Scientific Reports

https://doi.org/10.1038/s41598-026-46679-4

Article in Press

Real-world road damage dataset with potholes, cracks, and maintenance holes

Received: 19 December 2025

Accepted: 27 March 2026

Published online: 01 April 2026

Cite this article as: Giordani E., Arcioni L., Gil-Martín M. *et al.* Real-world road damage dataset with potholes, cracks, and maintenance holes. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-46679-4

Enrico Giordani, Lorenzo Arcioni, Manuel Gil-Martín, Gian Luca Foresti & Marco Raoul Marini

Real-World Road Damage Dataset with Potholes, Cracks, and Maintenance Holes

Enrico Giordani[1,+], Lorenzo Arcioni[1,+], Manuel Gil-Martín[2], Gian Luca Foresti[3], and Marco Raoul Marini[1,*]

[1]VisionLab, Department of Computer Science, Sapienza University, Rome, 00198, Italy
[2]Grupo de Tecnología del Habla y Aprendizaje Automático (THAU Group), Information Processing and Telecommunications Center, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid (UPM), Madrid, 28040, Spain
[3]Department of Computer Science, Mathematics and Physics, University of Udine, Via delle Scienze 206, Udine, UD 33100, Italy
[*]Corresponding Author, marini@di.uniroma1.it
[+]these authors contributed equally to this work

ABSTRACT

Road surface deterioration poses significant challenges to transportation safety, infrastructure longevity, and timely maintenance planning. Existing street-view datasets are often limited by wide-angle distortions that reduce geometric fidelity and hinder reliable damage analysis. This paper introduces the Road Damage Dataset: Potholes, Cracks, and Manholes, a novel dataset designed for robust detection of road-surface damage in urban and rural settings. The dataset was captured using two consumer-grade devices, acquiring diverse views that mimic real-world deployment situations. It contains high-resolution images with three major and often co-occurring road-damage classes: potholes, cracks, and maintenance holes. It includes 2,009 hand-labeled images containing 1,261 potholes, 2,519 cracks, and 957 maintenance holes with verified bounding boxes. All images were post-processed to improve visual quality and remove sensitive information. The dataset includes several districts in Rome (Italy) and nearby semi-urban and rural towns such as Sacrofano, offering more environmental heterogeneity than many existing datasets. Thanks to its varied capture circumstances, viewing angles, and scene contexts, this dataset supports the development of generalizable models for real-world road-damage detection.

Background & Summary

Degradation of pavement surfaces is a persistent issue in modern societies, especially when left unattended, as it can rapidly escalate into a costly infrastructure breakdown. The formation of cracks, potholes, or surface deterioration increases repair costs and poses serious risks to traffic safety and urban mobility. Therefore, early identification and repair are critical, yet traditional road maintenance strategies are inadequate due to their reactive nature and dependence on periodic manual inspection routines. Traditional methods for assessing road conditions, such as manual visual inspections or vibration-based surveys, are labor-intensive, subjective, and often hazardous for inspectors[1]. With the advent of deep learning and computer vision technologies, automated approaches have gained prominence, leveraging image data captured from vehicles, smartphones, or street-view platforms to detect and classify damage, such as cracks, potholes, and surface degradation.

As more transport networks develop and cities become increasingly complex, there is growing interest in applying artificial intelligence to power automated road condition monitoring. Artificial intelligence-based systems, particularly those using computer vision, can detect damage patterns early and effectively, reducing costs and minimizing traffic disruptions. Such systems rely heavily on annotated visual datasets to learn to identify road anomalies under various conditions.

In this context, advances in object detection, image classification, and semantic segmentation have enabled new lines of research in road condition monitoring through image-based methods. However, the applicability and performance of these approaches are significantly dependent on the realism and quality of the related datasets. Most existing road damage datasets are obtained from expensive devices unsuited for real-world applications, such as Google Street View or Baidu Maps, which may pose non-realistic imaging conditions, including artificial camera poses, wide-angle distortions, or uneven illumination. These limitations may limit the model's applicability in real mobile environments.

Numerous datasets have been developed to support the training and evaluation of automated models for detecting road damage. These datasets vary in size, geographic coverage, annotation types, and focus, often emphasizing cracks, potholes, or surface quality. Early datasets were limited in scale and diversity, but recent efforts have expanded to include multi-country data and long-term tracking.

As mentioned earlier, current publicly available road damage detection datasets have several limitations in terms of content, size, and usability. Some datasets, such as the Crack Forest Dataset (CFD)[2], focus solely on the segmentation of cracks, annotating only two classes: cracks and background, with a small number of images, typically in the hundreds. They are typically derived from top-view photographs obtained using smartphones or vehicle-mounted cameras and exhibit limited consistency in image resolution, as well as a variety of damage. More comprehensive data sets such as the German Asphalt Pavement Distress (GAPs)[3] provide thousands of annotated images that include cracks, potholes, inlaid patches, applied patches, open joints, and bleedings. Another example of these datasets is the Road Damage Dataset (RDD)[4] series, whose latest release comprises 47,420 images of varying resolutions, annotated with potholes and other types of cracks in cities across Japan, India, the Czech Republic, Norway, the United States, and China. These richer datasets are suitable for object detection and fine-grained analysis. However, their excessive data acquisition costs and specialized imaging hardware limit their scalability and real-world deployment. Moreover, the majority of existing road damage datasets rely on images recorded from online platforms such as Google Street View[5] or Baidu Maps[6], which may not reflect real on-road conditions in terms of camera placement, road texture, or lighting. These websites attach camera rigs to cars to record geotagged street view images at different angles on roads, which are subsequently processed to annotate damage. Although this method allows low-cost data collection on a large scale, it is at the expense of image standardization, realism, and controllability of capture parameters. In particular, the Street View Image Dataset for Automated Road Damage Detection (SVRDD)[7] was built with 8,000 street view images from Baidu Maps. More than 20,000 images of road damage were visually recognized and annotated in five administrative districts of Beijing City from these images. Although SVRDD offers various types of roads and surface conditions, its geographical range remains restricted, in contrast to the significant variability of road infrastructures in larger regions or countries.

One of the foundational datasets is the Road Damage Dataset 2020 (RDD2020), comprising 26,336 images collected from smartphones in India, Japan, and the Czech Republic[1,8]. It includes over 31,000 annotations for four damage types: longitudinal cracks, transverse cracks, alligator cracks, and potholes. RDD2020 was utilized in the Global Road Damage Detection Challenge (GRDDC) 2020, where models such as Faster R-CNN and YOLO were benchmarked, achieving F1 scores of around 0.66 on the test sets[1,9–11]. An extension, RDD2021, added 100,000 synthetic labeled images for domain adaptation studies[12].

The StreetSurfaceVis dataset introduces 9,122 crowdsourced street-level images, primarily from Germany, annotated by surface type (e.g., asphalt, sett) and quality[13]. Similarly, the Asphalt Road Surface Disease Dataset (ARSDD) comprises 2,297 high-resolution images from real-world projects, annotated according to road maintenance guidelines for six damage categories[14].

Larger-scale datasets include a recent repository with 51,012 images for distress identification and 8,928 for long-term tracking, enabling the prediction of degradation trends[15]. The CNRDD dataset, labeled according to China's JTG5210-2018 standards, features eight categories with severity levels (mild, moderate, severe)[16].

Other notable datasets originate from specific challenges, such as the Crowdsensing-based Road Damage Detection Challenge 2022 (CRDDC2022), which focuses on global applicability[17]. Japanese-focused datasets from earlier challenges (e.g., 2018 IEEE BigData Cup) provided initial benchmarks but were limited to one country[18].

These datasets have driven progress, but challenges persist: many are geographically constrained (e.g., a single-country focus), lack long-term tracking for degradation analysis, or have imbalanced classes (e.g., fewer instances of potholes). Moreover, annotations often vary in quality and standardization, and few include explicit baselines for reproducibility.

Table 1. Comparison of Key Road Damage Detection Datasets

Dataset	Images	Damage Types	Geographic Coverage
RDD2020[8]	26,336	4 (cracks, potholes)	India, Japan, Czech Republic
StreetSurfaceVis[13]	9,122	Surface types/quality	Mostly Germany
SVRDD[7]	8,000	Various damages	Beijing, China
ARSDD[14]	2,297	6 categories	Real-world projects (unspecified)
Large-Scale Repository[15]	51,012 + 8,928 (tracking)	Distress types	Unspecified (large-scale)
CNRDD[16]	Unspecified	8 with severity	China
CRDDC2022[17]	Challenge-specific	Various	Multi-country

Automated methods have evolved from traditional image processing to deep learning-based approaches. Early techniques used feature extraction like Gabor filters or histogram of oriented gradients for crack detection but struggled with environmental variations[19].

Deep learning methods dominate recent work, categorized into classification, object detection, and segmentation. Classification approaches, such as CNNs for detecting damage presence, are efficient but lack localization capabilities. Object detection models like Faster R-CNN[9–11] and YOLO variants (e.g., YOLOv5[20], YOLOv7[17]) provide bounding boxes and classes, achieving F1 scores of 0.49–0.74 on RDD2020[1]. For instance, ensemble YOLO models yielded an F1 score of 0.67 on the GRDDC test sets[1].

Domain adaptation addresses cross-country generalization, as in DA-RDD, which uses adversarial learning and synthetic data from RDD2021 to align features across domains[12]. Attention mechanisms enhance baselines, e.g., coordinate attention in YOLOv7[17] or attention fusion in CNRDD's model[16].

Segmentation methods offer pixel-level accuracy but are computationally intensive[19].

Despite advancements, models often overfit to specific datasets, leading to poor performance on unseen data (e.g., from different countries or weather conditions). Baselines are rarely provided, hindering direct comparisons.

redExisting datasets and methods have advanced the field, but critical gaps constrain their applicability to emerging deployment scenarios. While large-scale datasets such as RDD2022[4] provide extensive geographic coverage, their unlabeled device heterogeneity prevents systematic cross-device generalization assessment, which is essential when deployment hardware differs from training conditions. Street-view platforms introduce artificial imaging conditions (wide-angle distortions, inconsistent preprocessing) that reduce experimental controllability. Most datasets omit maintenance holes despite their visual similarity to potholes, limiting development of discriminative models that must distinguish infrastructure from damage.

The broader road perception landscape encompasses complementary tasks beyond damage detection. Road surface classification using texture features supports applications such as active suspension control in intelligent vehicles, where distinguishing between structured surfaces (asphalt, cement) and unstructured surfaces (grass, earth) enables real-time adaptation of vehicle control strategies[21]. While such texture-based classification addresses material composition and surface type, our dataset focuses specifically on structural damage detection (potholes, cracks, and maintenance holes) for infrastructure maintenance prioritization. Both approaches leverage visual analysis but serve distinct operational needs: surface-type recognition for vehicle-system optimization versus damage localization for repair planning.

End-to-end neural networks with multi-level attention mechanisms[22] have demonstrated effectiveness in pavement damage detection by adaptively weighting spatial and channel features at multiple network depths. These attention-driven architectures are particularly valuable for crack detection, where subtle visual cues must be discriminated from road markings and texture variations, thereby reducing false positives. Our dataset supports development of such models through diverse imaging conditions and deliberate inclusion of maintenance holes, a challenging class often omitted from existing datasets despite its visual similarity to potholes.

Recent work by MYCD[23] achieved competitive performance (mAP@50 = 0.677) using a DenseNet-enhanced YOLO architecture evaluated on a field-collected dataset acquired manually with a smartphone. While this setup reflects realistic field conditions, handheld acquisition may allow operators to capture damage instances from favorable viewpoints and distances. In contrast, our dataset is also collected using a fixed vehicle-mounted setup, where viewing angle, camera height, and distance to the pavement are constrained by deployment conditions. This configuration introduces more challenging geometric perspectives and partial visibility, better reflecting real-world on-road monitoring scenarios. As such, our evaluation considers not only detection accuracy but robustness under constrained acquisition geometry.

Our work addresses these issues by introducing a new dataset that emphasizes potholes and temporal degradation signs, collected from diverse sources with standardized annotations. In addition to the dataset, we provide a comprehensive benchmark including standard baseline models as well as comparisons with state-of-the-art optimized architectures, designed to fully explore and leverage the dataset's capabilities.

Based on the context presented, the dataset in this work was manually collected in real-world environments using two low-cost devices: a GoPro HERO7 camera mounted on the dashboard of a moving vehicle and a Samsung Galaxy A14 smartphone for images taken from outside the car while standing. This dual-device configuration captures diverse perspectives, simulating realistic deployment scenarios for future road inspection systems. The dataset comprises images captured in various urban and rural settings around Rome and Sacrofano, providing a range of backgrounds, pavement textures, and environmental conditions.

To the best of our knowledge, currently available and widely adopted road damage datasets do not include imagery collected on Italian road networks. This absence limits the geographic and infrastructural diversity represented in existing benchmarks. By introducing images captured in Italian urban and rural contexts, characterized by distinct pavement materials, maintenance standards, and traffic-induced wear patterns, our dataset contributes to a previously underrepresented domain, enabling more geographically robust evaluation and reducing regional bias in model development.

