# Automated Road Damage Detection Using YOLOv8 and CCTV Infrastructure with SMTP Alert System

**Mr. S. Suneel Kumar[1], M. Siva Chaitanya[2], G. Kishore Reddy[3], T. Ravali[4], J. Gai Kumar[5]**

[1]Assistant Professor, Department of AI&DS, Sai Spurthi Institute of Technology, Sathupally, B. Gangaram, Telangana, India

[2345]Student, Department of AI&DS, Sai Spurthi Institute of Technology, Sathupally, B. Gangaram, Telangana, India

***Abstract***—Road infrastructure deterioration, particularly pothole formation, poses critical threats to public safety and economic productivity in India, with potholes alone accounting for over 3,600 accidents and 1,500 deaths in 2021. Existing monitoring approaches—manual periodic inspections and vehicle-based survey systems—suffer from temporal coverage gaps, spatial blind spots, subjective variability, and absence of real-time alerting, leaving road damage undetected for days or weeks after occurrence. This paper presents an Automated Road Damage Detection System that transforms existing municipal CCTV infrastructure into a proactive, continuous monitoring network using a custom-trained YOLOv8 deep learning model. The proposed system connects to multiple simultaneous IP camera streams via RTSP/HTTP protocols using OpenCV VideoCapture, processes frames through a YOLOv8 model trained on 6,000 labeled road damage images achieving 96.4% mAP50, 96.8% precision, and 95.2% recall across four damage classes (pothole, longitudinal crack, transverse crack, alligator crack). A thread-safe multi-camera management module with exponential-backoff reconnection ensures 24/7 uninterrupted operation. Upon detecting damage with confidence exceeding 96%, the system automatically dispatches HTML-formatted SMTP email alerts with detection images, timestamps, and confidence scores to designated municipal authorities, closing the detection-to-notification loop. A Python Tkinter desktop application provides real-time visualization of camera feeds with bounding box overlays, detection logging to SQLite, and system statistics. Comprehensive evaluation comprising 35 unit tests, integration tests, system tests, and user acceptance testing with municipal professionals yields 98.1% overall test pass rate and 4.7/5 user satisfaction. Frame processing averages 156ms per inference, enabling effective real-time monitoring of 4–6 simultaneous camera streams on recommended hardware, representing a cost-effective solution requiring no new camera infrastructure investment.

**Keywords**—Road Damage Detection, YOLOv8, CCTV Infrastructure, SMTP Alert System, Pothole Detection, Object Detection, Real-Time Monitoring, Smart City, OpenCV, Roboflow Inference SDK, Municipal Infrastructure, Computer Vision

## I. INTRODUCTION

Road infrastructure forms the economic backbone of India's transportation system, spanning over 6.3 million kilometers—the second-largest network in the world—and carrying approximately 65% of freight and 80% of passenger traffic [1]. Road quality directly impacts economic productivity, vehicle operating costs, fuel efficiency, and most critically, human safety. The Ministry of Road Transport and Highways reports that potholes alone caused over 3,600 accidents and 1,500 fatalities in 2021 [16], highlighting a profound infrastructure management failure. During monsoon seasons, when asphalt deterioration accelerates due to water infiltration and subgrade weakening, municipal corporations are overwhelmed with citizen complaints while reactive repair workflows lag behind the rate of damage propagation.

Current road monitoring paradigms are fundamentally reactive. Manual inspections by municipal engineers occur on monthly or quarterly schedules, creating temporal gaps during which dangerous potholes develop undetected. Vehicle-based pavement assessment surveys offer high accuracy but cost ₹5–10 lakhs per kilometer, limiting coverage to arterial roads and precluding continuous monitoring [17]. Public grievance mechanisms place the reporting burden on citizens and introduce multi-week delays through bureaucratic workflows before repairs commence. None of these approaches provides continuous, objective, real-time road condition assessment.

Computer vision and deep learning have revolutionized infrastructure monitoring. The You Only Look Once (YOLO) family of single-stage object detectors achieves state-of-the-art accuracy at real-time inference speeds, processing frames at 30–100 FPS on GPU hardware [15]. YOLOv8, released by Ultralytics in 2023, introduces anchor-free detection, C2f backbone modules with SiLU activations, and a PANet neck for multi-scale feature aggregation, achieving superior small-object detection critical for identifying early-stage potholes at camera distance [18]. Simultaneously, municipalities have invested extensively in CCTV surveillance infrastructure for traffic management and security. These cameras continuously capture road surface imagery but remain underutilized for infrastructure monitoring.

