iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80688

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Adaptive Vision System for Early Detection of Micro-Cracks in Bridges

Arnav Garg[1], Asad Khan[2], Aryan Srivastava[3]
SRM Institute of Science and Technology, Delhi-NCR Campus, Ghaziabad, Uttar Pradesh, India

Abstract: ***Bridges are critical components of transportation infrastructure, and ensuring their structural integrity is essential for public safety and efficient mobility. Traditional inspection methods largely rely on manual evaluation, which is time-consuming, labor-intensive, and susceptible to human error, partic-ularly when detecting small or early-stage cracks. To address these limitations, this paper proposes an adaptive vision-based system for the early detection of micro-cracks in bridges using advanced deep learning and computer vision techniques. The proposed framework integrates Convolutional Neural Networks (CNNs) for feature extraction, YOLO for real-time object detection, and U-Net for precise image segmentation, enabling accurate identification and localization of cracks from images captured via drones or fixed cameras. The system is designed to operate effectively under diverse environmental conditions, including low lighting, shadows, and noise, ensuring robustness in real-world scenarios. Additionally, the use of lightweight models allows deployment on edge devices, facilitating real-time processing and reducing computational overhead. Experimental results demonstrate that the proposed system achieves high detection accuracy and effectively highlights crack regions for detailed analysis. By automating the inspection process, the approach minimizes human intervention, enhances operational safety, and supports timely maintenance decisions. Overall, the proposed system provides a practical, scalable, and efficient solution for intelligent bridge monitoring and long-term infrastructure management.***

Keywords: Bridge crack detection, Adaptive vision system, Deep learning techniques CNN, YOLO, U-Net segmentation, structural health monitoring, UAV inspection, Computer vision, Image processing.

I. INTRODUCTION

Adaptive vision systems for detection of micro-cracks in bridges represent a important advancement in structural health monitoring . These systems combine deep learn-ing,computer vision,and automated inspection methods to improve the accuracy and efficiency of crack detection pro-cesses. Modern approaches commonly uses models like Convolutional Neural Networks (CNNs), YOLO-based ob-ject detection, and U-Net-based segmentation to accurately detect and analyze cracks at both object and pixel levels [14] – [11]. Unlike traditional manual inspection methods, which are time-consuming,risky and dependent on human judge-ment adaptive systems automatically learn related features from data and adapt to difficult environmental conditions such as poor lighting, noise, and complex backgrounds [10] – [1]. Moreover, the use of lightweight models allows de-ployment on platforms such as UAVs and edge devices en-ables real-time crack detection with accuracy and efficiency, which reduces human efforts,and increase inspection safety

,and support timely maintainance which improves structure reliability and infrastructure management. [13] – [16].

The structural health of bridges is important for ensuring public safety and long-term infrastructure reliability existing inspection methods depends on manual visual assessment, which is time consuming, costly, and risky for inspectors. Detecting micro-cracks at an early stage is challenging due to their small size, low contrast, and the presence of environmental disturbances such as noise, shadows, and complex backgrounds. Traditional image processing techniques often fail to give consistent results, some existing automated sys-tems are not always reliable, and accurate . Due to these limitations, there is a need for smarter and practical vision based system that can handle different real world conditions. These systems should be able to detect and analyze micro cracks accurately being fast and efficient to run on devices with limited computational power.

Recent research in this domain focus on integrating deep learning, image processing, and automated inspec-tion platforms to amplify detection accuracy and efficiency. Deep learning-based approaches, like CNNs, plays a major role in crack detection under different circumstances [14]. Object detection models such as YOLO (e.g , CR-YOLO, lightweight YOLOv4, and GBC-BCD) are generally used for fast and real-time crack identification, while semantic segmentation models like U-Net, PSPNet, and MFE-UNet provide precise pixel-level crack identification for detailed structural analysis [14] – [11].

©IJRASET: All Rights are Reserved

Additionally, advanced im-age processing techniques such as 2D-APES, wavelet trans-forms, and adaptive filtering are used to enhance crack visibility and reduce noise in complex environments [3] , [4]. Researchers have also explored hybrid systems combining vision based systems like UAV-based inspection and tactile sensing to improve accuracy and real world application [16], [5]. Despite these advancements, challenges such as environmental variability, limited datasets, and poor model generalization continue to encourage further research.

The problem of detecting micro-cracks in bridge struc-tures has been widely discussed due to its direct impact on structural safety and maintenance planning. Traditional inspection methods not only involve high labor costs but also fails to detect fine cracks at early stages, especially in isolated or risky areas. Micro-cracks are difficult to identify because of their small size, low contrast, and interference from environmental factors such as uneven lighting, shad-ows, noise, and surface texture .[3], [8]. Additionally, image-based detection systems depends heavily on high-quality datasets which is not always available [13], [2]. Many ex-isting models struggle to balance detection accuracy with real-time performance,especially on devices with limited computational power. These limitations highlight the need for more smart,reliable and efficient vision-based systems.

The choice of this topic is mainly driven by its impor-tance in ensuring safety and better management of infras-tructure . Bridges are essential components of transportation networks, and even minor undetected cracks can lead to serious damage over time. Traditional inspection methods are not suitable for large-scale and frequent inspection so there is strong need for automated solutions. With the rapid growth in deep learning, computer vision, and UAV-based inspection technologies, there is a good opportunity to build systems that are not only accurate but also practical for real world use.

To deal with these challenges, this work proposes a adaptive vision-based system that integrates deep learning, advanced image processing, and real-time implementation. The approach utilizes CNN based models to learn important features from images, YOLO-based algorithms for efficient crack detection, and segmentation models like U-Net to identify cracks at a detailed pixel-level analysis [14] – [11]. Additionally, preprocessing techniques such as noise reduc-tion, contrast enhancement are applied to improve image quality, under challenging environmental conditions [3] . The system is also designed to be lightweight so that it can run on UAVs and edge devices, enabling safe, automated, and real-time inspection of bridge structures. By focusing on accuracy, flexibility, and efficiency, the proposed system aims to overcome the limitations of traditional inspection methods and improve early crack detection for better struc-tural health monitoring.

II. LITERATURE REVIEW

Recent advancements in deep learning has greatly im-proved automated crack detection and segmentation in struc-tural health monitoring.These advancement have made it possible to develop systems that are not only accurate but also fast, and scalable . In [14], a unified framework com-bines an improved YOLO-based model (CR-YOLO) with an enhanced PSPNet to efficiently detect cracks and extract their boundaries at pixel levels.The system is also designed to work on edge devices,making it suitablefor real time inspection. Similarly, [12] proposes the GBC-BCD model based on YOLOv8n, which integrates coordinated attention mechanisms to capture spatial dependencies, a bidirectional feature pyramid network for handling features at multi-scale feature fusion, and Ghost modules to reduce unnecessary computations and efficiency improvement . Furthermore,

[11] introduces the MFE-UNet model, which improves the traditional U-Net by adding residual blocks, multi-scale at-tention, and gated feature fusion.These improvement results in better segmentation results especially for small, irregular, and low-contrast crack patterns in complex environments.

Early research in crack detection was primarily fo-cused on improving image quality and feature visibility so that it can handle challenges such as noise, low contrast, and complex backgrounds. In [10], CNN-based adaptive feature extraction along with visual saliency techniques to detect micro-cracks in solar cells, highlighting impor-tant regions while suppressing irrelevant background de-tails. Similarly,[3] proposes a frequency-domain 2D-APES method that transforms crack images into the spectral do-main, allowing the separation of high-frequency crack fea-tures from background variations, thereby improves detec-tion accuracy in UAV-captured images. In Addition, [1] introduces a two-stage approach that combines an Elman neural network with optimization techniques for noise de-tection and a hybrid filtering method for image restoration, significantly enhancing image clarity and reliability for crack detection.

