iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

Real-Time Pothole Detection and Mapping System Using YOLOv8n and Edge Devices

Sanskruti Suryakant Trimukhe[1], Siddhi Sachin Soni[2], Tanish Swaroop Todkar[3], Yash Zumbarsing Suryawanshi[4], Amol Jagtap[5]
Department Of Computer Engineering, AISSMS College of Engineering, Pune, India

Abstract: ***Poor road conditions caused by potholes lead to increased accident risk, vehicle damage, and higher maintenance costs, yet many urban regions still rely on manual inspections or public complaints for detection. This paper presents a low-cost, real-time pothole detection and mapping system implemented using the YOLOv8n object detection model and deployed on a Raspberry Pi 4 platform. The system processes live road video from a Pi Camera, detects potholes using deep learning, and attaches GPS coordinates obtained from a Neo6M module. Detected frames are stored locally on an SD card while metadata such as timestamp, image name, pothole count, and geolocation are logged in a CSV file for later analysis. A Flask-based web dashboard visualizes the pothole locations on an interactive Leaflet.js map and groups nearby detections within a 150-meter radius into clusters to indicate low, medium, or high-severity road segments. The model was trained on a dataset of approximately 5000–6000 annotated images, augmented with rotation, flipping, brightness adjustment, and scaling, and evaluated using precision (0.832), recall (0.631), F1-score (0.718), mAP@0.5 (0.723), and mAP@0.5:0.95 (0.366). During deployment, the system achieved about 15 FPS for camera capture and 3.3 FPS for detection after ONNX optimization. The framework demonstrates that embedded AI can support practical, scalable road-condition monitoring suitable for municipal and smart-city applications.***

Keywords: pothole detection, deep learning, YOLO, Raspberry Pi, real-time mapping, road maintenance.

I. INTRODUCTION

Pothole formation poses serious threats to urban and rural transport infrastructure, as potholes increase the risk of accidents, vehicle damage, and higher maintenance costs for municipal authorities. In many regions, pothole detection relies on manual inspection or citizen complaints, which often leads to delays and prolonged exposure to hazardous road conditions. A continuous monitoring system enables administrators to identify defects early and plan repairs more effectively.

Recent advances in computer vision and deep learning have made real-time object detection feasible even on low-power embedded platforms. YOLO-based models, especially lightweight variants such as YOLOv8n, are widely used for edge applications because they balance detection speed and accuracy well. When combined with a camera, a GPS module, and a simple web dashboard, such a detector transforms a basic prototype into a practical road-monitoring tool.

This paper presents a real-time pothole detection and mapping system built around the YOLOv8n model and deployed on a Raspberry Pi 4 platform. The system captures live road footage from a Pi Camera, performs pothole detection using deep learning, annotates detections with GPS coordinates from a Neo6M module, stores relevant images and metadata locally, and visualizes pothole locations on a web dashboard. The framework is designed to be low-cost, scalable, and suitable for deployment in municipal or campus-level road networks.

II. PROBLEM STATEMENT

Urban roads often develop potholes that are difficult to monitor continuously through manual inspection, and delays in reporting can lead to vehicle damage, traffic disruption, and safety risks. There is a need for a low-cost, automated system that can detect potholes in real time, geotag their locations accurately, store the data locally, and present the results in a usable map-based interface for maintenance planning.

In this project, the problem is to design and implement an edge-based pothole detection and monitoring system that works reliably on limited hardware such as a Raspberry Pi 4. The system should identify potholes from live road video, capture GPS coordinates at the time of detection, log the information efficiently, and visualize pothole density and severity so that road authorities can prioritize repair work effectively.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

III. LITERATURE REVIEW

Pothole detection has become an important research area due to its impact on road safety and infrastructure maintenance. Traditional methods mainly relied on manual inspection and citizen complaints, which were time-consuming and inefficient. Early automated systems used sensors such as accelerometers and ultrasonic sensors to detect road irregularities, but their accuracy was affected by vehicle movement and environmental noise.

With advancements in computer vision, researchers introduced vision-based techniques using image processing methods like edge detection and texture analysis. Although these methods improved automation, they struggled with shadows, lighting variations, and complex road textures, leading to inaccurate results.

