of defects at different stages of development.

Future Enhancements

The proposed YOLOv10-based vehicle detection system can be further improved through several advanced enhancements. Future work may incorporate deep learning–based image enhancement techniques, such as illumination normalization and generative adversarial networks (GANs), to improve visibility in extremely low-light conditions. These methods can significantly enhance image quality and enable more accurate detection in challenging environments.

The system can also be extended to include multi-object tracking, allowing continuous monitoring of vehicles across video frames. This would enhance motion analysis and support applications such as traffic flow monitoring and behavior analysis. Additionally, integrating sensor fusion techniques that combine data from cameras, LiDAR, radar, and infrared sensors can improve robustness under adverse weather conditions and poor visibility.

Further optimization can be achieved through model compression techniques such as pruning and quantization, which reduce computational complexity and power consumption. These improvements would facilitate deployment on a wider range of embedded and edge devices. The framework can also be expanded to include vehicle classification and behavior prediction, enabling more advanced functionalities for autonomous driving and intelligent transportation systems.

Conclusion

This work presented an improved vehicle detection framework based on the YOLOv10 architecture, specifically designed to address the challenges associated with low-light and nighttime onboard environments. Detecting vehicles under poor illumination remains a critical issue in intelligent transportation systems, where accuracy and response time directly impact safety. To overcome these limitations, the proposed system integrates image enhancement techniques, including adaptive histogram equalization, contrast adjustment, and noise suppression, which significantly improve image quality prior to detection.

The YOLOv10 model was fine-tuned using a diverse dataset consisting of low-illumination and nighttime driving scenarios. This training strategy enables the model to learn robust feature representations, resulting in improved detection of small, distant, and partially occluded vehicles. In addition, the system effectively reduces false detections caused by glare, reflections, and environmental noise, which are common in nighttime conditions.

Experimental evaluation demonstrates that the proposed approach achieves better performance compared to existing models such as YOLOv8 and traditional detection frameworks. Improvements are observed in key metrics, including detection accuracy, localization precision, and inference speed. Furthermore, the lightweight and efficient design of YOLOv10 allows real-time deployment on resource-constrained embedded systems, making it highly suitable for practical applications in autonomous driving and intelligent transportation.

In summary, the proposed system enhances the reliability and robustness of vehicle detection in challenging lighting conditions. It provides a scalable and efficient solution that can serve as a foundation for future advancements in vision-based perception systems.

References

1. A. Haydari and Y. Yilmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Trans. Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2022.
2. E. Marti et al., "A review of sensor technologies for perception in automated driving," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 94–108, 2019.
3. Z. Lv, R. Lou, and A. K. Singh, "AI-enabled communication systems for intelligent transportation systems," *IEEE Trans. Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4579–4587, 2021.
4. L. Zhu et al., "Big data analytics in intelligent transportation systems: A survey," *IEEE Trans. Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2019.
5. M. De Ryck et al., "Automated guided vehicle systems: State-of-the-art control techniques," *Journal of Manufacturing Systems*, vol. 54, pp. 152–173, 2020.
6. S. Sun et al., "MIMO radar for autonomous driving: Advantages and challenges," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 98–117, 2020.
7. G. Li et al., "Collision avoidance decision-making for autonomous vehicles," *Transportation Research Part C*, vol. 122, 2021.
8. A. Herrero et al., "Pedestrian movement prediction using deep learning," in *Proc. IEEE ITSC*, 2023.
9. S. Ahmed et al., "Pedestrian and cyclist detection: A survey," *Applied Sciences*, vol. 9, no. 11, 2019.
10. G. Delgado et al., "Multi-object tracking validation for automotive systems," in *Proc. IEEE IVMSP*, 2022.
11. X. Hu et al., "SINet: Scale-insensitive CNN for vehicle detection," *IEEE Trans. Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1010–1019, 2019.
12. H. Wang et al., "Vehicle detection using LiDAR and vision fusion," *Journal of Sensors*, 2019.
13. O. Elharrouss et al., "Video surveillance systems: A review," *Journal of Visual Communication*, 2021.
14. Z. Charouh et al., "Efficient CNN-based vehicle detection," *Sensors*, vol. 22, 2022.
15. Y. Guo and M. Zhao, "Nighttime vehicle detection using improved Faster R-CNN," 2021.
16. X. Zhang et al., "Nighttime vehicle detection using multi-camera fusion," *IEEE Trans. ITS*, 2022.
17. T.-Y. Lin et al., "Focal loss for dense object detection," *IEEE TPAMI*, vol. 42, no. 2, pp. 318–327, 2020.
18. S. Ren et al., "Faster R-CNN: Real-time object detection," *IEEE TPAMI*, vol. 39, no. 6, pp. 1137–1149, 2017.
19. F. Yu et al., "BDD100K: A large-scale driving dataset," in *Proc. CVPR*, 2020.
20. I. Urbieta et al., "WebLabel: Multi-sensor labeling framework," *Multimedia Tools and Applications*, 2023.