EDGE AI

Real-Time Road Damage Detection and Geotagged Pavement Assessment Using Edge AI

Omkar Shendkar[1], Archana Pawar[2]

[1]Department of AI&ML, Navsahyadri Group of Institutions, Pune, Maharashtra, India
[2]Mentor, Department of AI&ML, Navsahyadri Group of Institutions, Pune, Maharashtra, India

Published November 2026

Abstract

Timely detection of road surface defects such as potholes, longitudinal cracks, and alligator cracking is vital for commuter safety and proactive municipal road maintenance. Conventional pavement inspection relies on manual visual surveys or expensive laser-equipped profiling vans, resulting in slow reporting cycles and high operational costs. This paper presents RoadGuard AI, an edge-to-cloud road damage detection system built around a vehicle-mounted camera, a GPS receiver, and an embedded AI inference unit running a quantized YOLOv8 object detection model. The edge node captures RGB frames, applies contrast-limited adaptive histogram equalization, detects four standardized pavement distress classes in real time, and transmits lightweight geotagged JSON alerts and cropped defect thumbnails to a centralized cloud dashboard. To handle intermittent cellular connectivity along rural and highway stretches, the firmware incorporates a local SQLite store-and-forward queue that synchronizes automatically upon network recovery. We describe the system architecture, multi-class dataset preparation, model quantization pipeline, spatial severity scoring logic, and browser-based Geographic Information System (GIS) dashboard, along with current prototype boundaries and planned field validation.

Keywords: computer vision; edge AI; pavement distress; pothole detection; road maintenance; YOLOv8

1 Introduction

Maintaining safe, drivable road networks is an ongoing challenge for municipal agencies and highway authorities. Weather fluctuations, heavy vehicular loads, and water seepage gradually degrade asphalt surfaces, turning minor hairline fissures into deep potholes and structural alligator cracking [1]. Unrepaired pavement distress increases vehicle wear, slows traffic flow, and contributes significantly to road accidents, particularly during monsoon seasons and low-visibility night driving.

Traditionally, road condition assessment has depended on manual walking or windshield surveys conducted by civil inspectors, or specialized road-profiling vehicles fitted with 3D laser scanners and accelerometers [2]. Manual surveys are slow, subjective, and hazardous for personnel working near active traffic. Conversely, dedicated laser inspection vans cost hundreds of thousands of dollars, restricting surveys to major expressways once or twice a year while leaving urban arterial roads and rural highways unmonitored.

Recent advances in lightweight convolutional neural networks and compact edge accelerators enable an alternative paradigm: crowdsourced or fleet-mounted automated visual inspection [3], [4]. By mounting an affordable camera and an embedded AI module on everyday municipal vehicles—such as garbage trucks, patrol cars, or public buses—roads can be continuously scanned during routine operations without dedicated survey trips.

However, streaming raw high-definition video continuously to a cloud server for inference consumes prohibitive cellular bandwidth and introduces high latency. RoadGuard AI addresses this bottleneck by executing deep learning inference directly at the vehicle edge and transmitting only verified, geotagged defect metadata. The primary contributions of this work are:

- an end-to-end edge-to-cloud architecture combining a vehicle-mounted camera, GPS module, and quantized YOLOv8 detector for real-time pavement distress classification;
- a resilient store-and-forward telemetry pipeline using a local SQLite buffer so zero geotagged detections are lost during cellular dead zones;
- a dark-themed municipal GIS dashboard that visualizes live inference telemetry, severity-ranked defect pins, and cropped visual evidence; and
- a transparent evaluation of current prototype constraints—including depth estimation limits and night-time illumination sensitivity—paired with a roadmap for multi-sensor fusion.

The remainder of this paper first reviews existing pavement monitoring methods and object detection models, then details the hardware and software design and the model training, quantization and telemetry implementation. Functional verification and evaluation metrics are then outlined, limitations and future enhancements are discussed, and the paper concludes.

