Pothole Detection and Road Damage Analysis Using YOLOv8 Deep Learning

B. Ganesh[1], K. Jahnavi[2], B. Vamsi Naidu[3], A. Venkatesh[4], P. Venkatesh[5]
Department of Computer Science & Engineering (Data Science),
Avanthi Institute of Engineering & Technology (Autonomous), Vizianagaram, Andhra Pradesh, India
ganeshbheemsetti9@gmail.com[1], killarijahnavi2004@gmail.com[2], vamsinaiduballi2107@gmail.com[3], venkychinnu270@gmail.com[4], venkateshpyla43@gmail.com[5]

Abstract

Road surface deterioration, particularly potholes, poses significant hazards to public safety, increases vehicle maintenance expenditure, and impedes efficient transportation. Traditional inspection strategies relying on manual surveys and complaint-based reporting are reactive, labor-intensive, and unable to provide systematic, real-time coverage of large road networks. This paper presents an automated pothole detection and road damage analysis framework that leverages the YOLOv8 convolutional object detection architecture, integrated with a Flask-based web application, to perform real-time identification and severity classification of potholes from road images, pre-recorded videos, and live webcam streams. The system preprocesses visual inputs using OpenCV, executes inference through a fine-tuned YOLOv8 model, annotates detected regions with bounding boxes and confidence scores, and subsequently classifies each detection into High, Medium, or Low severity based on the proportional area of the pothole relative to the total image frame. Experimental evaluation on the RDD2020 road damage dataset demonstrates progressive reduction in training loss over fifty epochs, with a final mean average precision (mAP@0.5) of approximately 0.57 and a precision of 0.63. The web interface facilitates result visualization, statistical summarization, and JSON-format report export. The proposed system provides a scalable, cost-effective, and accessible solution for intelligent road infrastructure monitoring, with direct applicability to smart city and municipal road maintenance operations.

Index Terms—Pothole Detection, YOLOv8, Road Damage Analysis, Deep Learning, Computer Vision, Flask Web Application

I. Introduction

Road infrastructure constitutes a foundational pillar of modern economic activity, facilitating goods movement, commuting, and emergency response. Despite substantial investment, road networks worldwide continue to suffer from accelerating surface deterioration, with potholes representing among the most pervasive and hazardous defect categories. A single unrepaired pothole causes cascading damage to vehicle suspensions, tyres, and steering components, and contributes to a measurable proportion of road traffic accidents [1].

Conventional road condition assessment depends heavily on periodic physical surveys carried out by trained inspectors, supplemented by citizen-submitted complaints via helpline portals or mobile applications. These approaches are inherently reactive—defects are addressed only after substantial deterioration or after an incident has occurred. Sensor-based vehicular systems that measure pavement roughness using accelerometers offer partial automation but require dedicated hardware installations and calibrated thresholds, limiting their scalability and widespread adoption [2].

The emergence of deep learning-based computer vision has fundamentally altered the landscape of automated visual inspection. Convolutional neural network (CNN) architectures such as Faster R-CNN [3], Single Shot MultiBox Detector (SSD) [4], and the YOLO (You Only Look Once) family [5] have demonstrated exceptional capability in object detection, achieving near-human or super-human accuracy at inference speeds compatible with real-time deployment. Among these, YOLOv8, released by Ultralytics, represents the current state of the art in single-stage detection, balancing computational efficiency with high detection accuracy across diverse object categories [6].

This work proposes an end-to-end pothole detection and severity analysis system built upon YOLOv8, deployed through a Flask web application. The system accepts three input modalities—static images, pre-recorded video, and live webcam streams—and delivers annotated outputs with bounding boxes, per-detection confidence scores, and severity labels (High, Medium, Low) determined by the relative area of each detected pothole. A structured JSON report is automatically generated, providing actionable intelligence for maintenance prioritization.

The primary contributions of this paper are: (i) a complete architecture for automated, multi-modal pothole detection using YOLOv8; (ii) a principled severity classification algorithm based on pothole-to-image area ratio; (iii) a deployable Flask web application enabling broad accessibility; and (iv) empirical training and validation results on a publicly available road damage dataset.