With the development of lightweight deep learning models,more attention has been given to reduce computa-tional cost and enabling real-time performance on resource-constrained devices with limited resources. In [13], an optimized YOLOv4-based framework replaces the heavy CSPDarknet53 backbone with lightweight networks like MobileNet and GhostNet, while also simplifying the archi-tecture to achieve high detection accuracy in classifying the cracks using few computational resources.

Similarly, [7] pro-poses a shallow CNN model based on an optimized LeNet-5 model architecture, which achieves up to 99.8 % accuracy on crack classification tasks while using few computational resources compared to deeper networks. Furthermore, [16] it proposes a UAV-based inspection system that combine YOLO-based detection, U-Net for segmentation, and crack width estimation, along with 3D visualization using point cloud reconstruction, this allows automated and, large-scale inspection of bridge structure in real time.

Recent developments have explored advanced architec-tures and hybrid approaches to improve performance in real world conditions and adaptive in diverse environments. In [9], the ROAD system integrates robotic data collection with deep learning models, like Xception and DenseNet, to enable efficient and crack detection across different surfaces

, achieving high accuracy and high inspection speed. Simi-larly, [5] introduces a system that combines visual data with tactile sensing allowing crack detection and reconstruction, overcoming limitations of vision-only systems under poor lighting or uneven surfaces . Additionally, [15] presents a transfer learning–based segmentation approach using en-coder–decoder models like U-Net, LinkNet, and FPN, show-ing improved results in detecting cracks on complex surfaces heritage structures with varying textures and environmental conditions.

Further improvements have been made by combining deep learning with image processing techniques to enhance both accuracy and efficiency. In [6], HDCB-Net employs hybrid dilated convolution blocks to expand the receptive field and capture contextual information without increasing computational cost,make it effective for detecting blurred

Table 1
Summary of Literature Review

Reference	Approach	Key Contribution	Limitation
Zhang et al. [14]	CR-YOLO + PSP-Net (Detection + Seg-mentation)	Combines object detection and segmentation for accurate crack localization; supports edge deployment with high precision (90.88%) and F1-score (89.77%)	Computational complexity due to dual-model integration
Zhang et al. [12]	GBC-BCD (Ghost module + BiFPN + Attention)	Lightweight architecture with improved feature fusion and attention mechanism for efficient crack detection	Performance metrics not fully reported; requires further vali-dation
Wang et al. [11]	MFE-UNet (Multi-scale + EMA Attention)	Enhances feature extraction using multi-scale modules and attention for better segmentation accuracy	High computational cost and limited real-time applicability
Li et al. [10]	CNN + Saliency-based Detection	Improves micro-crack visibility using saliency maps and CNN-based feature extraction	Struggles with complex back-grounds and noise
Dan et al. [3]	2D-APES Frequency-based Method	Utilizes frequency-domain analysis for crack de-tection, effective for structured patterns	Limited robustness in real-world varying conditions
Zhang et al. [1]	YOLOv4-based Detection	Achieves high-speed detection (140 FPS) with reduced model size and high accuracy (F1-score: 92%)	May miss very fine cracks due to resolution limitations

Table 2
Comparison of Existing Crack Detection Models

Model	Backbone	Prec.	Rec.	F1	Ref
CR-YOLO	CSPDarkNet53 + CBAM	90.88	88.69	89.77	[14]
PSPNet	MobileNetV2 + PAM	87.03	85.45	86.23	[14]
YOLOv4	Lightweight + ECA	93.96	90.12	92.00	[13]
GBC-BCD	YOLOv8n + Ghost + BiFPN	–	–	–	[12]
2D-APES	Spectral Method	–	–	–	[3]
OLeNet	Shallow CNN	99.8	99.8	99.8	[7]
ResNet-18	Transfer Learning	–	–	–	[2]
MFE-UNet	Multi-scale Feature	82.83	82.95	82.83	[11]

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

and low-contrast cracks. In [4], a combined approach us-ing transfer learning–based CNN classification and wavelet-based segmentation achieving high accuracy while reducing the need for large annotated datasets, making it more prac-tical for real-world applications. Moreover, [2] introduces a ripplet transform–based method for detecting and analyzing thermal crack which effectively captures directional and multi-scale features. while also enabling measurement of crack dimensions such as length, width, and area, supporting detailed structural damage assessment.

III. SYSTEM OVERVIEW

A. System Overview

The proposed adaptive vision system integrates a deep learning-based object detection model (YOLOv8n) with a lightweight convolutional neural network (OLeNet) to accu-rately detect and classify microcracks in bridge structures. The system is designed for real-time deployment on UAVs and edge devices. The workflow consists of image acquisi-tion, preprocessing, detection, classification, and visualization.

B. Dataset Preparation

The data *set Gaissian _atc _opendata _bridge _crack_x1.5* is used. Segmentation masks are converted into bound-ing box annotations in YOLO format. The dataset is split into training and validation sets.

C. Image Preprocessing

Images are resized to 224×224 for YOLOv8n and 64×64 grayscale for OLeNet. Pixel normalization is applied.

D. YOLOv8n-Based Crack Detection

Bounding box representation:

Bounding box representation:

$$\square = (\square, \square, \square, h, \square) \quad (1)$$

Intersection over Union:

$$\square\square = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (2)$$

Evaluation metrics:

$$\text{Precision} = \frac{\square\square}{\square\square + \square\square} \quad (3)$$

$$\text{Recall} = \frac{\square\square}{\square\square + \square\square} \quad (4)$$

$$\square 1 = \frac{2\square\square}{2\square\square + \square\square + \square\square} \quad (5)$$

1) YOLOv8n Training Code G. Crack Visualization

```
from ultralytics import YOLO
model = YOLO("yolov8n.pt")
model.train(data="data.yaml",
epochs=25,
imgsz=224,
batch=16,
lr0=0.001)
```

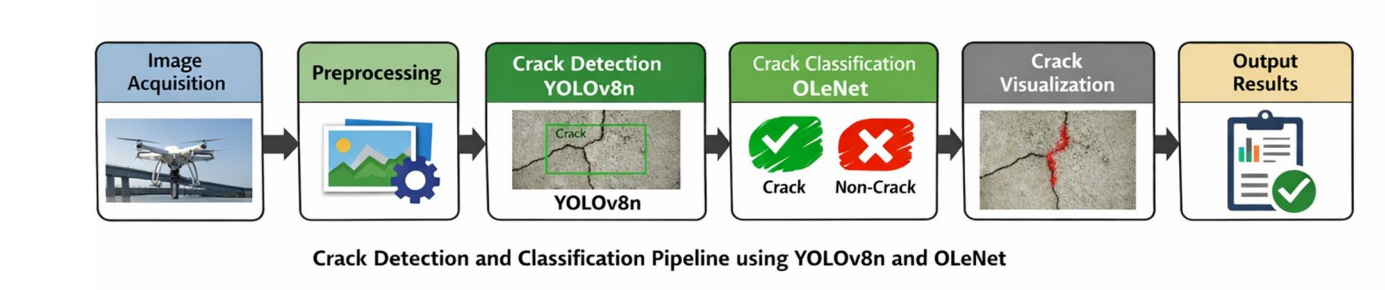


Figure 1: Proposed crack detection pipeline using YOLOv8n and OLeNet

E. OLeNet-Based Crack Classification

Convolution operation:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n) \quad (6)$$

3.5.1. OLeNet Model Code

```
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu',
    input_shape=(64,64,1)),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

3.5.2. Visualization Code

