Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 47 No. 02 (2026)

Smart Edge-AI Solution for Automated Real-Time Surface Defect Identification

Dr. V. Sarala Devi[1]

[1]Department of Computer Applications,

Dr. M.G.R Educational and Research Institute, Chennai, India

Abstract—This paper presents a Smart Edge-AI framework for automated real-time surface defect identification using deep learning and computer vision. The system employs a lightweight YOLOv8n object detection model deployed on a distributed architecture consisting of a Raspberry Pi 4 for video acquisition and wireless streaming, and a host computer for deep learning inference. The framework incorporates dataset refinement techniques, including data augmentation, negative sample integration, and annotation verification, to enhance model robustness. Model optimization through ONNX conversion, input resolution tuning, and confidence threshold calibration further improves inference speed. The system achieves a detection precision of approximately 0.96, an F1 score of 0.93, and operates at 15–20 frames per second in real-time. The proposed solution is portable, cost-effective, cloud-independent, and suitable for deployment in infrastructure monitoring, industrial inspection, and public safety applications. Experimental results validate that the combination of edge computing and deep learning provides a scalable and efficient alternative to conventional inspection methods.

Keywords—Edge AI, YOLOv8n, Surface Defect Detection, Object Detection, Distributed Architecture, Real-Time Inspection, Deep Learning, Raspberry Pi, Computer Vision, ONNX

I. INTRODUCTION

Surface inspection is a critical process in maintaining the structural integrity of civil infrastructure such as buildings, bridges, tunnels, and roads. The conventional approach relies on manual examination by trained personnel, a method that is inherently time-consuming, labor-intensive, subjective, and susceptible to human error caused by fatigue and perceptual inconsistency. Early-stage anomalies, particularly fine cracks, are frequently overlooked during manual inspection, leading to delayed maintenance, progressive structural degradation, and heightened safety risks.

The rapid advancement of Artificial Intelligence (AI) and Computer Vision has catalyzed the development of automated inspection systems capable of surpassing human performance in defect detection tasks. Deep learning models, particularly Convolutional Neural Networks (CNNs) and real-time object detection frameworks, have demonstrated superior accuracy in identifying surface anomalies from image and video data [1].

Nevertheless, most existing deep learning-based solutions are tightly coupled to cloud infrastructure, which introduces significant drawbacks: high transmission latency, dependency on reliable internet connectivity, potential data privacy vulnerabilities, and unsuitability for deployment in remote or resource-constrained environments [2]. These limitations motivate the development of an edge-native, real-time inspection system that operates independently of cloud services.

This paper presents a Smart Edge-AI framework for automated surface defect detection. The proposed system integrates the YOLOv8n model—a state-of-the-art, lightweight object detection architecture—with a distributed edge computing design. A Raspberry Pi 4 handles video capture and MJPEG-based wireless streaming, while a host computer performs deep learning inference. The architecture decouples data acquisition from computation, optimizing performance within the constraints of edge hardware.

The system incorporates multiple optimization strategies, including data augmentation, hyperparameter tuning, negative sample injection, ONNX model conversion, and confidence threshold calibration, achieving real-time performance at 15–20 FPS with detection precision of approximately 0.96.

The remainder of this paper is organized as follows: Section II reviews related work; Section III describes the proposed system architecture; Section IV details the dataset and training methodology; Section V presents experimental results; Section VI discusses applications; Section VII addresses challenges and limitations; Section VIII outlines future work; and Section IX concludes the paper.

II. Related Work

Several approaches have been explored for automated surface anomaly detection, spanning manual methods, classical image processing, cloud-based AI, and deep learning at the edge.

A. Classical Image Processing

Traditional computer vision techniques, including Canny edge detection, Sobel operators, Hough transforms, and morphological operations, were early approaches to automating surface inspection [3]. While computationally inexpensive, these methods are highly sensitive to illumination variations and background complexity. Their inability to generalize across diverse surface textures and crack morphologies limits practical deployment.

B. CNN-Based Defect Detection

The introduction of deep convolutional neural networks significantly elevated detection accuracy. Models such as VGG-16 and ResNet demonstrated the capacity to extract hierarchical visual features relevant to crack detection [4]. However, these architectures incur substantial computational costs, making real-time inference on resource-constrained hardware infeasible.