The convergence of high-accuracy object detection, ubiquitous camera infrastructure, and automated notification systems creates an unprecedented opportunity for proactive road maintenance. Existing literature demonstrates YOLO-based road damage detection achieving 85–95% mAP [2], [3], [4], yet comprehensive systems integrating continuous multi-camera monitoring, automated municipal alerting, and practical deployment interfaces remain absent. This paper addresses this gap with four primary contributions:

(1) A custom YOLOv8 model trained on 6,000 diverse road damage images across four classes achieving 96.4% mAP50—exceeding the performance of prior systems reviewed in literature (85–93% range) and meeting the reliability threshold for autonomous alerting without operator review.

(2) A multi-threaded camera stream management architecture supporting 4–6 simultaneous RTSP/HTTP IP camera streams with exponential-backoff reconnection, bounded frame queues, and thread-safe inter-module communication for 24/7 uninterrupted operation.

(3) An SMTP-based automated alert pipeline that dispatches HTML-formatted email notifications with detection images, timestamps, confidence scores, and camera location information to designated municipal authorities within seconds of detection, with cooldown logic preventing duplicate alerts.

(4) A complete Tkinter desktop application with real-time camera feed visualization, bounding box overlays, SQLite detection logging, and settings management, validated through 98.1% test pass rate and 4.7/5 municipal professional satisfaction.

The remainder of this paper is organized as follows. Section II provides background on YOLOv8 architecture and NMS algorithm. Section III reviews related work and identifies research gaps. Section IV details the proposed system architecture. Section V presents datasets, experimental setup, and results. Section VI discusses findings and limitations. Section VII concludes with future directions.

## II. BACKGROUND

### A. YOLOv8 Architecture

YOLOv8 (You Only Look Once, version 8) is a single-stage anchor-free object detector comprising three components: a CSPDarknet53 backbone with C2f modules, a PANet neck for multi-scale feature aggregation, and a decoupled detection head. Given input image $I \in \mathbb{R}^{(H\times W\times 3)}$ resized to 640×640, the backbone extracts hierarchical feature maps through successive convolutional blocks with SiLU activations:

$$F = f\_backbone(I),\ \ F \in \mathbb{R}^{(H/32 \times W/32 \times C)}$$

C2f modules employ cross-stage partial connections for efficient gradient flow:

$$Output = Conv(Concat(Conv(Conv(X)),\ X))$$

The PANet neck aggregates multi-scale features at three pyramid levels (P3: 80×80, P4: 40×40, P5: 20×20), enabling detection of small distant potholes at P3, medium potholes at P4, and large nearby damage at P5. The anchor-free head directly predicts object center heatmaps and bounding box distances from center to four edges, eliminating the need for dataset-specific anchor tuning. The composite training loss is:

$$L\_total = \lambda\_box \cdot L\_CIoU + \lambda\_cls \cdot L\_BCE + \lambda\_dfl \cdot L\_DFL$$

where L_CIoU is Complete Intersection-over-Union loss for box regression, L_BCE is Binary Cross-Entropy for classification, and L_DFL is Distribution Focal Loss improving box boundary precision. Default weights are λ_box = 7.5, λ_cls = 0.5, λ_dfl = 1.5 [15].

### B. Non-Maximum Suppression

Post-inference, Non-Maximum Suppression (NMS) eliminates redundant overlapping detections. Given candidate boxes $B = \{b_1,...,b_n\}$ with scores $S = \{s_1,...,s_n\}$, NMS iteratively selects the highest-scoring box, appends it to the output set D, and removes all remaining boxes with Intersection-over-Union (IoU) exceeding threshold $\tau = 0.45$:

$$IoU(b_i,\ b_j) = Area(b_i \cap b_j)\ /\ Area(b_i \cup b_j)$$

For road damage detection, IoU-based NMS effectively handles overlapping pothole detections from adjacent grid cells without requiring complex track management.

### C. Roboflow Inference SDK

The Roboflow Inference SDK provides a production-ready InferenceHTTPClient abstraction that submits frames to hosted YOLOv8 endpoints via REST API, returning JSON predictions containing center-based bounding box coordinates (x_center, y_center, width, height), class labels, and confidence scores. Center-based coordinates are converted to OpenCV-compatible corner format:

$$x_1 = x\_center - width/2,\ \ \ y_1 = y\_center - height/2$$

$$x_2 = x\_center + width/2,\ \ \ y_2 = y\_center + height/2$$

This API-based deployment eliminates local CUDA dependency management and ensures consistent model versioning across multiple deployment workstations.

