

Article

Lightweight YOLOv8-PSN for Real-Time Detection of Underground Structure Defects: A Comparison of Three Importance Criterion Pruning Methods

Kai Li ¹, Chao Ma ², Xingyu Nie ², Mengxia Zhang ¹, Yujie Meng ¹ and Dilong Zhang ^{1,*}

¹ China Academy of Building Research, Beijing 100013, China; likai202407@163.com (K.L.); zhangmengxia@cabr.com.cn (M.Z.); mengyujie@cabr.com.cn (Y.M.)

² School of Civil and Transportation Engineering, Beijing University of Civil Engineering and Architecture, Beijing 102616, China; machao@bucea.edu.cn (C.M.); 2108140123004@stu.bucea.edu.cn (X.N.)

* Correspondence: zhangdilong@cabr.com.cn

Abstract

To address the limited computational capacity of edge devices in underground defect detection tasks, this study utilizes an improved lightweight model, YOLOv8-PSN, as the baseline to systematically compare the compression mechanisms and performance of three structured pruning criteria: DepGraph, LAMP, and Taylor. The three pruning criteria elicited distinct network-restructuring behaviors. DepGraph exhibited a tendency to substantially compress intermediate layers, while LAMP induced more uniformly distributed channel pruning and was correlated with greater loss of high-frequency texture information. In contrast, Taylor pruning displayed an “inverted-pyramid” restructuring pattern. This pattern effectively reduces redundancy in the detection head while preserving the backbone’s robust feature representation capability. Based on comprehensive quantitative metrics and visual analyses, the Taylor-based method achieves significant acceleration of inference under high compression ratios. Crucially, it maintains higher detection confidence and bounding box continuity, demonstrating greater visual robustness to complex interference—such as strong illumination and low-light conditions—particularly when detecting fine cracks. This “metric-visual” discrepancy effectively highlights the limitations of relying solely on mAP for model evaluation in practical engineering. Furthermore, the proposed GPU-oriented channel alignment strategy improved inference efficiency on the tested NVIDIA RTX 5080 workstation GPU; this result is specific to the tested GPU architecture and may differ on embedded platforms. This study provides valuable guidance on the lightweight design and real-time deployment of defect detection models for underground engineering applications.

Keywords: underground structure; defect detection; YOLOv8; structured pruning; lightweight model; feature preservation



Academic Editor: Hugo Rodrigues

Received: 18 June 2026

Revised: 20 July 2026

Accepted: 23 July 2026

Published: 2 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

As one of the most extensively utilized construction materials, concrete finds wide application in the construction of underground structures. Nevertheless, under the combined influence of complex loading and environmental factors, the concrete surfaces of underground structures are extremely prone to various surface defects, including cracks, spalling, and efflorescence. These defects not only degrade the visual quality of the structures but, if not detected and treated in a timely manner, can also result in the gradual deterioration

of load-bearing capacity and deformation performance, thus presenting serious safety hazards [1,2]. Recent studies and reviews on tunnel and civil-infrastructure inspection have highlighted that optical inspection, robotic inspection, and computer-vision-based condition assessment are becoming important tools for the large-scale maintenance of concrete linings and related infrastructure [3–6]. However, underground structures are generally characterized by limited spaces, poor illumination, and complex environmental interferences. Traditional manual inspection methods are plagued by low efficiency, high subjectivity, and substantial safety risks, and they are unable to meet the practical requirements of large-scale, long-term monitoring. Therefore, achieving rapid, accurate, and automated detection of concrete surface defects under complex environmental conditions is of great importance for enhancing the operational safety of structures and improving operation and maintenance efficiency [7–9].

Currently, the detection of concrete surface defects in underground structures mainly relies on manual inspection. Nevertheless, the efficiency and consistency of this method are significantly influenced by the subjective experience of the inspectors, which makes it challenging to effectively detect early-stage surface defects. Moreover, limited by factors like confined spaces, insufficient illumination, and the complex working environments characteristic of underground structures, manual inspection also presents certain safety hazards. As a result, it cannot satisfy the practical requirements for the refined and long-term monitoring of engineering structures [10]. With the advancement of computer vision and image processing technologies, automated defect recognition methods that rely on image information have been gradually adopted. Recent large-scale and multi-class defect benchmarks, such as dacl10k and GYU-DET, have further promoted reproducible research on concrete and bridge surface defects in the broader civil-infrastructure community [11,12]. Traditional computer-vision methods have enhanced the inspection efficiency by extracting hand-crafted features from concrete surface images and making rule-based decisions. However, their dependence on filtering operations, fixed thresholds, and manually designed features renders them sensitive to illumination variations, background interference, and changes in crack morphology [13]. As a result, their detection robustness is restricted under the complex working conditions of underground structures, and they still find it difficult to achieve stable and accurate recognition of surface defects [14–16].

Recent deep learning studies on underground and concrete surface inspection have primarily focused on improving detection robustness, defect characterization, and deployment efficiency. Multi-scale feature fusion, attention mechanisms, deformable convolutions, and detail-preserving encoders have been employed to enhance the recognition of defects with irregular morphologies, weak textures, and indistinct boundaries under uneven illumination and complex backgrounds [17–24]. Detection–segmentation frameworks, transfer learning, laser-scanning techniques, and data-efficient learning strategies have further improved defect localization and quantitative assessment while reducing the dependence on large-scale, densely annotated visible-light datasets [25–32]. In parallel, lightweight network architectures and inspection systems based on mobile robots or UAVs have promoted automated inspection in resource-constrained and GNSS-denied underground environments [3,33–36]. Collectively, these studies have improved detection accuracy, environmental robustness, and system-level deployment. However, most lightweight studies focus on architectural modification or report the performance of a single compression method. A systematic understanding of how different structured pruning criteria redistribute network channels and affect the preservation of task-relevant features—particularly weak-texture and small-defect features in underground environments—remains limited.

Despite the remarkable progress made by the aforementioned deep learning methods in detection accuracy, the majority of these models are encumbered with a large number

of parameters and high computational complexity. As a result, it is challenging to conduct real-time inference on mobile inspection robots, UAVs, or embedded devices, which significantly constrains their deployment and practical application in underground engineering [5,37–39]. The trade-off between detection accuracy and inference speed remains a central consideration in the design and scaling of modern object detectors [40]. Recent reviews of YOLO architectures further indicate that single-stage detectors are continuously evolving towards faster and more deployment-oriented designs [41]. Model compression techniques have emerged as a vital approach to surmount this bottleneck. For instance, Zhu et al. [42] applied knowledge distillation to a YOLOv5-based framework, substantially reducing the parameter count while maintaining or even enhancing detection accuracy. Similarly, Ou et al. [43] integrated channel pruning into a YOLOv10-based multi-scale feature fusion model to further decrease computational complexity for deployment in resource-constrained environments. Among these methods, structured pruning has received special attention because of its capacity to maintain regular network architectures, thus enabling hardware acceleration. In structured pruning, channel importance evaluation criteria are mainly classified into magnitude-based approaches, topology-dependency-based approaches, and gradient-sensitivity-based approaches, which correspond to different assumptions regarding network redundancy and feature preservation.

To this end, this study systematically conducts structured pruning on a real underground-structure defect dataset based on the previously proposed YOLOv8-PSN model. Three representative structured pruning criteria are selected: DepGraph [44], based on topology dependency, LAMP [45] based on layer-adaptive magnitude, and Taylor pruning [46] based on gradient sensitivity. Instead of proposing a fundamentally new pruning mechanism, this empirical study aims to explore how different importance criteria reshape a YOLO detector designed for underground inspection and how these structural alterations affect the trade-off among accuracy, inference speed, and visual robustness in underground inspection scenarios.

Although DepGraph, LAMP, and Taylor pruning have been extensively studied in general vision tasks, our understanding of their comparative performance on defect detectors optimized for underground concrete surface inspection is still insufficient. Underground defect images usually present features like low illumination, local over-exposure, water stains, rough concrete textures, indistinct defect boundaries, and multi-scale defect morphology. These characteristics may potentially result in asymmetric layer sensitivities during pruning, especially among the backbone, neck, attention modules, and detection head. Therefore, it is crucial to identify the criterion that can better preserve task-relevant feature channels when a detector has been optimized for underground concrete surface defect detection.

Reliable underground inspection may depend on both efficient on-device inference and robust visual communication. Pseudo-analog video transmission has been explored to preserve visually important regions under constrained wireless links [47]; however, transmission optimization is outside the scope of this study. It is retained here only as broader deployment context, whereas the present experiments focus exclusively on structured pruning and inference efficiency.

The main works of this study are as follows: (1) a controlled comparison of DepGraph, LAMP, and Taylor pruning is carried out by employing YOLOv8-PSN as a representative optimized detector for underground concrete surface defect detection; (2) the layer-wise restructuring patterns resulting from different pruning criteria are analyzed, which reveals the asymmetric sensitivity among the backbone, neck, attention modules, and detection head; and (3) the speed–accuracy–robustness trade-off is assessed using quantitative met-

detection head structures, which are highly sensitive to input and output channels. This approach reduces structural redundancy while preventing excessive interference with feature representation.

Based on these pruning targets, this study constructs a unified structured pruning framework that integrates three representative channel importance criteria: dependency-graph-based pruning (DepGraph), layer-adaptive magnitude-based pruning (LAMP), and first-order Taylor-expansion-based pruning (Taylor). As shown in Figure 2, the pruning workflow comprises three main steps. First, dependency grouping is carried out to identify coupled channels that need to be pruned together. Second, channel importance scores are computed according to the selected pruning criterion, and pruning masks are generated using global or layer-adaptive thresholds. Third, channel removal and structural repair are implemented to obtain a valid pruned model. Through this unified workflow, the impacts of different pruning criteria can be compared under the same module-level pruning strategy.

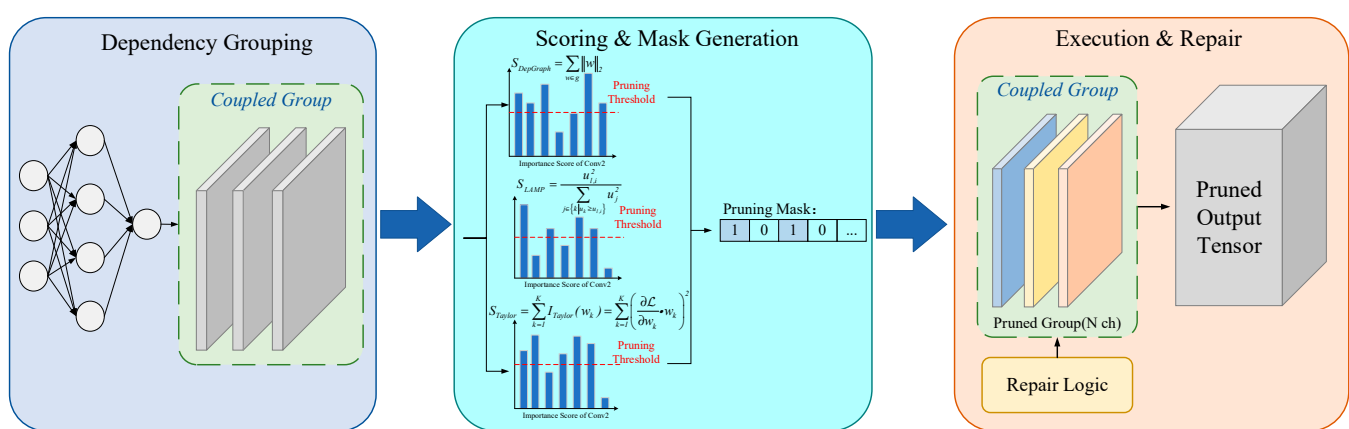


Figure 2. Overall flowchart of the proposed structured pruning process.

2.2.1. Layer-Adaptive Magnitude-Based Pruning (LAMP)

Traditional global magnitude pruning methods usually sort and prune channels according to the absolute values of their weights. Nevertheless, in deep neural networks, the numerical distributions of parameters differ notably among different layers. Applying a unified global threshold directly often leads to excessive retention in shallow layers and excessive pruning in deep layers, thus undermining the hierarchical representation ability of the network.

Layer-Adaptive Magnitude Pruning (LAMP) introduces a scale-invariant importance metric to adaptively normalize intra-layer parameters, thereby alleviating the pruning bias caused by inter-layer scale inconsistencies. In the context of structured pruning, taking the convolutional kernel (or channel) as the basic unit, let the L_2 norm of the i -th convolutional kernel in the l -th layer be denoted as $u_{l,i} = \|w_{l,i}\|_2$. The LAMP score for this convolutional kernel is defined as:

$$S_{LAMP} = \frac{u_{l,i}^2}{\sum_{j \in \{k | u_k \geq u_{l,i}\}} u_j^2} \quad (1)$$

Equation (1) shows that the LAMP score solely relies on the relative ranking of the parameter and its cumulative energy contribution within the distribution of the current layer, rather than its absolute numerical value. Through this non-linear mapping, regardless of the weight distribution range among different layers, the relatively important feature channels in each layer can receive higher scores. This theoretically guarantees a more balanced retention ratio across all network layers during the global pruning process.