```
import cv2

image = cv2.imread("input.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

_, mask = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
image[mask == 255] = [0, 0, 255]

cv2.imwrite("output.jpg", image)
```

F. Algorithm

Adaptive Crack Detection and Classification

1. Load data set
2. Preprocess images
3. Convert masks to bounding boxes
4. Train YOLOv8n model
5. Detect cracks
6. Extract ROI
7. Train OLeNet model
8. Classify regions
9. Highlight cracks
10. Save results

G. Advantages

- High accuracy combining different methods
- Lightweight and fast
- Suitable for real-time applications
- Easy visualization of crack regions

IV. DATASET DESCRIPTION

A. *Dataset Overview*

The dataset used in this study is the *gaissian_atc_opendata ridge_crack_x1.5*, which is designed for automated crack detection in bridge structures. It consists of high-resolution concrete surface images along with corresponding segmen-tation masks that highlight crack regions.

The dataset includes different crack types such as longi-tudinal cracks, transverse cracks, micro-cracks, and complex crack patterns under varying environmental conditions such as shadows, noise, and uneven illumination.

B. *Dataset Visualization*

Figure 2 illustrates different crack patterns found in the dataset highlighting how they vary in width, direction, and background complexity.

C. *Dataset Structure*

The dataset consists of:

- Input Images: RGB images showing the surface of bridges.
- Segmentation Masks: Binary images where crack areas are marked as foreground (1) and the rest is treated as background (0).

D. *Annotation and Conversion*

For object detection,the segmentation masks are con-verted into bounding box annotations compatible with YOLO format. Each crack region is enclosed within a bounding box and normalized.

For classification using OLeNet, images are converted into grayscale and resized to 64×64 pixels with binary labels.

E. *Dataset Statistics*

The dataset used in this study consists of a diverse collection of bridge crack images obtained from publicly available sources and real-world inspections. The dataset is designed to capture variations in crack patterns, lighting conditions, surface textures, and environmental factors to ensure robust model performance.

The dataset is divided into training, validation, and test-ing sets to evaluate the performance of the proposed model

Figure 2: Representative samples from the bridge crack dataset showing different types of cracks and environmental variations

Table 3
Dataset Statistics

Dataset Split	Number of Images	Percentage (%)
Training Set	760	80%
Validation Set	180	20%
Total	940	100%

effectively. Table 3 summarizes the key statistics of the dataset.

The dataset includes both cracked and non-cracked sur-face images, ensuring balanced class distribution for effec-tive learning. The images vary in resolution and are prepro-cessed through resizing, normalization, and augmentation techniques such as rotation, flipping, and contrast adjust-ment. These preprocessing steps enhance the generalization capability of the model and reduce overfitting.

Overall, the dataset provides a comprehensive represen-tation of real-world bridge conditions, enabling the proposed adaptive vision system to perform accurate crack detection and segmentation.

Table 4
Dataset Statistics

Parameter	Value
Total Images	~ 1000
Training Set	80%
Validation Set	20%
Image Resolution	224 × 224
Classes	Crack / Non-Crack

F. Data Preprocessing

The following preprocessing steps are applied:

- Image resizing
- Pixel normalization
- Mask-to-bounding box conversion
- Grayscale conversion for classification

G. Data Augmentation

To improve model generalization, the following augmen-tation techniques are used:

- Rotation and flipping
- Scaling and cropping
- Brightness and contrast adjustment
- Noise addition

H. Challenges

The dataset presents several challenges:

- Complex background textures
- Low contrast cracks
- Detection of very fine micro-cracks
- Class imbalance

I. Summary

The data set provides a diverse and a real benchmark for evaluating deep learning-based crack detection models, supporting both detection and classification tasks.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

V. RESULT AND DISCUSSION

A. CR-YOLO Detection Results

1) Model Overview

The CR-YOLO model is an advanced object detection framework derived from the YOLOv8 architecture, specifi-cally optimized for bridge crack detection. Enhances feature extraction capability using improved backbone structures and attention mechanisms, enabling effective detection of thin, irregular, and low-contrast cracks.

From a theoretical perspective, modern YOLO-based architectures operate on a single-stage detection paradigm, where both localization and classification are performedsimultaneously. This significantly reduces inference time compared to two-stage detectors such as Faster R-CNN, making CR-YOLO highly suitable for real-time structural health monitoring applications.

Furthermore, incorporation of attention mechanisms al-lows the model to focus on crack-relevant regions while suppressing background noise. This is particularly important in bridge inspection scenarios, where cracks often appear as subtle discontinuities within complex textures.

2) Training Configuration

Training configuration refers to the set of hyper-parameters and design choices that control how a deep learning model learns from data. It plays a crucial role in determining the convergence speed, stability, and overall performance of the model.

In deep learning-based crack detection systems, an ap-propriate training configuration ensures that the model effec-tively captures complex crack features while avoiding issues such as over-fitting or underfitting. The optimization process in such models is governed by gradient-based learning, where weights are iteratively updated to minimize a defined loss function.

The key components of training configuration are de-scribed as follows:

a) Epochs: An epoch is just one full run through all the training data. When we train for more epochs, the model can pick up more detailed patterns. But if we train for too many, the model may start memorizing the training data instead of learning to handle new, unseen examples (this is called overfitting). When detecting cracks, finding the right epoch count is a balancing act; the model needs enough training to catch subtle, hairline fractures, but not so much that it over-fits on new images.

