4th International Civil Engineering & Architecture Conference
17-19 May 2025, Trabzon, Türkiye

Automated structural crack detection using a deep learning-based YOLO algorithm and robotic dog

Ayşecan Bostan*[1], Kemal Hacıefendioğlu[1]

[1] *Karadeniz Technical University, Department of Civil Engineering, 61080 Trabzon, Türkiye*

Abstract. This study aims to automatically detect cracks in structural elements using the YOLO object detection algorithm, based on Deep Learning techniques, integrated with robotic dog technology. This approach seeks to enhance both the accuracy and efficiency of structural health monitoring processes. The study begins with the selection of appropriate datasets via the Roboflow platform, where a dataset consisting of 2,600 annotated crack images was utilized for model training. Training was conducted in the Google Colab environment, resulting in the development of an object detection model. Additionally, a segmentation model was created using the same dataset to ensure more precise crack boundary delineation. The performance of the trained models was evaluated through real-time tests using images captured by cameras mounted on the Unitree Go2 Edu Plus robotic dog. These tests were conducted on a reinforced concrete frame with induced cracks at the Structural Health Monitoring Laboratory of Karadeniz Technical University. The images were processed in real-time using YOLO models, enabling immediate crack detection. The findings demonstrate that the robotic system, combined with the YOLO algorithm, effectively detects cracks in real-time with high accuracy. This study highlights the potential of integrating Deep Learning and robotic technologies for structural health monitoring, offering a reliable and efficient solution for damage assessment in civil engineering applications. The results validate the proposed approach as a practical and dependable method for automated structural crack monitoring.

Keywords: Crack detection; Robotic dog; YOLO models; Deep Learning; Object detection

1. Introduction

Earthquakes are natural disasters that cause extensive physical destruction and significant loss of life on a large scale. In the aftermath of such catastrophic events, the rapid and reliable assessment of the structural integrity of buildings is of critical importance. Traditional damage detection methods primarily rely on manual inspections, which are often time-consuming, labor-intensive, and susceptible to human error. These limitations underline the necessity of developing innovative and automated approaches that can deliver fast and accurate evaluations of structural conditions (Qayoumi et al., 2025; Jia & Ye, 2023).

Recent advancements in robotics and artificial intelligence (AI) offer promising opportunities to revolutionize post-disaster structural assessments (Zhai et al., 2024; Ye et al., 2022). In this context, the primary aim of this study is to develop an integrated system that utilizes robotic technologies and deep learning-based methods for the rapid and automated detection of structural damages. The project involves the integration of the Unitree Go2 robotic dog platform with the YOLOv8 deep learning model to enable real-time detection of cracks, deformations, and other structural damages on building surfaces.

The robotic dog is designed to move safely and efficiently even in challenging terrain conditions. It is equipped with advanced sensors and reinforced with artificial intelligence algorithms, allowing it to autonomously explore and inspect areas that are difficult or dangerous for human access. Through the use of a deep learning model capable of processing environmental images, the system can accurately identify and classify structural damages without the need for human intervention (Morfidis et al., 2023; Ilmak et al., 2024).

Another significant feature of the developed system is its ability to process data in real time, providing operators with instant information and comprehensive reporting during inspections. This capability ensures that critical decisions can be made rapidly, which is particularly vital in the immediate aftermath of earthquakes when timely interventions can save lives and prevent further structural failures.

By combining robotic mobility, advanced sensing technologies, and AI-driven damage detection, this study aims to offer a fast, reliable, and efficient solution for post-earthquake structural assessments. The proposed system

*Corresponding author, E-mail: aysecanbostan61@gmail.com

https://doi.org/10.31462/icearc2025_ce_sme_419

represents a significant step forward in enhancing the effectiveness of disaster response efforts and minimizing the risks associated with manual inspection methods.

2. Materials and methods

The following section provides a comprehensive overview of the key components utilized in the study, including the robotic dog platform, the YOLO algorithm, the dataset employed for training and testing, and the testing model. These elements are essential for understanding the methodology and the process of damage detection that forms the basis of this research.

2.1. Unitree Go2 Robotic Dog

The Unitree Go2 is a next-generation robotic dog model developed by the Chinese robotics company, Unitree Robotics. It features a quadrupedal structure resembling that of a cat or dog. The Go2 builds upon the experience gained from previous models such as the A1, Go1, and AlienGo. Designed for both industrial and individual use cases, it offers advanced performance, agility, and enhanced artificial intelligence (AI) capabilities.

One of the most prominent features of the Go2 is its advanced mobility. Thanks to its quadrupedal design, it is capable of maintaining balance even in rugged terrains. The robot can easily navigate difficult environments, including stairs and obstacles, making it versatile in various scenarios. Compared to earlier models, the Go2's maximum speed has been increased, and its acceleration has been significantly improved, enhancing its overall performance.

In terms of artificial intelligence and sensing capabilities, the Go2 is equipped with a range of sensors, including depth cameras and LiDAR, enabling it to perceive its surroundings in three dimensions. The robot can perform functions such as object recognition, obstacle detection, and path planning with high accuracy. The combination of a powerful processor and AI algorithms further boosts the robot's ability to process data in real-time, enhancing its responsiveness to environmental changes.

Additionally, the durability and portability of the Go2 are noteworthy. Constructed with lightweight yet robust materials, its body is resistant to impacts and falls, ensuring longevity even in challenging environments. The robot’s rechargeable battery allows for extended periods of continuous operation, and different battery options may be available depending on the application. Furthermore, its foldable or easily portable design makes it convenient to transport from one location to another, offering practicality for various use cases. The Unitree Go2 robot dog and its equipment are shown in Fig. 1.



Fig. 1. Unitree Go2 robotic dog and equipment

2.2.YOLO (You Only Look Once) algorithm

YOLO (You Only Look Once) is a computer vision model introduced by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi at the CVPR conference in 2016, where it gained significant attention. Winning OpenCV's "People's Choice" awards, YOLO is a single-stage detector that performs object detection and classification through a single network pass (Nelson, 2025).

