Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

PLFD-YOLO: A Lightweight Multi-Scale Detector for Power Line Defects in UAV Imagery

Shibo Gao
Shihezi University
Ji Liu
Shihezi University
Hongmei Fei
Shihezi University
Xuening Liu
Shihezi University
Jinghui Liu
Shihezi University
Jiangpeng Li
Shihezi University
Yao Zhang
zhangyao2089@sina.com

Shihezi University

Article

Keywords:

Posted Date: August 14th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10453009/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

PLFD-YOLO: A Lightweight Multi-Scale Detector for Power Line Defects in UAV Imagery

Shibo Gao[1,+], Ji Liu[1,+], Hongmei Fei[1], Xuening Liu[1], Jinghui Liu[1], Jiangpeng Li[1], and Yao Zhang[2,3,*]

[1]College of Information Science and Technology, Shihezi University, Shihezi 832000, China
[2]Management Committee of Xinjiang Production and Construction Corps Shihezi National Agricultural Science and Technology Park, Shihezi 832011, China
[3]School of Economics and Management, Shihezi University, Shihezi 832000, Xinjiang, China
[*]zhangyao2089@sina.com
[+]These authors contributed equally as co-first authors.

ABSTRACT

Automated visual inspection of power transmission lines via Unmanned Aerial Vehicles (UAVs) provides an essential diagnostic mechanism for grid asset management. Standard single-stage object detectors, however, often struggle with micro-scale defect localization, dense target overlapping, and natural background clutter while maintaining high inference throughput. We present PLFD-YOLO, a lightweight detection model configured specifically for aerial transmission line imagery. The architecture incorporates a 4-head Path Aggregation Feature Pyramid Network (PAFPN) featuring a stride-4 P2 branch for high-resolution coordinate preservation, coupled with ADown downsampling layers and Deformable Convolution v2 (DCNv2) blocks to account for non-rigid component deformations. To mitigate background noise, multi-scale feature maps are fused via a learnable channel-weighting module (BiFPNConcat) integrated with direction-aware Coordinate Attention layers. During training, sample assignment is decoupled using bounded CIoU matching to prevent non-zero background similarity from corrupting anchor allocation, while coordinate regression is guided by a size-decay Soft-NWD-CIoU loss function. Evaluated on the PTL-AI Furnas dataset at 1280×1280 resolution, PLFD-YOLO obtains 91.57% mAP_{50} and 64.35% mAP_{50-95}—representing improvements of +2.91 and +4.68 percentage points over YOLO11n—at an inference throughput of 80.2 FPS on an RTX 5080 workstation, demonstrating strong potential for future embedded UAV deployment.

1 Introduction

Continuous structural monitoring of high-voltage power transmission grids is essential to prevent line trips and guarantee supply reliability. Transmission lines cross diverse geographical terrains and endure continuous atmospheric exposure and mechanical stress. Over time, components develop structural anomalies: insulators suffer flashover breakout slots, spacer bars shift and induce line gallop, Stockbridge dampers crack under fatigue stress, and bird nests or suspended debris introduce short-circuit hazards. A single undetected fault can trigger localized voltage sags or major grid interruptions, presenting high operational and safety risks to industrial and residential power networks. Consequently, automated visual inspection using Unmanned Aerial Vehicles (UAVs) equipped with high-resolution optical sensors has established itself as a primary acquisition platform for transmission line monitoring.

Historically, grid inspection relied on manual ground patrols and crewed helicopter flybys. Ground patrols are labor-intensive, time-consuming, and physically restricted in mountainous or marshland regions, while helicopter patrols incur substantial operating expenses and flight risks. In recent years, UAV platforms have overcome these limitations by offering flexible maneuverability, low operating costs, and close-range inspection capabilities. Mounting high-resolution RGB sensors on UAVs allows maintenance teams to capture detailed aerial imagery of transmission towers, conductor lines, and hardware fittings. Recent research has explored automated vision techniques for insulator-defect detection, cable segmentation, foreign-object recognition, and bolt inspection[1–5]. Nevertheless, manually evaluating thousands of high-resolution inspection images introduces a severe analysis bottleneck, driving the need for automated deep-learning object detectors capable of real-time multi-class fault identification.

Deep-learning object detectors have evolved from two-stage proposal-based frameworks such as Faster R-CNN[6] to single-stage real-time architectures including YOLOX, YOLOv6, YOLOv7, YOLOv9, YOLOv10, PP-YOLOE, and RTMDet[7–13], alongside real-time vision transformers like RT-DETR[14]. Despite these algorithmic advances, applying deep detectors to aerial transmission grid imagery involves four domain-specific technical challenges:

1. **Spatial-Scale Challenge (Micro-Scale and Dense Targets)**: Defect features (such as missing bolts on spacer bars, small damper fatigue cracks, or distant bird nests) occupy very few pixels in high-resolution aerial frames (1280×720 pixels or larger). Conventional backbones using large downsampling factors (e.g., stride 32) discard fine-grained spatial coordinates, leading to severe spatial aliasing and high miss rates for sub-pixel targets. Multi-scale pyramids and NWD-based localization provide relevant foundations for addressing this issue[15–18].

2. **Semantic-Background Challenge (Complex Clutter and Class Confusion)**: Transmission lines pass through complex natural backgrounds, including forest canopy, agricultural foliage, and structural steel tower frames. These environments generate high false-positive clutter. Furthermore, subtle visual differences between normal and defective states (e.g., Stockbridge dampers) cause severe inter-class confusion. Channel-spatial and coordinate-aware attention mechanisms motivate selective suppression of this clutter[19–21].

3. **Optimization Challenge (Extreme Class Imbalance and Unstable Regression)**: In real-world inspection datasets, normal components represent approximately 75% of total annotated instances (13,337 out of 17,808), while rare defect categories (such as damaged spacer bars) are underrepresented, causing classification gradients to be dominated by majority classes during backpropagation. Additionally, standard IoU loss exhibits a step-like gradient behavior for tiny bounding boxes, rendering regression optimization unstable. IoU-family regression objectives and Task-Aligned Learning provide the basis for the proposed decoupled strategy[22–25].

4. **Deployment Challenge (Post-Processing Latency and Edge Constraints)**: Standard post-processing routines (such as Soft-NMS[26] implemented in Python) introduce frequent CPU-GPU data movement and serial loops. This creates a computational bottleneck that lowers frame rates below real-time operational thresholds (30 FPS) on resource-constrained platforms.

Compared with generic object detection datasets, UAV-based power line inspection presents more severe small-object, long-tailed, and background-clutter challenges. Therefore, both localization sensitivity and deployment efficiency must be jointly optimized. To address these challenges and satisfy the demands of real-time transmission grid inspections, we propose PLFD-YOLO, a high-precision, real-time detector optimized for UAV-based power line defect detection.

Contributions

To address the aforementioned challenges in UAV-based power line inspection and advance existing knowledge in edge-deployed grid diagnostics, we propose the PLFD-YOLO framework. The main contributions of this work are summarized as follows:

1. **DCNv2-Enhanced Four-Head PAFPN with ADown Downsampling**: Unlike standard detectors using P3/P4/P5 outputs, we construct a high-resolution 4-head Path Aggregation Feature Pyramid Network (PAFPN) by introducing an additional P2 detection head (stride 4) to preserve fine-grained spatial coordinates for small-scale targets and prevent feature leakage. To retain spatial structures during feature reduction, standard strided convolutions in both backbone and neck are replaced with ADown modules, which perform detail-preserving downsampling by combining average pooling with channel-split convolutions. Simultaneously, Deformable Convolution v2 (DCNv2) is embedded into the deep backbone stages (P4 and P5) to adaptively model non-rigid geometric deformations under complex viewpoints.

2. **Channel-Weighted Feature Fusion via BiFPNConcat and Coordinate Attention**: We establish an attention-guided neck to tackle complex background clutter. Standard feature concatenation is replaced with BiFPNConcat, a learnable weighted fusion module that dynamically adjusts the contribution of different feature scales based on training feedback. Additionally, Coordinate Attention (CA) is uniformly inserted at key neck fusion nodes to capture both spatial directions and channel dependencies, maintaining precise spatial-channel alignment for linear power structures.

3. **Decoupled Post-Processing and Loss Optimization**: On the optimization level, a TAL-Loss Decoupling strategy restricts positive sample matching to bounded CIoU matching $\max(\text{CIoU}, 0)$ (preventing background noise propagation from NWD metrics) while regressing bounding boxes with a size-decay Soft-NWD-CIoU loss, complemented by a moderated class-weighted classification loss. Finally, a GPU implementation of `torchvision.ops.nms` replaces Python-based Soft-NMS, resolving the CPU-GPU communication bottleneck and achieving a real-time throughput of 80.2 FPS, demonstrating strong potential for future embedded UAV deployment.

2 Related Work

Traditional Inspection and Hand-crafted Vision

In the early stages of automated power line inspection, research relied heavily on hand-crafted texture, edge, and color descriptors followed by conventional classifiers. Such pipelines are sensitive to illumination changes, atmospheric conditions, and cluttered natural backgrounds, which limits their generalization in practical inspection scenes. Recent UAV-based visual and thermal inspection studies instead use learned representations to detect power-line structures directly[1].

CNN and YOLO-based Real-time Detectors

To overcome the limitations of hand-crafted features, Convolutional Neural Networks (CNNs) substantially improved object detection accuracy. Deep learning detectors are commonly divided into two-stage and one-stage families. Two-stage methods such as Faster R-CNN can provide strong localization accuracy, but their proposal-and-refinement pipeline has substantial computational overhead for resource-constrained UAV platforms[6].

Consequently, real-time inspection increasingly adopts one-stage detectors. Recent YOLO variants such as YOLOX, YOLOv6, YOLOv7, YOLOv9, and YOLOv10, together with efficient designs such as PP-YOLOE and RTMDet, perform classification and bounding-box regression in a single forward pass[7–13]. Despite their speed, standard one-stage detectors can still lose small-scale and dense defect features after repeated downsampling. In this study, YOLO11n is used as the engineering baseline evaluated on the same dataset and inference settings.

Small-object Detection and High-resolution Feature Modeling

Addressing this critical gap in micro-scale defect recognition has thus driven specialized research into small-object detection and high-resolution feature modeling. Typical strategies involve multi-scale feature pyramids such as FPN, PANet, and BiFPN, together with adaptive upsampling and scale-aware fusion through CARAFE and ASFF. Normalized Wasserstein Distance (NWD) further models tiny bounding boxes as 2D Gaussian distributions to reduce sensitivity to small coordinate displacements[15–18,27,28].

Attention and Feature Fusion in Dense/Cluttered Scenes

In addition to multi-scale architectures, attention mechanisms have been integrated into CNNs to suppress background noise in dense or cluttered scenes. SENet, CBAM, ECA-Net, SimAM, and Dynamic Head motivate selective channel-spatial feature modulation, while Coordinate Attention (CA) factorizes attention into horizontal and vertical one-dimensional encodings that preserve directional coordinate information for thin, linear structures[19–21,29–31].

Detection Losses and Assignment Strategies

However, integrating NWD into a detector requires careful design of bounding-box regression and sample assignment. GIoU, DIoU/CIoU, and EIoU illustrate the development of IoU-based regression objectives; Generalized Focal Loss, Varifocal Loss, and GFLv2 refine classification–localization consistency; and Task-Aligned Learning (TAL) jointly models classification and localization during assignment[22–25,32–34]. In the proposed model, assignment is decoupled from the NWD-based regression term to avoid propagating background similarity into positive matching.

Power-line Inspection Models and Deployment Constraints

Building upon these foundational advancements, several domain-specific detectors have been optimized for grid inspection. MAP-YOLOv8 addresses insulator-defect detection in UAV imagery; PowerLine-MTYOLO jointly performs cable segmentation and broken-strand detection; SESYOLO targets foreign-object detection; and YOLOv7-CWFD detects transmission-line bolt defects[2–5]. These studies confirm the domain relevance of lightweight, multi-scale, and attention-guided designs, while their tasks and datasets differ from the multi-class defect setting considered here.

Beyond standard one-stage architectures, real-time end-to-end transformers such as RT-DETR have emerged, building upon visual representations such as Swin Transformer. Representative refinements include Deformable DETR, Conditional DETR, DAB-DETR, and DINO[14,16,35–38]. These approaches provide strong global reasoning but can be costly for high-resolution on-board drone execution. In contrast, PLFD-YOLO uses an efficient YOLO11n baseline tailored for multi-class component fault detection on the PTL-AI Furnas dataset. Its high-resolution P2 head, BiFPNConcat, and Soft-NWD-CIoU loss balance computational efficiency and accuracy, demonstrating promising potential for future embedded UAV deployment.

3 Methods

In this section, we present the comprehensive methodological design of the proposed PLFD-YOLO framework. Specifically, the architectural details of the spatial-preservation backbone and neck are detailed in Section 3.1 and 3.2. The learnable channel

weighted feature fusion and attention mechanisms are formulated in Section 3.3 and 3.4. Finally, the decoupled training assigners, loss functions, and post-processing details are defined in Section 3.5 and 3.6.