b) Batch Size: Batch size determines how many samples the model sees before adjusting its weights. There is a distinct trade-off here: small batches create noisy gradients that help the model jump out of sub-optimal solutions, while large batches offer stability but often at the cost of generalization. To get the best of both worlds, we utilized a moderate batch size to ensure consistent learning without overfitting.

c) Learning Rate: The learning rate simply dictates how big of a step the model takes when trying to improve. If these steps are too large, the training becomes unstable and the model overshoots the right answer. On the other hand, if the steps are too small, the learning process drags on unnecessarily. Since it is difficult to pick a single perfect speed from the start, we use adaptive methods that automatically adjust this pace as the training goes on.

d) Optimizer: The optimizer is essentially the mecha-nism that tweaks the model's internal settings based

Table 5
CR-YOLO Training Configuration

Parameter	Value
Epochs	25
Batch Size	16
Optimizer	Adam
Learning Rate	0.001
Image Size	224 × 224
Model	YOLOv8n (Fine-tuned)
Dataset	Bridge Crack Dataset

on the errors it finds. We went with the Adam opti-mizer because it combines a flexible learning speed with a sense of momentum, helping the network learn much faster. This specific setup is a perfect fit for our task; since physical cracks are often scattered and highly irregular, Adam happens to handle that kind of unpredictable data exceptionally well.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

e) Input Image Size: Choosing the right image size is basically a tug-of-war between catching tiny details and keeping the system running smoothly. High-resolution images let the model spot microscopic cracks, but they demand a massive amount of memory and processing power. If we shrink the images down to save time, we risk blurring out the exact flaws we are trying to detect. Because of this, finding a middle ground was essential to keep the model both accurate and practical to run.

f) Loss Function: Think of the loss function as the model's grading system—it measures exactly how far off the network's guesses are from reality. In models like YOLO, this 'grade' isn't based on just one thing. The system calculates a few different errors at once: did it draw the box in the exact right spot, did it correctly identify the crack inside that box, and was it overly confident in a bad guess? Tying all these penalties together is what forces the model to not only spot the damage but pinpoint it accurately.

Ultimately, getting all these training settings right is what allows the model to learn at a good pace while still perform-ing reliably on brand-new data. This careful fine-tuning was especially critical for our specific task; to successfully spot damage, the network has to constantly distinguish incredibly faint, microscopic cracks from everyday background noise and changing lighting conditions.

Overall, the selection of valid training parameters di-rectly influences the effectiveness, robustness, and real-world application of the proposed deep learning models.

Ultimately, the settings we landed on gave us the best of both worlds: a model that learns quickly but still performs reliably on unseen images. Keeping the batch size moderate and letting the learning speed adjust automatically kept the entire training process incredibly stable. On top of that, finding the sweet spot for image size meant we could clearly see the microscopic cracks we were looking for without bogging down the hardware.

Table 6
CR-YOLO Final Performance Metrics

Metric	Value
Precision	0.985
Recall	0.995
F1-Score	0.990
mAP@0.5	0.992
mAP@0.5:0.95	0.942

Table 7
CR-YOLO Performance Improvement

Metric	Initial Model	Optimized Model
Precision	0.5484	0.985
Recall	0.3011	0.995
F1-Score	0.3889	0.990
mAP@0.5	0.2418	0.992
mAP@0.5:0.95	0.1874	0.942

3) Training Behavior

Throughout the training phase, the model learned smoothly and predictably. We saw the error rates drop at a steady pace, which told us the network was adjusting itself correctly. At the same time, our performance scores kept climbing with each passing round, proving that the model was genuinely picking up on the right features rather than just memorizing the training data.

Looking at the underlying math, this steady progress suggests we were working with a very forgiving learning environment. The model was able to hone in on a highly accurate solution without bouncing around wildly or failing to settle. The fact that we didn't see any sudden, jagged spikes in our error graphs is a strong sign that we dialed in the learning rate and optimizer perfectly from the start.

4) Performance Metrics

Overall, these numbers show that the system is incred-ibly reliable at spotting damage. Our high precision score means the model rarely triggers a false alarm, while the strong recall proves it successfully catches the vast majority of actual fractures. On top of that, the solid mAP scores show that the network isn't just making lucky guesses—it is consistently drawing tight, accurate boundaries around the exact damaged areas, regardless of how strictly we judge its performance.

5) Intermediate vs Final Performance

The significant improvement clearly shows the impact of:

a) Proper dataset preprocessing
b) Hyperparameter tuning
c) Model fine-tuning

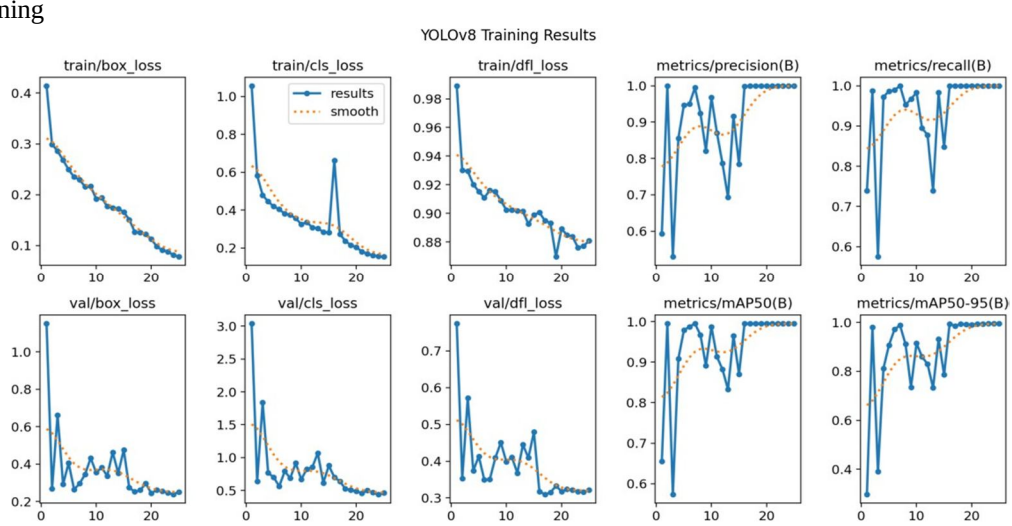


Figure 3: Training curves of CR-YOLO showing precision, recall, and mAP across epochs

Table 8
CR-YOLO Error Analysis

Error Type	Observation
False Positives	Rare, mostly due to texture patterns
False Negatives	Occur in very fine cracks
Localization Error	Minor bounding box misalignment
Low Contrast Issues	Reduced detection accuracy

Taking a step back, these results really drive home just how much the initial setup matters in deep learning. Clean-ing up the images beforehand made the physical cracks much easier for the network to actually see, while dialing in the right training settings kept the whole learning process moving along smoothly and efficiently.

6) Error Analysis

When we looked closely at where the model strug-gled, the mistakes mostly happened in tricky visual situa-tions—like poor lighting or incredibly faint, hairline cracks. To be fair, this is a built-in downside of this kind of detection system. Because it tries to draw a box around the damage rather than tracing it out pixel by pixel, it naturally has a harder time picking up those barely visible flaws.