Among single-stage detection models, YOLO remains one of the most well-known and widely used frameworks. However, it is not the only example of such models; for instance, MobileNetSSDv2 is another popular single-stage detector. Nonetheless, YOLO often demonstrates superior performance compared to other models in terms of both speed and accuracy.

YOLO models approach the object detection task through a single-shot regression method that predicts the bounding boxes of objects directly. This approach provides high processing speed and a compact network structure. These characteristics make YOLO not only an efficient tool for accelerating the training process but also highly suitable for deployment on edge devices with limited computational resources.

An example illustrating the working principle of YOLO is presented in Fig. 2 below. The visual representation clearly demonstrates the process of detecting and classifying objects by identifying their bounding boxes.

Thanks to these advantages, YOLO has become a highly preferred model in a wide variety of practical applications. Its ability to balance accuracy and speed has led to its widespread adoption across both academic research and industry sectors. Furthermore, the model's efficiency in resource-constrained environments makes it a practical choice for real-world deployments where computational power may be limited.

YOLO models appeal to a wide range of developers due to their ability to be trained on a single GPU. Machine learning practitioners can deploy these models at a low cost, either on edge devices or in cloud environments. The speed offered by YOLO is particularly advantageous in scenarios that require real-time object detection.

Before the emergence of YOLO, R-CNNs were commonly used for object detection; however, their slow processing speeds made them unsuitable for real-time applications. In contrast, YOLO greatly facilitates rapid inference tasks in applications such as vehicle detection, animal recognition, and security surveillance, where fast and accurate detection is critical.

YOLOv8 represents a state-of-the-art advancement in computer vision, developed by Ultralytics, the organization behind the widely adopted YOLOv5 model. Designed to address a variety of tasks—including object detection, image classification, and instance segmentation—YOLOv8 combines cutting-edge performance with developer accessibility through an intuitive command-line interface (CLI) and a well-structured python package. This dual interface approach significantly lowers the barrier to entry for both novice and experienced practitioners, promoting widespread adoption across research and industry.

A major innovation distinguishing YOLOv8 from its predecessors is its anchor-free architecture. Unlike traditional models that rely on predefined anchor boxes to estimate object locations, YOLOv8 directly predicts object centers. This design shift results in enhanced localization accuracy, reduced computational complexity, and improved inference speed, making the model particularly well-suited for real-time applications on both high-performance and resource-constrained devices.

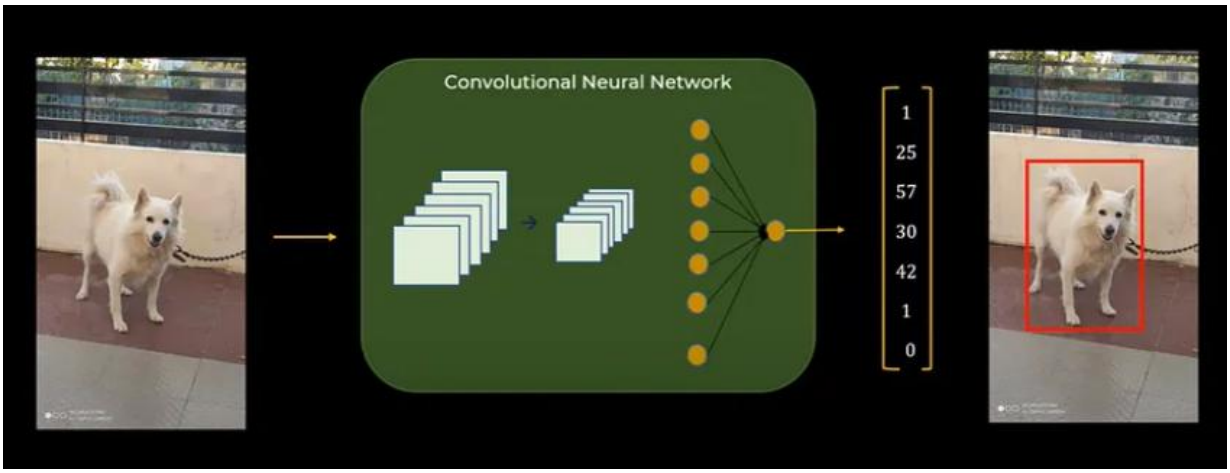


Fig. 2. General working mechanism of YOLO (Kumar, 2022)

Table 1. YOLOv8 detection models comparison (Ultralytics, 2025)

Model	Size (pixels)	mAPval (50-95)	Speed CPU (ms)	Speed T4 GPU (ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

In this study, the YOLOv8n model, whose features are presented in Table 1, has been selected. YOLOv8n operates with a 640x640 pixel input size and achieves a mAP (50-95) value of 37.3%. The model has only 3.2 million parameters (M) and 8.7 billion FLOPs (floating point operations). Due to its lightweight structure and low computational cost, it is particularly suitable for resource-constrained systems.

2.3. Characteristics of the dataset used

The dataset used in this study was obtained from the Roboflow platform. Roboflow is a comprehensive platform that provides all the necessary tools for creating, developing, and deploying computer vision models. The platform offers flexible usage by enabling integration at any step of the pipeline through APIs and SDKs. Additionally, it provides end-to-end interface support to automate the image processing workflow from start to finish.

Roboflow Annotate is an online annotation tool specifically designed for labeling images in tasks such as object detection, classification, and segmentation. This tool facilitates the accurate and rapid annotation of images, streamlining the dataset preparation process. These features significantly enhance the quality of the dataset used in this study, making a substantial contribution to the computer vision model development process. A visual taken from the Roboflow homepage is presented in Fig. 3.

The dataset used in this study consists of a total of 3,255 crack images. All these images have been carefully labed to precisely indicate the locations of the cracks. The dataset is divided into three groups for model training and evaluation: "training," "validation," and "test."

- Training Group: The largest group used for model training, consisting of 2,658 annotated images.
- Validation Group: This group, containing 300 annotated images, is used to evaluate the accuracy of the model during the training process.
- Test Group: This group, consisting of 297 annotated images, is reserved to independently assess the model's performance.

The meticulous preparation of this dataset significantly contributes to achieving high accuracy and reliability during the model training and evaluation processes. The labeling process ensured that the data was processed in a clear and consistent manner, positively impacting the model's performance.

