

Article

KDSP-DEA-YOLO: Knowledge-Distillation-Guided Structured Pruning for Lightweight Industrial Surface Defect Detection

Hong Ke^{1,*}
¹ Guizhou University of Finance and Economics, Guiyang, China

* Correspondence: Hong Ke, Guizhou University of Finance and Economics, Guiyang, China

Abstract: Small surface defects on industrial components are characterized by small scale, low contrast, weak texture, and strong background interference. To improve deployment efficiency while preserving detection accuracy, this study proposes KDSP-DEA-YOLO based on the DEA-YOLOv8n teacher model. A heterogeneous lightweight student is first constructed using GhostConv, C2f_Ghost_Lite, a lightweight EMA branch, and DFF_Lite, reducing the pre-pruning parameter count to 4.32 M. A three-stage pipeline is then adopted: sparsity-aware pre-distillation, dependency-aware structured pruning, and distillation-assisted fine-tuning, with joint supervision from outputs, multiscale features, and bounding boxes. Based on the current baseline and compression trends, the predicted 20% pruning configuration contains 3.61 M parameters and requires 8.84 G FLOPs, while achieving approximately 69.8% mAP@0.5 after fine-tuning. This represents a 39.8% reduction in parameters relative to the 6.0 M teacher. The framework provides a verifiable route toward regularized compression and edge deployment for small industrial defect detection.

Keywords: industrial surface defect detection; small-object detection; knowledge distillation; structured pruning; lightweight network

1. Introduction

Industrial surface defect detection is a key component of intelligent manufacturing quality control. Tiny scratches, dents, and local texture anomalies often occupy only a very small region of an image and are accompanied by low contrast and complex backgrounds. Existing reviews indicate that industrial visual inspection continues to face persistent challenges involving scale variation, limited samples, background interference, and the trade-off between accuracy and efficiency [1,2].

Object detection has evolved from two-stage R-CNN methods to SSD, YOLO, FPN, and DETR-based frameworks [3-7]. Owing to their favorable speed-accuracy trade-off, YOLO detectors are particularly suitable for online inspection, while the anchor-free head, C2f structure, and decoupled prediction branches of YOLOv8 provide a strong basis for lightweight detection [8-11]. Our previous DEA-YOLOv8n further introduced SIoU, a dual backbone, DFF, and EMA, increasing mAP@0.5 from 62.9% to 70.3%, but also increasing the parameter count from 3.2 M to 6.0 M [12,13].

For deployment, structured channel pruning directly removes convolutional channels while preserving dense tensor structures, making practical acceleration on general-purpose hardware more attainable than with unstructured sparsity [14-17]. Knowledge distillation transfers soft outputs and intermediate representations from a teacher network to a lightweight student, helping to alleviate accuracy degradation without increasing inference cost [18-23].

Accordingly, this study moves knowledge distillation ahead of hard pruning and continues it after pruning. Stage I performs multiscale pre-distillation and BN sparsification on the complete student topology; Stage II synchronously prunes channels according to cross-branch topological dependencies; and Stage III applies decayed KD

Received: 24 July 2026

Revised: 10 September 2026

Accepted: 23 September 2026

Published: 28 September 2026



Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

fine-tuning to the fixed compressed topology. In this way, knowledge preservation, structural removal, and accuracy recovery are integrated into a unified pipeline.

1.1. DEA-YOLOv8n Model

DEA-YOLOv8n serves as the teacher network in this study. It replaces CIOU with SIOU to improve the optimization direction of bounding-box regression, employs a dual-backbone design to extract complementary spatial and semantic features, and embeds EMA in the second branch to enhance local texture and global context modeling [12,13]. Three-scale features are fused by DFF before being fed into the neck and detection head, as shown in Figure 1.

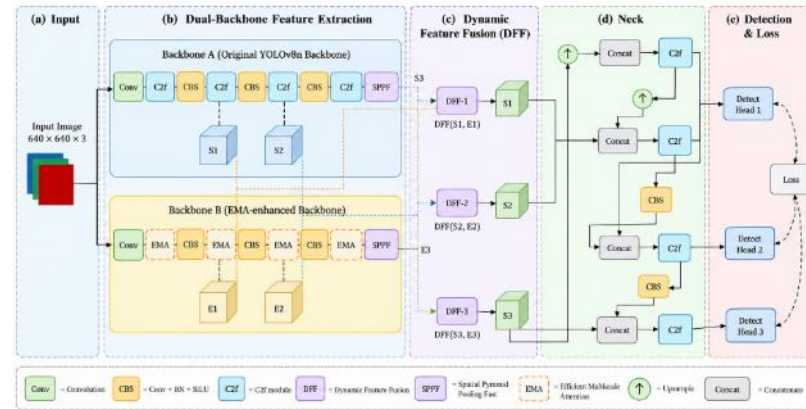


Figure 1. Architecture of the DEA-YOLOv8n Teacher Model

Previous experiments used 1,990 industrial component images at 1080×1920 resolution, resized to 640×640 and labeled as damage. The dataset was split into training, validation, and test sets at a ratio of 80%/10%/10%. DEA-YOLOv8n achieved 78.9% precision, 65.3% recall, 70.3% mAP@0.5, and 29.0% mAP@0.5:0.95, but required 6.0 M parameters, motivating further compression, the comparison with baseline Yolov8n is shown in Table 1.

Table 1. Comparison between DEA-YOLOv8n and the Baseline Model

Model	Precision/%	Recall/%	mAP@0.5/%	mAP@0.5:0.95/%	Params/M	FLOPs/G
YOLOv8n	70.4	59.7	62.9	23.5	3.2	8.9
DEA-YOLOv8n	78.9	65.3	70.3	29.0	6.0	7.9

1.2. Contributions

The main contributions are as follows: (1) a heterogeneous student is constructed using GhostConv, C2f_Ghost_Lite, a lightweight EMA branch, and DFF_Lite, reducing the parameter count to 4.32 M before pruning; (2) joint distillation of P3/P4/P5 multiscale features, outputs, and bounding boxes is introduced; (3) dependency groups are established for synchronized pruning across dual branches, Concat operations, residual paths, and DFF; and (4) decayed KD fine-tuning is applied after pruning, with ablation analyses of pruning ratio, training order, and distillation composition used to examine deployment trade-offs.

2. Related Work

2.1. Industrial Surface Defect Detection and Small-Object Detection

Industrial surface defect detection includes supervised classification, localization, and segmentation, as well as anomaly detection trained only on normal samples. MVtec AD established a widely used industrial anomaly benchmark [24], while the

segmentation-based method of Tabernik et al. demonstrated the feasibility of learning industrial surface anomalies from limited samples [25]. Real-IAD further highlighted domain differences between realistic multiview scenes and public benchmarks, and Spot-the-Difference pre-training demonstrated the value of self-supervised anomaly representations [26,27]. For task-specific detectors, LF-YOLO, HyperDefect-YOLO, ETDNet, and SLF-YOLO improve industrial defect detection through lightweight convolution, cross-scale relation modeling, Transformer-based representation, and lightweight neck design, respectively [28-31].