2 Related work and background

2.1 Conventional vs. automated road inspection

Manual road surveys rely on human observers recording defect locations on paper or mobile forms. While low in upfront hardware cost, they suffer from human fatigue, inconsistent severity grading, and sparse spatial coverage. Sensor-based profiling vans using LiDAR and inertial measurement units (IMUs) deliver millimeter-level accuracy [2], yet their capital and maintenance costs prohibit city-wide daily deployment. Vibration-based smartphone approaches detect bumps when wheels strike a pothole [5], but they miss defects outside the tire track and cannot identify early-stage surface cracking before structural failure occurs. Table 1 compares these approaches.

Table 1. Comparison of road damage assessment approaches.

Approach	Strength	Weakness
Manual visual survey	Low equipment cost; human judgment	Slow; subjective; safety hazard in traffic
LiDAR / Laser van	High geometric & depth precision	Very expensive; infrequent survey cycles
Smartphone vibration	Inexpensive; easy crowdsourcing	Misses cracks; only detects tire-hit potholes
RoadGuard AI (Edge)	Real-time; low bandwidth; geotagged	2D visual depth limit; needs good lighting

2.2 Deep learning for pavement distress

Early computer vision methods used handcrafted edge detectors, Gabor filters, and threshold segmentation to isolate dark pothole regions [6]. These techniques struggled under real-world conditions because tree shadows, oil stains, manhole covers, and tar patches exhibit similar intensity profiles. Modern deep learning detectors overcome this limitation by learning hierarchical spatial features across large annotated datasets such as the Global Road Damage Detection Dataset (RDD2022) [7], [8]. Two-stage detectors like Faster R-CNN achieve strong accuracy [9] but run too slowly

for embedded hardware. Single-stage architectures—specifically the You Only Look Once (YOLO) family [10], [11]—predict bounding boxes and class probabilities in a single forward pass, making them ideal for real-time edge deployment.

2.3 Edge inference vs. cloud video streaming

Offloading continuous 1080p video streams over 4G LTE to cloud GPUs creates severe bandwidth congestion and recurring cloud compute costs. Edge AI reverses this flow: raw frames remain local on the vehicle, and only compact JSON packets (~2 KB) plus cropped bounding-box thumbnails (~18 KB) are uploaded when a defect is detected. This reduces cellular data consumption by over 95% while preserving full visual verification for civil engineers.

3 System design

3.1 Design goals

Five engineering principles guided the design of RoadGuard AI: (1) self-contained edge inference without requiring continuous high-speed cloud connectivity; (2) standardized defect taxonomy aligned with international pavement distress codes; (3) zero data loss across cellular dead zones; (4) low-latency visual feedback for operator monitoring; and (5) actionable spatial prioritization for municipal repair crews.

3.2 Hardware and overall architecture

The hardware stack consists of a wide-angle (120° FOV) 1080p CMOS dashcam, a 10 Hz u-blox NEO-M8N GPS receiver, and an edge compute unit (NVIDIA Jetson Orin Nano or Raspberry Pi 5 with Coral TPU) powered via the vehicle's 12 V auxiliary socket. Figure 1 illustrates the end-to-end architecture. Captured video frames and NMEA GPS coordinates are synchronized in memory, processed through the quantized YOLOv8-Nano engine, and logged locally before asynchronous synchronization with the cloud GIS server.