2.2.2. Taylor Expansion-Based Pruning (Taylor)

Magnitude-based pruning methods concentrate on assessing the inherent scale of parameters. Conversely, Taylor expansion pruning evaluates parameter importance from the perspective of loss sensitivity. This criterion leverages gradient information from the optimization process to measure the potential influence on the final loss function when individual parameters are removed. By integrating weight values and gradient information, Taylor pruning pinpoints structural units that are crucial for sustaining model performance. It achieves this from the dynamic perspective of model optimization.

The essence of pruning is to identify a subset of parameters. The removal of this subset minimizes the increment of the model's loss function, denoted as ΔL . For a given weight parameter w , the removal operation is mathematically equivalent to applying a perturbation $\Delta w = -w$. Based on the Taylor series expansion theory, the variation in the loss function ΔL can be approximated as follows:

$$\Delta \mathcal{L} = \mathcal{L}(w + \Delta w) - \mathcal{L}(w) \approx \frac{\partial \mathcal{L}}{\partial w} \Delta w + \frac{1}{2} (\Delta w)^T \mathbf{H} \Delta w + (\|\Delta w\|^3) \quad (2)$$

where $\frac{\partial \mathcal{L}}{\partial w}$ denotes the first-order gradient of the loss function with respect to the weight, and $\mathbf{H}$ represents the Hessian matrix (i.e., the second-order derivative). In deep neural networks, computing the Hessian matrix incurs a prohibitively high computational cost. To strike a balance between computational efficiency and evaluation accuracy, higher-order terms are typically discarded, and only the first-order term is retained for approximation. However, in a mini-batch training environment, the signs of the gradients show random oscillatory characteristics. If the product of the gradient and the weight is directly used as the importance score, positive and negative gradients may cancel each other out during the accumulation process, leading to an underestimation of parameter sensitivity. To solve this problem, the method adopts the Squared Taylor Criterion, using the squared term of the first-order information to approximate the error energy induced by parameter removal, which also implicitly incorporates an empirical approximation of Fisher information. For a single weight w , its importance score $I_{Taylor}(w)$ is defined as the absolute-value approximation of the variation in the loss function:

$$I_{Taylor}(w) = \left(\frac{\partial \mathcal{L}}{\partial w} w \right)^2 \quad (3)$$

In structured pruning tasks, the fundamental unit of pruning has expanded from a single parameter to a coupled group consisting K of related parameters. To assess the overall contribution of the entire structured unit to the network's performance, an aggregation mechanism from element-level scores to group-level scores needs to be defined. Usually, the squared Taylor scores of all independent parameters within the group are summed up. Consequently, the importance evaluation score for a structured coupled group $g = \{w_1, \dots, w_K\}$ is defined as:

$$S_{Taylor} = \sum_{k=1}^K I_{Taylor}(w_k) = \sum_{k=1}^K \left(\frac{\partial \mathcal{L}}{\partial w_k} w_k \right)^2 \quad (4)$$

This aggregation mechanism implicitly relies on the assumption that the contributions of individual parameters are mutually independent. By accumulating the approximated error energy of all weights within the group, it quantifies the overall impact of removing the entire structural unit on the loss function. Furthermore, to reduce the estimation variance caused by Stochastic Gradient Noise, this approach adopts a multi-batch expectation estimation strategy. By aggregating importance scores across multiple mini-batches and calculating their expected value, the random fluctuations of gradients are effectively smoothed

out, thus ensuring that the final generated scores accurately reflect the true importance of the structural units within the context of the global data distribution.

2.2.3. Dependency Graph-Based Pruning (DepGraph)

Unlike the LAMP and Taylor criteria, which emphasize the static evaluation of parameters within independent layers, the Dependency Graph-based pruning (DepGraph) approach employs a structure-first and dynamically induced strategy. Rather than making pruning decisions directly on individual convolutional layers, this method initially constructs an explicit inter-layer dependency graph through recursive traversal of the network topology. This mechanism is fundamentally distinct from independent pruning as it ensures that pruning operations strictly comply with the network's connectivity constraints, thus preventing structural discontinuity resulting from dimension mismatch.

After establishing the coupled group as the fundamental pruning unit, this method quantifies its comprehensive contribution according to the L_2 norm principle. For a coupled group $g = \{w_1, \dots, w_K\}$ consisting of K correlated weight parameters, the sum of the L_2 norms of the parameters within the group is generally adopted as its importance score $I(g)$.

$$I(g) = \sum_{w \in g} \|w\|_2 \quad (5)$$

However, pruning directly from the aggregated scores of the original pre-trained weights may provide insufficient separation between redundant and task-relevant groups. This is because, under conventional training, network parameters generally follow a Gaussian distribution, and the numerical difference in weights between redundant and critical parameters is not significant, leading to a lack of discriminability in the importance evaluation. To address this issue, a group-level sparse training mechanism is introduced before pruning. Its core idea is to explicitly incorporate a regularization term, which is consistent with the importance evaluation criterion, into the optimization objective, thus forcing the model to automatically compress the weights of redundant groups during the training process. The total loss function incorporating the sparsity constraint is defined as follows:

$$L_{total} = L_{task} + \lambda \sum_{g \in G} I(g) \quad (6)$$

where L_{task} denotes the task loss, and λ represents the sparsity penalty coefficient. Equation (6) demonstrates objective-metric consistency: specifically, the penalty term in the optimization process is mathematically consistent with the evaluation metric used during pruning. By minimizing this loss function, the model automatically forces the weights of coupled groups that make minor contributions to the task towards zero. As a result, the pruning decision turns into a natural truncation of the optimized results: by setting a threshold τ according to the predefined compression ratio, coupled groups that satisfy $I(g) < \tau$ are removed, thus achieving model simplification while maintaining structural integrity.

2.3. GPU-Oriented Channel Alignment Strategy

Although unstructured pruning or arbitrary-ratio structured pruning can substantially reduce the theoretical computational complexity (GFLOPs) and the number of parameters of a model, in actual hardware deployment, such irregular sparsity is frequently difficult to translate into real inference acceleration. Modern Graphics Processing Units (GPUs) generally adopt a Single Instruction, Multiple Threads (SIMT) architecture, with 32 threads serving as a single execution unit. When the number of input or output channels of a convolutional kernel cannot be evenly divided by 32, the hardware computational units produce memory fragmentation during General Matrix Multiply (GEMM) or convolution

operations. This results in a decrease in the efficiency of memory coalescing and, consequently, causes computational resources to remain idle. To narrow the gap between the theoretical compression ratio and the measured inference speed on the tested NVIDIA RTX 5080 workstation GPU, this paper presents a hierarchical GPU-oriented channel alignment strategy at the final stage of the pruning decision.

This strategy applies a dynamic hierarchical mapping based on the number N_{raw} of raw retained channels calculated from the importance evaluation.

(1) 32-alignment priority: For layers with a large number of channels ($N_{raw} \geq 32$), the 32-alignment strategy is given priority. Specifically, the number of channels is rounded down to the nearest multiple of 32 using the formula $N_{final} = \lfloor N_{raw}/32 \rfloor \times 32$, so as to maximize the utilization of the Tensor Core.

(2) 8-alignment fallback: For layers having a smaller number of channels $8 < N_{raw} < 32$, in order to prevent accuracy loss resulting from excessive pruning, the strategy is adjusted to 8-alignment. This adjustment is carried out through the formula $N_{final} = \lfloor N_{raw}/8 \rfloor \times 8$, so as to meet the minimum bit-width requirements of vectorized instruction sets.

(3) Flexible protection for extremely narrow layers: For extremely narrow bottleneck layers where $N_{raw} \leq 8$, forced alignment would lead to significant parameter redundancy or severe information flow blockage. Therefore, within this interval, mandatory alignment is no longer carried out; instead, the originally calculated result N_{raw} (subject to the minimum retained channel constraint) is retained to prioritize ensuring the model's connectivity and baseline accuracy.

Through this hierarchical mechanism, the model seeks to improve GPU utilization in wide layers while retaining essential feature pathways in extremely narrow layers on the tested NVIDIA RTX 5080 workstation. The observed speed–accuracy balance is specific to the tested GPU architecture and software stack and may differ on embedded platforms.

3. Experimental Setup

3.1. Experimental Environment and Parameters

3.1.1. Experimental Platform and Data Overview

All experiments in this study were carried out on a unified high-performance computing workstation. The hardware environment is furnished with high-performance NVIDIA GPUs to fulfill the parallel computing requirements of deep learning models. The software environment is built upon the Windows operating system and the PyTorch deep learning framework, utilizing CUDA technology for acceleration. The detailed hardware and software configurations are summarized in Table 1.

Table 1. Experimental hardware and software configurations.

Component	Detailed Specification
GPU	NVIDIA GeForce RTX5080
CPU	Intel(R)Core™i9-14900K
RAM	16 GB
Operating System	Windows 11
Programming Language	Python 3.12.3
Deep Learning Framework	PyTorch 2.8.0
Acceleration Library	CUDA 12.8

The experimental dataset was initially established in our prior YOLOv8-PSN work [48] and specifically constructed for real-world underground structural inspection scenarios. It comprises 7353 high-resolution images and 39,094 labeled defect instances. The dataset centers on six prevalent surface defect categories in underground concrete structures: ex-

posed reinforcement, rust, cracks, spalling, efflorescence, and delamination. Representative images of these six defect categories are shown in Figure 3. These categories were chosen as they represent typical deterioration phenomena associated with durability and serviceability. For instance, inadequate concrete cover, carbonation, chloride ingress, and moisture penetration can accelerate reinforcement corrosion; the expansion of corrosion products may further lead to cracking, spalling, and exposure of reinforcement.

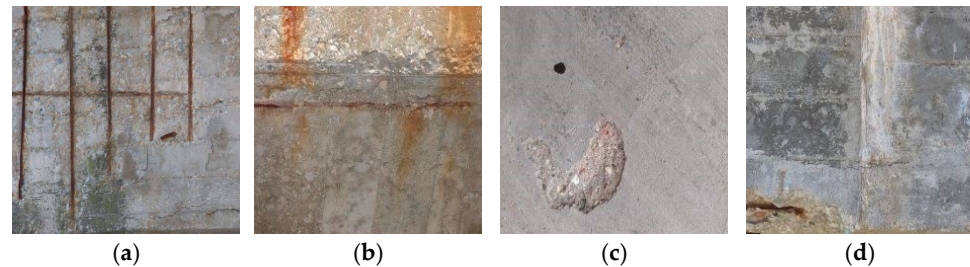


Figure 3. Sample images from the concrete surface defects dataset: (a) exposed reinforcing bars; (b) rust; (c) spalling; (d) efflorescence [48].

All ground-truth bounding boxes were manually annotated using Labelling (version 1.8.6) in the standard YOLO .txt format. The images feature challenging underground conditions, including insufficient illumination, local overexposure caused by artificial lighting, water leakage stains, rough concrete textures, and background clutter. To enhance the reliability of the annotations, the labels were inspected by researchers well-versed in concrete defect characteristics. Ambiguous samples were reviewed and rectified prior to training. The category distribution is presented in Figure 4. To minimize scene leakage between subsets, the dataset was partitioned based on discrete image acquisition sequences instead of simple random sampling, with an 8:1:1 ratio for training, validation, and testing. More comprehensive information regarding image acquisition, defect category definition, annotation protocol, dataset partitioning, YOLOv8-PSN module ablation, and baseline detector comparison can be found in our previous YOLOv8-PSN study [48].

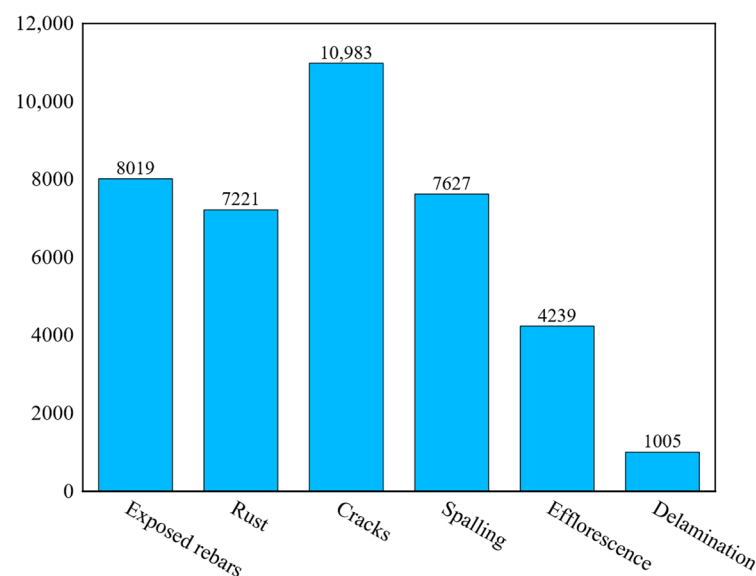


Figure 4. Class-wise distribution of labeled instances in the concrete surface defect dataset [48].