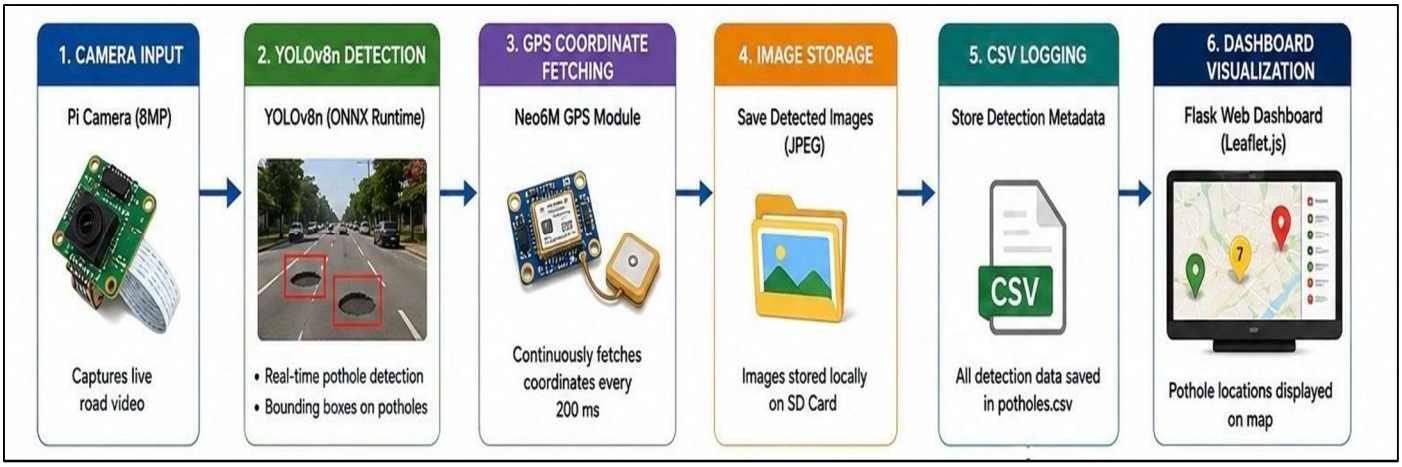
Recent studies focus on deep learning models such as CNN and YOLO for real-time pothole detection. These models provide higher accuracy and faster detection compared to traditional approaches. Researchers have also integrated these models with edge devices like Raspberry Pi and Jetson Nano to enable real-time processing and GPS-based pothole mapping.

Several studies highlight challenges such as varying weather conditions, limited datasets, and hardware constraints on edge devices. Researchers suggest improving dataset diversity, optimizing lightweight models, and integrating detection systems with GIS and smart city platforms for better road maintenance and traffic safety.

IV. PROPOSED METHODOLOGY

A. *System Workflow*

Figure 1: System Workflow Diagram



The system is a real-time edge AI solution for automated pothole detection and mapping, implemented on a Raspberry Pi 4. An 8 MP Pi Camera continuously captures road frames while the vehicle is in motion. Frames are processed locally by a YOLOv8n detection model running with ONNX Runtime; detections are returned as bounding boxes and pothole counts without streaming video to the cloud, preserving privacy and reducing bandwidth.

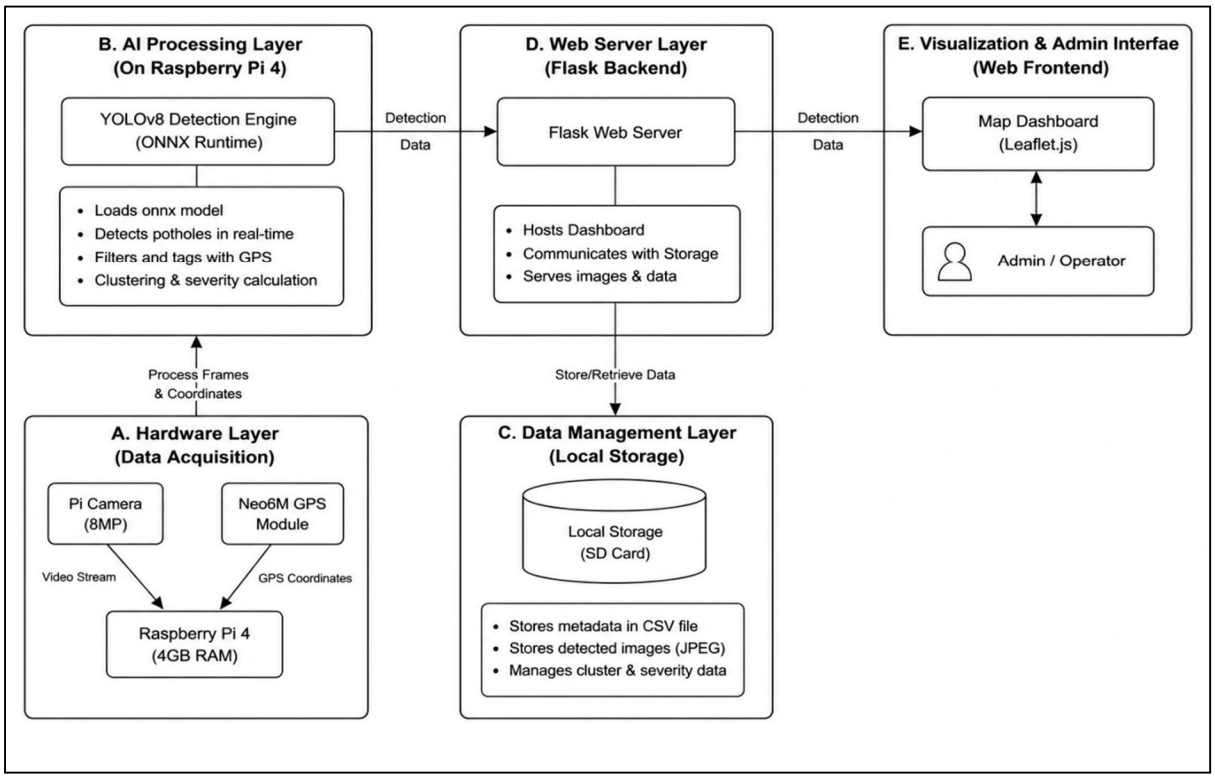
When a pothole is detected, the system uses a Neo6M GPS module—polled continuously at 200 ms intervals—to attach accurate coordinates and minimize timestamp lag. Detected frames are saved as JPEGs on the Pi's SD card and detection metadata (timestamp, image filename, pothole count, latitude, longitude, cluster_count) are appended to a CSV for offline analysis.