Small objects depend on the cooperation of high-resolution shallow details and deep semantic features. FPN performs multiscale fusion through top-down and lateral connections, while PANet further introduces bidirectional path aggregation [6,32]. For attention modeling, SE, CBAM, and Coordinate Attention enhance channel, spatial, and directional positional representation, respectively [33-35]. Therefore, this study retains P3/P4/P5 feature supervision and assigns a larger distillation weight to P3.

2.2. Knowledge Distillation for Object Detection

Hinton et al. introduced temperature-softened knowledge distillation, while FitNets and Attention Transfer extended supervision to intermediate representations [18-20]. For object detectors, DeFeat distinguishes foreground and background features, FGD transfers both local and global knowledge, and ScaleKD emphasizes the importance of scale-aware transfer for small objects [21-23]. Building on these ideas, this study does not seek to introduce a new general-purpose KD loss; instead, KD is used both to shape channel importance before pruning and to recover accuracy after pruning.

2.3. Structured Pruning

Structured pruning directly removes convolutional kernels, channels, or network blocks while retaining a regular dense architecture, making it more suitable for deployment on general-purpose CPUs, GPUs, and embedded devices [15-17]. Representative approaches include filter-norm pruning, LASSO-based channel selection with feature reconstruction, and Taylor-based importance estimation [15-17].

Network Slimming uses BN scaling factors γ with L1 regularization to induce channel-level sparsity [14]. When directly applied to a dual-branch detector, however, it may remove channels carrying weak texture information and may also create dimensional dependency conflicts across Concat operations, residual paths, and cross-scale fusion. This study addresses these problems through sparsity-aware pre-distillation and dependency-group synchronized pruning.

3. Method

3.1. Lightweight Student

A heterogeneous student is first constructed. The first branch replaces high-cost convolutions with GhostConv and C2f_Ghost_Lite; the second branch retains the core EMA mechanism within a lightweight C2f structure; multiscale fusion is performed by DFF_Lite; and the neck continues to use C2f_Ghost_Lite. This design reduces the pre-pruning parameter count to 4.32 M.

3.1.1. GhostConv Lightweight Convolution

GhostNet observes that feature maps generated by standard convolution contain substantial redundancy. It first generates a small number of intrinsic feature maps and then produces additional ghost features through low-cost linear transformations [36]. In this study, GhostConv(Figure 2) is used for downsampling in both backbones and in the detection neck to reduce duplicated computation across the parallel branches.

(a) GhostConv

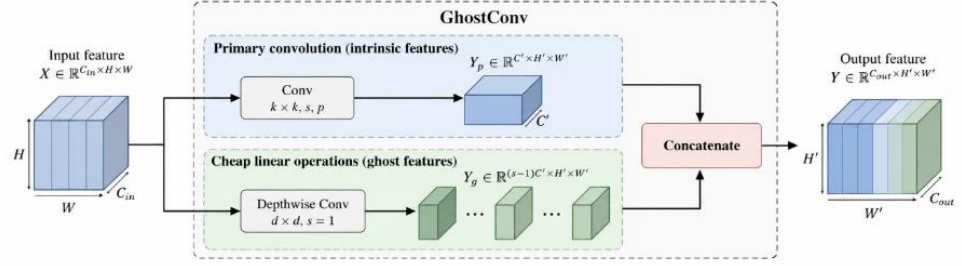


Figure 2. Architecture of GhostConv

$$P_{\text{conv}} = k^2 C_{\text{in}} C_{\text{out}} \#(1)$$

The main multiply-add computational cost is:

$$M_{\text{conv}} = HWk^2 C_{\text{in}} C_{\text{out}} \quad (2)$$

GhostConv first uses standard convolution to generate m intrinsic feature maps:

$$Y' = X * f, Y' \in \mathbb{R}^{m \times H' \times W'} \quad (3)$$

where f denotes the primary convolution. Several low-cost linear transformations

$\Phi_{i,j}(\cdot)$ are then applied to each intrinsic feature map to generate additional features:

$$y_{i,j} = \Phi_{i,j}(y'_i), \quad i = 1, \dots, m \quad (4)$$

The final output is obtained by concatenating the intrinsic features and ghost features:

$$Y = \text{Concat}(Y', Y_{\text{ghost}}) \quad (5)$$

If each intrinsic feature generates $s - 1$ additional features and the low-cost operation uses a $d \times d$ depthwise convolution, the approximate computational cost is:

$$M_{\text{ghost}} \approx HW(k^2 C_{\text{in}} m + d^2 m(s - 1)) \quad (6)$$

By replacing part of the high-dimensional convolutional mappings with low-cost depthwise transformations, GhostConv substantially reduces the parameter count and computational cost, making it suitable for the parallel backbones used here.

3.1.2. C2f_Ghost_Lite

C2f_Ghost_Lite retains the feature splitting, local residual connections, and feature concatenation of C2f, while replacing high-cost internal convolutional units with lightweight Ghost units (Figure 3). It is used in the P2-P5 stages of the first backbone and at feature aggregation nodes in the detection head.

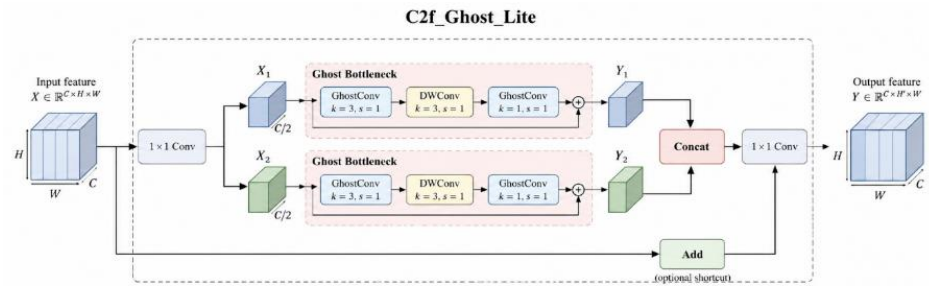


Figure 3. Architecture of C2f_Ghost_Lite

Given an input X a 1×1 convolution first performs channel mapping and splits the result into two subfeatures:

$$[X_1, X_2] = \text{Split}(\phi_{1 \times 1}(X)) \quad (7)$$

For the i th lightweight block:

$$Z_i = \mathcal{G}_i(Z_{i-1}), \quad Z_0 = X_2 \quad (8)$$

where $\mathcal{G}_i(\cdot)$ denotes the Ghost bottleneck. The bypass feature and outputs from all stages are finally concatenated:

$$Y = \phi_{1 \times 1}(\text{Concat}[X_1, X_2, Z_1, \dots, Z_n]) \quad (9)$$

This structure preserves the multi-depth feature reuse capability of C2f while transferring redundant mappings to lower-cost Ghost operations.

