International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Edge-Ready Road Damage Detection Using An Enhanced YOLO With Hyperparameter Tuning

Mr.N.S.R.K Prasad[1],B. Ankitha[2],D. Harika[3],G. Akash[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;Ankithareddy2870@Gmail.Com[2]

Abstract

Maintaining high-quality road infrastructure is essential for urban safety, efficient traffic flow, and reduced vehicle maintenance costs. Traditional manual inspections are time-intensive, costly, and prone to human error, highlighting the need for automated monitoring systems. Recent advances in deep learning and computer vision have made real-time road damage detection feasible. This study presents a lightweight road damage detection framework based on the YOLOv10n model, optimized for deployment on edge devices. The system integrates hyperparameter tuning to balance accuracy and computational efficiency. Experimental results show the framework achieves a precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and an F1-score of 0.978 for detecting cracks, potholes, and surface wear. Deployment on NVIDIA Jetson Nano achieved 7.5 FPS, and on NVIDIA AGX Orin, 67 FPS. These findings demonstrate that the proposed framework is suitable for scalable, real-time road monitoring, supporting proactive maintenance strategies and enhancing urban transportation safety.

Introduction

Road infrastructure is a critical component of modern transportation systems, directly impacting safety, efficiency, and overall driving experience. Well-maintained roads minimize accidents, improve comfort, and reduce vehicle operating costs. However, road surfaces inevitably deteriorate over time due to factors such as heavy traffic, adverse weather conditions, and natural material aging. Unaddressed damages like cracks, potholes, and surface wear can accelerate pavement degradation, increase maintenance costs, and pose hazards to road users. Consequently, timely and accurate detection of road damage is essential for the development of smart cities and intelligent transportation systems.

Traditionally, road inspections rely on manual surveys conducted by experts or municipal authorities. While effective on small scales, such inspections are labor-intensive, costly, and often subject to human error. With urban expansion and increasing road networks, there is a pressing need for automated solutions. Advances in computer vision and deep learning offer promising alternatives, enabling automated detection of road defects. However, many existing deep learning methods demand substantial computational resources, limiting their feasibility for real-time deployment on edge devices. This highlights the necessity for lightweight, high-performance models that balance accuracy and efficiency.

To address this need, this study proposes a real-time road damage detection framework based on **YOLOv10n**, optimized for deployment on edge devices such as NVIDIA Jetson Nano and NVIDIA AGX Orin. The framework employs advanced hyperparameter tuning and model optimization techniques to achieve high detection accuracy while minimizing computational overhead. Experimental evaluation demonstrates that the proposed model achieves a precision of 0.986, recall of 0.973, mAP@0.5 of 0.988, and an F1-score of 0.978. Furthermore, inference speeds of 7.5 FPS on Jetson Nano and 67 FPS on AGX Orin validate its suitability for scalable, real-time applications. This system supports proactive road maintenance, reduces reliance on manual inspections, and enhances overall road safety.

Scope of the Project

The primary focus of this project is the development of an efficient, accurate, and lightweight road damage detection system capable of real-time operation on edge devices. Specifically, the YOLOv10n-based framework is designed to detect various types of road surface defects, including cracks, potholes, and general surface wear. The project emphasizes achieving a balance between detection performance and computational efficiency, enabling large-scale deployment in smart city environments.

The framework has been validated on both low-power (NVIDIA Jetson Nano) and high-performance (NVIDIA AGX Orin) devices, demonstrating adaptability across diverse hardware platforms. The system facilitates proactive maintenance strategies, minimizes manual inspection efforts, and ensures safer transportation infrastructures. It is important to note that this study focuses solely on visual surface damage detection and does not address subsurface structural issues or

predictive maintenance, which could be explored in future research.

Objectives

The key objectives of this project are as follows:

1. Design and implement an automated road damage detection system leveraging **YOLOv10n**, a lightweight yet high-performing object detection model suitable for edge deployment.
2. Optimize model hyperparameters to maximize detection accuracy while minimizing computational costs.
3. Detect and classify different types of road damages, such as cracks, potholes, and surface deterioration, with high precision and reliability.
4. Evaluate system performance on edge devices like NVIDIA Jetson Nano and NVIDIA AGX Orin to demonstrate scalability and real-time capabilities.
5. Provide a practical, cost-effective solution for continuous road monitoring, supporting smart city initiatives and improving overall road safety.