Moreover, the proposed dataset offers three prevalent and visually confusing damage types: potholes, cracks, and maintenance holes. Unlike most other datasets that omit maintenance holes or misclassify them, we intentionally annotate this class, bringing in a valuable component of richness and real-world utility. All the annotations were performed using a

YOLO-compatible tool designed for consistency and accuracy. This way, the dataset supports robust training and testing of artificial intelligence models that generalize well across devices and deployment environments. By integrating real imagery, operational realism, and dense annotations, the dataset provides a solid foundation for advancing research on automated, device-independent road damage detection and intelligent road maintenance.

The main contributions of this work are as follows:

- Introduction of a novel road damage dataset of realistic images captured using two different consumer-grade devices.
- Depiction of diverse urban and rural locations in Italy, both central and peripheral zones, with visual contrast on the road surface and conditions.
- Annotation for three principal categories of road deterioration, such as potholes, cracks, and maintenance holes, where the latter is often overlooked but crucial due to its visual similarity to potholes.
- Support for controlled cross-device evaluation, allowing experiments where models are trained on images from one device and evaluated on the other to assess generalization robustness.
- Practical applicability: by demonstrating that standard object detection methods can be effectively implemented on consumer-grade devices, this study lowers the barrier to entry for street maintenance applications.

Methods

This section describes the materials and recording setup used for the data acquisition, the processing steps to curate the recorded data, and the annotation process. In addition, a detailed description of the dataset is also included. Figure 1 shows the pipeline followed to generate the dataset, including different modules.

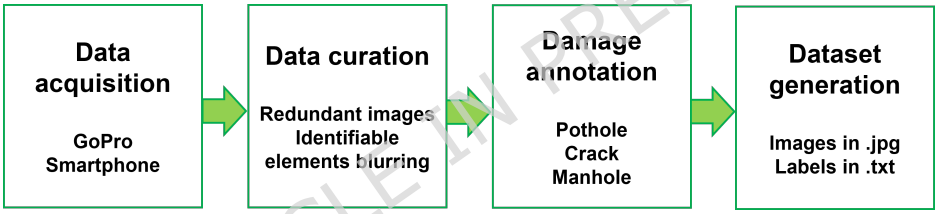

Figure 1. Dataset generation pipeline.

Data Acquisition

The visual data in this dataset were collected with two distinct devices: the GoPro HERO7 camera and the Samsung Galaxy A14 smartphone. These were selected to represent average, straightforward, and low-cost image acquisition equipment, simulating several real-world usage scenarios of urban mobility applications.

The GoPro HERO7 was configured to capture video at a native resolution of 1920×1080 pixels at 30 frames per second with a wide-angle field-of-view mode (approximately 120° horizontal FOV). Electronic image stabilization (HyperSmooth) was enabled to reduce motion artifacts during vehicle movement. The camera was mounted inside the vehicle using a suction cup mount affixed to the windshield, positioned to approximate a seated human eye-level perspective. The mounting height was set at 130 cm from ground level with a forward-facing angle of 15°–20° below the horizontal plane. This configuration was designed to simulate realistic viewpoints encountered in driver-assistance systems and mobile road inspection scenarios. All images released in the dataset were subsequently resized to 640×360 pixels since road damage patterns remain clearly recognizable at this resolution, and higher resolutions would increase computational cost without significant performance benefits. This choice reflects a deliberate trade-off between visual fidelity and efficiency, making the dataset suitable for edge and embedded deployment scenarios with limited hardware resources.

By contrast, a Samsung Galaxy A14 smartphone was used to take photographs while the car was stationary, providing a different perspective.

The Samsung Galaxy A14 captured images at a native resolution of 1280×720 pixels. The device was handheld or temporarily supported inside the stationary vehicle at an approximate height of 100–140 cm, with a viewing angle comparable to the GoPro setup. Smartphone images were also resized to 640×360 pixels for consistency within the released dataset.

Data were gathered with a heterogeneous method. To ensure adequate spatial resolution and minimize motion blur in extracted frames, data collection was conducted at vehicle speeds ranging from 0 to 40 km/h. This speed constraint enabled the capture of sharp, high-quality images suitable for detailed damage annotation while maintaining representative coverage of

urban, suburban, and semi-rural road networks. Frames were extracted from continuous video recordings at irregular intervals based on the presence and diversity of visible road damage, ensuring a balanced representation of damage types and varying environmental conditions. The smartphone was used to capture individual photos, manually triggered to capture particular examples of road damage under controlled framing and lighting conditions.

The dataset includes asphalt road surfaces from urban, suburban, and rural areas within the metropolitan region of Rome and the municipality of Sacrofano (Italy). Data acquisition took place under varying lighting conditions (sunny and cloudy daylight conditions, including morning and afternoon acquisitions, as well as rainy weather).

Figure 2 shows the in-car GoPro configuration used for gathering data, with the dashboard mounting and the forward-looking angle from within the windshield.

Figure 2. In-vehicle GoPro configuration used for data collection.

Images from the GoPro perspective were captured in urban districts such as Grottaferrata, Lariano, Morena, Rocca di Papa, Tor Bella Monaca, Tor Vergata, and Velletri. Smartphone images were collected in the village of Sacrofano, located northeast of Rome. The specific routes followed during data collection[24] are shown in Figure 3.

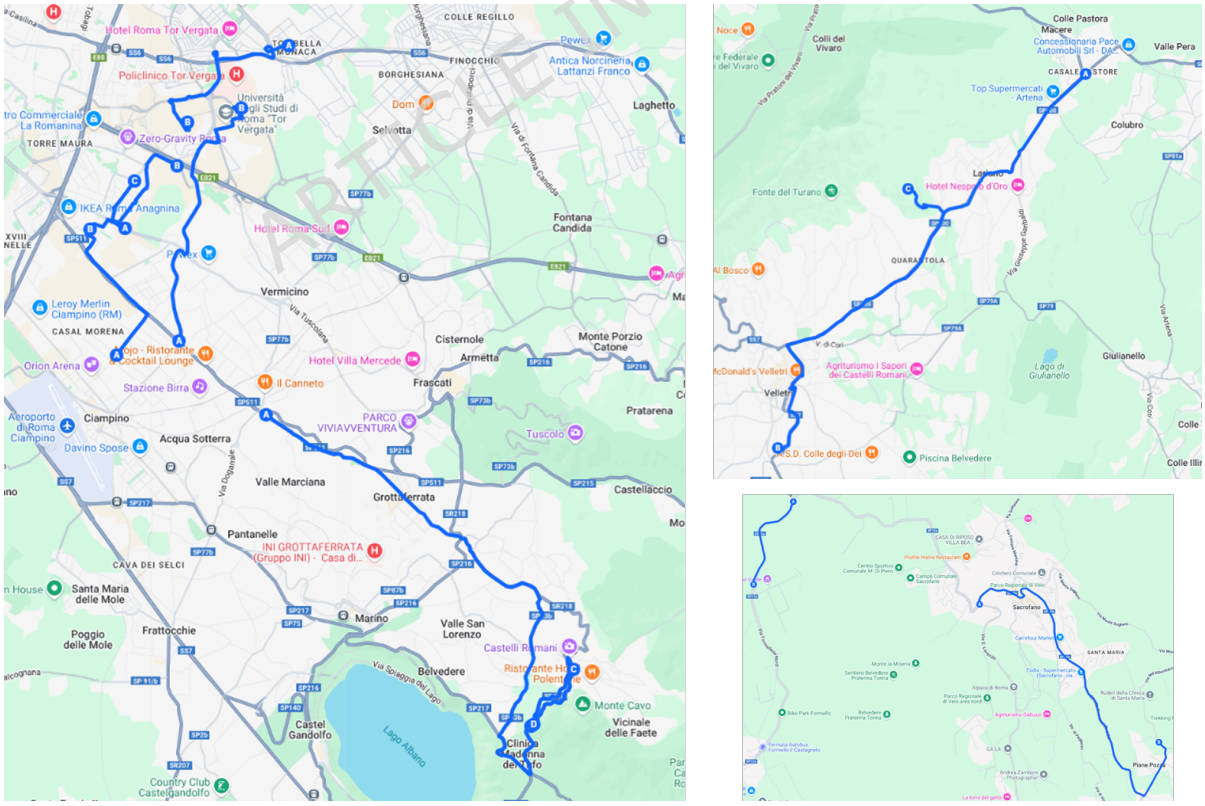

Figure 3. Routes followed during the data collection. Map data ©2025 Google.

Data curation and preparation

Following data collection, all images underwent a strict post-processing process to preserve privacy and guarantee the integrity of the dataset. Specifically, personally identifiable elements such as vehicle license plates and human faces in street view images were selectively blurred. This was done to comply with ethical standards for data use and avoid including personally identifiable information within the dataset. The chosen images were then screened to confirm that no sensitive visual content was present, thus preparing the dataset for public release and research applications.

In addition to filtering out privacy content, a rigorous image selection process was carried out before annotation. This involved removing poor-quality images that could impact the training and testing of machine learning models. Specifically, images that were significantly blurred, overexposed, underexposed, exhibited poor contrast, or contained no road damage to be annotated were excluded. Also images affected by strong occlusion, extreme lighting conditions, nighttime scenarios, or heavy rainfall were excluded from the final dataset to ensure annotation reliability and visual clarity. Furthermore, visually redundant frames, such as those capturing nearly identical images due to minimal variation in camera perspective, were removed to reduce dataset redundancy and bias. The curation step ensured that the resultant dataset contains high-quality, varied examples that are informative and effective for robust model training and benchmarking.

Damage annotation process

We utilized the DigitalSreeni Image Annotator and Toolkit[25], a feature-rich, PyQt5-based manual and semi-automatic image annotation tool, to annotate the dataset. The application supports both rectangular and polygon annotations, and it also includes support for paintbrush and eraser tools, along with multi-dimensional image format support, such as TIFF stacks and CZI files.

One of the basic functionalities of the tool is the inclusion of the Segment Anything Model (SAM2)[26] for semi-automatic annotations. The tool allows users to select a light-weight pre-trained SAM2 model, such as the tiny model, and activate it by defining an object using a bounding box. The tool automatically generates segmentation proposals, which can be manually corrected using brush tools and combined with other segments as needed. Manual annotation is also provided for areas where the model performs poorly, such as in images of poor quality or with blurry object edges.

The application enables projects to be saved, loaded, and exported in various general-purpose annotation formats, including COCO JSON, YOLO v8/v11, Pascal VOC, and semantic labels. The application provides annotation class management, visibility toggling, area measurement display, and in-place editing of active annotations.

For the annotation process of this dataset, we conducted manual annotation by selecting the appropriate damage class (pothole, crack, or maintenance hole) and drawing bounding rectangles on road surface features to achieve exact object boundary delineation based on the annotator's visual perception. The original annotations are provided as four-point polygons defined by eight normalized coordinates. However, all annotations correspond to axis-aligned rectangular bounding boxes. The polygons are ordered, convex, and non-rotated. Thus, no rotated bounding boxes are present in the dataset. To illustrate the annotation process using the tool, Figures 4 and 5 display samples of annotated images captured using the smartphone and the GoPro, respectively, with varying numbers of road damage objects. In the visual markings, potholes are marked with green rectangles, cracks with dark blue rectangles, and maintenance holes with light blue rectangles. This coloring makes it easy to identify and distinguish the marked object classes in manual review.

Data Records

The dataset is structured to support seamless integration with modern object detection frameworks such as YOLO (You Only Look Once)[27]. It is organized into a top-level `data/` directory that includes various subfolders for images and annotations:

- `images/`: stores road scene images in `.jpg` format. Each image may contain one or more instances of road surface damage such as potholes, cracks, or maintenance holes. The images whose filenames start with "vlcsnap" were taken with the GoPro device, and the rest were taken with the smartphone.

- `labels/`: contains the corresponding annotation files in `.txt` format with polygon-based labels. Each file shares its base name with the corresponding image and contains one line per annotated object. Each annotation consists of a class ID (0 = pothole, 1 = crack, 2 = manhole), followed by eight normalized values representing the coordinates of the four polygon vertices outlining the damaged area, expressed as $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$. All coordinates are normalized in the range [0,1] with respect to the image width and height.

- `labels-YOLO/`: provides axis-aligned bounding box annotations in YOLO format `(class_id, x_center, y_center, width, height)`, derived from the polygon annotations and normalized in the $[0, 1]$ range. Each file shares its base name with the corresponding image and contains one line per annotated object. This format is directly compatible with YOLO-based object detection frameworks.