Example excerpts from the image and label files in the dataset are presented in Fig. 4 and 5 below. These excerpts provide a clearer visual understanding of the dataset structure and the annotation format used.

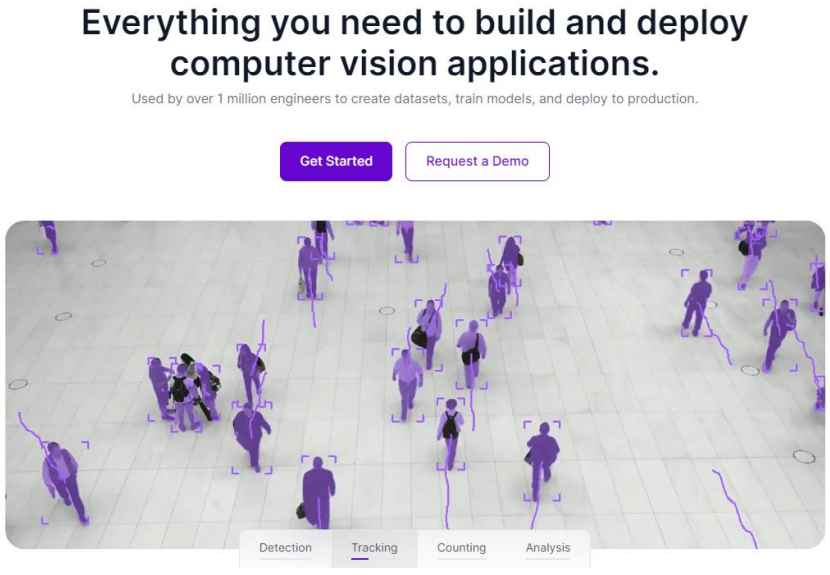


Fig. 3. Roboflow homepage (Roboflow, 2025)

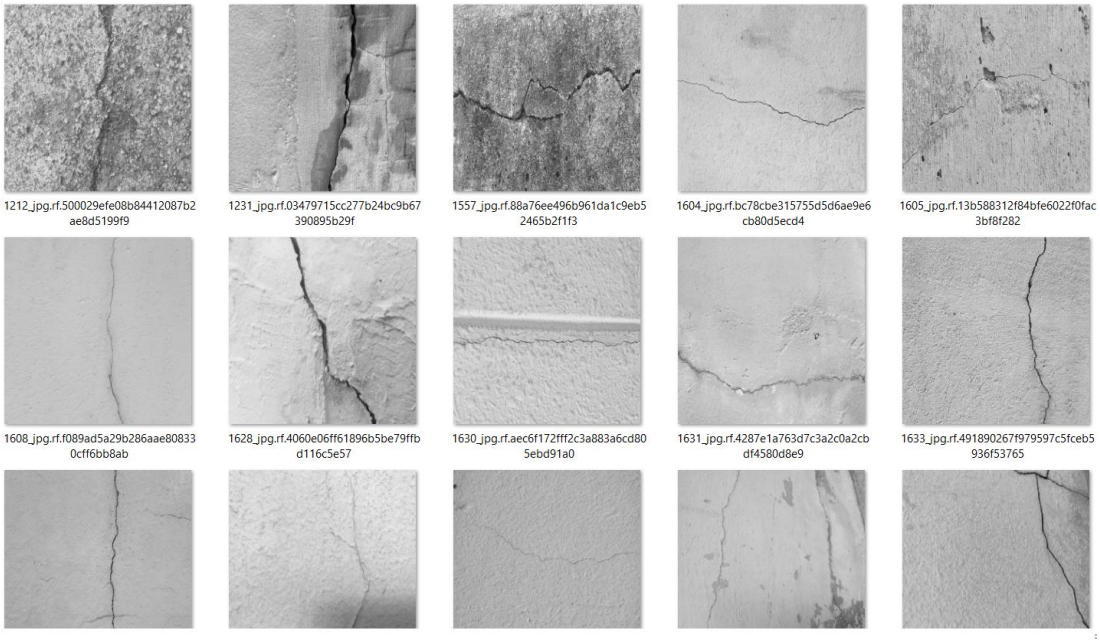


Fig. 4. Sample images from the dataset

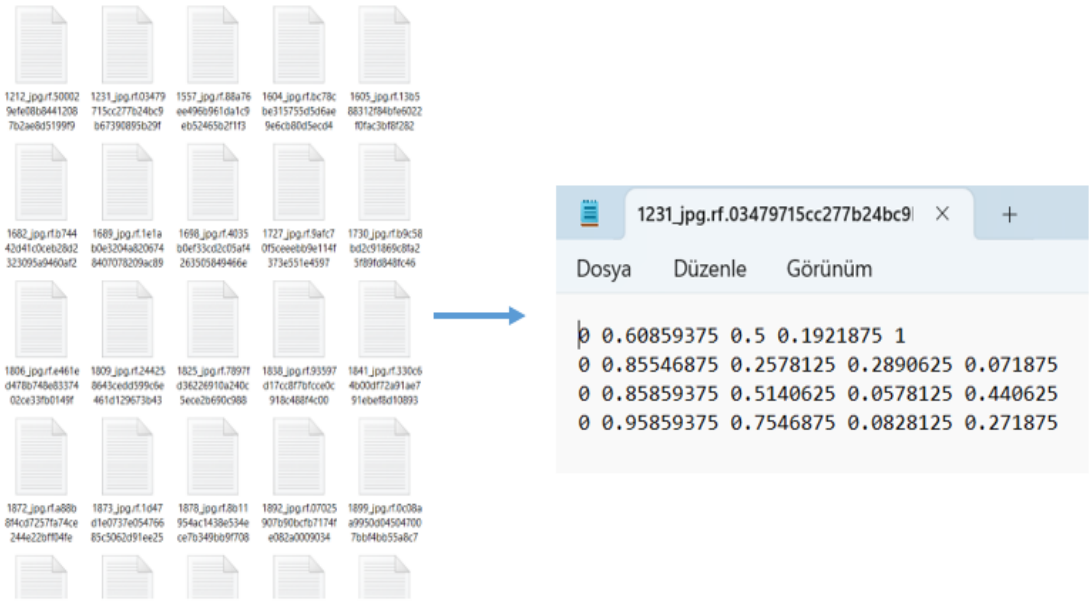


Fig. 5. Label files

The text file presented in Figure 5 is an annotation file in YOLO format, used to provide training data for an object detection model. Such files contain annotations that are essential for enabling the model to acquire object detection capabilities. Each line in the file represents an object and provides the following information: the class ID, the center coordinates of the bounding box, and the width and height of the box. All these values are normalized with respect to the dimensions of the corresponding image. This format helps the model learn the locations and sizes of objects within the images. Once the training process is completed, the model can detect and correctly classify similar objects in new data. The YOLO annotation structure plays a critical role in developing fast and efficient object detection models.