In the previous baseline comparison, YOLOv8-PSN also attained higher class-wise average precision (AP) values across all six defect categories compared to SSD, YOLOv6, YOLOv10, YOLOv11, and Faster R-CNN. Specifically, its AP values reached 83.3% for

exposed reinforcement, 70.9% for rust, 77.0% for cracks, 76.4% for spalling, 65.7% for efflorescence, and 90.1% for delamination [48]. These results justify the use of YOLOv8-PSN as the fixed baseline in the present pruning study.

3.1.2. Pruning and Fine-Tuning Strategy and Parameter Configuration

To reduce accuracy fluctuations caused by the altered parameter distribution after structured pruning, all layers of each pruned model are unfrozen and fine-tuned on the training set.

The pruned models were fine-tuned for 500 epochs to recover detection performance after channel removal. Early stopping was disabled to allow the compressed models to reach stable convergence. The remaining hyperparameters were kept consistent with the original YOLOv8-PSN configuration; Mosaic and Mixup augmentation and a cosine-annealing learning-rate scheduler were used. Table 2 lists the detailed settings. Wall-clock fine-tuning time, pruning time, and total GPU-hours were not systematically recorded under a dedicated profiling protocol and can no longer be recovered reliably; this omission is therefore acknowledged as a limitation.

Table 2. Parameter configurations for fine-tuning training.

Parameter Name	Configuration Value
Input Image Size	640 × 640
Total Fine-tuning Epochs	500
Batch Size	16
Optimizer Type	SGD
Initial Learning Rate	0.01
Momentum	0.937
Optimizer State	Reset (New initialization)
EMA Usage	Enabled
Weight Decay	0.0005
Data Augmentation Strategy	Mosaic (1.0); Mixup (0.2)

Although 500 epochs without early stopping could increase the risk of overfitting, the recorded training dynamics show no evident late-stage deterioration. In Figure 5, the training and validation losses converge and remain stable during the later epochs. Figure 6 shows that mAP@50 increases rapidly during early fine-tuning and then stabilizes, while recall exhibits only small oscillations. Under the current protocol, these curves suggest that extended fine-tuning primarily supports post-pruning recovery rather than obvious overfitting.

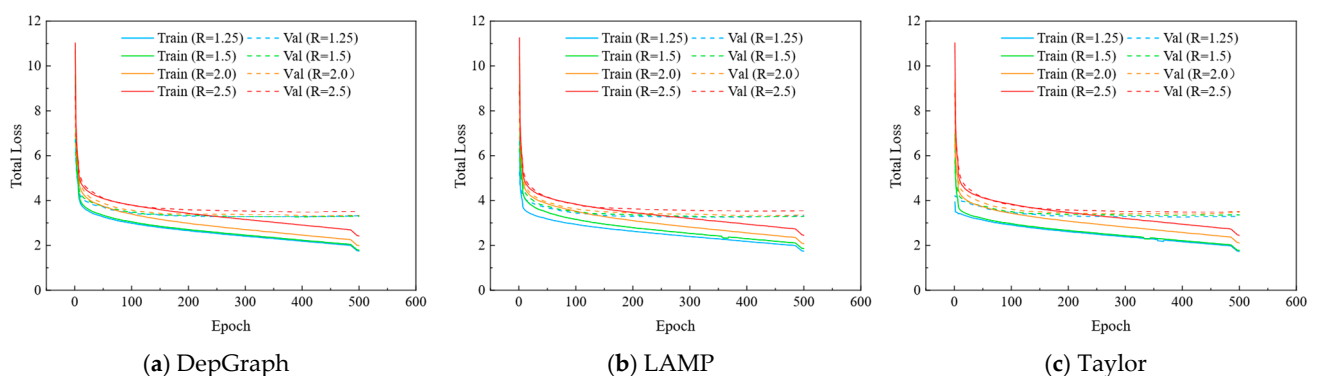


Figure 5. Training and validation loss curves during 500-epoch fine-tuning at different compression ratios.

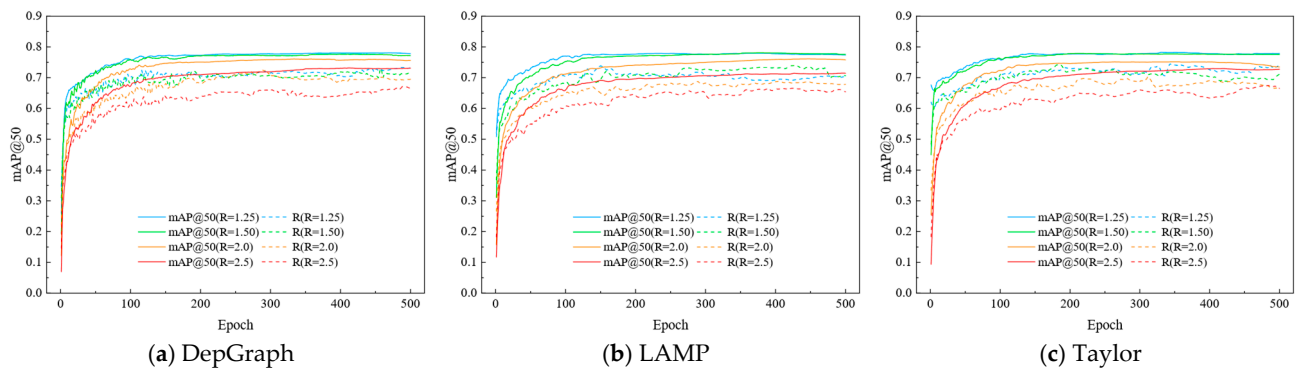


Figure 6. Evolution of mAP@50 and recall during 500-epoch fine-tuning at different compression ratios.

3.2. Evaluation Metrics

To comprehensively assess the performance of the pruned model in the underground structure defect detection task, this study selects precision, recall, mean Average Precision (mAP), Frames Per Second (FPS), Giga Floating-Point Operations (GFLOPs), and parameter count (Parameters, M) as the core evaluation metrics. Regarding the evaluation of detection accuracy, this paper employs Precision (P) and Recall (R) to respectively characterize the correctness and completeness of the model's prediction results. Based on this, to comprehensively gauge the overall detection performance of the model, mAP@50 (i.e., the mean Average Precision when the Intersection over Union (IoU) threshold is 0.5) is selected as the core evaluation metric. Meanwhile, mAP@50:95 (the average value when the IoU threshold dynamically varies in the range of 0.5 to 0.95) is introduced to further evaluate the model's robustness performance under stricter localization thresholds. The specific calculation formulas for each metric are as follows:

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (9)$$

In terms of inference speed, this paper adopts FPS (Frames Per Second) as the core evaluation metric. All FPS values were evaluated under a unified benchmarking protocol on the RTX5080 workstation: batch size of 16, FP16 half-precision formatting, and forward-pass latency only, excluding Non-Maximum Suppression (NMS) time. Therefore, the reported FPS values should be interpreted as controlled GPU-based relative acceleration indicators rather than direct evidence of embedded-device performance. This protocol isolates the computational efficiency of the network structures and supports fair comparison among pruning criteria.

To evaluate the lightweight properties and deployment feasibility of the proposed model, this study adopts GFLOPs and parameter count (Parameters, unit: M) as the primary metrics for model complexity. Specifically, GFLOPs quantifies the model's theoretical computational complexity, while the parameter count captures the total number of trainable parameters and provides an approximate estimation of the model's storage and memory footprint. For edge-computing platforms with constrained computational power and storage capacity, such as underground inspection robots and unmanned aerial vehicles (UAVs), reducing both the parameter count and computational complexity is critical for enabling efficient on-device deployment.

3.3. Sparse Training

For the Dependency Graph (DepGraph)-based pruning method, this paper presents a sparsity-aware training phase before performing structured pruning. This design strictly follows the core logic of the original DepGraph framework. Specifically, we apply sparse regularization to encourage the formation of structured redundancy among convolutional channels and to stabilize the dependency learning process across composite modules. This step guarantees the reliability of pruning decisions without sacrificing the structural consistency of the model. It is particularly important to note that this kind of sparse training is only applied to the dependency graph network, while the LAMP and Taylor pruning methods are directly implemented on the baseline model. After the pruning is completed, all models use exactly the same training settings for fine-tuning, thus ensuring the fairness of performance comparisons.

4. Experimental Results and Analysis

4.1. Performance Analysis Under Single Pruning Methods

DepGraph, LAMP, and Taylor pruning were evaluated using the same training configuration, inference environment, target modules, compression ratios, and GPU-oriented channel alignment procedure. DepGraph additionally uses sparse training before pruning, whereas LAMP and Taylor are applied directly to the baseline. This methodological difference is part of the standard DepGraph procedure and is considered when interpreting the results.

4.1.1. Analysis of the DepGraph Method

DepGraph constructs a pruning graph from network topology dependencies and introduces group-level sparse training before pruning. This design encourages separation between retained and removable channel groups while maintaining connectivity constraints. Table 3 shows that, as R increases from 1.25 to 2.5, the parameter count decreases from 8.31 M to 4.35 M and GFLOPs decrease from 21.64 to 13.00. FPS increases from 659.3 to 1121.6 under the tested workstation GPU protocol, indicating reduced structural redundancy and faster inference.

Table 3. Parameter comparison of the DepGraph pruning method under different compression ratios.

Method	Compression Ratio	Parameters (M)	GFLOPs	FPS
YOLOv8-PSN-DepGraph	1 (Baseline)	10.44	24.1	648.1
	1.25	8.3144	21.6412	659.3
	1.5	6.9862	18.9198	701.9
	2.0	5.4089	15.8205	715.4
	2.5	4.3469	13.0011	1121.6

DepGraph maintains relatively stable detection accuracy at low-to-medium compression ratios, as shown in Figure 7. At R = 1.25 and R = 1.5, mAP@50 is 78.1% and 77.6%, respectively. At R = 2.0 and R = 2.5, it decreases to 76.1% and 73.2%, illustrating the speed–accuracy trade-off as compression increases. The sparse-training stage helps retain task performance at the evaluated compression ratios.

DepGraph produces a non-uniform channel redistribution pattern. As shown in Figure 8, the shallow backbone layers are largely retained, whereas intermediate feature fusion modules in the neck are compressed more strongly, with some layers retaining relatively few channels. Figure 9 further shows that the detection head retains a comparatively high proportion of channels. This middle-layer-focused pattern explains how DepGraph

reduces computational cost while preserving relatively stable detection accuracy under several compression settings.

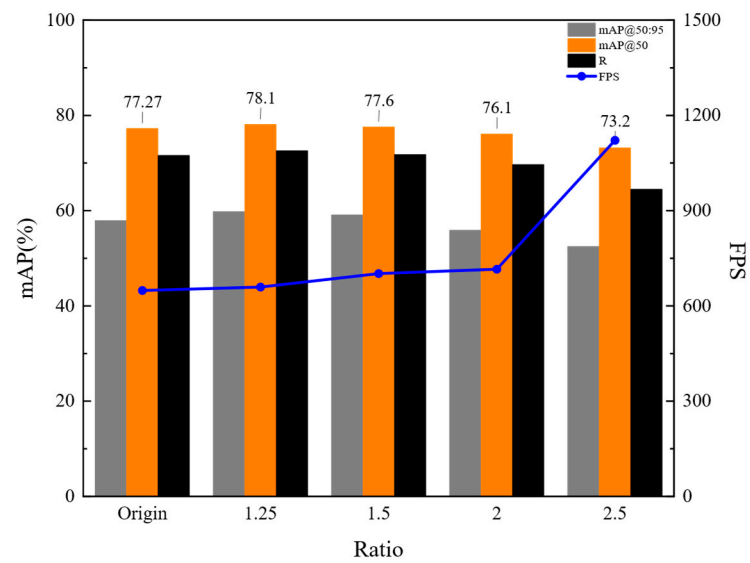


Figure 7. Comparison of detection performance and inference speed of DepGraph under different compression ratios.

