

Article

YOLOv8n-FEL: A Lightweight Visual Detector with a Progressive Compress-Then-Enhance Architecture for Autonomous Driving

Lei Li ^{1,2,*} , Xingrong Cheng ^{1,2}, Xiaofeng Yin ^{1,2}  and Runyu Mao ³¹ School of Automotive and Transportation Engineering, Xihua University, Chengdu 610039, China² Vehicle Measurement Control and Safety Key Laboratory of Sichuan Province, Xihua University, Chengdu 610039, China³ College of Electronic and Information Engineering, Tongji University, Shanghai 200092, China

* Correspondence: lilei@mail.xhu.edu.cn; Tel.: +86-1808-196-5903

Abstract

Real-time visual object detection on resource-constrained in-vehicle platforms requires a practical balance among detection accuracy, model size, computational complexity, and inference speed. Existing lightweight detectors may sacrifice accuracy, particularly for small and occluded objects, when model scale is compressed aggressively. To address this trade-off, this study investigates YOLOv8n-FEL, a lightweight visual detector built on YOLOv8n with a progressive “compress-then-enhance” architecture. The design contains three architecture-level modifications. First, the standard Bottleneck blocks in the C2f module are replaced with FasterBlock from FasterNet, forming the C2F-Faster module to reduce parameter count and computational cost. Second, Efficient Multi-Scale Attention (EMA) is integrated through the C2F-Faster-EMA configuration to refine features for small and occluded objects. Third, Large Separable Kernel Attention (LSKA) is introduced at the SPPF stage to refine multi-scale pooled features without quadratic complexity growth. Experiments on KITTI show that YOLOv8n-FEL reduces the parameter count from 3.006 M to 2.582 M and GFLOPs from 8.1 to 6.7, corresponding to reductions of approximately 14.1% and 17.3%, respectively. Under this lightweight setting, precision increases from 88.8% to 90.6% and mAP@0.5 increases from 89.1% to 89.5%; however, recall, mAP@0.5:0.95, and measured inference latency do not surpass the YOLOv8n baseline. At the class level, the main gain is observed for Cyclist detection, with AP@0.5 increasing from 89.9% to 92.6%, whereas Pedestrian detection and high-IoU localization remain challenging. Direct evaluation on DAIR-V2X and Carla simulation is used as supporting evidence under domain shift and controlled scenario variations rather than as deployment-level robustness validation. Overall, YOLOv8n-FEL is best understood as a lightweight accuracy–complexity trade-off model.



Academic Editors: Weiping Zhang and Jingyu Cong

Received: 5 June 2026

Revised: 7 July 2026

Accepted: 14 July 2026

Published: 20 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).**Keywords:** lightweight object detection; YOLOv8; attention mechanism; autonomous driving; model compression; EMA; LSKA**MSC:** 68U10; 68T07; 68T09

1. Introduction

Real-time visual object detection is a core component of autonomous driving perception. In practical in-vehicle systems, detectors must operate under limited computational resources while maintaining reliable recognition of safety-critical traffic participants such as small pedestrians, distant vehicles, cyclists, and partially occluded objects. One-stage

detectors, particularly the YOLO (You Only Look Once) series [1–4], have been widely adopted because of their favorable speed–accuracy trade-off. Among them, YOLOv8n provides a compact baseline for resource-constrained deployment. However, further reducing model size or computational cost can risk degrading the representation of small and difficult objects, which is undesirable in autonomous driving scenarios.

Existing studies address this challenge through two complementary directions: lightweight network design and attention-based feature enhancement. Lightweight methods reduce parameters and FLOPs, but practical speed gains are often attenuated by the gap between theoretical FLOPs and wall-clock latency [5]. Attention mechanisms can improve feature discrimination, but in resource-constrained detectors, their placement and complexity must be carefully controlled to avoid excessive runtime overhead. Therefore, lightweight compression and attention-based enhancement should be jointly designed to balance model compactness, representational capacity, and runtime cost.

These observations motivate a question central to this work: how can a lightweight detector reduce redundant computation while selectively preserving or restoring feature representation where it is most useful? Instead of treating model compression and feature enhancement as separate objectives, this study explores a progressive “compress-then-enhance” strategy. The basic idea is to first reduce redundant computation in the C2f-dominated feature extraction path and then reinvest part of the saved capacity into attention-enhanced feature refinement and spatial-context modeling.

To this end, this study investigates YOLOv8n-FEL, a lightweight visual detector built upon YOLOv8n. The main contributions of this work are summarized as follows:

1. We examine a progressive “compress-then-enhance” design strategy for lightweight object detection, aiming to improve the accuracy–efficiency trade-off by combining parameter-efficient feature extraction with selective attention-based enhancement.
2. We construct YOLOv8n-FEL by integrating C2F-Faster-EMA and SPPF-LSKA into the YOLOv8n detection framework. C2F-Faster-EMA introduces Partial Convolution-based spatial mixing and EMA-based feature refinement into the C2f replacement module, while SPPF-LSKA applies large-kernel spatial attention to SPPF-style multi-scale pooled features.
3. We evaluate the proposed model on the KITTI dataset and compare it with representative lightweight YOLO detectors. The results show that YOLOv8n-FEL provides a lightweight accuracy–complexity trade-off, reducing parameters and GFLOPs while slightly improving precision and mAP@0.5, although recall, mAP@0.5:0.95, and measured inference latency do not surpass the YOLOv8n baseline.
4. We further conduct direct evaluation on DAIR-V2X and Carla without additional fine-tuning after KITTI training. These experiments are used to examine whether the observed performance trend extends to cross-dataset and controlled simulation scenarios involving domain shift, distance variation, and occlusion.

The remainder of this paper is organized as follows. Section 2 reviews related work on lightweight YOLO-based detection, efficient convolution, and attention mechanisms for feature enhancement. Section 3 details the overall architecture of YOLOv8n-FEL and its core modules. Section 4 presents experimental validation on KITTI and direct evaluation on DAIR-V2X and Carla. Section 5 discusses the accuracy–efficiency trade-off, cross-domain behavior, limitations, and future work. Section 6 concludes the paper.