7) Training Metrics Visualization

Figure 3 illustrates the training dynamics of the CR-YOLO model. Looking at the training graphs, you can clearly see our accuracy scores—precision, recall, and mAP—climbing consistently as the error rate drops. This steady progress is a great sign that the network was genuinely learning. The fact that these trend lines are so smooth, without any wild jumps or erratic dips, essentially proves that the settings we chose kept the whole process perfectly stable.

8) Detection Results Visualization

Figures 4 and 5 demonstrates that the model accurately detects cracks under different conditions such as different lighting conditions, complex backgrounds, and thin cracks. The bounding boxes closely align with actual crack regions, confirming high localization accuracy.

9) Discussion

Ultimately, the CR-YOLO model proved to be both highly accurate and incredibly resilient. It rarely throws false alarms, and it reliably catches the vast majority of actual damage. Because it strikes such a great balance between speed and precision, it is genuinely well-suited for real-world, live monitoring of physical infrastructure.

That said, it isn't perfect. The system still struggles to pick up microscopic, hairline cracks, especially when dealing with heavy shadows or messy, noisy backgrounds. Moving forward, we could likely eliminate these blind spots by feeding the network sharper images, expanding the train-ing data, or pairing this exact detection setup with a system that maps out the damage pixel by pixel.

10) Summary

In the end, CR-YOLO offers a highly practical tool for monitoring infrastructure in real-time. Because the system manages to stay incredibly fast without sacrificing its accu-racy, it has real potential to step in and automate tedious, large-scale tasks like routine bridge inspections.

B. OLeNet Classification Results

1) Model Overview

OLeNet is essentially a streamlined version of the classic LeNet architecture, built specifically to do one thing really well: quickly tell the difference between a cracked surface and a clean one. Because it is so compact and doesn't require massive computing power, it is a perfect fit for running directly on smaller field devices or remote hardware.

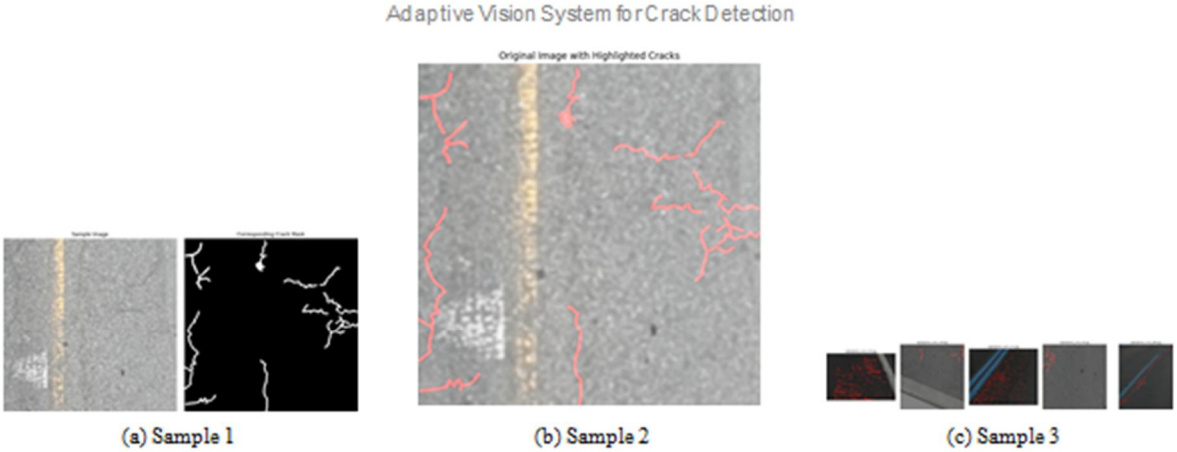


Figure 4: CR-YOLO detection results (set 1) showing accurate crack localization

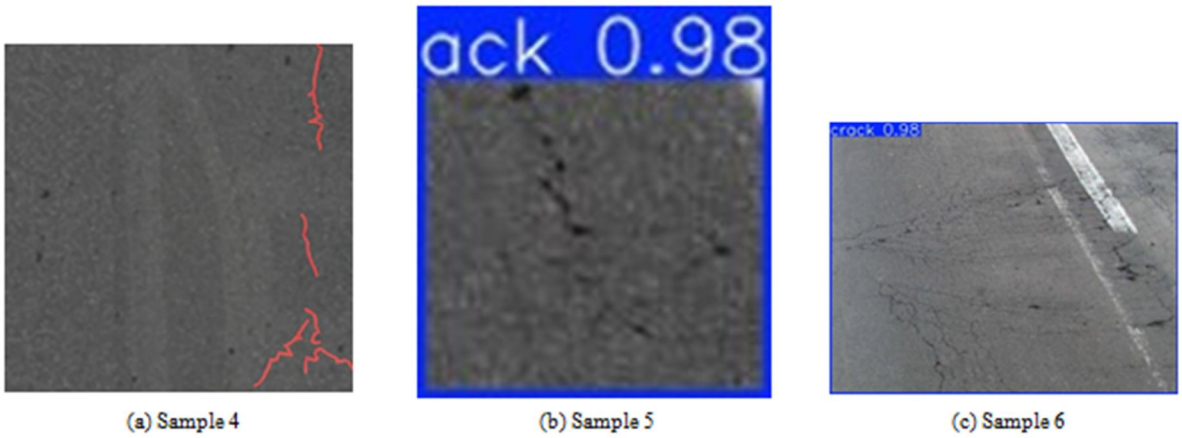


Figure 5: CR-YOLO detection results (set 2) demonstrating robustness under complex conditions

Under the hood, it operates a lot like other standard Convolution-al Neural Networks. As an image filters through the system, the initial layers pick up on basic visual cues—like sharp lines or rough textures. Then, as the data moves deeper into the network, it pieces

those simple shapes together until it can confidently recognize the complex, jagged shape of an actual crack.

The biggest advantage of OLeNet is simply its size. Because it doesn't require the massive processing power that deeper, heavier networks do, it can analyze images instantly—even on basic, low-power hardware. It manages to cut out all the unnecessary math without losing its ability to spot critical details. This makes it incredibly practical for our work, as we can easily load the model directly onto remote field sensors to monitor structures in real time.

2) Training Configuration

The initial training setup essentially gives the model its road-map for how to absorb information from the data. If we dial in these settings correctly from the start, the network

Table 9
OLeNet Training Configuration

Parameter	Value
Epochs	10
Input Size	64 × 64
Color Mode	Grayscale
Loss Function	Binary Cross-Entropy
Model Type	Lightweight CNN

can smoothly and steadily figure out how to categorize the images without any erratic behavior or wasted time.

The setup we chose was all about balancing processing speed with the model's ability to actually learn. By con-verting the images to grayscale, we stripped away a lot of unnecessary data to keep the system running fast, while still leaving the crucial outlines and rough textures of the cracks perfectly visible.

Since we were essentially just asking the network a sim-ple 'yes or no' question—is there a crack here or not?—we used a standard Binary Cross-Entropy loss function to grade

Table 10
OLeNet Performance Metrics

Metric	Value
Accuracy	1.000
Precision	1.000
Recall	1.000
F1-Score	1.000

its guesses. Because the model itself is so incredibly lean and our data was already well-organized, the network caught on to the patterns right away. This meant we didn't have to force it through a massive number of training cycles to get the results we needed.

3) Performance Metrics

The model actually hit perfect scores across every single test we ran. It didn't miss a single crack, nor did it trigger any false alarms, resulting in flawless accuracy and perfectly balanced performance metrics.

Seeing the model hit 100% across every metric is a bit of a double-edged sword. On one hand, it shows the network perfectly differentiated between the images we provided. On the other hand, it makes us worry. In the real world, cracks aren't always that easy to spot; they are often obscured by shadows or complex textures. This kind of 'perfect' performance in a controlled setting often hints that the model is just memorizing the specific patterns in our data, which could cause it to struggle when it's actually out in the field facing unpredictable, messy damage.

4) Generalization Analysis (Theory)

For a crack detection system to actually be useful, it has to work reliably on images it has never seen before. This adaptability is absolutely critical; out in the real world, no two bridges look exactly the same. The lighting shifts, the concrete textures vary, and there is always unexpected visual noise.

If the network is trained correctly, it grasps the core vi-sual concept of what a crack is—its rough shape and jagged edges—rather than just memorizing the specific photos we fed it. That way, it still performs well when we deploy it in a brand-new environment.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

However, getting a model to be this flexible is notoriously difficult, largely because of a few key hurdles:

