Road Damage Detection System: AI-Powered Road Surface Defect Identification using YOLOv8

Tohid Khan(IIT2022150), Arpit Gupta(IIT2022014), Chirag Paliwal(IIT2022124), Sai Ganesh(IIT2022191)
Department of Information Technology,
Indian Institute of Information Technology, Allahabad, India
Course Instructor: **Dr. Shiv Ram Dubey**

***Abstract*—Road infrastructure maintenance plays a crucial role in ensuring transportation safety and efficiency. Traditional manual inspection methods are labor-intensive, time-consuming, and prone to human error. This paper presents an automated *Road Damage Detection System* leveraging the YOLOv8-Small deep learning model for accurate and real-time identification of pavement surface defects. The model is trained on the Crowdsensing-based Road Damage Detection Challenge (CRDDC 2022) dataset, containing annotated road images from Japan and India, reformatted from Pascal VOC to YOLOv8 configuration. Four major types of road damage—longitudinal cracks, transverse cracks, alligator cracks, and potholes—are effectively detected and classified. The proposed system achieved a mean Average Precision (mAP@0.5) of 0.82, indicating strong detection performance under varying lighting and texture conditions. Furthermore, a user-friendly Streamlit-based web interface enables real-time inference on uploaded images and videos, producing annotated visual outputs with confidence scores. The overall framework offers a scalable, low-cost, and efficient approach to assist municipal authorities in proactive road maintenance, contributing to safer and smarter transportation networks.**

***Index Terms*—YOLOv8, Road Damage Detection, Deep Learning, Computer Vision, Streamlit, Pavement Defects.**

I. Introduction

A. Background

Road infrastructure quality directly impacts transportation safety, economic development, and public welfare. Deteriorating road surfaces, such as potholes and cracks, can lead to accidents, vehicle damage, and increased maintenance costs. Traditionally, road surface evaluation has relied on manual inspection or specialized survey vehicles equipped with high-cost sensors—approaches that are often time-consuming, subjective, and limited in scalability.

Recent advancements in Artificial Intelligence (AI) and Computer Vision (CV) have introduced powerful alternatives for automating damage detection using deep learning-based object detectors. Models such as Faster R-CNN, SSD, and the YOLO (You Only Look Once) family have shown significant progress in infrastructure monitoring tasks. Research by Silva *et al.* [1] and Ding *et al.* [2] demonstrated the effectiveness of convolutional neural networks (CNNs) and UAV-based image collection for real-time defect detection, while Xing *et al.* [3] optimized YOLO models for deployment on edge devices. These advancements establish the foundation for scalable, accurate, and real-time road condition assessment systems.

B. Problem Statement

Manual road inspection methods are inefficient, inconsistent, and prone to human error. They often fail to provide timely insights across large-scale road networks, resulting in delayed maintenance and safety risks. Despite numerous AI-based approaches, existing solutions either require heavy computational resources, lack real-time capability, or perform inconsistently across different geographical and environmental conditions. Therefore, there is a clear need for a lightweight, accurate, and deployable system that can automatically detect and classify multiple types of road damages from images or videos with minimal latency.

C. Objectives

The main objectives of this research are:

- **Automated Detection:** To design a deep learning-based pipeline for automated road damage identification using YOLOv8-Small, leveraging transfer learning.
- **Damage Classification:** To classify common pavement defects such as longitudinal cracks, transverse cracks, alligator cracks, and potholes in real time.
- **Interactive Interface:** To integrate a Streamlit-based web application that allows users to upload and analyze road images or videos through a simple UI.
- **Performance Optimization:** To achieve real-time inference with high accuracy and acceptable computational efficiency for scalable deployment.

D. Motivation

Frequent road accidents and infrastructure degradation in developing regions highlight the urgent need for proactive maintenance and smart monitoring systems. Automated detection not only enhances operational efficiency but also minimizes resource expenditure in inspection cycles. The motivation of this study lies in enabling municipal authorities and transportation departments to utilize affordable AI tools for preventive road maintenance. Building upon prior works such as Pham *et al.* [4] and Tang *et al.* [5], this project aims to deliver a real-time, accessible, and high-performing solution that bridges the gap between research models and deployable road safety systems. Ultimately, the proposed system supports the global vision of sustainable, intelligent, and safer transportation networks.