Existing System

Currently, automated road damage detection relies heavily on **YOLOv7**, a state-of-the-art object detection model. YOLOv7 improved upon previous versions such as YOLOv5 by introducing advanced training strategies, optimized anchor-free mechanisms, and enhanced architectural designs. It effectively detects road defects, including cracks, potholes, and surface wear, using images or video feeds from roadside cameras, vehicles, or drones.

While YOLOv7 demonstrates high accuracy in controlled environments with powerful computing resources, its deployment on edge devices faces significant challenges. Its complex architecture results in high computational requirements, slower inference speeds, and longer training times, making it unsuitable for real-time applications in resource-constrained scenarios. Additionally, lightweight YOLOv7 variants often sacrifice detection accuracy, particularly for small-scale damages, limiting their scalability for continuous monitoring in smart cities.

Limitations of Existing Systems

- High computational requirements, preventing real-time deployment on low-power edge devices.
- Slower inference speeds on resource-constrained hardware.
- Increased training complexity requiring significant computational resources and fine-tuning.
- Trade-offs between model size and accuracy, particularly affecting the detection of small damages.
- Limited adaptability for large-scale, continuous road monitoring applications.

Literature Survey

1. **Automatic Defect Detection of Pavement Diseases** (L. Zhao et al., 2022): Proposed DASNet, a deep convolutional neural network that employs deformable convolutions and residual feature enhancement to automatically detect road damages, addressing irregular shapes and foreground-background imbalances.
2. **Real-Time Road Defect Monitoring from UAV Visual Data Sources** (I. Katsamenis et al., 2023): Explored UAV-based monitoring using high-resolution imagery and deep learning algorithms to enable efficient, safe, and cost-effective road inspections.
3. **Road Damage Detection and Classification Using Smartphone Sensor Data** (A. Basak et al., 2023): Created a large-scale road damage dataset and demonstrated high-accuracy detection using CNN-based object detection models, supporting real-time deployment on smartphones and servers.
4. **Digital Twin: Challenge Road Damage Detection on Edge Device** (H. Mahmudah et al., 2024): Discussed the application of digital twin technology in real-time fault monitoring, enabling predictive maintenance and system health monitoring across industries, including transportation.
5. **YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors** (C. Wang et al., 2023): Demonstrated YOLOv7's superior speed and accuracy compared to other real-time detectors and transformer-based models, highlighting its effectiveness for object detection tasks.

Proposed System

This study introduces a **YOLOv10n-based framework** for real-time road damage detection, specifically designed for edge device deployment. YOLOv10n combines lightweight architecture with high detection accuracy, achieving efficient inference without compromising performance. Hyperparameter tuning and model optimization further enhance the balance between computational efficiency and precision.

The framework processes road images or live video feeds to detect damages such as potholes, cracks, and surface wear. Its lightweight design allows fast inference on devices like NVIDIA Jetson Nano (7.5 FPS) and NVIDIA AGX Orin (67 FPS), demonstrating both scalability and robustness. The system facilitates proactive road maintenance, reduces reliance on manual inspections, and contributes to safer urban transportation systems.

Advantages of Proposed System

- Lightweight architecture for faster inference and lower hardware resource usage.
- High detection accuracy: precision 0.986, recall 0.973, mAP@0.5 0.988.
- Real-time deployment on edge devices, supporting various hardware capacities.
- Reduced training complexity compared to heavier models.

- Scalable for large-scale smart city road monitoring, enabling proactive maintenance and improved road safety.

PROJECT DESCRIPTION

This project focuses on developing a **lightweight and efficient framework** for real-time road damage detection using the **YOLOv10n** model. Road surface defects such as potholes, cracks, and general wear pose significant risks to traffic safety, increase vehicle maintenance costs, and negatively impact transportation efficiency. Traditional road inspection methods rely heavily on manual surveys, which are time-consuming, expensive, and prone to human error. Such approaches are inadequate for large-scale monitoring, particularly in smart city environments where continuous and timely assessment of road conditions is essential. To address these limitations, this project leverages **deep learning and computer vision techniques** to automate the detection process, enabling faster and more reliable evaluation of road conditions.