2. Related Work

2.1. Lightweight YOLO-Based Object Detection

One-stage object detectors have become a major branch of real-time object detection because they directly predict object categories and bounding boxes in a single forward pass. The YOLO family has evolved through successive improvements in backbone design, multi-scale feature fusion, label assignment, and detection head design. YOLOv3 [2] introduced multi-scale prediction for objects of different sizes, YOLOv4 [3] integrated CSPDarknet53 and PANet to improve feature extraction and fusion, and YOLOv8 [4] further adopted C2f modules, an anchor-free decoupled head, and the Task-Aligned Assigner. YOLOv8n, the nano-scale variant, provides a compact baseline and is therefore suitable for studying lightweight detection in autonomous driving scenarios.

Recent lightweight detection studies often improve YOLO-style models by replacing heavy backbone or neck components, introducing efficient convolution modules, pruning redundant structures, or redesigning feature-fusion paths. These methods reduce model size and computational cost, but their effectiveness depends strongly on where compression is applied. Excessive compression in feature extraction or multi-scale fusion may weaken the representation of small, distant, or occluded objects. Therefore, lightweight detector design should consider not only the amount of compression but also the architectural location where the saved computation can be most effectively reinvested.

Recent surveys of small-object detection summarize the rapid development of datasets, benchmarks, application scenarios, and practical design guidance for small-target perception [6]. Recent YOLOv8-based lightweight studies further show that efficiency-oriented modifications can be applied at different architectural locations. Light-FC-YOLO [7] improves YOLOv8s for dense flower counting by introducing lightweight convolution, an efficient detection head, and LSKA-based feature enhancement, suggesting that lightweight design and attention-based refinement can be combined in a compact detector. For small-object and traffic-related detection, SOD-YOLOv8 [8] enhances YOLOv8 through improved multi-path feature fusion, an additional high-resolution detection layer, C2f-EMA, and PIoU loss, indicating that preserving shallow spatial details is important for small targets in aerial imagery and traffic scenes. In autonomous-driving scenarios, YOLOv8-QSD [9] improves small-object detection by combining structural re-parameterization, BiFPN-based feature fusion, and a query-based mechanism, targeting long-distance and safety-critical traffic objects.

Among these studies, FEB-YOLOv8 [10] and YOLOv8_Adv [11] are structurally related to YOLOv8n-FEL because they also introduce PConv/FasterNet-style lightweight spatial mixing into YOLOv8 and combine it with EMA attention. These works provide architectural evidence that lightweight spatial mixing and attention-based feature refinement can be jointly embedded into YOLOv8 variants. However, their detection scenarios, additional modules, and module placements differ from those of YOLOv8n-FEL. FEB-YOLOv8 is designed for underwater object detection and further introduces P2/BiFPN-based multi-scale fusion, whereas YOLOv8_Adv targets infrared electrical-equipment detection and combines FasterNet with GSConv, VoVGSCSP, EMA in the Neck, and an additional small-object detection branch. Therefore, they are discussed here as related YOLOv8 variants rather than as directly reproducible numerical baselines under the present KITTI training protocol.

Table 1 provides a concise architecture-level comparison, focusing on the shared lightweighting-and-attention design, the main architectural differences, and reproducibility considerations. The published methods target different object domains and do not provide task-specific configuration files that fully specify the reported architectures and training settings. Under such conditions, performance differences on KITTI could reflect implementation and tuning choices rather than the intrinsic effectiveness of the original algorithms.

Table 1. Simplified architecture-level comparison with related YOLOv8 variants using PConv/FasterNet-style lightweighting and EMA attention.

Method	Target Scenario	Shared Design with YOLOv8n-FEL	Main Architectural Difference	Reproducibility/Role in This Study
FEB-YOLOv8 [10]	Underwater object detection	Uses PConv/FasterNet-style C2f replacement and EMA attention	Adds P2 features and BiFPN-based multi-scale fusion; designed for underwater datasets	No task-specific YAML is reported; discussed as a structurally related method rather than a direct KITTI baseline
YOLOv8_Adv [11]	Infrared electrical-equipment detection	Combines FasterNet/PConv-style lightweighting with EMA attention	Uses GSConv, VoVGSCSP, EMA in the Neck, and an additional small-object detection branch	No task-specific YAML is reported; discussed as a structurally related method rather than a direct KITTI baseline
YOLOv8n-FEL (ours)	Traffic-object detection for autonomous driving	Uses C2F-Faster-EMA for lightweight feature refinement	Adds SPPF-LSKA for large-kernel spatial-context refinement and keeps the three-scale Detect head	Final configuration is specified in this study; evaluated on KITTI, DAIR-V2X, and Carla

Zhu et al. [12] also studied a YOLOv8-C2f-Faster-EMA variant for underwater trash detection, providing additional evidence that C2f-Faster-EMA can serve as a lightweight C2f replacement. However, their work focuses on an underwater detection scenario and mainly examines the C2f replacement itself, whereas the present study further combines C2F-Faster-EMA with SPPF-LSKA and evaluates this compress-then-enhance placement strategy for traffic-object detection. Thus, this prior work is treated as closely related evidence for the C2f replacement component rather than as the same research setting as YOLOv8n-FEL.

YOLOv8n-ALM [13] validates layer-adaptive pruning on KITTI, offering a pruning-based lightweighting reference under a comparable evaluation setting. Different from post-training pruning, YOLOv8n-FEL is positioned as a progressive architecture-level placement study: redundant computation in C2f-dominated feature extraction is first reduced, and part of the saved capacity is then selectively reinvested into EMA-based feature refinement and SPPF-LSKA-based spatial-context modeling.