The object detection results for the first two lines provide detailed information about the identified objects, including their class, position, and bounding box dimensions in normalized coordinates. In Line 1, the object is assigned a Class ID of 0, with the center of its bounding box located at 60.86% of the image width and 50% of the image height. The bounding box spans 19.22% of the image width and covers the full height of the image, indicating a vertically elongated object occupying a significant portion of the image's vertical axis. In Line 2,

another object with the same Class ID (0) is detected, with its bounding box center positioned at 85.55% of the image width and 25.78% of the image height. The corresponding bounding box dimensions are 28.91% of the image width and 7.19% of the image height, suggesting a more compact and horizontally wider object. Lines 3 and 4 follow the same format, representing additional detected objects using normalized center coordinates and relative dimensions.

2.4. Training process and code block used

The code block required to initiate the training process was obtained from the official page of the "Ultralytics YOLOv8" application and is presented in Fig. 6. Various modifications were made to the code to align it with the specific requirements of the project, and subsequently, the training process was initiated. The training was configured to run for a total of 100 epochs and took approximately 2.293 hours to complete. Throughout the training phase, the model parameters were optimized with the aim of achieving high accuracy. This stage played a critical role in enhancing the overall performance of the model and ensuring successful results in real-world applications.

The command shown in Fig. 7 is a terminal or Python code line used to train a YOLOv8 model on a dataset. Below is an explanation of each part of the command:

- !yolo detect train: Used to train the YOLOv8 model for an object detection task. This command indicates that the model will be run in training mode.
- data=/content/drive/MyDrive/object_detection2/config.yaml: The path to the configuration file of the dataset to be used for training. This file includes the dataset path, number of classes, and other relevant settings.
- model=yolov8n.pt: The pre-trained weight file of the YOLOv8 model to be used for training. (yolov8n.pt refers to the "nano" version of YOLOv8, which is lightweight and fast.)
- epochs=100: Indicates that the model will be trained for 100 epochs. An epoch means the entire training dataset is passed through the model once.
- imgsz=640: The resolution to which the images will be resized during training. Here, all images are resized to 640x640 pixels.
- workers=8: The number of CPU cores to be used for data loading. More workers can speed up the data loading process.
- batch=8: The number of images to be processed in each iteration during training. Larger batch sizes require more memory.
- device=0: Specifies which device will be used for the training process. (0: training will be performed on GPU. If training is to be performed on CPU, this should be set as device=cpu.)
- name=yolov8_crack_detection2: The name under which the trained model file will be saved. This will also be the name of the folder where training output files are stored.



Fig. 6. Sample code block (Ultralytics, 2025)

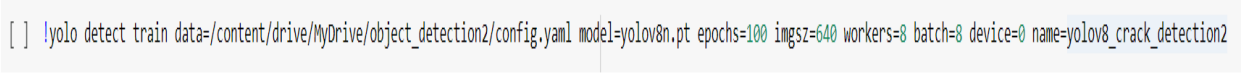


Fig. 7. Education code tailored to requirements

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    98/100      1.18G     0.9092     0.7805      1.153          2        640: 100% 333/333 [01:11<00:00,  4.63it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 19/19 [00:05<00:00,  3.38it/s]
                  all        300        367      0.839      0.826      0.857      0.611

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    99/100      1.18G     0.9002     0.7743      1.151          2        640: 100% 333/333 [01:18<00:00,  4.24it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 19/19 [00:03<00:00,  5.52it/s]
                  all        300        367      0.825      0.842      0.861      0.607

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    100/100     1.17G      0.904     0.7676      1.156          2        640: 100% 333/333 [01:13<00:00,  4.52it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 19/19 [00:03<00:00,  5.47it/s]
                  all        300        367      0.843      0.822       0.86      0.606

100 epochs completed in 2.293 hours.
Optimizer stripped from runs/detect/yolov8_crack_detection2/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/yolov8_crack_detection2/weights/best.pt, 6.3MB