The system utilizes **YOLOv10n**, a state-of-the-art lightweight object detection model, which offers a strong balance between computational efficiency and high accuracy. Unlike heavier models such as YOLOv7, YOLOv10n is specifically optimized for **edge device deployment**, ensuring high performance on low-power hardware while maintaining real-time capabilities. Through hyperparameter tuning and model optimization, the framework achieves a precision of 0.986, recall of 0.973, and mean average precision (mAP@0.5) of 0.988. This allows the model to accurately detect different types of road damages without requiring high-end computing resources. The framework has been validated on **NVIDIA Jetson Nano** and **NVIDIA AGX Orin**, achieving inference speeds of 7.5 FPS and 67 FPS, respectively, which demonstrates its scalability and practical applicability in real-world scenarios. By enabling proactive maintenance strategies, the project reduces road hazards, improves traffic safety, and supports the development of intelligent urban infrastructure. Overall, the system provides a cost-effective, reliable, and scalable solution for **smart city road monitoring**.

The methodology of the project is implemented through a **modular approach**, where each module performs a specific function, forming a complete pipeline for efficient and accurate detection of road defects. The data collection and preprocessing module gathers road images from various sources, including public datasets, roadside cameras, and drone footage. This data is cleaned, annotated, and augmented to improve diversity, and preprocessing steps such as resizing, normalization, and noise reduction are applied to ensure compatibility with YOLOv10n training requirements. The model selection module focuses on choosing YOLOv10n due to its lightweight architecture and ability to deliver high accuracy with low computational cost. Hyperparameters are tuned to optimize the balance between precision, recall, and inference speed. During the training and validation phase, the model is trained on the prepared dataset, with weights and loss functions optimized while a validation set monitors performance and prevents overfitting. Once trained, the model is deployed on edge devices, where inference time and frame rates are measured to ensure real-time efficiency. Finally, the road damage detection and output generation module allows the deployed model to process live video feeds or static images, identifying and classifying road defects such as cracks and potholes. The results are visualized with bounding boxes and labels, which can be integrated into smart city dashboards to assist decision-making.

The existing system for road damage detection primarily relies on **YOLOv7**, a real-time object detection model. YOLOv7 employs a single-stage detection pipeline, performing both classification and localization in one forward pass. It incorporates **Extended Efficient Layer Aggregation Networks (E-ELAN)** to improve feature extraction and gradient flow, enhancing accuracy without excessive computational cost. Additionally, it uses both anchor-free and anchor-based detection heads, improving detection of objects of varying sizes, including small damages like fine cracks. Advanced training techniques such as coarse-to-fine lead loss and model re-parameterization further improve convergence, reduce overfitting, and optimize inference efficiency. Despite these strengths, YOLOv7 is computationally heavy, making it unsuitable for real-time deployment on low-power edge devices.

To overcome these limitations, the proposed system adopts **YOLOv10n**, a lightweight variant of the YOLOv10 family designed for edge deployment and real-time applications. Like other YOLO models, YOLOv10n is a single-stage detector, simultaneously performing classification and localization in a single forward pass. It introduces task-aligned learning and a decoupled head architecture to improve the precision of object detection, particularly for small-scale damages. The anchor-free mechanism reduces the complexity of anchor box design, enabling detection of varied shapes and sizes of road damages. Advanced loss functions, including **Distribution Focal Loss (DFL)** and **VariFocal Loss**, ensure a balance between precision and recall, resulting in robust predictions. The “Nano” architecture reduces model size and computational complexity, allowing smooth deployment on low-power edge devices without compromising accuracy. In this project, YOLOv10n achieves excellent performance metrics, including a precision of 0.986, recall of 0.973, mAP@0.5 of 0.988, and F1-score of 0.978.

Inference speeds of 7.5 FPS on Jetson Nano and 67 FPS on AGX Orin demonstrate its suitability for real-time applications. Compared to YOLOv7, YOLOv10n offers an optimal trade-off between detection accuracy and computational efficiency, making it highly suitable for **large-scale smart city road monitoring systems**.