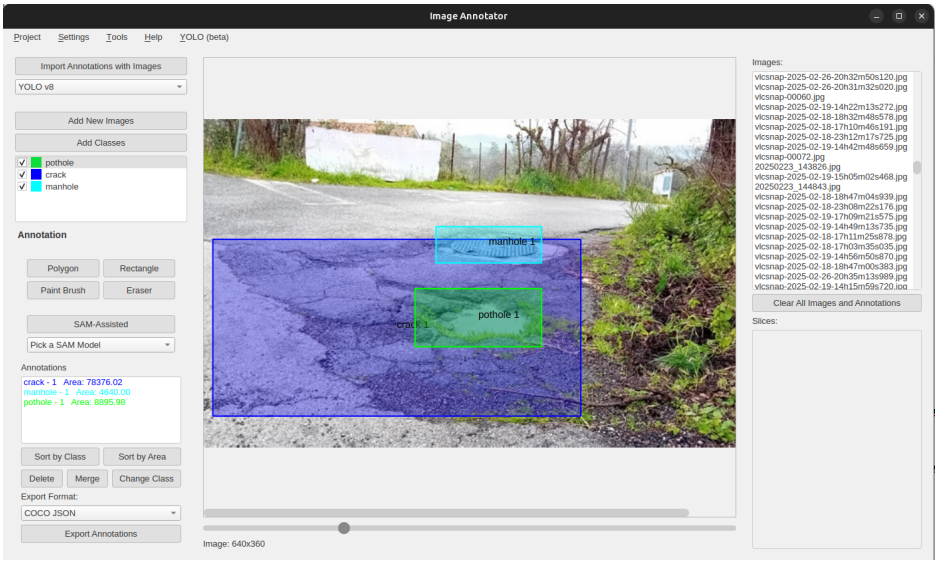

Figure 4. Smartphone image taken outside the car during annotation with one pothole, crack, and maintenance hole.

- `annotations_coco.json`: contains the dataset annotations in COCO JSON format, with bounding boxes expressed as `[x_min, y_min, width, height]` in pixel coordinates, enabling compatibility with COCO-based detection pipelines.
- `YOLO-conversion-script.py`: converts the polygon-based annotations in `labels/` into YOLO-format bounding boxes stored in `labels-YOLO/`.
- `COCO-conversion-script.py`: converts the polygon-based annotations in `labels/` into the COCO JSON annotation file `annotations_coco.json`.

All images were collected under diverse real-world conditions, including various lighting scenarios (e.g., bright daylight, overcast skies, low-light environments), different road surface types, and diverse weather contexts. In particular, the dataset images were captured in dry and wet conditions. Figure 6 shows examples from the images in the dataset. This variety enhances the realism of the dataset, ensuring that the trained models can generalize to challenging visual conditions. The clear and consistent folder structure ensures easy loading, parsing, and reproducibility within machine learning workflows.

Dataset overview

The "Road Damage Dataset: Potholes, Cracks and Manholes" comprises 2,009 annotated images of road surface conditions, collected across various urban areas and rural roads to ensure diversity in visual appearance and scene context. These images were acquired in February and March 2,025 under various lighting conditions and weather scenarios, including rainwater in potholes, to support robust, generalizable detection models. Additionally, the dataset comprises 1,907 images collected with a GoPro HERO7 camera and 102 images collected with a Samsung Galaxy A14 smartphone.

The annotations cover three types of road surface damage and features, described as follows:

- **Pothole:** A bowl-shaped depression or hole in a road surface, typically formed when the pavement material breaks away due to traffic and weather. Sharp edges and vertical sides near the top of the hole characterize them. Potholes can range from a few centimeters to a meter wide and several centimeters deep.
- **Crack:** A narrow, linear fracture or separation in the pavement surface caused by stresses from traffic loads, temperature variations, or aging of materials. Cracks can vary in length, width, and pattern, appearing as straight lines, branching networks, or irregular shapes.
- **Manhole:** A covered opening in a street, sidewalk, or other surface that provides access to underground utilities like sewers, water lines, or communication cables. These openings allow for inspection, maintenance, and repair of the infrastructure below. The cover, typically made of metal, is designed to be removable and to withstand traffic and other loads.

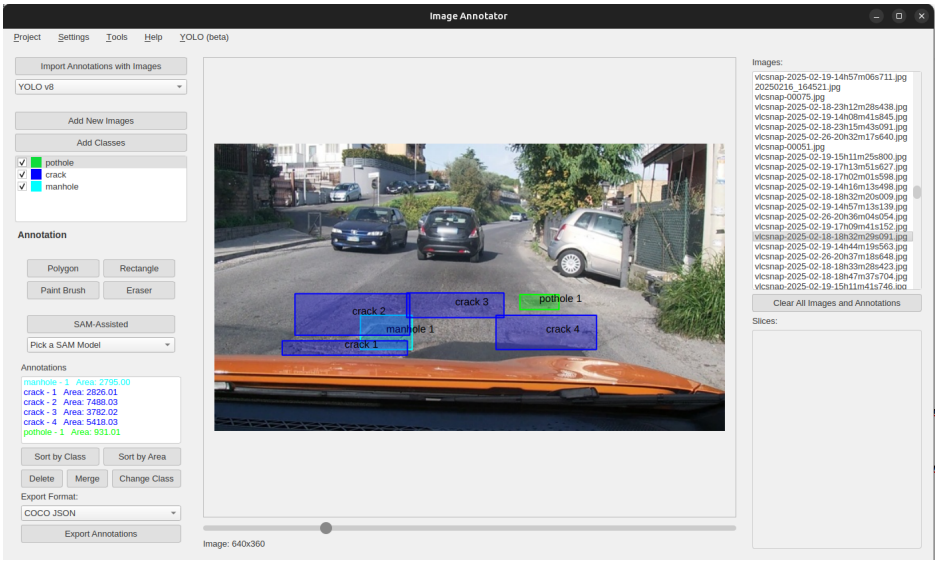

Figure 5. GoPro image taken inside the car during annotation with one pothole, four cracks, and one maintenance hole.

Table 2 summarizes the total counts and average number of objects per image for each road damage class (pothole, crack, maintenance hole). The dataset consistently contains the highest average number of cracks per image, followed by potholes and maintenance holes across both subsets. This trend is visually reinforced in Figure 7, which displays the total road damage counts per class in the dataset: 1,261 potholes, 2,519 cracks, and 957 maintenance holes. Table 3 reports the average normalized bounding box dimensions, including standard deviations, for each class over the entire dataset. Cracks tend to have larger and more variable bounding boxes than potholes and maintenance holes, with smaller and more consistent size distributions. These patterns are further illustrated in Figure 8, showing box plots of the bounding box width and height distributions per class. Moreover, Figure 9 shows the bounding box area distribution in pixels, illustrating how potholes and maintenance holes usually have similar areas, which are bigger than the ones from the cracks. The spatial distribution of damage instances in Figure 10 reveals distinct class-specific patterns, with all damage types predominantly concentrated in the lower-center region of images corresponding to the road surface directly visible from the dashboard camera perspective. Potholes exhibit the most compact clustering, while cracks demonstrate a broader horizontal spread reflecting their linear nature, and manholes show intermediate spatial concentration consistent with their typical placement along the driving path. These statistics and visualizations offer valuable insights into the dataset composition and object size variation, which are essential for model training and evaluation.

Class	Count	Avg per Image
pothole	1,261	0.6277
crack	2,519	1.2539
manhole	957	0.4764

Table 2. Count and average per image by road damage class.

Class	Width (mean $\pm$ std)	Height (mean $\pm$ std)
pothole	0.0856 ± 0.0634	0.0680 ± 0.0520
crack	0.1588 ± 0.1285	0.1424 ± 0.1011
manhole	0.0904 ± 0.0388	0.0568 ± 0.0250

Table 3. average $\pm$ std bounding box dimensions by class in the total dataset.

Figure 6. Examples of images from the dataset.

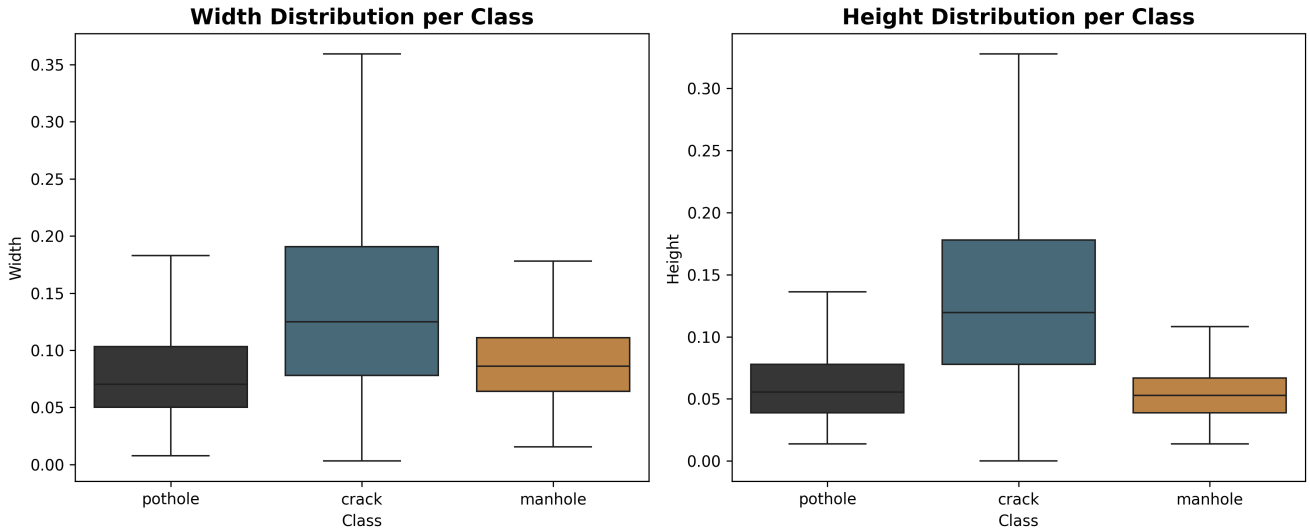

Figure 8. Box plots of normalized bounding box width and height distribution per class across the entire dataset. Cracks exhibit the largest and most variable dimensions.

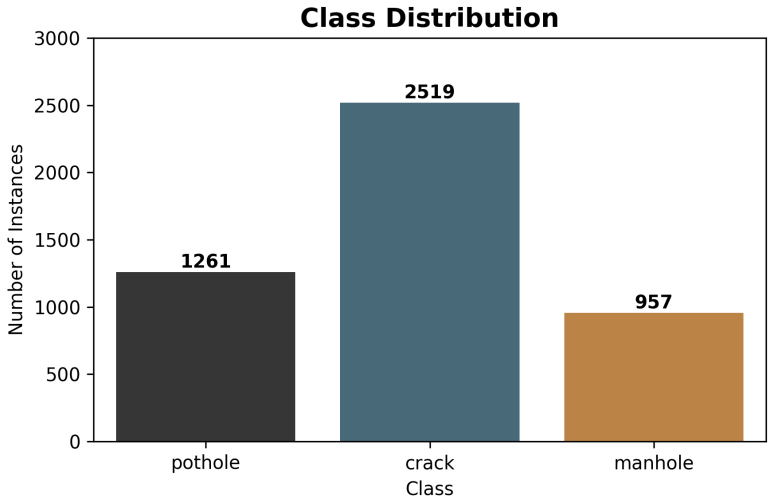

Figure 7. Total road damage target counts per class in the dataset. Cracks are the most frequent class, followed by potholes and maintenance holes.

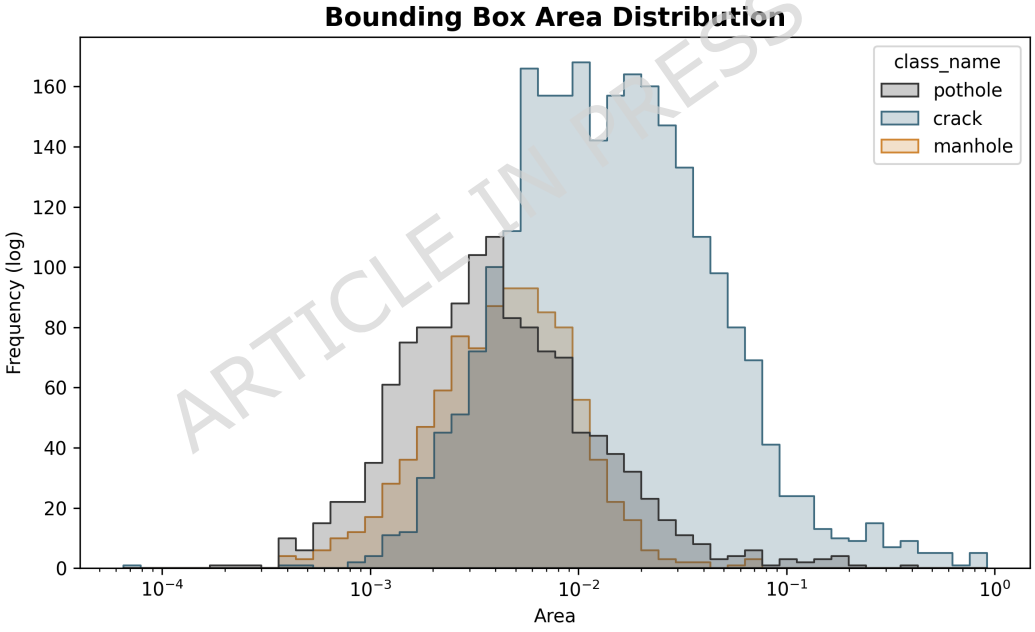

Figure 9. Bounding box area distribution per class.