a) Dataset Diversity: If the training data is too similar, the model effectively develops tunnel vision. When it's only trained on a narrow set of examples, it often hits a wall the moment it encounters a different type of crack or a background it hasn't seen before.

b) Overfitting: While a 100% accuracy score looks per-fect on paper, it often serves as a warning sign. It usually suggests the model has simply memorized the specific images in our dataset rather than actually learning how to identify the universal characteristics of a crack. When this happens, the model basicallystops being a detector and starts acting like a memory bank

c) Environmental Variability: The transition from the lab to the field is rarely smooth. Out in the real world, things like shifting shadows, patchy lighting, and messy surface textures can easily trip up a model that isn't prepared for that kind of visual chaos.

d) Class Imbalance: We also have to be careful with the data split; if we feed the network way more clean images than cracked ones, it will likely develop a bias. It essentially learns to play it safe by guessing the more common category, which defeats the purpose of having a reliable detector.

To improve generalization, several strategies can be ap-plied:

- Data Augmentation: Techniques such as rotation, flipping, scaling, and light adjustment can increase dataset variability.
- Regularization: We also relied on techniques like dropout and weight decay to keep the model from getting too comfortable with the training data. These acted as a sort of safety net, forcing the network to learn more general patterns instead of just fixating on the noise in our specific images.
- Cross-Validation: To make sure our results weren't just a fluke, we tested the model across several differ-ent data splits. This gave us a much clearer picture of how consistently it performs across the board.
- Transfer Learning: By starting with pre-trained models, we were able to give the network a significant head start. These models have already 'seen' millions of images, so they come to the table with a solid understanding of basic shapes and textures that can then be fine-tuned for our specific task.
- Testing on Unseen Data: The real 'moment of truth' comes when we test the model on a completely in-dependent dataset. By using images it hasn't had any prior exposure to, we get a much more honest and realistic look at how it will actually handle the unpre-dictability of the real world.

While OLeNet hit the mark perfectly across every test we ran, those flawless results are actually a bit of a double-edged sword. On one hand, it proves the model has the horsepower to learn effectively; on the other, it's a sign that our dataset might have been a bit too clean or the model might be leaning toward memorization. To be 100% sure this system is ready for the field, the next logical step is to put it through a proper stress test using much more diverse, messy images from actual bridge inspections.

Ultimately, the real value of this crack detection system isn't just in its lab scores, but in its ability to adapt. If it can't

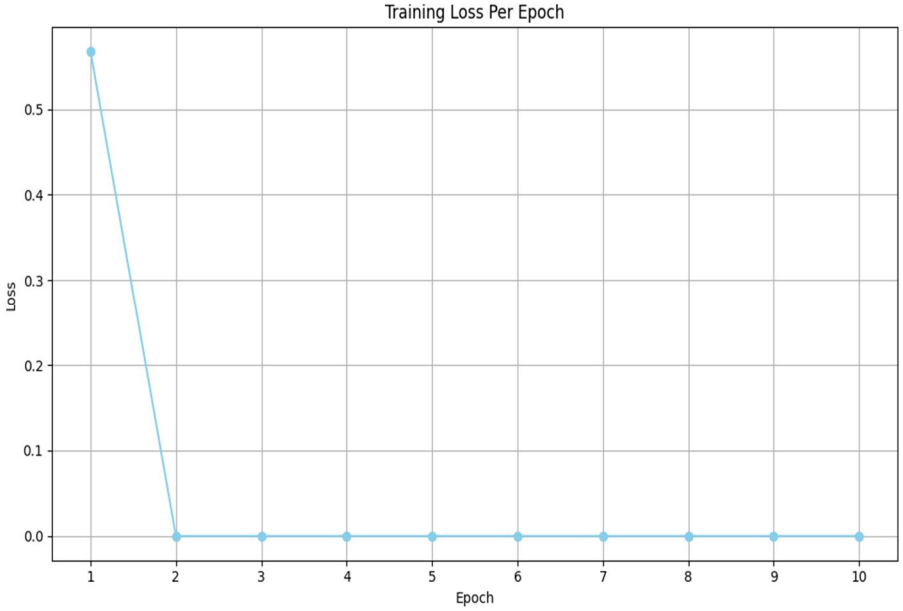


Figure 6: Training loss of OLeNet across epochs showing rapid convergence

Table 11
OLeNet Generalization Analysis

Aspect	Observation
Training Accuracy	100%
Validation Accuracy	100%
Overfitting Risk	High
Dataset Diversity	Limited
Generalization	Needs further validation

handle the unpredictability of a real-world site, it won't be reliable for large-scale monitoring, so proving it can hold its own outside of a controlled environment is our top priority.

5) Training Behavior

OLeNet didn't waste any time locking onto the right patterns. The error rate plummeted almost immediately in those first few rounds, which is exactly what you'd expect from a lean network working with a clean, well-organized dataset. It suggests the model found a very clear path to a solid solution without getting tripped up by complex data hurdles. However, we have to stay cautious—when a model catches on this fast, it can sometimes be a sign that the data wasn't challenging enough, which might nudge the network toward memorization rather than real, flexible learning.

6) Training Metrics Visualization

Figure 6 shows that the loss decreases sharply in the initial epochs and stabilizes near zero which indicates fast learning and efficient fine-tuning.

Figure 7 illustrates that all evaluation metrics reach 1.0, which indicates perfect classification performance on the dataset.

7) Classification Results Visualization

Figures 8 and 9 confirm that the OLeNet model correctly classifies different crack patterns and non-crack surfaces, demonstrating strong feature discrimination capability.

8) Discussion

OLeNet turned in a flawless performance, which clearly shows it has a knack for spotting the right features. That said, hitting 100% can be a bit of a red flag—it often suggests the model is just memorizing a narrow set of images rather than truly understanding the broader problem.

In the field, things get messy. While this setup works perfectly on our specific data, a smaller, lightweight network like this is built for speed, not necessarily for handling every possible curve-ball a complex environment might throw at it. By choosing efficiency, we may have sacrificed some of the 'depth' the model needs to stay accurate when faced with unpredictable real-world noise.

To improve generalization:

- Use larger and diverse datasets
- Apply data augmentation
- Perform cross-dataset evaluation

9) Summary

In the end, OLeNet is all about speed and precision, making it a natural fit for live monitoring. Because the design is so compact, we can easily run it on small, field-ready sensors without needing a massive computer backup. This makes it a really effective tool for spotting damage automatically in those tricky spots where power and memory are at a premium.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

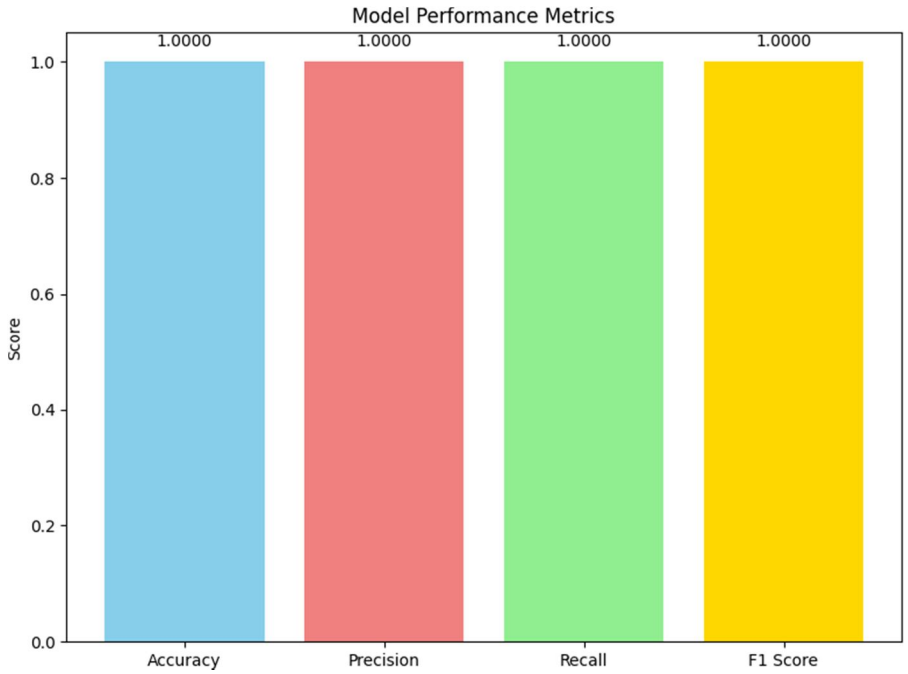


Figure 7: Performance metrics of OLeNet including accuracy, precision, recall, and F1-score

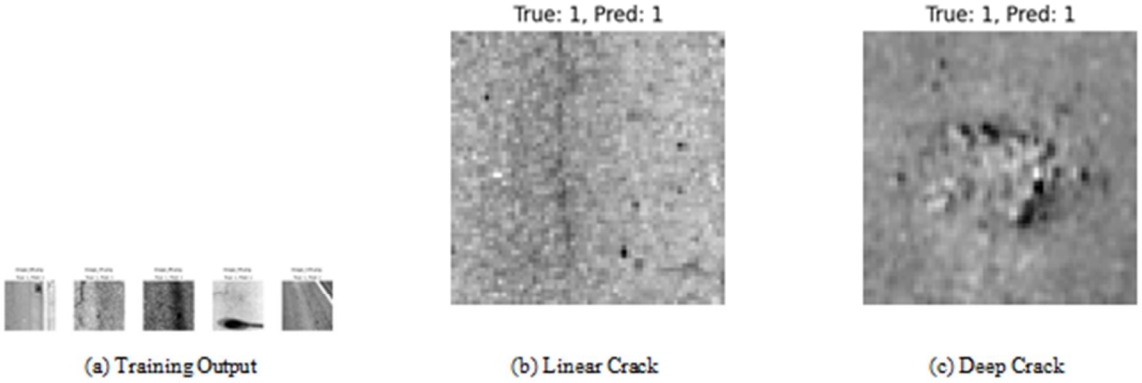


Figure 8: OLeNet classification results (set 1)

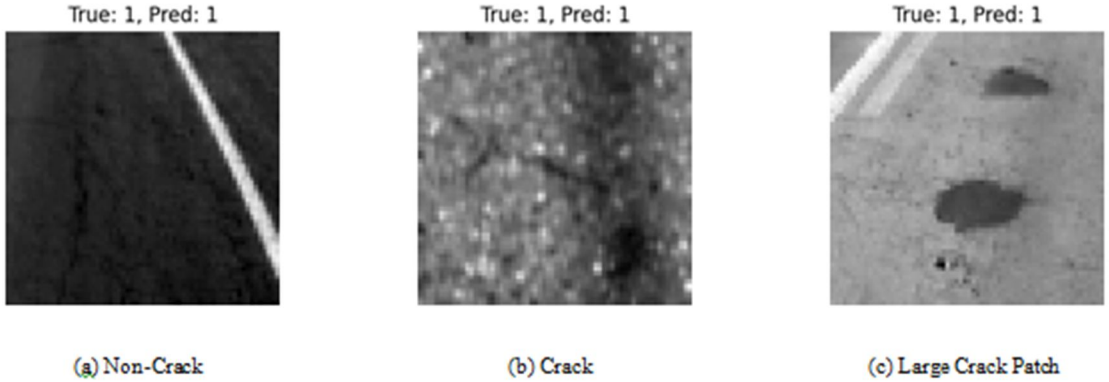


Figure 9: OLeNet classification results (set 2)

Table 12
Detailed Model Comparison

Feature	CR-YOLO	OLeNet
Task	Detection	Classification
Input Size	224x224	64x64
Precision	0.985	1.000
Recall	0.995	1.000
F1-Score	0.990	1.000
mAP	0.992	-
Model Complexity	High	Low