2.2. Efficient Convolution and Partial Convolution

Efficient convolution operators are widely used to reduce the computational burden of convolutional neural networks. Depthwise separable convolution, popularized by MobileNet [14], decomposes standard convolution into depthwise and pointwise operations. ShuffleNet improves inter-group information exchange through channel shuffle. These designs can reduce theoretical FLOPs, but practical acceleration may be limited when memory access becomes fragmented or arithmetic intensity decreases.

FasterNet [5] analyzes this discrepancy and proposes Partial Convolution (PConv) to reduce redundant spatial computation while preserving regular memory access. Instead of applying spatial convolution to all channels, PConv applies a regular convolution to only a subset of channels and passes the remaining channels through an identity path. This design maintains the spatial modeling ability of standard convolution on selected channels while reducing unnecessary computation on the rest. For YOLOv8n, the C2f module appears frequently in both the Backbone and Neck/Head path, making it a natural target for PConv-based lightweighting.

2.3. Attention Mechanisms for Feature Enhancement

Attention mechanisms aim to improve feature representation by emphasizing informative channels, spatial locations, or contextual regions. SENet introduces channel attention through global average pooling and gating, while CBAM extends feature recalibration to

both channel and spatial dimensions. Coordinate Attention further encodes positional cues through directional pooling, making attention more sensitive to object location. However, attention modules differ substantially in how they preserve spatial structure and how much computational overhead they introduce. For lightweight traffic-object detection, global descriptor-based attention may be less effective for small or partially occluded targets if fine-grained spatial cues are overly compressed, whereas spatially aware attention can improve feature discrimination at the cost of additional latency. This motivates the use of EMA for grouped multi-scale feature refinement and LSKA for large-kernel spatial-context modeling in the proposed compact detector.

EMA (Efficient Multi-Scale Attention) [15] addresses this issue by using grouped multi-scale processing and cross-spatial interaction to preserve positional information with limited computational overhead. This makes EMA suitable for attention-enhanced refinement inside lightweight feature extraction modules. LSKA (Large Separable Kernel Attention) [16] models long-range spatial dependencies through decomposed large-kernel operations, reducing the complexity of large-kernel attention while preserving an expanded receptive field. Such large-kernel spatial modeling is relevant to the SPPF stage, where multi-scale context is already aggregated through serial pooling.

Recent YOLOv8 variants also show the usefulness of attention and context-enhancement modules. SOD-YOLOv8 [8] embeds EMA into a C2f-EMA structure to strengthen feature extraction for small-object detection, which is consistent with the need to preserve spatially informative features in compact detectors. YOLOv8-MU [17] introduces a large-kernel block, a C2fSTR module, an SPPFCSPC-EMA module, and multi-scale fusion components for underwater detection, illustrating the value of combining broader receptive fields with attention-enhanced context aggregation. Light-FC-YOLO [7] also incorporates LSKA to improve feature-detail extraction under dense-object conditions.

Different from these enhancement-oriented designs, the proposed YOLOv8n-FEL does not introduce attention modules as isolated add-ons. EMA is placed inside the C2F-Faster-EMA module after the lightweight FasterBlock transformation, while LSKA is applied after SPPF-style multi-scale pooling. This placement links compression and enhancement more explicitly: C2F-Faster-EMA addresses redundant computation in repeated C2f positions, whereas SPPF-LSKA strengthens long-range spatial context at the terminal Backbone stage.

2.4. Summary and Research Gap

Based on the above review, the present study is positioned as an architecture-level integration and placement study. Existing lightweight detection studies often emphasize reducing parameters or FLOPs, but the resulting loss of representational capacity and its compensation are not always analyzed together. Meanwhile, attention-based enhancement is frequently introduced as an accuracy-improving module, but its placement within a compact detector is not always co-designed with the compression step. The closely related work by Zhu et al. [12] has already shown the usefulness of the C2F-Faster-EMA combination for underwater trash detection. Different from that component-level replacement setting, the present work examines whether C2F-Faster-EMA can be combined with SPPF-LSKA in a traffic-object detector so that compression in the repeated C2f path and spatial-context enhancement at the SPPF stage are treated as linked architectural decisions. Post-training compression methods such as structured pruning and channel-wise distillation for YOLOv8 [18,19] provide another route for edge deployment, but they compress an already trained detector rather than co-designing lightweight feature extraction and attention-based refinement within the architecture. Recent Transformer-based real-time detectors such as RT-DETR [20] offer an NMS-free alternative to CNN-based YOLO architectures, yet their parameter budgets are generally larger than nano-scale CNN

detectors. These observations motivate the proposed “compress-then-enhance” design: redundant computation is reduced in the C2f-dominated feature-extraction path, and feature refinement and spatial-context modeling are then selectively enhanced through EMA and SPPF-LSKA.

3. Methodology

This section presents the overall YOLOv8n-FEL architecture and its core module-level modifications under a progressive “compress-then-enhance” design pipeline. Section 3.1 describes how the detector is built upon the YOLOv8n baseline while retaining the original Backbone–Neck–Head detection paradigm. Sections 3.2–3.4 then detail the three key components: C2F-Faster for lightweight feature extraction, C2F-Faster-EMA for attention-enhanced feature refinement, and SPPF-LSKA for long-range dependency modeling.

3.1. Overall Architecture of YOLOv8n-FEL

YOLOv8 [4] adopts a modular Backbone–Neck–Head detection architecture, where the Backbone extracts hierarchical features through repeated C2f modules and an SPPF module, the Neck performs PAN-FPN-style bidirectional fusion across P3/P4/P5 scales, and the Head uses an anchor-free decoupled design for classification and bounding-box regression. YOLOv8n, the nano-scale variant, is the most compact of the family and serves as the baseline for this study.