For area-level assessment, detections are clustered by GPS location with a 150 m radius. Each cluster is assigned a severity colour on the Flask + Leaflet.js dashboard: green (1–4 potholes) for low, yellow (5–9) for moderate, and red (10+) for high severity. The dashboard enables quick visualization of pothole distribution and severity for municipal maintenance planning.

Advantages include on-device, real-time processing; low hardware cost; offline data storage and later aggregation; GPS-tagged detections for precise mapping; severity-aware clustering to prioritize repairs; and scalability across large road networks. Practical considerations addressed in the system include ONNX conversion for faster inference on the Pi, reduced input resolution (512×512) to balance accuracy and performance, and continuous GPS polling to improve location reliability.

B. System Architecture

Figure 2: System Architecture Diagram



The architecture of the proposed system is designed to support real-time pothole detection, GPS-based location tracking, local data storage, and web-based visualization using low-cost edge hardware. The system follows a modular structure with six main components: camera input, pothole detection, GPS coordinate acquisition, image storage, CSV-based logging, and dashboard visualization.

A. Hardware Layer

The first layer is the data acquisition layer. A Pi Camera continuously captures road video, while the Neo6M GPS module provides location coordinates. Both inputs are connected to the Raspberry Pi 4, which acts as the central controller for collecting frames and synchronizing them with GPS data.

B. AI Processing Layer

The second layer is the AI processing layer running directly on the Raspberry Pi 4. Here, the YOLOv8 detection engine powered by ONNX Runtime analyses incoming frames in real time, detects potholes, and tags them with GPS information. This layer also performs filtering, clustering, and severity calculation so that raw detections are converted into meaningful road-condition data.

C. Data Management Layer

The third layer handles local data storage. Detected pothole images are saved as JPEG files on the SD card, and metadata such as timestamp, latitude, longitude, pothole count, and cluster information are written to a CSV file. This design keeps the system lightweight and offline-capable, so it does not depend on continuous internet access.

D. Web Server Layer

The fourth layer is the Flask web server. It acts as the bridge between the edge device and the web interface by storing and retrieving detection data and serving the dashboard content. The server communicates with local storage and provides the pothole images and metadata needed for visualization.

E. Visualization Layer

The final layer is the visualization and admin interface. A Leaflet.js-based map dashboard displays detected potholes and clustered road segments, while the admin or operator can monitor locations and severity levels interactively. This makes it easy to identify damaged road sections and prioritize maintenance actions.