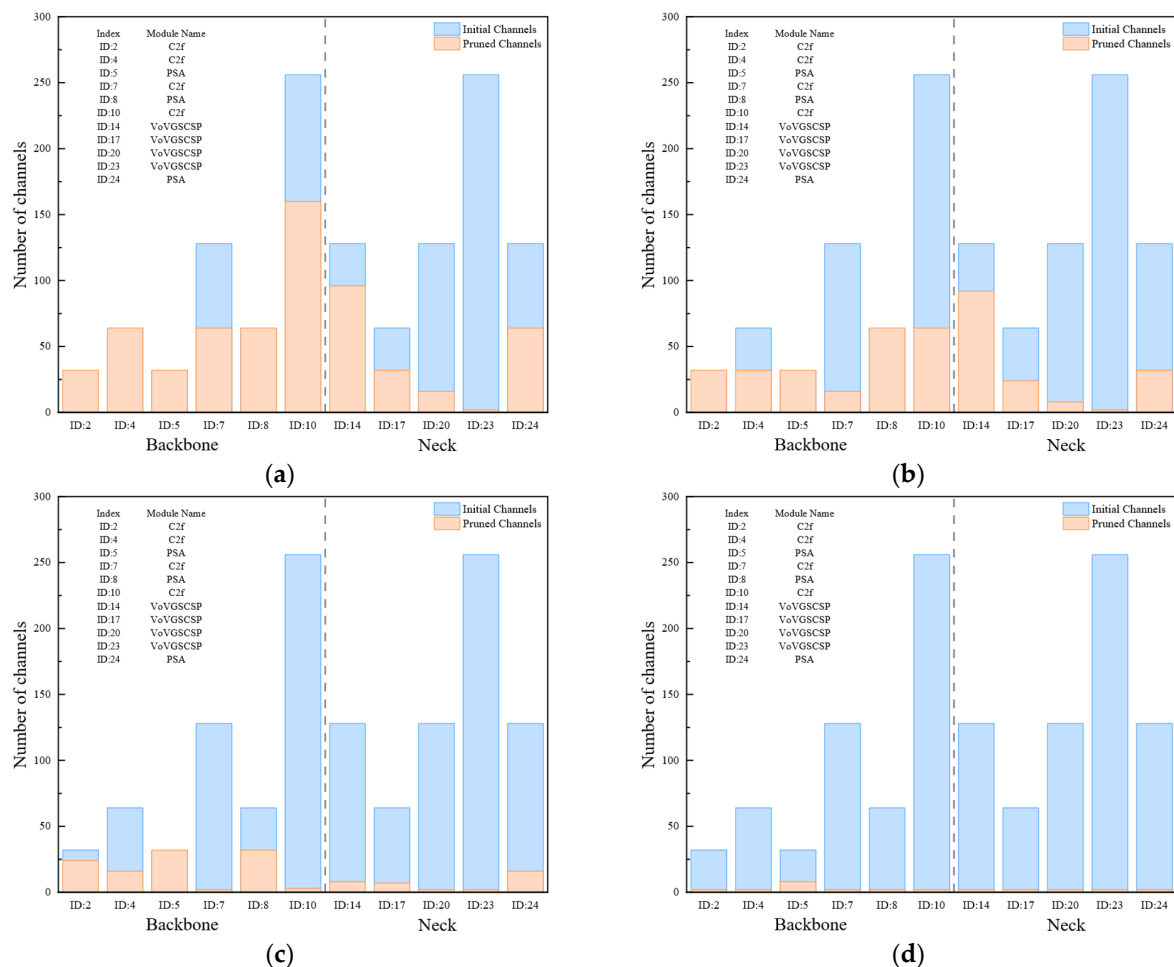


Figure 8. Channel number distribution in the backbone and neck of the DepGraph model under different compression ratios: (a) R = 1.25; (b) R = 1.5; (c) R = 2.0; (d) R = 2.5. The gray dashed line indicates the boundary between the backbone and neck.

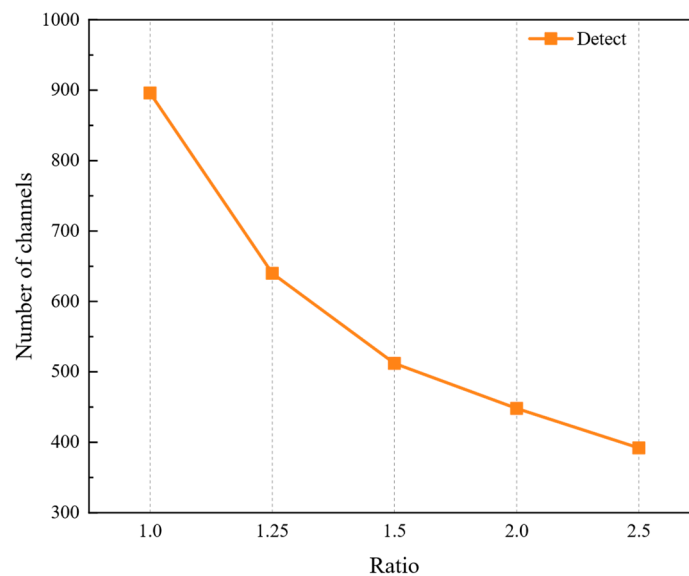


Figure 9. Channel compression trends within the detection head under different DepGraph compression ratios.

4.1.2. Analysis of the LAMP Method

Layer-Adaptive Magnitude Pruning (LAMP) does not require an additional sparse-training stage. It normalizes the within-layer weight-magnitude distribution to reduce pruning bias caused by scale differences across layers. Compared with DepGraph, it avoids explicit topology-dependent sparse regularization and is simpler to apply. Table 4 shows that, from $R = 1.25$ to $R = 2.5$, the parameter count decreases from 8.36 M to 4.26 M, GFLOPs decrease from 22.18 to 12.02, and FPS reaches 1274.6. Under the current protocol, LAMP therefore provides the highest maximum FPS among the three methods, although its accuracy decreases more at the largest compression ratio.

Table 4. Parameter comparison of the LAMP method under different compression ratios.

Method	Compression Ratio	Parameters (M)	GFLOPs	FPS
YOLOv8-PSN-LAMP	1 (Baseline)	10.44	24.1	648.1
	1.25	8.3593	22.1768	688.5
	1.5	6.9007	18.8616	701
	2.0	5.2127	14.7319	935.7
	2.5	4.2594	12.0249	1274.6

LAMP shows a larger accuracy decrease at high compression ratios as shown in Figure 10. At $R = 2.5$, mAP@50 decreases to 71.5%, below the corresponding DepGraph and Taylor results. This behavior is consistent with the possibility that magnitude-based scoring removes some low-magnitude channels that remain relevant to weak-texture or low-contrast defects.

LAMP shows a relatively uniform within-layer pruning tendency driven by normalized weight magnitude. At high compression ratios, several backbone and neck modules become narrow as shown in Figure 11. The corresponding channel distribution in the detection head is shown in Figure 12. This configuration reduces model complexity but can limit the hierarchical representation of fine-grained textures and multi-scale semantic information, which is consistent with its high FPS and larger accuracy loss under aggressive pruning.

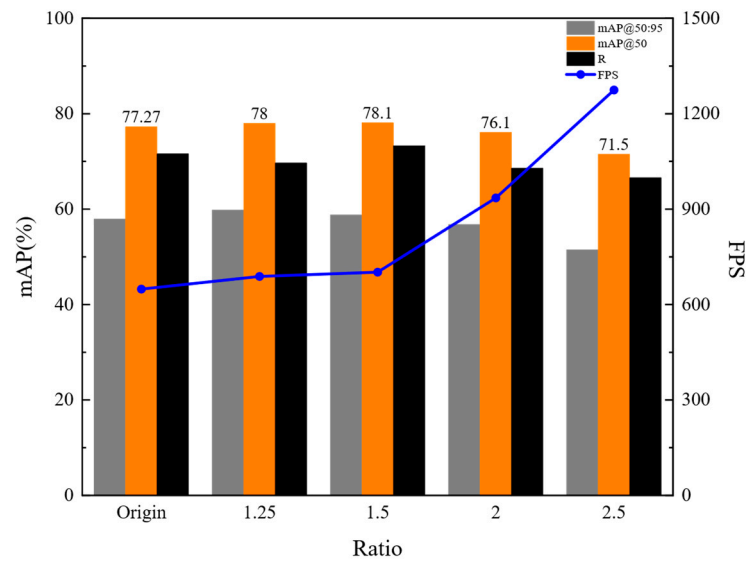


Figure 10. Comparison of detection performance and inference speed of LAMP under different compression ratios.

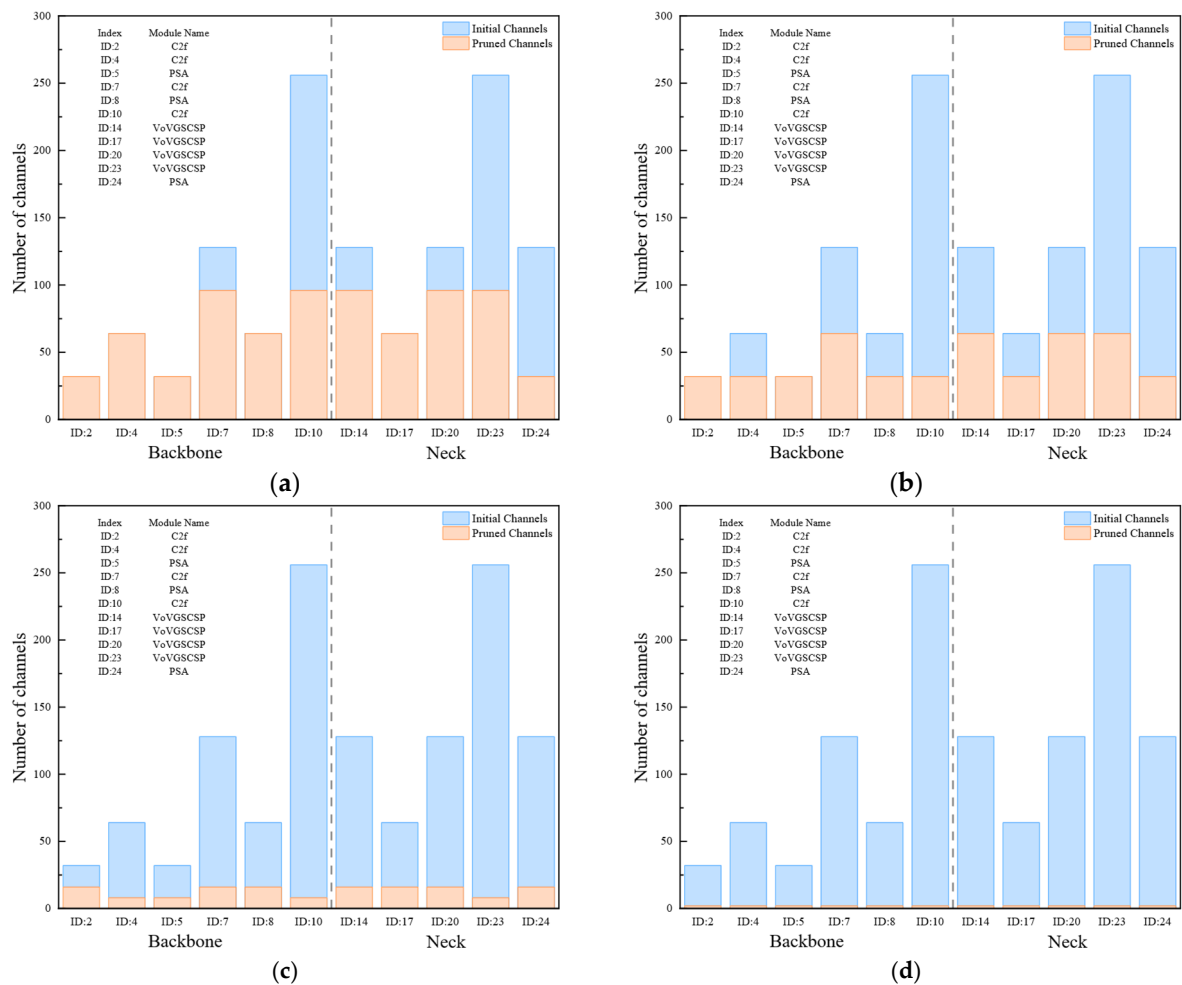


Figure 11. Channel number distribution in the backbone and neck of the LAMP model under different compression ratios: (a) R = 1.25; (b) R = 1.5; (c) R = 2.0; (d) R = 2.5. The gray dashed line indicates the boundary between the backbone and neck.

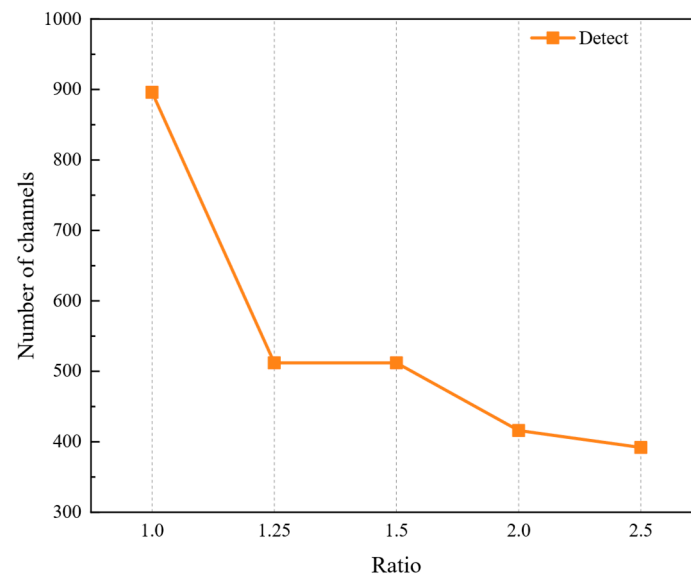


Figure 12. Channel compression trends within the detection head under different LAMP compression ratios.

4.1.3. Analysis of the Taylor Method

First-order Taylor pruning estimates channel importance using weight values and gradients of the loss function. The criterion approximates the change in loss caused by removing a channel, thereby incorporating task-related sensitivity into pruning decisions. Unlike magnitude-only criteria, it uses optimization information to distinguish channels with different effects on the objective.

Table 5 shows that Taylor pruning provides gradual structural compression across the evaluated ratios. From $R = 1.25$ to $R = 2.5$, the parameter count decreases from 10.44 M to 4.27 M, a reduction of approximately 59%. GFLOPs decrease from 24.1 to 12.23, while FPS increases from 648.1 to 1264.3 under the tested protocol. This maximum FPS is slightly below that of LAMP (1274.6 FPS), so the Taylor result is interpreted as a competitive speed–accuracy trade-off rather than the highest acceleration.