A key strength of this dataset lies in its potential to support device-independent detection of road damage, particularly for frequent classes. Although manholes are underrepresented in the smartphone subset, their high detectability mitigates the effect of the imbalance, while the dataset still highlights challenges for more difficult or variable objects such as cracks, providing insight into domain shift effects in cross-device evaluation. However, we note that the smartphone subset is strongly imbalanced, with particularly few examples of manholes. Despite their scarcity, manholes are relatively easy to detect due to their distinct shape and appearance, and experimental results indicate that precision for manholes is higher than for other classes. As a result, while cross-device evaluation for rare or small-scale objects like cracks remains sensitive to domain shifts, the limited number of manhole instances has a minimal impact on overall performance and the conclusions drawn from the dataset. The dataset introduces natural variability in image characteristics, such as color balance and perspective, by including images captured with a GoPro HERO7 and a Samsung Galaxy A14 smartphone, which offer different lenses, image processing pipelines, and

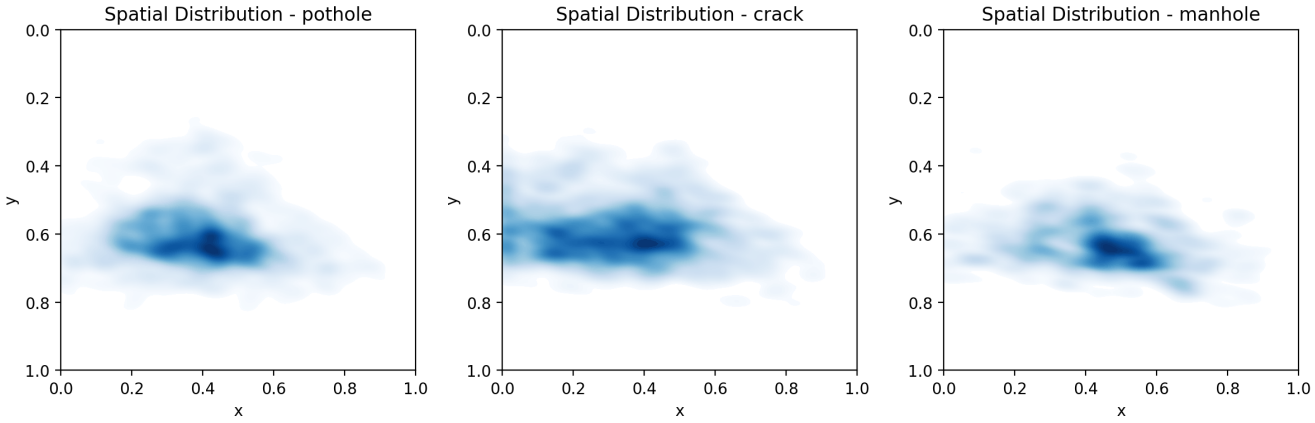

Figure 10. Spatial distribution heatmaps showing the normalized position of bounding box centers for each damage class across the dataset.

mounting positions. Although the number of images captured by the smartphone is lower, it is possible to observe in Figure 11 that they contain all the road damage targets annotated in this dataset. This diversity enables researchers to explore cross-device generalization scenarios, such as training a model exclusively on GoPro data and evaluating it on smartphone images, or vice versa. Such experiments are valuable for assessing the robustness and adaptability of computer vision models in real-world conditions, where the deployment hardware may differ from that used during training.

Moreover, another necessary strength of this dataset is the explicit annotation of maintenance holes, a standard road feature often confused with potholes in real-world scenarios. Maintenance holes and potholes can appear visually similar: circular and embedded in the road surface, showing a similar size as observed in Figure 9. However, they carry entirely different implications for detection systems. While potholes represent actual damage that requires repair, maintenance holes are infrastructure elements that should not trigger maintenance alerts. Including maintenance hole annotations enables training more discriminative models capable of distinguishing between actual road damage and harmless elements.

Technical Validation

To demonstrate the quality, practical utility, and robustness of the proposed road damage dataset, we conducted comprehensive validation experiments using state-of-the-art object detection architectures. The validation strategy was designed to assess dataset performance across multiple dimensions: achievable performance under optimal conditions, usability with standard configurations, and consistency and reliability across different architectural paradigms. This multi-faceted approach provides potential users with both the dataset's maximum potential and realistic performance expectations under typical usage scenarios.

As a benchmark, we employed a set of state-of-the-art YOLO object detection models[27] to evaluate how effectively models trained on our dataset can detect and localize key categories of road damage, specifically potholes, cracks, and manholes—across diverse real-world scenarios. Training and validation splits were generated dynamically using the k-fold cross-validation technique. In this setup, the dataset is partitioned into k subsets (folds); models are trained on $k-1$ folds and evaluated on the remaining fold, with the process repeated k times. This ensures that every instance is used for both training and validation, yielding more reliable and robust performance estimates while simultaneously assessing dataset effectiveness across different object detection architectures, ranging from lightweight mobile-optimized models to more sophisticated frameworks.

Moreover, because road-damage detection systems are typically deployed on edge devices, such as embedded processors mounted on vehicles, that have limited computational resources and cannot accommodate large-scale deep learning architectures, we deliberately choose lightweight models for our evaluation. Models such as YOLO-N and YOLO-S, as well as MobileNet-based detectors, were selected due to their reduced parameter count, fast inference speed, and suitability for resource-constrained environments. Validating the dataset using these compact architectures enables us to assess its effectiveness under realistic operational constraints, ensuring that the resulting detection systems can achieve an adequate frame rate when integrated into moving vehicles.

Validation Framework

Dataset Validation Strategy and Model Architectures

To comprehensively characterize dataset performance, we employed a dual validation paradigm combining optimized and standard configuration evaluations across architecturally diverse detection models. This approach enables assessment of both

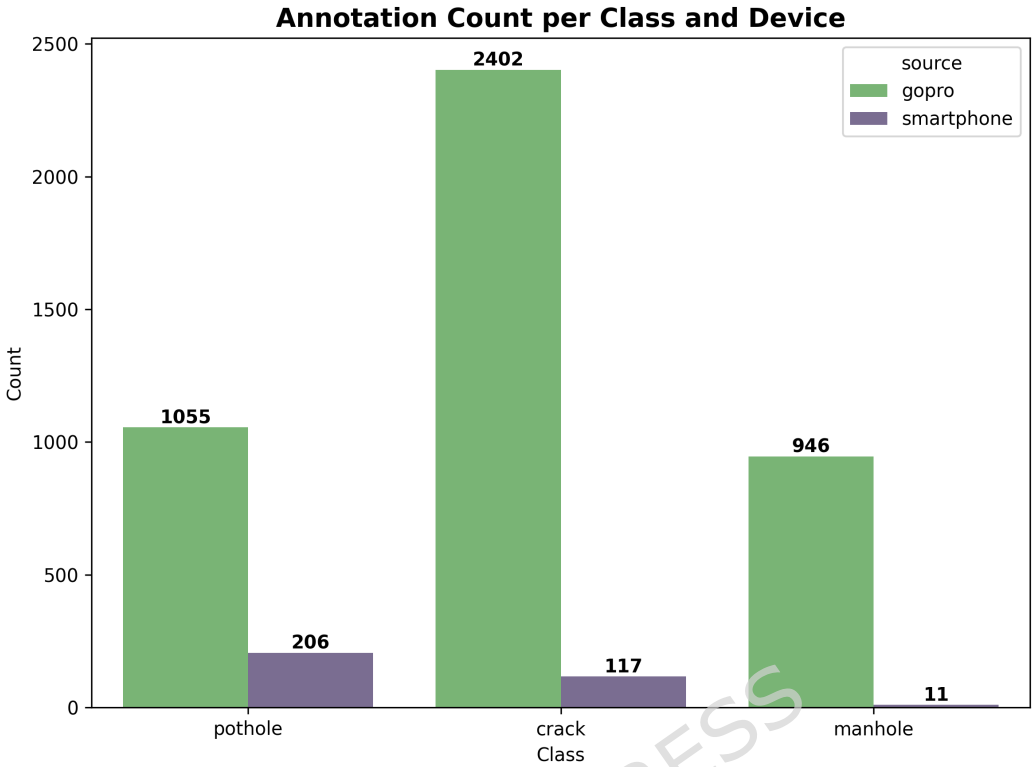

Figure 11. Total road damage target counts per class and device in the dataset.

the dataset's maximum potential under careful tuning and its practical utility in typical deployment scenarios.

For optimized performance assessment, we selected YOLOv11n[28] (nano) and YOLOv11s (small) as primary validation architectures due to their architectural innovations and practical relevance for embedded road monitoring systems. These lightweight variants incorporate modern detection mechanisms, including decoupled heads, anchor-free detection, and distribution-aware localization, making them representative of current state-of-the-art frameworks optimized for edge computing applications. Specifically, YOLOv11n provides efficient inference suitable for resource-constrained deployment on Raspberry Pi devices, while YOLOv11s offers enhanced feature representation while maintaining computational efficiency for Raspberry Pi enhanced with Hailo AI accelerators. To demonstrate the dataset's responsiveness to systematic tuning, we applied Bayesian hyperparameter optimization[29] across seven key parameters: batch size, dropout rate, initial learning rate, final learning rate, momentum, optimizer, and weight decay. The optimization process aims to maximize cross-validation precision, calculated as the average of per-class precision scores over all cross-validation folds, reflecting the critical importance of minimizing false positives in road damage detection applications.

To assess the practical usability under standard deployment conditions, we evaluated four baseline architectures with default configurations, each representing a distinct detection paradigm. YOLOv8[30] establishes a performance baseline using the previous YOLO generation, enabling the assessment of dataset compatibility with widely adopted frameworks. MobileNetV3-SSD Lite[31] represents a mobile-optimized architecture specifically designed for edge computing, demonstrating dataset suitability for resource-constrained embedded systems. VGG16-SSD[32,33] provides a comparison with traditional CNN backbones across different feature extraction paradigms, although it requires potentially higher computational resources, which can be mitigated with hardware acceleration through dedicated AI processing units. Finally, Faster R-CNN with a ResNet-50-FPN-V2 backbone[34–36] introduces a two-stage detection framework that combines region proposal networks with deep residual features, enabling evaluation under high-accuracy, high-capacity architectures. These baseline models simulate typical usage scenarios where practitioners apply existing models without extensive customization, thereby characterizing realistic performance expectations for dataset integration into standard workflows and practical embedded road damage detection systems.

Importantly, this comparison was performed while maintaining identical input data splits across all models (same cross-validation seed), ensuring comparability. All experiments were conducted using a fixed random seed (seed = 0), consistently applied to data splitting, model initialization, and training procedures. Deterministic training was enforced where supported

by the framework to maximize reproducibility, although minor non-determinism due to GPU execution may still occur. The core aim of the experiment is to observe how each model behaves out-of-the-box on this dataset. In future work, Bayesian hyperparameter optimization could be extended beyond YOLOv11n/s to all evaluated models for more exhaustive performance tuning.

Regarding input resolution, YOLO-based models were trained and evaluated using a rectangular input size of 640×360 pixels. In contrast, SSD300, SSD320, and Faster R-CNN were trained using their standard square input configurations (300×300, 320×320, and 300×300, respectively) to preserve architectural consistency and computational efficiency. For inference latency evaluation, benchmarking was performed with batch size equal to 1 under the respective input settings described above.

Cross-Validation Protocol

A 4-fold cross-validation with random partitioning (seed = 0) was implemented to ensure reproducible evaluation. The image lists for each fold are provided in the supplementary materials as `fold_1.txt` through `fold_4.txt`. Beyond standard performance assessment, this protocol specifically evaluated three complementary aspects of dataset quality. First, internal consistency was examined through performance variance analysis across folds, which revealed data homogeneity and the absence of systematic biases. Second, class-level representation was investigated through model-wise performance patterns, identifying potential strengths or limitations in specific damage categories. Third, architectural sensitivity was assessed by comparing model behavior across different detector families, demonstrating the dataset's ability to support diverse architectural paradigms without systematic preference.

To address the inherent class imbalance in road damage detection (as evidenced in Table 2), dynamic class weighting was implemented for YOLOv11 variants using inverse frequency weighting. Specifically, for each class $i \in \{0,1,2\}$, corresponding to *pothole*, *crack*, and *manhole*, we computed the frequency of training instances frequency_i in the current fold. The class weight was then calculated as:

$$\text{weight}_i = \frac{\text{inv\_freq}_i}{\sum_j \text{inv\_freq}_j}, \quad \text{where} \quad \text{inv\_freq}_i = \frac{1}{\text{frequency}_i}.$$

This weighting strategy mitigates class imbalance while ensuring balanced performance evaluation across all damage categories. These weights were updated independently for each fold to account for variations in class distribution. This dynamic class weighting strategy was applied exclusively to YOLOv11 models, ensuring fair treatment of underrepresented classes during training. For the baseline architectures, including YOLOv8, SSD variants, and Faster R-CNN, class weighting was not applied, as we aimed to keep the baselines as standard as possible for fair comparison and reproducibility across frameworks.

Tables 4 and 5 summarize the dataset distribution across folds for the training and validation sets, respectively. For each fold, the tables report the total number of images, the breakdown by acquisition device (GoPro vs. smartphone), the number of annotated object instances per damage class (potholes, cracks, and manholes), and the average number of objects per image.

These summaries provide a clear overview of dataset composition and highlight the internal consistency maintained across folds. The device-wise counts ensure that images from different acquisition sources are reasonably balanced, while the per-class object counts illustrate the inherent class imbalance addressed through dynamic class weighting. Finally, the average objects per image confirm that the object density is relatively uniform across folds, supporting fair and reliable cross-validation evaluations.

Table 4. Training set summary per fold: images per device, object instances per class, and average objects per image.