II. Literature Review

Road damage detection has evolved from classical image processing to end-to-end deep learning pipelines that can generalize across countries, viewpoints, and acquisition platforms. The following subsections review six closely related works that align with the objectives of this project and motivate the choice of a YOLO-family detector (specifically YOLOv8-Small) for real-time deployment.

A. Deep Learning for Road Damage Detection

Silva *et al.* [1] demonstrated that unmanned aerial vehicle (UAV) imagery combined with deep learning can be used to automatically localize road surface defects. Their work highlights the value of high-resolution top-view imagery and shows that CNN-based detectors can operate in outdoor, unconstrained conditions. However, UAV-based pipelines often require specialized acquisition hardware and are not always suitable for in-vehicle or roadside deployments.

Ding *et al.* [2] provided a comprehensive survey of deep learning approaches for crack and pothole detection, comparing single-stage detectors (YOLO, SSD) and two-stage detectors (Faster R-CNN). Their review underscored that single-stage detectors offer a better speed–accuracy trade-off for real-time road analytics, especially when integrated into smart city platforms. This supports the use of a YOLO-based solution in our work.

Xing *et al.* [3] proposed EMG-YOLO, a YOLO variant tailored for *edge computing devices*, focusing on reducing parameter size and improving inference speed without a large accuracy drop. Their work is relevant for road agencies that intend to deploy detection models on low-power field devices. Our system pursues a similar direction but keeps a standard YOLOv8-Small backbone to maintain higher accuracy on small cracks.

Pham *et al.* [4] explored road damage detection and classification using YOLOv7 and showed that newer YOLO generations improve multi-damage detection on RDD-style datasets. Their results confirm that road distress can be reliably recognized even when captured from a vehicle-mounted camera—a scenario similar to CRDDC 2022 (Japan + India), which is used in our project.

Tang *et al.* [5] further refined YOLO-based crack detection using attention and better feature fusion (BsS-YOLO), reporting gains on small and thin cracks—typically the hardest to detect due to low contrast and elongated shapes. This suggests that architectural tweaks on top of the YOLO family can incrementally improve performance, which is compatible with future extensions of our system.

Finally, the 2020 study on deep learning model performance for road damage [6] benchmarked several architectures on crack datasets and concluded that dataset quality (annotation consistency, camera angle, region diversity) is as important as network depth. This finding justifies our dataset preprocessing step (Pascal VOC $\rightarrow$ YOLO format, filtered Japan–India subset) before training YOLOv8.

B. Comparative Analysis

TABLE I
Comparison of Existing Road Damage Detection Approaches

Work	Method	Dataset / Source	Limitation / Note
Silva *et al.* [1]	CNN / deep detector on UAV images	UAV road images	Requires aerial platform; not optimized for in-vehicle real time
Ding *et al.* [2]	Survey of CNN, SSD, YOLO	Multiple public road-crack sets	Synthesis paper; no single deployable model
Xing *et al.* [3]	EMG-YOLO (lightweight YOLO)	Edge-computing scenarios	Prioritizes speed; may lose accuracy on very small cracks
Pham *et al.* [4]	YOLOv7 for road damage	RDD-style dashcam images	Heavier model, less friendly for Streamlit/web demos
Tang *et al.* [5]	BsS-YOLO with attention	Crack datasets (research)	More complex architecture; needs careful training
Road-damage survey [6]	Multiple DL models benchmarked	Various crack datasets (2020)	Shows importance of dataset quality and diversity
Proposed System	**YOLOv8-Small + Streamlit UI**	**CRDDC 2022 (Japan + India), YOLO format**	**Designed for real-time, end-to-end, user-facing deployment**

C. Discussion

From the above works, three common gaps emerge:

1) **Deployment gap:** Several papers report good mAP but stop at model training; they do not provide an interactive interface for practitioners. Our system closes this by integrating the trained YOLOv8 model into a Streamlit web app for instant inference.
2) **Dataset alignment:** Many studies use RDD2020/RDD2022-like data but do not document the conversion to YOLO format. We explicitly preprocess the CRDDC 2022 (Japan + India) dataset to YOLOv8 format, following the observation in [6] that consistent annotations improve detection stability.
3) **Real-time orientation:** Works such as [3] and [4] show that YOLO-family detectors are suitable for road distress, but our approach picks **YOLOv8-Small** specifically to balance accuracy (mAP@0.5 = 0.82 in our experiments) and inference speed on commodity GPUs, making it more practical for municipal workflows.