REQUIREMENTS ENGINEERING

The proposed road damage detection system is designed to deliver high accuracy and reliability, as evidenced by low error rates across different datasets. This performance is attributed to the strong discriminatory power of the features extracted by the YOLOv10n model and the robust regression capabilities of its classifiers. When compared to previous approaches, the system demonstrates competitive performance, achieving superior detection precision and recall while maintaining real-time efficiency.

The **hardware requirements** for the system are specified to ensure smooth operation and serve as a foundational guideline for implementation. The system is designed to run on a dual-core 2 Duo processor, supported by 4 GB of DDR RAM and a 250 GB hard disk. These specifications ensure that both the model training and real-time deployment on edge devices can be executed efficiently without compromising performance.

From a **software perspective**, the system leverages Python as the primary programming language, implemented within the Spyder IDE environment. The operating system can be Windows 7, 8, or 10, providing flexibility in development and deployment. This software setup allows efficient coding, debugging, and testing, while the front-end interface is also built using Spyder3, ensuring seamless integration with the model and facilitating visualization of detection results.

The **functional requirements** of the system define its core capabilities, focusing on automated road damage detection and real-time analysis. The system collects and preprocesses road images, trains the YOLOv10n model, and deploys it on edge devices to identify cracks, potholes, and surface wear. It then generates output with bounding boxes and labels, which can be integrated into smart city dashboards for proactive maintenance planning. By automating these processes, the system achieves minimal user intervention while ensuring precise and reliable road monitoring.

The **non-functional requirements** highlight additional qualities that ensure system robustness and usability. In terms of **usability**, the system is designed to operate automatically, reducing the need for human intervention. Its **reliability** is reinforced by Python's stable and well-tested environment, ensuring that the system performs consistently under different conditions. **Performance** is enhanced through the use of high-level programming languages and advanced back-end techniques, enabling rapid response times for end users. The system also emphasizes **supportability**, as it is cross-platform compatible and can run on a wide range of hardware and software environments. The implementation utilizes a web-based environment with Jupyter Notebook serving as the interface, while the server manages intelligence operations. Windows 10 Professional is employed as the underlying platform, providing a stable environment for seamless interaction between the user and the deployed detection system.

Design Engineering

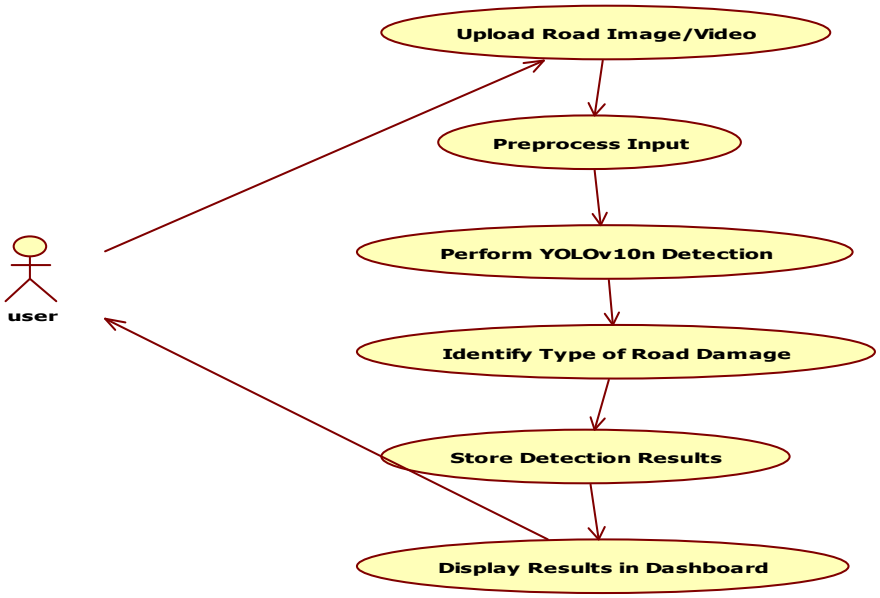
Design engineering in software development involves translating system requirements into meaningful representations that guide implementation. It plays a crucial role in ensuring software quality by defining how components interact, behave, and achieve intended functionalities. This process relies heavily on Unified Modeling Language (UML) diagrams to visualize, document, and plan the software system before development. Through design engineering, abstract requirements are converted into structured models that form the blueprint for building the software, ensuring that all functional and non-functional aspects are addressed systematically.