Fold	Total Images	GoPro	Smartphone	Pothole	Crack	Manhole	Avg Objects/Image
1	1,506	1,435	71	936	1,918	704	2.36
2	1,507	1,428	79	954	1,883	697	2.35
3	1,507	1,428	79	960	1,889	734	2.38
4	1,507	1,430	77	933	1,867	736	2.35

Table 5. Validation set summary per fold: images per device, object instances per class, and average objects per image.

Fold	Total Images	GoPro	Smartphone	Pothole	Crack	Manhole	Avg Objects/Image
1	503	472	31	325	601	253	2.34
2	502	479	23	307	636	260	2.40
3	502	479	23	301	630	223	2.30
4	502	477	25	328	652	221	2.39

Experimental Configuration

Training Environment and Setup

Hyperparameter optimization experiments were conducted on a dual Tesla T4 GPU configuration (15GB RAM, Intel Xeon CPU @ 2.20GHz) using the Kaggle[37] platform, enabling efficient parallel processing of cross-validation folds. The optimization employed Weights & Biases[38] for systematic experiment tracking, conducting 120 total optimization trials (60 per YOLOv11 variant) to ensure comprehensive hyperparameter space exploration.

Baseline model validation utilized a Tesla P100 GPU configuration, providing adequate computational resources while maintaining environmental consistency across all experiments.

Training Protocol

All models utilized transfer learning initialization with pre-trained weights (COCO[39] for both YOLO variants and SSD backbones) to maximize dataset effectiveness through leveraging learned representations from large-scale datasets. Training employed default data augmentation strategies inherent to each framework, demonstrating dataset robustness across standard augmentation techniques without requiring specialized preprocessing.

Early stopping mechanisms were implemented based on validation performance: fitness score monitoring for YOLO variants (combining mAP@50-95, precision, recall, and mAP@50) and validation loss monitoring for other baseline models.

Consistent evaluation thresholds (IoU = 0.7, confidence = 0.25) were applied across all models to ensure methodological consistency and fair comparability. The IoU value of 0.7 matches the default used in YOLOv11 (Ultralytics) and provides a stricter evaluation that better captures localization accuracy, particularly for thin, elongated structures such as cracks, without disproportionately affecting other classes or model architectures.

Rigorous reproducibility protocols were implemented to ensure the reliability of the dataset validation. All training was conducted with fixed random seeds (seed = 0) and deterministic operations enabled across all frameworks, while identical cross-validation splits were maintained for every model architecture.

Experimental Results

Following the validation framework described previously, we present comprehensive experimental outcomes demonstrating the dataset's performance characteristics under systematic optimization and architectural comparison. This section presents the empirical results obtained from 480[1] individual training runs across 120 configurations for hyperparameter optimization, along with cross-architectural validation that reveals substantial performance variations across different detection paradigms. These results provide quantitative evidence of the dataset's quality, consistency, and practical utility for road damage detection tasks.

Optimal Performance Results

Table 6 presents the best validation performance achieved by each YOLOv11 model variant. The results were obtained by selecting the run with the highest mean precision across folds and reporting metrics averaged over all validation folds. Both models achieve mean precision values above 0.64, with recall exceeding 0.24 for YOLOv11n and 0.30 for YOLOv11s. The mAP@50 values reach 0.45 and above, while the F1-score[2] highlights a clear performance gap between the two variants.

In addition to mAP@50, we also report mAP@75 and mAP@50-95 to provide a more comprehensive evaluation under stricter IoU thresholds. The mAP@75 values (0.1719 for YOLOv11n and 0.1841 for YOLOv11s) confirm the superior localization capability of the larger model under more demanding overlap criteria. Similarly, the mAP@50-95 metric further highlights the consistent improvement of YOLOv11s over YOLOv11n across multiple IoU thresholds.

Overall, the larger YOLOv11s model consistently outperforms the YOLOv11n variant across all reported metrics, demonstrating improved balance between precision and recall.

Model	Parameters	Precision	Recall	mAP@50	mAP@75	mAP@50-95	F1-Score
YOLOv11n	2.6M	0.6497	0.2444	0.4538	0.171889	0.213361	0.2785
YOLOv11s	9.5M	**0.6539**	**0.3033**	**0.4659**	**0.184132**	**0.221682**	**0.3564**

Table 6. Best validation performance and parameters for YOLOv11 model variants. Bold values indicate the best performance for each metric.

YOLOv11n achieved its best performance with a batch size of 8, dropout of 0.113, initial and final learning rates of 0.007 and 0.026, respectively, momentum of 0.091, the RAdam optimizer, and a weight decay of 0.005, converging in approximately 40 epochs per fold. YOLOv11s likewise exhibited a strong optimal configuration, particularly with a batch size of 16, dropout

[1] Each model was evaluated over 60 independent runs per fold. So, 60 runs $\times$ 2 models $\times$ 4 folds = 480 total runs.

[2] The reported F1-score corresponds to the macro-F1, i.e., the mean F1 across all classes, which can be lower than the F1 computed from overall precision and recall.

of 0.255, initial and final learning rates of 0.008 and 0.07, momentum of 0.159, the RAdam optimizer, and a weight decay of 0.007.

Notably, the larger YOLOv11s model required a higher dropout rate (0.255 versus 0.113) and weight decay (0.007 versus 0.005), suggesting that stronger regularization is necessary to prevent overfitting given its increased capacity of 9.4M parameters compared to YOLOv11n's 2.6M. Furthermore, both models converged efficiently with the RAdam optimizer and relatively modest learning rate schedules, indicating that the dataset characteristics favor adaptive optimization strategies over more aggressive momentum-based approaches.

Figure 12 illustrates the training progression for the best-performing configurations, revealing consistent convergence behavior across different cross-validation folds with moderate variance. These metrics are derived from YOLO's internal training validation routine, which evaluates the model on the validation set at the end of each training epoch for monitoring and early stopping purposes. In contrast, all other results presented were obtained through YOLO's evaluation using the model's `val()` method on the same validation folds after training was completed. During this final evaluation, dropout is disabled, batch normalization layers are frozen, and no training-time data augmentations are applied. The observed performance differences between the two approaches can be attributed to the fact that YOLO's internal training validation retains certain training-time regularization mechanisms such as dropout and data augmentation in the pipeline, whereas the evaluation with `val()` mode disables these components, resulting in higher precision and overall performance metrics. The close agreement in performance trends between the two validation strategies confirms the robustness of the reported metrics and validates the reliability of YOLO's built-in training evaluation mechanisms for this particular dataset and task.

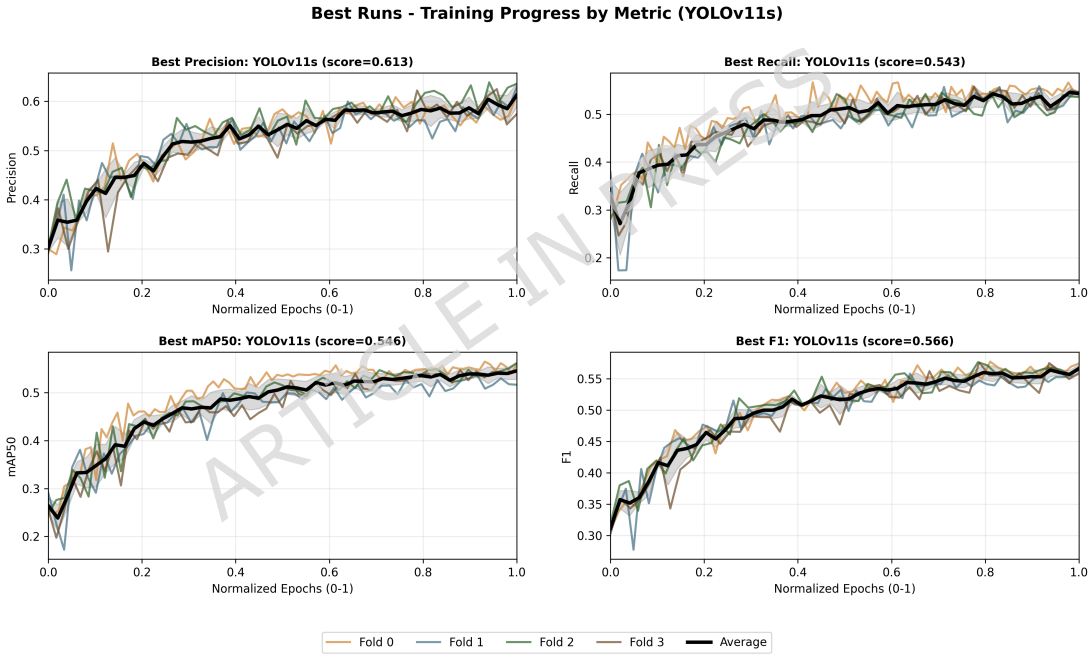

Figure 12. Training progression for the best-performing YOLOv11s configurations for each metric. Each subplot displays individual fold curves (colored lines), the average performance across folds (black line), and standard deviation bands (gray shading) for the precision, recall, mAP@50, and F1-score metrics. These curves reflect YOLO's internal validation during training for monitoring and early stopping, and are shown to illustrate the training progression rather than final evaluation performance. The consistent convergence patterns and relatively narrow variance bands across folds demonstrate the dataset's internal consistency and balanced class representation, enabling reliable model training regardless of the specific train-validation split.

While the dataset splits (folds) are engineered to maintain a consistent class distribution across training and validation subsets, this does not imply equal absolute counts of cracks, potholes, and manholes in the entire dataset. The reported recall values (0.24–0.30) reflect the inherent difficulty of detecting small-scale and partially occluded defects. The model achieves high precision by maintaining a conservative confidence threshold, which naturally limits recall. This trade-off is particularly evident for smaller objects like manholes, where visual ambiguity and occlusion reduce the model's ability to capture all instances despite the balanced training distribution.

Cross-Validation Consistency

The training curves reveal stable convergence with progressive improvement across epochs. YOLOv11s generally required fewer epochs to reach optimal performance (on average 35 epochs per fold) compared to YOLOv11n (average 40 epochs per fold), reflecting the dataset's appropriate difficulty level for modern detection architectures.

Figure 13 quantifies the average number of epochs required to reach convergence for each fold. The minimal inter-fold variation demonstrates consistent training behavior regardless of the specific data partition, validating the dataset's balanced composition. YOLOv11s consistently converged faster than YOLOv11n across all folds, suggesting that additional model capacity facilitates efficient learning from the dataset's visual patterns.

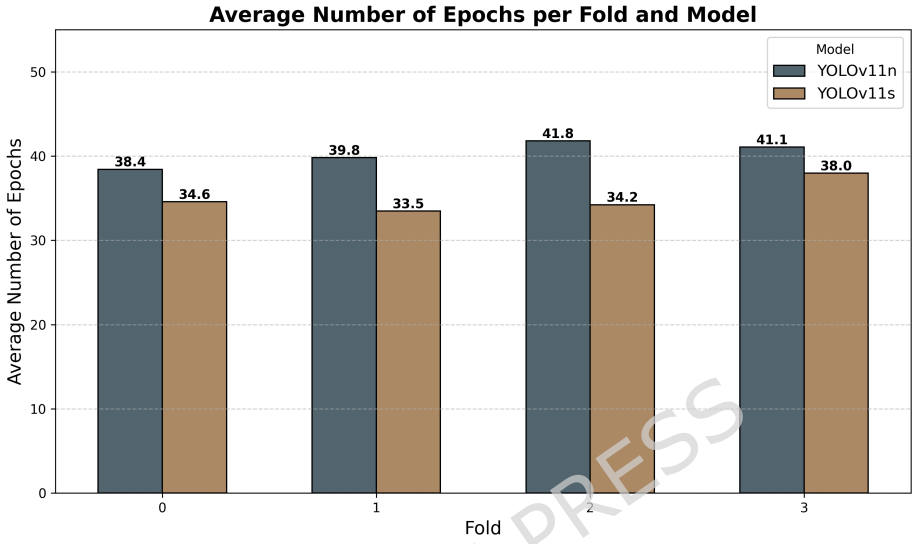

Figure 13. Average number of epochs to convergence per fold for YOLOv11n and YOLOv11s models. Values represent means across all optimization runs.

To evaluate the consistency of the dataset splits, we conducted a statistical analysis of the precision metric across cross-validation folds. The Kruskal–Wallis test did not reveal statistically significant differences among folds ($H = 5.1337$, $p = 0.1623$), indicating that model performance is comparable across different data partitions. Post-hoc Mann–Whitney U tests are reported in Table 7. Overall, the analysis indicates minimal variability in model performance across folds, suggesting that the dataset splits are well-balanced and representative of the underlying data distribution. This consistency supports the robustness of the evaluation protocol and ensures that model performance is not dependent on a specific training–validation split.

Table 7. Post-hoc Mann–Whitney U tests for cross-validation folds (precision metric).