3.1 Overall Network Architecture

The proposed PLFD-YOLO detector is designed as a unified, real-time framework optimized for detecting small-scale, dense, and geometrically deformed defects on power transmission lines under complex natural backgrounds. The detector consists of three primary components: a modified backbone with deformable convolutional layers for adaptive feature extraction, a channel-weighted 4-head Path Aggregation Feature Pyramid Network (PAFPN) neck for multi-scale feature fusion, and a decoupled classification-regression head. The overall network topology is illustrated in Figure 1.

To establish a direct mapping between the physical challenges of UAV-based power line inspection and our algorithmic enhancements, we summarize our design rationale in Table 1. By aligning specific network modifications with corresponding inspection difficulties, we ensure that our detector addresses spatial resolution losses, geometric viewpoints, background noise, class imbalance, regression instability, and computational latency in a coordinated manner.

To optimize feature representation across different physical scales, the network's downsampling strides are aligned with the physical dimensions of power line components. Mathematically, for a small-scale target like a spacer bolt (typically less than 32×32 pixels in raw resolution), standard downsampling by a factor of 32 reduces its spatial footprint to less than 1×1 pixel. Under the Shannon-Nyquist sampling theorem, this severe downsampling exceeds the spatial Nyquist frequency, causing high-frequency coordinate details to suffer from spatial aliasing and detail loss. Therefore, we introduce the low-level P2 scale (stride 4, feature map size 320×320), which preserves rich geometric coordinates and is dedicated to small-scale targets like spacer bolts and damper cracks. Although shallow P2 feature maps retain fine-grained spatial structures, they lack semantic abstraction, which can introduce background clutter in complex natural environments. To mitigate this semantic-resolution trade-off, the P2 features are fed into a PAFPN neck, where they are fused with deep semantic top-down features and selectively filtered by Coordinate Attention layers before reaching the detection heads. The intermediate P3 (stride 8, 160×160) and P4 (stride 16, 80×80) scales capture structural details of standard spacers and dampers. The deep P5 scale (stride 32, 40×40) captures global semantic shapes, making it suitable for detecting large bird nests and warning spheres under natural backgrounds.

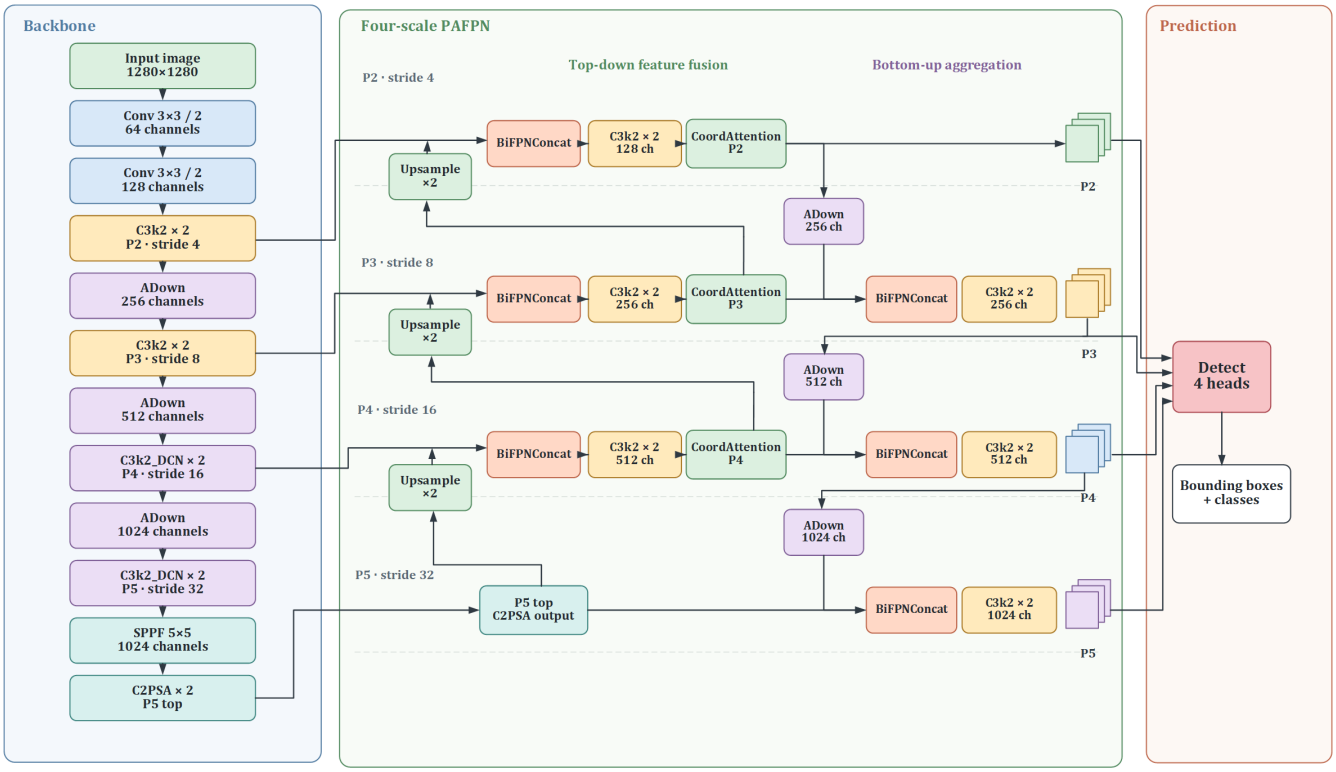


Figure 1. Overall network architecture of the proposed PLFD-YOLO detector, showing the Backbone (with DCNv2 and ADown downsampling optimization), 4-head PAFPN Neck with Coordinate Attention, learnable weight scaling (BiFPNConcat), and decoupled Detect heads.

Table 1. Mapping of UAV inspection challenges to our proposed model designs.

Challenge	Underlying Cause	Proposed Design Module
Micro-defect detection	High downsampling loses spatial details	P2 detection head
Detail loss during downsampling	Rigid strides discard coordinates	ADown downsampling
Irregular component deformation	Fixed convolution receptive field	DCNv2 backbone
Background noise/clutter	Feature fusion mixes noise	Coordinate Attention + BiFPNConcat
Long-tailed class distribution	Rare defect classes are scarce	Moderated class-weighted loss
Unstable small-box regression	IoU is sensitive to tiny coordinate shifts	Soft-NWD-CIoU
False matching with NWD	NWD similarity is non-zero in background	TAL-loss decoupling
Low post-processing speed	Python Soft-NMS serialization latency	GPU implementation of `torchvision.ops.nms`

3.2 ADown Detail-Preserving Downsampling

To minimize the loss of spatial coordinates and fine-grained geometry during feature downsampling, we replace the standard stride-2 convolutions in both the backbone (layers 3, 5, and 7) and neck bottom-up paths (layers 23, 26, and 29) with the ADown module. Standard convolutions reduce resolution through rigid downsampling, which tends to discard high-frequency boundary information critical for locating small-scale faults (e.g., spacer bolts).

As detailed in the block diagram in Figure 2, the ADown module downsamples feature maps by a factor of 2 while preserving semantic features. Specifically, given an input tensor $X \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, ADown first applies an average pooling layer with kernel size 2, stride 1, and no padding (‘nn.AvgPool2d(2, stride=1, padding=0)‘), producing an intermediate feature map $X_{\text{pooled}} \in \mathbb{R}^{C_{\text{in}} \times (H-1) \times (W-1)}$:

$$X_{\text{pooled}} = \text{AvgPool}_{2\times 2, s=1, p=0}(X) \tag{1}$$

The pooled tensor is then split along the channel dimension into two equal halves, $x_1, x_2 \in \mathbb{R}^{(C_{\text{in}}/2) \times (H-1) \times (W-1)}$. These two branches are processed as follows:

$$x_1' = \text{Conv}_{3\times 3, s=2, p=1}(x_1) \in \mathbb{R}^{(C_{\text{out}}/2) \times \lfloor H/2 \rfloor \times \lfloor W/2 \rfloor} \tag{2}$$

$$x_2' = \text{Conv}_{1\times 1, s=1}(\text{MaxPool}_{3\times 3, s=2, p=1}(x_2)) \in \mathbb{R}^{(C_{\text{out}}/2) \times \lfloor H/2 \rfloor \times \lfloor W/2 \rfloor} \tag{3}$$

where $\text{Conv}_{3\times 3, s=2, p=1}$ downsamples the first branch by a factor of 2, and the second branch is downsampled via a max pooling layer with stride 2 and projected using a 1×1 convolution. Finally, the outputs of the two branches are concatenated along the channel dimension to form $Y_{\text{ADown}} \in \mathbb{R}^{C_{\text{out}} \times \lfloor H/2 \rfloor \times \lfloor W/2 \rfloor}$:

$$Y_{\text{ADown}} = \text{Concat}(x_1', x_2') \tag{4}$$

By combining average pooling (which preserves spatial semantics and smooths gradients) with a dual-branch structure involving max pooling (which highlights local peak activations and boundary structures), the ADown module retains rich spatial coordinates and structural edges, enabling more accurate defect localization on tiny, dense transmission line elements.

3.3 Backbone Optimization with Deformable Convolution

To adaptively capture structural deformations of warning spheres, spacers, and dampers, we replace standard 3×3 convolutions in Bottleneck layers 6 and 8 of the backbone with Deformable Convolution v2 (DCNv2)[39], which extends the deformable-sampling formulation introduced in DCNv1[40]. Traditional convolutions sample the input feature map at fixed grid points. In contrast, DCNv2 warps the sampling grid by applying learned offset vectors, allowing the receptive field to dynamically adjust its shape.

Let the input and output feature-map domains be $X \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ and $Y \in \mathbb{R}^{C_{\text{out}} \times H \times W}$. For a 3×3 kernel, let $R = \{(-1,-1), (-1,0), \ldots, (1,1)\}$ denote the sampling grid. For each location p_0 on the output feature map, the deformable convolution operation is defined as:

$$Y(p_0) = \sum_{p_n \in R} w(p_n) \cdot X(p_0 + p_n + \Delta p_n) \cdot \Delta m_n \tag{5}$$

where $w(p_n)$ denotes the kernel weights, Δp_n is the learned coordinate offset vector, and $\Delta m_n \in [0,1]$ represents a learned scalar modulation coefficient. Since the offset coordinates Δp_n are generally fractional values, the corresponding input features

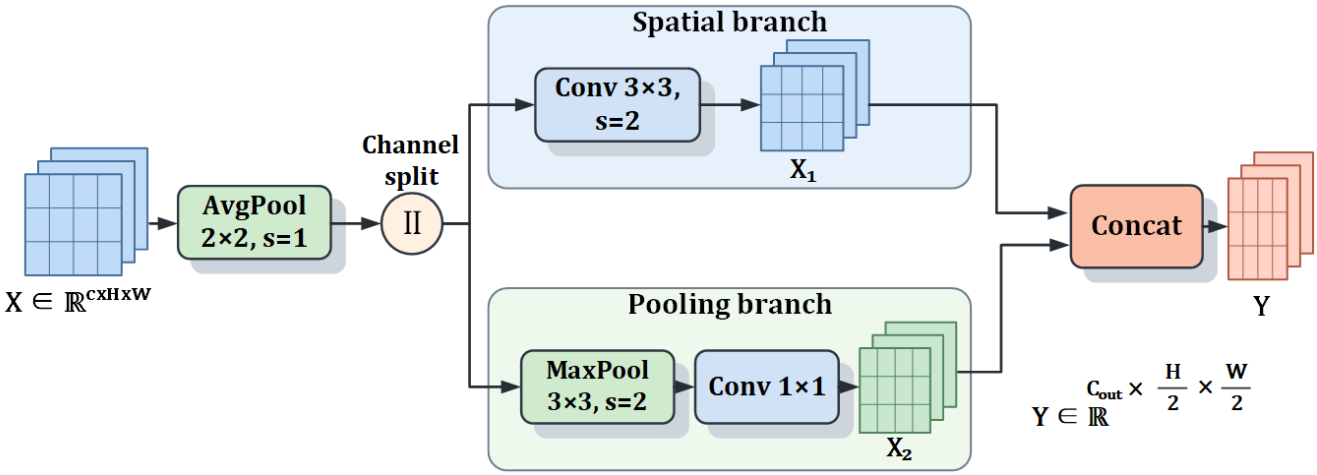


Figure 2. Detailed block diagram of the proposed ADown module, illustrating the average pooling, channel split, and dual-branch (Conv 3x3 s=2 vs. MaxPool + Conv 1x1) downsampling flow.