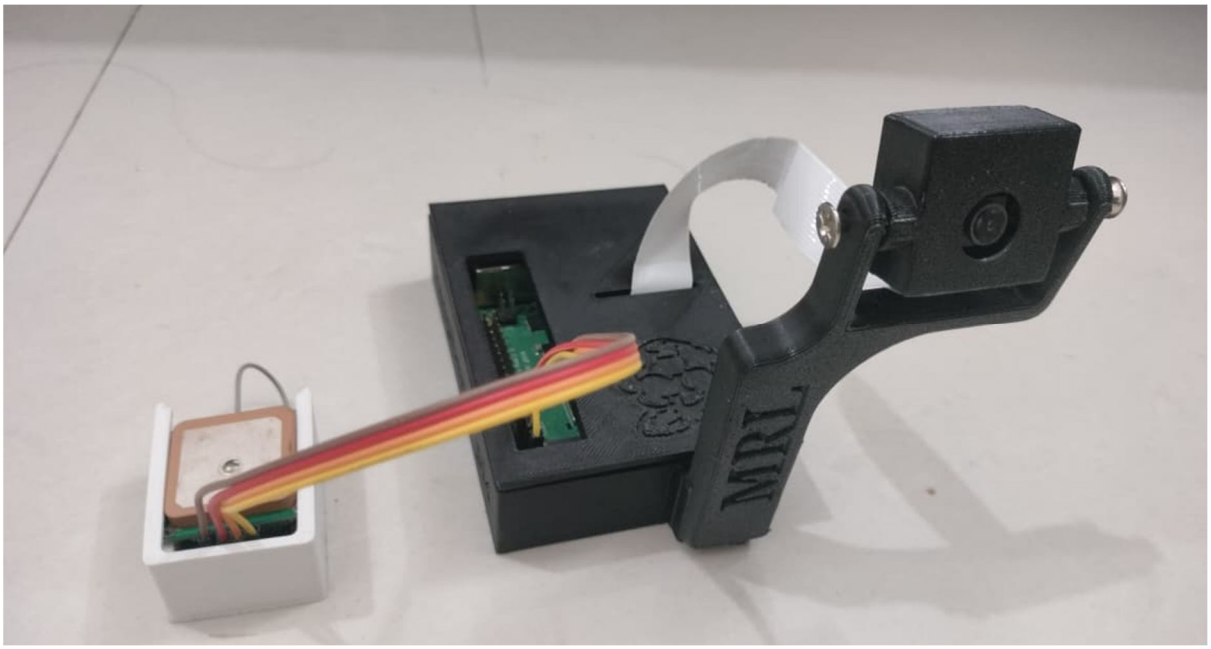
V. IMPLEMENTATION

A. System Setup and Development Environment

The proposed system runs on a Raspberry Pi 4 Model B with 4 GB RAM, which serves as the main edge device for data acquisition, inference, and local storage. An 8 MP Pi Camera continuously captures road video, while a Neo6M GPS module connects through the serial interface to provide location updates during operation.

The software stack includes Python, Ultralytics YOLOv8n, and ONNX Runtime for on-device detection. Flask serves as the backend web server, and Leaflet.js handles map-based visualization in the frontend. The Raspberry Pi OS provides compatibility with the camera, serial communication, and Python libraries, while the SD card stores images and CSV records locally.

Figure 3: Image of the Prototype



B. Dataset Preparation and Model Training

The pothole dataset comes from two sources: manually collected road footage and publicly available images from Kaggle. After augmentation, the dataset contains around 5,000 to 6,000 images and is divided into training, validation, and testing sets in a 70:20:10 ratio. Bounding-box annotations are prepared in Roboflow so that the model learns to localize potholes accurately.

To improve generalization, the dataset undergoes augmentation using horizontal flipping, small rotations, brightness and exposure variation, blur, and light noise. These transformations help the model handle different road surfaces, camera angles, and lighting conditions. YOLOv8n trains for 75 epochs with an input size of 512 × 512 and batch size 8 on a Google Colab T4 GPU, and the trained model is evaluated using precision, recall, F1 score, and mAP.

Figure 4: Annotated Dataset Images

C. Detection Pipeline and Data Logging

During runtime, the Pi Camera continuously captures live video frames and forwards them to the Raspberry Pi processing pipeline. Each frame is passed to the YOLOv8n model running through ONNX Runtime, where potholes are detected and marked with bounding boxes. The YOLOv8n variant balances detection accuracy with inference speed, which makes it suitable for edge deployment.

Whenever the system detects a pothole, it saves the corresponding frame as a JPEG image on the local SD card. At the same time, the system appends metadata such as timestamp, image filename, pothole count, latitude, longitude, and cluster count to a CSV file named potholes.csv. This approach keeps the system functional even without internet access and allows later review of the stored detections.

Figure 5: Metadata of Images saved in a CSV File

	A	B	C	D	E	F	G	H	I	J
1	timestamp	image	pothole_c	latitude	longitude	cluster_co	cluster_id			
2	06-02-2026 12:51	pothole_9c2af670fb7e4be6bbe3cf6eaf12bb67.jpg	1	18.60122	73.90489	1	65e3780b-ff92-4ac2-ac22-afbc84caa483			
3	06-02-2026 12:52	pothole_2a35c95e5f0343fd9a6b8c00a2b21ef9.jpg	1	18.60107	73.90503	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
4	06-02-2026 12:53	pothole_26cb233a319b45dabf9bc7e8e48c453d.jpg	1	18.60107	73.90503	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
5	06-02-2026 12:54	pothole_13c863baf0164f98832e61646ea80e2f.jpg	1	18.60102	73.90509	1	080411bf-d1bf-4c60-a92b-c845b98a20c8			
6	06-02-2026 12:54	pothole_77b79ddbb18943729ef6f9bbde040b56.jpg	3	18.60109	73.90502	3	8782d49b-0147-409d-a6c4-0ba711f746fe			
7	06-02-2026 12:56	pothole_b98613eed7db4398b908e69243376565.jpg	2	18.60108	73.90503	2	8782d49b-0147-409d-a6c4-0ba711f746fe			
8	06-02-2026 12:57	pothole_480f3e0fcf3441e8b261d8ebe61515c8.jpg	1	18.60108	73.90505	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
9	06-02-2026 12:57	pothole_50acf42db44f4f93b051f7b2bdbacec0.jpg	1	18.6011	73.90506	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
10	06-02-2026 12:57	pothole_6bb29d5392ee4c14906c95fc2e41cf9e.jpg	1	18.60111	73.90506	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
11	06-02-2026 12:58	pothole_c6eb09b755bd43699accc7514edec100.jpg	1	18.60111	73.90506	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
12	06-02-2026 12:59	pothole_29ee6b9a13da47409a8941d0f1c88d7f.jpg	1	18.60111	73.90506	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
13	06-02-2026 13:00	pothole_865c1990328b49c18a30912fd10d8e28.jpg	1	18.60113	73.90505	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
14	06-02-2026 13:00	pothole_28c6447220f44f20ac6f064645968414.jpg	1	18.60113	73.90505	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
15	06-02-2026 13:00	pothole_97b70699431c4d2b9dd4517698b85aab.jpg	1	18.60114	73.90505	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
16	06-02-2026 13:01	pothole_a33c4dc982f5448b89cb478344d6504b.jpg	1	18.60116	73.90504	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
17	06-02-2026 13:05	pothole_5039fc3a32fd499393ff4993e51b1819.jpg	1	18.60117	73.90504	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
18	06-02-2026 13:05	pothole_92d32796e9234370a066a07e0adf7f31.jpg	1	18.60117	73.90504	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
19	06-02-2026 13:05	pothole_9faf288525cf4f01bd4f9ff1f275259c.jpg	1	18.60118	73.90504	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
20	06-02-2026 13:22	pothole_f73a7f6ef90b43b39e3882c996cb4855.jpg	1	18.60114	73.90508	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
21	06-02-2026 13:24	pothole_3575035937fb4c15a6a9d9442f6c84a9.jpg	1	18.60113	73.90508	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
22	06-02-2026 13:25	pothole_dd9c81c8f9294a598cadb66234786620.jpg	1	18.60113	73.90506	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
23	06-02-2026 13:26	pothole_72b61ad382ba496e835a1708d7d571d8.jpg	1	18.60113	73.90505	1	8782d49b-0147-409d-a6c4-0ba711f746fe			
24	06-02-2026 13:27	pothole_89fb2cdaaa854c8a979e0ca7cc7aafef.jpg	3	18.60113	73.90505	3	8782d49b-0147-409d-a6c4-0ba711f746fe			
25	06-02-2026 13:28	pothole_268a6843c0994d7d9221896e00ae473a.jpg	1	18.60113	73.90508	1	8782d49b-0147-409d-a6c4-0ba711f746fe			