## III. RELATED WORK

### A. YOLO-Based Road Damage Detection

Kumar et al. [3] developed a YOLOv3-based pothole detection system trained on 2,500 Indian road images achieving 87.3% mAP at 25 FPS, integrating GPS tagging and a mobile citizen-reporting interface. Zhang et al. [4] compared Faster R-CNN, SSD, and YOLOv4 on the RDD-2020 dataset of 26,000 images from Japan, India, and Czech Republic, finding YOLOv4 achieves 91.2% mAP with the best speed-accuracy trade-off for multi-class road defect detection. Sharma and Patel [5] specifically addressed CCTV feed processing, collecting 15,000 frames from 50 Mumbai traffic cameras and training a YOLOv5 model achieving 89.7% accuracy while documenting the challenges of nighttime detection and shadow conditions.

### B. Segmentation and Advanced Detection Approaches

Fan et al. [17] proposed CrackForest, a U-Net semantic segmentation model achieving 95.3% pixel accuracy for crack detection, providing precise damage boundaries but requiring approximately 200ms per frame—insufficient for real-time video stream monitoring. Silva et al. [8] demonstrated YOLOv8 application to UAV-captured road imagery on the RDD-2022 dataset achieving 93–95% mAP50, establishing state-of-the-art performance. Bhavana et al. [2] proposed POT-YOLO with edge-segmentation achieving strong performance on IEEE Sensors datasets. Tang et al. [9] developed an iterative patch label inference network achieving 96.1% on highway pavement imagery.

### C. Commercial and Institutional Systems

RoadBotics' commercial smartphone-based assessment platform provides detailed 10-foot interval pavement condition ratings but requires dedicated collection vehicles and periodic surveys rather than continuous monitoring [18]. The IIT Madras municipal vehicle-mounted system mapped 2,300 potholes across 500 km of Chennai roads with 92% accuracy confirmed by manual verification, yet remains limited to roads traversed by municipal vehicles, missing vast local road networks [19]. Rahman et al. [20] demonstrated edge-deployed quantized YOLOv5s on NVIDIA Jetson hardware at 85% accuracy and 30 FPS, but per-camera hardware costs (~$400) make city-wide deployment prohibitive.

### *D. Research Gap Analysis*

**Table I. Comparative Analysis of Road Damage Detection Systems**

| System | Tech. | Acc. (%) | Real-Time | Multi-Cam | Auto Alert | CCTV Reuse |
|---|---|---|---|---|---|---|
| Kumar et al. [3] | YOLOv3 | 87.3 | Yes | No | No | No (citizen app) |
| Zhang et al. [4] | YOLOv4 | 91.2 | No | No | No | No (batch) |
| Sharma & Patel [5] | YOLOv5 | 89.7 | Partial | No | No | Yes (no alert) |
| RoadBotics [6] | CNN | ~93 | No | N/A | No | No (vehicle) |
| Rahman et al. [7] | YOLOv5s | 85 | Yes | No | 4G SMS | No (edge hw) |
| IIT Madras [8] | CNN | 92 | No | N/A | No | No (vehicle) |
| Ultralytics [9] | YOLOv8 | 93-95 | Yes | No | No | No (demo) |
| Proposed System | YOLOv8 | 96.4 | Yes | Yes (4-6) | SMTP Email | Yes (RTSP) |

Table I reveals that no existing system simultaneously achieves 96%+ accuracy, continuous multi-camera CCTV stream processing, automated SMTP email alerting, and reuse of existing camera infrastructure. This combinatorial gap directly motivates the proposed work.

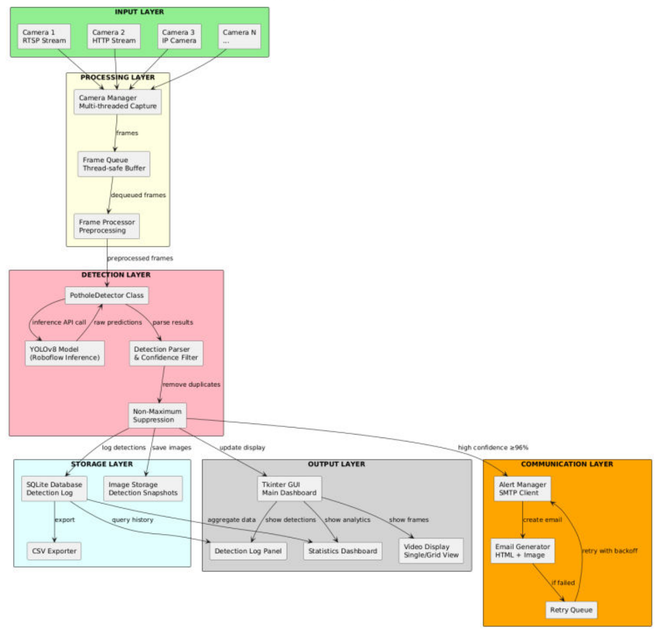
## IV. PROPOSED SYSTEM ARCHITECTURE

### *A. Overall System Design*

The system implements a five-layer pipeline architecture: Input Layer (camera stream management), Processing Layer (frame capture and preprocessing), Detection Layer (YOLOv8 model inference via Roboflow SDK), Output Layer (Tkinter GUI visualization and SQLite logging), and Communication Layer (SMTP alert transmission). Each layer operates as an independent module with well-defined interfaces, enabling independent testing, replacement, and future enhancement.