UML Diagrams

UML diagrams provide a structured approach to represent the components, interactions, and workflows of a software system. They enable developers to model system behavior, structure, and data flow clearly.

USE CASE DIAGRAM

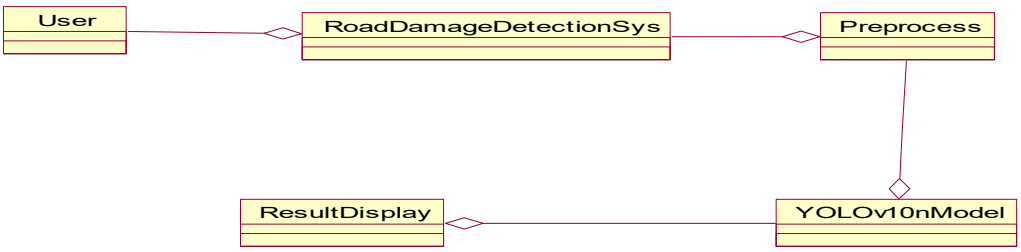
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



use case diagram illustrates the functional requirements of the system, highlighting the roles of actors, such as users, and the interactions they have with system functionalities. It provides an overview of the tasks that the system performs and the responsibilities of each actor in achieving the system's objectives.

The **class diagram** models the classes in the system along with their attributes and methods, demonstrating how they interact to achieve verification and security tasks. It serves as a structural blueprint for understanding relationships and responsibilities across system components. Complementing

OBJECT DIAGRAM

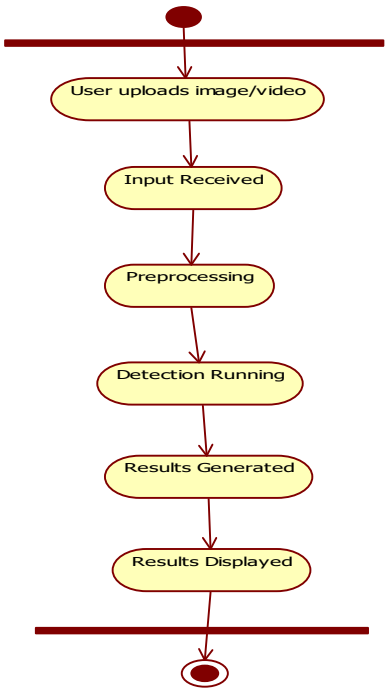


Object diagram depicts the flow of objects between these classes, showing partial or complete system structures and their dynamic interactions during execution.

State diagrams are used to represent the different states of system components and transitions between these states. They support modeling workflows involving sequential steps, choices, iterations, and concurrency, offering a simplified abstraction of system behavior.