Fold Comparison	U statistic	p-value
fold0 vs fold1	7,884.5	0.2034
fold0 vs fold2	6,750.0	0.4032
fold0 vs fold3	7,137.0	0.9075
fold1 vs fold2	5,995.5	0.0252
fold1 vs fold3	6,348.0	0.1133
fold2 vs fold3	7,540.5	0.5272

Table 8 reports the standard deviation of performance metrics across the four cross-validation folds, quantifying the stability of each model's performance across different data partitions. Lower standard deviation values indicate more consistent behavior across folds, suggesting robust generalization independent of the specific training-validation split. Detection-specialized models (FasterRCNN, SSD variants) exhibit notably lower variance across folds (F1 std < 0.015), indicating stable performance regardless of the specific train-validation split. In contrast, YOLO-based models show higher variance (F1 std 0.02–0.14), particularly YOLOv11s (F1 std = 0.1435), suggesting greater sensitivity to the composition of training data. This higher variability may reflect YOLO's optimization for speed over robustness, or could indicate that YOLO architectures benefit more from specific data characteristics present in certain folds (e.g., balanced class distribution, consistent lighting conditions).

Table 8. Standard Deviation of Performance Metrics Across Folds

Model	F1-score	mAP@50-95	mAP@50	Precision	Recall
FasterRCNN-ResNet50	0.0130	0.0088	0.0172	0.0140	0.0185
MobileNetv3-SSD320	0.0131	0.0017	0.0069	0.0383	0.0129
VGG16-SSD300	0.0140	0.0023	0.0046	0.0192	0.0113
YOLOv11n	0.0981	0.0335	0.0593	0.1621	0.0899
YOLOv11s	0.1435	0.0226	0.0268	0.1842	0.1121
YOLOv8n	0.0203	0.0053	0.0134	0.0345	0.0451

The two-stage detectors (FasterRCNN-ResNet50, MobileNetv3-SSD320, and VGG16-SSD300) exhibit remarkably low standard deviations for mAP@50-95 and mAP@50, indicating highly consistent performance across folds. Notably, MobileNetv3-SSD320 demonstrates exceptional stability with the lowest mAP@50-95 standard deviation (0.0017), though this comes at the cost of higher precision variability (0.0383). The YOLO-family models show moderately higher variability, particularly in recall metrics, where YOLOv8n and YOLOv11s reach standard deviations above 0.04. This suggests that while YOLO architectures achieve superior absolute performance (as shown in Figure 16), their sensitivity to the specific training data partition is slightly elevated compared to traditional two-stage detectors. YOLOv11s exhibits the highest overall variability across metrics (F1: 0.0271, mAP@50: 0.0255), which may reflect its larger model capacity, adapting more strongly to fold-specific characteristics. Despite these differences, all models maintain standard deviations below 0.05 for most metrics, confirming that the observed performance differences in Figure 16 reflect genuine architectural capabilities rather than artifacts of data splitting.

Figures 14 and 15 present the Precision-Recall (PR) curves for YOLOv11n and YOLOv11s across all four cross-validation folds, providing a detailed view of the trade-off between precision and recall at varying confidence thresholds. These curves complement the aggregate metrics in Table 8 by visualizing the complete performance spectrum rather than single operating points. Note that these PR curves are generated directly from YOLO's internal training validation pipeline and may exhibit slight numerical differences compared to the standalone Ultralytics validation `val()` mode, though the overall performance trends remain consistent.

The PR curves reveal several noteworthy patterns. First, both models demonstrate high consistency in curve topology across folds, with minimal deviation in the precision-recall trade-off profiles. This visual consistency corroborates the quantitative stability metrics in Table 8. Second, class-specific performance hierarchies remain stable: manhole detection consistently achieves the highest precision across all recall levels, followed by pothole detection, while crack detection presents the most challenging scenario for both architectures. Third, YOLOv11s exhibits notably steeper curves in the high-recall region, indicating superior ability to maintain precision when maximizing detection coverage, a critical characteristic for real-world deployment where missing damaged areas could compromise road safety assessments.

Architectural Performance Trends

The YOLO family demonstrates substantial variation in precision-recall characteristics across evaluation metrics (Figure 16, left), with YOLOv11 variants achieving notably superior precision despite lower recall at the default confidence threshold of 0.25. YOLOv11s achieves the highest precision at 0.654 with recall of 0.303 (mAP@50: 0.466, mAP@50-95: 0.222), while YOLOv11n demonstrates comparable precision of 0.650 with recall of 0.244 (mAP@50: 0.454, mAP@50-95: 0.213). In contrast, YOLOv8n exhibits lower precision at 0.556 but higher recall at 0.490 (mAP@50: 0.530, mAP@50-95: 0.258), reflecting a more permissive detection threshold that accepts greater false positive rates to achieve broader damage coverage.

For road damage detection applications, the high precision of YOLOv11 variants represents a critical advantage over recall-optimized alternatives. In automated infrastructure monitoring systems, false positives generate unnecessary maintenance dispatches, wasting resources and undermining system credibility, while genuine damage instances can be captured through systematic coverage over time or by adjusting detection thresholds. Crucially, precision-optimized models like YOLOv11 offer operational flexibility: lowering the confidence threshold from the default 0.25 can increase recall while maintaining acceptable precision due to the model's inherent discriminative capability. Conversely, models with lower baseline precision such as YOLOv8n cannot recover high precision through threshold adjustment, raising the confidence threshold to reduce false positives necessarily decreases recall further, resulting in models that are simultaneously imprecise and incomplete. The YOLOv11 variants' 1.17× precision advantage over YOLOv8n (0.650–0.654 vs. 0.556) thus provides a more robust foundation for deployment, where detection sensitivity can be tuned post-training to match operational requirements without compromising classification accuracy.

The F1-score, which equally weights precision and recall, shows YOLOv8n achieving 0.520 compared to YOLOv11s at 0.356 and YOLOv11n at 0.279. However, this metric obscures the practical asymmetry between precision and recall costs in real-world deployment. In road maintenance contexts, the ability to confidently identify damage when detected (high

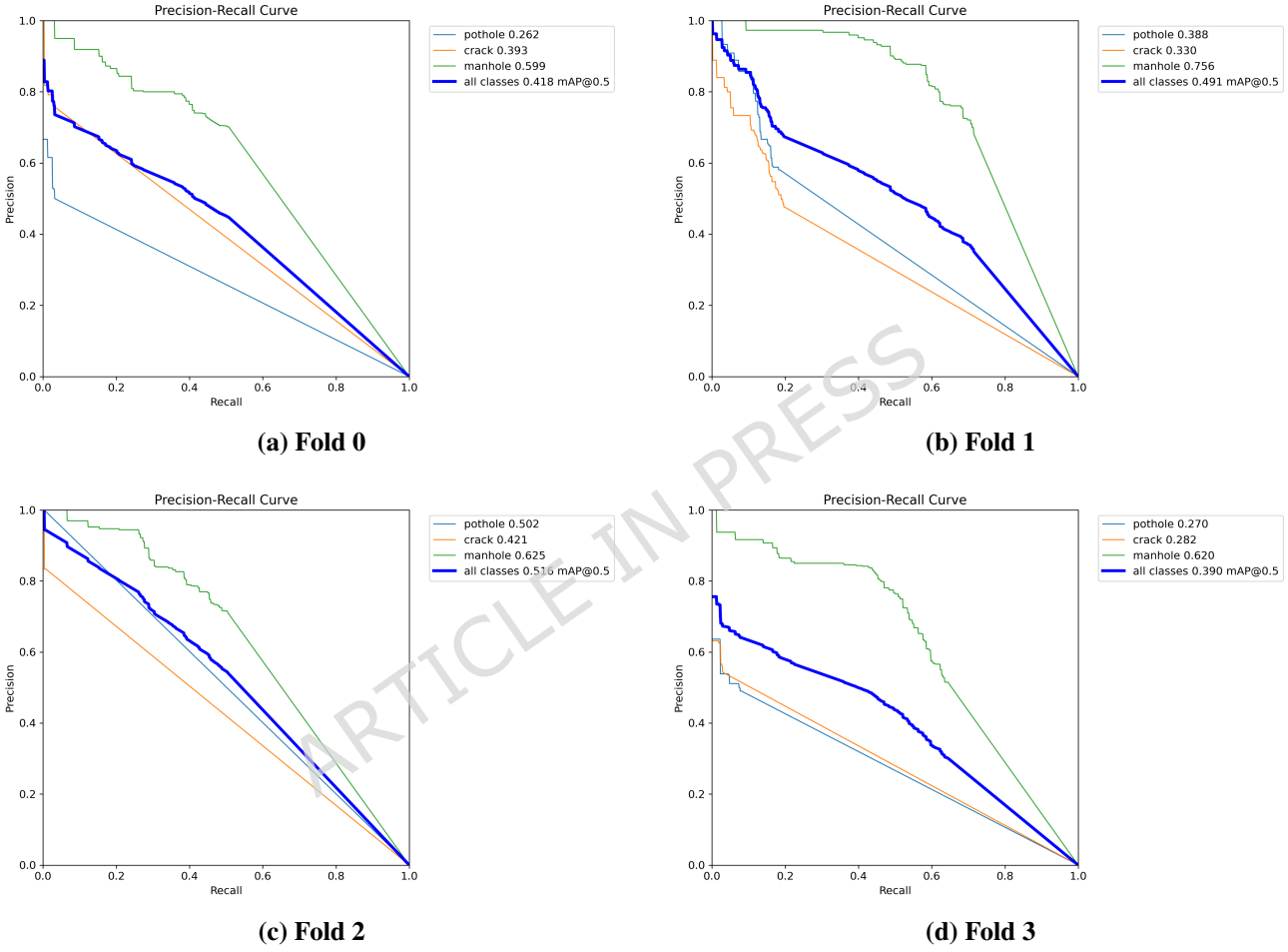

Figure 14. Precision-Recall curves for YOLOv11n across all four cross-validation folds. Each subplot shows class-specific performance (pothole, crack, manhole) and the overall mean average precision. The consistency of curve shapes across folds validates the balanced nature of data partitioning.

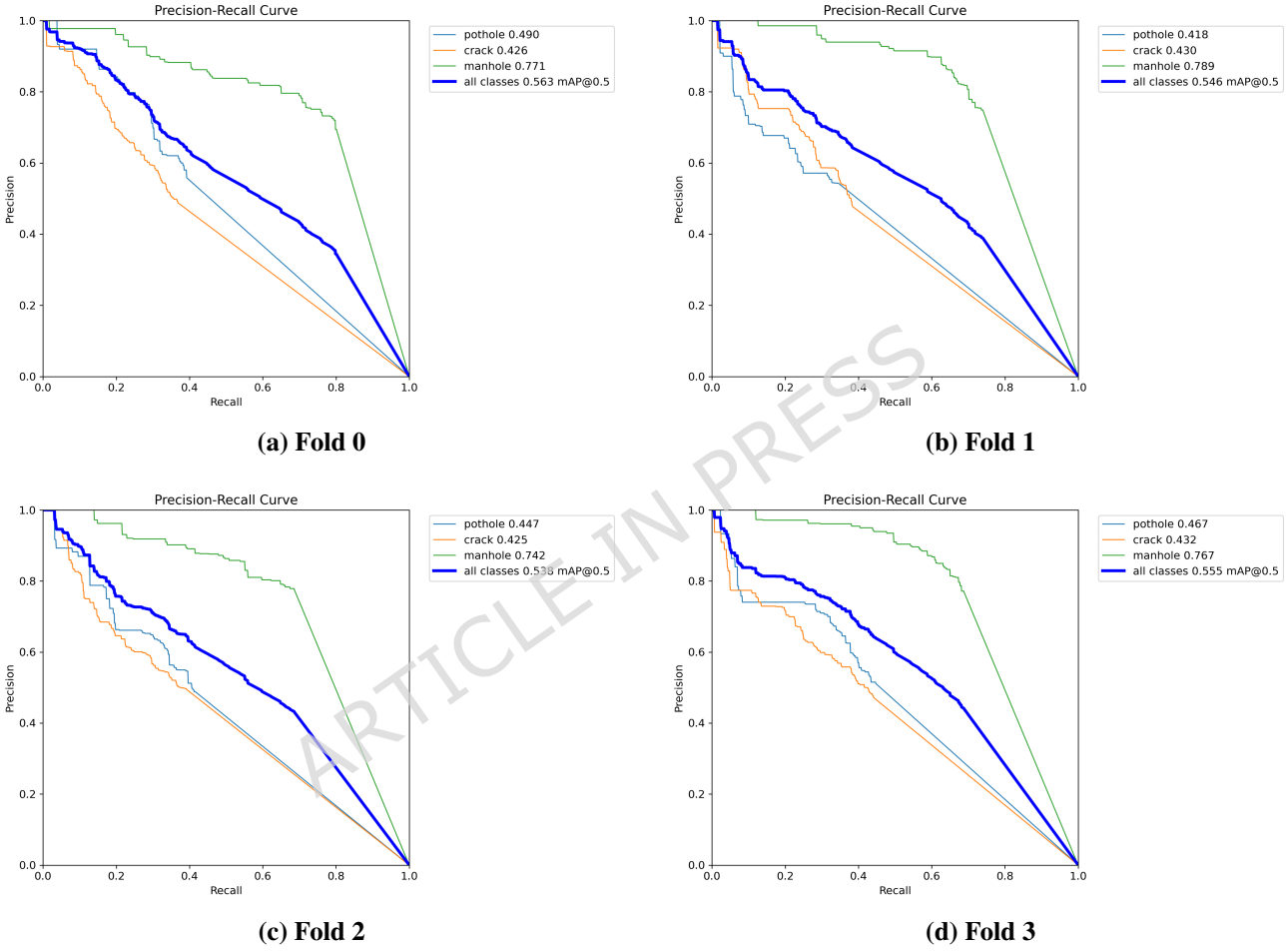

Figure 15. Precision-Recall curves for YOLOv11s across all four cross-validation folds. The larger model capacity of YOLOv11s is reflected in consistently higher precision values at equivalent recall levels compared to YOLOv11n (Figure 14), particularly for the challenging crack detection class.