YOLOv8n-FEL retains this overall detection paradigm and the three-scale detection outputs but replaces the repeated feature-extraction and context-refinement modules. The concrete modifications are as follows:

1. **Backbone C2f replacement:** the original C2f modules at layers 2, 4, 6, and 8 are replaced with C2F-Faster-EMA.
2. **SPPF replacement:** the original SPPF module at layer 9 is replaced with SPPF-LSKA.
3. **Neck/Head-path C2f replacement:** the original C2f modules at layers 12, 15, 18, and 21 are replaced with C2F-Faster-EMA.
4. **Detection output:** the Detect head receives the outputs from layers 15, 18, and 21, corresponding to P3, P4, and P5 feature scales.

Figure 1 presents the overall architecture of YOLOv8n-FEL. The diagram follows the YOLOv8 Backbone–Neck–Head paradigm, with the replaced modules highlighted along the forward path. In the Backbone, C2F-Faster-EMA units at layers 2, 4, 6, and 8 perform attention-enhanced lightweight feature extraction, and the SPPF-LSKA module at layer 9 provides large-kernel spatial-context refinement after multi-scale pooling. In the Neck/Head path, C2F-Faster-EMA modules at layers 12, 15, 18, and 21 replace the original C2f blocks for bidirectional feature fusion and refinement. The Detect head receives the processed feature maps from layers 15 (P3), 18 (P4), and 21 (P5), corresponding to the three detection scales.

The functional roles of these modules are summarized in Table 2. C2F-Faster serves as the lightweight structural basis, C2F-Faster-EMA adds embedded attention refinement to the C2f replacement path, and SPPF-LSKA provides spatial-context refinement after multi-scale pooling. In the full YOLOv8n-FEL configuration, C2F-Faster-EMA is used as the C2f replacement and SPPF-LSKA replaces the original SPPF module. Additional configurations, including C2F-Faster, SPPF-LSKA, C2F-Faster+SPPF-LSKA, and C2F-Faster+EMA, are used in the ablation study to examine component-level and combined effects.

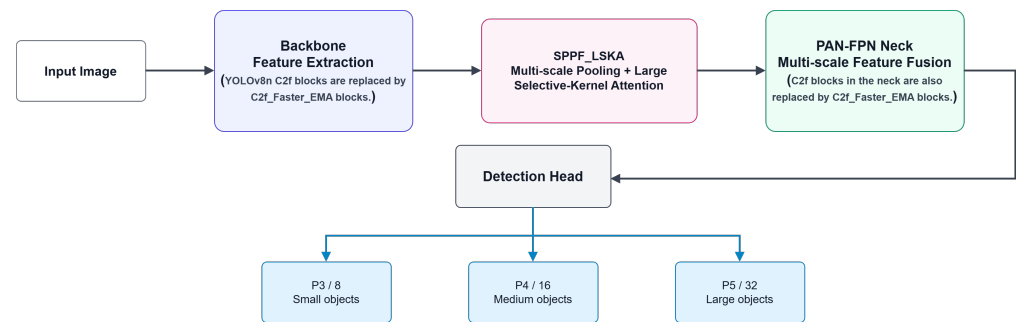


Figure 1. Overall architecture of YOLOv8n-FEL. C2F-Faster-EMA replaces the original C2f modules in the Backbone and Neck, while SPPF-LSKA replaces the SPPF module for spatial-context refinement. The Detect head produces P3, P4, and P5 outputs.

Table 2. Functional roles of the YOLOv8n-FEL modules.

Module	Applied Position	Primary Function	Mechanism
C2F-Faster	Intermediate ablation; structural basis of C2F-Faster-EMA	Parameter compression	PConv-based FasterBlock
C2F-Faster-EMA	Layers 2/4/6/8/12/15/18/21 in the full model	Lightweight feature refinement	FasterBlock with embedded EMA attention
SPPF-LSKA	Layer 9 in the full model	Spatial-context refinement	SPPF pooling followed by LSKA attention

3.2. C2F-Faster: Lightweight Feature Extraction

The C2f module in YOLOv8n repeatedly applies Bottleneck blocks for hierarchical feature extraction and therefore accounts for a non-negligible part of the computation in the nano-scale detector. C2F-Faster reduces this cost by replacing the Bottleneck units with FasterBlocks built upon Partial Convolution (PConv) [5].

For an input feature map $\mathbf{x} \in \mathbb{R}^{c \times h \times w}$, PConv applies a regular 3×3 convolution to only a subset of channels, denoted as $c_p = c/n_{div}$, while the remaining channels are passed through an identity path:

$$\mathbf{y}_{1:c_p} = \text{Conv}_{3 \times 3}(\mathbf{x}_{1:c_p}), \quad \mathbf{y}_{c_p+1:c} = \mathbf{x}_{c_p+1:c}. \quad (1)$$

With $n_{div} = 4$, only one quarter of the channels undergo spatial convolution. This design reduces spatial-convolution cost while keeping the operation less fragmented than many lightweight alternatives [5]. However, the actual inference latency still depends on the full network structure, attention overhead, implementation, and hardware platform.

The FasterBlock wraps PConv with a pointwise-convolution MLP (mlp_ratio = 2) and a residual connection with DropPath:

$$\mathbf{y} = \mathbf{x} + \text{DropPath}(\text{MLP}(\text{PConv}(\mathbf{x}))). \quad (2)$$

C2F-Faster is constructed by replacing the Bottleneck units inside each C2f module with FasterBlocks at the positions listed in Section 3.1, with the SPPF layer unchanged in this standalone configuration. In the final YOLOv8n-FEL model, these same positions are further upgraded to C2F-Faster-EMA (Section 3.3); the isolated effect of the PConv-based compression step is evaluated in the ablation study (Section 4).

3.3. C2F-Faster-EMA: Attention-Enhanced Feature Refinement

C2F-Faster reduces redundant spatial computation, but the lightweight PConv-based transformation may also weaken channel-spatial representation for small and occluded objects. C2F-Faster-EMA addresses this by embedding Efficient Multi-Scale Attention

(EMA) [15] into the FasterBlock to selectively strengthen informative feature responses with limited overhead.