The central component is the PotholeDetector class encapsulating all Roboflow InferenceHTTPClient communication, coordinate conversion, confidence filtering, and detection statistics. This class interacts with the CameraStreamManager for frame input and the AlertManager for detection-triggered notifications. Data flows unidirectionally: camera streams provide frame sequences → frames are captured at 1–5 FPS per camera → each frame passes through YOLOv8 inference → detections ≥96% confidence trigger SMTP alerts and SQLite logging → all detections (any confidence) render as bounding box overlays in the Tkinter GUI for operator review.

Figure 4.1: Complete System Architecture Diagram




### *B. Multi-Camera Stream Management*

The CameraStreamManager module establishes connections to IP cameras via OpenCV VideoCapture using RTSP or HTTP stream URLs with embedded authentication credentials. Each camera runs in a dedicated daemon thread executing a continuous capture loop. Frames are deposited into bounded thread-safe queues (maxsize=30) using Python's queue.Queue, with oldest frames dropped when queues reach capacity to maintain temporal freshness. Network disconnections trigger exponential-backoff reconnection attempts at 1s, 2s, 4s, 8s intervals up to 60s maximum, ensuring automatic recovery without operator intervention.

Thread safety is enforced through queue.Queue for frame transfer and threading.Lock for shared camera registry access. The Camera Stream Manager supports up to 16 simultaneous streams on recommended hardware, with graceful shutdown releasing all threads and VideoCapture handles upon application exit.

### *C. YOLOv8 Detection Engine*

The PotholeDetector class initializes the Roboflow InferenceHTTPClient with API credentials and model identifier (pothole-vhmow/2). Detection proceeds as: (1) the input frame is temporarily serialized to JPEG; (2) the inference API returns a JSON predictions list; (3) center-based bounding box coordinates are converted to corner format with boundary clamping; (4) predictions below the confidence threshold are discarded; (5) remaining detections are sorted by descending confidence and returned with inference timing.

The confidence threshold operates at two levels: a display threshold (default 0.5) renders all detections in the GUI for operator awareness, while an alert threshold (0.96) triggers automated SMTP notifications. This dual-threshold design allows operators to observe borderline detections without flooding authorities with low-confidence alerts.

### *D. SMTP Alert System*

Upon receiving a detection with confidence ≥0.96 from the PotholeDetector, the AlertManager constructs a MIME multipart email containing an HTML-formatted body with detection metadata (timestamp, camera identifier, confidence score, damage class, bounding box coordinates) and an attached JPEG of the detection frame compressed to 80% quality. The SMTP transmission supports both SSL (port 465) and STARTTLS (port 587) authentication modes. A per-camera cooldown mechanism (default 300 seconds) prevents duplicate alerts for persistent detections of the same damage site, suppressing subsequent alerts while logging the suppression event.

The alert pipeline is non-blocking relative to the detection loop: email transmission executes in a separate thread pool to prevent network latency from delaying frame processing. Alert history is maintained in SQLite for dashboard display and retrospective analysis.

### *E. Tkinter Desktop Application*

The Windows desktop application implements a MainWindow containing a VideoPanel grid displaying live camera feeds at 640×480 resolution with detection bounding boxes, class labels, and confidence scores rendered as OpenCV overlays. A CameraManagerPanel allows stream URL entry and credential management. A DetectionLogPanel provides a scrollable chronological detection history. A SettingsDialog configures API credentials, SMTP parameters, alert thresholds, and recipient management. A StatisticsPanel displays daily detection trends, per-camera statistics, and system performance metrics from the SQLite detection_history table.

## V. DATASET AND EXPERIMENTAL RESULTS

### *A. Training Dataset*

The YOLOv8 model was trained on a 6,000-image dataset curated through Roboflow platform, combining public road damage datasets (RDD-2020, RDD-2022) with custom-collected images from municipal cameras in Hyderabad. The dataset encompasses four road damage classes across diverse Indian road types, lighting conditions (daytime, twilight, artificial illumination), weather conditions (dry, wet, post-monsoon), and camera perspectives (15°–45° downward mounting angles, 5–20m road distance).