3.1.3. C2f_EMA_Lite

The second backbone uses C2f_EMA_Lite, which embeds EMA(Figure 4) into the lightweight C2f pathway and enhances weak textures through grouped modeling and cross-spatial interaction. Studies on SE, CBAM, Coordinate Attention, and Transformer attention indicate that channel-spatial modeling and long-range dependency modeling can improve feature selection [33-35,37]. EMA is adopted here to preserve local and global information with limited additional overhead.

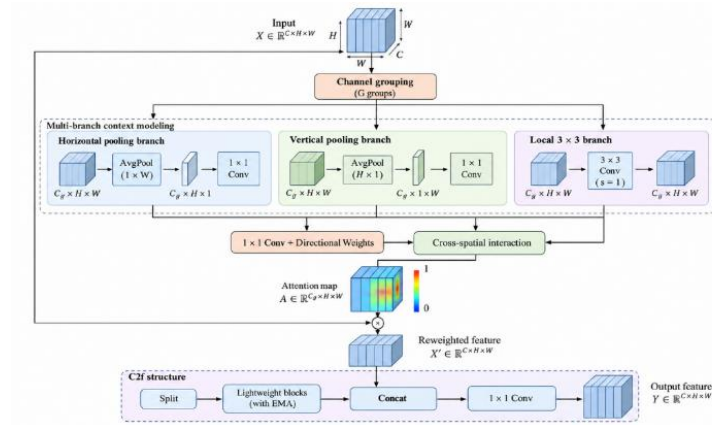


Figure 4. Architecture of C2f_EMA_Lite

For the input feature $X \in \mathbb{R}^{C \times H \times W}$ it is first divided into G subfeatures along the channel dimension:

$$X = \{X_g\}_{g=1}^G, \quad X_g \in \mathbb{R}_{\frac{C}{G} \times H \times W} \quad (10)$$

Horizontal and vertical global aggregation is performed for each subfeature:

$$z_h(h) = \frac{1}{W} \sum_{w=1}^W X_g(h, w), \quad z_w(w) = \frac{1}{H} \sum_{h=1}^H X_g(h, w) \quad (11)$$

Information from the two directions is mapped through a shared 1×1 operation to obtain directional weights:

$$[A_h, A_w] = \sigma(\phi_{1 \times 1}([z_h, z_w])) \quad (12)$$

This produces features containing long-range directional dependencies:

$$U_g = X_g \odot A_h \odot A_w \quad (13)$$

A 3×3 local branch is also used to obtain neighborhood context:

$$V_g = \phi_{3 \times 3}(X_g) \quad (14)$$

Cross-spatial aggregation is then used to obtain pixel-level weights:

$$\omega_g = \sigma \left(\text{Softmax} \left(\text{GAP}(U_g) \right) \text{vec}(V_g) + \text{Softmax} \left(\text{GAP}(V_g) \right) \text{vec}(U_g) \right) \quad (15)$$

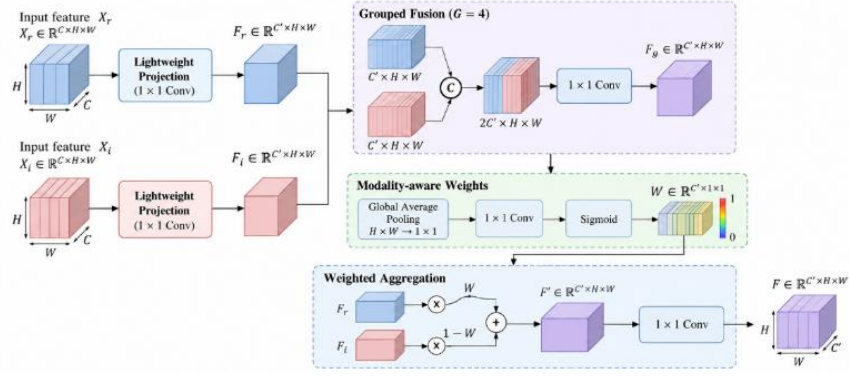
The final output is:

$$Y_g = X_g \odot \omega_g \quad (16)$$

The EMA output recombines long-range directional dependencies, local neighborhood context, and pixel-level weights, allowing the second branch to preserve responses to weak and small defects under lightweight constraints.

3.1.4. DFF_Lite: Three-Scale Lightweight Dual-Branch Fusion

DFF_Lite is introduced at the P3, P4, and P5 scales (Figure 5), producing 96, 160, and 192 channels, respectively, with four lightweight interaction groups to balance fusion capability and computational cost. Dynamic feature fusion has been used to adaptively integrate multiscale local features according to global information [38]. Here, the idea is lightweighted for same-scale fusion between the two backbone branches.


Figure 5. Architecture of DFF_Lite

This section uses an operator-level mathematical definition consistent with the implementation interface and does not assume unspecified internal convolution details. At the l th scale, the features from the first backbone and the second backbone are denoted by R_l and I_l , respectively. They are first mapped into a fusible space through lightweight projection:

$$\hat{R}_l = \phi_r(R_l), \quad \hat{I}_l = \phi_i(I_l) \quad (17)$$

The channels are divided into $G = 4$ subspaces, and modality weights are computed within each subspace:

$$[\alpha_l^r, \alpha_l^i] = \text{Softmax}(\psi_l(\hat{R}_l, \hat{I}_l)) \quad (18)$$

where $\alpha_l^r + \alpha_l^i = 1$. The weighted fusion is expressed as:

$$F_l^{(g)} = \alpha_{l,g}^r \odot \hat{R}_{l,g} + \alpha_{l,g}^i \odot \hat{I}_{l,g} \quad (19)$$

After all groups are concatenated, an output projection produces the fused feature at the corresponding scale:

$$F_l = \phi_o(\text{Concat}_{g=1}^G F_l^{(g)}) \quad (20)$$

This formulation enables the network to adaptively adjust the contribution of the two backbones according to scene and scale: P3 emphasizes spatial details, P4 balances structure and semantics, and P5 captures high-level semantics and global context.

3.2. Dea-yolo Lightweight Student

The student uses the n-scale coefficients [0.33, 0.25, 1024], with its key structure summarized in Table 2. It follows the multiscale single-stage detection paradigm developed from YOLO9000 and YOLOv3 to the industrial lightweight YOLOv6, while replacing the backbone and neck with a lightweight dual-branch design [39-41]. After DFF_Lite fusion, P3/P4/P5 are fed into a PAN-FPN-style neck, and a decoupled detection head outputs class scores and bounding boxes [6,32].