C. YOLO Family for Real-Time Detection

The YOLO (You Only Look Once) family of detectors introduced a paradigm shift by framing object detection as a single-pass regression problem, achieving unprecedented inference speeds [5]. YOLOv4 and YOLOv5 demonstrated strong performance across industrial inspection benchmarks [6]. The latest YOLOv8n variant further reduces model complexity while retaining competitive accuracy, making it well-suited for edge deployment.

D. Edge Computing for Inspection

Recent work has explored deploying lightweight models on edge hardware such as NVIDIA Jetson Nano and Raspberry Pi for field inspection tasks [7]. However, few studies have examined distributed architectures that separate video acquisition from inference, a design choice that significantly reduces the processing burden on constrained edge devices while maintaining real-time throughput. The proposed system addresses the gaps identified in the literature by combining a state-of-the-art lightweight detector with a distributed edge architecture optimized for practical deployment.

III. Proposed System Architecture

A. System Overview

The proposed framework adopts a distributed two-node architecture. Node 1, the Raspberry Pi 4 Model B (4 GB RAM, Broadcom BCM2711 SoC), is responsible for video acquisition using a 5-megapixel camera module and real-time MJPEG streaming over a local Wi-Fi network. Node 2, a host computer equipped with an Intel Core i5 processor and 8 GB RAM, performs YOLOv8n inference, bounding box rendering, and result display.

This decoupled design offloads computationally intensive inference to the more capable host system while leveraging the Raspberry Pi's wireless interface for flexible field deployment. Communication between nodes uses Flask-based MJPEG streaming, ensuring low overhead and compatibility across platforms.

B. Hardware Configuration

The edge node comprises a Raspberry Pi 4 Model B, a Raspberry Pi Camera Module V2, and a USB Wi-Fi adapter for network connectivity. Power is supplied via a 5V/3A USB-C adapter, enabling operation from a portable power bank for field deployments. The inference node connects to the same local network via Ethernet or Wi-Fi and runs the detection pipeline using standard Python libraries.

C. Software Architecture

The software stack is built entirely on open-source tools. The Raspberry Pi runs Raspberry Pi OS Lite with Python 3.9, OpenCV 4.5 for frame capture, and Flask for HTTP streaming. The host node runs Python 3.10 with PyTorch 2.0, Ultralytics YOLOv8, and OpenCV for inference and visualization. Model export to ONNX format enables runtime-agnostic inference using ONNXRuntime, which improves throughput compared to native PyTorch inference.

D. Data Flow

The pipeline operates as follows: (1) the camera module captures raw frames at 1280×720 resolution; (2) OpenCV encodes frames as JPEG and Flask serves them via an HTTP MJPEG stream; (3) the host node fetches frames from the stream using OpenCV VideoCapture; (4) frames are resized to 640×640 and passed to the YOLOv8n model; (5) detected bounding boxes with confidence scores above the threshold are rendered; (6) the annotated frame is displayed in real time with FPS overlay.

IV. Dataset And Training Methodology

A. Dataset Composition

The training dataset was compiled from publicly available crack detection datasets, including the SDNET2018 and Concrete Crack Images datasets, supplemented with custom-annotated images captured in controlled indoor environments. The combined dataset contains approximately 12,000 images, split into 80% training, 10% validation, and 10% test sets. Annotations were generated using Roboflow, and all labels were verified through manual inspection to eliminate annotation errors.

B. Data Augmentation

To improve model generalization and robustness to real-world variability, the following augmentation pipeline was applied: (i) random horizontal and vertical flips; (ii) Gaussian blur with kernel sizes between 3 and 7; (iii) brightness and contrast jitter (±20%); (iv) random rotation (±15°); (v) mosaic augmentation combining four training images; and (vi) cutout regularization. These augmentations increase effective dataset diversity and reduce overfitting.

C. Negative Sample Integration

A critical refinement step involved incorporating negative samples—images containing no surface defects—into the training set at a 20% ratio. This technique explicitly trains the model to recognize non-defective surfaces, substantially reducing false positive detections on textured backgrounds such as concrete and asphalt.

D. Training Configuration

The YOLOv8n model was trained using the Ultralytics framework on Google Colab with an NVIDIA T4 GPU (16 GB VRAM). Training configuration is summarized in Table I.

Parameter	Value
Epochs	150
Batch Size	16
Optimizer	SGD (momentum = 0.937)
Learning Rate	0.01 (cosine decay)
Input Resolution	640×640
Augmentation	Mosaic, Flip, Blur, Jitter
Pretrained Weights	YOLOv8n COCO
Export Format	ONNX (opset 17)

TABLE I. YOLOv8n Training Configuration Parameters

V. Experimental Results

A. Detection Accuracy

The model was evaluated on the held-out test set using standard object detection metrics: precision (P), recall (R), mean Average Precision at IoU threshold 0.5 (mAP@50), and F1 score. Results are presented in Table II.