D. GPS Clustering and Severity Mapping

The Neo6M GPS module updates continuously every 200 ms, so the system always uses the latest available coordinates when a pothole is detected. This reduces delay between frame capture and geotagging and improves the accuracy of location mapping. The GPS readings are linked to each detected pothole and used for map-based visualization and clustering.

A clustering radius of 150 meters groups pothole detections that belong to the same damaged road segment. This helps convert individual detections into road-level severity information that is easier for maintenance teams to interpret. The clusters use color-coded severity levels: green for low density, yellow for moderate density, and red for high density pothole areas.

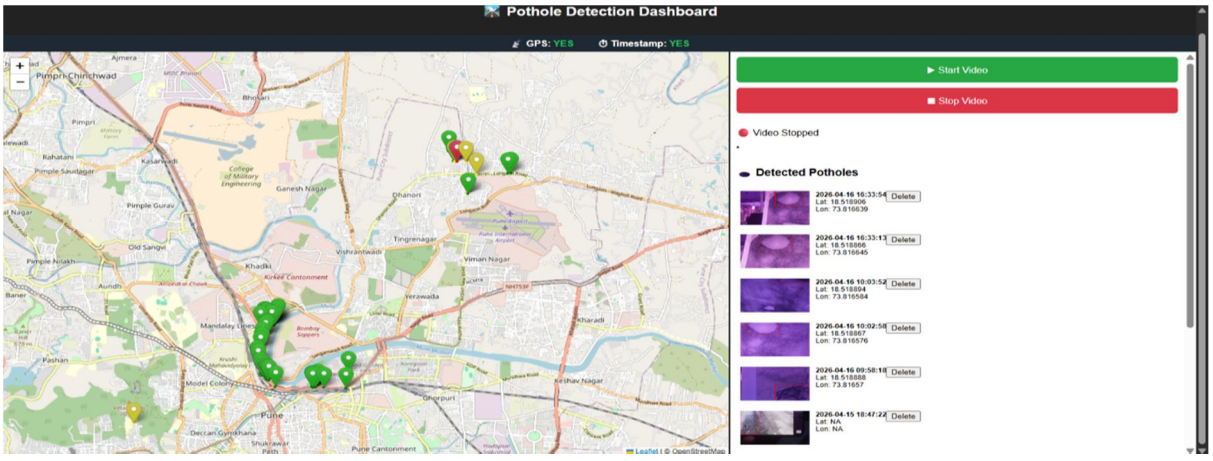
E. Dashboard and Visualization

The Flask backend connects the stored detection data with the web interface. It reads pothole records from the CSV file and retrieves stored images from local memory when required. Leaflet.js displays pothole locations and clusters on an interactive map, which allows the operator to inspect damaged road sections visually.

Figure 6: Potholes Detected by the Model in Real-Time

The dashboard presents pothole density and severity in a clear format, making it easier to identify high-priority areas for repair. Since inference and storage remain local on the Raspberry Pi, the dashboard continues to work in offline or low-connectivity environments. This makes the system practical for smart city monitoring, municipal planning, and low-cost field deployment.

Figure 7: Dashboard



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

VI. EXPERIMENTAL RESULTS AND ANALYSIS

The system achieves practical real-time pothole detection on edge hardware, with a test-set precision of 0.832, recall of 0.631, F1 score of 0.718, mAP@0.5 of 0.723, and mAP@0.5:0.95 of 0.366. These results show that the model identifies potholes with good precision, while recall remains lower, which means some potholes are still missed in challenging frames.