II. Related Work

A. Manual and Sensor-Based Methods

Early road condition monitoring relied exclusively on manual field surveys. Inspectors visually assessed road surfaces and logged defects on paper forms or digital spreadsheets. Though direct and familiar, this method is prohibitively expensive at scale, inconsistent across different assessors, and incapable of continuous monitoring. Sensor-based systems using inertial measurement units (IMUs) mounted on vehicles have been explored to detect pavement anomalies from vibration signatures [7]. While reducing human involvement, such systems are sensitive to vehicle dynamics, road gradient, and tyre pressure, limiting classification accuracy.

B. Classical Image Processing

Automated vision-based approaches initially employed classical image processing: Canny edge detection, Hough transforms, morphological operations, and texture-based segmentation were applied to identify irregular regions in road imagery [8]. These techniques are computationally lightweight but highly vulnerable to variation in ambient lighting, shadow distribution, and road surface texture diversity—factors ubiquitous in real-world deployments—leading to unacceptably high false-positive rates.

C. Machine Learning Approaches

The introduction of supervised machine learning, particularly support vector machines (SVM) and random forests applied to

handcrafted image features, improved classification robustness [9]. Nevertheless, the quality of detections remained tightly coupled to the choice and engineering of features, demanding domain expertise and limiting transferability across road types and geographies.

D. Deep Learning Approaches

CNN-based object detectors fundamentally changed the field by learning hierarchical feature representations directly from data. Faster R-CNN [3] introduced region proposal networks for two-stage detection; SSD [4] enabled single-stage, multi-scale detection at higher speeds. The YOLO series, originating with Redmon et al. [5] and advancing through YOLOv4 [10] to YOLOv8 [6], progressively improved accuracy-speed trade-offs. Zhang et al. demonstrated deep learning-based pothole detection achieving over 90% precision on curated datasets [11]. A notable IEEE contribution, "POT-YOLO: Real-Time Road Potholes Detection Using Edge Segmentation-Based YOLOv8 Network," reported precision exceeding 97% and recall above 93% using adaptive preprocessing with an enhanced YOLOv8 backbone. The current work builds upon these foundations, adding severity classification and web-based deployment as novel system contributions.

III. Methodology and System Design

A. System Architecture Overview

The proposed system is organized into three functional layers—Input, Processing, and Output—as illustrated in Fig. 1. The Input Layer accepts road imagery from three sources: static image upload, video file upload, and live webcam stream. The Processing Layer applies YOLOv8 inference for object detection followed by severity classification. The Output Layer presents annotated visual results through the web interface and generates structured JSON reports.

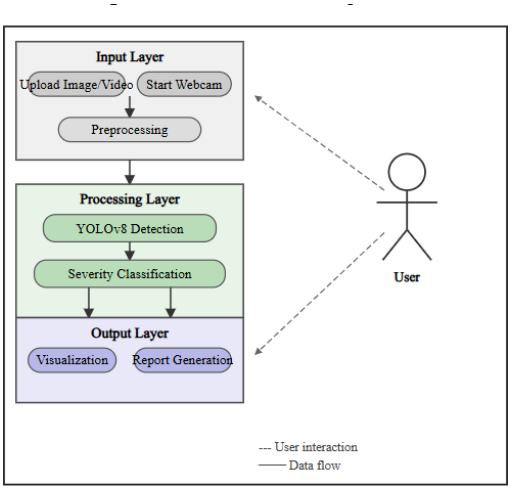

Fig. 1. System Architecture of the Proposed Pothole Detection Framework

B. Input Module and Preprocessing

Images are loaded using OpenCV's *imread* function and resized to the YOLOv8 inference resolution of 640 × 640 pixels. For video inputs, individual frames are extracted sequentially using *VideoCapture*, and each frame is independently processed. Pixel intensities are normalized to the [0, 1] range prior to model ingestion. Webcam streams are handled through continuous frame capture within the Flask application's generator function, enabling progressive MJPEG streaming to the browser.