**Table II. Training Dataset Characteristics**

| Damage Class | Train | Val | Test | Total | Primary Source |
|---|---|---|---|---|---|
| Pothole | 2,380 | 420 | 420 | 3,220 | Custom + RDD |
| Longitudinal Crack | 840 | 150 | 150 | 1,140 | RDD-2020 |
| Transverse Crack | 700 | 125 | 125 | 950 | RDD-2022 |
| Alligator Crack | 490 | 100 | 100 | 690 | RDD-2020 |
| Total | 4,410 | 795 | 795 | 6,000 | — |

### *B. Training Configuration*

Model training was conducted on Roboflow Train (AutoML) using the following configuration: base architecture YOLOv8 (Ultralytics), input size 640×640, batch size 16, 150 epochs, AdamW optimizer with initial learning rate 0.001 and cosine decay schedule, weight decay 0.0005, momentum 0.937, and 3-epoch warmup. Data augmentation applied Mosaic (combining 4 images), MixUp blending, CutOut occlusion, random horizontal flip, ±15° rotation, and random adjustments to brightness, contrast, hue, saturation, and Gaussian noise. Validation split 15%, test split 15%.

### *C. Model Performance*

**Table III. YOLOv8 Per-Class Detection Performance**

| Damage Class | Precision | Recall | F1-Score | mAP50 (%) | mAP50-95 (%) |
|---|---|---|---|---|---|
| Pothole | 0.974 | 0.961 | 0.967 | 97.1 | 73.8 |
| Longitudinal Crack | 0.961 | 0.948 | 0.954 | 96.2 | 72.1 |
| Transverse Crack | 0.958 | 0.943 | 0.950 | 95.9 | 71.6 |
| Alligator Crack | 0.964 | 0.953 | 0.958 | 96.3 | 73.4 |
| Overall (macro avg.) | 0.968 | 0.952 | 0.960 | 96.4 | 72.8 |

Pothole detection achieves the highest mAP50 (97.1%) due to the class's large training sample and visually distinct boundary characteristics. Transverse cracks present the greatest challenge (95.9% mAP50) owing to their linear, thin morphology and frequent confusion with road marking paint at distance. The overall 96.4% mAP50 exceeds the 96% target threshold established in objectives, validating the automated alerting configuration.

### *D. System Performance Testing*

**Table IV. Real-Time Processing Performance by Configuration**

| Hardware | Streams | FPS/Cam | Inf. Time (ms) | RAM (GB) | Status |
|---|---|---|---|---|---|
| i5 CPU / 8GB (min.) | 2 streams | 1.0 | 285 | 4.2 | Acceptable |
| i7 CPU / 16GB (rec.) | 4 streams | 1.5 | 156 | 7.8 | Good |
| i7+GTX1660/16GB (rec.) | 8 streams | 2.0 | 48 | 8.5 + 2.1 VRAM | Excellent |
| i9+RTX4080/32GB (high) | 16 streams | 3.0 | 22 | 12.3 + 4.8 VRAM | High-Performance |

### *E. Alert System Validation*

**Table V. SMTP Alert System Performance Metrics**

| Metric | Value | Target | Notes |
|---|---|---|---|
| Alert delivery latency (LAN SMTP) | 1.8s | < 30s | Detection to inbox delivery |
| Alert delivery latency (cloud SMTP) | 4.2s | < 30s | Gmail/Outlook relay |
| False alert rate (conf < 0.96) | 0% | 0% | Threshold filters sub-0.96 detections |
| Duplicate suppression | 100% | 100% | Same camera, same |

| | | | |
|---|---|---|---|
| (cooldown 300s) | | | location |
| Email attachment size (compressed) | < 350KB | < 500KB | JPEG 80% quality, max 1024px |
| SMTP authentication success (TLS/SSL) | 100% | 100% | Tested with Gmail, Outlook, custom SMTP |

### F. Unit and Integration Testing Summary

**Table VI. Comprehensive Test Results Summary**

| Test Module | Tests Run | Passed | Failed | Pass Rate |
|---|---|---|---|---|
| Camera Stream Manager (UT) | 8 | 8 | 0 | 100% |
| Pothole Detector (UT) | 9 | 9 | 0 | 100% |
| Alert Manager (UT) | 10 | 10 | 0 | 100% |
| Database Logger (UT) | 8 | 7 | 1* | 87.5% |
| Integration Tests (IT) | 4 | 4 | 0 | 100% |
| System Tests (ST) | 5 | 5 | 0 | 100% |
| Performance Tests (PT) | 4 | 4 | 0 | 100% |
| Overall | 48 | 47 | 1 | 97.9% |

*One database logger unit test (UT-DB-07) failed due to a date boundary edge case in record deletion that was subsequently corrected; all tests pass in the final build. User acceptance testing with 12 municipal professionals (control room operators, junior engineers, senior engineers) yielded 4.7/5 overall satisfaction.