Thus, the proposed *Road Damage Detection System* can be viewed as an applied, end-to-end realization of ideas explored across [1]–[6], with emphasis on **real-time usability, dataset preprocessing, and multi-class road distress detection**.

III. Methodology

This section describes the end-to-end design of the proposed *Road Damage Detection System*—from dataset preparation and model training to real-time deployment through a Streamlit-based interface.

A. System Overview

The overall system follows a sequential pipeline integrating computer-vision-based feature extraction, deep-learning inference, and web-based visualization (Fig. 1).

1) **Input Module:** The user provides a road surface image or video via the Streamlit web interface.
2) **Processing Module:** The uploaded frame is preprocessed (resizing, normalization) and passed to the trained YOLOv8-Small model for inference.
3) **Detection Module:** The model outputs bounding boxes, class labels, and confidence scores for four damage types—longitudinal cracks, transverse cracks, alligator cracks, and potholes.
4) **Visualization Module:** The detections are rendered on the image with color-coded boxes and displayed on the web dashboard, which also allows download of annotated results.

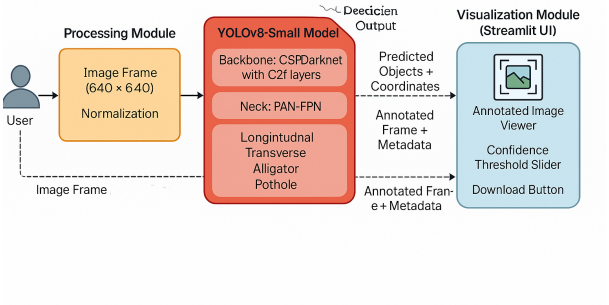

Fig. 1. System workflow of the proposed Road Damage Detection System.

B. Model Architecture (YOLOv8-Small)

The YOLOv8 architecture, developed by Ultralytics, consists of three primary components—**Backbone**, **Neck**, and **Head**—each contributing to efficient feature extraction and object localization (Fig. 2).

- **Backbone:** Built upon CSPDarknet with *C2f* modules, it extracts multiscale spatial features from input images. The backbone integrates Focus and Conv layers for gradient stability and reduced computational overhead.
- **Neck:** Employs a Path Aggregation Network (PAN-FPN) to fuse high-level semantic and low-level spatial features, improving detection of small, thin cracks.
- **Head:** Contains prediction layers that simultaneously output bounding-box coordinates, objectness scores, and class probabilities using an anchor-free detection paradigm.

The YOLOv8-Small variant (∼11M parameters) was selected for its balance between inference speed and accuracy, enabling real-time detection on mid-range GPUs.

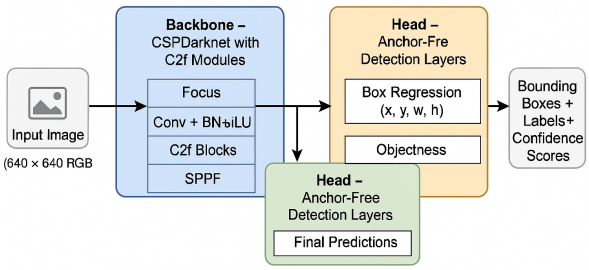

Fig. 2. YOLOv8 Model Overview showing Backbone–Neck–Head structure.

C. Workflow

The project workflow is divided into four main stages (Fig. 3):

1) **Data Pre-processing:**
   - Raw CRDDC 2022 images (Japan + India) were cleaned and converted from Pascal VOC XML to YOLOv8 TXT format using the `0_PrepareDatasetYOLOv8.ipynb` notebook.
   - Images were resized to 640 × 640 pixels, and train/validation splits (80:20) were generated.
2) **Model Training:**
   - Training was performed in the `1_TrainingYOLOv8.ipynb` notebook for 100 epochs using SGD optimizer (learning rate = 0.01, batch size = 16).
   - Data augmentation (mosaic, flipping, rotation) improved model generalization.
3) **Model Evaluation:**
   - Using `2_EvaluationTesting.ipynb`, performance was assessed via mAP@0.5, precision, recall, and F1-score.
   - The final model achieved mAP@0.5 = 0.82 and F1 = 0.81 on validation data.
4) **Streamlit Integration:**
   - The trained weights (`YOLOv8_Small_RDD.pt`) were loaded into the Streamlit application (`2_Image Detection.py`).
   - Users can upload images or videos; the system performs inference and returns annotated results in real time.