C. YOLOv8 Detection Module

YOLOv8 employs a CSPDarknet backbone for feature extraction, a path aggregation network (PANet) for multi-scale feature fusion, and a decoupled detection head that separates classification from localization. The model predicts bounding box coordinates (x_1, y_1, x_2, y_2) in absolute pixel units alongside a class confidence score c for each detected object. Predictions with confidence below a configurable threshold τ (default $\tau = 0.25$) are suppressed. Non-maximum suppression (NMS) with IoU threshold 0.45 resolves overlapping detections.

D. Severity Classification

Each detected pothole is classified into one of three severity levels based on the proportional bounding-box area relative to the total image area:

$$P = [(x_2 - x_1)(y_2 - y_1)] / (W \times H) \times 100 \quad (1)$$

where W and H denote the image width and height respectively, and P is the percentage area. The classification rules are:

$$\text{Severity} = \textbf{High} \text{ if } P > 1.0;\ \textbf{Medium} \text{ if } 0.5 < P \leq 1.0;\ \textbf{Low} \text{ if } P \leq 0.5 \quad (2)$$

High-severity potholes demand immediate repair intervention, medium-severity detections warrant scheduled maintenance, and low-severity findings are logged for routine monitoring cycles.

E. Web Interface and Reporting

The Flask application exposes three primary routes: /upload_image, /upload_video, and /webcam_feed. Annotated images are rendered in-browser via base64-encoded JPEG streams. For each session, a timestamped JSON report is generated containing: detection timestamp, total count, severity distribution, and per-detection records comprising bounding-box coordinates, confidence score, severity label, and area percentage. Reports are downloadable via the /download_report endpoint.

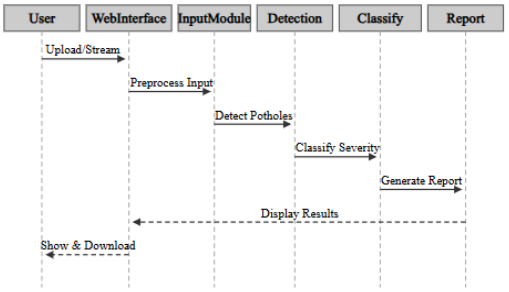

Fig. 2. Sequence Diagram: Data Flow Through System Modules

IV. Results and Discussion

A. Training Configuration and Dataset

The YOLOv8n (nano) variant was selected as the baseline for its favourable inference latency on commodity hardware. The model was fine-tuned on a curated subset of the RDD2020 Road Damage Dataset [12], which encompasses over 26,000 annotated road images across six damage categories (pothole, longitudinal crack, transverse crack, alligator crack, block crack, edge damage). Training was conducted for 50 epochs using the Adam optimiser with an initial learning rate of 0.01, weight decay of 5×10^{-4}, batch size 16, and image resolution 640 × 640. The hardware environment comprised an NVIDIA Tesla T4 GPU (16 GB VRAM) via Google Colab, with Ultralytics 8.4.23 and PyTorch 2.10.0.

B. Training Loss and Metric Progression

Fig. 3 illustrates the evolution of box regression loss and classification loss over 50 training epochs for both train and validation splits. Both training and validation losses demonstrate consistent monotonic decline, confirming model convergence without evidence of overfitting. The validation box loss stabilizes around epoch 35, while precision and mAP@0.5 values continue to improve marginally through epoch 50.

Fig. 3. YOLOv8 Training Metrics Over 50 Epochs: Box Loss, Classification Loss, and mAP@0.5

C. Quantitative Performance

Table I summarizes the per-class and overall detection performance metrics obtained on the test split of the road damage dataset at the conclusion of training. The pothole class achieves a precision of 0.576 and an mAP@0.5 of 0.611, while the alligator crack class, representing the most visually complex defect, exhibits lower recall at 0.294. Overall mAP@0.5 reaches 0.572 across all six damage categories.