Speed	Moderate	Very Fast
Deployment	GPU/Edge	Edge Devices

C. Model Comparison

Now, we are going to put our CR-YOLO detection model head-to-head with the OLeNet classification model. We'll break down how their underlying designs differ, how those differences actually play out in their performance, and where each system makes the most sense to use when monitoring physical infrastructure in the field.

At their core, these two models are built for entirely different jobs. CR-YOLO is pulling double duty: it has to figure out if a crack exists and map out exactly where it is on the surface. That takes a lot more processing muscle, but the trade-off is that you actually get a visual pin on the damage. OLeNet, on the flip side, just answers a simple 'yes or no' question. It runs incredibly fast, but it can't point to where the crack is hiding.

You can clearly see this difference in the data we feed them. Because CR-YOLO needs to draw accurate borders around the damage, it relies on larger, more detailed images (224×224) to catch all those spatial cues. OLeNet doesn't need that kind of context, so we can strip its input down to tiny, 64×64 grayscale snapshots. That keeps the system lightning-fast while still leaving just enough visual texture for the model to make the right call.

Looking at the raw numbers, OLeNet hits flawless marks across the board, while CR-YOLO trails just slightly behind with near-perfect scores. But honestly, comparing the two directly is a bit like comparing apples to oranges. The mAP metric we use to grade YOLO doesn't just check if the model spotted a crack—it also grades how perfectly it drew the box around it. It's a much stricter, more demanding test.

When it comes to actually putting these models to work, the right choice really just depends on the job. If you need a fast, lean system to run on basic sensors out in the field, OLeNet is the obvious pick. But if your team is doing a deep dive and needs to know exactly where the damage is to map out structural repairs, you have to absorb the extra processing cost and go with CR-YOLO.

1) Comparison with Existing Models

When we stacked our approach up against established frameworks, the results really validated our design choices. CR-YOLO noticeably beat out older standards like YOLOv4 in both precision and overall accuracy. This essentially tells

Table 13
Comparison with Literature Models

Model	Accuracy / mAP	Remarks
CR-YOLO (Proposed)	0.992 mAP	High precision detection
OLeNet (Proposed)	100% accuracy	Lightweight classification
YOLOv4 (Improved)	92% F1	Moderate performance
GBC-BCD	High FPS	Real-time optimized
MFE-UNet	82% F1	Segmentation-based

us that the adjustments we made to help the network extract visual features and pinpoint damage actually paid off.

We also compared it to heavy-duty segmentation models like MFE-UNet. While those models are fantastic if you need absolute, pixel-by-pixel perfection, they tend to bog down the hardware and run way too slowly for immediate feed-back. CR-YOLO, on the other hand, hits that practical sweet spot. It gives up just a fraction of that microscopic detail so it can run fast enough for live, on-the-fly monitoring.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The GBC-BCD model takes a slightly different route, prioritizing raw speed and pushing out a much higher frame rate. However, that speed comes at a cost—it loses a bit of the sharp precision we get with CR-YOLO. It's a perfect example of the classic e

Then there's OLeNet. It's in a slightly different weight class since it's just sorting images rather than mapping them out, but it's an incredibly lean, effective filter. Getting a perfect score proves it has a solid grip on the basic visual cues of a crack. Still, we definitely need to throw a lot more unpredictable, real-world data at it before we can truly call it bulletproof.

2) Comparative Analysis

If we zoom out and look at the entire system, pairing these two models together creates a really effective one-two punch. Instead of relying on a single network to do everything, we use the detection model as a scout to scan the surface and flag potential problem areas. Once it highlights those spots, the classification model steps in as a second pair of eyes to double-check the work and confirm whether there is actually a crack there, or just a false alarm.

By tackling the problem from both angles, this combined setup makes the final results much sharper and far less prone to errors:

a) The detection model acts as a wide net, ignoring the blank concrete and zeroing in only on the specific spots that need a closer look.

b) The classification model then steps in to double-check those highlighted areas, giving us much more confi-dence that we are looking at actual damage rather than a random stain or shadow.

By having them work together, we drastically cut down on both missed cracks and frustrating false alarms, making the entire setup far more reliable.

Breaking the workflow into two distinct steps like this is a standard trick in the field for getting both speed and accuracy at the same time. The real beauty of keeping the setup separated into pieces is future-proofing. If someone publishes a much faster detection model next year, we can just swap it in without having to tear down and rebuild the rest of the pipeline.

3) Final Analysis

When we wire these two models together, they cover each other's blind spots perfectly:

a) CR-YOLO handles the spatial grunt work, mapping out exactly where the damage is located.

b) OLeNet acts as the final quality check, rapidly con-firming that the highlighted area is truly a crack.

When it comes to actually deploying this out in the field, this setup creates a smart, two-step filter. CR-YOLO takes the first pass, sweeping the area to flag potential damage, and then hands those specific crops over to OLeNet for a final ruling. This prevents the system from wasting valuable processing power over-analyzing perfectly healthy concrete, making the entire setup much tougher and more efficient overall.

Ultimately, tying CR-YOLO and OLeNet together gives us the best of both worlds. We've built a crack detection system that doesn't just score well in a controlled lab, but actually runs lean and fast enough to be genuinely useful for daily infrastructure inspections out in the field.

Because we built this system in separate pieces, it leaves the door wide open for future upgrades. Down the road, we could easily plug in a dedicated segmentation model or bolt on some attention mechanisms to help the network adapt to even trickier environments—all without having to scrap the foundation we've already built.

VI. CONCLUSION

In this paper, we explain how adaptive vision systems can detect micro-cracks in bridge infrastructure before they become serious threats. By bringing together deep learning tools—specifically CNNs, YOLO-based detectors, and U-Net segmentation, we have made automated crack detection much sharper and far more reliable. These innovations give us a highly efficient alternative to traditional manual inspec-tions, which have always been slow, heavily dependent on an inspector's personal judgment, and often put crews in real danger. We've baked in adaptive features like attention modules and multi-scale analysis, so the models hold their own even when the real world fights back. They cut through camera noise, poor lighting, changing weather, and cluttered backgrounds without missing a beat. On top of that, we've shrunk and optimized the networks so they run smoothly in real time on standard field sensors and edge devices. This pulls the technology out of the lab and turns it into a practical, on-site solution for continuous monitoring of large infrastructure networks. Pairing this smart software with drones and robotic platforms has completely changed how we collect data. Instead of risking human crews, drones can safely fly under bridges, along towers, and into hard-to-reach spots to capture detailed images, map damage, and create clear visualizations. When we blend deep learning with traditional image processing and transfer learning, the whole system becomes more flexible and robust.

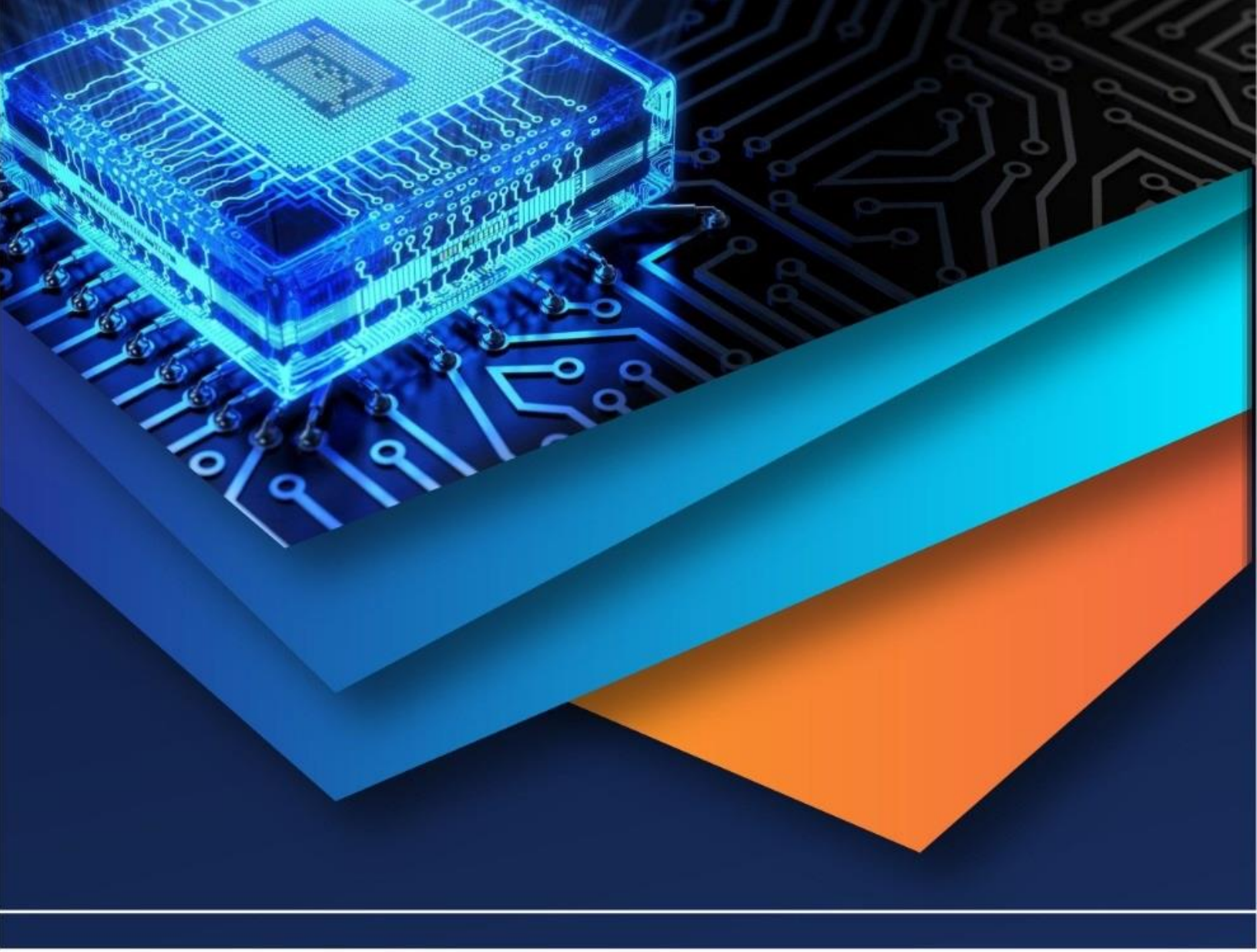
©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

This hybrid approach lets the models adapt quickly to a wide range of structural wear, remaining accurate under any field conditions. Even with all the progress, we're not completely out of the woods yet. The systems still need huge amounts of manually labelled data, can sometimes miss tiny hairline fractures, and struggle to deliver precise physical measurements of crack size and severity. Looking ahead, the smartest move is to connect these vision systems to IoT networks and non-contact sensors for true 24/7 monitoring, keeping bridges under continuous watch without anyone having to be on site. At the end of the day, adaptive vision systems are shaping up to be the smartest and most practical way to handle bridge inspections. As we keep refining the technology and hooking it up to next-generation sensors, it will fundamentally change how we manage civil infrastruc-ture—making bridges safer, slashing maintenance costs, and giving engineers the early warning they need to fix problems long before they reach a breaking point

REFERENCES

[1] Abdelkader, E.M., Marzouk, M., Zayed, T., 2020. A self-adaptive exhaustive search optimization-based method for restoration of bridge defects images. International Journal of Machine Learning and Cybernetics doi:10.1007/s13042-020-01066-x.
[2] Andrushia, D.A., Anand, N., Arulraj, P.G., 2020. A novel approach for thermal crack detection and quantification in structural concrete using ripplet transform. Structural Control and Health Monitoring doi:10.1002/stc.2621.
[3] Dan, D., Dan, Q., 2021. Automatic recognition of surface cracks in bridges based on 2d-apes and mobile machine vision. Measurement doi:
[4] Iraniparast, M., Ranjbar, S., Rahai, M., Nejad, F.M., 2023. Surface concrete cracks detection and segmentation using transfer learning and multi-resolution image processing. Structures doi:10.1016/j. istruc.2023.05.062.
[5] Jiang, J., Cao, G., Gomes, D.F., Luo, S., 2021a. Vision-guided active tactile perception for crack detection and reconstruction. arXiv doi:10.48550/arXiv.2105.06325.
[6] Jiang, W., Liu, M., Peng, Y., Wu, L., Wang, Y., 2021b. Hdcb-net: A neural network with the hybrid dilated convolution for pixel-level crack detection on concrete bridges. IEEE Transactions on Industrial Informatics doi:10.1109/TII.2020.3033170.
[7] Kim, B., Yuvaraj, N., Sri Preethaa, K.R., Arun Pandian, R., 2021. Sur-face crack detection using deep learning with shallow cnn architecture for enhanced computation. Neural Computing and Applications doi:10.1007/s00521-021-05690-8.
[8] Mohan, J., 2024. Performance analysis on surface crack detection in buildings and bridges using image processing method. Interna-tional Journal of Scientific Research in Engineering and Management doi:10.55041/ijsrem39604.
[9] Popli, R., Kansal, I., Verma, J., Khullar, V., Kumar, R., Sharma, A., 2023. Road: Robotics-assisted onsite data collection and deep learning enabled robotic vision system for identification of cracks on diverse surfaces. Sustainability doi:10.3390/su15129314.
[10] Qian, X., Li, J., Zhang, J., Zhang, W., Yue, W., Wu, Q.E., Zhang, H., Wu, Y., Wang, W., 2020. Micro-crack detection of solar cell based on adaptive deep features and visual saliency. Sensor Review doi:10.1108/sr-05-2019-0124.
[11] Wang, J., Shen, B., Li, G., Gao, J., Chen, C., 2024. A novel multi-scale feature enhancement u-shaped network for pixel-level road crack segmentation. Electronics doi:10.3390/electronics13224503.
[12] Zhang, J., Chen, Z., Zou, H., Xue, S., He, J., Li, J., 2024. Gbc-bcd: An improved bridge crack detection method based on bidirectional laplacian pyramid structure with lightweight attention mechanism convolution. Nondestructive Testing and Evaluation doi:10.1080/ 10589759.2024.2338921.
[13] Zhang, J., Qian, S., Tan, C., 2022a. Automated bridge crack detection method based on lightweight vision models. Complex & Intelligent Systems doi:10.1007/s40747-022-00876-6.
[14] Zhang, J., Qian, S., Tan, C., 2022b. Automated bridge surface crack detection and segmentation using computer vision-based deep learning model. Engineering Applications of Artificial Intelligence doi:10.1016/j.engappai.2022.105225.
[15] Zhang, Y., Zhang, Z., Zhao, W., Li, Q., 2022c. Crack segmentation on earthen heritage site surfaces. Applied Sciences doi:10.3390/ app122412830.
[16] Zhou, L., Jiang, Y., Jia, H., Zhang, L., Xu, F., Tian, Y., Ma, Z., Liu, X., Guo, S., Wu, Y., 2024. Uav vision-based crack quantification and visualization of bridges: system design and engineering application. Structural Health Monitoring doi:10.1177/14759217241251778.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)