A. Detection Performance

The YOLOv8n model runs at about 3.3 FPS for detection on the Raspberry Pi 4, while the camera capture pipeline runs at 15 FPS. This gap shows that the system prioritizes stable on-device inference over maximum throughput, which is appropriate for a low-cost embedded platform. Converting the model to ONNX format improves inference efficiency and makes deployment on the Raspberry Pi more practical.

B. Accuracy Analysis

The precision value of 0.832 indicates that most detected potholes are correct, so false positives stay relatively low. The recall value of 0.631 shows that the model still misses some potholes, especially in visually difficult scenes such as poor lighting, motion blur, or complex road textures. The mAP@0.5 score of 0.723 suggests that the detector performs well at a standard overlap threshold, but the lower mAP@0.5:0.95 score of 0.366 shows that localization becomes less stable under stricter evaluation criteria.

C. Effect of Input Size

The system uses 512 × 512 input resolution in the final deployment because the Raspberry Pi has limited compute and memory resources. This reduction improves runtime performance and reduces storage and processing overhead, but it also causes a small trade-off in detection detail. In practice, the chosen size provides a balanced compromise between accuracy and real-time usability on the edge device.

D. GPS and Clustering Analysis

Continuous GPS polling every 200 ms improves the alignment between pothole detections and their coordinates. Compared with event-based GPS retrieval, this method reduces location lag and produces more reliable geotagged records. The 150 m clustering radius groups nearby potholes into meaningful damaged road segments, which helps the dashboard represent road severity instead of only isolated detections.

E. Dashboard Output

The Flask and Leaflet.js dashboard present the detections in a clear map-based format, making it easy to inspect pothole locations and cluster severity. Green, yellow, and red markers provide a quick visual summary of low, moderate, and high pothole density areas. This makes the system useful for municipal monitoring because it converts raw detections into actionable road-condition information.

Overall, the results confirm that the proposed system works as a low-cost edge solution for pothole detection and mapping. The system performs well in precision and map-based visualization, while recall and high-threshold localization accuracy leave room for improvement. The main strengths are offline operation, local storage, real-time mapping, and deployment on accessible hardware, making the approach suitable for smart city and maintenance applications.

VII. CHALLENGES AND LIMITATIONS

The proposed system works well as a low-cost edge solution, but it also faces several practical challenges. These limitations mainly come from hardware constraints, environmental conditions, and the nature of real-world road scenes.

A. Hardware Constraints

Raspberry Pi 4 has limited processing power compared with desktop GPUs, so running deep learning inference in real time becomes difficult. Even after converting the model to ONNX, the system still achieves only moderate detection speed. Continuous inference can also increase heat buildup, which may affect long-duration operation if proper cooling is not provided.

B. GPS Accuracy

The Neo6M GPS module depends heavily on signal quality and open-sky visibility. In dense urban areas, under bridges, near tall buildings, or in covered roads, coordinate accuracy may drop. Although continuous polling every 200 ms improves timing, the system still cannot fully eliminate GPS noise or signal delay.

C. Detection Reliability

The model detects potholes reasonably well, but performance still varies with lighting, shadows, motion blur, and road texture. Potholes that are small, partially covered, or poorly visible can be missed. This is reflected in the lower recall compared to precision, which means the system is better at confirming potholes than finding every single one.

D. Input Resolution Trade Off

The system uses 512 × 512 input size to balance speed and resource use on the Raspberry Pi. This improves runtime, but it also reduces the amount of fine visual detail available to the model. As a result, tiny potholes or weak road defects may be harder to detect than in higher-resolution training or inference settings.

E. Environmental Conditions

The system works best in daylight and relatively clear road conditions. Night-time scenes, rain, wet roads, glare, and heavy traffic can reduce detection/' quality. These situations create visual noise that makes pothole boundaries less distinct.

To summarize, the system provides a practical prototype for pothole detection and mapping, but it is not yet ideal for all field conditions. Its main weaknesses are limited edge performance, dependence on GPS signal quality, and reduced reliability in difficult lighting or weather. These limitations make it suitable for pilot deployment and monitoring, while future improvements are needed for full-scale municipal use.

VIII. ADVANTAGES AND FUTURE SCOPE

The proposed system offers several important advantages for road-condition monitoring and maintenance. It performs real-time pothole detection using live camera input, which helps identify damaged road sections quickly while the vehicle is in motion. Since the system runs on a Raspberry Pi 4 with commonly available hardware components, it remains low-cost and practical for deployment in resource-constrained environments. Another major advantage is that detection happens locally on the edge device, so the system does not depend on continuous internet connectivity and avoids delays caused by cloud processing. In addition, each pothole detection is linked with GPS coordinates, which makes it easier to locate affected road segments accurately on a map. The system also stores images and metadata locally on the SD card, which supports offline operation in remote areas. The GPS-based clustering mechanism further improves usability by grouping nearby potholes into road segments and assigning severity levels, helping authorities prioritize repairs more effectively. Because the architecture is modular, it can also be extended easily for larger road networks or additional features.