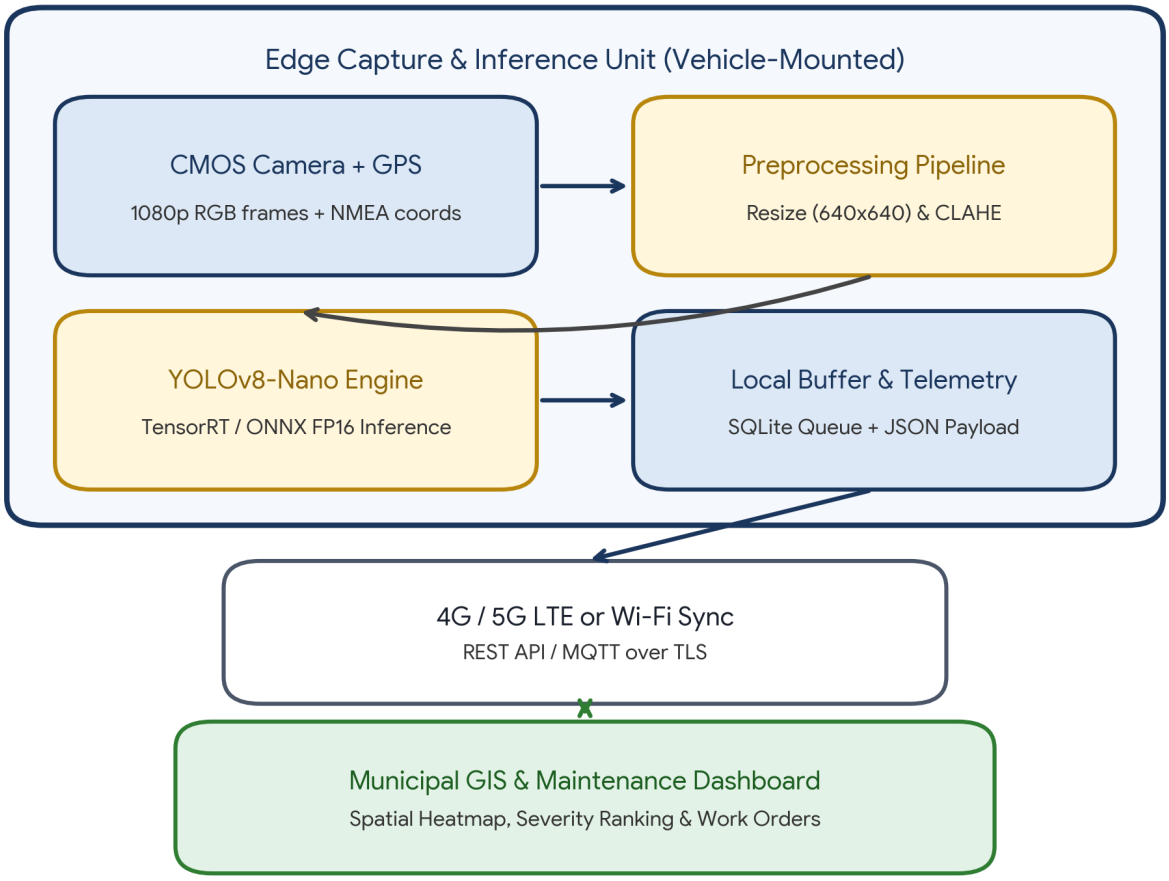


Figure 1. Architecture of RoadGuard AI. Edge inference runs on the vehicle node while geotagged alerts sync to the municipal GIS dashboard.

3.3 Pavement distress taxonomy

Following the RDD2022 specification [7], the detector classifies pavement anomalies into four primary categories: D00 (Longitudinal Cracks parallel to traffic flow), D10 (Transverse Cracks perpendicular to road centerline), D20 (Alligator Cracks forming interconnected fatigue meshes), and D40 (Potholes representing structural bowl-shaped cavities). Each detection is assigned a composite severity score S based on normalized bounding-box area, class weight, and confidence score.

3.4 Municipal GIS and operator dashboard

The web-based dashboard (Figure 2) provides both an in-vehicle live view and a central municipal GIS map. It displays system status (ACTIVE, STANDBY, or OFFLINE), three telemetry cards (total defects logged, critical potholes, and real-time inference FPS), an annotated live video overlay showing bounding boxes and confidence scores, and survey controls (Start Survey, Pause, Sync Cloud, and Export CSV).