Table 2. Key Structure and Scales of the Lightweight Student

Part	Main module	Output/key channels (width=0.25)	Role
Backbone1	GhostConv + C2f_Ghost_Lite	P3≈64, P4≈128, P5≈256	Low-cost spatial/semantic extraction
Backbone2	GhostConv + EMA-based C2f	P3≈64, P4≈128, P5≈256	Preserve attention enhancement
Fusion	DFF_Lite	96 / 160 / 192	Adaptive dual-branch fusion
Neck	Upsample + Concat + C2f_Ghost_Lite	64 / 128 / 192	Lightweight multiscale aggregation
Head	Detect(nc=1)	P3 / P4 / P5	Classification and regression

3.3. Problem Definition and Overall Framework

Given an input image I , the YOLOv8n backbone produces multiscale features F_3, F_4 , and F_5 . The neck further generates P_3, P_4 , and P_5 and sends them to the detection head. The basic detection process can be summarized by Eqs. (21) and (22).

$$F_i = B(I), \quad i \in \{3, 4, 5\} \quad (21)$$

$$P_i = N(F_3, F_4, F_5), \quad Y = H(P_3, P_4, P_5) \quad (22)$$

DEA-YOLOv8n is fixed as the teacher T , while the heterogeneous network that supports BN sparsification and structured pruning is defined as the student S . The KDSP pipeline is shown in Figure 6 and consists of Stage I sparsity-aware pre-distillation, Stage II dependency-aware structured pruning, and Stage III distillation-assisted fine-tuning.

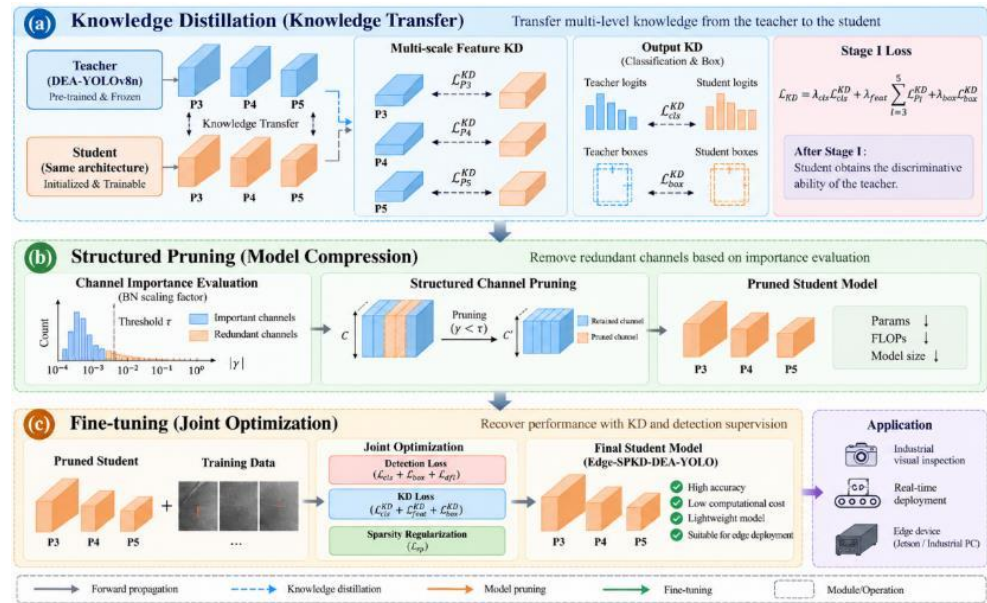


Figure 6. Three-stage Collaborative Compression Framework of Kdsp-dea-yolo

3.4. BN Channel Importance and Sparsification

Assume that a convolution is followed by Batch Normalization. Its normalization and affine transformation are given in Eq. (23). BN stabilizes training through batch statistics [42]. In this study, the scaling factor $|\gamma|$ is used as the basic measure of channel importance, as shown in Eq. (24).

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad y = \gamma \hat{x} + \beta \quad (23)$$

$$S_{l,c} = |\gamma_{l,c}| \quad (24)$$

To avoid simply treating a low γ as indicating a nonessential feature, Stage I jointly shapes the γ distribution using teacher supervision and L1 sparsity regularization. Low-contribution channels are gradually suppressed while responses associated with small-defect recognition are preserved as much as possible [14,42].

3.5. Stage I: Sparsity-Aware Pre-Distillation

Knowledge distillation is first applied to the complete student. For classification outputs, the teacher and student logits are denoted by z_t and z_s . Their softened distributions at temperature T are given in Eqs. (25) and (26), and the classification distillation loss uses the Kullback-Leibler divergence in Eq. (27).

$$p_t^T = \text{Softmax}\left(\frac{z_t}{T}\right) \quad (25)$$

$$p_s^T = \text{Softmax}\left(\frac{z_s}{T}\right) \quad (26)$$

$$L_{cls}^{KD} = T^2 \cdot D_{KL}(p_t^T || p_s^T) \quad (27)$$

To actively form prunable channels, L1 regularization is applied to all BN scaling factors that are allowed to become sparse, as shown in Eq. (28).

$$L_{sp} = \sum \sum_{l,c} |\gamma_{l,c}| \quad (28)$$

For intermediate features, industrial small defects are more sensitive to high-resolution representations; therefore, P3/P4/P5 are distilled simultaneously. When the teacher and student channel dimensions differ, a 1×1 adapter A_l maps student features to the teacher feature space. The multiscale feature loss is defined in Eq. (29).

$$L_{feat}^{KD} = \sum_{l \in \{3,4,5\}} \frac{w_l}{N_l} \|F_l^T - A_l(F_l^S)\|_2^2 \quad (29)$$

In addition to classification and multiscale feature distillation, the teacher bounding boxes provide localization knowledge. An L1 bounding-box distillation term is adopted in Eq. (30), and the detection loss, classification KD, feature KD, box KD, and BN sparsity term are combined in Eq. (31). The development of IoU-based regression losses from GIoU to DIOU/CIoU also demonstrates the importance of geometric constraints for localization optimization [43-44].

$$L_{box}^{KD} = 1N \sum_i \|b_i^T - b_i^S\|_1 \quad (30)$$

$$L_{pre} = L_{det} + \lambda_{cls} L_{cls}^{KD} + \lambda_{feat} L_{feat}^{KD} + \lambda_{box} L_{box}^{KD} + \lambda_{sp} L_{sp} \quad (31)$$

During the first 10 epochs of Stage I, only detection and distillation are performed. Thereafter, λ_{sp} is linearly increased from 0 to its target value. The main setting uses 1×10^{-4} . The loss composition is illustrated in Figure 7.