Concretely, given an input $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, EMA divides channels into groups and performs attention within each group to reduce computational cost. Within each group, directional pooling along horizontal and vertical dimensions encodes positional information, and the resulting spatial descriptors interact with local convolutional features to produce channel–spatial attention weights. The grouped design and cross-spatial interaction make EMA suitable for a nano-scale detector: the per-group computation limits overhead, while directional pooling preserves positional cues useful for small and occluded traffic objects.

C2F-Faster-EMA replaces all C2f modules at the positions listed in Section 3.1. Inside each Faster_Block_EMA, EMA is inserted after the PConv–MLP branch and before residual addition. Let $\mathbf{x}'$ denote the output of the PConv–MLP branch; the residual update becomes:

$$\mathbf{y} = \mathbf{x} + \text{EMA}(\text{DropPath}(\mathbf{x}')). \quad (3)$$

Figure 2a illustrates the internal structure of the C2F-Faster-EMA module. The module is constructed upon the C2f split–transform–concatenate design, with each Bottleneck unit replaced by a Faster_Block_EMA. Inside each Faster_Block_EMA, the input first passes through the PConv layer for efficient spatial mixing on a subset of channels, followed by a pointwise-convolution MLP for channel-wise interaction. DropPath is applied for stochastic depth regularization. The EMA mechanism then refines the output through grouped multi-scale attention with cross-spatial interaction, producing channel–spatial attention weights that strengthen the representation of informative features. Finally, a residual connection adds the original input to the attention-refined output.

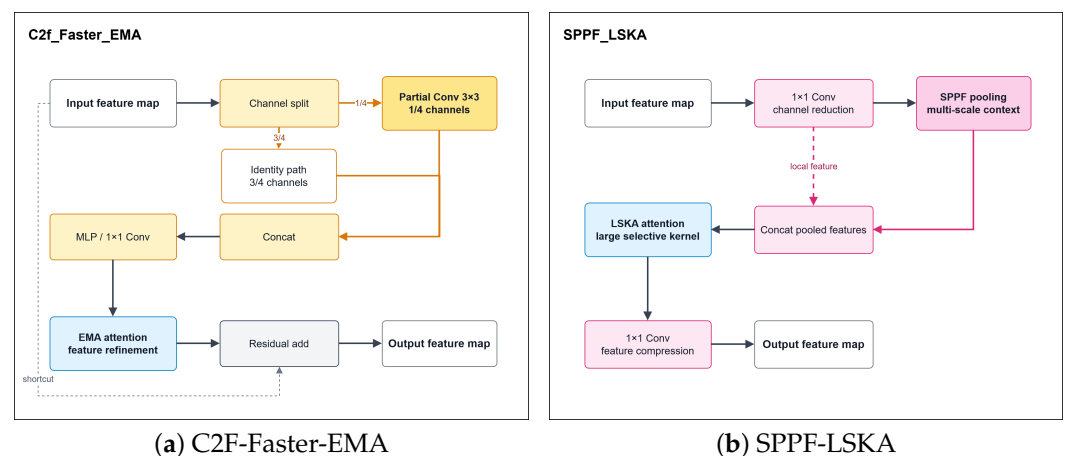


Figure 2. Core modules of YOLOv8n-FEL. (a) C2F-Faster-EMA integrates PConv-based lightweight spatial mixing and EMA-based feature refinement within the C2f replacement module. (b) SPPF-LSKA retains SPPF-style serial max-pooling and introduces LSKA for large-kernel spatial-context refinement.

In the final YOLOv8n-FEL configuration, C2F-Faster-EMA is combined with SPPF-LSKA (Section 3.4). The ablation study (Section 4) compares C2F-Faster-EMA with the C2F-Faster-only variant to isolate the contribution of EMA-based refinement.

3.4. SPPF-LSKA: Large-Kernel Spatial Context Refinement

The SPPF module in YOLOv8n aggregates multi-scale context through serial max-pooling and concatenation, but the pooling operations are parameter-free and lack learnable spatial weighting. For traffic-object detection, contextual cues around occluded or distant objects help distinguish foreground targets from background clutter. SPPF-LSKA addresses

this by retaining the SPPF pooling path and applying Large Separable Kernel Attention (LSKA) [16] after multi-scale feature concatenation, preserving the efficiency of SPPF while making context aggregation adaptive.

LSKA models long-range spatial dependencies through large-kernel attention implemented with separable depth-wise operations. A direct $K \times K$ depth-wise convolution incurs cost proportional to K^2 ; LSKA decomposes this into separable 1D operations and dilated convolutions, reducing the dominant cost to approximately linear in K . A simplified conceptual form is:

$$\mathbf{y} = \mathcal{A}_{\text{LSKA}}(\mathbf{x}), \quad \mathcal{A}_{\text{LSKA}}(\cdot) \approx \text{DWConv}_{1 \times K}(\text{DWConv}_{K \times 1}(\cdot)), \quad (4)$$

where $\mathcal{A}_{\text{LSKA}}$ denotes the large-kernel spatial-attention operation and DWConv denotes depth-wise convolution. In the inspected implementation, the SPPF-LSKA module uses $k_size=11$.

SPPF-LSKA replaces the SPPF module at layer 9 (Section 3.1). The input feature first passes through $cv1$; the resulting feature is processed by serial max-pooling and concatenation; LSKA is applied to the concatenated multi-scale features; and $cv2$ performs channel fusion:

$$\mathbf{y} = cv2\left(\text{LSKA}\left(\text{Concat}\left(\mathbf{z}, \text{MP}(\mathbf{z}), \text{MP}^2(\mathbf{z}), \text{MP}^3(\mathbf{z})\right)\right)\right), \quad \mathbf{z} = cv1(\mathbf{x}), \quad (5)$$

where MP denotes the serial max-pooling operation. Figure 2b depicts the internal structure of the SPPF-LSKA module. The module retains the original SPPF serial max-pooling path: the input first passes through $cv1$ for channel reduction, then undergoes three successive max-pooling operations whose outputs are concatenated with the initial feature map to form multi-scale pooled features. At this stage, the standard parameter-free pooling gives way to the LSKA block, which applies large-kernel spatial attention ($k = 11$) via decomposed 1D depth-wise convolutions. This learnable weighting adaptively emphasizes informative spatial regions across the pooled scales. Finally, $cv2$ fuses the attention-refined multi-scale features back to the desired channel dimension.