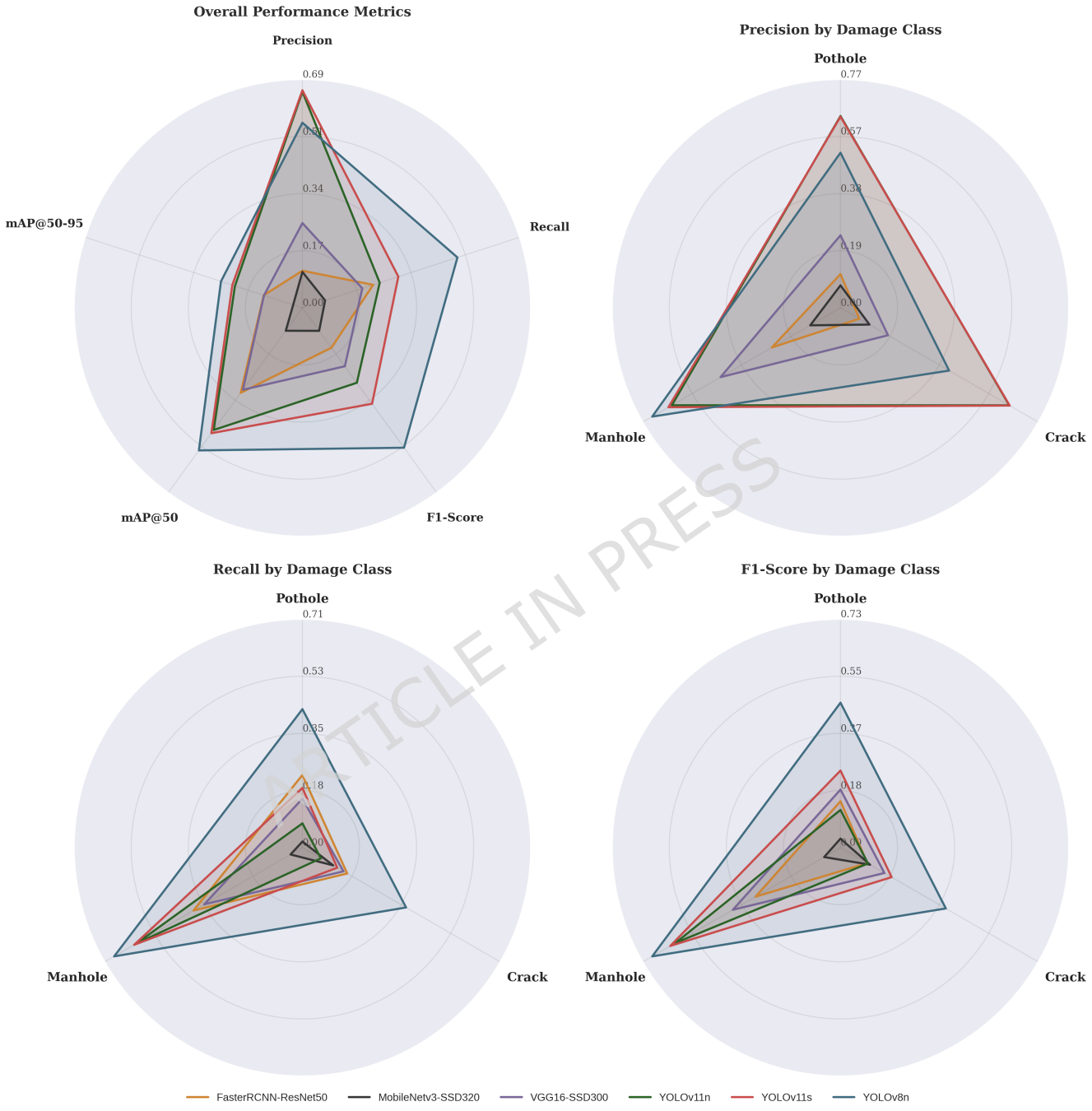

Figure 16. Performance comparison across detection models. Left: Overall performance metrics including precision, recall, F1-score, mAP@50, and mAP@50-95. Right: Class-specific precision for pothole, crack, and manhole detection. All metrics are averaged across a 4-fold cross-validation.

precision) enables immediate repair prioritization and resource allocation, whereas lower recall at conservative thresholds can be compensated through threshold relaxation based on tolerance for false alarms. The YOLOv11 architecture's precision-first optimization aligns with these operational priorities, providing a deployable baseline that can be adapted to varying sensitivity requirements.

Traditional detection architectures exhibit significantly lower performance across all metrics. VGG16-SSD300 achieves the best results among non-YOLO models (precision: 0.255, recall: 0.189, F1: 0.217, mAP@50: 0.305), demonstrating approximately 2.6× lower precision and 1.5× lower mAP@50 compared to YOLOv11s. MobileNetV3-SSD320 demonstrates the weakest performance (precision: 0.108, recall: 0.072, F1: 0.086, mAP@50: 0.085), achieving less than one-sixth the precision of YOLOv11 variants, confirming that mobile-optimized, lightweight architectures require substantial architectural modifications or extensive optimization for this dataset. Notably, Faster R-CNN with ResNet50 backbone, despite its two-stage detection mechanism and deeper feature extraction capability, underperforms relative to YOLO variants (precision: 0.112, recall: 0.224, F1: 0.148, mAP@50: 0.316), exhibiting the highest recall among non-YOLO architectures but substantially lower precision—approximately 5.8× lower than YOLOv11 models. This precision-recall trade-off suggests that the two-stage region proposal mechanism generates numerous candidate regions but struggles with accurate classification, resulting in a high false positive rate that undermines practical deployment in automated road inspection systems where classification confidence is paramount.

The reduced performance of SSD-based architectures (both VGG16-SSD300 and MobileNetV3-SSD320) and Faster R-CNN, all configured with input resolutions between 300×300 and 320×320 pixels, likely stems from the aggressive image downsampling process necessitated by their higher computational complexity. These traditional architectures employ deeper or more parameter-heavy backbones (VGG16 with 138M parameters, ResNet50 with 25.6M parameters) that require lower input resolutions to maintain acceptable inference efficiency and memory footprint. In contrast, YOLO variants leverage lightweight architectures (YOLOv11n: 2.6M parameters, YOLOv11s: 9.4M parameters, YOLOv8n: 3.2M parameters) that can operate efficiently at 640×640 resolution without computational penalty, enabling them to preserve spatial detail critical for damage detection. Resizing images to these lower resolutions, compared to the YOLO variants' standard 640×640 input dimensions, significantly compromises the models' ability to extract discriminative features characteristic of road damage, particularly for subtle defects such as fine cracks or small potholes, where spatial detail preservation is crucial for accurate detection and classification. The per-class precision analysis (Figure 16, right) further corroborates this trend. For the crack class, YOLOv11 models maintain exceptional precision at 0.653–0.655, while traditional architectures drop to 0.184 (VGG16-SSD300) and 0.073 (Faster R-CNN), highlighting their inability to effectively discriminate fine-grained damage patterns. This 3.6× to 9.0× precision advantage for YOLOv11 demonstrates that when these models detect cracks, they do so with substantially higher confidence and accuracy than traditional architectures. The substantial performance gap between modern and traditional detectors underscores the critical importance of both architectural design principles and appropriate input resolution choices for complex visual inspection tasks in transportation infrastructure monitoring.

The reduced performance of SSD-based architectures (both VGG16-SSD300 and MobileNetV3-SSD320) and Faster R-CNN, all configured with input resolutions between 300×300 and 320×320 pixels, likely stems from the aggressive image downsampling process necessitated by their higher computational complexity. These traditional architectures employ deeper or more parameter-heavy backbones (VGG16 with 138M parameters, ResNet50 with 25.6M parameters) that require lower input resolutions to maintain acceptable inference efficiency and memory footprint. In contrast, YOLO variants leverage lightweight architectures (YOLOv11n: 2.6M parameters, YOLOv11s: 9.4M parameters, YOLOv8n: 3.2M parameters) that can operate efficiently at 640×640 resolution without computational penalty, enabling them to preserve spatial detail critical for damage detection. Resizing images to 300–320 pixel resolutions significantly compromises the models' ability to extract discriminative features characteristic of road damage, particularly for subtle defects such as fine cracks or small potholes, where spatial detail preservation is crucial for accurate detection and classification. This resolution-complexity trade-off represents a fundamental architectural limitation: traditional detectors cannot simultaneously achieve high-resolution input and real-time inference without specialized hardware acceleration, whereas YOLO's efficient design enables both. The per-class precision analysis (Figure 16, right) further corroborates this trend. For the crack class, YOLOv11 models maintain exceptional precision at 0.653–0.655, while traditional architectures drop to 0.184 (VGG16-SSD300) and 0.073 (Faster R-CNN), highlighting their inability to effectively discriminate fine-grained damage patterns even with sophisticated detection mechanisms. This 3.6× to 9.0× precision advantage for YOLOv11 demonstrates that when these models detect cracks, they do so with substantially higher confidence and accuracy than traditional architectures. The substantial performance gap between modern and traditional detectors underscores the critical importance of both architectural efficiency, enabling higher resolution processing, and modern design principles for complex visual inspection tasks in transportation infrastructure monitoring where subtle damage features demand detailed spatial information.

Class-Level Performance Analysis

The four-panel radar chart (Figure 16) provides a comprehensive view of class-specific performance across precision, recall, and F1-score metrics, revealing systematic patterns in how different architectures handle the three damage categories. Manhole detection consistently achieves the highest precision across all YOLO models, with YOLOv8n reaching 0.729, YOLOv11s 0.666, and YOLOv11n 0.653, while traditional architectures show significantly lower performance (VGG16-SSD: 0.464, Faster R-CNN: 0.266, MobileNetV3-SSD: 0.117). This superior performance likely stems from the distinctive regular geometry, well-defined boundaries, and higher visual contrast of manholes compared to other damage types, making them more readily distinguishable even for baseline architectures. The consistently high precision across YOLO variants for manhole detection validates the dataset's annotation quality and confirms that modern architectures effectively capture these infrastructure features with minimal false positive rates.

The recall patterns (Figure 16, bottom-left) reveal expected trade-offs from precision optimization. Manhole detection maintains the highest recall across YOLO models (YOLOv8n: 0.673, YOLOv11s: 0.601, YOLOv11n: 0.592), confirming that these visually distinctive structures are consistently detected across architectures. For potholes and cracks, YOLOv11 variants demonstrate lower recall at the default 0.25 confidence threshold: pothole recall ranges from 0.075 (YOLOv11n) to 0.185 (YOLOv11s), while crack recall spans 0.066–0.124. However, this conservative detection behavior reflects design choices favoring classification confidence over coverage. Unlike models with inherently low precision, YOLOv11's high discriminative capability enables recall improvement through confidence threshold reduction without substantial precision degradation. YOLOv8n achieves higher recall for potholes (0.428) and cracks (0.370) but at the cost of increased false positives, a trade-off that cannot be reversed post-deployment. Traditional architectures show critically low recall across all classes, with MobileNetV3-SSD achieving only 0.019 recall for potholes, effectively failing to detect this damage type regardless of threshold settings.

The F1-score analysis (Figure 16, bottom-right) reflects the precision-recall trade-off at the fixed 0.25 confidence threshold used for evaluation. YOLOv8n achieves higher F1-scores across all damage types (manhole: 0.699, pothole: 0.466, crack: 0.391) compared to YOLOv11s (manhole: 0.632, pothole: 0.248, crack: 0.190) and YOLOv11n (manhole: 0.612, pothole: 0.121, crack: 0.102), reflecting its more balanced precision-recall profile at this specific operating point. However, F1-score optimization assumes equal cost for false positives and false negatives, an assumption that does not hold in road inspection contexts where false alarms incur immediate operational costs while missed detections can be addressed through repeated inspection or threshold adjustment. The lower F1-scores of YOLOv11 variants at the default threshold mask their operational advantage: the ability to trade recall for precision is reversible through threshold tuning in precision-optimized models, but cannot be recovered in recall-optimized architectures. Traditional architectures exhibit severely degraded F1-scores across all classes, with crack detection showing particularly poor performance (VGG16-SSD: 0.163, MobileNetV3-SSD: 0.110, FasterRCNN-ResNet50: 0.100). The radar plot geometric patterns, with YOLOv8n forming a larger triangle while YOLOv11 variants show compressed shapes biased toward high precision, visually confirm the architectural divergence between balanced detection and precision-optimized strategies. For practical road damage detection where classification accuracy drives maintenance decisions, YOLOv11's precision-first approach provides a more robust deployment foundation with tunable sensitivity to match operational requirements.