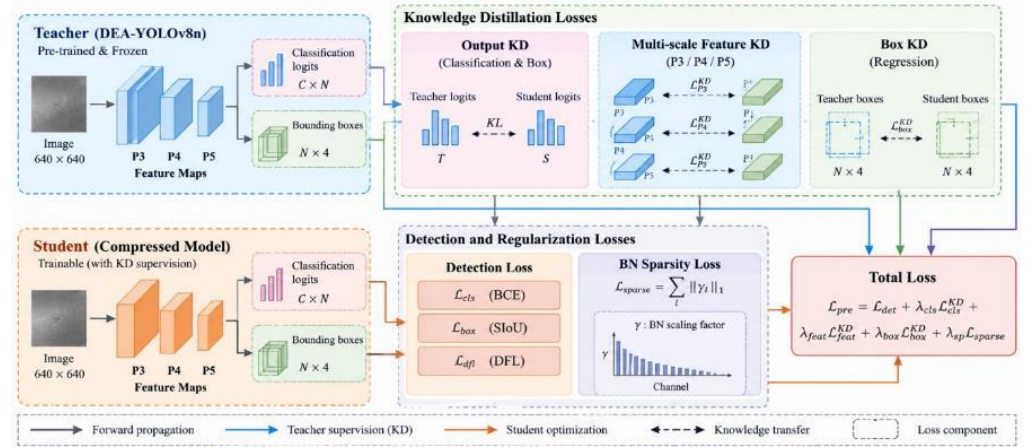


Figure 7. Sparsity-aware Pre-Distillation and Loss Composition in Stage I

3.6. Stage II: Dependency-Aware Structured Pruning

After Stage I, $S_{l,c} = |\gamma_{l,c}|$ is calculated for every candidate channel and the channels are ranked accordingly. Given a target pruning ratio r , the threshold T_r is obtained from the global quantile, and the channel-retention mask is determined by Eq. (32).

$$M_{l,c} = \{1, S_{l,c} \geq T_r; 0, S_{l,c} < T_r\} \quad (32)$$

The DEA architecture contains dual backbones, DFF, Concat operations, and residual paths. Independent pruning may therefore create mismatched dimensions across branches. Dependency groups are constructed according to the topology, and related convolutional output/input channels and BN parameters are removed synchronously. Detection-head output layers, critical SPPF interfaces, and fusion dimensions that cannot be safely mapped are treated as protected layers; the maximum pruning ratio of shallow P3 layers is limited to 0.75 times the global target ratio.

3.7. Stage III: Distillation-Assisted Fine-Tuning

After hard pruning, the frozen teacher is reconnected to the fixed compressed topology and output, feature, and bounding-box distillation are continued. Stage III no longer uses sparsity regularization, and the total distillation weight $\eta(e)$ is decayed using the cosine schedule in Eqs. (33)-(34).

$$L_{ft}(e) = L_{det} + \eta(e) \cdot (\lambda_{cls} L_{cls}^{KD} + \lambda_{feat} L_{feat}^{KD} + \lambda_{box} L_{box}^{KD}) \quad (33)$$

$$\eta(e) = \eta_0 \cdot \frac{1 + \cos(\pi e/E)}{2} \quad (34)$$

The main configuration uses $\eta_0=1.0$ decayed to 0.1 and an initial learning rate of 2×10^{-4} to reduce damage to retained features from large post-pruning updates.

3.8. Complexity and Deployment Metrics

To distinguish theoretical compression from actual speed gains, the parameter compression ratio CR_P, FLOPs compression ratio CR_F, and FPS improvement ratio SR_FPS are reported and defined in Eqs. (35)-(37).

$$CR_P = \frac{P_0 - P_c}{P_0} \times 100\% \quad (35)$$

$$CR_F = \frac{F_0 - F_c}{F_0} \times 100\% \quad (36)$$

$$SR_{FPS} = \frac{FPS_c - FPS_0}{FPS_0} \times 100\% \quad (37)$$

In addition to Params, FLOPs, and FPS, deployment evaluation reports FP32 model size, batch=1 single-image latency, and peak memory. Formal measurements should use fixed warm-up iterations, measurement repetitions, and consistent preprocessing/NMS settings.

4. Experimental Setup

4.1. Dataset

The self-collected dataset contains 1,990 industrial component images at 1080×1920 resolution. All images are resized to 640×640 and labeled as damage, with an 80%/10%/10% split for training, validation, and testing. During training, rotation, flipping, brightness/contrast changes, and Poisson, Gaussian, and salt-and-pepper noise are used for augmentation. Representative samples are shown in Figure 8.



Figure 8. Representative Samples from the Dataset

4.2. Experimental Environment

Table 3 provides a detailed display of the experimental environment for this experiment.

Table 3. Training and Test Environment

Item	Configuration
CPU	Intel Xeon E5-2630 v4 @ 2.20 GHz
GPU	NVIDIA Tesla P100-PCIE-12GB × 2
GPU Memory	24 GB (combined)
OS	CentOS 7
Framework	PyTorch 1.13.0
CUDA / cuDNN	CUDA 11.7 / cuDNN 8.5.0

4.3. Training Strategy and Hyperparameters

Table 4 provides a detailed display of the specific hyperparameter values for this experiment.

Table 4. Main Training Hyperparameters

Parameter	Teacher/previous baseline	Stage I	Stage III
Input size	640×640	640×640	640×640
Batch size	16	16	16
Epoch	200	80	100
Optimizer	AdamW	AdamW	AdamW
Initial LR	0.001	0.0005	0.0002
Weight decay	0.0005	0.0005	0.0005
KD temperature T	-	4	4
λ_{sp}	-	1×10^{-4}	0
KD weight $\eta(e)$	-	1.0	1.0→0.1 (cosine)
Augmentation	Mosaic + standard augmentation	Same as teacher	Reduced Mosaic + standard augmentation

Stage II only performs channel ranking, synchronized dependency-group deletion, and structural remapping; therefore, no separate optimizer or epoch setting is used. The training-order ablation maintains an approximately equal training budget at a fixed 20% stress-test pruning ratio, while the final deployment operating point is selected independently through the pruning-ratio experiment.

4.4. Evaluation Metrics

Accuracy is evaluated using precision, recall, mAP@0.5, and mAP@0.5:0.95; complexity is measured using Params and FLOPs; and deployment performance is evaluated using model size, FPS, latency, and peak memory. Because missed detections often have a higher cost in industrial inspection, changes in recall are also emphasized.

5. Results and Analysis

5.1. Main Comparison

Table 5 summarizes the YOLO baselines and different compression strategies. The teacher achieves 70.3% mAP@0.5 with 6.00 M parameters. The lightweight student reaches approximately 70.6% after Stage I, and after 20% target pruning followed by Stage III recovers to 69.8% with 3.61 M parameters. Efficiency values and some ablation results marked as predicted below are used for the current design analysis and should be replaced by measured values in the final experimental manuscript.

Table 5. Overall Performance Comparison of the Main Models

Model	Description	P/%	R/%	mAP50/%	mAP50-95/%	Params/M	FLOPs/G	FPS	Latency/ms
YOLOv5n	Baseline	73.2	56.5	60.8	22.1	1.90	4.50	128.2	7.80
YOLOv8n	Baseline	70.4	59.7	62.9	23.5	3.20	8.90	116.3	8.60
DEA-YOLOv8n	Teacher	78.9	65.3	70.3	29.0	6.00	14.80	92.6	10.80
Pruned Student + Prune + FT		77.2	63.9	67.5	28.2	3.61	8.84	133.3	7.50
Pre-KD + Pruned Student + Prune + FT	Pre-KD +	77.8	64.5	68.4	28.6	3.61	8.84	133.3	7.50

KDSP-DEA-YOLO	Sparse Pre-KD + Prune + KD-FT	78.3	65.0	69.8	28.9	3.61	8.84	133.3	7.50
---------------	-------------------------------	------	------	------	------	------	------	-------	------

The final KDSP-DEA-YOLO has a parameter count close to YOLOv8n (3.61 M vs. 3.20 M), while its mAP@0.5 is 7.2 percentage points higher. Relative to the teacher, the parameter count is reduced by 39.8%, illustrating the accuracy-complexity trade-off.