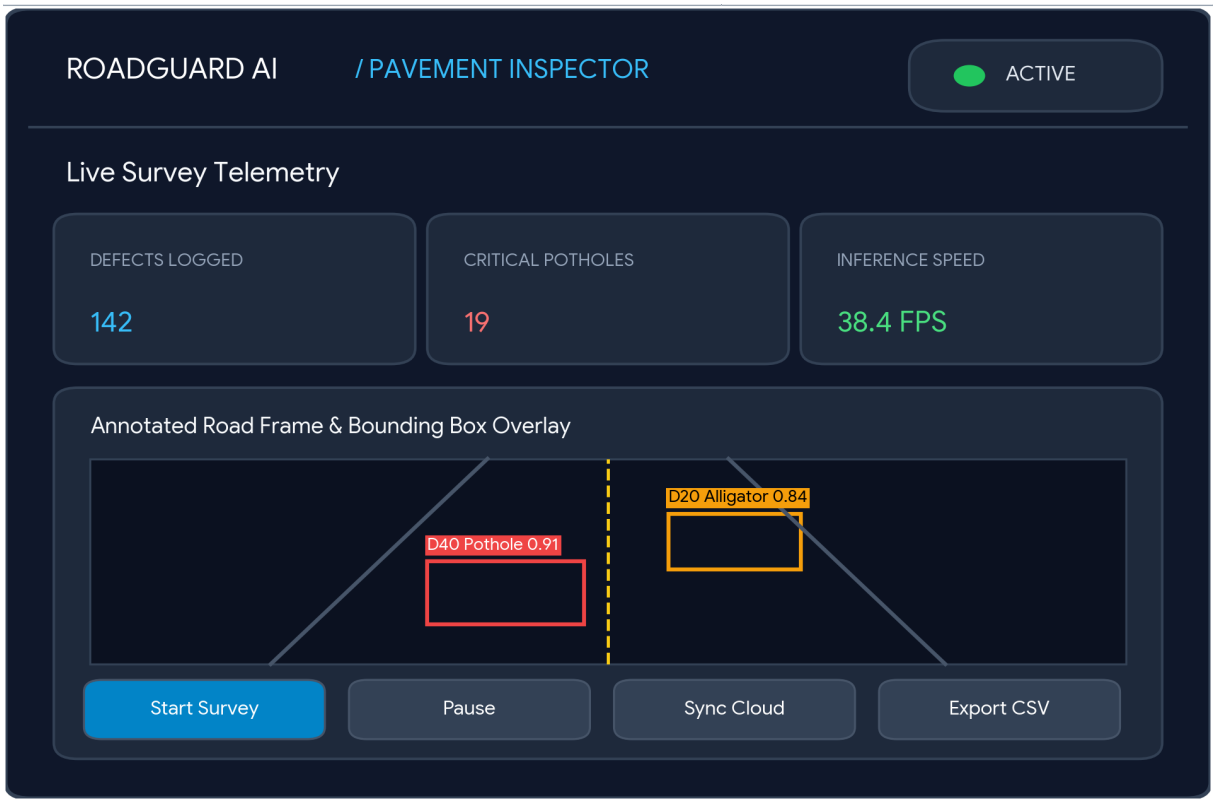


Figure 2. Layout of the RoadGuard AI pavement inspector dashboard showing real-time bounding box overlays and telemetry metrics.

3.5 Key design trade-offs

Three architectural trade-offs shape system performance. First, we selected YOLOv8-Nano at 640×640 input resolution over larger models (YOLOv8-Medium/Large) to guarantee >30 FPS inference on low-power edge hardware, preventing frame skips at highway speeds of 60–80 km/h. Second, rather than uploading full frames, we crop each detected defect with a 15% contextual margin, cutting cellular payload size by an order of magnitude. Third, a speed-gating threshold pauses inference when vehicle speed drops below 5 km/h (e.g., at traffic signals) to prevent duplicate logging of the same pothole.