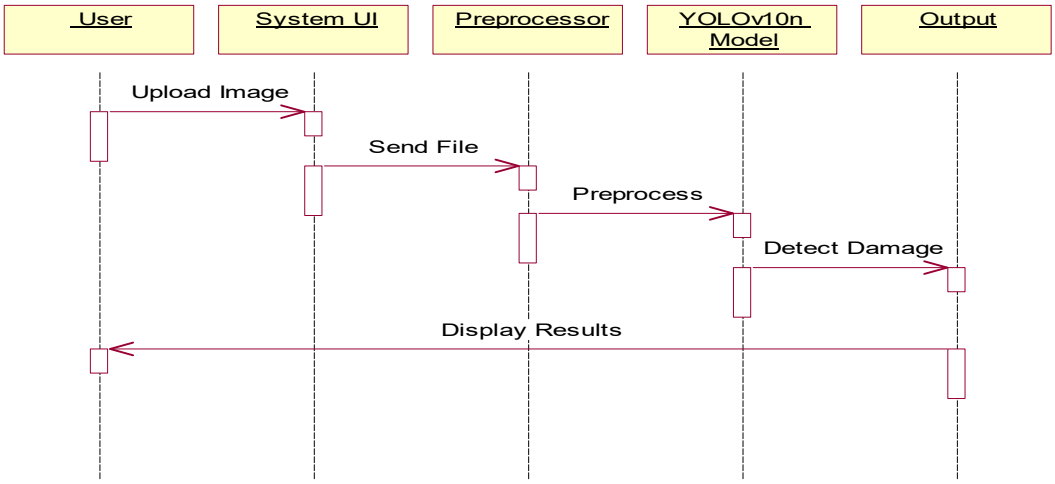
ACTIVITY DIAGRAM

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



Activity diagrams provide a graphical representation of workflows, detailing step-by-step operational processes, decision points, and the overall flow of control within the system.

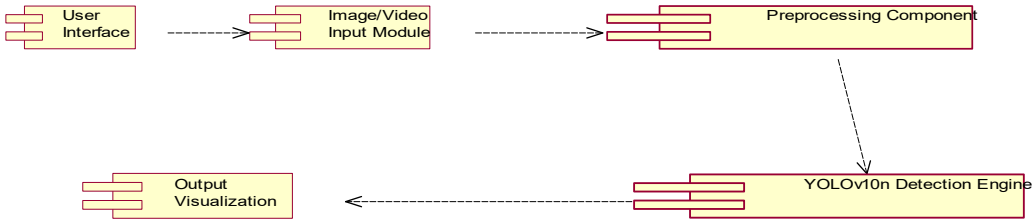
SEQUENCE DIAGRAM



The **sequence diagram** focuses on temporal interactions, illustrating how objects communicate over time through message exchanges to carry out system functionalities. The **collaboration diagram**, also known as a communication diagram, emphasizes the relationships and interactions between objects, providing insight into how components collaborate to execute processes.

COMPONENT DIAGRAM

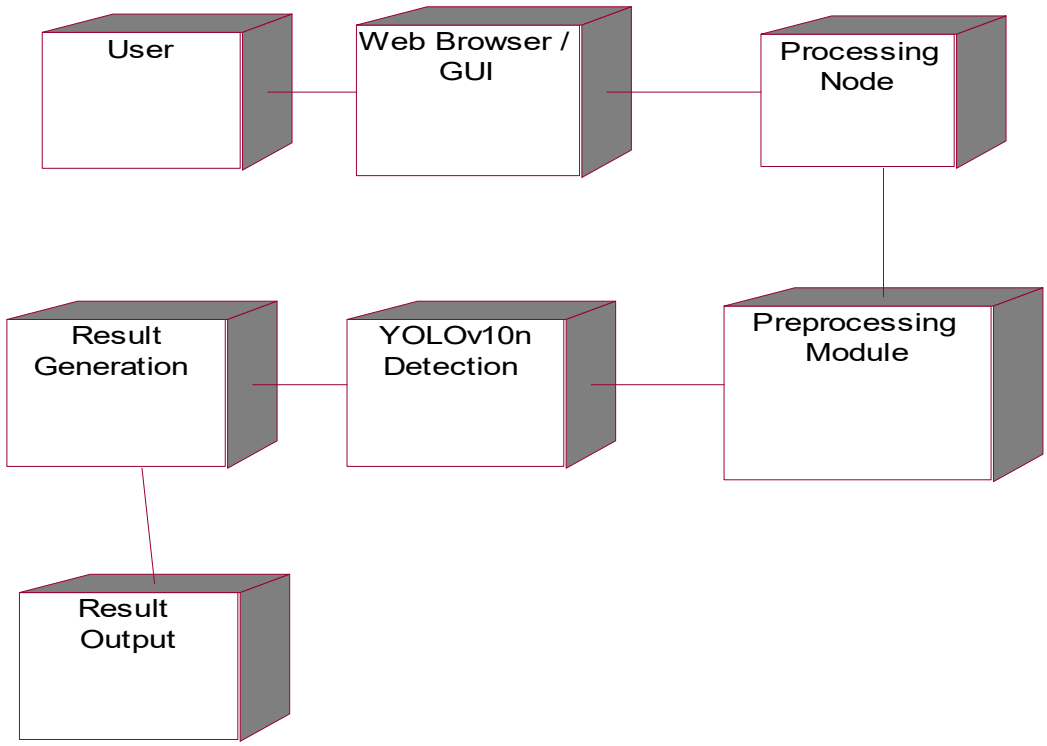
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



The **component diagram** depicts the modular structure of the system, showing how components are wired together to form larger systems. It illustrates dependencies between components, where user queries are broken into sub-queries, processed by data aggregators, and returned as results.