Table 5. Parameter comparison of the Taylor pruning method under different compression ratios.

Method	Compression Ratio	Parameters (M)	GFLOPs	FPS
YOLOv8-PSN-Taylor	1 (Baseline)	10.44	24.1	648.1
	1.25	8.3057	22.5680	723.2
	1.5	6.9908	20.9173	732.5
	2.0	5.2534	14.8759	1022
	2.5	4.2652	12.2259	1264.3

Taylor pruning shows a relatively gradual accuracy decrease at medium-to-high compression ratios as shown in Figure 13. At $R = 2.5$, its mAP@50 remains above that of LAMP and is close to DepGraph in some settings. Because Taylor scoring incorporates task-related gradients, it can distinguish channels with different local effects on the detection loss and reduce the risk of abrupt performance degradation under the current protocol.

Taylor pruning produces a distinct gradient-guided structural pattern. At low-to-medium compression ratios, most backbone and neck channels are retained, as shown in Figure 14, whereas a larger share of pruning occurs in the detection head, as shown in Figure 15. This inverted-pyramid pattern suggests that the gradient-based criterion assigns greater importance to feature extraction and multi-scale fusion layers while identifying more removable capacity in the prediction layers.

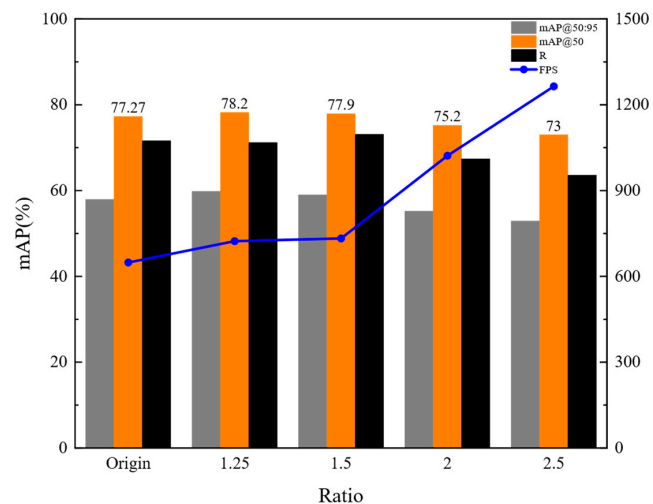


Figure 13. Comparison of detection performance and inference speed of Taylor under different compression ratios.

At $R = 2.0$ and $R = 2.5$, the backbone and neck channels are also reduced, but generally less aggressively than under LAMP. The detection head width remains relatively low and stable. This non-uniform strategy reduces parameter count and computational complexity while retaining feature extraction and multi-scale fusion capacity under the evaluated conditions.

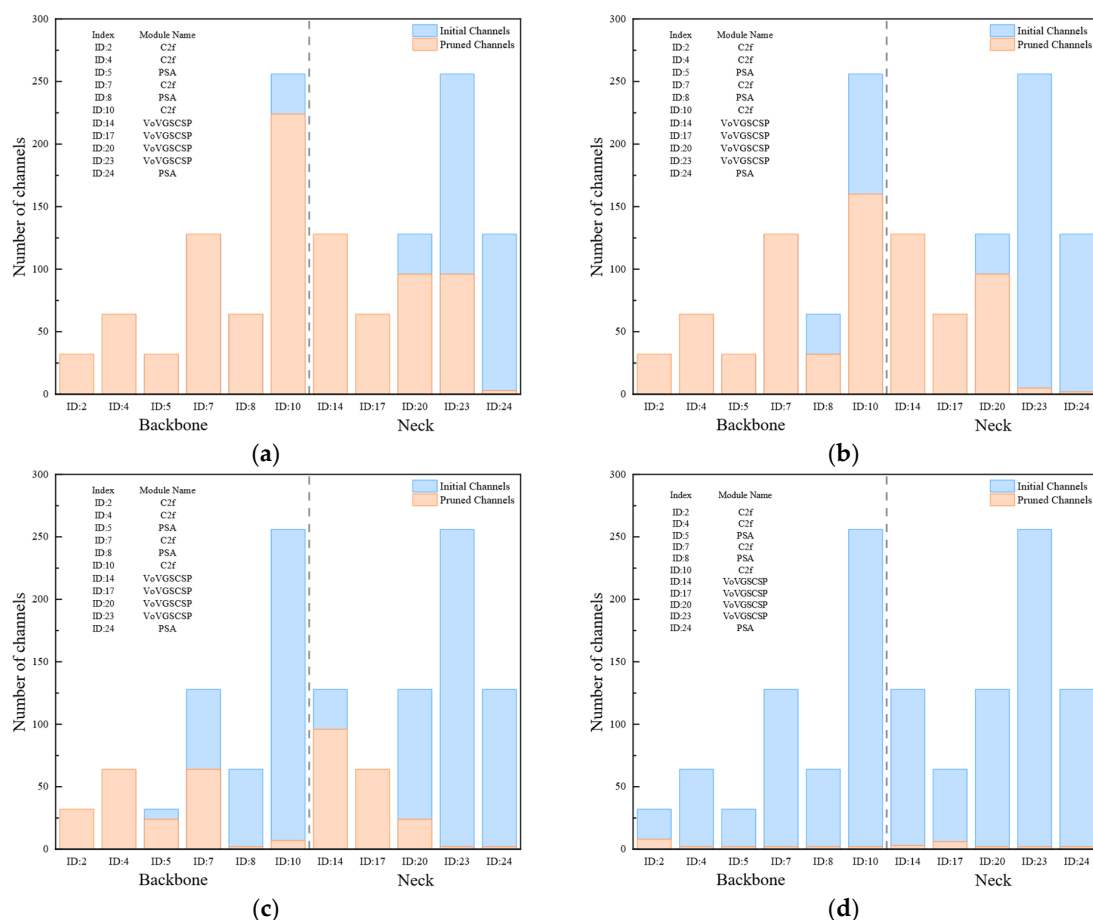


Figure 14. Channel number distribution in the backbone and neck of the Taylor model under different compression ratios: (a) $R = 1.25$; (b) $R = 1.5$; (c) $R = 2.0$; (d) $R = 2.5$. The gray dashed line indicates the boundary between the backbone and neck.

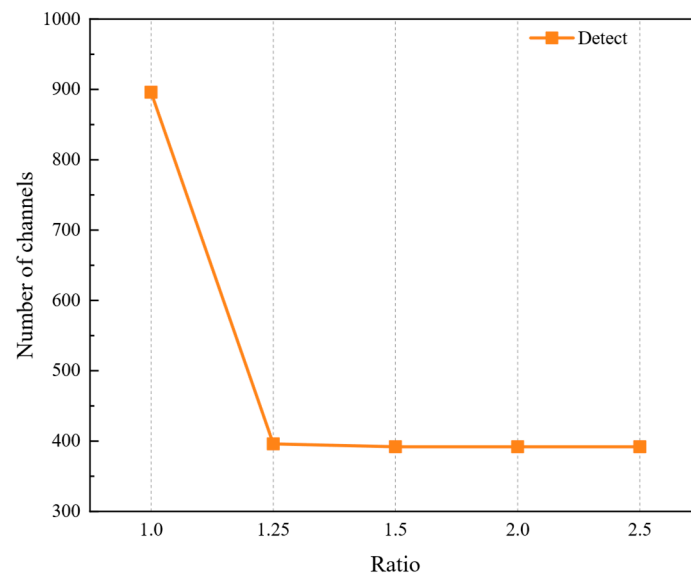


Figure 15. Channel compression trends within the detection head under different Taylor compression ratios.

The three criteria reduce redundancy through different structural mechanisms. DepGraph combines dependency-aware grouping with sparse training and retains relatively stable accuracy, but its acceleration is limited in some settings. LAMP achieves the highest maximum FPS and greater parameter and GFLOPs reduction, with a larger accuracy decrease at high compression. Taylor uses gradient sensitivity to retain more feature extraction and fusion channels while compressing prediction layers, providing a comparatively favorable balance among speed, accuracy, and qualitative localization behavior under the current protocol. These findings should be interpreted in terms of the interaction among pruning criterion, channel redistribution, and defect feature preservation rather than a single metric.

4.2. Comprehensive Detection Performance and Visual Analysis

Structured pruning should balance complexity reduction with detection performance and measured throughput. Accordingly, parameter count, GFLOPs, FPS, and mAP@50 are considered jointly. Class-wise and qualitative behavior are also examined to avoid ranking the methods on a single metric.

4.2.1. mAP@50-FPS Trade-Off Analysis

Figure 16 shows that all three pruning methods increase FPS while producing different degrees of accuracy loss. At low-to-medium compression, their detection accuracy is similar. At higher compression, DepGraph retains greater mAP stability, LAMP reaches the highest maximum FPS but shows a larger accuracy decrease, and Taylor provides a comparatively favorable balance between speed and detection robustness under the tested protocol.

In the low-to-medium compression range (approximately 650–800 FPS), the three pruning methods show relatively small differences in detection accuracy and generally retain the principal detection capability of the baseline model. As the compression ratio increases, however, their performance differences become more pronounced. The three pruning criteria elicited distinct network-restructuring behaviors. DepGraph exhibited a tendency to substantially compress intermediate layers, while LAMP induced more uniformly distributed channel pruning and was correlated with greater loss of high-frequency texture information. In contrast, Taylor pruning displayed an “inverted-pyramid” restructuring pattern.

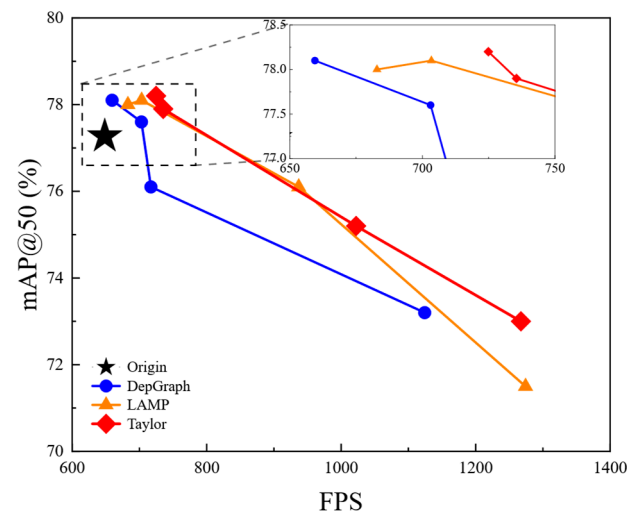


Figure 16. Trade-off analysis between inference speed and detection accuracy under different pruning strategies.

Around $R = 2.0$, the three methods operate within a relatively high-throughput range under the workstation GPU benchmark, but they exhibit different combinations of mAP, FPS, parameter count, and GFLOPs. These metrics are considered jointly to evaluate the trade-off between detection performance, structural compression, and inference efficiency. Because the reported FPS values were obtained using forward-pass measurements on a single NVIDIA RTX 5080 workstation GPU, they should be interpreted as controlled relative-throughput indicators rather than direct measures of field inspection speed or embedded-device performance. Nevertheless, the comparison provides a useful basis for evaluating the deployment potential of the three pruning criteria under the current experimental protocol.

Overall, the three pruning criteria exhibit different performance characteristics. DepGraph shows comparatively stable mAP and class-wise performance in several settings, whereas LAMP achieves the highest maximum throughput and greater reductions in parameter count and GFLOPs, accompanied by a larger accuracy decrease at high compression. Taylor provides competitive inference throughput and a comparatively favorable balance between quantitative detection performance and qualitative localization behavior under the tested conditions. These differences indicate that the pruning criterion should be selected according to deployment priorities, such as accuracy stability, maximum throughput, or localization consistency, rather than through a universal ranking based on a single metric.

4.2.2. Visual and Class-Wise Robustness Analysis

$R = 2.0$ is selected as a representative compression ratio for class-wise AP, PR curves, normalized confusion matrices, and qualitative detection examples because it provides a useful balance between accuracy and throughput for all three methods. Figure 17 shows different class sensitivities. Exposed reinforcement remains comparatively stable because of its distinct texture and geometry, whereas delamination and efflorescence are more sensitive to pruning because their features are diffuse or low-contrast. LAMP shows a larger AP decrease for delamination, Taylor performs less favorably for efflorescence, and DepGraph is more balanced across categories under this setting.

Figure 18 compares the PR curves of the three pruned models at $R = 2.0$. The curves provide information beyond a single mAP value by showing how precision changes with recall. Their overall trends are similar, indicating that the main detection capability of the baseline is retained after pruning. Differences become more pronounced in the high-recall region, where weak-texture and low-contrast defects produce more false positives or missed

detections. Under the tested conditions, Taylor shows a relatively stable precision–recall response, consistent with retaining channels relevant to localization and class separation.