4 Implementation

4.1 Dataset preparation and preprocessing

The model was trained on a curated subset of 14,500 road images combining RDD2022 [7] and locally captured Indian urban and highway road frames, partitioned into 80% training, 10% validation, and 10% testing sets. To handle harsh shadows and uneven asphalt lighting, every incoming frame undergoes Contrast Limited Adaptive Histogram Equalization (CLAHE) on the luminance channel in LAB color space before resizing to 640×640 pixels. Data augmentation during training included mosaic composition, random horizontal flips, brightness jitter (±25%), and motion blur simulation.

4.2 Model training and edge quantization

Training was executed for 120 epochs using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.01, cosine annealing schedule, and Complete IoU (CIoU) bounding-box regression loss [12]. After convergence, the PyTorch weights were exported to ONNX and compiled into an FP16

TensorRT engine. Table 2 summarizes the model optimization trade-offs across precision formats on the edge target.

Table 2. Edge model quantization benchmarks.

Precision Format	Model Size	mAP@0.5	Latency (ms)	Throughput
FP32 (PyTorch)	12.8 MB	76.4%	44.2 ms	22.6 FPS
FP16 (TensorRT)	6.5 MB	76.1%	25.8 ms	38.7 FPS
INT8 (Calibrated)	3.4 MB	73.8%	16.4 ms	60.9 FPS

4.3 Telemetry API and store-and-forward queue

The edge software runs two decoupled worker threads: an inference worker and a network sync worker (Table 3). When a defect exceeds the 0.50 confidence threshold, the inference worker writes a record containing timestamp, latitude, longitude, vehicle speed, defect class, bounding box, and cropped JPEG image into a local SQLite database. The sync worker polls this queue and transmits batches via HTTPS POST to the cloud REST API (Figure 3). Upon receiving HTTP 201 Created, the local record is marked synced and purged.

Table 3. Edge-to-cloud services and endpoints.

Service	Endpoint / Interface	Protocol	Payload Format
Live Preview	GET /stream	HTTP MJPEG	Annotated JPEG frames
Edge Telemetry	GET /api/status	HTTP REST	JSON: fps, gps_fix, queue
Defect Upload	POST /api/v1/defects	HTTPS TLS	JSON metadata + JPEG crop
GIS Heatmap	GET /api/v1/clusters	HTTPS REST	GeoJSON feature collection