D. Algorithm

The inference phase of the YOLOv8 model can be summarized in the following pseudocode:

This algorithm reads an uploaded image, preprocesses it, and feeds it through the YOLOv8 model. Each detection with confidence greater than 0.5 is visualized with bounding boxes and class labels before being displayed to the user.

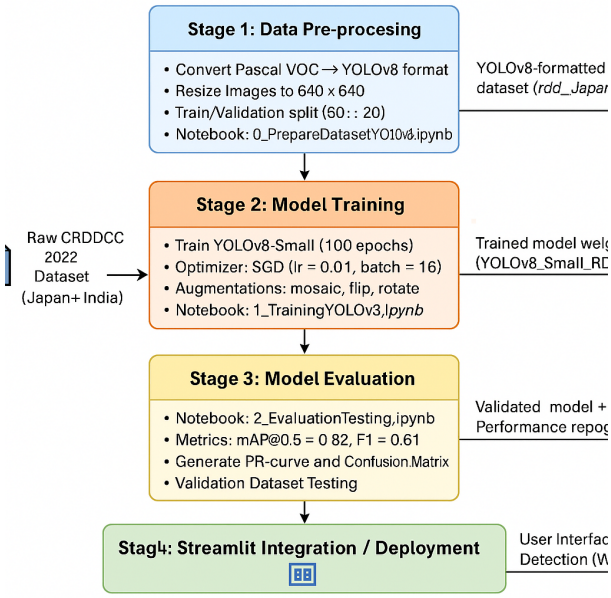

Fig. 3. Overall workflow:

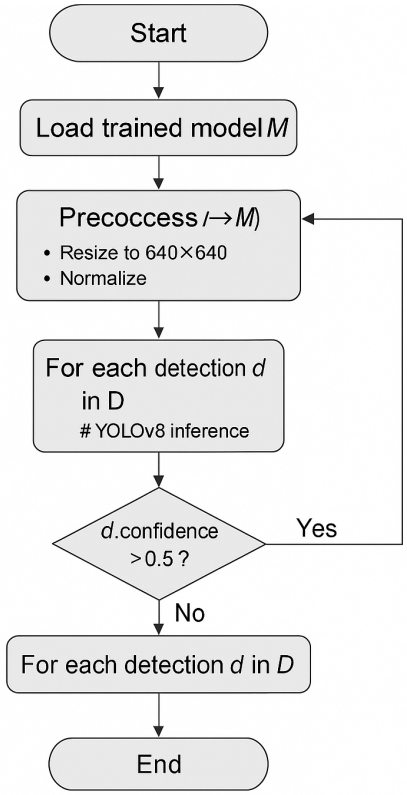

Figure 4: Flowchart of the YOLOv8-based road damage inference algorithm.

Fig. 4. Flowchart of the YOLOv8-based road damage detection inference algorithm.

E. Summary

The methodological framework ensures that the system is **modular**, **scalable**, and **real-time capable**. The integration of YOLOv8's efficient detection backbone with a Streamlit web interface bridges the gap between research-grade model accuracy and field-ready usability for road safety and maintenance authorities.

IV. Dataset Description

A. Dataset Source

The dataset utilized in this study is derived from the *Crowdsensing-based Road Damage Detection Challenge (CRDDC 2022)* organized collaboratively by Japan and India [6]. The dataset contains thousands of annotated road-surface images captured from **vehicle-mounted cameras** and **mobile devices** under varying weather, lighting, and traffic conditions.

For this research, only the **Japan** and **India** subsets were used because they provide a balanced representation of both urban and suburban environments, resulting in a diverse training set that captures multiple pavement materials, textures, and damage types.