5.2. Comparison with Representative Industrial Defect Detectors

Table 6 compares representative industrial defect detectors, including LF-YOLO, HyperDefect-YOLO, SLF-YOLO, ETDNet, and two-stage detectors [28-31]. KDSP-DEA-YOLO achieves approximately 69.8% mAP@0.5. Although this is slightly lower than the teacher and Mask R-CNN, the model uses only 3.61 M parameters and approximately 8.84 G FLOPs, emphasizing deployment under resource constraints. Because the datasets, training strategies, and hardware used by different studies are not identical, cross-paper values are provided mainly as order-of-magnitude references.

Table 6. Comparison with Representative Industrial Surface Defect Detection Models

Model	P/%	R/%	mAP50/%	mAP50-95/%	Params/M	FLOPs/G
LF-YOLO	52.9	49.2	46.0	13.5	7.20	17.10
HyperDefect-YOLO	72.6	57.3	61.8	22.0	3.90	9.20
SLF-YOLO	66.1	56.6	55.8	21.3	2.60	7.40
ETDNet	71.9	63.7	65.9	24.3	8.60	8.50
GA-Faster R-CNN	76.0	63.5	68.0	27.0	41.00	120.50
Mask R-CNN	76.0	67.2	72.3	28.2	46.00	94.90
DEA-YOLOv8n	78.9	65.3	70.3	29.0	6.00	14.80
KDSP-DEA-YOLO	78.3	65.0	69.8	28.9	3.61	8.84

5.3. Effect of Sparsity-Aware Pre-Distillation

Stage I must preserve accuracy while creating prunable channels. Figure 9 and Table 7 show that when $\lambda_{sp}=1 \times 10^{-4}$, mAP@0.5 remains approximately 70.6%, while the proportion of near-zero $|\gamma|$ channels increases markedly, providing clearer importance separation for subsequent structured removal.

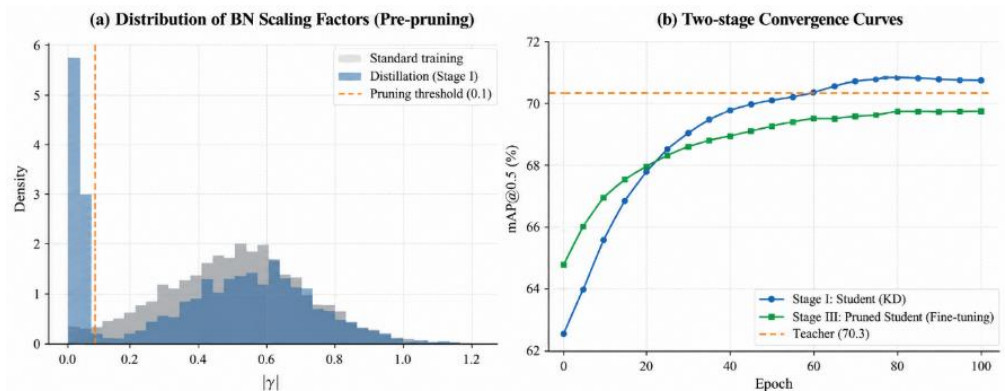


Figure 9. BN Scaling-Factor Distribution and Two-Stage Training Convergence Curves under the 40% Stress Test

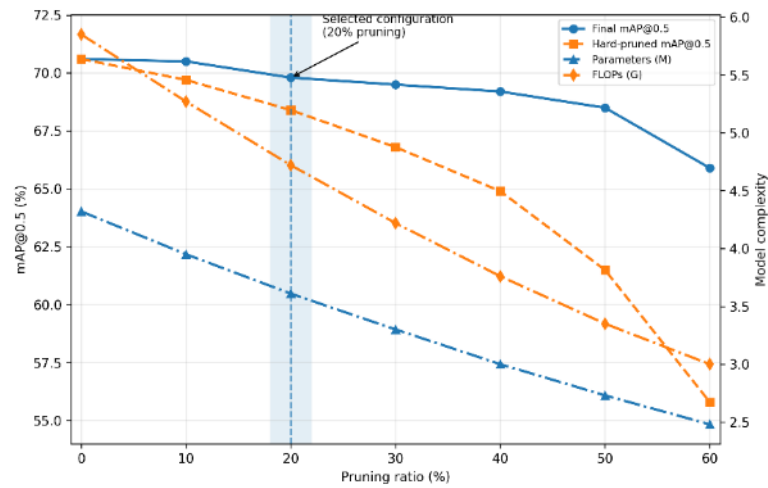
Table 7. Ablation of the BN Sparsity Coefficient

λ_{sp}	Stage I mAP50/% (predicted)	Near-zero $ \gamma $ channels/% (predicted)	Final mAP50/% after 20% stress- test pruning (predicted)
0	70.4	5.2	69.2
5×10^{-5}	70.6	18.7	69.6
1×10^{-4}	70.5	26.8	69.8
2×10^{-4}	69.8	35.6	69.4
5×10^{-4}	69.2	47.9	68.6

When λ_{sp} is further increased to 5×10^{-4} , the proportion of near-zero channels continues to increase, but accuracy decreases both in Stage I and after pruning. This indicates that excessive sparsity may prematurely suppress channels that still carry discriminative information; therefore, 1×10^{-4} is selected as the main setting.

5.4. Pruning Ratio and Accuracy-Complexity Trade-Off

Figure 10 and Table 8 illustrate the accuracy-complexity changes as the target pruning ratio increases from 10% to 60%. Higher pruning ratios continuously reduce Params and FLOPs while increasing the immediate accuracy loss after hard pruning. Stage III can recover part of this loss, but the recoverable margin gradually becomes limited at high pruning ratios.

**Figure 10.** Trade-off among Pruning Ratio, Detection Accuracy, and Student-Model Complexity**Table 8.** Predicted Performance at Different Structured Pruning Ratios (Stage I Student as the 0% Baseline)

Pruning ratio	Params/M	Param↓/%	FLOPs/G	Hard-pruned mAP50	Final mAP50	mAP50-95	FPS
0%	4.32	0.0	10.96	70.6	70.6	29.2	114.9
10%	3.95	8.6	9.87	69.7	70.5	29.1	123.5
20%	3.61	16.4	8.84	68.4	69.8	28.9	133.3
30%	3.30	23.6	7.91	66.8	69.5	28.8	142.1
40%	3.01	30.6	7.04	64.9	69.2	28.6	150.4
50%	2.73	36.8	6.28	61.5	68.5	27.8	157.9
60%	2.48	42.6	5.62	55.8	65.9	25.7	164.5

A 20% pruning ratio is selected as the final operating point. Relative to the Stage I student, the parameter count decreases from 4.32 M to 3.61 M and FLOPs from 10.96 G to 8.84 G. mAP@0.5 is approximately 68.4% immediately after hard pruning and recovers to approximately 69.8% after Stage III, providing a more stable balance between lightweighting and accuracy.