Validating runs/detect/yolov8_crack_detection2/weights/best.pt...
Ultralytics YOLOv8.2.93 🚀 Python-3.10.12 torch-2.4.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 19/19 [00:09<00:00,  2.09it/s]
                  all        300        367      0.846      0.826      0.867      0.613
Speed: 1.1ms preprocess, 4.3ms inference, 0.0ms loss, 6.1ms postprocess per image
Results saved to runs/detect/yolov8_crack_detection2
```

Fig. 8. Training

As illustrated in Fig. 8, the training process was successfully completed, and the resulting data were analyzed to assess the model's performance in terms of both accuracy and speed. The YOLOv8 model was trained over a total of 100 epochs, with the entire training phase taking approximately 2.293 hours. Thanks to the carefully prepared dataset and the selection of appropriate hyperparameters, the model was able to learn efficiently and achieve a high level of generalization. Upon completion of the training, two optimized weight files were generated: *last.pt*, representing the model's weights at the end of the training, and *best.pt*, corresponding to the model's best performance during training. Both files were designed to be lightweight, each with a file size of just 6.3 MB, making them suitable for real-time applications.

The model's performance was rigorously evaluated using a validation dataset. It achieved a precision of 84.6%, indicating its ability to correctly classify detected objects, and a recall of 82.6%, reflecting its effectiveness in identifying all relevant objects. The mean Average Precision at a 50% Intersection over Union (mAP@50) was recorded at 86.7%, demonstrating the model's strong object detection capability. Furthermore, the model achieved a mAP@50-95 score of 61.3%, which assesses overall accuracy across a broader range of IoU thresholds, confirming its robustness in more complex evaluation settings.

In addition to accuracy, the model also performed efficiently in terms of speed. During validation, the average preprocessing time per image was measured at 1.1 milliseconds, the inference time at 4.3 milliseconds, and the post-processing time at 6.1 milliseconds. These results highlight the model's suitability for real-time detection tasks where both high accuracy and low latency are critical.

These results show that the YOLOv8 model can perform object detection tasks with high accuracy and sensitivity. A mAP@50 value of 86.7% proves that the model achieved a strong learning capacity during training and could effectively detect objects in the dataset. Additionally, the lightweight optimized weight files and fast processing times indicate that the model is also suitable for deployment in environments with limited computational resources.

2.5. Training and validation results

The figure shown in Fig. 9 illustrates the losses and performance metrics obtained during the training process of the YOLOv8 model. Each of these graphs is used to understand how the model progresses throughout the training and how its performance changes during validation. A detailed interpretation of the graphs is presented below:

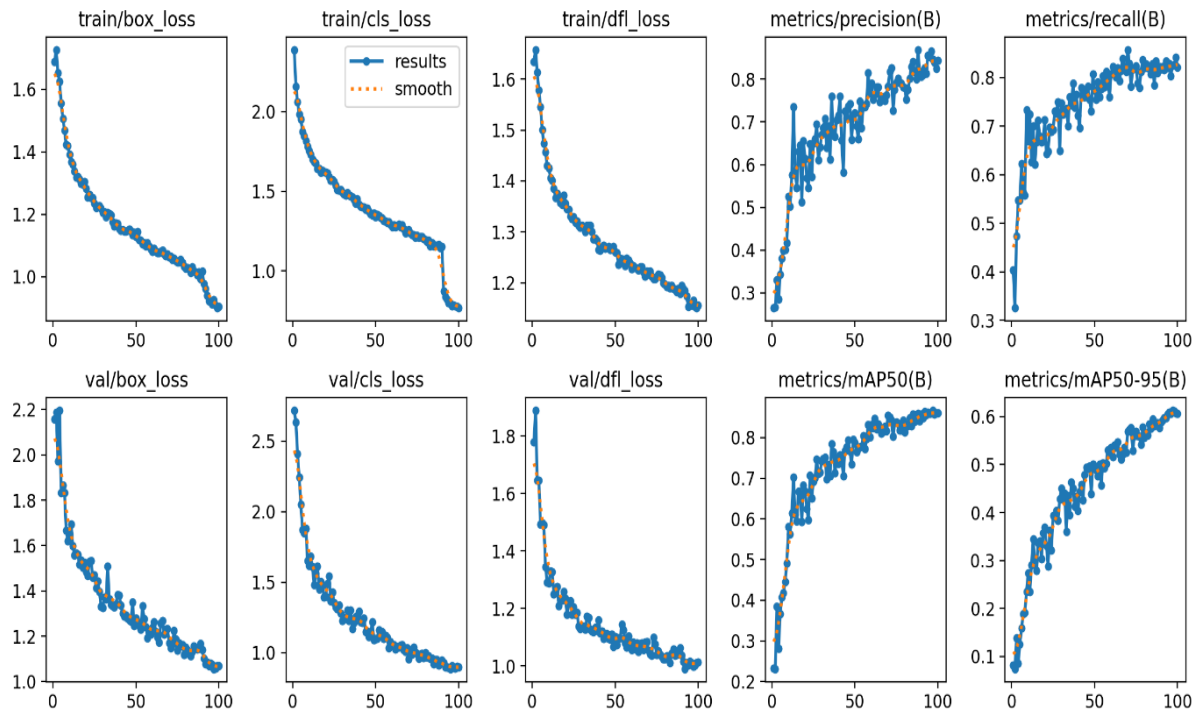


Fig. 9. Results

The evaluation of the training process included a detailed analysis of loss functions and performance metrics, which provided insights into the model's learning behavior and accuracy. The training losses, including box loss (*train/box_loss*), classification loss (*train/cls_loss*), and distribution focal loss (*train/dfl_loss*), consistently decreased over time. The reduction in box loss indicates that the model increasingly improved its accuracy in predicting bounding box locations. Similarly, the decrease in classification loss reflects enhanced accuracy in object classification, while the declining distribution focal loss shows progress in the precise localization of bounding boxes. Correspondingly, the validation losses—*val/box loss*, *val/cls loss*, and *val/dfl loss*—also demonstrated a parallel downward trend. This consistency between training and validation losses suggests that the model achieved generalizable performance without significant overfitting.

In terms of performance metrics, the model showed continuous improvement across all key indicators. Precision steadily increased with each epoch, eventually reaching approximately 85%, indicating a high rate of correctly classified detected objects. Recall also showed a gradual upward trend, achieving around 82%, which confirms the model's ability to detect most objects present in the dataset. The mean Average Precision at an IoU threshold of 50% (*mAP@50*) reached about 86%, reflecting strong object detection accuracy. Moreover, the more comprehensive *mAP@50-95* metric, which evaluates performance across multiple IoU thresholds, achieved a value of approximately 61%. This result highlights the model's robustness and effectiveness in varying detection scenarios, confirming its suitability for practical applications in real-world environments.

Throughout the training process, a consistent reduction in both training and validation losses was observed, indicating that the model successfully learned the underlying distributions of the training data. The continuous improvement in performance metrics (precision, recall, and mAP) suggests that the model progressively became better at making predictions. Moreover, the parallel decrease in training and validation losses confirms that the model has strong generalization capabilities and that overfitting did not occur.