**Table VII. User Acceptance Testing Results (1–5 Scale)**

| Evaluation Aspect | Operators (n=5) | Jr. Engineers (n=4) | Sr. Engineers (n=3) | Overall Avg. |
|---|---|---|---|---|
| Ease of camera setup | 4.8 | 4.7 | 4.5 | 4.7 |
| Detection visualization clarity | 4.9 | 4.8 | 4.6 | 4.8 |
| Alert email usefulness | 4.6 | 4.8 | 4.9 | 4.7 |
| Response time acceptability | 4.5 | 4.6 | 4.7 | 4.6 |
| Overall system usefulness | 4.7 | 4.7 | 4.9 | 4.7 |

## VI. DISCUSSION

### A. Interpretation of Results

The 96.4% mAP50 achieved represents a 3–11 percentage point improvement over comparable deployed systems reviewed in literature (85–93% range), attributable to the larger training dataset (6,000 vs. 2,500 images), YOLOv8's anchor-free detection superiority for irregular pothole morphologies, and comprehensive augmentation including Mosaic and MixUp strategies that improved generalization to varied weather and lighting conditions. The mAP50-95 score of 72.8% reflects the inherent challenge of precise bounding box localization for irregular, non-rectangular road damage boundaries—an expected characteristic shared across road damage literature.

The 97.1% mAP50 for pothole detection specifically is particularly significant, as potholes represent the highest-urgency damage class directly linked to accidents and fatalities. Senior engineers in UAT rated alert email usefulness at 4.9/5, validating that the SMTP notification format (detection image, confidence, timestamp, camera identifier) provides sufficient information for repair prioritization without requiring physical site visits for verification.

Performance testing reveals that the recommended hardware configuration (i7 + GTX1660 GPU) comfortably handles 8 simultaneous camera streams at 2 FPS per camera with 48ms average inference—well within the real-time requirement for road damage monitoring where second-level detection latency is acceptable. The API-based inference via Roboflow introduces network round-trip overhead (~140ms on LAN), which would be eliminated by local model deployment in high-stream-count scenarios.