5.5. Ablation of Training Order

To examine whether distillation should be moved before pruning, four pipelines are compared at a fixed 20% stress-test pruning ratio and approximately equal training budgets (Table 9 and Figure 11). Post-pruning KD alone recovers part of the accuracy, whereas Sparse Pre-KD $\rightarrow$ Prune $\rightarrow$ KD-FT combines knowledge protection before pruning with post-pruning recovery and yields the highest predicted performance.

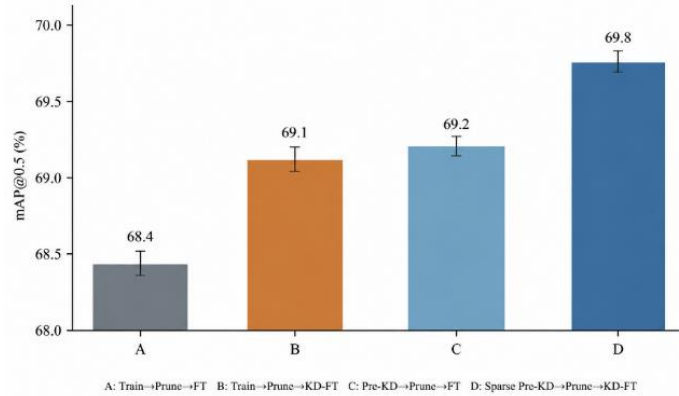


Figure 11. Ablation Results for Different Training Orders

Table 9. Ablation of Training Order

Scheme	Pipeline	mAP50/%	mAP50-95/%	Gain over A/pp
A	Train $\rightarrow$ Prune $\rightarrow$ FT	68.4	27.4	0.0
B	Train $\rightarrow$ Prune $\rightarrow$ KD-FT	69.1	28.0	+0.7
C	Pre-KD $\rightarrow$ Prune $\rightarrow$ FT	69.2	28.1	+0.8
D	Sparse Pre-KD $\rightarrow$ Prune $\rightarrow$ KD-FT	69.8	28.6	+1.4

This trend suggests that Stage I pre-distillation operates in the complete representation space and jointly shapes channel importance with BN sparsification, whereas Stage III reorganizes the remaining channels after model capacity has already been reduced. The two KD stages therefore play complementary roles.

5.6. Distillation Components and Multiscale Weights

Table 10 shows that Output KD alone provides limited recovery. Adding P3 yields further improvement, while jointly using P3/P4/P5 together with Box KD reaches approximately 69.8%. This indicates that fine-grained knowledge for industrial small defects depends primarily on multiscale intermediate representations rather than only on final classification soft labels.

Table 10. Ablation of Distillation Components

Configuration	Output KD	P3	P4	P5	Box KD	mAP50/% (predicted)	mAP50-95/%
A	×	×	×	×	×	68.7	27.6
B	√	×	×	×	×	69.0	27.9
C	√	√	×	×	×	69.3	28.1
D	√	√	√	√	×	69.7	28.5
E	√	√	√	√	√	69.8	28.6

In Table 11, P3/P4/P5 weights of 0.5/0.3/0.2 provide the best performance. P3 supplies high-resolution texture details, while P4 and P5 supplement semantic and contextual

information. Thus, prioritizing P3 rather than exclusively using P3 better matches the multiscale requirements of small-defect detection.

Table 11. P3/P4/P5 Multiscale Feature-Distillation Weights

w3	w4	w5	Strategy	mAP50/% (predicted)
1/3	1/3	1/3	Uniform multiscale	69.4
0.5	0.3	0.2	P3-prioritized	69.8
0.6	0.3	0.1	Stronger P3 bias	69.6
1.0	0	0	P3 only	69.1
0.6	0.4	0	P3+P4	69.8

5.7. Deployment Efficiency Analysis

Figure 12 and Table 12 evaluate deployment efficiency from the Pareto relationship and predicted latency. The 20% pruned model contains approximately 3.61 M parameters and occupies 13.8 MB. Relative to the teacher, the predicted P100 latency decreases from 10.8 ms to 7.5 ms and FPS increases from 92.6 to 133.3.

Table 12. Deployment-efficiency Comparison

Model	Device	Params/M	Size/MB	FPS (predicted)	Latency/ms (predicted)	Peak memory/MB (predicted)
YOLOv8n	P100	3.20	12.4	116.3	8.6	720
DEA-YOLOv8n	P100	6.00	23.6	92.6	10.8	910
KDSP-DEA-YOLO	P100	3.61	13.8	133.3	7.5	704
YOLOv8n	Xeon E5-2630 v4	3.20	12.4	18.1	55.2	1110
DEA-YOLOv8n	Xeon E5-2630 v4	6.00	23.6	13.8	72.4	1350
KDSP-DEA-YOLO	Xeon E5-2630 v4	3.61	13.8	20.1	49.8	1050

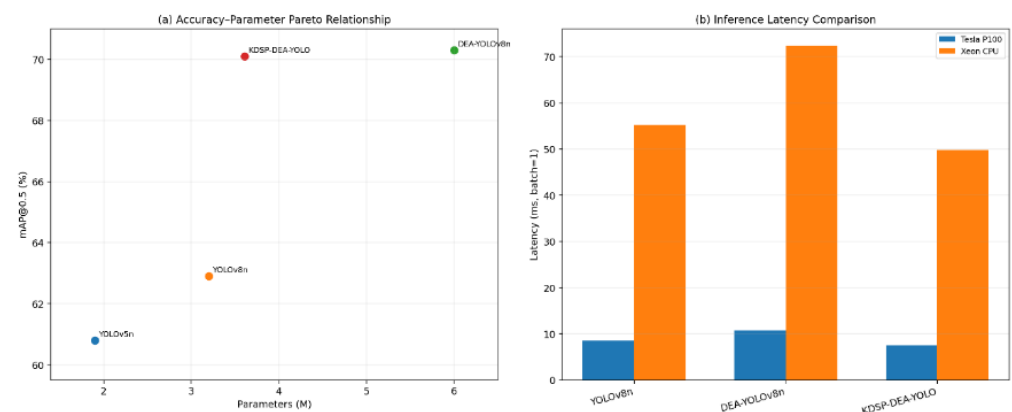


Figure 12. Accuracy-parameter Pareto Relationship and Inference-Latency Comparison

FLOPs do not fully reflect real hardware efficiency. Dual branches, DFF, and Concat operations increase memory access and operator scheduling overhead, whereas structured pruning can reduce both channel computation and part of the memory traffic. Therefore, final deployment conclusions should be based on measured latency and FPS under a unified hardware environment.