Data flow diagrams (DFDs), at Level 0 and Level 1, model the flow of data through the system. They represent input sources, data storage, processing points, and outputs, offering a visual understanding of how information moves and transforms within the system. Unlike other diagrams, DFDs focus on data movement rather than timing or sequence of operations.

DEPLOYMENT DIAGRAM



Deployment diagram maps software components onto physical hardware, specifying where and how each software module will execute. It ensures that the system is deployable on the chosen hardware infrastructure, supporting efficient real-time processing.

The overall **system architecture** integrates all these design elements into a cohesive framework. The architecture supports processing of road images or live video streams through YOLOv10n, performing real-time road damage detection, and delivering results via an intelligent interface for monitoring and maintenance decision-making. By combining UML-based design with a structured deployment strategy, the system achieves scalability, reliability, and ease of maintenance, making it suitable for real-world smart city applications.

Software Testing

Software testing is a critical phase of the development lifecycle, aimed at identifying and

rectifying errors to ensure system reliability. It is a structured process of examining a software system to verify that it meets functional and non-functional requirements while ensuring that it does not fail under unexpected conditions. Testing evaluates individual components, subsystems, and the integrated system, providing confidence that the software behaves as intended. Various types of testing are employed to address different aspects of system quality, including functionality, performance, reliability, and user satisfaction.

Developing Methodologies

The testing process begins with the creation of a comprehensive plan that outlines procedures for verifying system functionality across multiple platforms. This plan ensures that all features, both general and specialized, are rigorously examined under controlled conditions. Strict quality control measures are implemented to confirm that the software aligns with the specifications documented in the system requirements. Methodologies are developed to ensure systematic testing, addressing potential weaknesses and minimizing the risk of undetected errors before deployment.

Types of Tests

Unit Testing: Unit testing focuses on validating the internal logic of individual software components. Test cases are designed to ensure that each module performs as expected, producing correct outputs for all valid inputs and handling invalid inputs appropriately. This form of testing is performed after the completion of a unit and before integration, enabling developers to identify and correct errors at the component level. Unit testing is structural in nature, often invasive, and ensures that all decision branches and code paths are verified.

Functional Testing: Functional testing verifies that the software meets specified business and technical requirements. It ensures that identified functions operate correctly, valid inputs are accepted, invalid inputs are rejected, and the expected outputs are produced. Additionally, functional tests examine the interaction of the software with external systems or procedures, ensuring that integrated operations are performed as intended.

System Testing: System testing evaluates the behavior of the complete, integrated software system. It validates that all components function together as intended, producing consistent and predictable results. System testing emphasizes the integration of processes and workflows, ensuring that the system operates correctly in the intended configuration.

Performance Testing: Performance testing measures the responsiveness and efficiency of the system. It ensures that outputs are generated within acceptable time limits and that the system can handle user requests promptly. This type of testing is crucial for verifying that the application performs adequately under operational conditions and meets performance benchmarks.

Integration Testing: Integration testing assesses the interaction between multiple software components or applications. It identifies defects arising from interface incompatibilities, ensuring that individual modules function correctly when combined. This incremental testing approach helps detect errors at the integration level before full system deployment.

Acceptance Testing: User Acceptance Testing (UAT) is conducted to verify that the system meets the functional and operational expectations of end users. In the context of data synchronization, acceptance testing ensures that acknowledgments are received after data transmission, route operations are executed only when requested, and node status updates occur automatically in the cache. UAT is essential for validating real-world applicability and user satisfaction.

Test Planning

Effective testing requires a structured plan that breaks the project into units and defines testing strategies for each component. Unit testing helps identify bugs at the earliest stage, allowing developers to rectify errors before integration. By systematically testing each module, the overall reliability, performance, and correctness of the software system are ensured, contributing to a high-quality, robust application suitable for deployment.