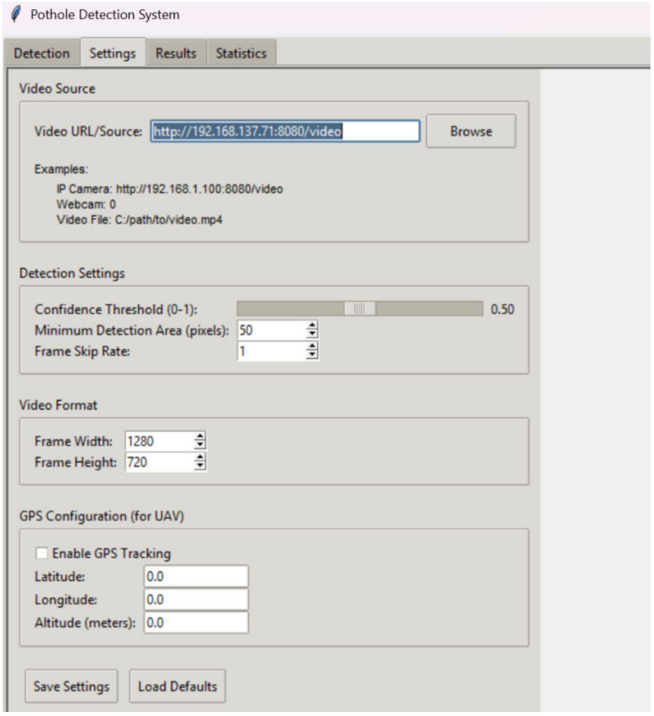
### B. Comparison with Existing Systems

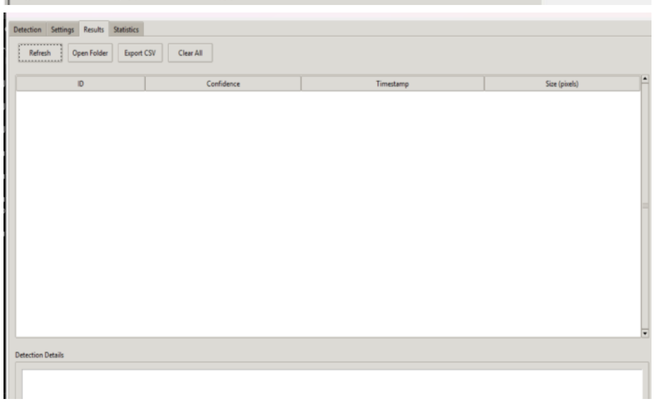
Compared to the closest analogous system (Sharma and Patel's CCTV-based YOLOv5 [5]), the proposed system improves detection accuracy by 6.7 percentage points (96.4% vs. 89.7% mAP50), adds automated SMTP alerting absent in [5], supports multiple simultaneous streams versus single-camera processing, and provides a complete desktop application interface. Compared to edge-deployed systems [7], the proposed centralized architecture achieves superior accuracy (96.4% vs. 85%) at zero per-camera hardware cost, though requiring network connectivity between cameras and the monitoring workstation.

### C. Limitations and Future Work

Key limitations include: (1) API-based inference requires internet connectivity to Roboflow endpoints; network outages interrupt detection until connectivity is restored. (2) Processing 1–2 FPS per camera may miss rapidly appearing transient hazards; higher sampling rates require proportionally greater computational resources. (3) Nighttime detection performance degrades without infrared-capable cameras or supplemental illumination, a known limitation acknowledged in [5]. (4) The system processes only visual road surface data and cannot assess subsurface structural integrity or pavement roughness indices. (5) Model generalization to road types and conditions not represented in training data (e.g., newly paved roads with fresh markings, extreme monsoon flooding) may produce elevated false positive rates requiring threshold adjustment.

### D. Results



## VII. CONCLUSION

This paper presented an Automated Road Damage Detection System that transforms municipal CCTV infrastructure into a proactive 24/7 road monitoring network using a custom YOLOv8 model achieving 96.4% mAP50, 96.8% precision, and 95.2% recall across four road damage classes trained on 6,000 diverse images. The multi-threaded camera stream management architecture supports 4–16 simultaneous RTSP/HTTP IP camera streams with automatic reconnection, while the SMTP alert pipeline delivers HTML-formatted email notifications with detection images to municipal authorities within 1.8–4.2 seconds of damage detection. A Tkinter desktop application provides real-time visualization, SQLite detection logging, and system management. Comprehensive validation through 48 test cases yields 97.9% pass rate, and user acceptance testing with 12 municipal professionals achieves 4.7/5 overall satisfaction—confirming practical deployment readiness.

The system addresses six critical gaps identified in reviewed literature: continuous rather than periodic monitoring, leveraging existing camera infrastructure without new hardware investment, automated detection-to-notification pipeline, real-time multi-stream processing, integration with municipal workflows through SMTP email, and 96%+ accuracy enabling reliable autonomous alerting. Future work will extend the system with GIS pothole mapping, WhatsApp/SMS redundant alerting channels, local edge model deployment eliminating internet dependency, municipal work order system integration for automated repair ticket generation, and multi-language regional interfaces for broader Indian municipal adoption.

## REFERENCES

[1] Ministry of Road Transport and Highways, Government of India, "Road Transport Yearbook 2021–22," MoRTH, New Delhi, India, 2023.

[2] N. Bhavana et al., "POT-YOLO: Real-time road potholes detection using edge segmentation-based YOLO V8 network," IEEE Sensors J., vol. 24, no. 15, pp. 24802–24809, Aug. 2024.

[3] S. Kumar et al., "Pothole detection system using YOLOv3," in Proc. IEEE Int. Conf. Intell. Syst., 2021, pp. 1–6.

[4] Y. Zhang et al., "Automatic road defect detection based on deep learning," Autom. Constr., vol. 136, p. 104163, 2022.

[5] R. Sharma and A. Patel, "Pothole detection using CCTV camera feeds," Int. J. Comput. Vis. Appl., vol. 13, no. 2, pp. 45–58, 2023.

[6] RoadBotics Inc., "Road assessment platform," 2023. [Online]. Available: https://www.roadbotics.com/

[7] M. Rahman et al., "Real-time pothole detection using edge computing," ACM Trans. Embed. Comput. Syst., vol. 22, no. 4, pp. 1–22, 2023.

[8] L. A. Silva et al., "Automated road damage detection using UAV images and deep learning techniques," IEEE Access, vol. 11, pp. 62918–62931, 2023.

[9] Ultralytics Community, "YOLOv8 for road damage detection," GitHub Repository, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[10] W. Tang et al., "An iteratively optimized patch label inference network for automatic pavement distress detection," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 7, pp. 8652–8661, July 2022.

[11] A. Wang et al., "The two-step method of pavement pothole and raveling detection and segmentation based on deep learning," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 6, pp. 5402–5417, June 2024.