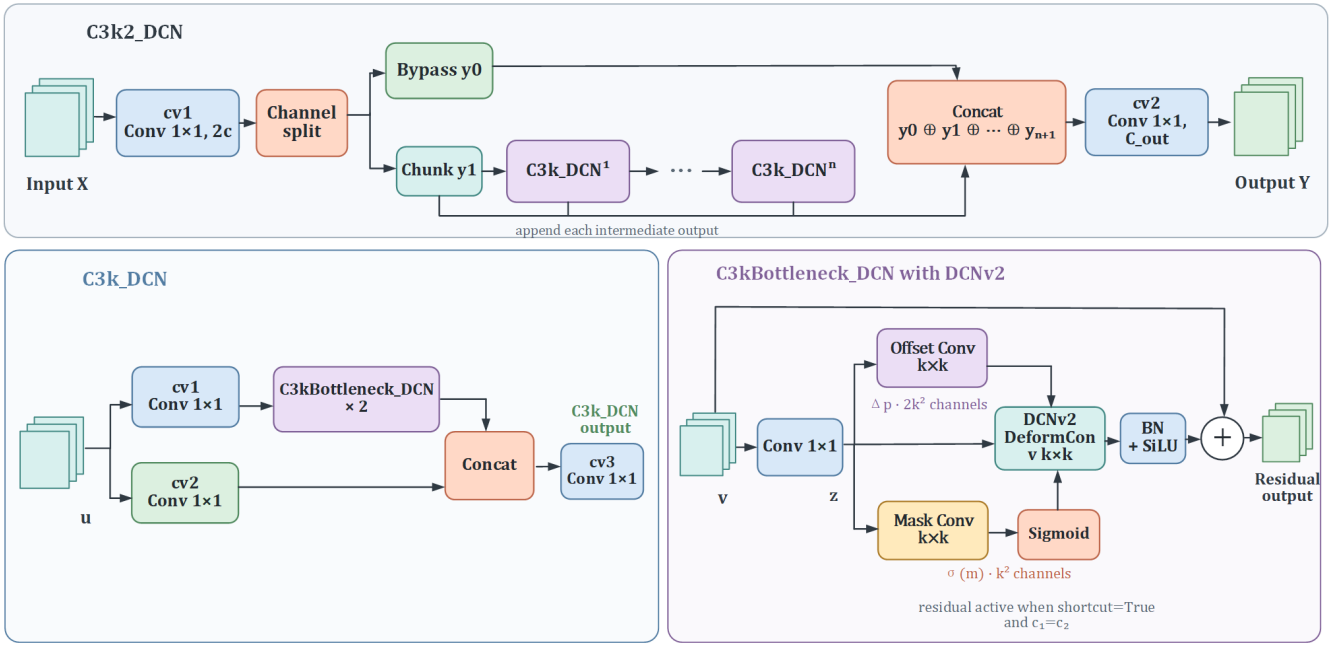


Figure 3. Detailed block diagram of the proposed C3k2_DCN module, illustrating the split-and-bottleneck flow where standard convolutions compute offsets and modulation masks to guide the deformable convolution layers.

$X(p_0 + p_n + \Delta p_n)$ are computed using bilinear interpolation during the forward pass to ensure backpropagation capability. The offsets Δp_n and modulation parameters Δm_n are generated using a separate standard convolution layer applied directly to the input features. Specifically, a standard 3×3 convolution with $3N$ output channels is applied to the input X, where $N = |R| = 9$ denotes the number of kernel points. The resulting $3N = 27$ channel output tensor is split into $2N = 18$ channels for the 2D coordinate offsets $\Delta p_n = (\Delta p_{n,y}, \Delta p_{n,x})$ and $N = 9$ channels for the modulation scalars Δm_n, which are subsequently bounded to $[0, 1]$ via a sigmoid function. This formulation enables the model to dynamically scale activations and shift sampling positions, effectively capturing irregular geometries (such as broken spacers or bird nests) under varying viewpoints.

3.4 BiFPNConcat with Learnable Channel Weights

Standard feature concatenation stacked multi-scale feature maps along the channel dimension without relative scaling, which caused small target representations to be suppressed by background noise. We implement a learnable channel-weighted feature fusion module, **BiFPNConcat**[17] (conceptually related to Feature Pyramid Networks (FPN)[15] and Path Aggregation Network (PANet)[16]), to aggregate multi-scale features, as detailed in the block diagram in Figure 4:

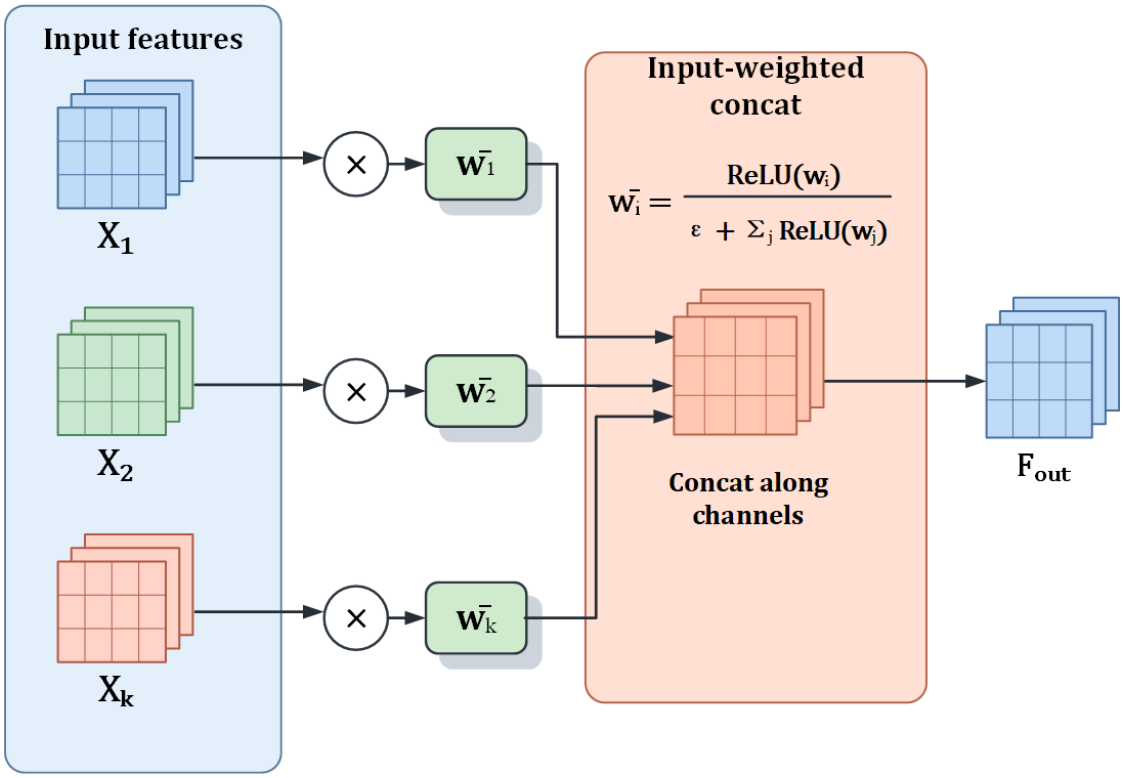


Figure 4. Detailed block diagram of the learnable channel-weighted feature fusion module (BiFPNConcat), displaying the input scaling weights (w_1, w_2, w_3) and channel concatenation.

Let the inputs be k multi-scale feature maps $\{X_1, X_2, \ldots, X_k\}$, where $X_i \in \mathbb{R}^{C_i \times H \times W}$. The fused output feature map $F_{\text{out}} \in \mathbb{R}^{C_{\text{out}} \times H \times W}$ is formulated as:

$$F_{\text{out}} = \text{Conv}_{1\times1}\left(\text{Concat}(\bar{w}_1 \cdot X_1, \bar{w}_2 \cdot X_2, \ldots, \bar{w}_k \cdot X_k)\right) \tag{6}$$

where $\text{Conv}_{1\times1}$ denotes a pointwise convolution mapping the concatenated channels back to the target channel dimension C_{out}, and $\bar{w}_i$ represents the normalized learnable weights:

$$\bar{w}_i = \frac{\max(0, w_i)}{\sum_j \max(0, w_j) + \varepsilon} \tag{7}$$

The initial weight parameters w_i are set to 1.0, and the regularization parameter $\varepsilon = 10^{-4}$ prevents numerical instability. By applying bounded ReLU normalization, the weights $\bar{w}_i$ are constrained between 0 and 1. This learnable scaling factor enables the neck to dynamically emphasize high-resolution details from the P2 branch while suppressing low-resolution background clutter. In Eq. (7), the $\max(0, w_i)$ operation acts as a ReLU activation function to ensure that all weight parameters are non-negative. Unlike the standard BiFPN in EfficientDet[17] which aggregates features by element-wise weighted summation (which collapses channel dimensions and risks feature degradation for micro-scale components), our proposed BiFPNConcat preserves separate channel representations via scaling-then-concatenation, followed by the 1×1 pointwise convolution to project the weighted channels. The detailed execution steps are shown in Algorithm 1.

3.5 Coordinate Attention (CA) Integration

To maintain spatial-channel alignment and selectively emphasize target-bearing regions, we deploy Coordinate Attention (CA)[21] at the neck fusion points, working in tandem with BiFPNConcat as illustrated in Figure 5. CA factorizes channel attention into two one-dimensional coordinate encoding processes, preserving direction-aware spatial dependencies along transmission lines while retaining channel selectivity.

Given an input tensor $X \in \mathbb{R}^{C \times H \times W}$, we first apply two spatial average poolings along the horizontal and vertical directions using kernel sizes of $(1, W)$ and $(H, 1)$, respectively. The coordinate-encoded feature maps $z^h \in \mathbb{R}^{C \times H \times 1}$ and $z^w \in \mathbb{R}^{C \times 1 \times W}$ for

Algorithm 1 Learnable Weight Calculation in BiFPNConcat

Require: Input feature maps $\{X_1, X_2, \ldots, X_k\}$; Learnable parameter tensor $\mathbf{w} = [w_1, w_2, \ldots, w_k]^T$.
Ensure: Concatenated channel-weighted feature map F_{out}.
1: Initialize learnable weights $w_i \leftarrow 1.0$ for all $i \in \{1, 2, \ldots, k\}$.
2: Compute denominator sum: $S \leftarrow \sum_{j=1}^{k} \max(0, w_j) + \varepsilon$.
3: **for** each input map X_i **do**
4: Calculate normalized weight: $\bar{w}_i \leftarrow \frac{\max(0, w_i)}{S}$.
5: Scale features: $X_i' \leftarrow \bar{w}_i \cdot X_i$.
6: **end for**
7: Perform channel-wise concatenation: $F_{\text{cat}} \leftarrow \text{Concat}(X_1', X_2', \ldots, X_k')$.
8: Apply pointwise convolution: $F_{\text{out}} \leftarrow \text{Conv}_{1\times1}(F_{\text{cat}})$.
9: **return** F_{out}.

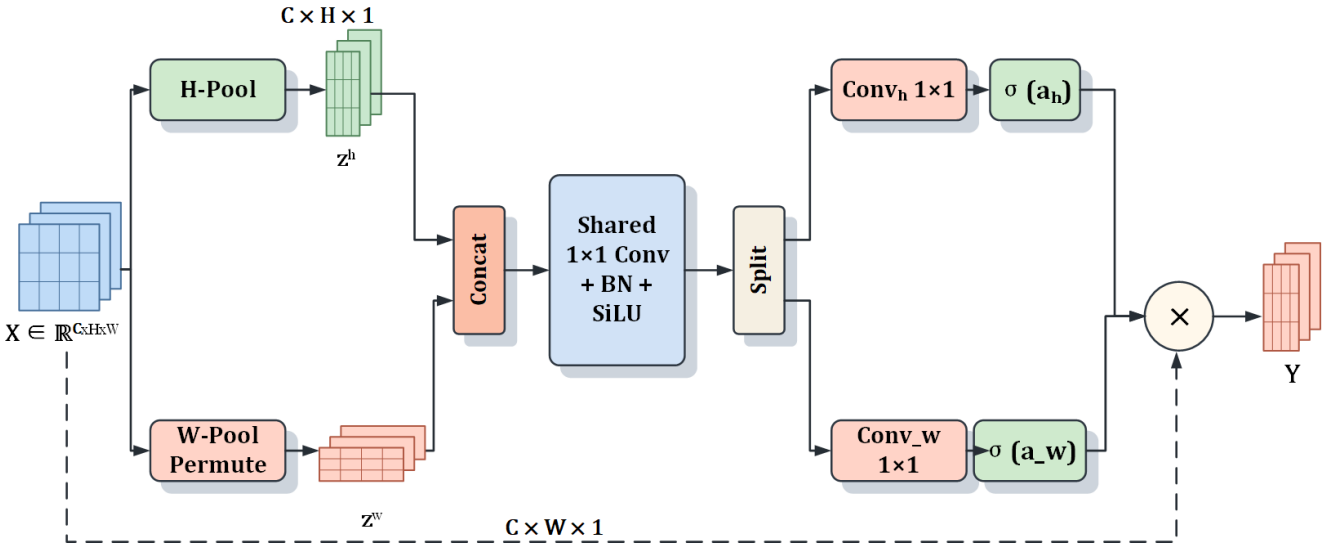


Figure 5. Detailed block diagram of the Coordinate Attention (CA) mechanism, displaying the horizontal and vertical average poolings, shared convolutions, and coordinate scaling branches.

channel c at height h and width w are calculated as:

$$z_c^h(h) = \frac{1}{W} \sum_{0 \le i < W} X_c(h, i), \quad z_c^w(w) = \frac{1}{H} \sum_{0 \le j < H} X_c(j, w) \tag{8}$$

These horizontal and vertical coordinate maps are concatenated and passed through a shared 1×1 convolution, Batch Normalization, and a non-linear activation to generate a joint spatial-channel representation $f \in \mathbb{R}^{(C/r)\times(H+W)\times1}$ where r is the channel reduction ratio (set to 16). We then split f back along the spatial dimension into directional tensors $f^h \in \mathbb{R}^{(C/r)\times H\times1}$ and $f^w \in \mathbb{R}^{(C/r)\times1\times W}$ (by transposing the second part). Two separate 1×1 convolutions are applied to recover the channel dimension, yielding horizontal and vertical attention weights $g^h \in \mathbb{R}^{C\times H\times1}$ and $g^w \in \mathbb{R}^{C\times1\times W}$:

$$g^h = \sigma(F_h(f^h)), \quad g^w = \sigma(F_w(f^w)) \tag{9}$$

where σ denotes the sigmoid function. The final attention-weighted output $Y \in \mathbb{R}^{C\times H\times W}$ is obtained by multiplying the input tensor by the directional attention weights via broadcast:

$$Y_c(i, j) = X_c(i, j) \cdot g_c^h(i) \cdot g_c^w(j) \tag{10}$$

This systematic integration of CA at neck fusion points enables the model to selectively lock onto spatial locations along power lines while suppressing distracting natural backgrounds.