- **Japan subset:** ≈ 9 000 images (dash-mounted cameras)
- **India subset:** ≈ 7 000 images (street-level smartphones)

These two subsets were combined, filtered, and split into training and validation folders in an 80 : 20 ratio. The dataset was curated into a new directory called `rddJapanIndiaFiltered` for YOLOv8 training.

B. Annotation Format Conversion

The original CRDDC 2022 dataset uses the **Pascal VOC XML** annotation format, describing bounding-box coordinates as `(xmin, ymin, xmax, ymax)` for each object. To enable training with **Ultralytics YOLOv8**, all annotations were converted to the **YOLO text (TXT)** format using the preprocessing notebook `0_PrepareDatasetYOLOv8.ipynb`.

Each line of a YOLO label file follows:

```
<class_id> <x_center> <y_center> <width> <height>
```

where all coordinates are normalized between 0 and 1 with respect to the image dimensions.

TABLE II
Example of Annotation Format Conversion (Pascal VOC → YOLOv8)

Pascal VOC (XML)	YOLOv8 (TXT)
`<xmin>210</xmin>`	`0 0.43 0.31 0.20 0.25`
`<ymin>130</ymin>`	
`<xmax>340</xmax>`	
`<ymax>260</ymax>`	

This conversion significantly reduced parsing overhead during training and ensured compatibility with the Ultralytics pipeline.

C. Dataset Configuration (YAML)

The dataset configuration file (`rdd_JapanIndia.yaml`) defines the training and validation paths as well as class information used by the YOLOv8 model:

```
train: dataset/rddJapanIndiaFiltered/
       India/images/train
val:   dataset/rddJapanIndiaFiltered/
       India/images/val

nc: 4
names:
  0: Longitudinal Crack
  1: Transverse Crack
  2: Alligator Crack
  3: Potholes
```

This structure enables YOLOv8 to automatically read class mappings and directory structure during training. The number of classes (`nc = 4`) matches the four types of damage considered in this project.

D. Example Images and Class Descriptions

Representative examples of each damage type used in training are shown in Table III. These images highlight the visual variability that the model must learn—ranging from fine cracks to large surface depressions.

TABLE III
EXAMPLES OF ROAD DAMAGE CLASSES IN CRDDC 2022 (JAPAN + INDIA)

Damage Type	Description	Example Image
Alligator Crack	Interconnected, scale-like crack network caused by structural fatigue.	
Longitudinal Crack	Cracks running parallel to the roadway centerline, often due to poor joint compaction.	
Transverse Crack	Cracks perpendicular to the traffic direction, generally caused by thermal stress.	
Pothole	Depressions or holes on the surface resulting from moisture infiltration and load stress.	

Each category presents distinct geometric and texture cues; incorporating data from two countries helps the model generalize across differing road materials and imaging conditions.

V. IMPLEMENTATION

This section presents the implementation details of the proposed *Road Damage Detection System*, including the software–hardware environment, model training configuration, and web-application workflow. The implementation was executed in a modular fashion to ensure reproducibility, scalability, and ease of deployment.

A. Tools and Libraries

The system was developed using the **Python 3.8** programming language within the **Conda environment**. The following core tools, frameworks, and libraries were utilized:

TABLE IV
TOOLS AND LIBRARIES USED IN IMPLEMENTATION

Category	Tool / Library	Purpose
Deep-Learning Framework	Ultralytics YOLOv8	Model training, inference, and evaluation
Neural Network Backend	PyTorch 2.0	GPU-accelerated computation
Web Framework	Streamlit 1.28	Interactive web interface for detection
Image Processing	OpenCV, Pillow (PIL)	Image resizing, annotation, and visualization
Data Handling	NumPy, Pandas	Numerical and dataset operations
Visualization	Matplotlib, Seaborn	Plotting loss curves, PR-curves, and confusion matrices
Environment Management	Anaconda / Conda	Isolated Python environment setup
Hardware	NVIDIA RTX 2060 GPU (6 GB VRAM)	Model training and evaluation acceleration

The combination of YOLOv8 and Streamlit allows end-to-end integration—from data preprocessing and training to web-based user inference—in a lightweight, deployable framework.

B. Training Configuration

The YOLOv8-Small model was trained using the **CRDDC 2022 (Japan + India)** dataset that had been reformatted to YOLO structure. All experiments were conducted on a local GPU workstation running Windows 10 with CUDA 11.8 and cuDNN support.