The future scope of the system is broad and practical. One important improvement is cloud integration, which can replace or complement local CSV storage with a centralized database for better data management and access across multiple locations. A mobile application can also be developed so that field workers and municipal staff can view pothole locations, verify detections, and update repair status in real time. The system can be enhanced for night-time and low-light conditions by using better training data, improved preprocessing, or infrared camera support. In future versions, the model can be extended beyond potholes to detect other road defects such as cracks, patches, manholes, and bumps. Depth-based analysis can also be added to estimate pothole severity more accurately. An automated alert system can notify municipal authorities whenever severe clusters are detected, which would make the system more actionable. Further model optimization through quantization, pruning, or more efficient edge hardware can improve speed and reduce power consumption. Finally, the system can be deployed across multiple vehicles or road survey units to build a larger city-wide monitoring network.

IX. CONCLUSION

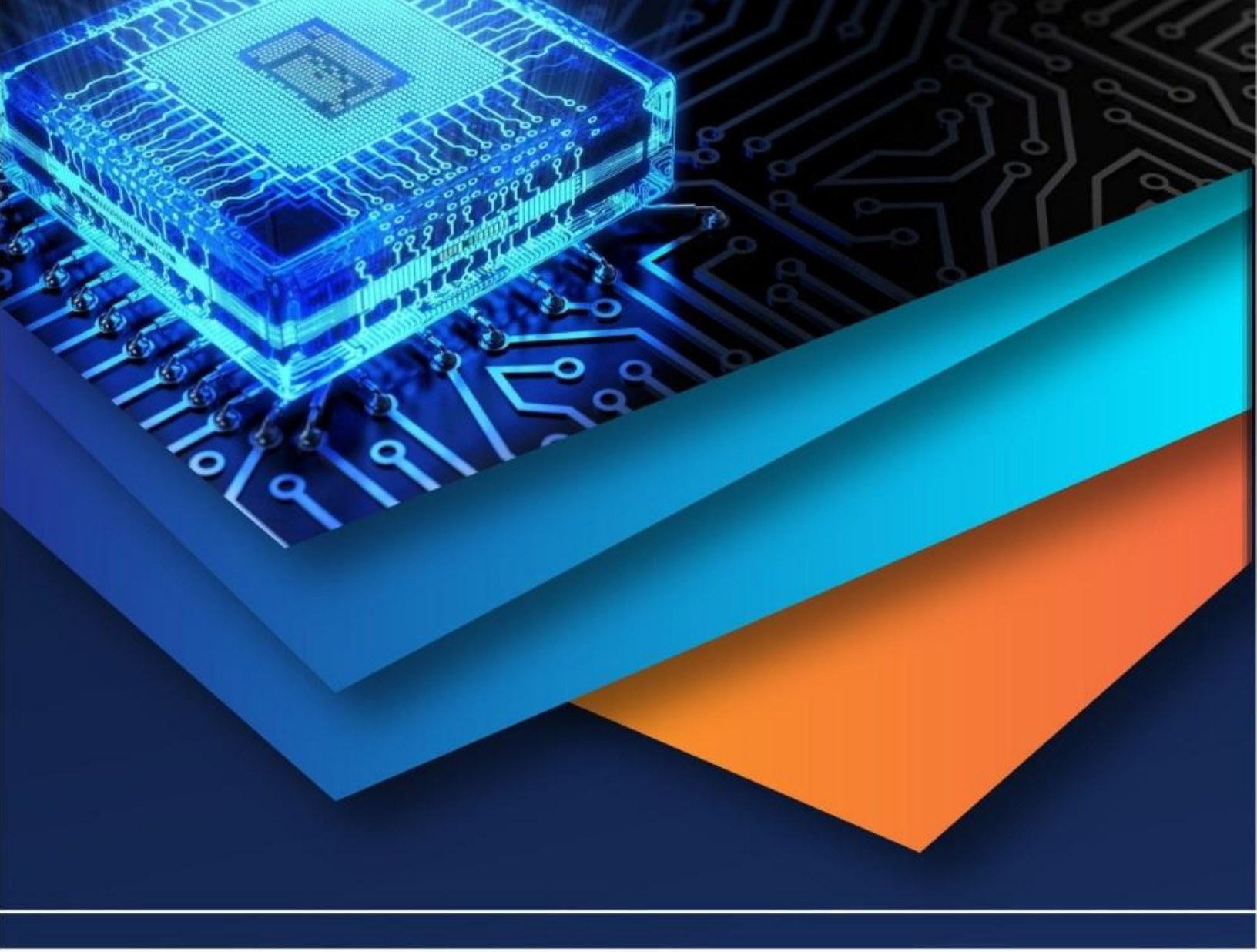
The proposed system demonstrates that real-time pothole detection and mapping can be achieved effectively using low-cost edge hardware. By combining a Pi Camera, Raspberry Pi 4, YOLOv8n inference through ONNX Runtime, GPS-based tagging, local storage, and a Flask–Leaflet dashboard, the system provides a complete workflow for automated road-condition monitoring.