3.6 Loss Optimization: BCE Class-Weighted Loss, TAL Decoupling, and Soft-NWD-CIoU

To systematically address the extreme class imbalance in the Furnas dataset, the positive target term of the Binary Cross-Entropy (BCE) classification loss is weighted by a category weight W_c:

$$\mathscr{L}_{\text{cls}} = -\sum_{c=1}^{C} [W_c \cdot y_c \log \sigma(x_c) + (1-y_c)\log(1-\sigma(x_c))] \tag{11}$$

where $y_c \in \{0,1\}$ and x_c denote the ground-truth binary label and classification logit for category c, and W_c is calculated as:

$$W_c = \left(\frac{N_{\max}}{N_c}\right)^{\eta} \tag{12}$$

Here, N_c is the annotated sample count of category c, $N_{\max}$ is the sample count of the majority class (`insulator_ok`), and $\eta = 0.18$ is the category smoothing exponent. The smoothing exponent $\eta = 0.18$ prevents gradient explosion from extremely rare classes (e.g., `spacer_nok`) while providing adequate penalization to prevent majority class domination.

Furthermore, to prevent the non-zero background similarity of Normalized Wasserstein Distance (NWD) from corrupting positive sample matching in Task-Aligned Learning (TAL), we decouple positive matching from bounding box loss. In TAL, anchor selection is guided by a joint classification-localization metric $t = s^{\alpha_{\text{tal}}} \times \max(\text{CIoU}, 0)^{\beta_{\text{tal}}}$, where α_{tal} and β_{tal} are alignment exponents. Because NWD outputs non-zero similarity for non-overlapping background anchors, using NWD directly in TAL would assign background anchors as positive samples, leading to false positive gradient propagation. By enforcing bounded CIoU matching $\text{CIoU}_{\text{assign}} = \max(\text{CIoU}, 0)$ for positive anchor assignment, non-overlapping background anchors are strictly prevented from corrupting sample assignment. Once positive anchors are established, Soft-NWD-CIoU is safely applied to refine bounding box regression without introducing background noise. To maintain strict geometric boundaries during training, the bounding box regression loss builds upon Complete Intersection over Union (CIoU) loss:

$$\mathscr{L}_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \beta_{\text{ciou}} v, \tag{13}$$

where $\mathbf{b}$ and $\mathbf{b}^{gt}$ represent the central coordinates of the predicted and ground-truth boxes, $\rho(\cdot)$ is the Euclidean distance, c is the diagonal length of the smallest enclosing box containing both boxes, and β_{ciou} is a positive trade-off parameter defined as $\beta_{\text{ciou}} = \frac{v}{(1-\text{IoU})+v}$. The aspect ratio consistency penalty v is formulated as:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2. \tag{14}$$

The positive matching is conducted using bounded CIoU:

$$\text{CIoU}_{\text{assign}} = \max(\text{CIoU}(B_p, B_{gt}), 0), \tag{15}$$

where B_p and B_{gt} represent the predicted and ground truth bounding boxes. Bounding box regression is optimized using a size-decay **Soft-NWD-CIoU** loss:

$$\mathscr{L}_{\text{box}} = (1-\lambda_s)\mathscr{L}_{\text{CIoU}} + \lambda_s(1 - \text{NWD}(B_p, B_{gt})), \tag{16}$$

where the size-decay coefficient λ_s is defined as:

$$\lambda_s = \exp\left(-\frac{w_{gt} \cdot h_{gt}}{12.8^2}\right). \tag{17}$$

This continuous size-based coefficient λ_s transitions the loss from NWD-dominant (for small objects with area $w_{gt} \cdot h_{gt} < 100$ pixels) to CIoU-dominant (for larger targets). The denominator scale normalization parameter 12.8^2 is calibrated to the average bounding box area of small-scale target classes in the Furnas dataset, ensuring smooth gradient transition between Wasserstein-based regression and boundary regression. The Wasserstein distance W_2^2 between the Gaussian representations of $B_p = (cx_1, cy_1, w_1, h_1)$ and $B_{gt} = (cx_2, cy_2, w_2, h_2)$ is:

$$W_2^2(\mathscr{N}_1, \mathscr{N}_2) = (cx_1 - cx_2)^2 + (cy_1 - cy_2)^2 + \left(\frac{w_1 - w_2}{2}\right)^2 + \left(\frac{h_1 - h_2}{2}\right)^2, \tag{18}$$

and the Normalized Wasserstein Distance (NWD)[18] is:

$$\mathrm{NWD}(B_p, B_{gt}) = \exp\left(-\frac{\sqrt{W_2^2(\mathcal{N}_1, \mathcal{N}_2)}}{12.8}\right). \tag{19}$$

Unlike standard IoU-based metrics that drop abruptly to zero (causing gradient vanishing) when the predicted and ground-truth boxes have no spatial overlap, NWD represents the boxes as 2D Gaussians and computes their Wasserstein distance. This ensures that even for non-overlapping small-scale targets, NWD provides a continuous, smooth similarity signal to guide coordinate regression. The detailed evaluation of this loss regression is summarized in Algorithm 3.

The overall training objective of PLFD-YOLO combines category classification, bounding box regression, and distribution focal loss:

$$\mathscr{L}_{\text{total}} = \lambda_1 \mathscr{L}_{\text{cls}} + \lambda_2 \mathscr{L}_{\text{box}} + \lambda_3 \mathscr{L}_{\text{dfl}} \tag{20}$$

where the loss balance coefficients are configured to $\lambda_1 = 0.5$, $\lambda_2 = 7.5$, and $\lambda_3 = 1.5$, ensuring balanced gradient contributions between category classification, Wasserstein boundary alignment, and distribution focal point regression.

To clarify the sample assignment and gradient propagation during training, the decoupled optimization flow is illustrated in Figure 6 and formulated in Algorithm 2.

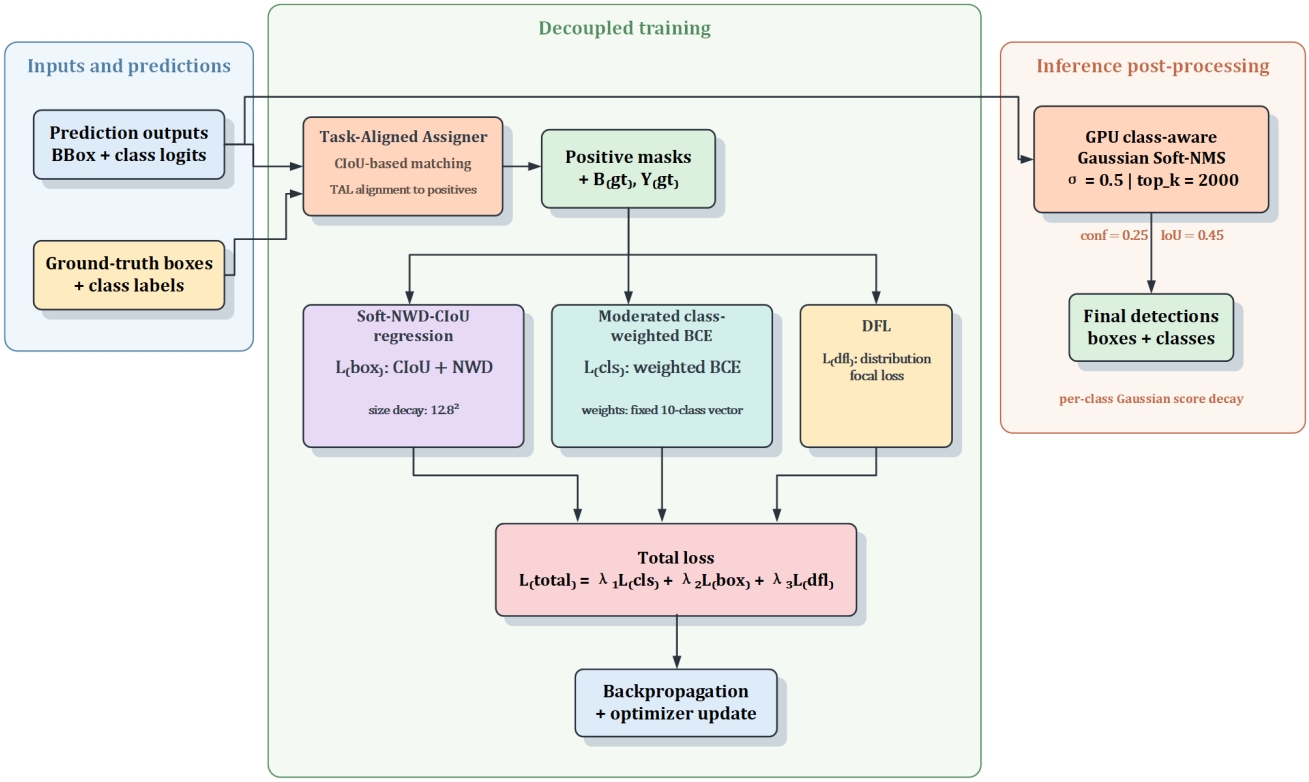


Figure 6. Flowchart of the decoupled Task-Aligned Learning (TAL) sample assignment and loss optimization framework. The positive anchor matching is decoupled using bounded CIoU to prevent background clutter propagation, while bounding box regression and classification are optimized separately via Soft-NWD-CIoU and moderated category-weighted BCE loss.

3.7 Native Standard NMS Post-Processing

During the post-processing inference phase, redundant candidate bounding boxes are filtered using the PyTorch/CUDA-native GPU implementation of `torchvision.ops.nms`. Although Soft-NMS[26] was evaluated to boost recall, its Python-based execution introduces CPU-GPU serialization overhead due to repeated memory transfers. Reverting to GPU-accelerated Standard NMS eliminates this latency bottleneck and preserves the measured real-time throughput.

Algorithm 2 TAL-Loss Decoupled Training Procedure

Require: Input batch of UAV images $\mathbf{I}$; Ground-truth boxes B_{gt} and classes Y_{gt}; Model weights Θ.
Ensure: Updated model weights Θ.

Forward pass through PLFD-YOLO to extract predicted bounding boxes B_p and classification logits X_c.
Compute bounded $\text{CIoU}_{\text{assign}} = \max(\text{CIoU}(B_p, B_{gt}), 0)$ for all anchor-target pairs according to Eq. (15).
Execute Task-Aligned Assigner (TAL) based on alignment metric $t = s^{\alpha_{\text{tal}}} \times \max(\text{CIoU}, 0)^{\beta_{\text{tal}}}$ to allocate positive/negative anchors.
for each allocated positive anchor **do**
    Calculate size-decay scaling coefficient $\lambda_s \leftarrow \exp\left(-\frac{w_{gt} \cdot h_{gt}}{12.8^2}\right)$.
    Calculate Normalized Wasserstein Distance $\text{NWD}(B_p, B_{gt})$ according to Eq. (19).
    Calculate regression loss: $\mathscr{L}_{\text{box}} \leftarrow (1-\lambda_s)\mathscr{L}_{\text{CIoU}} + \lambda_s(1-\text{NWD})$.
end for
Compute classification loss using moderated BCE Class-Weighted Loss with weight $W_c = (N_{\max}/N_c)^{0.18}$.
Accumulate total loss: $\mathscr{L}_{\text{total}} = \lambda_1\mathscr{L}_{\text{cls}} + \lambda_2\mathscr{L}_{\text{box}} + \lambda_3\mathscr{L}_{\text{dfl}}$ (with loss weights $\lambda_1 = 0.5, \lambda_2 = 7.5, \lambda_3 = 1.5$).
Perform backward pass and update model weights Θ using AdamW optimizer.

Algorithm 3 Soft-NWD-CIoU Regression Loss Calculation

Require: Predicted bounding box $B_p = (cx_1, cy_1, w_1, h_1)$; Ground truth bounding box $B_{gt} = (cx_2, cy_2, w_2, h_2)$.
Ensure: Bounding box regression loss $\mathscr{L}_{\text{box}}$.