Metric	Value
Precision (P)	0.961
Recall (R)	0.912
mAP@50	0.947
mAP@50:95	0.623
F1 Score	0.936
Inference Speed (FPS)	15–20
Model Size	6.2 MB
ONNX Inference Latency	48 ms/frame

TABLE II. Performance Metrics of YOLOv8n Model on Test Set

B. Training Convergence

Training and validation loss curves exhibited consistent monotonic decrease over 150 epochs, with no divergence or oscillation observed after epoch 60. The convergence of validation metrics alongside training

metrics confirms that the model generalizes effectively and does not exhibit overfitting, attributable to the augmentation pipeline and negative sample integration.

C. Precision–Recall Analysis

The precision–recall curve achieved a high area under the curve (AUC), remaining close to the upper boundary across the full confidence threshold range. Precision stabilized near 0.96 at confidence thresholds above 0.60, while recall decreased gradually with increasing threshold, reflecting the expected precision–recall trade-off inherent to detector calibration.

D. F1 Score and Optimal Threshold

The peak F1 score of 0.936 was achieved at a confidence threshold of 0.55–0.60, representing the optimal operating point that balances detection completeness with prediction reliability. This threshold was adopted for all real-time inference experiments.

E. Real-Time Performance

The system achieved a steady-state frame rate of 15–20 FPS on the host node using the ONNX-optimized model. ONNX conversion over PyTorch inference reduced per-frame latency from approximately 80 ms to 48 ms, a 40% improvement. The Raspberry Pi streaming overhead measured at the host was consistently below 12 ms per frame over a 2.4 GHz Wi-Fi link, confirming that the network is not a bottleneck at the target frame rate.

F. Comparison with Baseline Methods

Table III compares the proposed system against representative methods from the literature. The YOLOv8n-based edge system achieves competitive or superior accuracy to cloud-dependent approaches while delivering real-time performance without internet connectivity.

Method	Precision	FPS	Cloud?	Edge HW
Manual Inspection	~0.70	—	No	None
Canny + Hough [3]	0.72	30+	No	CPU
ResNet-50 Cloud [4]	0.94	5–8	Yes	None
YOLOv5s Edge [7]	0.91	10–12	No	RPi 4
Proposed YOLOv8n	0.961	15–20	No	RPi 4

TABLE III. Comparison of Surface Defect Detection Methods

VI. Applications

A. Infrastructure Monitoring

The system is directly applicable to the automated monitoring of civil infrastructure including bridges, tunnels, retaining walls, and building facades. Continuous real-time monitoring enables proactive maintenance scheduling by detecting cracks at early stages before propagation poses structural risk. Deployment with a pan-tilt camera mount enables systematic scanning of large surface areas.

B. Industrial Quality Control

In manufacturing environments, the system can be integrated into production line conveyor systems to perform real-time surface inspection of machined components, concrete blocks, ceramic tiles, and prefabricated

structural elements. The high precision of 0.961 minimizes false rejections, reducing material waste and improving throughput compared to manual quality control.

C. Road Surface Inspection

Mounted on a vehicle or drone, the system can perform automated pavement inspection to detect cracks, potholes, and surface deterioration. Early detection of pavement defects enables targeted maintenance interventions, extending pavement service life and improving road safety. The cloud-independent operation is particularly advantageous for remote highway inspection where cellular connectivity is unreliable.

D. Smart Cities and Public Safety

Integration with smart city sensor networks enables continuous structural health monitoring of public infrastructure. Automated alerts can be generated when defect detections exceed configurable frequency thresholds, enabling rapid response by maintenance teams. The system architecture supports multi-camera deployments with a shared inference node, enabling cost-effective large-scale monitoring.

E. Remote and Field Inspection

The portability of the Raspberry Pi 4 node, operable from a power bank, enables deployment in remote environments inaccessible to cloud-connected systems. Field inspectors can operate the wireless streaming client from a laptop over a mobile hotspot, obtaining real-time defect annotations without specialized equipment.

VII. Challenges And Limitations

A. Dataset Diversity

The primary limitation of the current system is its dependence on dataset quality and diversity. The model was trained predominantly on concrete crack images; performance on dissimilar surface types such as metal corrosion or asphalt fatigue cracking may degrade without domain-specific fine-tuning. Acquiring large-scale, annotated datasets for diverse defect categories remains a significant practical challenge.