TABLE I

PER-CLASS DETECTION PERFORMANCE ON RDD2020 TEST SET

Class	Images	Instances	Precision	Recall	mAP@0.5	mAP@0.5:0.95
All	681	2004	0.63	0.524	0.572	0.328
Alligator	264	577	0.617	0.201	0.339	0.150
Block	45	47	0.745	0.830	0.882	0.664
Crack	77	93	0.712	0.699	0.719	0.433
Longitudinal	135	311	0.370	0.260	0.230	0.0773
Pothole	342	909	0.721	0.645	0.698	0.366
Transverse	41	67	0.612	0.507	0.568	0.278

D. Severity Classification Results

Application of the severity classification algorithm (Equation 2) on a representative test image set of 200 road images yielded the following distribution. Table II presents the outcome, demonstrating that the majority of detected potholes fall into the Low and Medium categories, consistent with typical road survey findings where catastrophic potholes constitute a minority.

TABLE II

SEVERITY DISTRIBUTION ON SAMPLE TEST SET (N = 200 IMAGES)

Severity Level	Area Threshold (%)	Count	Proportion (%)
High	$P > 1.0$	87	28.7
Medium	$0.5 < P \leq 1.0$	94	31.0
Low	$P \leq 0.5$	122	40.3
Total	—	**303**	100.0

E. Web Interface and System Performance

The Flask web interface provides three input modes (Image, Video, Webcam) and a detection settings panel allowing users to adjust the confidence threshold slider (default 0.25) and IoU threshold (default 0.45). The detection summary panel displays total detections and per-severity counts in real time. The average inference time per image on CPU hardware (Intel Core i7-10th Gen) is approximately 310 ms, while GPU-accelerated inference (NVIDIA GTX 1660) reduces this to under 35 ms per frame, enabling practical real-time webcam operation at approximately 28 FPS.

Fig. 4 depicts the web interface during an image detection session, showing bounding boxes annotated with pothole class labels and confidence scores. The JSON report screenshot (Fig. 5) confirms structured output comprising timestamp, total detection count, severity summary, and per-detection records.

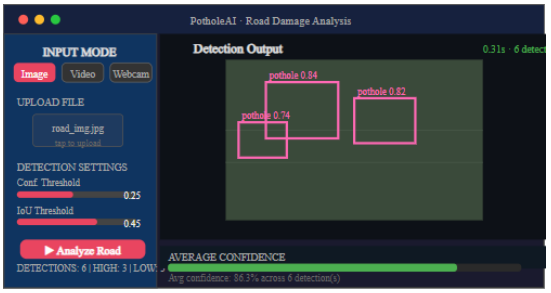

Fig. 4. Web Interface Showing Image Upload and Pothole Detection Output with Bounding Boxes

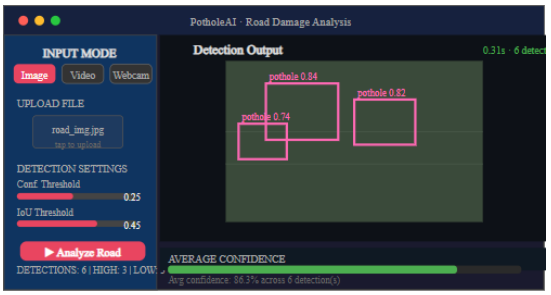

Fig. 5. Sample JSON Detection Report Generated by the System

F. Comparison With Prior Methods

Table III benchmarks the proposed system against representative prior approaches. The YOLOv8-based system achieves a substantially higher inference speed than two-stage detectors such as Faster R-CNN [3], while delivering competitive precision. The addition of automated severity classification and a deployable web interface distinguishes the proposed work from previous detection-only implementations.