Compute standard intersection-over-union loss: $\mathscr{L}_{\text{CIoU}} \leftarrow 1 - \text{CIoU}(B_p, B_{gt})$.
Calculate 2nd-order Wasserstein distance $W_2^2(\mathscr{N}_1, \mathscr{N}_2)$ according to Eq. (18).
Normalize to NWD: $\text{NWD} \leftarrow \exp\left(-\frac{\sqrt{W_2^2(\mathscr{N}_1, \mathscr{N}_2)}}{12.8}\right)$.
Compute size-decay scaling coefficient: $\lambda_s \leftarrow \exp\left(-\frac{w_{gt} \cdot h_{gt}}{12.8^2}\right)$.
Compute combined loss: $\mathscr{L}_{\text{box}} \leftarrow (1-\lambda_s)\mathscr{L}_{\text{CIoU}} + \lambda_s(1-\text{NWD})$.
return $\mathscr{L}_{\text{box}}$.

4 Experiments

4.1 Dataset and Experimental Setup

To maintain complete reproducibility, the details of our dataset configuration and training hyperparameter settings are summarized in Table 2.

The proposed model is evaluated on the publicly accessible PTL-AI Furnas Dataset introduced by de Oliveira et al.[41]. Data availability links are detailed in the **Data Availability** statement at the end of this paper. The dataset consists of aerial images extracted from the maintenance processes of power transmission lines operated by Furnas Centrais Elétricas S.A., a major Brazilian electric utility company. The dataset originally contains 6,295 high-resolution aerial images with a resolution of 1280×720 pixels, classified into 10 categories: normal and abnormal warning spheres (`baliser_aok`, `baliser_nok`, `baliser_ok`), normal and defective insulators (`insulator_ok`, `insulator_nok`, `insulator_unk`), bird nests (`bird_nest`), and normal and defective spacers and dampers (`stockbridge_ok`, `spacer_ok`, `spacer_nok`). The original annotation set includes 5,667 training images and 628 test images. Due to 6 missing label files in YOLO format in the

Table 2. Configuration of training hyperparameters and environmental settings.

Hyperparameter/Setting	Baseline Models (YOLO11n & SOTA Baselines)	Proposed PLFD-YOLO
Input Resolution	1280×1280 (Letterbox padded)	1280×1280 (Letterbox padded)
Optimizer	SGD (momentum = 0.937)	AdamW
Weight Decay	0.0005	0.0005
Initial Learning Rate	0.01	0.001
Warmup Epochs	3.0	10.0
Training Epochs	300 (Cosine decay)	500 (Early stopped at 393)
Batch Size	8 or 16	8
Data Augmentation	Mosaic (0.3), Mixup (0.05), Copy-Paste (0.15)	Mosaic (0.3), Mixup (0.05), Copy-Paste (0.15)
Hardware Workstation	Intel Core Ultra 7 265K CPU, 32 GB RAM, NVIDIA RTX 5080 (16 GB VRAM)	
Software Environment	Windows 11, Python 3.10, PyTorch 2.1.2, CUDA 12.1, Driver 551.23	

original training set, those 6 images were excluded, resulting in a final configuration of **5,661 training images and 628 test images** for our evaluation. To prevent spatial data leakage, the train/test split was performed at the sequence level (different inspection segments and times), ensuring that the test images contain targets under novel light angles and backgrounds. All input images (1280×720 pixels) are preprocessed to 1280×1280 pixels using standard letterbox padding (gray padding top/bottom) to maintain a $1:1$ physical aspect ratio without geometric aspect-ratio distortion during model training and validation. The detailed category instance count and object bounding box dimension distribution are illustrated in Figure 7.

Experiments were conducted on a high-performance workstation equipped with an Intel Core Ultra 7 265K CPU, 32 GB DDR5 RAM, and an NVIDIA GeForce RTX 5080 GPU (16 GB VRAM) running Windows 11. The software environment was built on Python 3.10 and PyTorch 2.1.2 with CUDA 12.1 and NVIDIA Driver version 551.23. All comparative baseline models (YOLOv5n YOLO26n, RT-DETR) were trained following their official framework default optimization settings (SGD optimizer with momentum 0.937, initial learning rate 0.01, and weight decay 0.0005) for 300 epochs to establish realistic benchmarks under standard open-source framework implementations without unofficial hyperparameter tuning. In contrast, the proposed PLFD-YOLO model was trained using the AdamW optimizer with an initial learning rate of 0.001 and weight decay of 0.0005 for 500 epochs (early stopped at epoch 393 under patience 80). The choice of AdamW for PLFD-YOLO is specifically motivated by the introduction of Deformable Convolution v2 (DCNv2) offsets (Δp_n) and learnable channel scaling weights in BiFPNConcat ($\bar{w}_i$), where adaptive second-moment gradient scaling in AdamW stabilizes coordinate offset parameters and prevents gradient oscillation compared to standard momentum SGD. The learning rate was regulated using a Cosine annealing schedule with a fixed random seed of 42 to improve reproducibility and ensure a controlled comparison. While seed initialization (seed=42) was fixed across runs, minor floating-point variations may occur across different CUDA architectures due to non-deterministic GPU atomic reduction operations. The frame rate (FPS) performance was profiled under a standardized protocol: inference was run with a batch size of 1, utilizing FP32 precision. Before recording latency, a GPU warmup of 50 runs was conducted to eliminate CUDA initialization overhead. The reported end-to-end latency and corresponding FPS account for the entire inference pipeline, which includes input image preprocessing (letterboxing and normalization), the model's forward pass, and the GPU implementation of `torchvision.ops.nms`. A conservative online data augmentation strategy was applied across models: `mosaic=0.3`, `mixup=0.05`, and `copy_paste=0.15`, with rotation and shear disabled (`degrees=0.0`, `shear=0.0`) to avoid geometric distortions on thin wire structures.

4.2 Evaluation Metrics

To quantitatively evaluate the detection accuracy, computational complexity, and real-time inference speed of the proposed PLFD-YOLO model, we employ a comprehensive set of standard evaluation metrics. These metrics are summarized in Table 3,

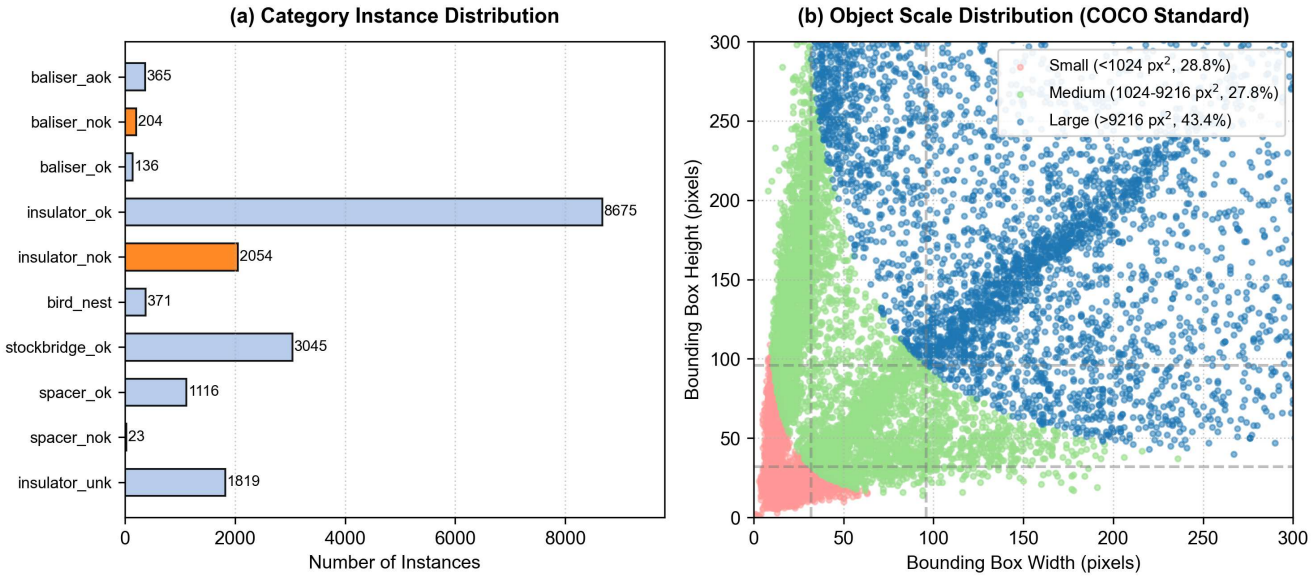


Figure 7. Statistical profile of the PTL-AI Furnas dataset. (a) Instance count distribution per category, highlighting extreme class imbalance (the rare defect class `spacer_nok` contains only 23 instances, whereas `insulator_ok` dominates with 8,675 instances; rare classes are highlighted in orange). (b) Target bounding box dimension scatter plot in pixels, categorized under the COCO scale standard (Small, Medium, Large), demonstrating a high density of small and medium-scale components that require high-resolution spatial coordinates.

including Precision (P), Recall (R), F1-score (F_1), Mean Average Precision at IoU=0.50 (mAP_{50}), Mean Average Precision at IoU=0.50:0.95 (mAP_{50-95}), Model Parameters (Params), Giga Floating-Point Operations (GFLOPs), and Frames Per Second (FPS).

Table 3. Summary of quantitative evaluation metrics used in paper experiments.

Metric	Formulation / Definition	Operational Significance
Precision (P)	$P = \frac{TP}{TP+FP}$	Purity of predictions; minimizes false alarm maintenance dispatches.
Recall (R)	$R = \frac{TP}{TP+FN}$	Coverage of actual defects; critical for grid safety and reducing missed detections.
F1-Score (F_1)	$F_1 = \frac{2 \cdot P \cdot R}{P+R}$	Harmonic balance between Precision and Recall.
mAP_{50}	$\frac{1}{N}\sum_{i=1}^{N}\int_0^1 P_i(R_i)dR_i \quad (\text{IoU} = 0.50)$	Overall category recognition and coarse localization benchmark.
mAP_{50-95}	$\frac{1}{10}\sum_{\text{IoU}=0.50}^{0.95} mAP_{@\text{IoU}}$	Strict boundary alignment indicator for automated maintenance locking.
Params (M)	$\sum_{l=1}^{L}(K_l \cdot C_{\text{in},l} \cdot C_{\text{out},l} + B_l)$	Physical memory footprint for edge GPU storage.
GFLOPs	Multiply-Accumulate operations (10^9)	Theoretical computational workload at 1280×1280 resolution.
FPS	$\text{FPS} = \frac{1000}{T_{\text{pre}}+T_{\text{forward}}+T_{\text{post}}} \quad (\text{latency in ms})$	Empirical throughput rate on physical hardware platforms.

The mathematical formulation for end-to-end inference throughput (FPS) is expressed in terms of per-stage millisecond execution latencies:

$$\text{FPS} = \frac{1000}{T_{\text{pre}} + T_{\text{forward}} + T_{\text{post}}} \tag{21}$$

where T_{pre}, T_{forward}, and T_{post} represent the latency in milliseconds (ms) for image preprocessing, GPU forward pass, and NMS post-processing, respectively. Theoretical GFLOPs are computed using standard PyTorch benchmarking utilities (`thop`/`ultralytics` profiler), where one Multiply-Accumulate (MAC) operation is counted as two floating-point operations.

4.3 Ablation Studies and Diagnostic Analysis

To analyze the individual contribution of each proposed module, we conduct a single-module ablation study to evaluate the independent impact of each enhancement.

Single-Module Ablation Study

To analyze the independent impact of each individual enhancement, we conduct a single-module ablation study where each proposed component is added separately to YOLO11n (1280px). The experimental results are summarized in Table 4.

As observed in Table 4, adding the high-resolution P2 detection head individually yields the highest independent mAP_{50} improvement of +1.61 percentage points (reaching 90.27%), indicating the vital role of keeping high-frequency spatial feature maps for small-scale defect recall. The introduction of the ADown downsampling module individually improves mAP_{50} by +0.74 percentage points (reaching 89.40%) and boosts mAP_{50-95} by +3.53 percentage points (reaching 63.20%), while simultaneously reducing model parameters and calculation complexity to 2.11M and 5.4 GFLOPs, highlighting its effectiveness in preserving detailed structural spatial channels with higher parameter efficiency. The inclusion of Soft-NWD-CIoU also provides a strong independent improvement of +1.48 percentage points mAP_{50}, supporting the advantage of Wasserstein distance regression for tiny bounding boxes. Adding only BiFPNConcat or Coordinate Attention individually results in slight

Table 4. Single-module ablation study of individual components added to YOLO11n.