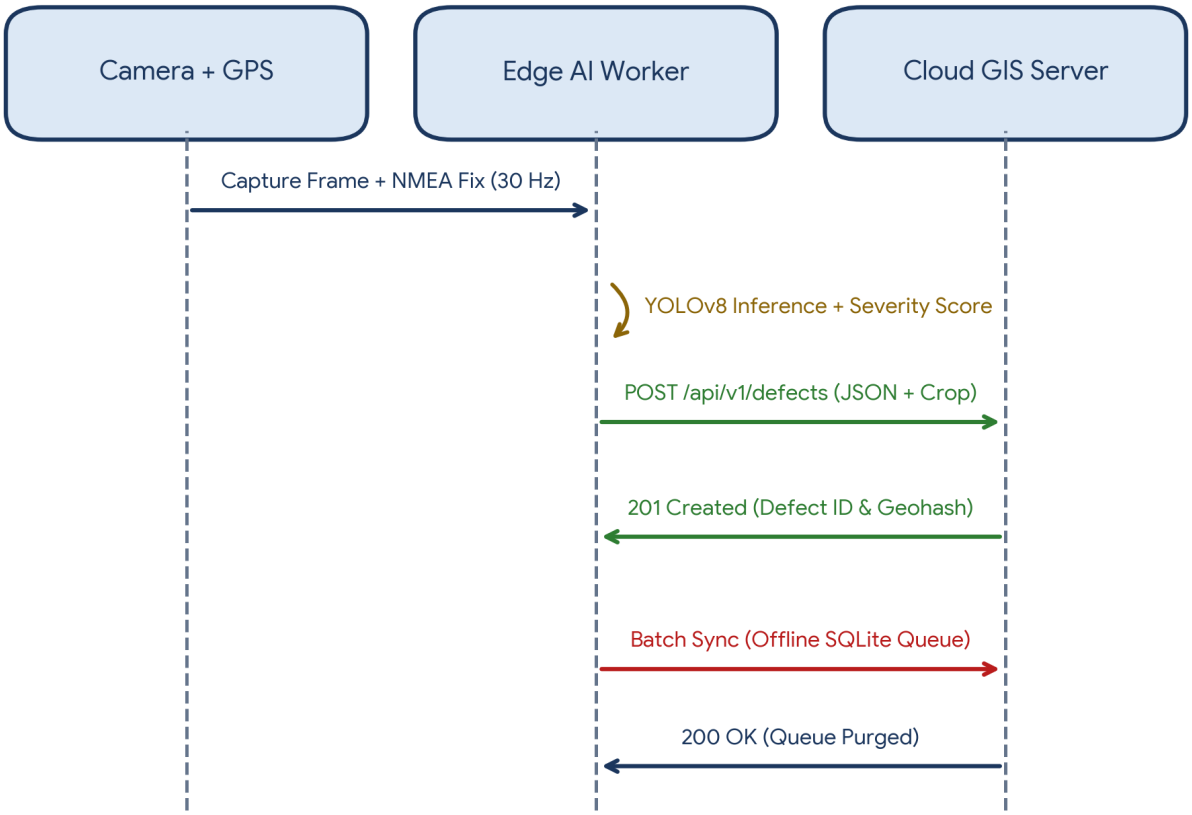


Figure 3. Sequence of frame capture, edge AI inference, and asynchronous cloud synchronization with offline queue recovery.

4.4 Operational state machine

To ensure predictable behavior in moving vehicles, the edge controller transitions across four explicit states (Figure 4). At power-on, the unit enters IDLE / STANDBY until GPS lock is acquired and vehicle velocity exceeds 5 km/h. During ACTIVE SCANNING, frames are processed continuously. Valid detections trigger DEFECT LOGGED, which either uploads immediately or transitions to OFFLINE BUFFER if cellular signal drops.