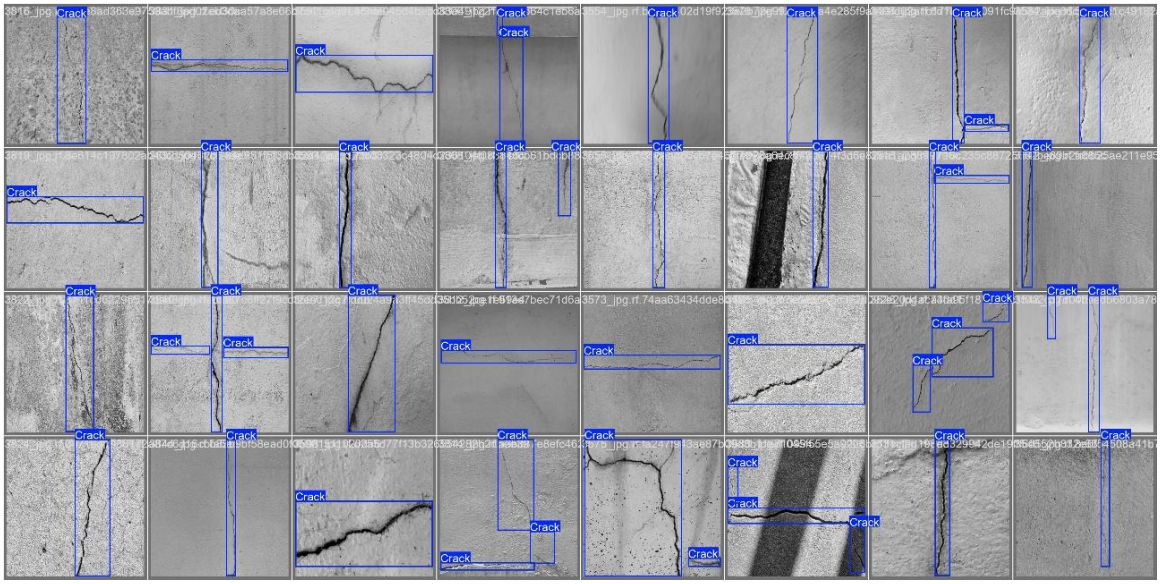


Fig. 10. Validation data set

The image presented in Fig. 10 shows the crack detection results performed by the model on the validation dataset. In the image, each detected crack is labeled with blue bounding boxes, indicating that the model has accurately identified each crack area. Such images are highly useful for evaluating how well the model has performed during training and for assessing its accuracy in object detection tasks.

Each blue bounding box represents a crack that the model has correctly detected. Observing that the objects in the image are correctly classified and the bounding boxes are properly placed demonstrates that the model operates with high accuracy and successfully performs the object detection task. Furthermore, the presence of cracks of different sizes and at various locations within the image reflects the model's ability to effectively detect various types of cracks. These kinds of outputs confirm that the model has processed the examples encountered during training correctly and has developed a strong generalization capability.

3. Testing process and analysis results

In this study, a YOLOv8 crack detection model was developed, and a comprehensive testing process was conducted to evaluate the model's accuracy. To assess the model's applicability, a successful connection was established between the robot and the YOLOv8 model, allowing the monitoring of the model's performance through this integration. The testing environment was set at the Structural Health Monitoring Laboratory of Karadeniz Technical University, where real-world measurements were conducted to assess the model's success under practical conditions. The use of a reinforced concrete steel frame structure for testing was deemed an appropriate choice to achieve highly accurate results in crack detection tasks.

The technical specifications of the tested reinforced concrete model in Fig. 11. were carefully defined. The concrete strength class was set at C25/30, a standard that ensures sufficient strength and durability of the material. The steel reinforcement strength was selected as B420C, guaranteeing high structural resilience. The physical dimensions of the reinforced concrete model were also an important criterion: the model measured 550 × 1700 × 300 mm (width × length × height) with a total height of 1850 mm (Çolakoğlu et al., 2024). These dimensions provided sufficient structural elements and an adequate testing area, enabling the development of a model suitable for crack detection studies.

This well-structured model increased the diversity of the data used during the training phase and allowed for broader testing of the model's capabilities.Tests conducted on the structure aimed to determine the effectiveness of the YOLOv8 model in crack detection. During these tests, the model's accuracy was compared against real-time data, and the results showed that the model performed with high accuracy not only in the training environment but also in real-world applications. This process demonstrated that the YOLOv8 model is a reliable and effective tool for crack detection in reinforced concrete structures. Moreover, integrating the model with robot technology proved to be an important innovation in the field of structural health monitoring.

The measurement process was meticulously carried out, and all stages were carefully monitored. The activated robot dog, as shown in Fig. 12, was remotely controlled via a remote control, enabling it to reach every point within the testing area. The camera mounted on the robot recorded real-time images and transmitted them to the computer screen, allowing for a detailed inspection of the entire reinforced concrete model through these images. Each surface of the model was observed via the camera footage, and potential cracks were carefully identified.

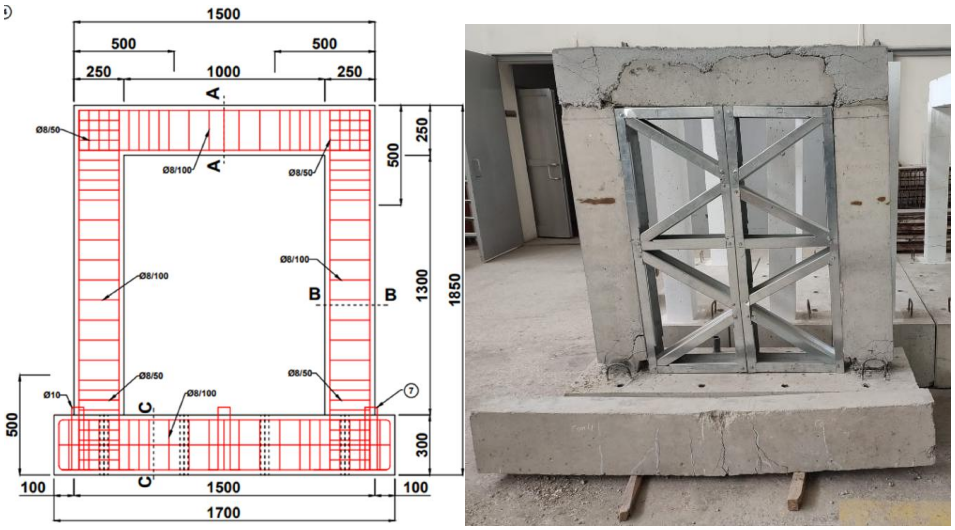


Fig. 11. Test model



Fig. 12. Images of the measurement process

A portion of the Python code that establishes the connection between the YOLOv8 model and the Unitree Go2 robot dog, enabling real-time analysis of the images captured by the robot's onboard camera, is presented in Fig. 13. This code processes the images obtained from the robot's camera, allowing the model to accurately detect cracks and simultaneously evaluate these detections with the model.

The Python script ensures that the images captured by the camera are converted into a format suitable for the model, allows the YOLOv8 model to make predictions on these images, and displays the results in real time on the screen. In this way, the system provides instant crack detection results to the user.

Additionally, the code is designed to operate in synchronization with the robot dog's movements. The images captured from different angles are accurately analyzed, and classification results are generated for each frame. This structure plays a crucial role in both testing the model's accuracy and enhancing the robot dog's effectiveness in field applications.