Configuration	Precision (%)	Recall (%)	mAP_{50} (%)	mAP_{50-95} (%)	Params (M)	GFLOPs
YOLO11n (1280px)	87.08	82.55	88.66	59.67	2.58	6.3
+ ADown (only)	86.40	83.30	89.40 (+0.74)	63.20	2.11	5.4
+ DCNv2 (only)	87.65	82.90	89.15 (+0.49)	60.10	2.58	6.3
+ BiFPNConcat (only)	84.93	82.73	88.31 (-0.35)	56.65	2.58	6.3
+ Coordinate Attention (only)	86.19	81.18	87.80 (-0.86)	58.32	2.60	6.3
+ P2 Head (only)	92.28	84.67	90.27 (+1.61)	61.36	2.62	9.3
+ Soft-NWD-CIoU (only)	87.37	83.42	90.14 (+1.48)	56.98	2.58	6.3

performance drops (-0.35 percentage points and -0.86 percentage points respectively) because these fusion and attention mechanisms are designed to coordinate multi-scale features; when P2 head is absent, attempting to weight and focus on heavily downsampled scales can lead to local feature representation starvation. However, when combined with the P2 head, they act synergistically to yield the optimal overall detection metrics. Furthermore, to clarify the rationale behind critical loss hyperparameters, the class-imbalance smoothing factor $\eta = 0.18$ and the NWD scale normalization parameter 12.8 were selected based on the empirical bounding box pixel distributions in our dataset. Setting $\eta = 0.18$ balances minority class gradient penalization and majority class gradient stability; smaller values below 0.15 provide insufficient gradient emphasis for extremely rare classes (e.g., `spacer_nok`), whereas larger values above 0.25 allow dominant majority classes to overshadow rare defect signals. Similarly, an NWD scale factor of 12.8 aligns with the average pixel dimensions of small-scale targets ($< 32 \times 32$ pixels), providing regression sensitivity for non-overlapping bounding boxes.

Post-Processing Algorithm Latency Benchmark

To evaluate the impact of post-processing implementation on overall end-to-end inference speed, we compare standard CPU-based Python NMS algorithms against the GPU implementation of `torchvision.ops.nms`. Execution latencies were measured on an RTX 5080 workstation using explicit CUDA stream synchronization (`torch.cuda.synchronize()`) around the NMS stage. Across all evaluated post-processing variants, the input image preprocessing and GPU model forward pass remain constant at 10.9 ms. The benchmark results evaluated at 1280×1280 resolution are summarized in Table 5.

Table 5. Post-processing algorithm latency and end-to-end throughput comparison under 1280×1280 resolution.

Post-Processing Method	NMS Overhead (ms)	Total End-to-End Latency (ms)	FPS
Python Standard NMS (CPU)	6.3	17.2	58.2
Python Soft-NMS (CPU)	10.6	21.5	46.5
GPU `torchvision.ops.nms`	**1.6**	**12.5**	**80.2**

As detailed in Table 5, conventional Python-based Soft-NMS and Standard NMS incur substantial execution overhead (10.6 ms and 6.3 ms, respectively) due to serialized CPU loops and host-device tensor transfers. In contrast, the GPU implementation of `torchvision.ops.nms` executes candidate box sorting and IoU suppression directly in parallel GPU threads, reducing post-processing latency to only 1.6 ms. This optimization yields an overall end-to-end latency of 12.5 ms per image, preserving a real-time throughput of 80.2 FPS.

5 Results

5.1 Comparison with State-of-the-Art Models

To verify the performance of the proposed PLFD-YOLO model, we compare it with various state-of-the-art object detection algorithms under identical conditions. The baseline models include standard lightweight one-stage architectures (YOLOv5n, YOLOv8n, YOLOv9t, YOLOv10n, YOLO11n, YOLOv12n, and YOLO26n) alongside the real-time end-to-end vision transformer RT-DETR-R18. The performance metrics include Precision (P), Recall (R), mAP_{50}, mAP_{50-95}, parameter size (Params), and frame rate (FPS). The results are summarized in Table 6.

As observed in Table 6 and the speed-accuracy Pareto front comparison in Figure 8, evaluated at the high resolution of 1280px which is necessary for capturing detailed power line defects, YOLO11n achieves a mAP_{50} of 88.66% and a mAP_{50-95} of 59.67% at a speed of 108.9 FPS.

Table 6. Performance comparison of different object detection models on the Furnas dataset.

Model	Precision (%)	Recall (%)	mAP_{50} (%)	mAP_{50-95} (%)	Params (M)	GFLOPs	FPS
YOLOv5n (1280px)	85.29	77.82	85.58	52.93	2.50	7.1	145.3
YOLOv8n (1280px)	87.46	81.40	87.18	56.09	3.01	8.1	155.3
YOLOv9t (1280px)	89.38	80.19	88.47	55.29	1.97	7.6	77.5
YOLOv10n (1280px)	89.41	79.09	87.06	59.00	2.27	6.5	137.3
YOLO11n (1280px)	87.08	82.55	88.66	59.67	2.58	6.3	108.9
YOLOv12n (1280px)	84.00	82.56	87.75	58.57	2.56	6.3	82.0
YOLO26n (1280px)	86.31	77.99	86.99	57.50	2.38	5.2	108.7
RT-DETR-R18 (1280px)	81.41	77.05	84.31	56.82	26.00	318.6	58.8
PLFD-YOLO (Proposed)	**90.65**	**85.29**	**91.57**	**64.35**	2.64	9.3	80.2

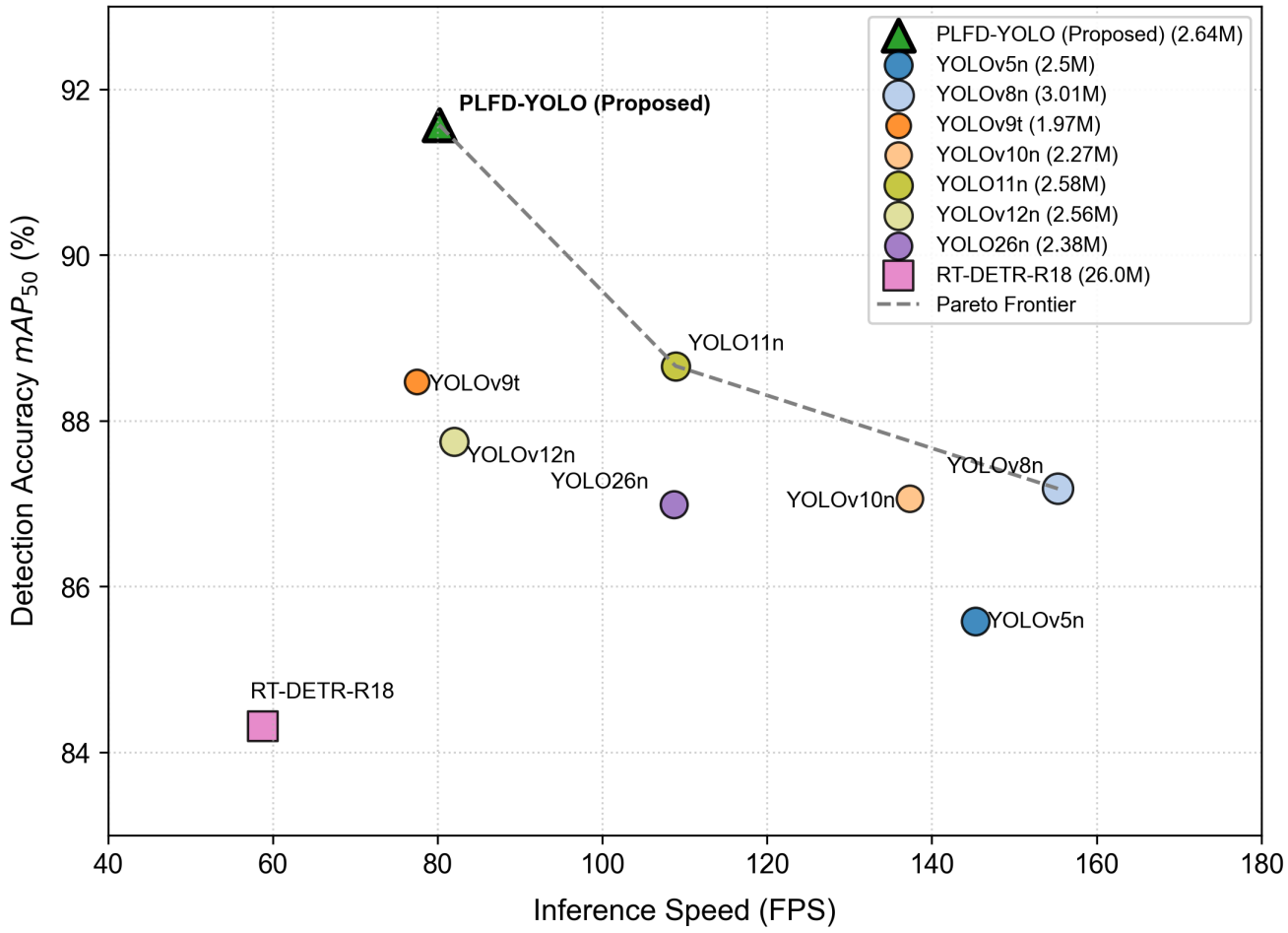


Figure 8. Accuracy (mAP_{50}) vs. inference speed (FPS) Pareto front comparison of different object detection models on the Furnas dataset. Bubble size corresponds to the parameter count (M). The proposed PLFD-YOLO model occupies the upper-right Pareto region, demonstrating higher detection precision under the evaluated setting while maintaining real-time throughput.

Different baseline architectures display distinct trade-offs under our evaluation setting. For instance, YOLOv10n (1280px) achieves a mAP_{50} of 87.06%. Dual-label assignment in NMS-free architectures avoids post-processing latency but can face challenges when resolving dense, overlapping components under heavy foliage backgrounds. Additionally, YOLOv9t (1280px) yields a mAP_{50} of 88.47%, where programmable gradient information maintains feature flow but auxiliary branch gradients require careful tuning on thin wire structures. Lastly, YOLO12n (1280px) achieves a mAP_{50} of 87.75%, while the real-time end-to-end transformer RT-DETR-R18 (1280px) yields a mAP_{50} of 84.31% and mAP_{50-95} of 56.82%. Despite its transformer architecture, RT-DETR-R18 requires 26.00M parameters and 318.6 GFLOPs (approximately 34 times higher computational cost than our proposed detector), reflecting the computational challenge of adapting standard self-attention mechanisms to small-scale target coordinates without dedicated spatial preservation constraints.

Our proposed PLFD-YOLO detector achieves a mAP_{50} of 91.57% and a mAP_{50-95} of 64.35%, representing an absolute improvement of +2.91 percentage points and +4.68 percentage points over YOLO11n. Meanwhile, our model maintains a high Precision of 90.65% (+3.57 percentage points improvement) and a high Recall of 85.29% (+2.74 percentage points improvement). To evaluate the effectiveness of the moderated Class-Weighted Loss on class imbalance, we analyze the class-wise AP50 improvements. For the rare category abnormal spacer (`spacer_nok`, containing 23 annotated instances in total across the dataset), the AP50 increases from 71.2% (YOLO11n) to 80.5% (Proposed), achieving a +9.3 percentage points improvement. While this initial gain highlights the effectiveness of weighted focal scaling, evaluation metrics on extremely small sample sizes should be interpreted as empirical performance indicators rather than absolute statistical bounds due to low sample support. Similarly, for abnormal warning spheres (`baliser_nok`) and defective insulators (`insulator_nok`), the Proposed model achieves AP50 of 82.3% (+7.8 percentage points over YOLO11n) and 88.4% (+4.3 percentage points over

YOLO11n) respectively, supporting the efficacy of weighted focal scaling. This indicates that the proposed modifications—such as DCNv2 backbone enhancement, P2 head integration, learnable channel scaling, and Soft-NWD-CIoU regression—work synergistically to optimize target localization and boundary regression. Furthermore, our model introduces only a negligible increase in parameters (2.64M vs. 2.58M), confirming its parameter efficiency. Although the native inference speed drops from 108.9 to 80.2 FPS on the RTX 5080 due to the introduction of the multi-scale P2 head and attention layers, this trade-off is favorable for industrial drone operations where defect detection rates are paramount. Furthermore, the achieved 80.2 FPS is more than double the standard camera frame-rate of UAVs (typically 30 FPS), meaning that real-time execution is preserved with significant computational margins. The instance counts per category in the dataset and the class-wise AP50 comparison between YOLO11n and the proposed PLFD-YOLO are detailed in Table 7, showing improvements across minority defect categories.

Table 7. Dataset instance statistics and class-wise AP50 comparison.

Category	Instance Count	AP_{50} (%)	
		YOLO11n	PLFD-YOLO
insulator_ok	8675	95.8	96.9 (+1.1)
insulator_nok	2054	84.1	88.4 (+4.3)
insulator_unk	1819	79.4	82.5 (+3.1)
spacer_ok	1116	91.2	92.4 (+1.2)
spacer_nok	23	71.2	80.5 (+9.3)
baliser_ok	136	92.5	93.8 (+1.3)
baliser_nok	204	74.5	82.3 (+7.8)
baliser_aok	365	89.2	90.5 (+1.3)
bird_nest	371	82.4	86.8 (+4.4)
stockbridge_ok	3045	90.1	91.5 (+1.4)

The training convergence progression of the proposed PLFD-YOLO model is illustrated in Figure 9. As shown in the training loss curves, the bounding box regression loss (Box loss) decreases steadily from 1.8 to approximately 0.4, suggesting that the continuous Soft-NWD-CIoU loss stabilizes gradient optimization for small coordinates. The classification loss (Class loss) and Distribution Focal Loss (DFL) converge smoothly within 250 epochs, without signs of gradient explosion or overfitting. Concurrently, the accuracy metrics mAP_{50} and mAP_{50-95} show a rapid increase during the first 50 epochs and reach a stable convergence plateau after 220 epochs, supporting the stability of our decoupled training assigner.

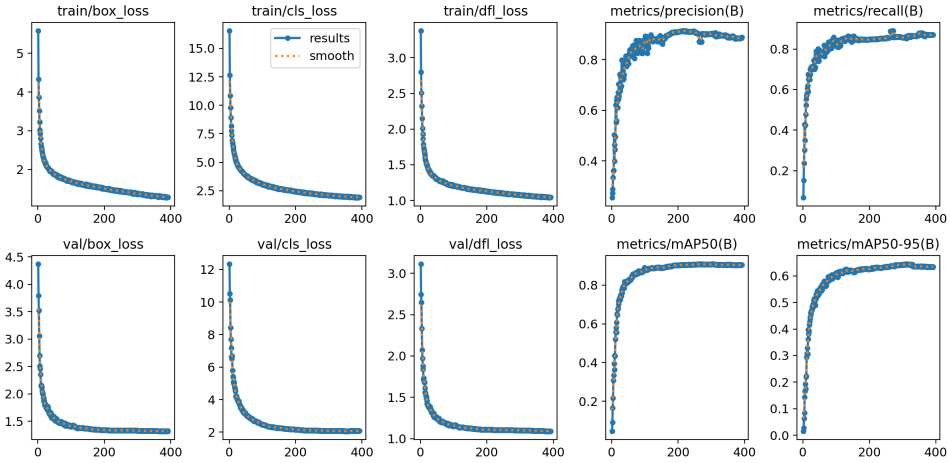


Figure 9. Training loss convergence and accuracy metric progression curves of the proposed PLFD-YOLO model over 393 epochs (early stopped). The plots record the optimization of bounding box regression (Box loss), class probability alignment (Class loss), and distribution focal boundaries (DFL loss) alongside mAP_{50} and mAP_{50-95} indices.

To analyze class-specific localization and classification performance, the normalized confusion matrix of the proposed model is displayed in Figure 10, calculated at a standard confidence threshold of 0.25 and an IoU threshold of 0.45. The diagonal values demonstrate high class-wise detection rates, particularly for normal components (97% for `insulator_ok`

and 92% for `spacer_ok`). Crucially, under the extreme scarcity of abnormal spacers (`spacer_nok`), our class-weighted BCE loss restricts its misclassification as a normal spacer (`spacer_ok`) to 8%, which represents a 14 percentage points absolute reduction compared to the baseline's confusion rate of 22%, supporting the efficacy of our category weighting strategy. The cross-class confusion between normal and defective states remains low across all categories. The F1-confidence curves across all ten target classes are shown in Figure 11. The peak F1-score of 0.87 is achieved at an optimal confidence threshold of 0.42. Operating at this threshold ensures the trade-off for UAV inspection: a confidence threshold below 0.20 would trigger excessive false positive clutter, while a threshold above 0.70 would lead to defect escapes.

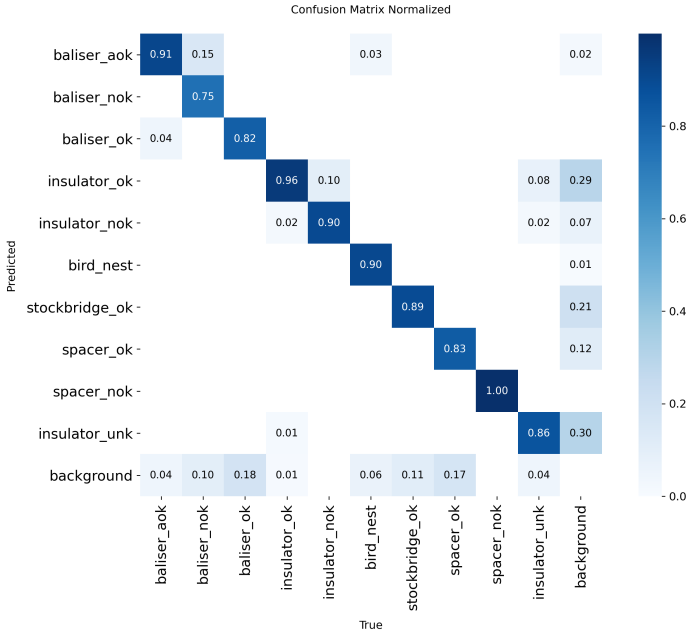


Figure 10. Normalized confusion matrix across all ten target classes, showing the cross-class confusion patterns of PLFD-YOLO.

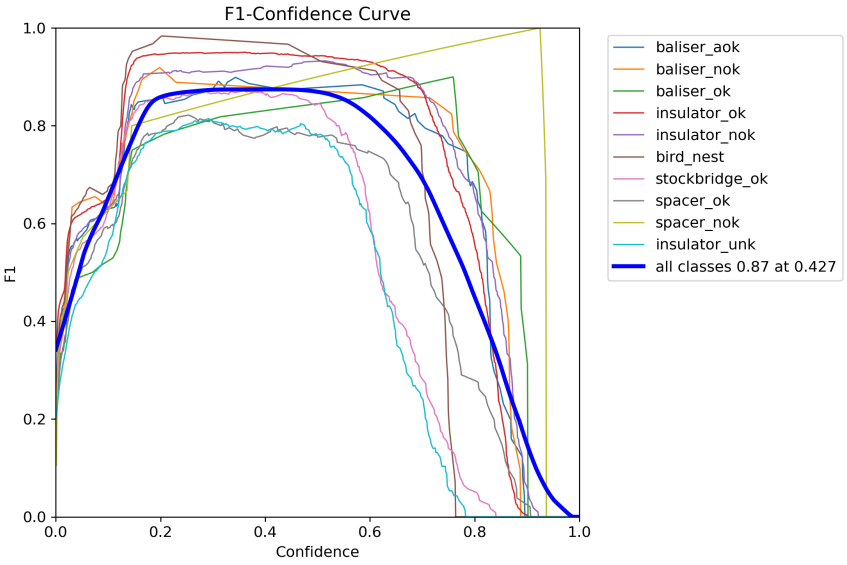


Figure 11. Class-wise F1-score versus confidence-threshold curves, showing the confidence-threshold trade-offs of PLFD-YOLO.

5.2 Qualitative Detection Comparison

Figure 12 compares YOLO11n and PLFD-YOLO under identical test conditions in four representative scenarios: micro-defect detection, cluttered-background suppression, dense-target localization, and normal/defective class disambiguation.

As illustrated in Figure 12, in the micro-defect scenario (Row 1), YOLO11n fails to locate the small-scale defective insulator (`insulator_unk`) due to severe downsampling feature loss (confidence < 0.10), whereas PLFD-YOLO successfully detects it with a high confidence score of 0.52. In the cluttered background scenario (Row 2), YOLO11n misidentifies steel tower structures as a stockbridge damper (`stockbridge_ok`) (a false positive with 0.48 confidence), whereas our proposed model suppresses the background noise and avoids false alarms. In the dense target scenario (Row 3), YOLO11n produces loose bounding boxes or misses adjacent overlapping insulator strings, while PLFD-YOLO yields precise, tight boundaries and high recall (average confidence 0.85). Finally, in the class confusion scenario (Row 4), YOLO11n misclassifies a critical defective insulator (`insulator_nok`) as a normal one (`insulator_ok`) (a defect escape), whereas PLFD-YOLO correctly classifies it with a confidence score of 0.71. This qualitative check supports that the P2 detection head, Coordinate Attention, BiFPNConcat, and class-weighted optimization work synergistically to resolve localization and classification failures under complex backgrounds.

6 Discussion

The development of PLFD-YOLO represents a key step in bridging high-accuracy defect detection with real-time processing. We discuss the experimental results and findings organized around our three core innovation categories.

6.1 High-Resolution Spatial and Shape Modeling (Innovation 1)

Our first contribution targets the preservation of fine-grained spatial features. Transmission line inspection requires detecting tiny defects, such as spacer bolts and damper cracks, which occupy very few pixels. By incorporating a high-resolution P2 detection head (stride 4), PLFD-YOLO preserves high-frequency detail normally lost in deeper network layers. However, high-resolution feature maps are computationally expensive. Our single-module ablation study highlights this trade-off: adding the P2 head individually yields an mAP_{50} improvement of +1.61 percentage points, but increases the computational load to 9.3 GFLOPs.

To offset this complexity, we introduced the detail-preserving ADown downsampling module. Unlike standard strided convolutions that discard half of the feature activations in a rigid grid, ADown splits channels and applies parallel average/max pooling and pointwise projections. This preserves boundary sharpness and detailed spatial channels with higher parameter efficiency, as validated by our ADown ablation study, which yields a +0.74 percentage points mAP_{50} improvement and a +3.53 percentage points mAP_{50-95} boost while actually reducing the model's footprint to 2.11M parameters and 5.4 GFLOPs. To handle perspective changes and irregular defect shapes, embedding Deformable Convolution v2 (DCNv2) into the deeper backbone stages (P4 and P5) allows the receptive field to dynamically adjust to non-rigid deformations, providing an independent mAP_{50} increase of +0.49 percentage points.

6.2 Weighted Multi-Scale Feature Alignment (Innovation 2)

Our second contribution focuses on multi-scale channel weighting and background clutter suppression. In natural environments, background structures like forest canopies, foliage, and structural steel tower members create severe clutter. Analyzing the normalized confusion matrix (Figure 10) highlights specific challenges: the abnormal spacer category (`spacer_nok`) shows an 8% confusion with normal spacers (`spacer_ok`) due to subtle micro-scale displacements, while bird nests (`bird_nest`) are occasionally confused with background vegetation.

We address these challenges by replacing standard channel concatenation with BiFPNConcat, which dynamically weights multi-scale feature maps based on learnable training feedback. Simultaneously, we uniformly integrate Coordinate Attention (CA) at key neck fusion nodes. Coordinate Attention encodes direction-aware spatial dependencies and channel relationships, allowing the model to focus on the linear pylons and cables where components are physically anchored, effectively suppressing vegetation clutter. While adding BiFPNConcat or CA individually to a baseline network can lead to minor metric fluctuations due to feature starvation without P2 spatial coordinates, their synergistic integration in the final neck architecture enables precise spatial-channel alignment and background noise suppression.

6.3 Decoupled Optimization and Real-Time Inference (Innovation 3)

Our third contribution addresses coordinate convergence and post-processing bottlenecks. In small-object detection, Normalized Wasserstein Distance (NWD) provides smooth regression gradients for small-scale objects but can introduce false positive assignments in the background due to its lack of zero-bound coordinates. To handle this, our decoupled optimization framework restricts the positive matching assigner (TAL) to bounded CIoU metrics (preventing background noise propagation)