In the final YOLOv8n-FEL configuration, SPPF-LSKA is combined with C2F-Faster-EMA; the standalone variant is evaluated in the ablation study (Section 4) to isolate its contribution.

4. Experiments and Analysis

This section evaluates YOLOv8n-FEL from three perspectives. First, the KITTI experiments examine the accuracy–efficiency trade-off under the primary training domain. Second, the DAIR-V2X experiment evaluates cross-dataset behavior by directly testing KITTI-trained models on a different vehicle-side dataset. Third, the Carla simulation experiment assesses model behavior under controllable scenarios involving distance variation, occlusion, and scene complexity. Unless otherwise specified, model comparisons are conducted under matched training and inference settings to support fair comparison.

4.1. Experimental Setup

4.1.1. Datasets and Evaluation Protocol

Three datasets are employed for systematic validation, and each dataset serves a different experimental purpose.

KITTI Dataset: KITTI is used as the primary training and validation dataset. The dataset is converted into YOLO-format annotations with three object categories: Car, Pedestrian, and Cyclist. According to the inspected project directory, the KITTI training split contains 6001 images and 6001 corresponding label files, while the validation split contains 1480 im-

ages and 1480 corresponding label files. All models in the main KITTI comparison are trained on the KITTI training split and evaluated on the KITTI validation split under the same input resolution. This dataset provides the primary evidence for the ablation study and the comparison with representative lightweight detectors.

DAIR-V2X Dataset: The vehicle-side subset of DAIR-V2X is used to examine cross-dataset behavior beyond KITTI. In this protocol, models are trained on KITTI and directly evaluated on DAIR-V2X without additional training or fine-tuning. No DAIR-V2X samples are used for training, validation-based model selection, or parameter tuning in the direct-evaluation setting. This design is intended to evaluate whether the performance trend observed on KITTI can extend to a real-world dataset with different sensor placement and scene distribution.

Carla Simulation Data: Carla 0.9.15 is used to construct controlled driving scenarios for scenario-level evaluation. The evaluation includes near-range, mid-to-far-range, complex-environment, and real-time motion detection scenarios. The quantitative comparison focuses on three representative settings: near-range, mid-to-far-range, and complex scenes. Carla provides controllable traffic environments and ground-truth annotations, and the results are evaluated using a KITTI-compatible metric module. The Carla data are used only for direct evaluation and are not included in model training or fine-tuning. Since the Carla experiment uses KITTI-trained models directly tested in simulation and is not designed as a Carla-specific class-wise benchmark, the Carla analysis is reported at the scenario level rather than at the Pedestrian/Cyclist class level.

Within each dataset, all compared models are evaluated using the same class mapping, image split, and evaluation protocol to ensure paired comparison under consistent conditions.

4.1.2. Hardware Platform and Software Environment

All experiments are conducted under the following computing environment (Table 3):

Table 3. Hardware and software configuration.

Configuration Item	Specification
Operating System	Ubuntu 20.04
Programming Language	Python 3.10.15
Deep Learning Framework	PyTorch 2.3
Detection Framework	Ultralytics YOLOv8-based codebase
Simulation Platform	Carla 0.9.15
Key Dependencies	CUDA, Open3D, OpenCV
GPU for Training	NVIDIA GeForce RTX 4090
GPU for Inference Testing	NVIDIA RTX 3080
CPU	Intel i9-13900K

4.1.3. Implementation Details

During training, input images are uniformly resized to 640×640 , the batch size is set to 32, and the number of training epochs is set to 300. The optimizer is SGD with an initial learning rate of 0.01, final learning-rate factor of 0.01, momentum of 0.937, and weight decay of 0.0005. The warm-up stage lasts 3 epochs, with warm-up momentum set to 0.8 and warm-up bias learning rate set to 0.1. The inspected training configuration uses a random seed of 0 with deterministic mode enabled. The models are trained from scratch in the inspected configuration, i.e., `pretrained = false`, and automatic mixed precision is enabled during training (`amp = true`).

The data augmentation settings follow the inspected Ultralytics training arguments. HSV augmentation is applied with $\text{hsv_h} = 0.015$, $\text{hsv_s} = 0.7$, and $\text{hsv_v} = 0.4$; random translation and scaling are set to 0.1 and 0.5, respectively; horizontal flipping is applied with a probability of 0.5; Mosaic augmentation is enabled; MixUp and Copy-Paste are disabled. Validation is performed during training on the validation split. The inspected validation configuration uses an NMS IoU threshold of 0.7 and a maximum number of detections of 300.

For inference-speed evaluation, the available FPS script uses an image size of 640, batch size of 1, a warm-up stage of 200 iterations, and 1000 timed iterations. The default script setting uses FP32 inference unless half precision is explicitly enabled. Therefore, latency and FPS results should be interpreted together with the exact inference scope, especially whether preprocessing and NMS post-processing are included.

All compared models are trained from scratch under the inspected configuration and evaluated using identical validation settings unless otherwise stated. The reported latency and FPS results should be interpreted according to the inference script described above, which measures model processing under the specified input size, batch size, warm-up iterations, timed iterations, and precision setting.

4.1.4. Evaluation Metrics

The following metrics are adopted for quantitative assessment:

- Precision (P): the proportion of true positives among all detections.
- Recall (R): the proportion of ground-truth objects correctly detected.
- mAP@0.5: mean Average Precision across categories at IoU threshold 0.5.
- mAP@0.5:0.95: average mAP over IoU thresholds from 0.5 to 0.95 (step 0.05).
- Params (M): total parameter count, measuring model size.
- GFLOPs: computational complexity in billions of floating-point operations, calculated under the same input resolution.
- Inference Time (ms) and Frame Rate (FPS): measuring inference efficiency under the specified inference protocol.