The results show that the approach is practical for field deployment, especially where internet connectivity is limited and rapid inspection is needed. The use of GPS clustering and severity-based visualization makes the output more useful for municipal planning by converting individual detections into actionable road-maintenance information.

At the same time, the system highlights the trade-off between accuracy and resource constraints on edge devices. Challenges such as limited inference speed, GPS dependency, and reduced performance in difficult lighting conditions indicate areas where further improvement is possible. Overall, the project offers a scalable and cost-effective foundation for smart transportation and road maintenance applications.

REFERENCES

[1] Y. Safyari, M. Mahdianpari, and H. Shiri, "A Review of Vision-Based Pothole Detection Methods Using Computer Vision and Machine Learning," Sensors, vol. 24, no. 5652, pp. 1–39, Aug. 2024, doi: 10.3390/s24175652.
[2] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, A. Mraz, T. Kashiyama, and Y. Sekimoto, "Deep Learning-Based Road Damage Detection and Classification for Multiple Countries," Automation in Construction, vol. 132, p. 103935, Sep. 2021, doi: 10.1016/j.autcon.2021.103935.
[3] Z. Wang, Z. Ma, Z. Wang, S. Gao, and J. Peng, "A Novel Road Damage Detection Model with Efficient Attention and Dynamic Snake Convolution," Engineering Applications of Artificial Intelligence, vol. 163, p. 112618, 2026, doi: 10.1016/j.engappai.2025.112618.
[4] P. Chede, S. Chitale, A. Mansuke, P. Ghodake, and S. P. Vidhate, "Real-Time Pothole Detection Using AI and ML," International Research Journal of Engineering and Technology (IRJET), vol. 11, no. 10, pp. 812–816, Oct. 2024.
[5] B. R. Kumar, K. Hrushikesh, K. N. Reddy, and A. Sriram, "A Paper on Pothole Detection," Journal of Engineering Sciences, vol. 14, no. 3, pp. 287–292, 2023.
[6] K. Singh, S. Hazra, C. Mukherjee, S. G., and S. Gowda, "IoT-Based Real-Time Pothole Detection System Using Image Processing Techniques," International Journal of Scientific and Technology Research, vol. 9, no. 2, pp. 785–789, Feb. 2020.
[7] P. Chede, S. Chitale, A. Mansuke, P. Ghodake, and S. P. Vidhate, "Real-Time Pothole Detection Using ML," International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS), vol. 7, no. 4, pp. 7913–7919, Apr. 2025, doi: 10.56726/IRJMETS72686.
[8] M. Prakash B and S. K. C, "Enhanced Pothole Detection System Using YOLOX Algorithm," Autonomous Intelligent Systems, vol. 2, no. 22, pp. 1–17, 2022, doi: 10.1007/s43684-022-00037-z.
[9] G. R. S. K. V. Rao, P. R. K. A. Tiwari, and N. B. B. Sahu, "Pothole Detection in Road Images Using Machine Learning," International Journal of Engineering Research and Technology (IJERT), vol. 8, no. 4, pp. 1–6, 2020.
[10] D. V. D. H. S. Kumari, "Machine Learning Approach for Pothole Detection and Classification," International Journal of Computer Applications, vol. 182, no. 37, pp. 1–4, 2019, doi: 10.5120/ijca2019918443.
[11] M. H. Asad, S. Khaliq, M. H. Yousaf, M. O. Ullah, and A. Ahmad, "Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective," Advances in Civil Engineering, vol. 2022, Article ID 9221211, 13 pages, 2022, doi: 10.1155/2022/9221211.
[12] K. A. Vinodhini and K. R. A. Sidhaarth, "Pothole Detection in Bituminous Road Using CNN with Transfer Learning," Measurement: Sensors, vol. 31, art. 100940, Feb. 2024, doi: 10.1016/j.measen.2023.100940.
[13] S. K. Satti, S. K. Devi, P. Maddula, and N. V. Ravipati, "Unified Approach for Detecting Traffic Signs and Potholes on Indian Roads," Journal of King Saud University – Computer and Information Sciences, 2023 (in press).
[14] B. Kang and S. Choi, "Pothole Detection System Using 2D LiDAR and Camera," in Proc. Int. Conf. on Ubiquitous and Future Networks (ICUFN), pp. 744–746, 2017.
[15] S. M. R. Ghosh, "Deep Learning for Computer Vision: A Brief Review," IEEE Access, vol. 7, pp. 171903–171912, 2019, doi: 10.1109/ACCESS.2019.2957284.
[16] M. Alzahrani and M. T. Alghamdi, "A Survey on Pothole Detection and Classification Techniques," Sensors, vol. 21, no. 1, p. 123, 2021, doi: 10.3390/s210100123.
[17] M. A. I. P. Mehta and K. S. Kumar, "Smart Pothole Detection System Using IoT and Machine Learning," in Proc. 3rd Int. Conf. on Intelligent Computing and Control Systems (ICICCS), pp. 682–687, 2020, doi: 10.1109/ICICCS48848.2020.9142490.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY