Automation in Construction 176 (2025) 106273

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

# Integrative AI and UAV-based visual recognition with metaheuristics for automated repair cost analysis of bridge structural deterioration

Jui-Sheng Chou [a,*], Jhe-Shian Lien [a], Chi-Yun Liu [a,b]

[a] Department of Civil and Construction Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
[b] School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ, USA

ARTICLE INFO

*Keywords:*
Bridge maintenance
Deterioration detection
Unmanned aerial vehicles (UAVs)
Multi-stage computer vision
Metaheuristic optimization
Cost estimation

ABSTRACT

Aging bridges urgently need maintenance, as many exceed their lifespans. Traditional inspections are manual, time-consuming, costly, and error-prone. This has prompted a shift toward integrating advanced technologies to automate inspection processes and provide more efficient and accurate maintenance solutions. This paper introduces a multi-stage automated inspection system for bridge maintenance designed to classify bridge components and accurately assess the type and extent of deterioration. Unmanned aerial vehicles (UAVs) capture high-resolution images of bridge components, enabling comprehensive visual data collection without requiring manual access to challenging or hazardous areas. The inspection process employs the Vision Transformer (ViT) model for precise image classification, while You Only Look Once (YOLO) is used for instance segmentation. To further enhance the system's effectiveness, the Pilgrimage Walk Optimization (PWO)-Lite algorithm is applied to optimize the detection of deteriorated areas and estimate repair costs. This integration improves structural assessments, extends bridge longevity, and benefits bridge management agencies.

## 1. Introduction

Bridges are fundamental to global infrastructure, ensuring connectivity, facilitating transportation, and driving economic growth. However, as these structures age, maintaining their structural integrity has become increasingly urgent [1]. Aging, environmental stressors, and heavy usage have escalated the demand for more efficient and reliable maintenance solutions, making advanced inspection methods indispensable [2].

Taiwan exemplifies this global challenge. Its subtropical climate and varied terrain—including mountains, plains, and rivers—make bridges vital for connectivity and economic growth [3]. However, with many bridges over 30 years old, maintaining this aging infrastructure poses a significant challenge [4]. Approximately 28,000 bridges span the country, many of which have surpassed their intended lifespans, underscoring the pressing need for effective maintenance solutions [5].

Traditional visual inspections remain common but often prove inadequate, particularly for hard-to-reach components such as bridge undersides [6]. These areas, typically situated high above riverbeds or other challenging environments, necessitate specialized equipment, such as bridge inspection vehicles or rope-based methods, which increases safety risks and reduces the efficiency of inspections.

To address these challenges, this study utilizes unmanned aerial vehicles (UAVs) to photograph the undersides of bridges in western Taiwan, focusing on critical structural components, including bridge decks, piers, abutments, and wingwalls. The study examines multiple deterioration types: concrete cracking, spalling, honeycombing, exposed corroded rebar, peeling paint on steel plates, steel plate corrosion, and bolt corrosion. The analysis utilizes advanced computer vision models—Vision Transformer (ViT) for image classification and You Only Look Once (YOLO)v8 for instance segmentation. These models enable precise identification of bridge components, accurate delineation of deteriorated areas, and reliable estimation of maintenance costs.

The research follows a structured approach with several key phases (Fig. 1). First, bridge images are collected, classified, and preprocessed, with deterioration areas annotated using LabelMe. Next, the ViT and YOLOv8 models are trained and evaluated. Subsequently, the YOLOv8 model undergoes performance optimization through Particle Swarm Optimization (PSO) and Pilgrimage Walk Optimization (PWO)-Lite algorithms, followed by a comparative analysis of their effectiveness. The final phase develops a comprehensive maintenance cost estimation system that provides bridge inspectors with a practical tool featuring

* Corresponding author at: Department of Civil and Construction Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan.
*E-mail addresses:* jschou@mail.ntust.edu.tw (J.-S. Chou), m11105111@mail.ntust.edu.tw (J.-S. Lien), d10905002@mail.ntust.edu.tw, chiyunliu@asu.edu (C.-Y. Liu).

https://doi.org/10.1016/j.autcon.2025.106273
Received 6 February 2025; Received in revised form 2 May 2025; Accepted 7 May 2025
Available online 24 May 2025

reduced time and labor requirements, improved assessment accuracy, and a user-friendly interface for field operations.

## 2. Literature review

This section synthesizes four critical research domains to establish the foundation for an integrated bridge deterioration assessment system: (1) the limitations of current bridge management systems in external defect detection, (2) technological gaps in automated inspection methods, (3) unresolved challenges in deep learning-based defect classification, and (4) the unmet need for precise damage localization. The review explicitly identifies these gaps, which the proposed ViT-YOLOv8 hybrid model aims to address.

### 2.1. Overview of bridge management systems

A Bridge Management System (BMS) is a comprehensive computer system designed to support decision-making throughout the entire lifecycle of a bridge project, encompassing design, construction, operation, and maintenance. By streamlining infrastructure management, BMS enhances efficiency and reduces unnecessary costs. Many countries, including Finland, Germany, and Japan, have invested significantly in

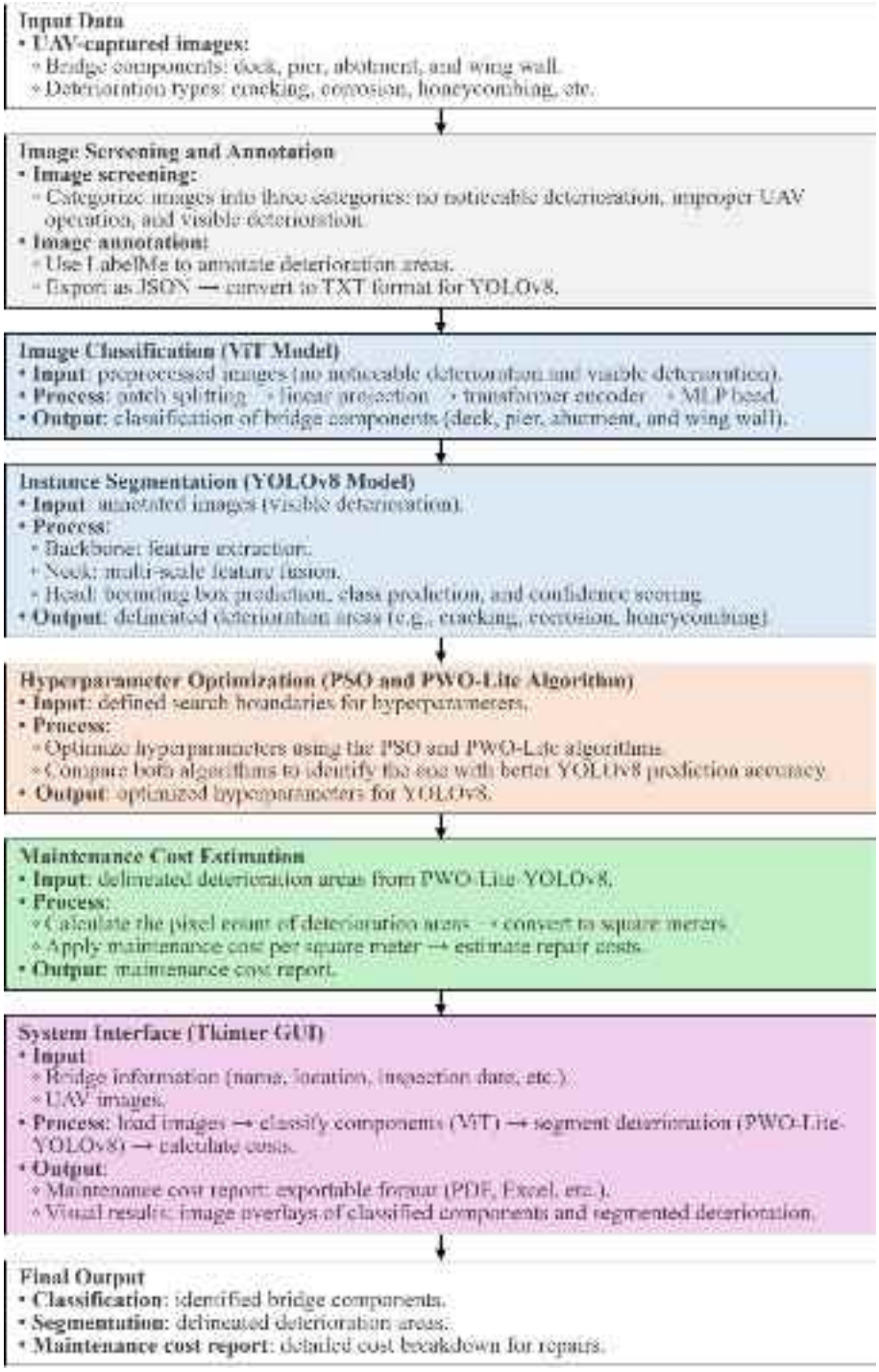


**Fig. 1.** Research workflow.

developing effective BMS to improve the sustainability and safety of their infrastructure [7].

Academically, Wan et al. developed a BMS based on Building Information Modeling (BIM) technology, introducing a standardized structural modeling method to establish bridge BIM models rapidly [8]. Using numerical methods to encode bridge information, they defined defect levels and prioritized maintenance and repair tasks, facilitating more efficient maintenance monitoring. Moreover, integrating the Web-BIM BMS with portable devices enables collaborative management and information synchronization, significantly enhancing the efficiency of bridge management operations.

Yang et al. developed an intelligent bridge management and maintenance framework that integrates data from 3000 bridges [9]. This framework includes primary bridge data, inspections, structural damage, maintenance records, and personnel management. It is structured into several layers, including a data source, storage and computing layer, knowledge representation layer, knowledge computation layer, and knowledge service layer. This framework enables more intelligent bridge management by incorporating cloud computing, big data analytics, and AI technologies, seamlessly integrating inspection processes with structural health monitoring.

Similarly, Lee et al. [10] introduced an innovative predictive model based on artificial neural networks, termed the backward prediction model. This model predicts historical bridge condition ratings using limited inspection records and incorporates external non-bridge data, such as traffic volume, population, and climate, to establish correlations with existing bridge condition ratings. Predicting historical condition ratings of various bridge components provides valuable insights for developing proactive maintenance strategies.

In Taiwan, significant progress has been made in the development of BMSs. The Taiwan Bridge Management Information System, established in 2001 by the Institute of Transportation under the Ministry of Transportation and Communications, was initially focused on registering basic bridge information and maintenance records. Over time, this system has evolved to incorporate new technologies, including real-time integration with water conditions, weather, and seismic information. More recently, the focus has shifted toward integrating BIM and utilizing data analysis to predict bridge deterioration and execute maintenance work more effectively [11].

Despite these advancements, most research emphasizes internal structural monitoring using fiber optic and piezoelectric sensors. However, accurately assessing external deterioration—especially in hard-to-reach areas, such as substructures—requires the broader adoption of computer vision. This study addresses this critical gap and contributes to emerging developments in BMS.

### 2.2. Advances and challenges in automated bridge inspection technologies

Inspecting beneath bridge decks and substructures is inherently challenging due to their elevated and often inaccessible locations, making close-up visual inspections difficult [12]. Traditional methods primarily rely on manual inspections, which are time-consuming, labor-intensive, and susceptible to subjectivity and inconsistency, often falling short of modern safety and efficiency standards [13,14]. However, recent advancements in drone technology and computer vision have transformed bridge inspection practices [15]. Drones equipped with high-resolution cameras and sensors are now widely used to capture detailed images of structural damage, enabling more accurate and efficient assessments [16].

Integrating computer vision and AI techniques has further enhanced the capabilities of automated bridge inspection systems [17]. For instance, deep convolutional neural networks (CNNs) have been widely adopted for detecting surface defects such as cracks, spalling, and corrosion [18]. Often enhanced by transfer learning, these models have demonstrated remarkable accuracy in locating and quantifying defects [19]. However, despite their success, challenges remain in deploying these models in real-world scenarios, mainly due to their limited ability to capture long-range dependencies in image data and their reliance on large-scale annotated datasets.

Recent research has explored ViT models, which utilize self-attention mechanisms to capture global contextual information in images [20]. Unlike CNNs, which focus on local features through convolutional operations, ViT models excel at modeling relationships between distant regions of an image, making them particularly suitable for detecting complex and multi-scale defects in bridge structures [21]. However, ViT models typically require substantial computational resources and large datasets for training, which can limit practical applications [22].

On the other hand, YOLOv8 is renowned for its speed and accuracy in object detection and instance segmentation tasks [23]. YOLOv8 builds on the strengths of its predecessors by incorporating advanced architectural improvements, such as anchor-free detection and a more efficient feature extraction backbone, making it highly suitable for real-time applications [24]. However, YOLOv8's reliance on local feature extraction can sometimes result in missed detections of subtle or globally distributed defects [25].

These technologies—drones, CNNs, ViTs, and YOLOv8—represent significant progress in modernizing bridge inspection. Nevertheless, challenges remain, including limited annotated datasets, high computational demands, and difficulties adapting to varied field conditions. To address these limitations, this study proposes a hybrid inspection framework that combines ViT's contextual sensitivity with YOLOv8's detection efficiency, aiming to enhance the accuracy and practicality of automated defect identification across diverse bridge environments.

### 2.3. Deep learning-based classification for bridge component defect detection

Image classification models are a cornerstone of computer vision, having undergone significant advancements with the rise of deep learning [26,27]. These models are particularly valuable in bridge inspection, enabling automated detection and classification of structural defects [28]. Recent studies have demonstrated the effectiveness of deep learning models in identifying and localizing deterioration in bridge images [29]. For example, object detection models like YOLO have been widely used to mark areas of deterioration with bounding boxes, providing a visual representation of damage [30].

A framework based on Feature Pyramid Networks (FPN) has been proposed to facilitate the automatic segmentation of fatigue cracks in high-resolution images of steel box girders [31]. The model improved computational efficiency and accurate crack delineation by applying image preprocessing techniques, precisely resizing, and partitioning. Furthermore, an automatic hierarchical model has been proposed to classify and correlate bridge surface images at multiple levels, including structure, component, and defect type [32]. This approach enhances the accuracy and efficiency of bridge inspections by comprehensively analyzing bridge conditions.

However, most existing models focus on surface-level defects [14,33], with limited exploration of substructure deterioration. Additionally, the types of deterioration identified are often simplistic [34], and there is a lack of models capable of detecting and classifying complex or multi-scale defects. This study aims to bridge these gaps by leveraging ViT for component-wise classification of critical bridge elements: (1) bridge decks, (2) piers, (3) abutments, and (4) wing walls. The ViT architecture's self-attention mechanism enables robust modeling of distinctive visual patterns across multiple spatial scales, facilitating more accurate differentiation among these structurally diverse components [35].

### 2.4. Instance segmentation for bridge damage detection and localization

While image classification enables effective categorization of structural components, actionable bridge maintenance requires precise

localization and delineation of defects. Instance segmentation fulfills this need by combining object detection with pixel-level segmentation, offering detailed insights into the geometry and spatial extent of damage such as cracks, spalling, and corrosion [36–38].

In bridge inspection, instance segmentation enables accurate detection and outline of localized damage, such as cracks, spalling, and corrosion, supporting condition assessment and decision-making [39]. Recent models, such as Mask R-CNN and YOLO-based segmentation variants, have shown promising results in infrastructure applications [40–42]. YOLOv8, among them, stands out for its anchor-free design and lightweight architecture, making it well-suited for real-time deployment on mobile inspection platforms [43,44].

However, despite its efficiency, YOLOv8's reliance on convolutional operations limits its ability to model long-range dependencies and global contextual features—capabilities that are especially important for detecting large-scale or irregular deterioration patterns, such as corrosion beneath bridge decks or delamination in substructure components [45].

To address this, we propose a hybrid instance segmentation framework that integrates the global contextual reasoning of ViT with YOLOv8's real-time localization capabilities. This complementary synergy enhances the segmentation of complex defects, enabling more robust and interpretable inspection results across diverse bridge components.

Despite these innovations, key challenges persist, including (1) the limited availability of annotated datasets for diverse and complex defect types and (2) the high computational costs of integrating transformer-based architectures. This study utilizes a self-collected, well-annotated image dataset encompassing various bridge types and defect classes to mitigate these issues. Moreover, we employ model optimization techniques to improve inference efficiency, making the framework more practical for deployment in real-world inspection environments. This hybrid approach enhances defect detection accuracy and facilitates more informed maintenance decisions, ultimately leading to safer, more durable infrastructure.

## 3. Methodology

This study proposes an integrative AI and visual recognition model for classifying bridge components and segmenting deterioration patterns embedded into a bridge maintenance cost estimation system. The methodology employs ViT for classification and YOLOv8 for segmentation, enhanced by the metaheuristic optimization algorithms (PWO-Lite and PSO) for hyperparameter tuning and a multi-stage inspection system. The workflow begins with ViT classifying components (deck, pier, abutment, wing wall), guiding YOLOv8 to segment deterioration patterns such as cracking, exposed bars, honeycombing, paint peeling, and corrosion, ensuring precise localization and quantification of deterioration while maintaining computational efficiency.

### 3.1. Vision transformer (ViT)

Introduced by Google in 2021 [46], ViT is an image classification model based on the Transformer architecture. Unlike traditional CNNs that rely on local receptive fields, ViT uses the self-attention mechanism to capture long-range dependencies across images, enhancing performance in tasks with complex structures [47]. ViT is adaptable to various image resolutions and sizes, making it a powerful tool in visual image classification across different domains [48].

The ViT architecture in Fig. 2 includes a primary backbone network and a classification layer. The backbone comprises the Linear Projection of Flattened Patches and the Transformer Encoder. Images are divided into fixed-size patches, flattened, and embedded into the Transformer, which processes them using multi-head attention and feed-forward neural networks to extract global features. The multilayer perceptron (MLP) head produces the final classification result.

#### 3.1.1. Backbone

The backbone begins with Patch Splitting, where the input image is divided into non-overlapping fixed-size patches (e.g., 16 × 16 pixels), as shown in **Eq. (1)**.

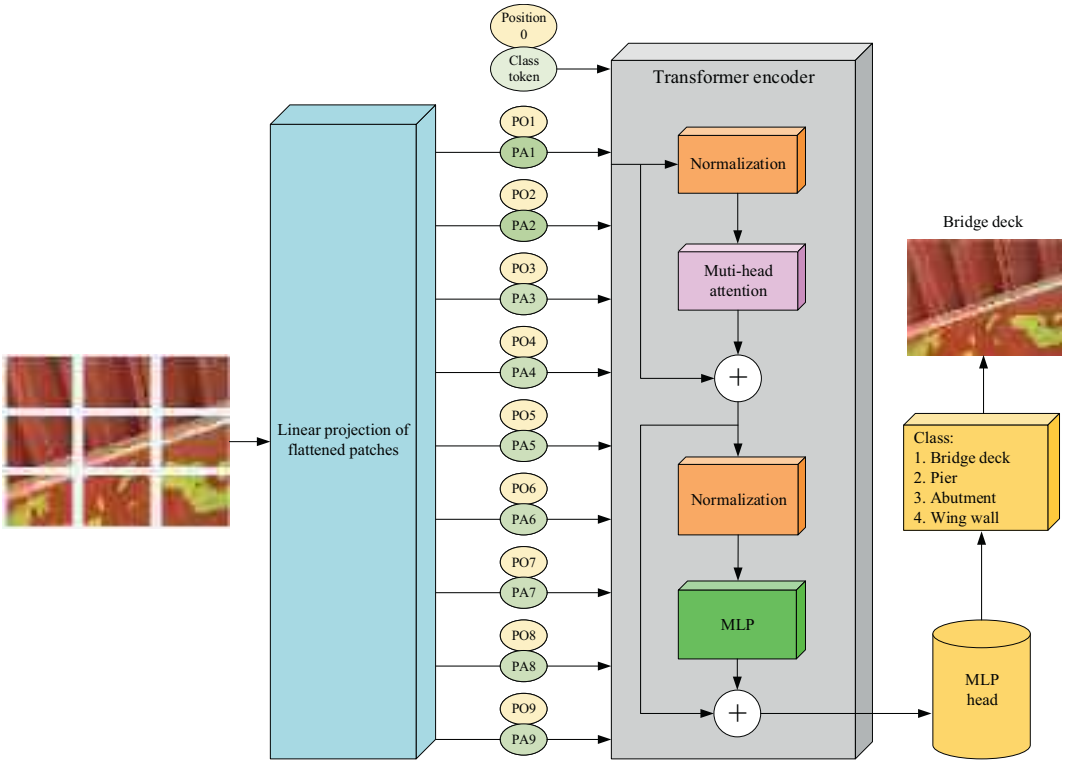


**Fig. 2.** ViT model architecture.

$$N = \frac{H \cdot W}{P^2} \tag{1}$$

Where $N$ is the number of non-overlapping image patches, $H$ is the height of the image, $W$ is the image's width, and $P$ is the size of the image patch. These patches are flattened and projected into high-dimensional feature vectors through a linear layer. Positional encoding is then added to these vectors to preserve the spatial structure of the image.

The Transformer Encoder processes these vectors, employing the Multi-Head Attention Mechanism to capture diverse features. This is followed by feed-forward layers that enhance the model's expressiveness and stability, which is crucial for image classification tasks.

#### 3.1.2. Prediction head

The MLP head transforms the output of the Transformer Encoder into the final classification result. This layer comprises several fully connected layers, utilizing nonlinear activation functions such as *ReLU*, which enable the model to learn higher-level features. The input to the MLP head is denoted as $Z$, the feature vector produced by the Transformer Encoder. The output $H$ of the fully connected layer is computed as shown in **Eq. (2)**.

$$H = ReLU(Z \bullet w_1 + b_1) \tag{2}$$

Where $H$ is the output of the fully connected layer, $w_1$ is the weight matrix, and $b_1$ is the bias vector. The final production passes through a Softmax function, which converts it into a probability distribution for each class and provides the final classification result.

### 3.2. You only look once (YOLO)v8

The YOLOv8 model architecture consists of three primary components: the Backbone, Neck, and Head networks (Fig. 3). The Backbone is the feature extraction module that processes the input image to capture essential visual patterns. Following this, the Neck performs multi-scale feature fusion to enhance detection accuracy across different object sizes. Finally, the Head network generates the detection outputs, including bounding box coordinates, class predictions, and confidence scores. The model begins by normalizing input images to a standardized size of 640 × 640 pixels before processing them through this sequential pipeline.

#### 3.2.1. Backbone

The Backbone network utilizes a combination of convolutional layers and advanced modules for efficient feature extraction. Conventional Conv layers perform the initial processing, detecting fundamental image features such as edges and textures. The architecture incorporates C2f (Cross Stage Partial Fusion) modules, which split the feature maps into two paths: one undergoes convolutional processing while the other remains unchanged. These paths are subsequently merged, balancing computational efficiency with effective feature extraction in **Eq. (3)**.

$$o\left(i,j\right) = \sum_{M=0}^{M-1} \sum_{N=0}^{N-1} I\left(i+m, j+n\right) \cdot k\left(m,n\right) \tag{3}$$

Where $o(i,j)$ is the output feature map at position $(i,j)$, $I$ is the input image, and $k$ is the kernel. In this equation, $M$ and $N$ denote the height and width of the convolution kernel, respectively, while $m$ and $n$ are the indices that iterate over the kernel's dimensions.

Additionally, the Backbone employs SPPF (Spatial Pyramid Pooling - Fast) to capture multi-scale contextual information. This technique applies max-pooling operations with varying window sizes to the feature maps (**Eq. 4**), enabling the model to effectively recognize objects at different scales.

$$P\left(i,j\right) = \max_{0 \le m < M_p, 0 \le n < N_p} I\left(i+m, j+n\right) \tag{4}$$

Where $P(i,j)$ represents the value of the pooled feature map at position $(i,j)$, $I$ is the input image, and $M_p$ and $N_p$ define the dimensions of the pooling window. In this equation, m and n are the indices that iterate over the height and width of the pooling window, respectively. This method enhances the model's ability to aggregate spatial information across different scales, improving its performance in object detection tasks.

#### 3.2.2. Neck

The Neck module specializes in fusing multi-level features from the Backbone to improve detection performance across varying object scales. It incorporates two key components: SPP (Spatial Pyramid Pooling) and PANet (Path Aggregation Network). The SPP layer enhances spatial context by processing features at multiple resolutions, while PANet aggregates features from different network paths to enrich the model's representational capacity. This fusion process concatenates

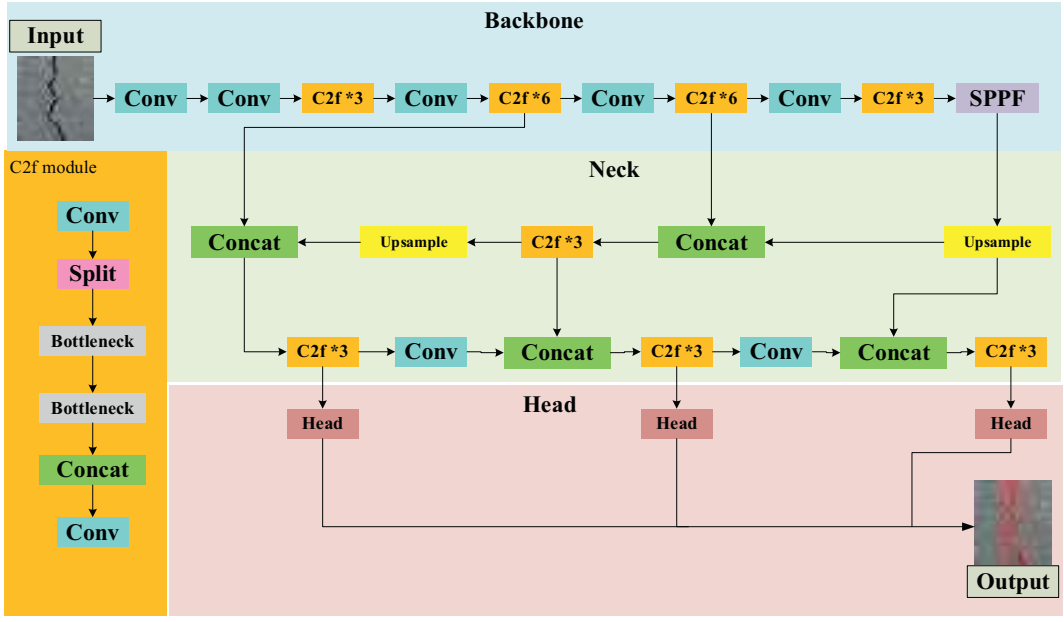


**Fig. 3.** YOLOv8 model architecture.

features from numerous layers (**Eq. 5**), creating comprehensive feature maps that improve detection accuracy.

$$F^{'} = concat(F_1, F_2, \cdots, F_n) \tag{5}$$

Where $F^{'}$ is the concatenated feature map, combining the individual feature maps $F_1$, $F_2$, …, $F_n$. This process enhances the model's accuracy and reliability in object detection and instance segmentation tasks.

#### 3.2.3. Head

The Head network produces the final detection results through an anchor-free approach that simplifies computation while maintaining precision. It generates three key outputs: bounding box coordinates, class probabilities, and confidence scores. The class probabilities are computed using softmax activation (**Eq. 6**), which converts raw scores into probability distributions across all candidate classes.

$$P(c_i|x) = \frac{e^{s_i}}{\sum_{j=1}^{c} e^{s_j}} \tag{6}$$

Where $P(c_i|x)$ is the probability that the input image belongs to class $i$, $s_i$ is the score for class $i$, $c$ is the total number of classes, and $j$ is the index that iterates over all classes.

### 3.3. Pilgrimage walk optimization (PWO)-Lite

#### 3.3.1. Concept of PWO-Lite algorithm

During the instance segmentation model training process, we integrate the lightweight version of the Pilgrimage Walk Optimization (PWO) algorithm, known as PWO-Lite [49], which was inspired by the traditional Taiwanese folk activity "Mazu Pilgrimage" as proposed by Chou and Liu [50]. This algorithm is employed to optimize hyperparameter settings, thereby improving the accuracy of model predictions. The search process of the algorithm emulates the gathering and movement patterns of Matsu devotees, as illustrated in Fig. 4. The pseudocode for the PWO-Lite algorithm is presented in Fig. 5.

Fig. 4 uses a cultural metaphor to illustrate the algorithm's main stages and supporting mechanisms. The core phases—divination-block casting, pilgrimage, and return ceremony—occur sequentially, guiding the algorithm's initialization, exploration, and convergence processes. In this representation, divination-block casting is depicted by two crescent-shaped blocks, symbolizing the initiation of the search. Pilgrimage, shown as two bearers carrying a sacred palanquin, represents the journey of exploration. Finally, the return ceremony, illustrated as a figure holding a divine image being reseated in the hall, signifies the completion of the search.

During the pilgrimage phase, auxiliary mechanisms—leisure ceremony, crawling beneath the palanquin, and palanquin robbing—are employed intermittently to prevent stagnation and enhance convergence based on the search conditions. Each mechanism is represented metaphorically: Leisure ceremony (a temple icon) encourages additional exploration to reduce the risk of stagnation; crawling beneath the palanquin (figures crawling under the palanquin) promotes convergence by refining promising regions; and palanquin robbing (competing figures) diversifies the search to avoid local minima.

#### 3.3.2. PWO-Lite algorithmic operations

*3.3.2.1. Divination-block casting (population initialization).* The process is divided into three main stages: divination-block casting, pilgrimage, and return ceremony. First is the divination-block casting stage, as shown in **Eq. (7)**, where $\mathbf{X}_i$ represents the chaotic logistic value associated with the position of the $i^{th}$ particle, symbolizing the devotees in the Matsu pilgrimage tradition. This value generates the initial population of particles $X_0$, which ranges between 0 and 1.

(1) Divination-block casting:

$$\mathbf{X}_{inew} = \eta \mathbf{X}_i(1 - \mathbf{X}_i), 0 \le \mathbf{X}_i \le 1 \tag{7}$$

In this context, η is a fixed parameter, typically set to 4.0.

*3.3.2.2. Pilgrimage phase (exploration and exploitation).* The pilgrimage stage combines two particle movement strategies: Matsu pilgrimage movement (**Eq. (8)**) and Matsu random movement (**Eq. (9)**). A predefined threshold of 0.5 governs the action selection process. During each iteration, a random number between 0 and 1 is generated. If the

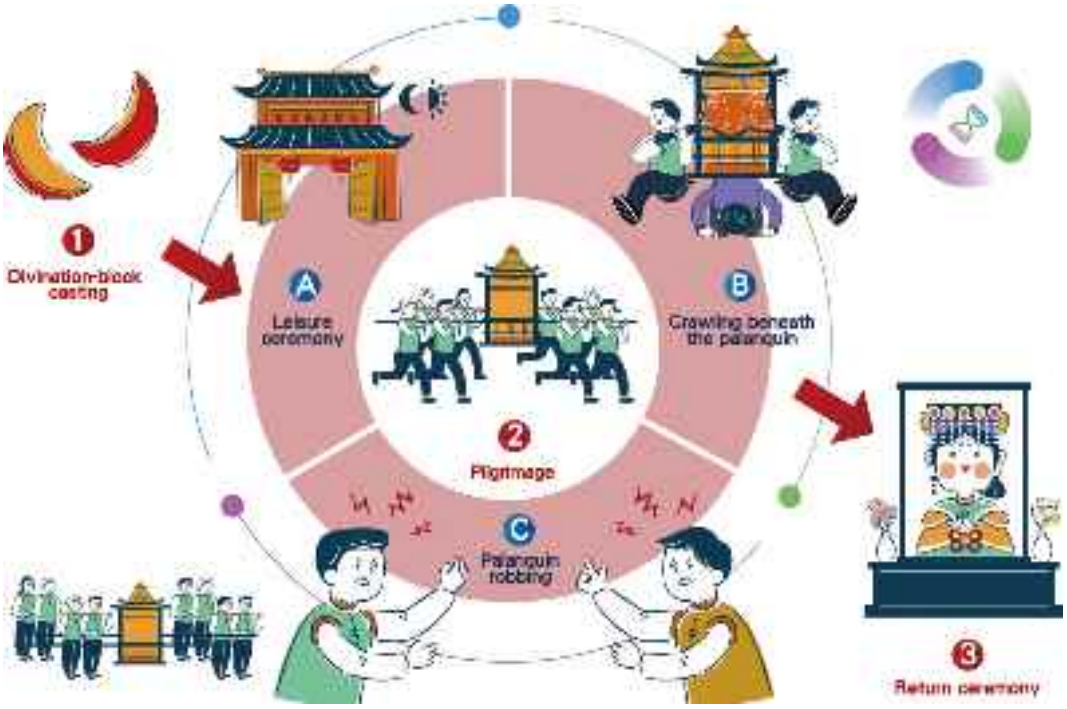


**Fig. 4.** Flowchart of the PWO-Lite algorithm.

```
Begin
  Define objective function f(X), X = (x1,…,xd)^T
  Set the search space, population size (NP), and maximum iteration (Max_iter)
  /* Divination-block casting */
  Initialize population of devotees X_i (i = 1, 2, ..., NP) using logistic chaotic map Eq. (7)
  Devotees (X_i) want to be as close to Matsu's palanquin (X^best) as possible, to increase the likelihood
  of receiving Matsu's blessing. Calculate the spiritual power of the devotee at each X_i, f(X_i)
  Repeat
    /* Pilgrimage phase*/
      For i=1:NP do
    If rand ≥ 0.5
      Matsu pilgrimage moves using Eq. (8)
    Else
      Matsu randomly moves using Eq. (9)
    End if
    Check boundary conditions, calculate f(X_i), and update X_i and X^best
    /* Bobee phase */
    Calculate the time governing equation c_i using Eq. (13)
    If c_i ≥ 0.5
      Crawling-beneath-the-palanquin event occurs using Eq. (11)
      Check boundary conditions, calculate f(X_i), and update X_i and X^best
    Else
      Leisure-ceremony event occurs using Eq. (10)
      Check boundary conditions, calculate f(X_i), and update X_i and X^best
      Crawling-beneath-the-palanquin event occurs using Eq. (11)
      Check boundary conditions, calculate f(X_i), and update X_i and X^best
      Palanquin-robbing event occurs using Eq. (12)
      Check boundary conditions, calculate f(X_i), and update X_i and X^best
    End if
    End for i
  /* Iterative search and return ceremony */
    Until maximum iteration (Max_iter) is met
    Output the best results
    End
```

**Fig. 5.** Pseudocode of PWO-Lite algorithm.

random number is greater than or equal to the threshold, the Matsu pilgrimage movement is executed; otherwise, the Matsu random movement is carried out.

The current location of Matsu's palanquin is identified as the optimal position $\mathbf{X}^{best}$, while $\mathbf{X}^{mean}$ represents the average position of all particles. As the Matsu pilgrimage procession, guided by Matsu, progresses toward the next temple, the exact route is not predetermined but is instead dictated by Matsu herself. The random movement strategy mirrors a Lévy flight, defined as a random walk approach where step lengths follow a Lévy distribution, as shown in **Eq. (9)**.

(2) Pilgrimage:

$$\mathbf{X}_{inew} = \mathbf{X}_i + \mathrm{rand}(0,1) \bullet \left(\mathbf{X}^{best} - \mathbf{X}_i\right) + \mathrm{rand}(0,1) \bullet \left(\mathbf{X}^{mean} - \mathbf{X}_i\right) \quad (8)$$

$$\mathbf{X}_{inew} = \mathbf{X}^{best} + (-1)^{U(1,2)} \bullet \mathrm{Lévy}(\delta) \quad (9)$$

Here, U(1,2) is a randomly generated 1 or 2, Ub and Lb represent the upper and lower bounds in the search space dimensions, and Lévy(δ) is a strategy for particle random walk. The algorithm can perform local and global fine-grained searches through step-length jumps. The parameter δ is used to adjust the scale of the Lévy distribution.

*3.3.2.3. Bobee phase (exploitation).* During the pilgrimage stage, three rituals are performed: the leisure ceremony, crawling beneath the palanquin, and palanquin robbing. The leisure ceremony stage, as shown in **Eq. (10)**, symbolizes when Matsu's palanquin sways back and forth in front of a temple, indicating her desire to stay overnight, where U(1,2) represents a randomly generated 1 or 2, and Ub and Lb are the upper and lower bounds in the search space dimensions. Thus, the oscillation of particles at the upper and lower boundaries is interpreted as the behavior of the leisure ceremony.

A. Leisure ceremony:

$$\mathbf{X}_{inew} = \mathbf{X}^{mean} + (-1)^{U(1,2)} \bullet \mathrm{rand}(0,1) \bullet (\mathrm{Ub} - \mathrm{Lb}) \quad (10)$$

Crawling beneath the palanquin stage, as shown in **Eq. (11)**, symbolizes the act of devotees kneeling in succession during the pilgrimage so that the palanquin can pass over them. $X_{r1}$ represents a randomly selected devotee, and {r1}∈{1,2, …,NP}, where r1 is randomly chosen. Consequently, the devotees gather closely, awaiting the passage of the palanquin.

B. Crawling beneath the palanquin:

$$\mathbf{X}_{\text{inew}} = \mathbf{X}_{\text{i}} + (2 \times \text{rand}(0,1) - 1) \bullet \left(\mathbf{X}^{\text{best}} - \mathbf{X}_{\text{i}}\right) + (2 \times \text{rand}(0,1) - 1) \bullet (\mathbf{X}_{\text{r1}} - \mathbf{X}_{\text{i}}) \tag{11}$$

The palanquin-robbing stage, represented by **Eq. (12)**, symbolizes the act of one group of bearers attempting to seize Matsu's palanquin from another group, diverting it from the intended pilgrimage route. The equation $\mathbf{X}_{\text{i}}^{\text{rob}}(t) = \mathbf{U}_{\text{b,d}} + \mathbf{L}_{\text{b,d}} - \mathbf{X}_{\text{i}}(t)$ was formulated to describe this process. This stage reflects the mirrored positions of particles in the search space. By diversifying the population positions, the algorithm increases the chances of a more comprehensive search, thereby enhancing convergence.

C. Palanquin robbing:

$$\mathbf{X}_{\text{inew}} = \mathbf{X}_{\text{i}}^{\text{rob}} + \text{rand}(0,1) \bullet \left(\mathbf{X}^{\text{best}} - \mathbf{X}_{\text{i}}^{\text{rob}}\right) + \text{rand}(0,1) \bullet \left(\mathbf{X}^{\text{mean}} - \mathbf{X}_{\text{i}}^{\text{rob}}\right) \tag{12}$$

*3.3.2.4. Time-governing mechanism.* During the initial stages of PWO-Lite, the movement strategies involve either pilgrimage processions or random movements, followed by the bobee phase: leisure ceremony, crawling beneath the palanquin, and robbing the palanquin. As PWO-Lite progresses, a movement primarily consists of pilgrimage processions or random movements coupled with crawling beneath the palanquin. A time-governing equation is used to convert these stages. The time-governing mechanism is defined as follows:

$$c_{\text{i}} = \left|\frac{t}{\text{Max}_{\text{iter}}} \times (3 \times \text{rand}(0,1) - 1)\right| \tag{13}$$

Where t is the time specified given by the iteration number, and $\text{Max}_{\text{iter}}$ is the maximum number of iterations, an initialized parameter.

*3.3.2.5. Boundary constraints.* The palanquin carriers for Matsu and devotees find themselves tightly packed into a confined space. Therefore, when particles move outside the search space, their values will be set to the upper and lower boundaries, as illustrated by the mathematical equation below.

$$\begin{cases} \mathbf{X}'_{\text{i,d}}(t) = \mathbf{U}_{\text{b,d}} \text{ if } \mathbf{X}_{\text{i,d}}(t) > \mathbf{U}_{\text{b,d}} \\ \mathbf{X}'_{\text{i,d}}(t) = \mathbf{L}_{\text{b,d}} \text{ if } \mathbf{X}_{\text{i,d}}(t) < \mathbf{L}_{\text{b,d}} \end{cases} \tag{14}$$

Where $\mathbf{X}_{\text{i,d}}(t)$ is the location of the $i^{\text{th}}$ particle in the $d^{\text{th}}$ dimension, $\mathbf{X}_{\text{i,d}}(t)$ is the updated location after checking boundary constraints.

*3.3.2.6. Iterative search and return ceremony (optimal solution obtained).* Finally, the return ceremony stage, as shown in **Eq. (15)**, symbolizes the end of the pilgrimage. This ritual metaphorically represents the algorithm's iterative pursuit of the optimal solution, concluding with reverence. When the predetermined stopping conditions are met, the optimization process ends, and the algorithm identifies the final optimal position $\mathbf{X}_{\text{opt}}$ as the best solution.

(3) Return ceremony:

$$\mathbf{X}_{\text{opt}} = \mathbf{X}^{\text{best}} \tag{15}$$

### 3.4. Particle swarm optimization

Particle Swarm Optimization (PSO) is a population-based stochastic optimization algorithm inspired by the social behavior of organisms like bird flocks or fish schools. Introduced by Kennedy and Eberhart in 1995 [51], PSO is widely used for its simplicity, efficiency, and adaptability [52]. It iteratively refines a set of candidate solutions (particles) that move through the search space based on their experiences and the swarm's collective knowledge. Thus, this study uses PSO as a baseline method to evaluate the effectiveness of the proposed PWO-Lite algorithm in hyperparameter tuning.

PSO operates on two key principles: social influence and cognitive learning. Each particle represents a potential solution, defined by its position $x_i(t)$ and velocity $v_i(t)$ in the search space. Particles are guided by their personal best position (pbest) and the swarm's global best position (gbest). The algorithm balances exploration and exploitation to converge toward an optimal solution [53].

The PSO process begins with initializing a swarm of particles with random positions and velocities. The fitness of each particle is evaluated using an objective function $f(x_i(t))$, and pbest and gbest are initialized. During each iteration, the velocity and position of each particle are updated using:

$$v_i(t+1) = w(t) \cdot v_i(t) + c_1 \cdot r_1 \cdot (\text{pbest}_i - x_i(t)) + c_2 \cdot r_2 \cdot (\text{gbest} - x_i(t)) \tag{16}$$

Here, w(t) is the inertia weight, which controls the particle's momentum. A higher inertia weight encourages search space exploration, while a lower value promotes the exploitation of known regions. The cognitive acceleration coefficient $c_1$ determines the influence of the particle's personal best position $\text{pbest}_i$, and the social acceleration coefficient $c_2$ determines the influence of the swarm's global best position gbest. The terms $r_1$ and $r_2$ are random numbers uniformly distributed in the range [0,1], introducing stochasticity to the search process. The updated velocity $v_i(t+1)$ is then used to compute the new position of the particle using the position update equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{17}$$

The inertia weight w is often designed to decrease linearly over time to facilitate a smooth transition from exploration to exploitation. This is achieved using the equation:

$$w(t) = w_{\text{max}} - \left(\frac{w_{\text{max}} - w_{\text{min}}}{t_{\text{max}}}\right) \cdot t \tag{18}$$

Here, $w_{\text{max}}$ is the maximum (initial) value of the inertia weight, $w_{\text{min}}$ is the minimum (final) value, t is the current iteration, and $t_{\text{max}}$ is the maximum number of iterations. This linear reduction ensures that the algorithm extensively explores the search space in the early stages and focuses on refining solutions later.

After updating positions, fitness is re-evaluated, and pbest and gbest are updated if better solutions are found. The process repeats until a stopping criterion is met.

### 3.5. Model evaluation

Model performance evaluation is fundamental in deep learning-based computer vision systems. This study adopts two principal evaluation metrics: (1) classification accuracy, measuring the proportion of correct predictions for the image classification task, and (2) mean average precision at 50 % Intersection over Union (IoU) threshold ($\text{mAP}_{50}$), the established benchmark for evaluating instance segmentation performance.

#### 3.5.1. Hold-out validation

Hold-out validation is a common technique for evaluating machine and deep learning models. The dataset is split into training, validation, and testing sets in an 8:1:1 ratio. Training data is used for model training, validation data monitors the training process to prevent overfitting, and testing data evaluates the model's generalization ability. This method ensures consistent class distribution across all sets.

#### 3.5.2. Confusion matrix

The confusion matrix evaluates classification model performance by comparing predicted results with actual outcomes on a test dataset. It visually displays prediction accuracy and misclassifications, providing insights into the model's behavior and guiding targeted optimizations.

The matrix is essential for assessing classification performance and visualizing differences between predicted and actual results.

The confusion matrix categorizes the comparison between predicted results and accurate labels into four distinct categories:

- **True Positive (TP):** The number of instances where the actual condition is positive (deteriorated), and the model correctly identifies it as positive (deteriorated).
- **True Negative (TN):** The number of instances where the actual condition is negative (non-deteriorated), and the model correctly identifies it as negative (non-deteriorated).
- **False Positive (FP):** The number of instances where the actual condition is negative (non-deteriorated), but the model incorrectly identifies it as positive (deteriorated).
- **False Negative (FN):** The number of instances where the actual condition is positive (deteriorated), but the model incorrectly identifies it as negative (non-deteriorated).

#### 3.5.3. Intersection-over-Union

IoU is a metric commonly used to evaluate segmentation tasks by measuring the overlap between the predicted and actual target regions. It is calculated by dividing the intersection area by the area of the union of these regions, as shown in **Eq. (19)**. In segmentation tasks, a higher IoU indicates greater accuracy. Typically, an IoU threshold is set between 0 and 1; if the IoU exceeds the threshold, the result is considered a TP; otherwise, it is an FP. This study follows the literature [54] by defining an IoU greater than 0.5 as a correct detection.

$$\text{Intersection over Union (IoU)} = \frac{\text{Area of predicted mask} \cap \text{Area of ground truth mask}}{\text{Area of predicted mask} \cup \text{Area of ground truth mask}} \tag{19}$$

#### 3.5.4. Average precision

Average Precision (AP) is a crucial evaluation metric in object detection and information retrieval, measuring a model's precision and recall at different thresholds. Precision, as shown in **Eq. (20)**, is the ratio of TP to the total number of instances predicted as positive (TP + FP). Recall, as shown in **Eq. (21)**, is the ratio of TP to the total number of actual positive instances (TP + FN).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{20}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{21}$$

AP is calculated from the area under the Precision-Recall ($P(\text{R})$) curve (**Eq. (22)**). When the IoU threshold is 0.5, it is referred to as $\text{AP}_{50}$. The $\text{mAP}_{50}$ is obtained by averaging $AP_i$ across all categories (**Eq. (23)**).

$$\text{AP}_{50} = \int_0^1 P(\text{R})\text{dR} \tag{22}$$

$$\text{mAP}_{50} = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{23}$$

Here, $P(\text{R})$ denotes precision as a function of recall, R represents recall with values ranging from 0 and 1, and $N$ indicates the total number of detection categories.

## 4. Model development and results

### 4.1. Bridge deterioration image data collection

The image classification and instance segmentation tasks utilized different datasets. However, both datasets were collected during the same period (from December 2022 to December 2023). This study used the Parrot Anafi UAV to capture images of bridge deterioration in urban and rural areas of western Taiwan. Images were collected from 13 composite bridges, 19 steel bridges, and 37 reinforced concrete bridges, as illustrated in Fig. 6. Following preprocessing; the dataset was divided into two subsets: 2000 images were used to fine-tune the ViT model for bridge component classification, and 1540 images were used to train the YOLOv8 model for instance segmentation of deterioration patterns. Table 1 provides further details on the dataset distribution.

Field comparisons and a literature review identified areas of deterioration, primarily in bridge decks, piers, abutments, and wing walls, focusing on concrete cracking, spalling, honeycombing, rebar corrosion, steel plate paint peeling, and bolt corrosion. The ViT model was specifically trained to recognize bridge structural components, including decks, piers, abutments, and wing walls, whereas YOLOv8 was optimized to detect deterioration patterns in these components. YOLOv8's hyperparameters were fine-tuned using PWO and PSO algorithms. After individual training and performance comparison, the models were integrated into a comprehensive bridge deterioration assessment system to estimate maintenance costs. Fig. 7 shows the bridge inspection framework using ViT and YOLOv8 models.

### 4.2. Data preprocessing

Preprocessing is crucial before model training, as it enhances performance, reduces overfitting, and ensures consistency. Initial data screening categorizes images into three groups: (a) no significant deterioration (Fig. 8a), (b) improper UAV operation (Fig. 8b), and (c) visible deterioration (Fig. 8c). The first group is used for training the ViT classification model, the second group is excluded from further processing, and the third group is employed in training both classification (ViT) and segmentation (YOLOv8) models.

The next step is to use LabelMe [55] for data annotation, which enables the manual annotation of objects within images. The annotated areas are saved in JSON format, which corresponds to the original images. Finally, the JSON files are converted to TXT for model training.

### 4.3. Validation of ViT model performance

This study utilizes the ViT model, which was trained with a batch size of 32 and optimized using the Adam optimizer with a learning rate of 0.001. The training performance is illustrated in Figs. 9 and 10. As shown in Fig. 9, the training loss steadily decreased across the epochs, starting from approximately 0.63 and converging to around 0.088. The test loss followed a similar downward trend, reaching 0.104 by the end of training, indicating good generalization performance without significant overfitting.

Fig. 10 presents the accuracy trends over ten epochs. The training accuracy improved consistently, starting at approximately 79 % and reaching a peak of 98.12 %. The test accuracy remained consistently high, reaching a maximum of 97.87 % and stabilizing at 96.63 %, confirming the model's robustness and effectiveness after training even without hyperparameter tuning.


**Fig. 6.** Bridge location distribution and UAV operation diagram.

**Table 1**
Bridge deterioration image statistics.

| Model | Component category | No. of pictures | Total |
|---|---|---|---|
| ViT | Bridge deck | 500 | 2000 |
| | Bridge pier | 500 | |
| | Abutment | 500 | |
| | Wing wall | 500 | |
| YOLOv8 | Concrete cracking | 220 | 1540 |
| | Exposed rebar | 220 | |
| | Concrete honeycombing | 220 | |
| | Steel plate paint peeling | 220 | |
| | Bolt corrosion | 220 | |
| | Steel plate corrosion | 220 | |
| | Concrete spalling | 220 | |

### 4.4. Validation of YOLOv8 model performance

The initial hyperparameter settings for YOLOv8 were based on the default values (i.e., Table 3) recommended in the official guidelines. A total of 23 hyperparameters were utilized, with a batch size of 32 consistently applied throughout the training process. The ranges of these hyperparameters are provided in **Table A1 of Appendix A**. The training results for the YOLOv8 instance segmentation model are presented in Table 2, showing overall $mAP_{50}$ values of 62.0 % for the validation set and 64.5 % for the test set.

The analysis indicates that the model effectively identifies defects such as steel plate paint peeling, bolt corrosion, exposed rebar, and steel plate corrosion. However, it struggles to accurately delineating the boundaries of deterioration types like concrete cracking, honeycombing, and spalling. These challenges may stem from the visual similarity among concrete-related defects, making a precise distinction more difficult. In contrast, steel bridges often feature back plates painted in distinct colors, which may help the model differentiate between background and deterioration patterns during instance segmentation. Overall, the model's hyperparameters require further fine-tuning to enhance performance.

### 4.5. YOLOv8 model hyperparameter tuning

In this study, the hyperparameters of the YOLOv8 model were optimized using the PSO and PWO-Lite metaheuristic algorithms. Both PSO and PWO-Lite were applied iteratively to optimize the hyperparameters, and the results are summarized in Table 3, which includes details on learning rates, data augmentation settings, and other key parameters.

After hyperparameter tuning, the instance segmentation model optimized with PWO-Lite, referred to as PWO-Lite-YOLOv8, outperformed both the PSO-optimized model (PSO-YOLOv8) and the baseline YOLOv8 model across multiple evaluation metrics. Specifically, PWO-Lite-YOLOv8 achieved a 2.4 % improvement in overall prediction performance on the validation set and a 4.5 % improvement on the test set compared to the baseline YOLOv8 model. In contrast, PSO-YOLOv8 showed a 1.2 % improvement on the validation set and a 1.6 % improvement on the test set. The $mAP_{50}$ also showed more remarkable enhancements with PWO-Lite across all deterioration categories. For instance, the "steel plate corrosion" category saw the most significant increase, rising from 68.5 % to 75.6 % with PWO-Lite, whereas PSO achieved only a marginal improvement to 71.5 %. These results are detailed in Tables 2, 4, and 5.

The outstanding performance of PWO-Lite can be attributed to its enhanced capability to systematically and comprehensively explore the hyperparameter space, effectively mitigating the risk of being trapped in local optima while accelerating convergence toward the global optimum. This advantage renders PWO-Lite well-suited to address complex optimization challenges, such as deep learning hyperparameter tuning. Conversely, although PSO has demonstrated effectiveness, it often encounters difficulties navigating high-dimensional search spaces and is prone to premature convergence to suboptimal solutions. The comparative analysis presented in Tables 4 and 5 unequivocally indicates that PWO-Lite is more reliable and efficient for optimizing YOLOv8 hyperparameters.

The PWO-Lite-YOLOv8 model effectively detects most major corrosion areas on steel plates, as indicated by the orange segmentation regions in Fig. 11. However, it struggles to identify finer and more subtle corrosion spots, likely due to low contrast between the rust and the underlying metal surface, or the model's tendency to lose small-scale

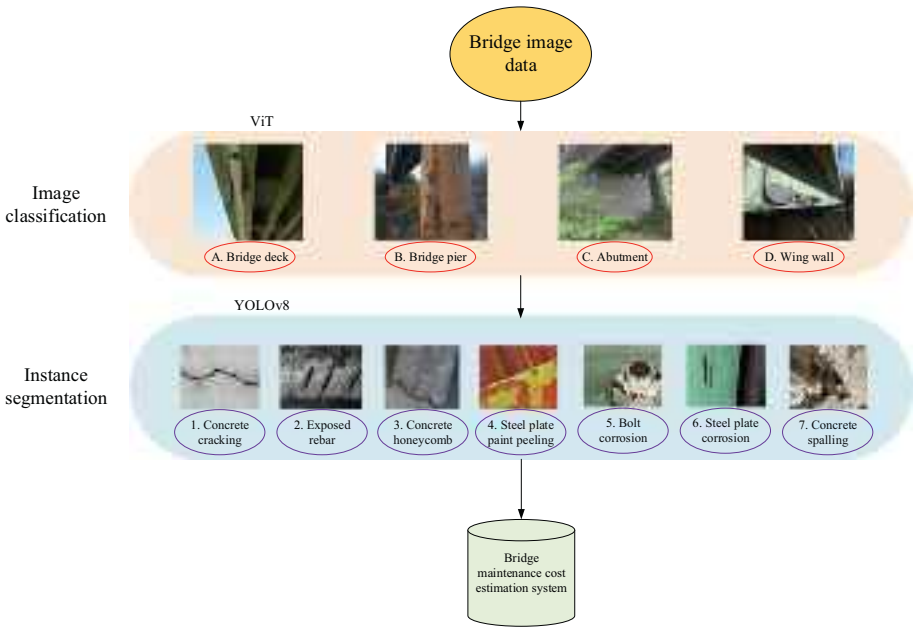


**Fig. 7.** UAV-based bridge inspection framework using ViT and YOLOv8 models.

(a) No noticeable deterioration. (b) Improper operation. (c) Visible deterioration.

**Fig. 8.** Examples of collected bridge images: (a) no noticeable deterioration, (b) improper UAV operation, and (c) visible deterioration.

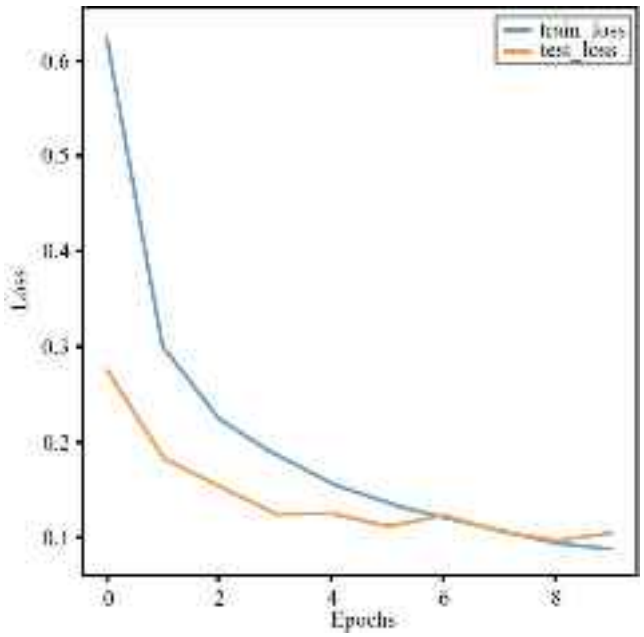


**Fig. 9.** Training and testing loss curves of the ViT model across epochs.

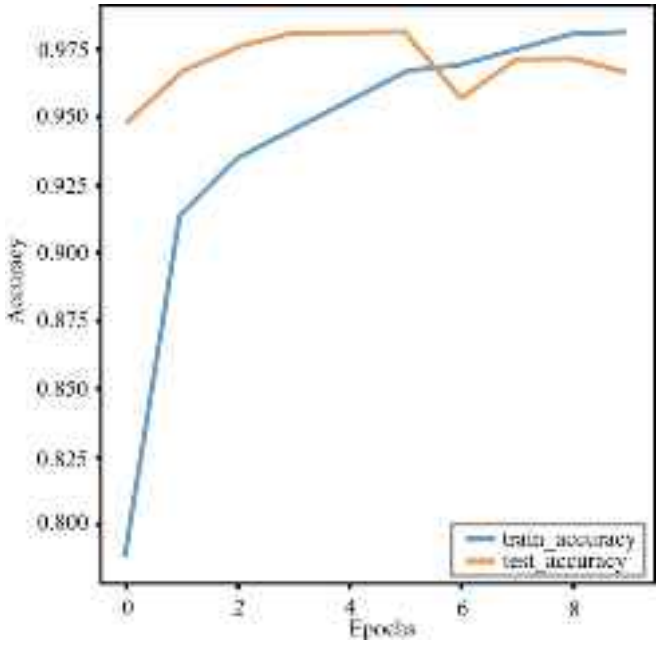


**Fig. 10.** Training and testing accuracy curves of the ViT model across epochs.

**Table 2**
YOLOv8 performance evaluation using default settings.

| Class | Validation set | | Testing set | |
|---|---|---|---|---|
| | $mAP_{50}$ (%) | Rank | $mAP_{50}$ (%) | Rank |
| Overall average | **62.0** | – | **64.5** | – |
| Concrete cracking | 60.0 | 5 | 64.4 | 5 |
| Exposed rebar | 65.9 | 4 | 77.7 | 3 |
| Concrete honeycombing | 35.8 | 7 | 41.2 | 6 |
| Steel plate paint peeling | 83.6 | 1 | 83.8 | 2 |
| Bolt corrosion | 78.8 | 2 | 84.5 | 1 |
| Steel plate corrosion | 70.0 | 3 | 68.5 | 4 |
| Concrete spalling | 40.2 | 6 | 31.6 | 7 |

**Table 3**
Default hyperparameters and optimized model settings for YOLOv8.

| No. | Hyper-parameter | Default value | PSO-optimized model parameter | PWO-optimized model parameter |
|---|---|---|---|---|
| 1 | lr0 | 0.01 | 0.0721 | 0.0685 |
| 2 | lrf | 0.01 | 0.00702 | 0.00651 |
| 3 | momentum | 0.937 | 0.89841 | 0.89332 |
| 4 | weight_decay | 0.0005 | 0.00067 | 0.00065 |
| 5 | warmup_epochs | 3.0 | 3.08 | 3.0 |
| 6 | warmup_momentum | 0.8 | 0.7284 | 0.7225 |
| 7 | box | 7.5 | 9.36180 | 9.35796 |
| 8 | cls | 0.5 | 0.5369 | 0.5427 |
| 9 | dfl | 1.5 | 1.32845 | 1.32091 |
| 10 | hsv_h | 0.015 | 0.02135 | 0.01983 |
| 11 | hsv_s | 0.7 | 0.86472 | 0.85807 |
| 12 | hsv_v | 0.4 | 0.44983 | 0.44109 |
| 13 | degrees | 0.0 | 0.0 | 0.0 |
| 14 | translate | 0.1 | 0.13602 | 0.12991 |
| 15 | scale | 0.5 | 0.35842 | 0.35057 |
| 16 | shear | 0.0 | 0.0 | 0.0 |
| 17 | perspective | 0.0 | 0.0 | 0.0 |
| 18 | flipud | 0.0 | 0.0 | 0.0 |
| 19 | fliplr | 0.5 | 0.52638 | 0.51839 |
| 20 | bgr | 0.0 | 0.0 | 0.0 |
| 21 | mosaic | 1.0 | 0.83091 | 0.82246 |
| 22 | mixup | 0.0 | 0.0 | 0.0 |
| 23 | copy_paste | 0.0 | 0.0 | 0.0 |

**Table 4**
YOLOv8 performance evaluation with PSO-tuned hyperparameters.

| Class | Validation set | | Testing set | |
|---|---|---|---|---|
| | $mAP_{50}$ (%) | Rank | $mAP_{50}$ (%) | Rank |
| Overall average | **63.2** | – | **66.1** | – |
| Concrete cracking | 63.8 | 5 | 67.3 | 5 |
| Exposed rebar | 70.9 | 3 | 71.3 | 4 |
| Concrete honeycombing | 38.4 | 6 | 42.3 | 6 |
| Steel plate paint peeling | 83.0 | 1 | 78.8 | 1 |
| Bolt corrosion | 81.7 | 2 | 73.8 | 2 |
| Steel plate corrosion | 68.6 | 4 | 71.5 | 3 |
| Concrete spalling | 35.7 | 7 | 34.2 | 7 |

**Table 5**
YOLOv8 performance evaluation with PWO-Lite-tuned hyperparameters.

| Class | Validation set | | Testing set | |
|---|---|---|---|---|
| | $mAP_{50}$ (%) | Rank | $mAP_{50}$ (%) | Rank |
| Overall average | **64.4** | – | **69.0** | – |
| Concrete cracking | 67.5 | 4 | 70.2 | 5 |
| Exposed rebar | 75.8 | 3 | 82.6 | 3 |
| Concrete honeycombing | 40.9 | 6 | 43.4 | 6 |
| Steel plate paint peeling | 83.6 | 2 | 88.2 | 1 |
| Bolt corrosion | 84.5 | 1 | 85.9 | 2 |
| Steel plate corrosion | 67.3 | 5 | 75.6 | 4 |
| Concrete spalling | 31.1 | 7 | 36.8 | 7 |

details during downsampling. Additionally, uneven rust patterns and data imbalance may contribute to missed detections or fragmented segmentation.

In the case of concrete honeycomb defects, the model accurately identifies the primary defective area (marked by a pink mask in Fig. 11) but struggles to achieve precise boundary segmentation. This results in blurred transitions between honeycomb defects and intact concrete, potentially due to irregular surface patterns, inconsistent lighting conditions, or limitations in the model's ability to preserve edge details.

Through this research, we observed that the model's performance may vary depending on the material composition of the bridge (e.g., steel vs. concrete) and environmental factors such as lighting, weather conditions, and surface degradation patterns. These factors can influence the visibility of defects and the model's ability to generalize across diverse scenarios. Future work could focus on improving the model's robustness to these variations.

## 5. Bridge maintenance cost estimation system

The bridge maintenance cost system relies on accurately detecting and assessing component deterioration to develop maintenance plans and budgets. This study leverages drone technology and deep learning models—ViT and PWO-Lite-YOLOv8—to analyze bridge components and deterioration images, improving detection accuracy and estimating maintenance costs. This section presents an overview of the system's development.

### *5.1. Automated deterioration area estimation and maintenance cost calculation*

The bridge deterioration recognition model developed in this study effectively distinguishes between major bridge components and identifies specific areas of deterioration. The model accurately calculates the pixel count of the target in the image by predicting the x- and y-coordinates of the polygonal frame, thereby determining the area of deterioration within the image.

This study assumes that, due to the relatively large distance between the drone and the bridge, the captured image can be approximated as a top-down (orthographic) view, minimizing perspective distortion. Based on this assumption, the deteriorated area visible in the picture is considered proportional to the actual maintenance area, as illustrated in Fig. 12. Once the deteriorated area is converted into square meters—using known camera parameters and flight altitude—and multiplied by the corresponding maintenance unit price, the total repair cost can be estimated.

This method quickly provides accurate information on deteriorated areas, influencing maintenance decisions and cost estimation. It is a reliable and efficient tool for bridge condition monitoring and routine maintenance, enabling engineers to perform inspections more efficiently and helping maintenance management units assess deterioration levels and develop maintenance plans.

### *5.2. System interface design and development*

The system developed in this study uses Tkinter, a graphical user interface (GUI) toolkit based on Tool Command Language (Tcl) and provided through Python encapsulation [56]. The GUI interface offers a user-friendly method for loading and processing images and viewing the model's predictions for instance segmentation, including deterioration image recognition, region delineation, and coordinate area calculation. This approach enhances users' efficiency and convenience, enabling non-experts to easily apply the YOLOv8 model for estimating maintenance costs in bridge inspection tasks.

#### *5.2.1. System platform operation instructions*

The system interface and corresponding operation steps are


**Fig. 11.** Examples of segmentation challenges: missed corrosion spots on steel surfaces and imprecise boundary detection in concrete honeycombing.


**Fig. 12.** Deterioration area calculation.

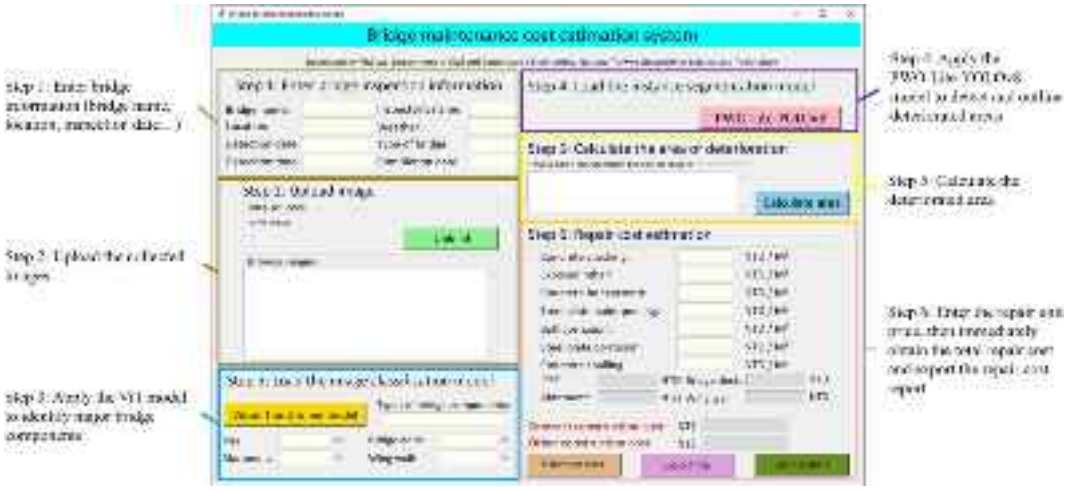


**Fig. 13.** Bridge deterioration maintenance cost estimation system interface.

presented in Fig. 13, as follows:

**Step 1:** Enter the basic information of the bridge, including the bridge name, location, inspection date, inspection time, inspector's name, weather conditions, bridge type, and completion date.

**Step 2:** Upload the deterioration images of the bridge collected by the unmanned aerial vehicle (UAV).

**Step 3:** Use the ViT image classification model to identify the significant bridge components and count the number of images.

**Step 4:** Utilize the PWO-Lite-YOLOv8 instance segmentation model to predict the detailed areas of deterioration on the bridge.

**Step 5:** Calculate the area of deterioration within the bridge images.

**Step 6:** Input the required unit prices for the repair of each deterioration category. Upon completion, click the "Estimate cost" button to instantly generate the total maintenance cost and the cost for each component. Finally, click the "Export file" button to export the bridge maintenance cost report.

### 5.2.2. *Bridge maintenance cost report design*

This study uses the Dongshi Bridge in Chiayi County as a demonstration example to generate a bridge maintenance cost report, detailed in **Table A2** of **Appendix A**. The estimated cost for bridge maintenance is 719,877 NTD, with 93,400 NTD allocated for the repair of the bridge back wall and 341,900 NTD for the piers. The report distinguishes between contract work costs and other construction costs. The detailed bridge work items under contract costs are based on the seven types of identified bridge deterioration. Miscellaneous work and other construction costs are priced according to the "Manual for Estimating Public Construction Project Costs," compiled by the Executive Yuan Public Construction Commission. The essential maintenance cost is calculated by multiplying the quantity column by the unit price column in the

**Table 6**
Detailed bridge component maintenance items.

| A. Contracted Engineering Costs | | | | |
|---|---|---|---|---|
| a. Bridge Maintenance Details | | | | |
| 1. Bridge Deck | | | | |
| | Unit | Quantity | Unit Price | Price |
| Rebar | $M^2$ | 372 | | 93,000 |
| Rebar Exposure Rust Removal and Repair | $M^2$ | 372 | 250 | 93,000 |
| Steel Plates | $M^2$ | 2 | | 400 |
| Steel Plate Rust Removal and Repainting | $M^2$ | 0 | 200 | 0 |
| Steel Plate Paint Removal and Repainting | $M^2$ | 2 | 200 | 400 |
| Subtotal (NTD) | 93,400 | | | |
| 2. Bridge Pier | | | | |
| | Unit | Quantity | Unit Price | Price |
| Concrete Components | $M^2$ | 733 | | 341,900 |
| Concrete Crack Coating | $M^2$ | 295 | 300 | 88,500 |
| Concrete Honeycomb Repair | $M^2$ | 266 | 500 | 133,000 |
| Concrete Crack and Spalling Repair | $M^2$ | 172 | 700 | 120,400 |
| Subtotal (NTD) | 341,900 | | | |
| Total (NTD) | 435,300 | | | |

table.

To ensure flexibility and practical use across various real-world settings, the system features a function that enables contractors to enter their unit prices. This customization accommodates regional differences in labor and material costs, making the report more adaptable for diverse project conditions. Additionally, Table 6 presents a detailed breakdown of maintenance items and corresponding costs for each bridge component, offering users a clear and organized view that facilitates easy review, modification, and verification.

Automating the bridge maintenance cost estimation process offers substantial benefits for infrastructure management. By generating reports through an automated system, agencies gain accurate and comprehensive cost projections while streamlining the estimation workflow. The resulting data-driven insights enable evidence-based decision-making, allowing transportation departments to optimize resource allocation and prioritize maintenance activities more effectively.

## 6. Conclusions

Bridges are vital for national development, especially in regions like Taiwan, where challenging terrain and frequent natural disasters, such as typhoons and earthquakes, demand resilient infrastructure. Robust bridge structures are essential for maintaining transportation networks, enabling rapid disaster response, and minimizing economic and human losses. However, as bridges age, timely and effective maintenance becomes increasingly urgent to extend their lifespan and reduce operational costs. Traditional inspection methods, often manual and labor-intensive, struggle to address the complexities of modern infrastructure, particularly in hard-to-reach areas like beneath bridge decks and substructures.

This paper introduced a system that combines drone technology with advanced deep-learning models for bridge inspection. Unlike traditional methods, our approach uses drones to access difficult areas, ensuring comprehensive coverage while minimizing safety risks and labor costs. Integrating ViT for image classification and YOLOv8 for segmentation enhances the automated and precise analysis of bridge conditions, significantly improving inspection efficiency and accuracy. A key contribution is our bridge component deterioration maintenance cost system, which automates cost estimation and report generation, addressing a critical gap in existing manual and subjective assessment methods.

Our approach utilizes state-of-the-art deep learning models enhanced by the PWO-Lite algorithm for hyperparameter tuning and a multi-stage inspection system. These innovations have led to significant improvements in model performance. For example, the ViT model achieved an average accuracy of 96.63 %. In comparison, the YOLOv8 model demonstrated a 4.5 % performance improvement in $mAP_{50}$ after hyperparameter optimization—a considerable gain in deep learning, where even marginal improvements can be challenging. The proposed system automates the inspection workflow, from data collection using drones to deterioration analysis and cost estimation. This reduces the time and labor required for inspections, offering a more efficient alternative to traditional methods. Maintenance units can now quickly generate detailed reports and cost estimates, enabling faster decision-making and resource deployment.

Using drones eliminates the need for inspectors to access hazardous or hard-to-reach areas, improving safety and reducing risks. Integrating advanced deep learning models and optimization techniques ensures higher accuracy in identifying and classifying deterioration patterns than traditional visual inspections. The automated workflow also reduces human error and speeds up the process. The system reduces operational expenses and improves budget planning efficiency by enabling quick cost estimation and report generation, offering a clear advantage over manual and semi-automated methods. Additionally, the framework is scalable and adaptable, making it suitable for inspecting other critical bridge components or different types of infrastructure.

While the results are promising, instance segmentation performance can still be improved. Future research could broaden the inspection scope to include areas beneath bridge decks and substructures, applying the framework to other key bridge components. Optimization experiments could refine the model's architecture, potentially leading to a high-performance, multi-task recognition model that accurately differentiates between bridge components and identifies areas of deterioration. Studies could also develop a model to distinguish honeycombing from cracking in concrete bridges. Real-time inspection on drones or edge devices may face power and computational limits, so optimizing the model through pruning or quantization will be key for deployment.

## CRediT authorship contribution statement

**Jui-Sheng Chou:** Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization, Writing – review & editing, Writing – original draft. **Jhe-Shian Lien:** Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Writing – original draft. **Chi-Yun Liu:** Visualization, Validation, Software, Methodology, Data curation, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. Acknowledgments: The authors would like to thank the National Science and Technology Council (NSTC), Taiwan, for financially supporting this research under grant numbers NSTC 110-2221-E-011-080-MY3 and 113-2221-E-011-050-MY3.

## Acknowledgments

The authors would like to thank the National Science and Technology Council (NSTC), Taiwan, for financially supporting this research under grant numbers NSTC 110-2221-E-011-080-MY3 and 113-2221-E-011-050-MY3.

## Appendix A

**Table A.1**
PWO-Lite hyperparameter search range for YOLOv8.

| No. | Hyper-parameter | Lower limit | Upper limit | Description |
|---|---|---|---|---|
| 1 | lr0 | 1.00E-0.5 | 1.00E-0.1 | initial learning rate (SGD = 1E-2, Adam = 1E-3) |
| 2 | lrf | 0.01 | 1.0 | final OneCycleLR learning rate (lr0 × lrf) |
| 3 | momentum | 0.6 | 0.98 | SGD momentum/Adam beta1 |
| 4 | weight_decay | 0.0 | 0.001 | optimizer weight decay |
| 5 | warmup_epochs | 0.0 | 5.0 | warmup epochs (fractions ok) |
| 6 | warmup_momentum | 0.0 | 0.95 | warmup initial momentum |
| 7 | box | 0.02 | 0.2 | box loss gain |
| 8 | cls | 0.2 | 4.0 | cls loss gain |
| 9 | dfl | 0.0 | 1.0 | dynamic fitness landscape |
| 10 | hsv_h | 0.0 | 0.1 | image HSV-Hue augmentation |
| 11 | hsv_s | 0.0 | 0.9 | image HSV-Saturation augmentation (fraction) |
| 12 | hsv_v | 0.0 | 0.9 | image HSV-Value augmentation (fraction) |
| 13 | degrees | 0.0 | 45.0 | image rotation (+/− deg) |
| 14 | translate | 0.0 | 0.9 | image translation (+/− fraction) |
| 15 | scale | 0.0 | 0.9 | image scale (+/− gain) |
| 16 | shear | 0.0 | 10.0 | image shear (+/− deg) |
| 17 | perspective | 0.0 | 0.001 | image perspective (+/− fraction), range 0–0.001 |
| 18 | flipud | 0.0 | 1.0 | image flip up-down (probability) |
| 19 | fliplr | 0.0 | 1.0 | image flip left-right (probability) |
| 20 | bgr | 0.0 | 0.1 | batch gradient regularization |
| 21 | mosaic | 0.0 | 1.0 | image mosaic (probability) |
| 22 | mixup | 0.0 | 1.0 | image mixup (probability) |
| 23 | copy_paste | 0.0 | 1.0 | image copy-paste (probability) |

**Table A.2**
Bridge maintenance cost report.

National Taiwan University of Science and Technology
Department of Construction Engineering - Project Intelligence and Management Lab
Bridge Maintenance Cost Report

| Bridge name | Dongshi Bridge | Inspector | Jhe-Shian Lien | | | |
|---|---|---|---|---|---|---|
| Location | Chiayi County | Weather | Partly Cloudy | | | |
| Inspection Date | 10/24/2021 | Bridge Type | Concrete Bridge | | | |
| Inspection Time | 09:06 AM | Completion Date | 08/2022 | | | |
| Bridge Component | ■Bridge Deck □Bridge Pier □Abutment □Wing wall | | | | | |
| No. | Project Item | Unit | Quantity | Unit Price | Price | Notes |
| A | Contracted Engineering Costs | $M^2$ | | | 571,331 | |
| a | Bridge Engineering | $M^2$ | | | 435,300 | |
| 1 | Concrete Components | $M^2$ | 733 | | 341,900 | |
| | Concrete Crack Coating | $M^2$ | 295 | 300 | 88,500 | |
| | Concrete Honeycomb Repair | $M^2$ | 266 | 500 | 133,000 | |
| | Concrete Crack and Spalling Repair | $M^2$ | 172 | 700 | 120,400 | |
| 2 | Rebar | $M^2$ | 372 | | 93,400 | |
| | Rebar Exposure Rust Removal and Repair | $M^2$ | 22 | 1200 | 26,400 | |
| 3 | Steel Plates | $M^2$ | 2 | | 400 | |
| | Steel Plate Rust Removal and Repainting | $M^2$ | 0 | 200 | 0 | |
| | Steel Plate Paint Removal and Repainting | $M^2$ | 2 | 200 | 400 | |
| 4 | Bolts | $M^2$ | 0 | | 0 | |
| | Bolt Rust Removal and Repainting | $M^2$ | 0 | 100 | 0 | |
| b | Miscellaneous Engineering (25 % of a) | | 1 | | 108,825 | |
| c | Labor Safety and Health Equipment (5 % of item a) | | 1 | | 27,206 | |
| B | Other Construction Costs | | 1 | | 148,546 | |
| a | Air Pollution Control Fee {(Items a to c) * 8 %} | | 1 | | 45,707 | |
| b | Construction Management Fee {(Items a to c) * 10 %} | | 1 | | 57,133 | |
| c | Temporary Electricity Cost {(Items a to c) * 8 %} | | 1 | | 45,707 | |
| Total (Contracted Engineering Costs and Other Construction Costs) | | | | | 719,877 | |

## Data availability

Data will be made available on request.

## References

[1] S. Li, M. Dang, Y. Xu, A. Wang, Y. Guo, Bridge damage description using adaptive attention-based image captioning, Autom. Constr. 165 (2024) 105525, https://doi.org/10.1016/j.autcon.2024.105525.
[2] Y. Gao, G. Xiong, H. Li, J. Richards, Exploring bridge maintenance knowledge graph by leveraging GrapshSAGE and text encoding, Autom. Constr. 166 (2024) 105634, https://doi.org/10.1016/j.autcon.2024.105634.

[3] J.-H. Chen, M.-C. Su, S.-K. Lin, W.-J. Lin, M. Gheisari, Smart bridge maintenance using cluster merging algorithm based on self-organizing map optimization, Autom. Constr. 152 (2023) 104913, https://doi.org/10.1016/j.autcon.2023.104913.

[4] M.-Y. Cheng, A.F.K. Khitam, Y.-B. Kueh, Risk score inference for bridge maintenance projects using genetic fuzzy weighted pyramid operation tree, Autom. Constr. 165 (2024) 105488, https://doi.org/10.1016/j.autcon.2024.105488.

[5] Y.-C. Sung, C.-C. Chen, H.-H. Hung, K.-C. Chang, Z.-K. Lee, J.-G. Su, Bridge monitoring and structural health diagnosis, Chin. Inst. Civ. Hydraul. Eng. 45 (5) (2018) 22–29. http://www.ciche.org.tw/wordpress/?page_id=8000.

[6] M. Alsharqawi, T. Zayed, S.A. Dabous, Common practices in assessing conditions of concrete bridges, in: MATEC Web of Conferences vol. 120, EDP Sciences, 2017, p. 02016, https://doi.org/10.1051/matecconf/201712002016.

[7] F. Brighenti, V.F. Caspani, G. Costa, P.F. Giordano, M.P. Limongelli, D. Zonta, Bridge management systems: a review on current practice in a digitizing world, Eng. Struct. 321 (2024) 118971, https://doi.org/10.1016/j.engstruct.2024.118971.

[8] C. Wan, Z. Zhou, S. Li, Y. Ding, Z. Xu, Z. Yang, Y. Xia, F. Yin, Development of a bridge management system based on the building information modeling technology, Sustainability 11 (17) (2019) 4583, https://doi.org/10.3390/su11174583.

[9] J. Yang, F. Xiang, R. Li, L. Zhang, X. Yang, S. Jiang, H. Zhang, D. Wang, X. Liu, Intelligent bridge management via big data knowledge engineering, Autom. Constr. 135 (2022) 104118, https://doi.org/10.1016/j.autcon.2021.104118.

[10] J. Lee, K. Sanmugarasa, M. Blumenstein, Y.-C. Loo, Improving the reliability of a bridge management system (BMS) using an ANN-based backward prediction model (BPM), Autom. Constr. 17 (6) (2008) 758–772, https://doi.org/10.1016/j.autcon.2008.02.008.

[11] S. Kang, S. Kim, S. Kim, Improvement of the defect inspection process of deteriorated buildings with scan to BIM and image-based automatic defect classification, J. Build. Eng. 99 (2025) 111601, https://doi.org/10.1016/j.jobe.2024.111601.

[12] M. Mandirola, C. Casarotti, S. Peloso, I. Lanese, E. Brunesi, I. Senaldi, Use of UAS for damage inspection and assessment of bridge infrastructures, Int. J. Disaster Risk Red. 72 (2022) 102824, https://doi.org/10.1016/j.ijdrr.2022.102824.

[13] H. Song, X. Zhu, H. Li, G. Yang, Multimodal deep learning-based automatic generation of repair proposals for steel bridge shallow damage, Autom. Constr. 171 (2025) 105961, https://doi.org/10.1016/j.autcon.2025.105961.

[14] R. Hou, Y. Xia, Review on the new development of vibration-based damage identification for civil engineering structures: 2010–2019, J. Sound Vib. 491 (2021) 115741, https://doi.org/10.1016/j.jsv.2020.115741.

[15] T. Panigati, M. Zini, D. Striccoli, P.F. Giordano, D. Tonelli, M.P. Limongelli, D. Zonta, Drone-based bridge inspections: current practices and future directions, Autom. Constr. 173 (2025) 106101, https://doi.org/10.1016/j.autcon.2025.106101.

[16] X. Huang, Y. Liu, L. Huang, S. Stikbakke, E. Onstein, BIM-supported drone path planning for building exterior surface inspection, Comput. Ind. 153 (2023) 104019, https://doi.org/10.1016/j.compind.2023.104019.

[17] N. Ejaz, S. Choudhury, Computer vision in drone imagery for infrastructure management, Autom. Constr. 163 (2024) 105418, https://doi.org/10.1016/j.autcon.2024.105418.

[18] Z. Fei, G.M. West, P. Murray, G. Dobie, CNN-based automated approach to crack-feature detection in steam cycle components, Int. J. Press. Vessel. Pip. 207 (2024) 105112, https://doi.org/10.1016/j.ijpvp.2023.105112.

[19] A. Mayya, N.F. Alkayem, L. Shen, X. Zhang, R. Fu, Q. Wang, M. Cao, Efficient hybrid ensembles of CNNs and transfer learning models for bridge deck image-based crack detection, Structures 64 (2024) 106538, https://doi.org/10.1016/j.istruc.2024.106538.

[20] O. Elharrouss, Y. Himeur, Y. Mahmood, S. Alrabaee, A. Ouamane, F. Bensaali, Y. Bechqito, A. Chouchane, ViTs as backbones: leveraging vision transformers for feature extraction, Inform. Fus. 118 (2025) 102951, https://doi.org/10.1016/j.inffus.2025.102951.

[21] S. Usmani, S. Kumar, D. Sadhya, Spatio-temporal knowledge distilled video vision transformer (STKD-VViT) for multimodal deepfake detection, Neurocomputing 620 (2025) 129256, https://doi.org/10.1016/j.neucom.2024.129256.

[22] L. Huang, J. Zeng, M. Yu, W. Ding, X. Bai, K. Wang, Efficient feature selection for pre-trained vision transformers, Comput. Vis. Image Underst. 254 (2025) 104326, https://doi.org/10.1016/j.cviu.2025.104326.

[23] F. Pan, J. Li, Y. Yan, S. Guan, B. Biswal, Y. Zhao, Enhanced surface defect detection of cylinder liners using Swin transformer and YOLOv8, J. Automat. Intell. (2025), https://doi.org/10.1016/j.jai.2025.01.004.

[24] Y. Zhou, X. Wu, Y. Li, H. Sun, D. Fan, Algorithm for surface flow velocity measurement in trunk canal based on improved YOLOv8 and DeepSORT, Eng. Appl. Artif. Intell. 148 (2025) 110344, https://doi.org/10.1016/j.engappai.2025.110344.

[25] M.Z. Li, Z.T. Yan, X.G. Yang, S. Zhao, Structural displacement monitoring via improved YOLOv8 structure under complex scenarios, Structures 73 (2025) 108302, https://doi.org/10.1016/j.istruc.2025.108302.

[26] J. Kim, S. Chi, Graph neural network-based propagation effects modeling for detecting visual relationships among construction resources, Autom. Constr. 141 (2022) 104443, https://doi.org/10.1016/j.autcon.2022.104443.

[27] H. Son, C. Kim, Integrated worker detection and tracking for the safe operation of construction machinery, Autom. Constr. 126 (2021) 103670, https://doi.org/10.1016/j.autcon.2021.103670.

[28] X. Wang, Q. Yue, X. Liu, Crack image classification and information extraction in steel bridges using multimodal large language models, Autom. Constr. 171 (2025) 105995, https://doi.org/10.1016/j.autcon.2025.105995.

[29] Y. Gao, H. Li, W. Fu, Few-shot learning for image-based bridge damage detection, Eng. Appl. Artif. Intell. 126 (2023) 107078, https://doi.org/10.1016/j.engappai.2023.107078.

[30] R. Raushan, V. Singhal, R.K. Jha, Damage detection in concrete structures with multi-feature backgrounds using the YOLO network family, Autom. Constr. 170 (2025) 105887, https://doi.org/10.1016/j.autcon.2024.105887.

[31] J. Zhang, J. Li, R. Ly, Y. Liu, J. Shu, Deep Learning-Based Fatigue Cracks Detection in Bridge Girders using Feature Pyramid Networks, arXiv preprint arXiv: 2410.21175, 2024.

[32] H. Zhang, Z. Shen, Z. Lin, L. Quan, L. Sun, Deep learning-based automatic classification of three-level surface information in bridge inspection, Comput. Aided Civ. Inf. Eng. 39 (10) (2024) 1431–1451, https://doi.org/10.1111/mice.13117.

[33] J. Zhu, J. Song, An intelligent classification model for surface defects on cement concrete bridges, Appl. Sci. 10 (3) (2020) 972, https://doi.org/10.3390/app10030972.

[34] X. Xiao, Q. Li, Two-stage deterioration model updating of RC structures in marine environment using long-term field inspection data, Constr. Build. Mater. 400 (2023) 132817, https://doi.org/10.1016/j.conbuildmat.2023.132817.

[35] M.M. Hossain, M.M. Ahmed, A.A.N. Nafi, M.R. Islam, M.S. Ali, J. Haque, M. S. Miah, M.M. Rahman, M.K. Islam, A novel hybrid ViT-LSTM model with explainable AI for brain stroke detection and classification in CT images: a case study of Rajshahi region, Comput. Biol. Med. 186 (2025) 109711, https://doi.org/10.1016/j.compbiomed.2025.109711.

[36] D. Tian, Y. Han, B. Wang, T. Guan, H. Gu, W. Wei, Review of object instance segmentation based on deep learning, J. Electron. Imaging 31 (4) (2022), https://doi.org/10.1117/1.JEI.31.4.041205, 041205–041205.

[37] B. Xiao, H. Xiao, J. Wang, Y. Chen, Vision-based method for tracking workers by integrating deep learning instance segmentation in off-site construction, Autom. Constr. 136 (2022) 104148, https://doi.org/10.1016/j.autcon.2022.104148.

[38] W. Gu, S. Bai, L. Kong, A review on 2D instance segmentation based on deep neural networks, Image Vis. Comput. 120 (2022) 104401, https://doi.org/10.1016/j.imavis.2022.104401.

[39] L. Huang, G. Fan, J. Li, H. Hao, Deep learning for automated multiclass surface damage detection in bridge inspections, Autom. Constr. 166 (2024) 105601, https://doi.org/10.1016/j.autcon.2024.105601.

[40] S. Zhao, D.M. Zhang, H.W. Huang, Deep learning–based image instance segmentation for moisture marks of shield tunnel lining, Tunn. Undergr. Space Technol. 95 (2020) 103156, https://doi.org/10.1016/j.tust.2019.103156.

[41] L. Ge, A. Sadhu, Deep learning-enhanced smart ground robotic system for automated structural damage inspection and mapping, Autom. Constr. 170 (2025) 105951, https://doi.org/10.1016/j.autcon.2024.105951.

[42] X.-L. Han, N.-J. Jiang, Y.-F. Yang, J. Choi, D.N. Singh, P. Beta, Y.-J. Du, Y.-J. Wang, Deep learning based approach for the instance segmentation of clayey soil desiccation cracks, Comput. Geotech. 146 (2022) 104733, https://doi.org/10.1016/j.compgeo.2022.104733.

[43] C. Liu, Y. Tao, J. Liang, K. Li, Y. Chen, Object detection based on YOLO network, 2018 IEEE 4th information technology and mechatronics engineering conference (ITOEC), IEEE (2018) 799–803, https://doi.org/10.1109/ITOEC.2018.8740604.

[44] H. Bian, Y. Liu, L. Shi, Z. Lin, M. Huang, J. Zhang, G. Weng, C. Zhang, M. Gao, Detection method of helmet wearing based on uav images and yolov7, in: 2023 IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) Vol. 6, IEEE, 2023, pp. 1633–1640, https://doi.org/10.1109/ITNEC56291.2023.10082536.

[45] M. Bakirci, Advanced aerial monitoring and vehicle classification for intelligent transportation systems with YOLOv8 variants, J. Netw. Comput. Appl. 237 (2025) 104134, https://doi.org/10.1016/j.jnca.2025.104134.

[46] D. Alexey, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, arXiv preprint arXiv: 2010.11929, 2020.

[47] S.S.A. Zaidi, M.S. Ansari, A. Aslam, N. Kanwal, M. Asghar, B. Lee, A survey of modern deep learning based object detection models, Digit. Signal Process. 126 (2022) 103514, https://doi.org/10.1016/j.dsp.2022.103514.

[48] J. Kaur, W. Singh, A systematic review of object detection from images using deep learning, Multimed. Tools Appl. 83 (4) (2024) 12253–12338, https://doi.org/10.1007/s11042-023-15981-y.

[49] J.-S. Chou, C.-Y. Liu, Optimized lightweight edge computing platform for UAV-assisted detection of concrete deterioration beneath bridge decks, ASCE, J. Comput. Civ. Eng. 38 (6) (2024), https://doi.org/10.1061/JCCEE5/CPENG-5905.

[50] J.-S. Chou, C.-Y. Liu, Pilgrimage walk optimization: folk culture-inspired algorithm for identification of bridge deterioration, Autom. Constr. 155 (2023) 105055, https://doi.org/10.1016/j.autcon.2023.105055.

[51] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks 4, 1995, pp. 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.

[52] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft. Comput. 22 (2) (2018) 387–408, https://doi.org/10.1007/s00500-016-2474-6.

[53] T.M. Shami, A.A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M.A. Summakieh, S. Mirjalili, Particle swarm optimization: a comprehensive survey, IEEE Access 10 (2022) 10031–10061, https://doi.org/10.1109/ACCESS.2022.3142859.

[54] T. Wu, Y. Dong, YOLO-SE: improved YOLOv8 for remote sensing object detection and recognition, Appl. Sci. 13 (24) (2023) 12977, https://doi.org/10.3390/app132412977.
[55] A. Torralba, B.C. Russell, J. Yuen, Labelme: online image annotation and applications, Proc. IEEE 98 (8) (2010) 1467–1484, https://doi.org/10.1109/JPROC.2010.2050290.
[56] I. Banerjee, B. Nguyen, V. Garousi, A. Memon, Graphical user interface (GUI) testing: systematic mapping and repository, Inf. Softw. Technol. 55 (10) (2013) 1679–1694, https://doi.org/10.1016/j.infsof.2013.03.004.