In the Carla experiments, inference time is reported as the model processing time per frame, and FPS describes the number of frames processed per second. Following the real-time requirement commonly used in autonomous-driving perception, an inference time below 33 ms and a throughput above 30 FPS are treated as practical real-time criteria.

4.2. KITTI Evaluation

The KITTI evaluation includes two parts. First, an ablation study is conducted to examine the contribution of the proposed modules relative to the YOLOv8n baseline. Second, YOLOv8n-FEL is compared with representative lightweight or compact YOLO detectors under the same KITTI evaluation setting. These two experiments jointly evaluate whether the proposed architecture provides a favorable accuracy–complexity trade-off in the primary training domain.

4.2.1. Ablation Study

To examine the contribution of the proposed modules to the YOLOv8n baseline, we conduct an ablation study on the KITTI validation split using controlled model variants. The evaluated configurations include C2F-Faster only, SPPF-LSKA only, C2F-Faster combined with SPPF-LSKA, C2F-Faster combined with EMA, and the full YOLOv8n-FEL model. The ablation results are shown in Table 4.

Table 4. Ablation study results on the KITTI validation set.

Model	C2f-Faster	EMA	SPPF-LSKA	Params/M	GFLOPs	P	R	mAP@0.5	mAP@0.5:0.95	Inf./ms
YOLOv8n baseline	No	No	No	3.006	8.1	0.888	0.818	0.891	0.629	6.1
C2F-Faster	Yes	No	No	2.301	6.3	0.896	0.807	0.885	0.604	6.6
SPPF-LSKA	No	No	Yes	3.279	8.3	0.898	0.811	0.892	0.624	6.5
C2F-Faster+SPPF-LSKA	Yes	No	Yes	2.574	6.5	0.878	0.810	0.885	0.613	7.0
C2F-Faster+EMA	Yes	Yes	No	2.310	6.5	0.895	0.806	0.890	0.616	14.2
YOLOv8n-FEL full	Yes	Yes	Yes	2.582	6.7	0.906	0.811	0.895	0.618	13.9

Several observations can be drawn from Table 4. First, replacing the original C2f modules with C2F-Faster substantially reduces model complexity. Compared with the YOLOv8n baseline, C2F-Faster decreases the parameter count from 3.006 M to 2.301 M and GFLOPs from 8.1 to 6.3. However, this compression also reduces recall, mAP@0.5, and mAP@0.5:0.95, indicating that PConv-based lightweighting alone may weaken feature representation and localization quality.

Second, SPPF-LSKA alone slightly improves precision and mAP@0.5, increasing mAP@0.5 from 0.891 to 0.892, but it also increases parameters and GFLOPs. When SPPF-LSKA is combined with C2F-Faster without EMA, the resulting C2F-Faster+SPPF-LSKA model remains lightweight, with 2.574 M parameters and 6.5 GFLOPs, but it does not outperform the baseline in recall or mAP@0.5:0.95. This suggests that spatial-context refinement at the SPPF stage is useful, but it does not fully compensate for the representational loss caused by compression.

Third, the C2F-Faster+EMA variant achieves 2.310 M parameters and 6.5 GFLOPs, while keeping mAP@0.5 close to the baseline. Nevertheless, its inference latency increases to 14.2 ms under the present measurement protocol. This result shows that theoretical complexity reduction does not necessarily translate into lower wall-clock latency, especially when attention operations introduce additional runtime overhead.

Finally, the full YOLOv8n-FEL model provides a lightweight accuracy–complexity trade-off rather than a uniform improvement across all metrics. Compared with YOLOv8n, it reduces parameters by approximately 14.1% (3.006 M→2.582 M) and GFLOPs by approximately 17.3% (8.1→6.7). At the same time, precision increases from 0.888 to 0.906 and mAP@0.5 increases slightly from 0.891 to 0.895. However, recall decreases from 0.818 to 0.811, mAP@0.5:0.95 decreases from 0.629 to 0.618, and inference latency increases from 6.1 ms to 13.9 ms. Therefore, the ablation results support YOLOv8n-FEL as a compact detector with improved precision and AP@0.5-level performance, but not as a model that outperforms the baseline on every metric.

To further analyze class-level behavior, Table 5 reports per-class AP values for Cyclist, Pedestrian, and Car. YOLOv8n-FEL improves Cyclist AP@0.5 from 0.899 to 0.926, which is the most evident class-level gain. However, the improvements are not uniform across classes or IoU thresholds. In particular, Pedestrian AP@0.5 decreases from 0.807 to 0.792, and Cyclist AP@0.5:0.95 also decreases from 0.627 to 0.615. These results indicate that the proposed modules are more beneficial for AP@0.5-level cyclist detection than for stricter localization quality or pedestrian detection.

Figure 3 provides representative qualitative comparisons between YOLOv8n and YOLOv8n-FEL on KITTI validation scenes. The left and right panels show detections from YOLOv8n and YOLOv8n-FEL, respectively, under identical input conditions. These examples are intended to visually support the quantitative results in Tables 4 and 5, rather than to serve as standalone evidence of robustness or generalization.

Table 5. Per-class AP results on the KITTI validation set.

Model	Cyclist AP@0.5	Cyclist AP@0.5:0.95	Pedestrian AP@0.5	Pedestrian AP@0.5:0.95	Car AP@0.5	Car AP@0.5:0.95
YOLOv8n baseline	0.899	0.627	0.807	0.458	0.966	0.802
C2F-Faster	0.902	0.596	0.792	0.433	0.962	0.783
SPPF-LSKA	0.911	0.615	0.803	0.459	0.962	0.800
C2F-Faster+SPPF-LSKA	0.912	0.616	0.778	0.436	0.965	0.787
C2F-Faster+EMA	0.918	0.620	0.792	0.442	0.962	0.785
YOLOv8n-FEL full	0.926	0.615	0.792	0.450	0.965	0.790

**Figure 3.** Qualitative detection examples on the KITTI validation set. Each pair compares YOLOv8n baseline detections (left) with YOLOv8n-FEL detections (right) under identical input conditions. The examples are illustrative and should be interpreted together with the quantitative AP results.