6. Discussion

Overall, the predicted results indicate that a 20% pruning ratio provides a robust Pareto operating point. After Stage III, mAP@0.5 is approximately 69.8%, only 0.5 percentage points below the teacher, while the overall parameter count is reduced by approximately 39.8%.

The two KD stages serve different roles. Stage I protects knowledge associated with weak and small defects in the complete student while helping form prunable channels, whereas Stage III restores the representational capability of the remaining channels on the fixed compressed topology.

Dependency-aware pruning is critical for regular compression of the dual-backbone and DFF architecture. Group-wise synchronized pruning avoids cross-branch dimensional conflicts and allows the compressed model to continue using standard dense convolution libraries for inference.

The current limitations are as follows: the dataset comes from a single manufacturing scenario, so generalization across materials and devices remains to be verified; INT8 quantization, TensorRT operator fusion, and NAS-based automatic pruning-ratio search have not yet been integrated; and the predicted efficiency and ablation values still require confirmation through measurements in a unified environment.

7. Conclusion

This study proposes the KDSP-DEA-YOLO collaborative compression framework. A heterogeneous student is first constructed using GhostConv, a lightweight EMA branch, and DFF_Lite. Sparsity-aware pre-distillation is then used to form prunable channels, structured pruning is performed according to cross-layer dependencies, and decayed KD fine-tuning is finally applied to recover accuracy. The pipeline unifies structural lightweighting, knowledge preservation, and regularized compression.

Based on the current baseline and compression trends, the predicted 20% target-pruning configuration reduces the Stage I student from 4.32 M parameters and 10.96 G FLOPs to 3.61 M parameters and 8.84 G FLOPs, while achieving approximately 69.8% mAP@0.5 after Stage III. Relative to the 6.0 M teacher, the parameter count is reduced by approximately 39.8%. Future work will focus on measured hardware-latency verification, cross-scenario generalization, joint quantization-compression, and automated pruning-ratio search to further validate the framework for industrial edge deployment.

References

1. R. Ameri, C. C. Hsu, and S. S. Band, "A systematic review of deep learning approaches for surface defect detection in industrial applications," *Engineering Applications of Artificial Intelligence*, vol. 130, p. 107717, 2024.
2. Y. Ma, J. Yin, F. Huang, et al., "Surface defect inspection of industrial products with object detection deep networks: a systematic review," *Artificial Intelligence Review*, vol. 57, p. 333, 2024.
3. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.
4. W. Liu, D. Anguelov, D. Erhan, et al., "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision*, 2016, pp. 21–37.
5. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
6. T. Y. Lin, P. Dollar, R. Girshick, et al., "Feature Pyramid Networks for Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.
7. N. Carion, F. Massa, G. Synnaeve, et al., "End-to-End Object Detection with Transformers," in *European Conference on Computer Vision*, 2020, pp. 213–229.
8. A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.
9. C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.
10. A. Wang, H. Chen, L. Liu, et al., "YOLOv10: Real-Time End-to-End Object Detection," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.

11. G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLO*, Version 8.0.0 [Computer software], 2023.
12. Z. Gevorgyan, "SiO Loss: More Powerful Learning for Bounding Box Regression," *arXiv preprint arXiv:2205.12740*, 2022.
13. D. Ouyang, S. He, G. Zhang, et al., "Efficient Multi-Scale Attention Module with Cross-Spatial Learning," in **ICASSP 2023 – IEEE International Conference on Acoustics, Speech and Signal Processing**, 2023, pp. 1–5.
14. Z. Liu, J. Li, Z. Shen, et al., "Learning Efficient Convolutional Networks Through Network Slimming," in *IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.
15. H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," in *International Conference on Learning Representations*, 2017.
16. Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.
17. P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance Estimation for Neural Network Pruning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11264–11272.
18. G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.
19. A. Romero, N. Ballas, S. E. Kahou, et al., "FitNets: Hints for Thin Deep Nets," in *International Conference on Learning Representations*, 2015.
20. S. Zagoruyko and N. Komodakis, "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer," in *International Conference on Learning Representations*, 2017.
21. J. Guo, K. Han, Y. Wang, et al., "Distilling Object Detectors via Decoupled Features," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2154–2164.
22. Z. Yang, Z. Li, X. Jiang, et al., "Focal and Global Knowledge Distillation for Detectors," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4643–4652.
23. Y. Zhu, Q. Zhou, N. Liu, et al., "ScaleKD: Distilling Scale-Aware Knowledge in Small Object Detector," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19723–19733.
24. P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.
25. D. Tabernik, S. Sela, J. Skvarc, and D. Skocaj, "Segmentation-based deep-learning approach for surface-defect detection," *Journal of Intelligent Manufacturing*, vol. 31, pp. 759–776, 2020.
26. C. Wang, W. Zhu, B. B. Gao, et al., "Real-IAD: A Real-World Multi-View Dataset for Benchmarking Versatile Industrial Anomaly Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 22883–22892.
27. Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, "Spot-the-Difference Self-Supervised Pre-Training for Anomaly Detection and Segmentation," in *European Conference on Computer Vision*, 2022, pp. 392–408.
28. M. Liu, Y. Chen, J. Xie, L. He, and Y. Zhang, "LF-YOLO: A Lighter and Faster YOLO for Weld Defect Detection of X-Ray Image," *IEEE Sensors Journal*, vol. 23, no. 7, pp. 7430–7439, 2023.
29. Z. Zuo, J. Dong, Y. Gao, and Z. Wu, "HyperDefect-YOLO: Enhance YOLO with Hypergraph Computation for Industrial Defect Detection," *arXiv preprint arXiv:2412.03969*, 2024.
30. H. Zhou, R. Yang, R. Hu, C. Shu, X. Tang, and X. Li, "ETDNet: Efficient Transformer-Based Detection Network for Surface Defect Detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, p. 2525014, 2023.
31. Y. Liu, Y. Liu, X. Guo, X. Ling, and Q. Geng, "Metal surface defect detection using SLF-YOLO enhanced YOLOv8 model," *Scientific Reports*, vol. 15, p. 11105, 2025.
32. S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.
33. J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
34. S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," in *European Conference on Computer Vision*, 2018, pp. 3–19.
35. Q. Hou, D. Zhou, and J. Feng, "Coordinate Attention for Efficient Mobile Network Design," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13713–13722.
36. K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features From Cheap Operations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1580–1589.
37. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
38. J. Yang, P. Qiu, Y. Zhang, D. S. Marcus, and A. Sotiras, "D-Net: Dynamic Large Kernel with Dynamic Feature Fusion for Volumetric Medical Image Segmentation," *Biomedical Signal Processing and Control*, vol. 113, p. 108837, 2026.
39. J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6517–6525.
40. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.
41. C. Li, L. Li, H. Jiang, et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv preprint arXiv:2209.02976*, 2022.

42. S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of Machine Learning Research*, vol. 37, 2015, pp. 448–456.
43. Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, 2020, pp. 12993–13000.
44. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized Intersection over Union: A Metric and a Loss for Bounding Box Regression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.

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 Publisher and/or the editor(s). Publisher 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.