Future Enhancements

Although the proposed YOLOv10n-based framework has demonstrated high accuracy and efficiency for real-time road damage detection, several avenues exist to further enhance its capabilities. One promising improvement is **severity classification**, where the system not only detects the presence of damage but also estimates its depth and size, enabling authorities to prioritize repair efforts more effectively.

Another enhancement involves **multi-sensor data fusion**, integrating visual information with LiDAR, thermal imaging, or IoT-enabled road sensors. This approach can improve detection accuracy under challenging conditions such as low light, heavy rainfall, or dense traffic. Additionally, leveraging **predictive analytics** with historical data and machine learning models would allow the system to forecast potential road failures and implement preventive maintenance strategies.

The deployment of the framework in a **cloud-edge hybrid environment** is another viable direction. Edge devices could handle critical real-time detection tasks locally, while the cloud manages large-scale data processing, analytics, and model updates. This ensures both low-latency responses and scalable data management. Furthermore, adopting **transfer learning** and **continual learning** strategies would enable the system to adapt to new

types of road damage and changing environmental conditions without full retraining, enhancing long-term robustness.

Implementing these enhancements would make the system more intelligent, adaptive, and practical for large-scale smart city applications, ultimately improving road safety, reducing maintenance costs, and supporting sustainable urban mobility.

Conclusion

Efficient road maintenance is critical for ensuring transportation safety and supporting the growth of smart cities. Traditional manual inspection methods, while reliable on a small scale, are labor-intensive, time-consuming, and prone to human error, limiting their suitability for widespread monitoring. This study addressed these limitations by developing an automated road damage detection system using the **YOLOv10n algorithm**, which is lightweight, efficient, and optimized for real-time edge deployment.

The proposed system demonstrates high accuracy while maintaining low computational requirements, outperforming earlier models such as YOLOv7. Experimental results show a **precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and an F1-score of 0.978**, effectively identifying potholes, cracks, and surface wear. Deployment on **NVIDIA Jetson Nano** and **NVIDIA AGX Orin** further validates its adaptability, achieving real-time inference speeds of 7.5 FPS and 67 FPS, respectively.

Beyond performance metrics, the system contributes to intelligent transportation systems by enabling **proactive maintenance planning**, reducing reliance on manual inspections, and providing actionable insights into road conditions. These capabilities can decrease accidents, reduce vehicle maintenance costs, and enhance the overall reliability of urban transport networks.

Future work may focus on integrating predictive analytics, IoT-based sensors, or satellite imagery to expand system intelligence. Additionally, incorporating **damage severity classification** and repair prioritization would further enhance utility for municipal authorities. In summary, the YOLOv10n-based framework offers a **scalable, efficient, and accurate solution** for real-time road damage detection, providing a strong foundation for safer, smarter, and more sustainable urban transportation infrastructures.

References

1. T. Qiu et al., “Edge computing in industrial Internet of Things: Architecture, advances and challenges,” *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 2020.
2. H. Mahmudah et al., “Digital twin: Challenge road damage detection on edge device,” *Chem. Eng. Trans.*, vol. 109, pp. 601–606, 2024.
3. L. Zhao et al., “Automatic defect detection of pavement diseases,” *Remote Sens.*, vol. 14, no. 19, p. 4836, 2022.
4. I. Katsamenis et al., “Real-time road defect monitoring from UAV visual data sources,” Proc. *16th Int. Conf. Pervasive Technol.*, pp. 603–609, 2023.
5. M. Rathee et al., “Automated road defect and anomaly detection for traffic safety: A systematic review,” *Sensors*, vol. 23, no. 12, p. 5656, 2023.
6. B. Setiadji et al., “Surface distress index updates to improve crack damage evaluation,” Proc. *11th Asia–Pacific Transp. Environ. Conf.*, pp. 274–281, 2019.
7. W. Nur et al., “Relationship between PCI, PSI, and SDI on Soekarno Hatta Road, Bandung,” *J. Teknik Sipil*, vol. 26, no. 2, pp. 111–120, 2019.
8. M. Surbakti et al., “Evaluation of road maintenance program based on IRI and SDI,” Proc. *6th Int. Conf. Civil, Offshore Environ. Eng.*, pp. 764–771, 2020.