```python
import imutils
import cv2
import time
import random
import numpy as np
import sys
from ultralytics import YOLO
from unitree_sdk2py.core.channel import ChannelFactoryInitialize
from unitree_sdk2py.go2.video.video_client import VideoClient

# Predefined variables
confidence_score = 0.5
text_color_b = (0, 0, 0)  # Black
text_color_w = (255, 255, 255)  # White
background_color = (0, 255, 0)
font = cv2.FONT_HERSHEY_SIMPLEX

# Load YOLO model
model = YOLO("/home/kemalheo/Desktop/YOLO/models/yolov8n_büyük_veri_setli.pt")  # Model path
labels = model.names
colors = [[random.randint(0, 255) for _ in range(3)] for _ in labels]

# Initialize Unitree Go2 video client
if len(sys.argv) > 1:
    ChannelFactoryInitialize(0, sys.argv[1])
else:
    ChannelFactoryInitialize(0)

client = VideoClient()  # Create a video client
client.SetTimeout(3.0)
client.Init()
```

Fig. 13. A portion of the Python code

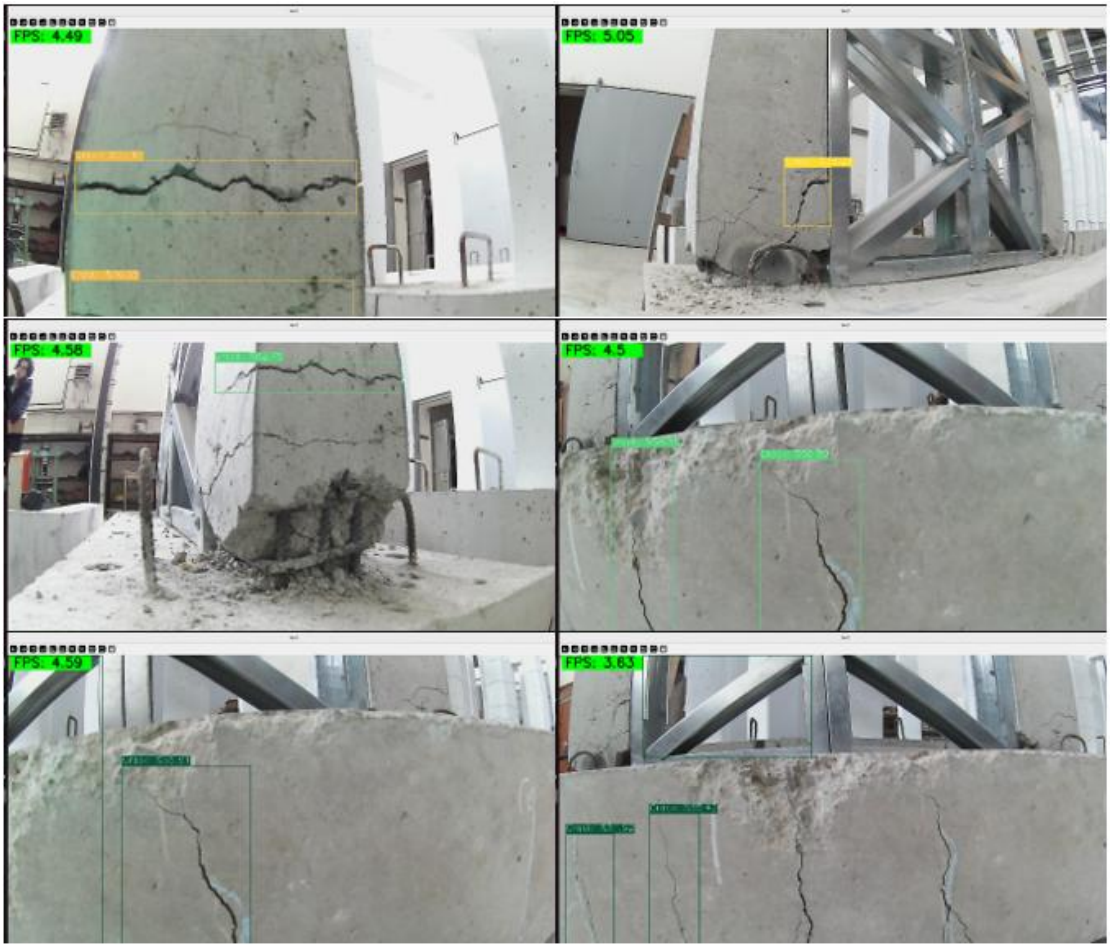


Fig. 14. Analysis results

During the testing phase, the YOLOv8-based crack detection model achieved highly successful results, particularly in accurately identifying and classifying cracks in structural elements. Analyses conducted on the captured images demonstrated that the model precisely detected cracks and accurately assigned them to the correct categories.

As shown in Fig. 14, the bounding boxes generated by the model were accurately placed over the regions containing cracks, and the locations of these cracks were determined with high precision. This level of accuracy indicates that the developed YOLOv8 model is capable of effectively detecting various types of cracks, including both micro-cracks and more visible structural damages. Moreover, the model's ability to maintain high detection performance across cracks of different sizes and locations highlights its robustness and versatility under varying conditions.

The results obtained during the tests confirmed that the model performs reliably not only on training datasets but also under real-world application conditions. Comprehensive measurements and evaluations demonstrated that the YOLOv8 model can be effectively utilized in the field of Structural Health Monitoring (SHM), offering significant advantages for on-site inspections.

Additionally, during this process, the integration between the robot dog's image processing capabilities and the YOLOv8 model was successfully achieved, enabling real-time crack detection. Through the analysis of images captured from multiple angles, the system exhibited considerable potential for advanced applications such as large-scale area scanning and automated defect detection.

In conclusion, this study successfully validated the reliability and efficiency of the developed YOLOv8 crack detection model under both laboratory and real-world conditions, confirming its applicability for critical structural health monitoring tasks.

4. Conclusions

In this study, a robust and efficient crack detection system was developed by integrating a state-of-the-art YOLOv8 deep learning model with the Unitree Go2 quadruped robot, aiming to enable autonomous, real-time structural inspection. The primary goal was to evaluate the feasibility and effectiveness of combining advanced object detection algorithms with mobile robotic platforms to facilitate the automated monitoring of structural integrity, particularly for reinforced concrete systems. The system was extensively tested in a controlled experimental environment at the Structural Health Monitoring Laboratory of Karadeniz Technical University. These tests were designed to simulate realistic inspection scenarios and assess the model's performance under practical conditions, including variable lighting, surface textures, and crack orientations.

Performance evaluation of the model revealed promising results. The normalized confusion matrix showed that the YOLOv8 model achieved a high crack detection accuracy of 89%, while demonstrating perfect classification (100%) for background regions. Although 11% of crack instances were incorrectly classified as background, the overall performance highlights the model's strong generalization capability and ability to reliably distinguish structural anomalies from non-defective areas. Moreover, the F1-Confidence Curve analysis indicated that the model achieved its peak F1-score of 0.84 at a confidence threshold of 0.531. This suggests that fine-tuning the confidence threshold can further optimize the model's detection performance, especially in safety-critical applications where false negatives must be minimized.