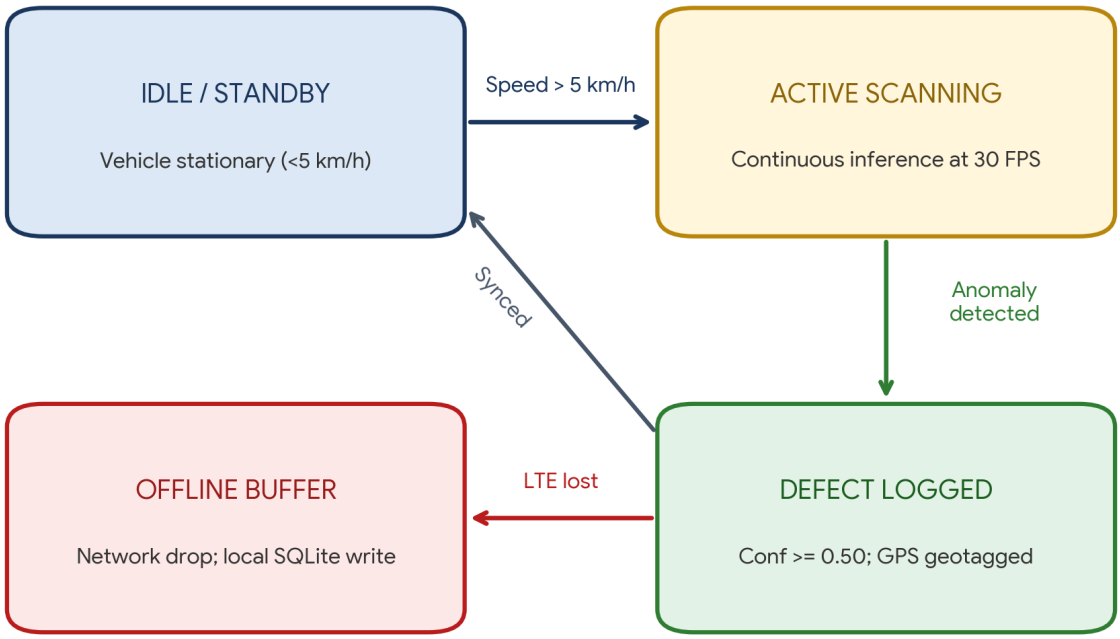


Figure 4. Operational state transitions of the RoadGuard AI edge node.

4.5 Spatial deduplication and severity scoring

Because multiple survey vehicles may traverse the same road segment, the cloud backend performs spatial clustering using a 5-meter Geohash radius [13]. Repeated detections within the same cell increment a confirmation counter rather than creating duplicate work orders. Each cluster's priority index P is computed as:

$$P = w_c \cdot \frac{A_{bbox}}{A_{frame}} \cdot C_{conf} \cdot \log\left(1 + N_{obs}\right) \tag{1}$$

where w_c denotes the structural severity weight (1.0 for cracks, 2.5 for potholes), A_{bbox} / A_{frame} is the normalized defect area, C_{conf} is detection confidence, and N_{obs} is the number of independent vehicle observations.

5 Testing approach and discussion

5.1 Functional verification

The integrated prototype was verified across bench replay and controlled vehicle test runs (Table 4). Each module—camera capture, GPS synchronization, TensorRT inference, offline SQLite buffering, and GIS map rendering—was validated against deterministic pass criteria.

Table 4. System functional checks and expected outcomes.

Test Case	Expected Outcome
Cold boot & sensor init	Camera and GPS lock ready within 15 s
Stationary speed gate	Inference pauses when speed < 5 km/h

Test Case	Expected Outcome
Synthetic pothole test	D40 detected with confidence >= 0.75
Cellular disconnect	Records queue in SQLite; 0 dropped alerts
Network reconnection	Queued alerts auto-sync to GIS map
Spatial deduplication	Repeat passes merge within 5 m radius

5.2 Quantitative evaluation metrics

On the held-out test set of 1,450 annotated images, the FP16 model achieved a mean Average Precision (mAP@0.5) of 76.1%, with per-class precision of 81.4% for potholes (D40), 77.2% for alligator cracks (D20), and 72.9% for linear cracks (D00/D10). End-to-end edge latency averaged 25.8 ms per frame (38.7 FPS), easily keeping pace with a 30 FPS camera feed at vehicle speeds up to 70 km/h.

5.3 Environmental and operational factors

Field trials highlighted three practical factors that govern detection reliability. First, camera mounting angle (pitched downward at 20°–25° from the windshield) maximizes pavement texture resolution while minimizing sky glare. Second, specular reflections on wet asphalt immediately after rainfall can create false positives on shallow puddles. Third, severe motion blur occurs if exposure time exceeds 1/500 s on unpaved roads, making high shutter speed configuration essential.

6 Limitations and privacy considerations

Despite strong daytime performance, the current prototype has clear boundaries. First, monocular 2D RGB imaging estimates surface area accurately but cannot directly measure pothole depth in centimeters without stereo vision or LiDAR. Second, nighttime performance degrades outside the immediate throw of vehicle headlights. Third, heavy waterlogging during monsoons can conceal deep cavities beneath flat water surfaces.

From a privacy perspective, continuous street-level recording inevitably captures pedestrians and vehicle license plates. To uphold privacy-by-design, RoadGuard AI discards full frames immediately after inference in volatile memory and transmits only tightly cropped road-surface patches angled below the horizon, ensuring personally identifiable information (PII) never leaves the vehicle.