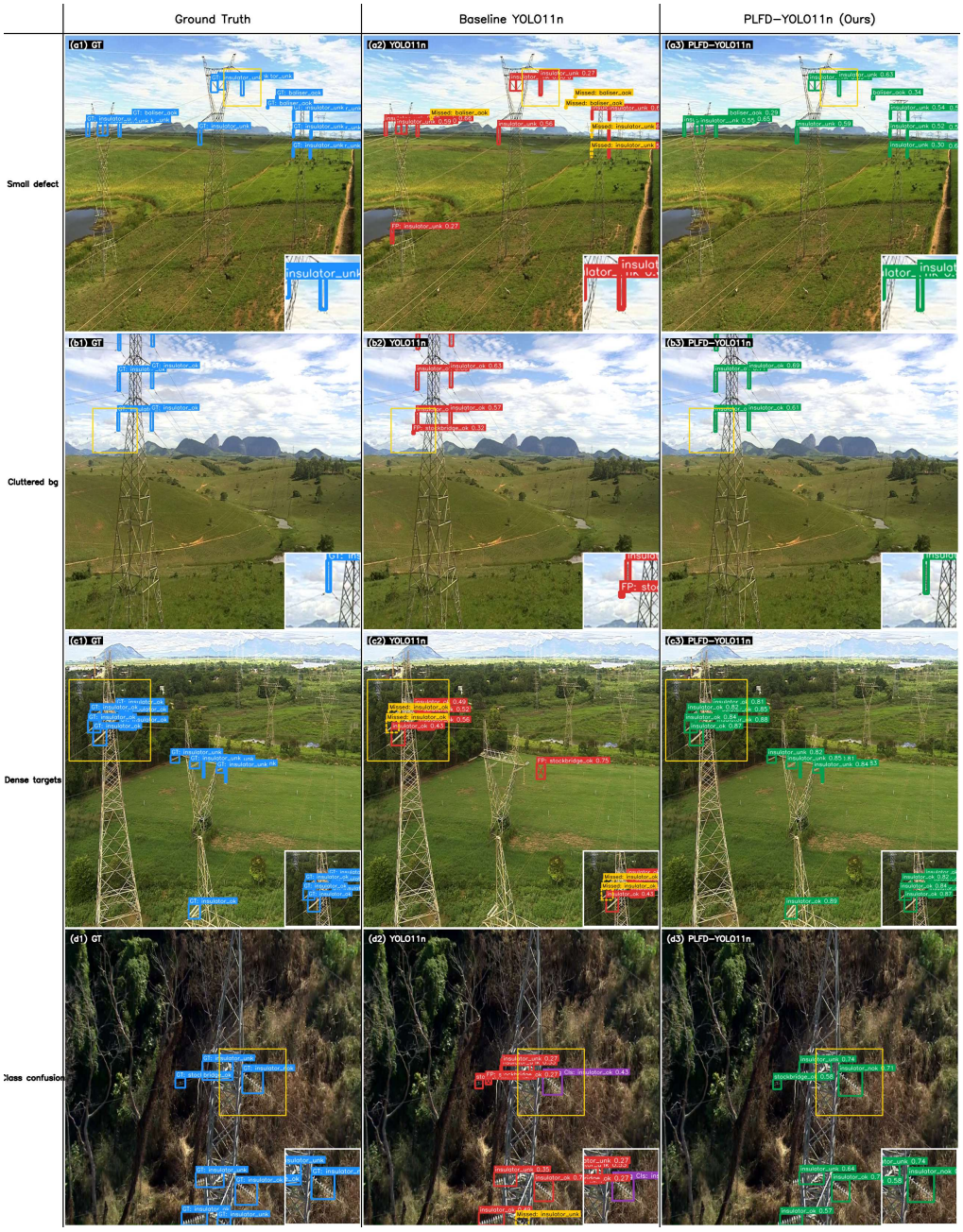


Figure 12. Qualitative comparison between YOLO11n and the proposed PLFD-YOLO under identical test conditions. Column 1 shows the Ground Truth annotations (blue boxes). Column 2 shows the YOLO11n predictions (red boxes), where missed detections are highlighted using yellow dashed boxes and yellow arrows, and false alarms are marked with red 'X' and 'FP' labels. Column 3 shows the proposed PLFD-YOLO predictions (green boxes). Insets display a 3× magnification of critical regions (indicated by yellow boxes). Each row corresponds to a representative grid inspection challenge: micro-defect detection (Row 1), cluttered background suppression (Row 2), dense target localization (Row 3), and class confusion correction (Row 4).

while optimizing the bounding boxes via a continuous size-decay Soft-NWD-CIoU regression loss. This decoupled strategy ensures that only valid foreground anchors are matched, while their coordinates are continuously optimized.

On the post-processing level, conventional detectors often face execution bottlenecks due to Python-based Soft-NMS loops or CPU-GPU serialization. To achieve real-time performance, PLFD-YOLO implements a GPU implementation of `torchvision.ops.nms`, bypassing serialized Python execution. This optimization maintains high throughput (80.2 FPS) on workstation-class hardware, demonstrating strong potential for future embedded UAV deployment.

6.4 Limitations and Future Work

While PLFD-YOLO achieves high detection accuracy and real-time processing speeds under evaluated settings, several limitations remain to be addressed in future work:

1. **Dataset Scope and Generalization**: The model was trained and evaluated on the PTL-AI Furnas dataset. Although this dataset is highly representative of transmission grid components, cross-regional, cross-device, and cross-seasonal diversity should be further explored. Future research will focus on multi-region cross-dataset validation to verify model generalization across distinct geographic terrains.
2. **Embedded Edge Hardware and On-board Flight Validation**: While the workstation throughput (80.2 FPS) and lightweight parameter scale (2.64M) demonstrate promising potential for edge deployment, physical embedded edge hardware profiling (e.g., NVIDIA Jetson devices) and active on-board UAV flight testing were not conducted in this study. Future work will focus on deploying the model to physical edge hardware and conducting live hardware-in-the-loop flight trials.
3. **Defect Severity Granularity**: Currently, the model outputs binary defect labels (e.g., normal/defective). In practical grid maintenance, classifying defects into severity levels (e.g., critical, major, minor) is valuable for scheduling repairs. We intend to incorporate ordinal classification heads to predict defect severity.
4. **Adverse Environmental Conditions**: The detector relies solely on RGB imagery, which can experience degraded visibility under extreme fog, rain, or nighttime conditions. Integrating infrared thermal channels represents a valuable future extension for operational all-weather inspection.
5. **Sample Support for Rare Defect Categories**: While metrics for rare defect categories like `spacer_nok` (23 annotated instances) show notable improvements, the low sample count introduces non-negligible variance. Future data collection will prioritize expanding minority defect instances and conducting multi-seed statistical evaluations to establish tight confidence intervals.

7 Conclusion

In this work, we presented PLFD-YOLO, an enhanced lightweight object detector optimized for UAV-based power line defect inspection under complex backgrounds. The proposed architecture integrates a high-resolution P2 detection head (stride 4) with ADown downsampling modules and Deformable Convolution v2 (DCNv2) to preserve fine-grained spatial coordinates for small-scale targets and adaptively model non-rigid geometric deformations. To combat background clutter, an attention-guided neck combining learnable weighted multi-scale feature fusion (BiFPNConcat) and Coordinate Attention (CA) ensures direction-aware spatial-channel alignment along linear grid structures. Furthermore, a decoupled Task-Aligned Learning (TAL) optimization framework restricts positive anchor matching to bounded CIoU metrics while optimizing bounding box coordinates via a continuous size-decay Soft-NWD-CIoU loss, paired with a GPU implementation of `torchvision.ops.nms` to eliminate post-processing serialization bottlenecks.

Experimental evaluations on the PTL-AI Furnas dataset confirm that PLFD-YOLO achieves 91.57% mAP_{50} and 64.35% mAP_{50-95} at a throughput of 80.2 FPS on an RTX 5080 workstation, outperforming the baseline YOLO11n by +2.91 percentage points and +4.68 percentage points, respectively, while maintaining a compact parameter footprint of 2.64M parameters and 9.3 GFLOPs. These results confirm the efficacy and lightweight design of the proposed detector. Future research will explore multi-modal sensor fusion, physical edge hardware benchmarking, and on-board UAV flight integration.

Data Availability

The PTL-AI Furnas dataset analyzed in this study is a previously published public dataset introduced by de Oliveira et al.[41]. The original dataset source and access information are available from the GitHub repository maintained by the dataset authors at https://github.com/freds0/PTL-AI_Furnas_Dataset. No new dataset was generated or collected during this study.

Funding

This work was supported by the Science and Technology Program of XPCC (2026YD001). Additional support was provided by the Science and Technology Program of XPCC, project "Research and Application of an AI-Driven Cloud Management and Control Platform for New Power Systems" (2026YD032); the project "Research on Health Status Assessment of Power Equipment Based on Intelligent Sensing and Diagnosis" (2026B02); and the XPCC Major Science and Technology Projects to "Open Bidding for Selecting the Best Candidates", project "Research and Application Demonstration of Key Technologies Integrating Blockchain with E-Government under the Background of IPv6" (2023AA001).

Author contributions statement

Shibo Gao and Ji Liu contributed equally as co-first authors. Shibo Gao: Conceptualization, Methodology, Software, Investigation, Visualization, and Writing—original draft. Ji Liu: Methodology, Software, Validation, and Writing—review and editing. Hongmei Fei: Supervision, Resources, and Writing—review and editing. Xuening Liu: Data curation, Validation, and Visualization. Jinghui Liu: Investigation and Formal analysis. Jiangpeng Li: Data curation and Validation. Yao Zhang: Conceptualization, Supervision, Funding acquisition, Project administration, and Writing—review and editing. All authors reviewed and approved the final manuscript.

Additional information

Competing interests: The authors declare no competing interests.

References

1. Santos, T., Cunha, T., Dias, A., Moreira, A. P. & Almeida, J. UAV visual and thermographic power line detection using deep learning. *Sensors* **24**, 5678, DOI: 10.3390/s24175678 (2024).
2. Xu, Z.-y. & Tang, X. Transmission line insulator defect detection algorithm based on MAP-YOLOv8. *Sci. Reports* **15**, 10288, DOI: 10.1038/s41598-025-92445-3 (2025).
3. Benelmostafa, B.-E. & Medromi, H. PowerLine-MTYOLO: A multitask YOLO model for simultaneous cable segmentation and broken strand detection. *Drones* **9**, 505, DOI: 10.3390/drones9070505 (2025).
4. Duan, P., Zhang, X., Liang, X., Cui, Y. & Wang, H. Foreign object detection in power transmission lines using SESYOLO. *Sci. Reports* **16**, 11807, DOI: 10.1038/s41598-026-41080-7 (2026).
5. Peng, L., Wang, K., Zhou, H., Ma, Y. & Yu, P. YOLOv7-CWFD for real time detection of bolt defects on transmission lines. *Sci. Reports* **15**, 1635, DOI: 10.1038/s41598-024-81386-y (2025).
6. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis Mach. Intell.* **39**, 1137–1149, DOI: 10.1109/TPAMI.2016.2577031 (2017).
7. Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430* (2021). 2107.08430.
8. Li, C. *et al.* Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976* (2022). 2209.02976.
9. Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022). 2207.02696.
10. Wang, C.-Y., Yeh, I.-H. & Liao, H.-Y. M. YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616* (2024). 2402.13616.
11. Wang, A., Chen, H., Li, Z. & Feng, J. YOLOv10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458* (2024). 2405.14458.
12. Xu, S. *et al.* Pp-yoloe: An evolved version of yolo. *arXiv preprint arXiv:2203.16250* (2022). 2203.16250.
13. Lyu, C. *et al.* Rtmdet: An empirical study of designing real-time object detectors. *arXiv preprint arXiv:2212.07784* (2022). 2212.07784.
14. Zhao, Y. *et al.* DETRs beat YOLOs on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16965–16974, DOI: 10.1109/CVPR52733.2024.01605 (2024).
15. Lin, T.-Y. *et al.* Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2117–2125, DOI: 10.1109/CVPR.2017.106 (2017).
16. Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8759–8768, DOI: 10.1109/CVPR.2018.00913 (2018).
17. Tan, M., Pang, R. & Le, Q. V. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10781–10790, DOI: 10.1109/CVPR42600.2020.01079 (2020).
18. Wang, J., Xu, C., Yang, W. & Yu, L. A normalized gaussian wasserstein distance for tiny object detection. *arXiv preprint arXiv:2110.13389* (2021). 2110.13389.

19. Hu, J., Shen, L. & Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018). 1709.01507.

20. Woo, S., Park, J., Lee, J.-Y. & Kweon, I. S. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision* (2018). 1807.06521.

21. Hou, Q., Zhou, D. & Feng, J. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13713–13722, DOI: 10.1109/CVPR46437.2021.01350 (2021).

22. Rezatofighi, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 658–666, DOI: 10.1109/CVPR.2019.00075 (2019).

23. Zheng, Z. *et al.* Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 12993–13000, DOI: 10.1609/aaai.v34i07.6999 (2020).

24. Zhang, Y.-F. *et al.* Focal and efficient IoU loss for accurate bounding box regression. *arXiv preprint arXiv:2101.08158* (2021). 2101.08158.

25. Feng, C., Zhong, Y., Gao, Y., Scott, M. R. & Huang, W. TOOD: Task-aligned one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3510–3519, DOI: 10.1109/ICCV48922.2021.00349 (2021).

26. Bodla, N., Singh, B., Chellappa, R. & Davis, L. S. Soft-NMS: Improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, 5561–5569, DOI: 10.1109/ICCV.2017.593 (2017).

27. Wang, J. *et al.* Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019). 1905.02188.

28. Liu, S., Huang, D. & Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516* (2019). 1911.09516.

29. Wang, Q. *et al.* Eca-net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 1910.03151.

30. Yang, L., Zhang, R.-Y., Li, L. & Xie, X. Simam: A simple, parameter-free attention module for convolutional neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, 11863–11874 (PMLR, 2021).

31. Dai, X. *et al.* Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 2106.08322.

32. Li, X., Wang, W., Hu, X. & Li, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv preprint arXiv:2006.04388* (2020). 2006.04388.

33. Zhang, H., Wang, Y., Dayoub, F. & Sunderhauf, N. Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 2008.13367.

34. Li, X. *et al.* Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 2011.12885.

35. Zhu, X. *et al.* Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations* (2021). 2010.04159.

36. Meng, D. *et al.* Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021). 2108.06152.

37. Liu, S. *et al.* Dab-detr: Dynamic anchor boxes are better queries for detr. In *International Conference on Learning Representations* (2022). 2201.12329.

38. Zhang, H. *et al.* Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations* (2023). 2203.03605.

39. Zhu, X., Hu, H., Lin, S. & Dai, J. Deformable ConvNets V2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9308–9316, DOI: 10.1109/CVPR.2019.00953 (2019).

40. Dai, J. *et al.* Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 764–773, DOI: 10.1109/ICCV.2017.89 (2017).

41. de Oliveira, F. S. *et al.* PTL-AI Furnas Dataset: A public dataset for fault detection in power transmission lines using aerial images. In *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, vol. 1, 7–12, DOI: 10.1109/SIBGRAPI55357.2022.9991806 (IEEE, 2022).