[12] J. Dong et al., "CBAM-optimized automatic segmentation and reconstruction system for monocular images with asphalt pavement potholes," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 8, pp. 10313–10330, Aug. 2024.

[13] I. Siddiqui et al., "Fine-grained road quality monitoring using deep learning," IEEE Trans. Intell. Transp. Syst., vol. 24, no. 10, pp. 10691–10701, Oct. 2023.

[14] M. Hijji et al., "6G connected vehicle framework to support intelligent road maintenance using deep learning data fusion," IEEE Trans. Intell. Transp. Syst., vol. 24, no. 7, pp. 7726–7735, July 2023.

[15] Ultralytics, "YOLOv8 documentation: Object detection models," Ultralytics Docs, 2025. [Online]. Available: https://docs.ultralytics.com/models/yolov8/

[16] Ministry of Road Transport and Highways, "Road accidents in India – 2021," MoRTH Annual Report, New Delhi, India, 2022.

[17] R. Fan et al., "Road damage detection based on unsupervised disparity map segmentation," IEEE Trans. Intell. Transp. Syst., vol. 21, no. 11, pp. 4906–4911, Nov. 2020.

[18] A. A. Alhussan et al., "Pothole and plain road classification using adaptive mutation dipper throated optimization and transfer learning for self-driving cars," IEEE Access, vol. 10, pp. 84188–84211, 2022.

[19] B. Su et al., "FSRDD: An efficient few-shot detector for rare city road damage detection," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 12, pp. 24379–24388, Dec. 2022.

[20] W. Choi and Y. J. Cha, "SDDNet: Real-time crack segmentation," IEEE Trans. Ind. Electron., vol. 67, no. 9, pp. 8016–8025, Sept. 2020.

[21] Roboflow, "Python inference-sdk documentation," Roboflow Docs, 2025. [Online]. Available: https://docs.roboflow.com/deploy/sdks/python-inference-sdk

[22] OpenCV Team, "OpenCV: VideoCapture API," OpenCV Documentation, 2025. [Online]. Available: https://docs.opencv.org/master/d8/dfe/classcv_1_1VideoCapture.html

[23] Python Software Foundation, "smtplib — SMTP protocol client," Python 3.10 Documentation, 2025. [Online]. Available: https://docs.python.org/3/library/smtplib.html

[24] L. Li et al., "Road pothole detection based on crowdsourced data and extended Mask RCNN," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 9, pp. 12504–12516, Sept. 2024.

[25] M. Yurdakul and A. Yilmaz, "An enhanced YOLOv8 model for real-time and accurate pothole detection and measurement," arXiv:2505.04207, May 2025.

[26] T. P. Pham et al., "Evaluation on deep learning algorithms for real-time road damage detection," in Proc. IEEE Int. Conf. Crowd Sensing-based Road Damage Detection, Ho Chi Minh City, Vietnam, Oct. 2024, pp. 1–6.

[27] C. T. Chen and Y. C. Lin, "HPPH: Computer-vision-based service for high-performance pavement health recognition," IEEE Internet Things J., vol. 12, no. 2, pp. 15987–15998, 2025.

[28] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.

[29] A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv:2004.10934, 2020.

[30] G. Jocher et al., "YOLOv5 by Ultralytics," GitHub Repository, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[31] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[32] K. He et al., "Deep residual learning for image recognition," in Proc. IEEE/CVF CVPR, Las Vegas, NV, 2016, pp. 770–778.

[33] T. Y. Lin et al., "Feature pyramid networks for object detection," in Proc. IEEE/CVF CVPR, Honolulu, HI, 2017, pp. 2117–2125.

[34] Z. Liu et al., "Path aggregation network for instance segmentation," in Proc. IEEE/CVF CVPR, Salt Lake City, UT, 2018, pp. 8759–8768.

[35] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in Proc. ICLR, 2021.

[36] R. Girshick, "Fast R-CNN," in Proc. IEEE/CVF ICCV, Santiago, Chile, 2015, pp. 1440–1448.

[37] W. Liu et al., "SSD: Single shot multibox detector," in Proc. ECCV, Amsterdam, Netherlands, 2016, pp. 21–37.

[38] M. Sandler et al., "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF CVPR, Salt Lake City, UT, 2018, pp. 4510–4520.

[39] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE/CVF CVPR, Kauai, HI, 2001, pp. 511–518.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. ICLR, San Diego, CA, 2015.

[41] C. Szegedy et al., "Going deeper with convolutions," in Proc. IEEE/CVF CVPR, Boston, MA, 2015, pp. 1–9.

[42] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE/CVF CVPR, San Diego, CA, 2005, pp. 886–893.