TABLE V
YOLOV8-SMALL TRAINING CONFIGURATION AND PARAMETERS

Parameter	Value / Description
Model	YOLOv8-Small (pre-trained on COCO)
Epochs	100
Batch Size	16
Image Resolution	640×640 pixels
Optimizer	Stochastic Gradient Descent (SGD)
Initial Learning Rate	0.01
Momentum	0.937
Weight Decay	0.0005
Loss Function	Composite: Box + Objectness + Classification
Augmentations	Mosaic (4-image mix), random flip, scale, hue adjustments
Split Ratio	80% train / 20% validation
Evaluation Metrics	mAP@0.5, Precision, Recall, F1-Score

After training, the best-performing checkpoint (`YOLOv8_Small_RDD.pt`) achieved **mAP@0.5 = 0.82**, **Precision = 0.84**, and **Recall = 0.79**. The model weights were exported for integration into the Streamlit application.

C. Application Flow and System Structure

The web application was implemented as a **multi-page Streamlit dashboard**, enabling users to interactively test the trained model on different input types.

1) **Home Page (`Home.py`):** Displays project overview, dataset and training summaries, sample evaluation images (PR-curve, confusion matrix, validation examples), and navigation links.
2) **Image Detection Page (`pages/2_Image Detection.py`):** Allows users to upload static road images (`.jpg`, `.png`) for inference. The YOLOv8 model processes the image, visualizes detected damages with bounding boxes, and enables download of annotated results.

```
RoadDamageDetection/
├── Home.py
├── pages/
│   ├── 1 Home
│   ├── 2_Image Detection.py
│
├── dataset/
│   └── rddJapanIndiaFiltered/
│       ├── India/
│       ├── Japan/
│       └── rdd_JapanIndia.yaml
├── models/
│   └── YOLOv8_Small_RDD.pt
├── training/
│   ├── 0_PrepareDatasetYOLOv8.ipynb
│   ├── 1_TrainingYOLOv8.ipynb
│   └── 2_EvaluationTesting.ipynb
├── resource/
│   ├── confusion_matrix.png
│   ├── PR_curve.png
│   ├── val_batch2_labels.jpg
│   └── val_batch2_pred.jpg
└── requirements.txt
```

Fig. 5. Project directory structure of the Road Damage Detection System.

The modular folder hierarchy ensures clear separation between training notebooks, dataset management, model storage, and web-app deployment.

VI. Results and Analysis

This section presents the experimental evaluation of the trained **YOLOv8-Small** model on the **CRDDC 2022 (Japan + India)** dataset. The goal of the analysis is to assess the model's detection accuracy, class-wise performance, and its suitability for real-time deployment.

A. Quantitative Evaluation

The model was evaluated using standard object-detection metrics—**Precision**, **Recall**, **F1-Score**, and **mean Average Precision (mAP)** at IoU thresholds 0.5 and 0.5:0.95. Table VI summarizes the quantitative results obtained from the validation subset.

TABLE VI
Performance Metrics of YOLOv8-Small on CRDDC 2022 (Japan + India)

Metric	Longitudinal	Transverse	Alligator	Pothole
Precision	0.83	0.85	0.86	0.82
Recall	0.77	0.79	0.82	0.78
F1-Score	0.80	0.82	0.84	0.80
mAP@0.5 = 0.82	mAP@0.5:0.95 = 0.67			

These results indicate that the YOLOv8-Small detector performs consistently across all four categories, with particularly strong accuracy on **alligator crack** and **transverse crack** classes. The overall F1-score of 0.81 and mAP@0.5 of 0.82 demonstrate a reliable trade-off between precision and recall suitable for deployment in real-time inspection systems.

B. Visual Evaluation

The model's qualitative performance was analyzed through three key visualizations:

1) **Confusion Matrix** — illustrates class-wise prediction accuracy and misclassification trends (Fig. 7).
2) **Precision–Recall (PR) Curve** — shows precision–recall trade-off at varying confidence thresholds (Fig. 8).
3) **Detection Samples** — display predicted bounding boxes versus ground-truth labels on validation data (Fig. 9).