7 Future work

Future iterations will expand both sensing modality and fleet intelligence. Incorporating a low-cost 3-axis accelerometer synchronized with camera frames will provide a physical impact spike to verify pothole depth alongside visual area. Adding infrared (IR) illuminators will enable consistent night-time crack detection. Finally, federated learning across municipal vehicle fleets will allow the detector to continuously adapt to regional road textures without centralizing raw video footage.

8 Conclusion

RoadGuard AI demonstrates that combining lightweight deep learning (YOLOv8-Nano) with edge-first telemetry transforms ordinary fleet vehicles into continuous, automated road inspection

platforms. By processing frames locally at 38.7 FPS, buffering detections in a resilient SQLite queue, and deduplicating geotagged alerts on a municipal GIS dashboard, the system overcomes the bandwidth and cost barriers of traditional pavement surveys while providing actionable repair prioritization.

Appendix

Table 5 lists key hardware and firmware configuration parameters for reproducing the prototype.

Table 5. Key system and model parameters.

Parameter	Configured Value
Input image resolution	640 × 640 pixels (RGB)
Camera field of view	120° wide-angle, 30 FPS
GPS update rate	10 Hz (NMEA-0183)
Confidence threshold	0.50 (NMS IoU = 0.45)
Speed gating threshold	5.0 km/h minimum
Spatial cluster radius	5.0 meters (Geohash-9)
Local offline buffer	SQLite WAL mode (max 50k rows)
Cloud sync interval	5 s batch polling

Acknowledgment

The author sincerely thanks Prof. Archana Pawar for her invaluable guidance, technical feedback, and continuous encouragement throughout this research, as well as the Department of AI&ML, Navsahyadri Group of Institutions, Pune, for providing the computing infrastructure and laboratory facilities.

References

[1] Loprencipe G, Pantuso A, Di Mascio P. Sustainable pavement management system in urban areas considering the vehicle operating costs. *Sustainability*. 2017;9(3):453.

[2] Kim T, Ryu SK. Review and analysis of pothole detection methods. *J Emerg Trends Comput Inf Sci*. 2014;5(8):603–608.

[3] Maeda H, Sekimoto Y, Seto T, Kashiyama T, Omata H. Road damage detection and classification using deep neural networks with smartphone images. *Comput-Aided Civil Infrastruct Eng*. 2018;33(12):1127–1141.

[4] Zhang A, et al. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Comput-Aided Civil Infrastruct Eng*. 2017;32(10):805–819.

[5] Eriksson J, et al. The pothole patrol: using a mobile sensor network for road surface monitoring. In: *Proc 6th Int Conf Mobile Syst, Appl, Services (MobiSys)*. 2008:29–39.

[6] Koch C, Brilakis I. Pothole detection in asphalt pavement images. *Adv Eng Inform*. 2011;25(3):507–515.

[7] Arya D, et al. RDD2022: a multi-national image dataset for automatic road damage detection. *Geosci Data J*. 2024;11(4):846–862.

[8] Arya D, et al. Deep learning-based road damage detection and classification for multiple countries. *Autom Constr*. 2021;132:103935.

[9] Ren S, He K, Girshick R, Sun J. *Faster R-CNN: towards real-time object detection with region proposal networks*. IEEE Trans Pattern Anal Mach Intell, Jun 2017. ;39(6):1137–1149.

[10] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: unified, real-time object detection. In: *Proc IEEE Conf Comput Vis Pattern Recognit (CVPR)*. 2016:779–788.

[11] Jocher G, Chaurasia A, Qiu J. *Ultralytics YOLOv8* [GitHub repository]. 2023. Available from: https://github.com/ultralytics/ultralytics

[12] Zheng Z, et al. Distance-IoU loss: faster and better learning for bounding box regression. In: Proc AAAI Conf Artif Intell, vol 34, no 7. 2020:12993–13000.

[13] Fielding RT, Reschke J. *Hypertext Transfer Protocol (HTTP/1.1): message syntax and routing*. RFC 7230. IETF; 2014 Jun.