B. Environmental Sensitivity

Detection accuracy is sensitive to extreme lighting conditions, including direct sunlight causing specular reflection and low-light environments where texture detail is lost. Shadow boundaries on textured concrete surfaces occasionally generate false positive detections. Incorporating adaptive preprocessing, such as histogram equalization and shadow removal, is expected to mitigate these issues.

C. Edge Hardware Constraints

While the distributed architecture substantially reduces the processing burden on the Raspberry Pi, the edge node remains a potential bottleneck for frame capture throughput at resolutions above 1280×720. Deployment of a fully edge-optimized inference pipeline using TensorRT or OpenVINO on a more capable edge device such as NVIDIA Jetson Nano would enable single-node operation at higher frame rates.

D. Single-Class Detection

The current model is trained to detect a single defect class (cracks). Multi-class extension to simultaneously detect corrosion, delamination, and spalling requires substantially larger and more diverse annotated datasets and may necessitate a deeper model variant (e.g., YOLOv8s or YOLOv8m) to preserve accuracy across classes.

VIII. Future Work

Several directions for future research and development are identified:

1) Multi-Defect Detection: Extending the model to jointly detect cracks, corrosion, dents, and spalling by expanding the annotated dataset and adopting a multi-class detection framework.

2) Fully Edge-Optimized Pipeline: Implementing TensorRT INT8 quantization for deployment on NVIDIA Jetson Nano, enabling single-node real-time inference at 30 FPS without a host computer.

3) Drone Integration: Mounting the camera node on an autonomous UAV platform to enable systematic aerial inspection of building facades and large infrastructure with GPS-tagged defect localization.

4) IoT and Alert Integration: Connecting the detection output to an IoT platform (e.g., MQTT/Node-RED) to generate automated maintenance alerts, trend analyses, and inspection reports.

5) 3D Defect Characterization: Incorporating stereo vision or LiDAR depth sensing to estimate crack depth and volume, enabling structural severity assessment beyond 2D detection.

6) Transfer Learning for New Domains: Developing a continual learning pipeline that allows rapid fine-tuning of the base model for new surface materials with minimal labeled data using few-shot learning techniques.

IX. Conclusion

This paper presented a Smart Edge-AI framework for automated real-time surface defect identification, addressing the limitations of conventional manual inspection and cloud-dependent AI systems. The proposed system employs the YOLOv8n lightweight object detection model within a distributed architecture consisting of a Raspberry Pi 4 acquisition node and a host inference node, communicating via MJPEG streaming over a local network.

Comprehensive dataset refinement, including data augmentation, negative sample integration, and annotation verification, combined with ONNX model optimization, enabled the system to achieve a detection precision of 0.961, an F1 score of 0.936, and sustained real-time inference at 15–20 FPS. Comparative evaluation confirmed that the proposed approach matches or exceeds the accuracy of cloud-based systems while operating entirely at the edge.

The system is portable, cost-effective, and deployable in environments with limited or no internet connectivity, making it suitable for infrastructure monitoring, industrial quality control, road surface inspection, and public safety applications. This work demonstrates that the integration of edge computing with state-of-the-art deep learning yields practical, scalable solutions for automated inspection that can meaningfully reduce maintenance costs and improve structural safety outcomes.

. Acknowledgment

The authors thank the Department of Computer Science and Engineering, XYZ Institute of Technology, for providing laboratory facilities and computational resources. The publicly available SDNET2018 and Concrete Crack Images datasets used in this work are gratefully acknowledged.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[2] M. Satyanarayanan, "The emergence of edge computing,"Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[3] J. Canny, "A computational approach to edge detection,"IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[4] L. Zhang et al., "Road crack detection using deep convolutional neural network," in Proc. IEEE ICIP, Phoenix, AZ, USA, Sep. 2016, pp. 3708–3712.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE CVPR, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

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

[7] Ultralytics, "YOLOv8: Next-generation object detection," Tech. Rep., 2023. [Online]. Available: https://docs.ultralytics.com

[8] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. New York, NY, USA: Springer, 2022.

[9] N. Koenig and A. Howard, "Design and use paradigms for Gazebo," in Proc. IEEE/RSJ IROS, Sendai, Japan, Sep. 2004, pp. 2149–2154.

[10] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image," in Proc. NeurIPS, Montreal, QC, Canada, Dec. 2014, pp. 2366–2374.