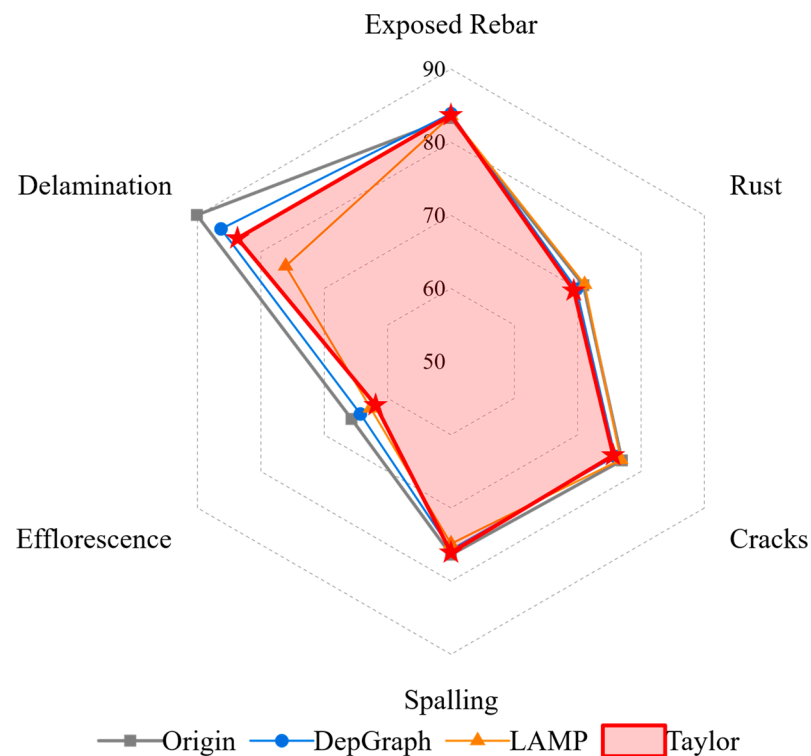


Figure 17. Comparison of the detection performance distribution of the three pruning criteria across six defect categories. The gray squares, blue circles, orange triangles, and red five-pointed stars represent the original model, DepGraph, LAMP, and Taylor, respectively.

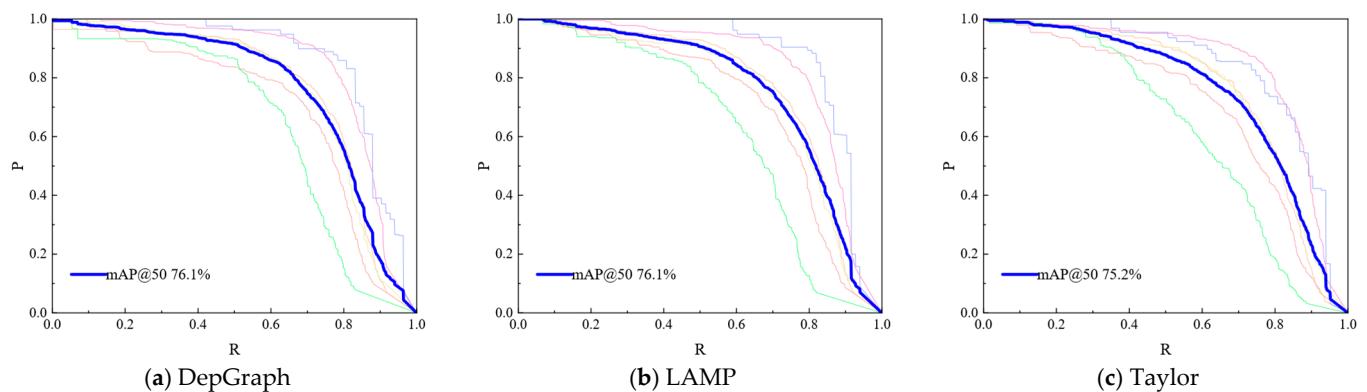


Figure 18. Precision–recall curves of the pruned YOLOv8-PSN models under the representative compression ratio $R = 2.0$. The magenta, red, light-blue, yellow, green, and blue thin curves represent Exposed Rebar, Rust, Cracks, Spalling, Efflorescence, and Delamination, respectively. The thick blue curve represents the overall mAP@50.

Figure 19 shows similar overall confusion patterns for the three methods, suggesting that pruning does not fundamentally change the main category confusion structure of YOLOv8-PSN. Taylor exhibits higher diagonal values across several categories, rather than uniformly across the entire confusion matrix. For the delamination category, Taylor achieves a diagonal value of 0.86, compared to 0.82 for DepGraph and 0.83 for LAMP. Taylor also obtains the highest diagonal values for the exposed reinforcement (0.45) and efflorescence (0.47) categories. The proportions of spalling samples misclassified as background are 0.50 for DepGraph, 0.51 for LAMP, and 0.49 for Taylor, respectively; for efflorescence,

the corresponding misclassification proportions are 0.42, 0.45, and 0.38. Since aggregate diagonal metrics and balanced accuracy statistics were not calculated in this analysis, these category-level observations cannot be interpreted as evidence that Taylor produces the strongest overall confusion matrix.

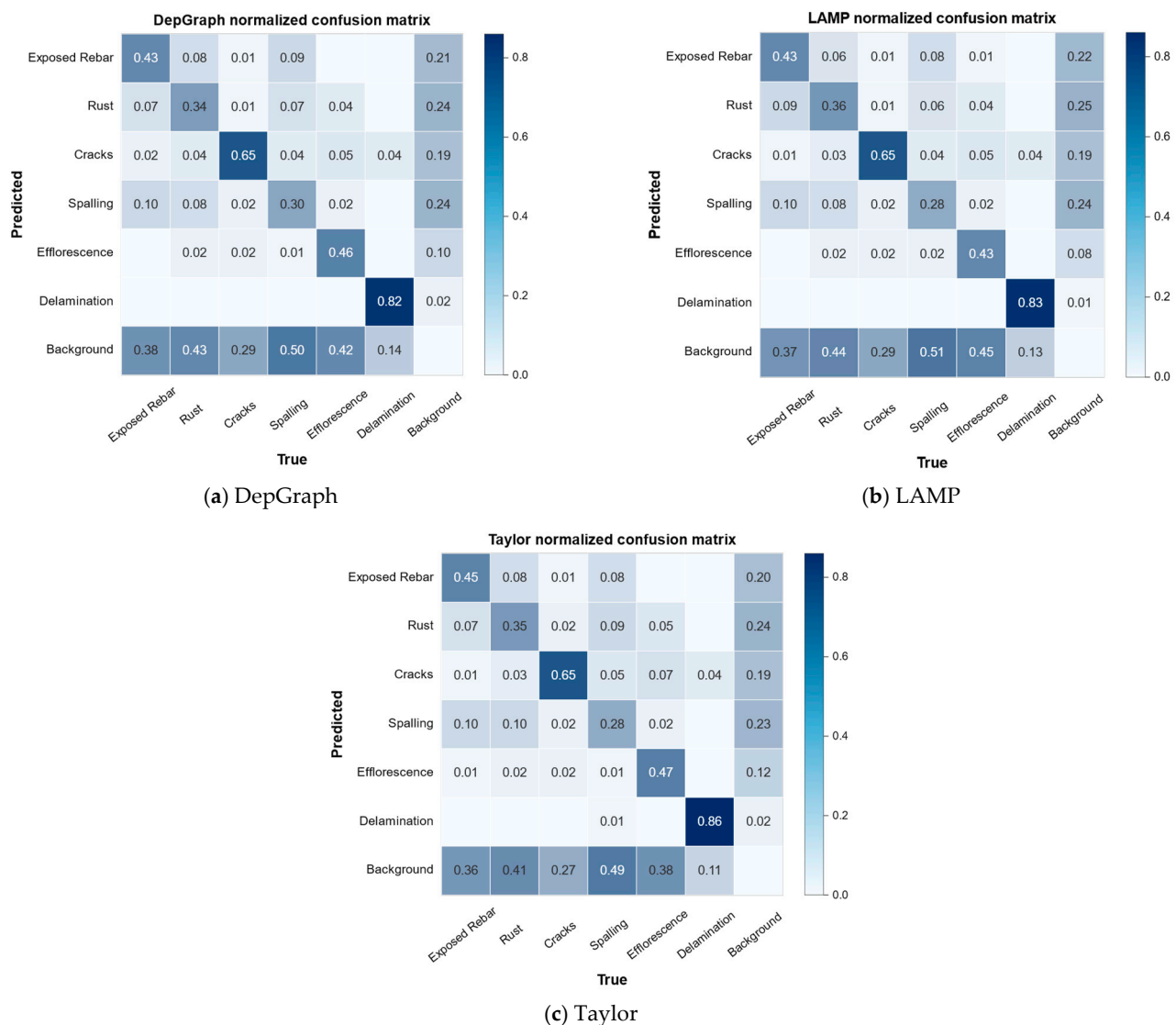


Figure 19. Normalized confusion matrices of the pruned YOLOv8-PSN models under the representative compression ratio $R = 2.0$.

The PR curves and confusion matrices provide complementary category-level evidence at $R = 2.0$. All methods show background confusion for visually ambiguous defects such as rust, spalling, and efflorescence. Taylor has higher diagonal values for several categories and a relatively stable PR response, while DepGraph remains competitive in class-wise balance. These results support a comparatively favorable speed–accuracy–robustness balance for Taylor under the current protocol but do not establish that it is uniformly better across settings.

Beyond dataset-level performance metrics, Figure 20 presents an instance-level qualitative comparison of crack detection under three challenging scenarios: strong illumination, dense target distribution, and low-light conditions. The unpruned YOLOv8-PSN is included as a visual baseline for comparison. We analyze discrepancies across pruned models along three key dimensions: missed detections, confidence score retention, and localization

consistency. Note that the figure only presents selected representative cases, hence these visual observations cannot be interpreted as definitive evidence of overall dataset-level detection accuracy.

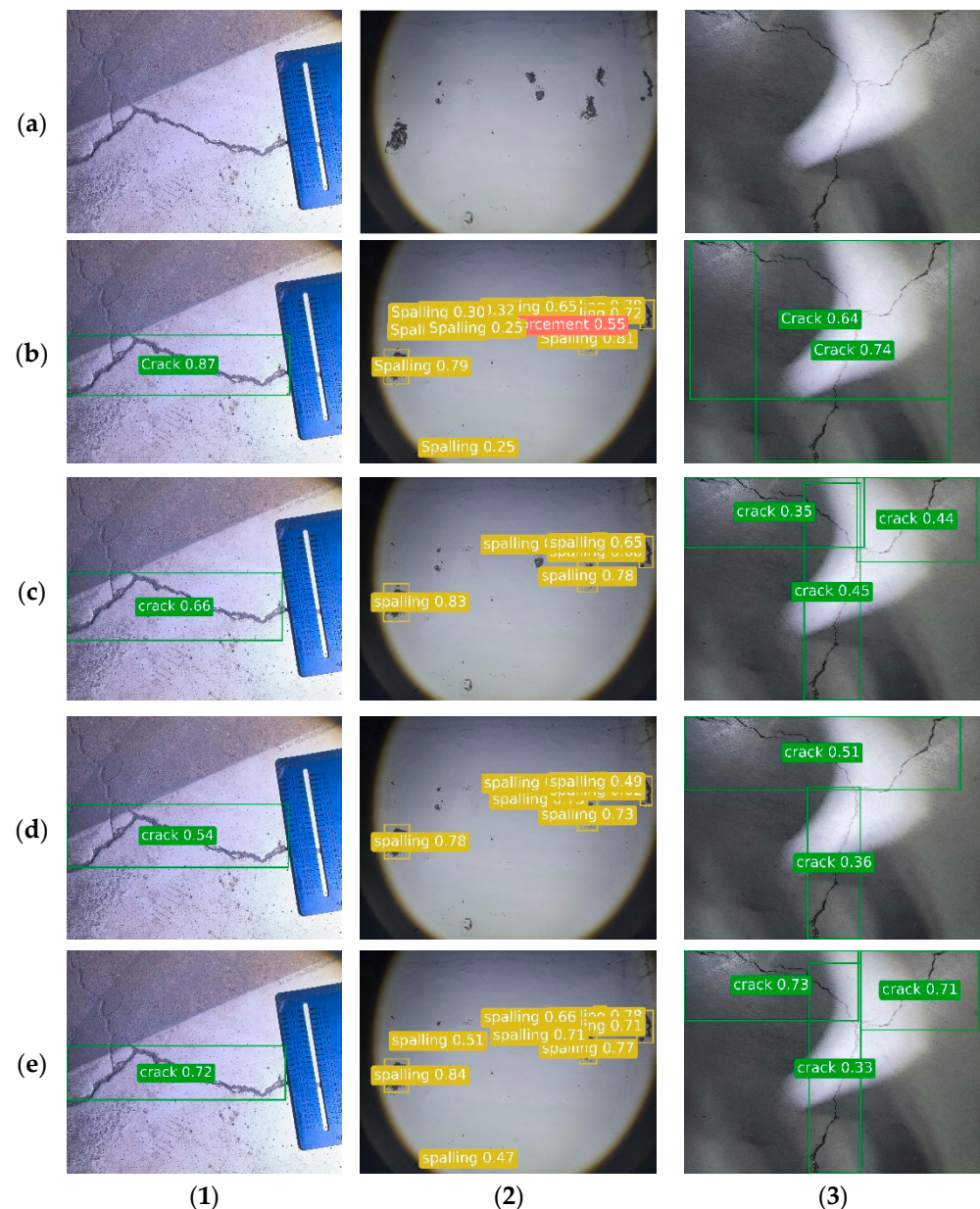


Figure 20. Visual comparison of detection results among the unpruned baseline and the pruned models. Rows represent the different models: (a) Original image; (b) Unpruned YOLOv8-PSN; (c) DepGraph-2.0; (d) LAMP-2.0; (e) Taylor-2.0. Columns (1), (2), and (3) illustrate the strong light interference, dense target, and low-light crack scenarios, respectively.