Discussion

Dataset Validation Insights

The comprehensive experimental validation demonstrates several key characteristics of the road damage dataset, establishing its quality and practical utility. First, the dataset supports models achieving precision above 0.65, recall above 0.49, and mAP@0.5 above 0.50 with appropriate optimization, demonstrating sufficient annotation quality and visual diversity to enable robust detection performance. Statistical analysis confirmed no significant performance differences across cross-validation folds ($p = 0.1623$), validating the dataset's internal balance and absence of partition-dependent biases, which ensures consistent model behavior regardless of data partitioning.

The predominantly negative training-validation loss gaps observed across all configurations indicate that the dataset promotes strong generalization rather than memorization, a critical characteristic for real-world deployment scenarios. Furthermore, significant performance differences across hyperparameter configurations ($p < 0.001$ for optimizer selection) confirm that the dataset responds appropriately to systematic optimization, enabling practitioners to improve performance through careful tuning. Both lightweight (YOLOv11n) and enhanced (YOLOv11s) model variants achieved strong performance with architecture-specific optimal configurations, demonstrating the dataset's versatility and suitability for diverse deployment scenarios ranging from resource-constrained edge devices to more capable embedded systems.

Additionally, models consistently converged within 35–40 epochs per fold, indicating that the dataset provides clear learning signals without requiring prohibitively long training durations. This training efficiency is particularly valuable for practitioners seeking to rapidly prototype and deploy road damage detection systems. Collectively, these validation results establish the

road damage dataset as a high-quality resource for training and evaluating pothole, crack, and manhole detection systems. The dataset's demonstrated capacity to support diverse model architectures, optimization strategies, and deployment scenarios positions it as a valuable contribution to the road infrastructure monitoring community.

Limitations and Future Work

While the current dataset provides a solid foundation for road damage detection research and applications, several limitations present opportunities for future enhancement. The dataset's geographic scope is currently limited to specific regions, which may affect model generalization to road conditions in different climates, construction standards, or maintenance practices. However, even if this fact could be considered a limitation, there is a contrasting aspect: a dataset collected for solving a specific problem could be more effective than a wider one, whose generalization could yield lower-quality results[40]. However, future iterations could incorporate data from diverse geographic locations and varying weather conditions to improve cross-regional transferability and robustness to environmental variations.

From a methodological perspective, future work could explore temporal analysis by collecting longitudinal data that tracks damage progression over time, enabling studies of deterioration patterns and predictive maintenance modeling. The integration of additional sensor modalities, such as depth information from LiDAR or thermal imaging, can complement the visual data and enhance detection accuracy under challenging lighting conditions. Furthermore, developing hierarchical annotation schemes that capture damage severity levels would enhance the dataset's utility for prioritizing maintenance activities and resource allocation.

Finally, given the dataset's focus on deployment in resource-constrained embedded systems, future validation efforts could explore emerging lightweight architectures specifically optimized for edge computing, such as next-generation YOLO variants, EfficientDet models, YOLO-CNN, and DenseNet[23], and neural architecture search-derived detectors that maintain the computational efficiency requirements of real-time road monitoring on devices like Raspberry Pi. Multi-task learning approaches that simultaneously detect damage, estimate severity, or predict maintenance urgency could be investigated, provided they remain compatible with embedded deployment constraints. Establishing benchmark challenges and leaderboards for this dataset, with explicit performance metrics for both accuracy and computational efficiency, would foster community engagement and drive continued improvements in practical road-damage detection methodologies. These enhancements would collectively strengthen the dataset's role as a comprehensive resource for advancing automated road infrastructure monitoring and maintenance planning in real-world, resource-constrained deployment scenarios.

Data Availability

The datasets generated and analysed during the current study are available in the Road Damage Dataset - Potholes, Cracks, and Manholes repositories on Zenodo[41] and on Kaggle at https://www.kaggle.com/datasets/lorenzoarcioni/road-damage-dataset-potholes-cracks-and-manholes. Example code for loading and using the dataset is available at https://github.com/lorenzo-arcioni/Road-Damage-Detection-Dataset-Analysis.

The dataset is released under Creative Commons Attribution 4.0 International license and is freely accessible without registration requirements.

Code Availability

The Python code used for the curation, analysis, training, and validation of the dataset is publicly available on GitHub, ensuring reproducibility and enabling further methodological and applied research. The repository is accessible at: https://github.com/lorenzo-arcioni/Road-Damage-Detection-Dataset-Analysis.

Author contributions statement

L.A., E.G., and M.R.M. conceived the methodology, L.A. and E.G. conducted the data collection and performed the baseline experiments, G.L.F. supervised the work and M.G.-M. wrote the original draft of the manuscript. All authors validated and formally analyzed the dataset and reviewed the final manuscript.

Competing Interests

The authors declare that they have no competing interests or personal relationships that could have influenced the work reported in this paper.

Funding

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan (PNRR) funded by the European Union – Next Generation EU, Mission 4, CUP G23C24000790006 (2024-25). This paper was partially supported by University of Udine project on “Piano Stretegico Dipartimentale on Artificial Intelligence” (PSD-AI) (2022-25) project at the University of Udine. This research was partially supported by "Ayudas para estancias de movilidad en el extranjero José Castillejo para jóvenes doctores" from Ministerio de Ciencia, Innovación y Universidades of Spain. Moreover, it was supported by the ASTOUND project (101071191 HORIZON-EIC-2021-PATHFINDERCHALLENGES-01) funded by the European Commission. In addition, the Spanish Ministry of Science and Innovation, through the projects BeWord, GOMINOLA, TRUSTBOOST (PID2021-126061OB-C43, PID2020-118112RB-C21 and PID2020-118112RB-C22, PID2023-150584OB-C21 and PID2023-150584OB-C22, funded by MCIN/AEI/10.13039/501100011033, and by the European Union "NextGenerationEU/PRTR").

Acknowledgements

Special thanks are extended to Kaggle for offering accessible GPU resources.

Additional information

The dataset used for training YOLO models is available at https://www.kaggle.com/datasets/lorenzoarcioni/pothole-test. The image annotation tool is available at https://github.com/bnsreenu/digitalsreeni-image-annotator.

References

1. Arya, D. et al. Global road damage detection: State-of-the-art solutions. In 2020 IEEE International Conference on Big Data (Big Data), 5533–5539, DOI: 10.1109/BigData50022.2020.9377790 (2020).
2. Cui, L., Qi, Z., Chen, Z., Meng, F. & Shi, Y. Pavement distress detection using random decision forests. In Zhang, C. et al. (eds.) Data Science, 95–102 (Springer International Publishing, Cham, 2015).
3. Stricker, R., Eisenbach, M., Sesselmann, M., Debes, K. & Gross, H.-M. Improving visual road condition assessment by extensive experiments on the extended gaps dataset. In 2019 International Joint Conference on Neural Networks (IJCNN), 1–8, DOI: 10.1109/IJCNN.2019.8852257 (2019).
4. Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D. & Sekimoto, Y. Rdd2022: A multi-national image dataset for automatic road damage detection (2022). 2209.08538.
5. Maniat, M., Camp, C. V. & Kashani, A. R. Deep learning-based visual crack detection using google street view images. Neural Comput. Appl. **33**, 14565–14582, DOI: 10.1007/s00521-021-06098-0 (2021).
6. Lei, X., Liu, C., Li, L. & Wang, G. Automated pavement distress detection and deterioration analysis using street view map. IEEE Access **8**, 76163–76172, DOI: 10.1109/ACCESS.2020.2989028 (2020).
7. Ren, M., Zhang, X., Zhi, X., Wei, Y. & Feng, Z. An annotated street view image dataset for automated road damage detection. Sci. Data **11**, 407, DOI: 10.1038/s41597-024-03263-7 (2024).
8. Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D. & Sekimoto, Y. Rdd2020: An annotated image dataset for automatic road damage detection using deep learning. Data Brief **36**, 107133, DOI: https://doi.org/10.1016/j.dib.2021.107133 (2021).
9. Kortmann, F. et al. Detecting various road damage types in global countries utilizing faster r-cnn. In 2020 IEEE International Conference on Big Data (Big Data), 5563–5571, DOI: 10.1109/BigData50022.2020.9378245 (2020).
10. Vishwakarma, R. & Vennelakanti, R. Cnn model & tuning for global road damage detection. In 2020 IEEE International Conference on Big Data (Big Data), 5609–5615, DOI: 10.1109/BigData50022.2020.9377902 (2020).
11. Pham, V., Pham, C. & Dang, T. Road damage detection and classification with detectron2 and faster r-cnn. In 2020 IEEE International Conference on Big Data (Big Data), 5592–5601, DOI: 10.1109/BigData50022.2020.9378027 (2020).
12. Lin, C. et al. Da-rdd: Toward domain adaptive road damage detection across different countries. IEEE Transactions on Intell. Transp. Syst. **24**, 3091–3103, DOI: 10.1109/TITS.2022.3221067 (2023).

13. Kapp, A., Hoffmann, E., Weigmann, E. & Mihaljević, H. Streetsurfacevis: a dataset of crowdsourced street-level imagery annotated by road surface type and quality. Sci. Data **12**, 92, DOI: 10.1038/s41597-024-04295-9 (2025).

14. Yin, T., Zhang, W., Kou, J. & Liu, N. Promoting automatic detection of road damage: A high-resolution dataset, a new approach, and a new evaluation criterion. IEEE Transactions on Autom. Sci. Eng. **22**, 2472–2484, DOI: 10.1109/TASE.2024.3379945 (2025).

15. Yang, H. et al. A large-scale image repository for automated pavement distress analysis and degradation trend prediction. Sci. Data **12**, 1426, DOI: 10.1038/s41597-025-05748-5 (2025).

16. Zhang, H. et al. A new road damage detection baseline with attention learning. Appl. Sci. **12**, DOI: 10.3390/app12157594 (2022).

17. Pham, V., Nguyen, D. & Donan, C. Road damage detection and classification with yolov7. In 2022 IEEE International Conference on Big Data (Big Data), 6416–6423, DOI: 10.1109/BigData55660.2022.10020856 (2022).

18. Alfarrarjeh, A., Trivedi, D., Kim, S. H. & Shahabi, C. A deep learning approach for road damage detection from smartphone images. In 2018 IEEE International Conference on Big Data (Big Data), 5201–5204, DOI: 10.1109/BigData.2018.8621899 (2018).

19. Arya, D. et al. Deep learning-based road damage detection and classification for multiple countries. Autom. Constr. **132**, 103935, DOI: https://doi.org/10.1016/j.autcon.2021.103935 (2021).

20. Guo, G. & Zhang, Z. Road damage detection algorithm for improved yolov5. Sci. Reports **12**, 15523, DOI: 10.1038/s41598-022-19674-8 (2022).

21. Pang, Z. et al. Road surface classification with texture-feature-embedded resnet for the active suspension systems in complex environments. Adv. Eng. Informatics **71**, 104280, DOI: https://doi.org/10.1016/j.aei.2025.104280 (2026).

22. Liu, Y. et al. A non-destructive automatic pavement damage detection scheme based on end-to-end neural networks with multi-level attention mechanism. Eng. Appl. Artif. Intell. **156**, 111246, DOI: 10.1016/j.engappai.2025.111246 (2025).

23. Yenni, H. et al. Mycd: Integration of yolo-cnn and densenet for real-time road damage detection based on field images. J. Appl. Data Sci. **7**, 384–395, DOI: 10.47738/jads.v7i1.1040 (2025).

24. Arcioni, L. & Giordani, E. Road damage dataset collection route map (2025). https://www.google.com/maps/d/viewer?mid=1WrrMPBqnh6v_GnQmfvKJWaq3R0YPD78&usp=sharing.

25. Bhattiprolu, S. Digitalsreeni image annotator (2024). https://github.com/bnsreenu/digitalsreeni-image-annotator.

26. Ravi, N. et al. Sam 2: Segment anything in images and videos (2024). 2408.00714.

27. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You only look once: Unified, real-time object detection (2016). 1506.02640.

28. Jocher, G. & Qiu, J. Ultralytics yolo11 (2024).

29. Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C., Bottou, L. & Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 25 (Curran Associates, Inc., 2012).

30. Jocher, G., Chaurasia, A. & Qiu, J. Ultralytics yolov8 (2023).

31. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks (2019). 1801.04381.

32. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition (2015). 1409.1556.

33. Liu, W. et al. SSD: Single Shot MultiBox Detector, 21–37 (Springer International Publishing, 2016).

34. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks (2016). 1506.01497.

35. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition (2015). 1512.03385.

36. Lin, T.-Y. et al. Feature pyramid networks for object detection (2017). 1612.03144.

37. Kaggle: Your Machine Learning and Data Science Community — kaggle.com. https://www.kaggle.com/.

38. Biewald, L. Experiment tracking with weights and biases (2020). Software available from wandb.com.

39. Lin, T. et al. Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014). 1405.0312.

40. Khosravian, A., Amirkhani, A., Kashiani, H. & Masih-Tehrani, M. Generalizing state-of-the-art object detectors for autonomous vehicles in unseen environments. Expert. Syst. with Appl. **183**, 115417, DOI: https://doi.org/10.1016/j.eswa.2021.115417 (2021).

41. Giordani, E., Arcioni, L., Gil-Martín, M. & Marini, M. R. Road damage dataset: Potholes, cracks and manholes, DOI: 10.5281/zenodo.17834373 (2025).