For the experimental validation, a full-scale reinforced concrete frame specimen was prepared using C25/30 concrete and B420C steel reinforcement. The specimen had dimensions of 550 mm × 1700 mm × 300 mm and a total height of 1850 mm, providing a realistic and sufficiently large surface for evaluating the model's ability to detect various types and sizes of cracks. The Unitree Go2 robot, equipped with a high-resolution onboard camera, was remotely navigated around the test specimen. As it moved, the robot continuously captured images of the concrete surface, which were processed in real time by the YOLOv8 model. This enabled the system to autonomously detect and localize cracks from different angles and distances, showcasing its adaptability to varying viewpoints.

Visual inspection results, supported by representative examples such as those presented in Fig. 10 and 14, confirmed the system's high localization accuracy. The model successfully identified and outlined crack regions with precise bounding boxes, even in complex surface conditions. This demonstrates the system's capability to perform detailed and reliable inspections without the need for manual intervention. The seamless integration of the YOLOv8 model with the Unitree Go2 robot resulted in a fully automated inspection workflow, capable of navigating, capturing, and analyzing structural data in real time.

In conclusion, the developed robotic inspection system has proven to be accurate, reliable, and efficient for detecting surface cracks in reinforced concrete structures. Its performance indicates strong potential for deployment in real-world Structural Health Monitoring (SHM) applications, particularly in scenarios where manual inspection is difficult, hazardous, or time-consuming. Future research will focus on improving the model's classification accuracy through advanced data augmentation, expanding the training dataset, and evaluating its performance across diverse structural materials and environmental conditions. Additionally, incorporating 3D

mapping and path-planning algorithms may further enhance the robot's autonomy and inspection coverage in complex environments.

References

Çolakoğlu, H. E., Hüsem, M., Demir, S., Akbulut, Y. E., & Yılmaz, Y. (2024). The behavior of reinforced concrete frames exposed to high temperature under cyclic load effect. *Structures*, *63*, 106446. Retrieved from https://doi.org/10.1016/j.istruc.2024.106446 .

Ilmak, D., Iban, M. C., & Şeker, D. Z. (2024). Deep learning-based scene classification of very high-resolution satellite imagery for post-earthquake damage assessment: A case study of the 2023 Kahramanmaraş earthquakes. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 249–256. Retrieved from https://doi.org/10.5194/isprs-archives-XLVIII-4-W9-2024-249-2024 .

Jia, J., Ye, W. (2023). Deep learning for earthquake disaster assessment: Objects, data, models, stages, challenges, and opportunities. *Remote Sensing* , 11. Retrieved from https://doi.org/10.3390/rs15164098.

Kumar, R. S. (2022 Dec 6). *Object detection with YOLO theory*. Retrieved from Object Detection with YOLO Theory .

Morfidis, K., Stefanidou, S., & Markogiannaki, O. (2023). A rapid seismic damage assessment (RASDA) tool for RC buildings based on an artificial intelligence algorithm. *Applied Sciences*, 3. Retrieved from https://doi.org/10.3390/app13085100.

Nelson, J. (2025, Jan 9). *What is YOLO? The ultimate guide*. Retrieved from What is YOLO? The Ultimate Guide [2025].

Ultralytics, (2025). *Ultralytics YOLOv8'i keşfedin.* Retrieved from Ultralytics YOLOv8 'i Keşfedin - Ultralytics YOLO Dokümanları .

Roboflow, (2025). *Everything you need to build and deploy computer vision applications*. Retrieved from Roboflow: Computer vision tools for developers and enterprises .

Qayoumi, A. S., Shalizi, H., & Dhankot, M. A. (2025). Review on artificial intelligence application in structural earthquake engineering. *International Journal of Science and Research Archive, 14*(01), 083-092. Retrieved from https://doi.org/10.30574/ijsra.2025.14.1.2659 .

Ye, X. W., Liu, Z. X., Ding, Y., & others. (2022). Post-earthquake damage recognition and condition assessment of bridges using UAV integrated with deep learning approach. *Structural Control & Health Monitoring,* 26. Retrieved from https://doi.org/10.1002/stc.3128.

Zhai, G., Xu, Y., & Spencer, B. F. (2024). Bidirectional graphics-based digital twin framework for quantifying seismic damage of structures using deep learning networks. *Structural Health Monitoring—An International Journal, 24.* Retrieved from https://doi.org/10.1177/14759217241231299.