In the strong-illumination example shown in Figure 20(1), all models detect the crack, but their confidence scores differ. DepGraph (0.66) and LAMP (0.54) yield lower confidence scores than the unpruned baseline (0.87), while Taylor reaches 0.72. For the selected image, Taylor’s method therefore maintains a confidence score closer to the baseline. This observation solely characterizes instance-level confidence retention, and does not indicate improved dataset-level detection accuracy or enhanced structural integrity.

In the dense-target example shown in Figure 20(2), the differences among the pruned models become more pronounced. DepGraph detects fewer targets and misses several small spalling regions. LAMP and Taylor detect most prominent targets but still show

some missed detections. This example suggests reduced sensitivity to dense small targets at high compression, although the conclusion remains qualitative.

In the low-light crack example shown in Figure 20(3), the methods differ mainly in confidence retention and bounding-box continuity. DepGraph covers the crack with three boxes but has lower confidence than Taylor. LAMP covers the crack with two boxes and produces a result similar to that of the unpruned model. Taylor covers the crack with three boxes and retains higher confidence than DepGraph. The unpruned model uses two boxes, indicating more compact localization in this particular image. These observations concern localization behavior in a selected example and are not measures of dataset-level accuracy.

The selected examples suggest that DepGraph exhibits lower confidence under strong illumination and more fragmented localization in low-light conditions. LAMP maintains relatively continuous localization in the low-light example but shows greater confidence reduction under strong illumination. Taylor retains comparatively higher confidence and more consistent localization in these examples. Since Figure 20 only presents selected qualitative case studies, these observations should be interpreted as illustrative evidence for confidence retention and localization consistency, rather than definitive validation of the overall detection accuracy.

These complementary analyses show why mAP alone does not fully characterize the behavior of a lightweight defect detector. The precision–recall (PR) curves show similar overall trends for the three pruned models, although the differences become more pronounced in the high-recall region. The normalized confusion matrices identify missed detections and background confusion as important error sources for weak-texture defects, including rust, spalling, and efflorescence. DepGraph retains competitive quantitative performance but shows confidence reduction or fragmented localization in some selected scenes. LAMP achieves the highest maximum throughput but is more affected by illumination interference and weak-texture defects in the qualitative examples. Taylor does not consistently achieve the highest mAP or FPS; however, it shows relatively stable PR behavior, higher diagonal values for several defect categories, and consistent localization in the selected challenging scenes. Under the tested protocol, these results indicate a comparatively favorable balance among speed, quantitative performance, and localization consistency for Taylor. Nevertheless, the confidence scores and localization behavior observed in individual images should not be interpreted as direct evidence of higher dataset-level detection accuracy.

4.3. Deployment-Oriented Ablation of GPU-Oriented Channel Alignment

To examine the association between channel alignment and workstation GPU inference behavior, models with and without alignment are compared at $R = 2.0$. Aligned and non-aligned models do not share identical final architectures: channel rounding modifies the configuration of retained channels, which can alter the number of parameters, GFLOPs, mAP@50, and FPS concurrently. As a result, this comparison does not strictly isolate alignment as a single independent variable, and instead evaluates the joint effect of the alignment rule and the resulting architecture under the same pruning criterion.

Table 6 reports the architecture and performance changes associated with alignment. For DepGraph, alignment increases the parameter count from 5.26 M to 5.41 M and GFLOPs from 13.6448 to 15.8205; FPS changes from 710.0 to 715.4, while mAP@50 increases from 73.6% to 76.1%. For LAMP, FPS increases from 714.9 to 935.7 and mAP@50 changes from 76.0% to 76.1%, with small changes in parameters and GFLOPs. For Taylor, FPS increases from 955.3 to 1022.0 and GFLOPs decrease from 19.7666 to 14.8759, while mAP@50 decreases from 76.2% to 75.2%. Since alignment changes the channel configuration, the

FPS differences cannot be attributed to channel divisibility alone; they reflect the combined effects of the resulting architecture, computational cost, and accuracy.

Table 6. Ablation results of the GPU-oriented channel alignment strategy on the tested NVIDIA RTX 5080 workstation GPU at $R = 2.0$. “√” indicates that GPU-oriented channel alignment was applied, whereas “×” indicates that the alignment strategy was not applied.

Method	Compression Ratio	Alignment	Parameters (M)	GFLOPs	FPS	mAP@50
DepGraph	2.0	×	5.26	13.6448	710.0	73.6
DepGraph	2.0	√	5.41	15.8205	715.4	76.1
LAMP	2.0	×	5.25	14.4706	714.9	76.0
LAMP	2.0	√	5.21	14.7319	935.7	76.1
Taylor	2.0	×	5.22	19.7666	955.3	76.2
Taylor	2.0	√	5.25	14.8759	1022.0	75.2

Overall, the association between GPU-oriented channel alignment and performance is method-dependent. LAMP and Taylor show larger FPS increases than DepGraph, but the accompanying changes in parameter count, GFLOPs, and mAP@50 differ across methods. These results indicate that regular channel dimensions can be beneficial on the tested NVIDIA RTX 5080 workstation GPU, but they do not establish a universal hardware-aware optimization rule. Different behavior may occur with other GPU architectures, embedded accelerators, precision modes, inference frameworks, or operator implementations.

4.4. Practical Deployment Discussion

The deployment potential of the pruned models should be considered using both structural compression and GPU-oriented inference results. The objective of this study is deployment-oriented lightweight defect detection for underground inspection platforms such as inspection robots and UAVs. In the previous YOLOv8-PSN study [48], the unpruned model achieved 99 FPS under a different evaluation protocol. The present study uses a batch size of 16, FP16 precision, and forward-pass latency on an RTX 5080 workstation. The FPS values from the two studies are therefore not directly comparable, and the current values should be treated as relative throughput indicators rather than field-inspection speed.

After GPU-oriented alignment, LAMP and Taylor reach 935.7 and 1022.0 FPS, respectively, at $R = 2.0$ on the tested workstation. These results indicate potential for high-throughput inference in a workstation GPU environment. They do not constitute embedded-device validation. Jetson Orin, Raspberry Pi AI Kit, and OpenVINO-based platforms were not available in this study, and the effects of alignment may differ on those devices. Future evaluation should include end-to-end latency, energy consumption, memory usage, batch-size-one inference, and field inspection speed.

5. Conclusions

This study uses YOLOv8-PSN as a fixed baseline to compare three representative structured pruning criteria for underground concrete surface defect detection: dependency-graph-based DepGraph, layer-adaptive magnitude-based LAMP, and gradient-sensitive Taylor pruning. The analysis focuses on how these pruning criteria reshape the network and influence model complexity, inference throughput, quantitative detection performance, and qualitative localization behavior. The main conclusions are summarized as follows.

(1) **Speed–accuracy behavior under high compression:** All three pruning methods reduce parameter count and GFLOPs while increasing inference throughput. At $(R = 2.5)$, LAMP achieves the highest FPS of 1274.6, followed closely by Taylor at 1264.3 FPS, whereas DepGraph reaches 1121.6 FPS. Taylor therefore does not provide the highest maximum

acceleration. Nevertheless, under the tested conditions, it achieves competitive throughput together with a comparatively favorable balance of mAP, class-wise performance, and qualitative localization consistency. The acceleration associated with GPU-oriented channel alignment is specific to the NVIDIA RTX 5080 workstation protocol and should not be generalized to other hardware platforms without further validation.

(2) Differences in pruning mechanisms and structural reshaping: The three criteria produce distinct channel redistribution patterns. DepGraph combines dependency-aware grouping with group-level sparse training and preferentially compresses intermediate feature fusion layers while retaining relatively more channels in the shallow backbone and detection head. This pattern contributes to relatively stable detection performance at low-to-medium compression ratios. LAMP applies normalized magnitude-based scoring and produces a more uniform within-layer pruning pattern. Although it achieves substantial complexity reduction and the highest maximum throughput, it can remove low-magnitude channels that remain relevant to weak-texture or high-frequency defect features, resulting in a larger accuracy decrease under aggressive compression. Taylor incorporates task-related gradient information and produces a gradient-guided inverted-pyramid pattern. It removes more redundancy from the detection head while retaining comparatively greater feature representation capacity in the backbone and feature fusion layers. This structural behavior is favorable within the current experimental setting but should not be assumed to apply to other datasets, detector architectures, or pruning implementations.

(3) Complementary quantitative and qualitative evaluation: The results show that mAP alone does not fully characterize the behavior of a lightweight defect detector. The precision–recall curves reveal similar overall trends among the three methods, although their differences become more pronounced in the high-recall region. The normalized confusion matrices identify missed detections and background confusion as important error sources for weak-texture defects such as rust, spalling, and efflorescence. DepGraph maintains competitive mAP and relatively balanced class-wise performance in several settings. LAMP provides the highest maximum throughput but exhibits a larger accuracy decrease at high compression. Taylor does not consistently achieve the highest mAP or FPS, but it shows relatively stable precision–recall behavior, higher diagonal values for several defect categories, and consistent localization in the selected challenging examples. The confidence scores and localization results shown in Figure 20 are instance-level observations. They are used to characterize confidence retention and localization consistency and should not be interpreted as direct evidence of higher dataset-level detection accuracy.

Overall, Taylor provides a comparatively favorable balance among inference throughput, quantitative detection performance, and qualitative localization behavior for the evaluated underground-structure defect dataset and workstation GPU protocol. This conclusion is limited to the current dataset, pruning implementation, compression settings, and NVIDIA RTX 5080 inference environment. It does not imply that Taylor pruning is universally preferable. DepGraph may be more appropriate when mAP stability and category balance are prioritized, whereas LAMP may be preferred when maximum inference throughput is the primary objective. The pruning criterion should therefore be selected according to the requirements of the intended application.

This study has several limitations. First, all experiments were conducted using the proprietary underground-structure defect dataset established in our previous work [48], and cross-dataset generalization was not evaluated on public crack or concrete defect benchmarks. Second, the reported FPS values were obtained using a single NVIDIA RTX 5080 workstation GPU and do not represent direct edge device or field inspection performance. Third, channel alignment changes the final retained-channel configuration. Therefore, the ablation comparison cannot strictly isolate the effect of channel divisibility

from simultaneous changes in parameter count, GFLOPs, and detection accuracy. Fourth, wall-clock pruning time, fine-tuning time, and total GPU-hours were not systematically recorded under a dedicated profiling protocol and therefore cannot be reported reliably. Fifth, this study focuses on structured pruning and does not evaluate complementary lightweight techniques such as quantization, knowledge distillation, neural architecture search, or unstructured pruning.

Future work will evaluate the pruned models on representative edge platforms, including Jetson Orin, Raspberry Pi AI Kit, and OpenVINO-based devices. Cross-dataset robustness will be examined using suitable public defect benchmarks. Future experiments will also record pruning time, fine-tuning time, GPU-hours, memory consumption, and energy consumption under a reproducible profiling protocol. In addition, structured pruning will be combined with quantization or knowledge distillation to further improve deployment efficiency.

Author Contributions: Conceptualization, K.L. and C.M.; methodology, K.L. and C.M.; software, X.N.; validation, M.Z. and Y.M.; formal analysis, M.Z.; investigation, K.L.; data curation, X.N.; writing—original draft preparation, K.L. and X.N.; writing—review and editing, D.Z. and C.M.; visualization, Y.M.; supervision, D.Z. and C.M.; funding acquisition, D.Z.; project administration, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2022YFC3801600.

Data Availability Statement: The proprietary concrete surface defect dataset is available from the corresponding author upon reasonable request. Access is limited to non-commercial academic research and is subject to institutional and project owner approval and applicable confidentiality restrictions. Corresponding annotations may be provided with the approved image data. Requests for selected de-identified sample data or trained model weights will be considered separately where permitted.