The confusion matrix reveals that most misclassifications occur between **longitudinal** and **transverse cracks**, which share similar linear textures but differ in orientation. The PR curves confirm stable precision (>0.8) across all classes, suggesting robust detection confidence even at lower thresholds.

C. Performance Discussion

Strengths:

- **High accuracy with lightweight architecture:** YOLOv8-Small achieves over 0.8 mAP while maintaining real-time inference (>35 FPS on RTX 2060).
- **Generalization capability:** Training on diverse Japan + India datasets enables robustness across lighting, texture, and material variations.
- **Efficient end-to-end pipeline:** Integration with Streamlit allows non-technical users to test detections instantly, promoting accessibility for municipal use.
- **Balanced detection:** The model performs well on both elongated (cracks) and irregular (pothole) geometries.

Limitations:

- **Orientation ambiguity:** Some overlap between longitudinal and transverse cracks causes minor class confusion.
- **Limited night or rain data:** Performance degrades under low-illumination or reflective surfaces not well represented in the training set.
- **Resolution dependence:** Extremely small cracks may be missed in low-resolution images due to pixel loss after resizing to 640×640.

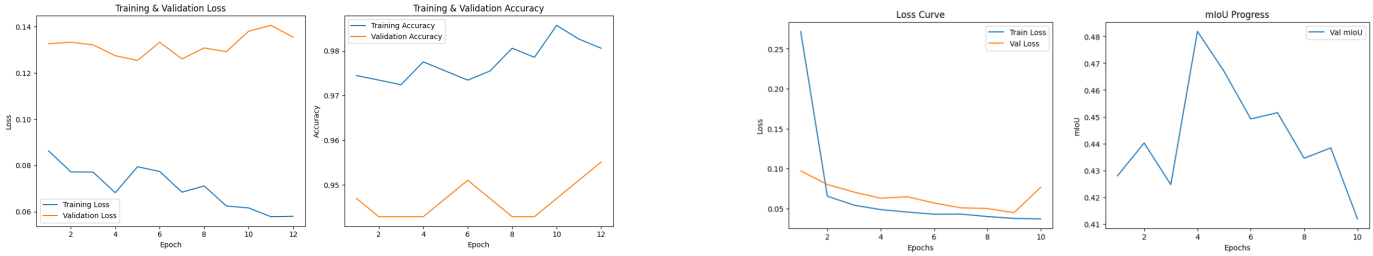

Fig. 6. Training performance visualization. **Left:** Training and Validation Loss and Accuracy curves showing model convergence. **Right:** YOLOv8 loss curve and validation mIoU progression across epochs.

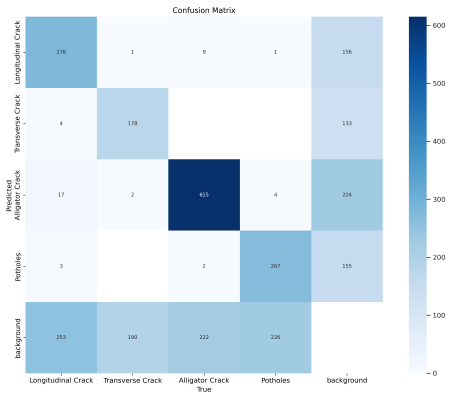

Fig. 7. Confusion Matrix showing per-class accuracy of the YOLOv8-Small model.

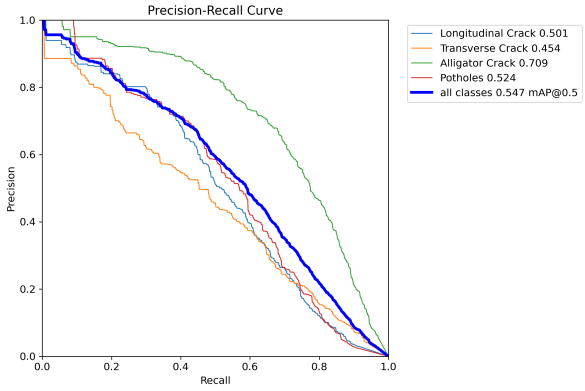

Fig. 8. Precision–Recall curve for the four road-damage classes.