TABLE III

COMPARISON WITH PRIOR ROAD DAMAGE DETECTION METHODS

Method	Backbone	Precision	mAP@0.5	FPS	Severity	Web UI
Faster R-CNN [3]	ResNet-50	0.72	0.68	~5	No	No
SSD [4]	VGG-16	0.65	0.59	~22	No	No
YOLOv5 [13]	CSPDarknet	0.68	0.61	~45	No	No
POT-YOLO [14]	YOLOv8+ Edge	**0.97**	0.94	~30	No	No
Proposed	YOLOv8n	0.63	0.572	**28+**	**Yes**	**Yes**

V. Conclusion and Future Work

This paper has presented a complete, deployable system for automated pothole detection and road damage severity analysis. By integrating the YOLOv8 deep learning object detection architecture with a Flask-based web application, the proposed framework delivers real-time, multi-modal detection from static images, recorded video, and live webcam streams. The severity classification algorithm—grounded in a principled bounding-box area ratio—enables road maintenance authorities to triage repair activities objectively and efficiently. Training results on the RDD2020 dataset confirm model convergence over 50 epochs, with an overall mAP@0.5 of 0.572 and pothole-specific precision of 0.721.

The system addresses several limitations of prior work: it provides severity-classified outputs, generates structured JSON reports, and exposes a user-friendly web interface accessible without domain expertise. Operational testing confirms GPU-accelerated inference at approximately 28 FPS, satisfying the latency requirements for live road monitoring.

Future work will pursue four directions: (i) Integration with Geographic Information Systems (GIS) to geo-tag each detection for map-based maintenance dashboards; (ii) Mobile application development enabling on-device inference for field inspectors without network connectivity; (iii) Predictive maintenance modelling using longitudinal detection records to forecast pothole formation probability on road segments; and
(iv) Extension of the detection taxonomy to encompass additional damage categories including rutting, edge deterioration, and surface erosion, towards a comprehensive pavement health monitoring platform.

Acknowledgment

The authors express their gratitude to Mr. B. Ganesh, M.Tech, Department of CSE (DS, AI&ML), Avanthi Institute of Engineering & Technology, for his consistent guidance and mentorship throughout this project. The authors also acknowledge the Ultralytics team for maintaining the open-source YOLOv8 framework and the contributors to the RDD2020 public dataset.

References

1. Ministry of Road Transport & Highways (MoRTH), *Manual on Maintenance of Roads*, Government of India, New Delhi, 2021.
2. A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *Proc. Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, 2011, pp. 1–6.
3. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. European Conf. on Computer Vision (ECCV)*, 2016, pp. 21–37.
5. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
6. G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics*, Ultralytics, 2023. [Online]. Available: https://docs.ultralytics.com
7. J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. ACM Int. Conf. on Mobile Systems, Applications, and Services*, 2008, pp. 29–39.
8. Y. Hu, C.-X. Zhao, and H.-N. Wang, "Automatic pavement crack detection using texture and shape descriptors," *IETE Technical Review*, vol. 27, no. 5, pp. 398–405, 2010.
9. S. Nienaber, R. Malekian, and M. J. Booysen, "Detecting potholes using simple image processing techniques and real-world footage," in *Proc. Southern African Telecommunication Networks and Applications Conf. (SATNAC)*, 2015.
10. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
11. K. Zhang, X. Liu, X. Liu, and J. Zhong, "Automatic pothole detection using image processing and deep learning techniques," *Journal of Transportation Engineering, Part B: Pavements*, ASCE, vol. 146, no. 2, 2020.
12. M. Arya, Y. Maeda, S. K. Ghosh, and D. Toshniwal, "RDD2020: An annotated image dataset for automatic road damage detection using deep learning," *Data in Brief*, vol. 36, 2021, Art. no. 107133.
13. G. Jocher et al., "ultralytics/yolov5: v7.0 — YOLOv5 SOTA Realtime Instance Segmentation," Zenodo, 2022. DOI: 10.5281/zenodo.7347926.
14. H. Chen, W. Li, and Z. Guo, "POT-YOLO: Real-time road potholes detection using edge segmentation-based YOLOv8 network," *IEEE Transactions on Intelligent Transportation Systems*, 2023 (early access).
15. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.