Conflicts of Interest: Author Kai Li was employed by the company China Academy of Building Research. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

YOLO	You Only Look Once
PSA	Pyramid Squeeze Attention
VoVGSCSP	A cross-stage module combining GS bottleneck and OSA
GSConv	A lightweight convolution module for Slim-neck
CSP	Cross-Stage Partial
DepGraph	Dependency Graph
LAMP	Layer-Adaptive Magnitude-based Pruning
mAP	Mean Average Precision
FPS	Frames Per Second
GFLOPs	Giga Floating-point Operations Per Second
IoU	Intersection over Union

References

1. Prakash, V.; Debono, C.J.; Musarat, M.A.; Borg, R.P.; Seychell, D.; Ding, W.; Shu, J. Structural health monitoring of concrete bridges through artificial intelligence: A narrative review. *Appl. Sci.* **2025**, *15*, 4855. [\[CrossRef\]](#)
2. Chen, W.; Hou, J.; Wang, Y.; Yu, M. Visualization analysis of concrete crack detection in civil engineering infrastructure based on knowledge graph. *Case Stud. Constr. Mater.* **2024**, *21*, e03711. [\[CrossRef\]](#)
3. Menendez, E.; Victores, J.G.; Montero, R.; Martínez, S.; Balaguer, C. Tunnel structural inspection and assessment using an autonomous robotic system. *Autom. Constr.* **2018**, *87*, 117–126. [\[CrossRef\]](#)

4. Sjölander, A.; Belloni, V.; Ansell, A.; Nordström, E. Towards automated inspections of tunnels: A review of optical inspections and autonomous assessment of concrete tunnel linings. *Sensors* **2023**, *23*, 3189. [[CrossRef](#)] [[PubMed](#)]
5. Zakaria, M.; Karaaslan, E.; Catbas, F.N. Advanced bridge visual inspection using real-time machine learning in edge devices. *Adv. Bridge Eng.* **2022**, *3*, 27. [[CrossRef](#)]
6. Ai, D.; Jiang, G.; Lam, S.-K.; He, P.; Li, C. Computer vision framework for crack detection of civil infrastructure—A review. *Eng. Appl. Artif. Intell.* **2023**, *117*, 105478. [[CrossRef](#)]
7. Hu, H.; Fang, H.; Wang, N.; Ma, D.; Dong, J.; Li, B.; Di, D.; Zheng, H.; Wu, J. Defects identification and location of underground space for ground-penetrating radar based on deep learning. *Tunn. Undergr. Space Technol.* **2023**, *140*, 105278. [[CrossRef](#)]
8. Lin, S.; Ye, H.; Tan, D.; Wang, J.; Yin, J. Identification of defects in underground structures using machine-learning-aided distributed fiber-optic sensing. *J. Rock Mech. Geotech. Eng.* **2025**, *17*, 2194–2207. [[CrossRef](#)]
9. Wang, H.; Xie, J.; Fu, J.; Zhang, C.; Chen, D.; Zhu, Z.; Zhang, X. Rapid acquisition and surface defect recognition based on panoramic images of small-section hydraulic tunnels. *Undergr. Space* **2025**, *21*, 270–290. [[CrossRef](#)]
10. Shalaby, Y.M.; Badawy, M.; Ebrahim, G.A.; Abdelalim, A.M. Condition assessment of concrete structures using an automated crack-detection method for different concrete surface types based on image processing. *Discov. Civ. Eng.* **2024**, *1*, 81. [[CrossRef](#)]
11. Flotzinger, J.; Rösch, P.J.; Braml, T. dacl10k: Benchmark for semantic bridge damage segmentation. In *Proceedings of the 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 8611–8620. [[CrossRef](#)]
12. Li, R.; Zhao, L.; Wei, H.; Hu, G.; Xu, Y.; Ouyang, B.; Tan, J. Multi-defect-type beam bridge dataset: GYU-DET. *Sci. Data* **2025**, *12*, 1101. [[CrossRef](#)] [[PubMed](#)]
13. Rajadurai, R.-S.; Kang, S.-T. Automated vision-based crack detection on concrete surfaces using deep learning. *Appl. Sci.* **2021**, *11*, 5229. [[CrossRef](#)]
14. Kim, B.; Choi, S.-W.; Hu, G.; Lee, D.-E.; Serfa Juan, R.O. An automated image-based multivariant concrete defect-recognition method using a convolutional neural network with an integrated pooling module. *Sensors* **2022**, *22*, 3118. [[CrossRef](#)] [[PubMed](#)]
15. Zhang, Y.; Chow, C.L.; Lau, D. Artificial intelligence-enhanced non-destructive defect detection for civil infrastructure. *Autom. Constr.* **2025**, *171*, 105996. [[CrossRef](#)]
16. Xu, J.; Wang, S.; Han, R.; Wu, X.; Zhao, D.; Zeng, X.; Yin, R.; Han, Z.; Liu, Y.; Shu, S. Crack segmentation and quantification in concrete structures using a lightweight YOLO model based on pruning and knowledge distillation. *Expert Syst. Appl.* **2025**, *283*, 127834. [[CrossRef](#)]
17. Dang, D.; Gong, L.; Jin, C.; Cui, X.; Yang, T.; Qin, J. SMI-YOLOv8: Intelligent detection of tunnel lining cracks via multiscale feature-attention fusion. *Measurement* **2026**, *257*, 118833. [[CrossRef](#)]
18. Duan, S.; Zhang, M.; Qiu, S.; Xiong, J.; Zhang, H.; Li, C.; Jiang, Q.; Kou, Y. Tunnel lining crack-detection model based on improved YOLOv5. *Tunn. Undergr. Space Technol.* **2024**, *147*, 105713. [[CrossRef](#)]
19. Jin, Q.; Yue, Z.; Zhou, X.; Xue, K.; Liu, H.; Chen, J. YOLOv11-based enhanced framework for robust blasthole detection in intelligent tunnel environments. *J. Comput. Civ. Eng.* **2026**, *40*, 04026036. [[CrossRef](#)]
20. Chen, X.; Zhang, K.; Wang, W.; Hu, K.; Xu, Y. Intelligent identification of tunnel water leakage based on super-resolution reconstruction and triple attention. *Measurement* **2024**, *225*, 114009. [[CrossRef](#)]
21. Feng, S.J.; Feng, Y.; Zhang, X.L.; Chen, Y.H. Deep learning with visual explanations for leakage-defect segmentation of metro shield tunnels. *Tunn. Undergr. Space Technol.* **2023**, *136*, 105107. [[CrossRef](#)]
22. Duan, S.; Zhang, M.; Xiong, J.; Qiu, S.; Xie, Z.; Jiang, Q.; Kou, Y. DSYOLO: A dynamic snake convolution-based segmentation model for tunnel lining cracks in complex environments. *Eng. Struct.* **2025**, *344*, 121325. [[CrossRef](#)]
23. Yang, K.; Bao, Y.; Li, J.; Fan, T.; Tang, C. Deep-learning-based YOLO for crack segmentation and measurement in metro tunnels. *Autom. Constr.* **2024**, *168*, 105818. [[CrossRef](#)]
24. Ma, Y.; Cui, B.; Jiang, W.; Yan, L.; Cui, X. CM CAB: An insertable block for concrete crack detection based on masked convolution attention. *J. Comput. Civ. Eng.* **2026**, *40*, 04026027. [[CrossRef](#)]
25. Mayya, A.M.; Alkayem, N.F. Triple-stage crack detection in stone masonry using a YOLO ensemble, MobileNetV2 U-Net, and spectral clustering. *Autom. Constr.* **2025**, *172*, 106045. [[CrossRef](#)]
26. Wang, P.; Shi, G. Image segmentation of tunnel water-leakage defects in complex environments using an improved U-Net model. *Sci. Rep.* **2024**, *14*, 24286. [[CrossRef](#)] [[PubMed](#)]
27. Man, K.; Liu, R.; Liu, X.; Song, Z.; Liu, Z.; Cao, Z.; Wu, L. Water leakage and crack identification in tunnels based on transfer learning and convolutional neural networks. *Water* **2022**, *14*, 1462. [[CrossRef](#)]
28. Lei, Y.; Zhu, K.; Jiang, B.; Wang, Y.; Jia, F.; Wang, Z.; Qi, F.; Qu, Q. Task-specific YOLO optimization for railway tunnel cracks and water leakage: Benchmarking and lightweight enhancement. *Comput. Mater. Contin.* **2026**, *87*, 41. [[CrossRef](#)]
29. Liang, H.-Y.; Shen, S.-L.; Zhou, A.; Zhao, W.-W. Automated detection and quantification of leakage areas in shield tunnel linings using laser-scanning data and a deep-learning network. *Eng. Appl. Artif. Intell.* **2025**, *160*, 111930. [[CrossRef](#)]

30. Liu, Z.; Gao, X.; Yang, Y.; Xu, L.; Wang, S.; Chen, N.; Wang, Z.; Kou, Y. EDT-Net: A lightweight tunnel water-leakage detection network based on LiDAR point-cloud intensity images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2025**, *18*, 7334–7346. [\[CrossRef\]](#)
31. Zhou, Z.; Zhang, J.; Gong, C.; Wu, W. Automatic tunnel lining crack detection via deep learning with generative-adversarial-network-based data augmentation. *Undergr. Space* **2023**, *9*, 140–154. [\[CrossRef\]](#)
32. Long, X.; Teng, J.; Zhu, Z.; Zhao, S.; Pu, M.; Shi, R.; Lv, Y.; Li, J.; Jing, G. Co-MixPL: An optimized semi-supervised learning method for tunnel water-leakage detection. *Adv. Eng. Inform.* **2026**, *71*, 104263. [\[CrossRef\]](#)
33. Li, B.; Chu, X.; Lin, F.; Wu, F.; Jin, S.; Zhang, K. A highly efficient tunnel lining crack-detection model based on Mini-UNet. *Sci. Rep.* **2024**, *14*, 28234. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Liu, K.-Q.; Zhong, S.-S.; Zhao, K.; Song, Y. Motion control and positioning system of a multi-sensor tunnel-defect inspection robot: From methodology to application. *Sci. Rep.* **2023**, *13*, 232. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Zhang, R.; Wang, C.; Li, Z. Automatic inspection and assessment of a cross-passage twin tunnel using a UAV. *Autom. Constr.* **2025**, *175*, 106200. [\[CrossRef\]](#)
36. Xu, Z.; Chen, B.; Zhan, X.; Xiu, Y.; Suzuki, C.; Shimada, K. A vision-based autonomous UAV inspection framework for unknown tunnel construction sites with dynamic obstacles. *IEEE Robot. Autom. Lett.* **2023**, *8*, 4983–4990. [\[CrossRef\]](#)
37. Sun, H.; Guo, S.; Pan, X.; Shen, Q.; Xu, Y.; Ma, J.; Qu, Z. MCP-YOLO: A pruned edge-aware detection framework for real-time insulator-defect inspection via UAV. *Sensors* **2025**, *25*, 7049. [\[CrossRef\]](#) [\[PubMed\]](#)
38. Jani, M.; Fayyad, J.; Al-Younes, Y.; Najjaran, H. Model compression methods for YOLOv5: A review. *arXiv* **2023**, arXiv:2307.11904. [\[CrossRef\]](#)
39. Cai, Y.; Li, H.; Yuan, G.; Niu, W.; Li, Y.; Tang, X.; Ren, B.; Wang, Y. YOLObile: Real-time object detection on mobile devices via compression–compilation co-design. *arXiv* **2020**, arXiv:2009.05697. [\[CrossRef\]](#)
40. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. Scaled-YOLOv4: Scaling cross-stage partial networks. *arXiv* **2020**, arXiv:2011.08036. [\[CrossRef\]](#)
41. Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1680–1716. [\[CrossRef\]](#)
42. Zhu, A.; Xie, J.; Wang, B.; Guo, H.; Guo, Z.; Wang, J.; Xu, L.; Zhu, S.; Yang, Z. Lightweight defect-detection algorithm for tunnel linings based on knowledge distillation. *Sci. Rep.* **2024**, *14*, 27178. [\[CrossRef\]](#) [\[PubMed\]](#)
43. Ou, J.; Zhang, J.; Li, H.; Duan, B. An improved YOLOv10-based lightweight multiscale feature-fusion model for road-defect detection and its applications. *Adv. Eng. Softw.* **2025**, *208*, 103976. [\[CrossRef\]](#)
44. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. DepGraph: Towards any structural pruning. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 16091–16101. [\[CrossRef\]](#)
45. Lee, J.; Park, S.; Mo, S.; Ahn, S.; Shin, J. Layer-adaptive sparsity for magnitude-based pruning. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 3–7 May 2021*.
46. Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 11264–11272. [\[CrossRef\]](#)
47. Tang, X.-W.; Huang, X.-L.; Hu, F.; Shi, Q. Human-perception-oriented pseudo-analog video transmissions with deep learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9896–9909. [\[CrossRef\]](#)
48. Ma, C.; Nie, X.; Fan, P.; Wang, G. A developed YOLOv8 model for the rapid detection of surface defects in underground structures. *Buildings* **2026**, *16*, 610. [\[CrossRef\]](#)
49. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1398–1406. [\[CrossRef\]](#)
50. Lyu, Z.; Yu, T.; Pan, F.; Zhang, Y.; Luo, J.; Zhang, D.; Chen, Y.; Zhang, B.; Li, G. A survey of model-compression strategies for object detection. *Multimed. Tools Appl.* **2024**, *83*, 48165–48236. [\[CrossRef\]](#)
51. Xu, S.; Wang, X.; Lv, W.; Chang, Q.; Cui, C.; Deng, K.; Wang, G.; Dang, Q.; Wei, S.; Du, Y.; et al. PP-YOLOE: An evolved version of YOLO. *arXiv* **2022**, arXiv:2203.16250. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.