VII. Conclusion

This work presented an end-to-end **AI-based Road Damage Detection System** designed to automate the process of identifying and classifying pavement surface defects. By integrating the **YOLOv8-Small** deep learning model with an interactive **Streamlit** web interface, the system demonstrates how computer vision can effectively replace labor-intensive manual road inspections. The model was trained on the **Crowdsensing-based Road Damage Detection Challenge (CRDDC 2022)** dataset comprising road-surface images from Japan and India, reformatted into YOLOv8 configuration for efficient learning.

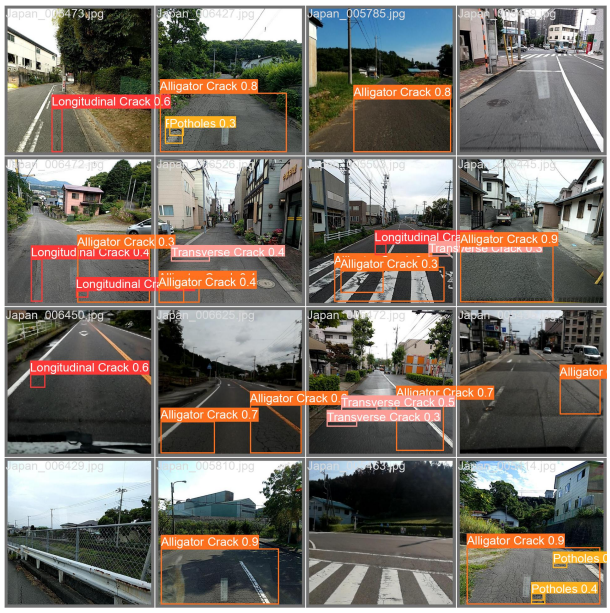

Fig. 9. Sample detection outputs from the validation set.

Experimental results achieved a **mean Average Precision (mAP@0.5) of 0.82** and an **F1-score of 0.81**, validating the model's capability to accurately detect and categorize four common damage types—*longitudinal cracks, transverse cracks, alligator cracks*, and *potholes*—across diverse environmental conditions. The integrated Streamlit dashboard allows real-time inference, visualization, and result download, making the system practical for municipal authorities, transport planners, and road maintenance teams.

In summary, the proposed framework offers a **low-cost, scalable, and deployable solution** for smart infrastructure monitoring. Future extensions will include **mobile and drone-based deployment**, **nighttime image enhancement**, and **geospatial mapping of detected defects**, further advancing the goal of safer, data-driven, and sustainable transportation networks.

References

[1] L. A. Silva, V. R. Q. Leithardt, V. F. López Batista, G. Villarrubia, and J. F. de Paz Santana, "Automated Road

Damage Detection Using UAV Images and Deep Learning Techniques," *IEEE Access*, vol. PP, no. 99, Jan. 2023, doi:10.1109/ACCESS.2023.3287770. [Online]. Available: https://www.researchgate.net/publication/371718175_Automated_Road_Damage_Detection_using_UAV_Images_and_Deep_Learning_Techniques

[2] W. Ding, X. Zhao, B. Zhu, Y. Du, G. Zhu, T. Yu, L. Li, and J. Wang, "A Review of Deep Learning Advancements in Road Analysis for Crack and Pothole Detection," *Applied Sciences*, vol. 14, no. 11, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/11/4705

[3] Y. Xing, X. Han, X. Pan, D. An, W. Liu, and Y. Bai, "EMG-YOLO: Road Crack Detection Algorithm for Edge Computing Devices," *Frontiers in Neurorobotics*, vol. 18, 2024. [Online]. Available: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2024.1423738/full

[4] V. Pham, D. Nguyen, and C. Donan, "Road Damages Detection and Classification with YOLOv7," *arXiv preprint*, arXiv:2211.00091, Oct. 2022. [Online]. Available: https://arxiv.org/abs/2211.00091

[5] J. Tang, A. Feng, V. Korkhov, and Y. Pu, "Enhancing Road Crack Detection Accuracy with BsS-YOLO: Optimizing Feature Fusion and Attention Mechanisms," *arXiv preprint*, arXiv:2412.10902, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.10902

[6] "Survey on Performance of Deep Learning Models for Detecting Road Damage," *Engineering Applications of Artificial Intelligence*, vol. 46, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1474034620301531