4.2.2. Comparison with Representative Lightweight Detectors

To further contextualize the KITTI results, YOLOv8n-FEL is compared with four representative YOLO detectors: YOLOv5n, YOLOv7-tiny, YOLOv8n, and YOLOv8s. YOLOv5n and YOLOv7-tiny are selected as lightweight YOLO-family references, YOLOv8n is used as the direct baseline, and YOLOv8s serves as a higher-capacity reference model. All models are evaluated on the same KITTI validation split under the experimental protocol described in Section 4.1. The results are presented in Table 6.

Table 6. Comparative results of representative YOLO-family detectors on the KITTI dataset.

Model	mAP@0.5/%	mAP@0.5:0.95/%	Params/M	Weights/MB
YOLOv5n	84.9	53.6	1.8	3.8
YOLOv7-tiny	88.9	60.1	6.0	12.3
YOLOv8n	89.1	62.9	3.006	6.2
YOLOv8s	92.7	67.3	11.1	22.6
YOLOv8n-FEL	89.5	61.8	2.582	5.4

As shown in Table 6, YOLOv8n-FEL remains within the compact-detector regime and slightly improves mAP@0.5 over the direct YOLOv8n baseline under the present experimental setting. Compared with YOLOv8n, YOLOv8n-FEL increases mAP@0.5 from

89.1% to 89.5% while reducing the parameter count from 3.006 M to 2.582 M. However, its mAP@0.5:0.95 is lower than that of YOLOv8n, so the result should be interpreted as a lightweight accuracy–complexity trade-off rather than an across-metric accuracy improvement.

Compared with YOLOv8s, YOLOv8n-FEL uses a much smaller parameter budget and weight size, while YOLOv8s retains higher detection accuracy due to its larger capacity. This comparison is used to contextualize the model-scale difference between nano-scale and small-scale YOLO detectors. The matched comparison with YOLO11n and YOLO12n in Table 7 further contextualizes YOLOv8n-FEL against recent YOLO-family baselines, while broader comparisons with additional lightweight detectors remain future work.

Figure 4 presents the training and validation curves of YOLOv8n-FEL over the 300-epoch training process on the KITTI dataset. The top two rows show the box loss, classification loss, and DFL loss for both the training and validation sets, while the bottom row reports the validation metrics (precision, recall, mAP@0.5, and mAP@0.5:0.95). Thus, Figure 4 should be interpreted as training/validation convergence evidence rather than as a per-class evaluation. The validation metrics gradually stabilize in the later training stage, while the final quantitative comparison is reported in Tables 4 and 5.

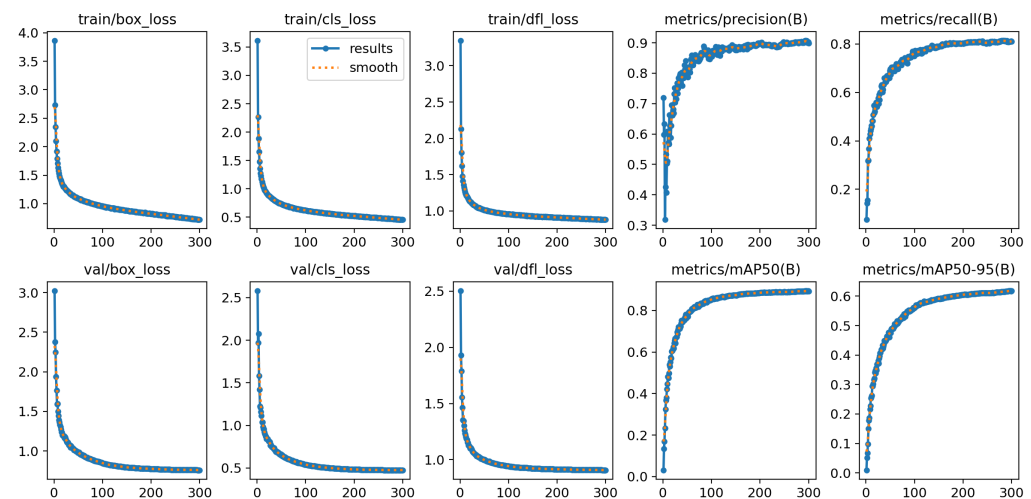


Figure 4. Training and validation curves of YOLOv8n-FEL on the KITTI dataset over 300 epochs. The top two rows display the training and validation loss curves (box_loss, cls_loss, df_l_loss); the bottom row shows the validation metrics (precision, recall, mAP@50, mAP@50-95).

Table 7. Matched comparison with recent YOLO-family baselines on the KITTI validation set.

Model	Params/M	GFLOPs	P	R	mAP@0.5	mAP@0.5:0.95	Inf./ms	Train/min
YOLOv8n baseline	3.006	8.1	0.888	0.818	0.891	0.629	6.1	136.0
YOLO11n	2.583	6.3	0.899	0.810	0.890	0.623	8.7	491.6
YOLO12n	2.510	5.8	0.887	0.803	0.873	0.610	11.5	1324.4
YOLOv8n-FEL	2.582	6.7	0.906	0.811	0.895	0.618	13.9	190.1

To further address the comparison with recent YOLO variants, YOLO11n and YOLO12n are trained from scratch on KITTI for 300 epochs using the same image size, batch size, SGD optimizer, random seed, and validation protocol. YOLO12n validation results are obtained using the FP32 best checkpoint. Training time is reported as a practical implementation-level efficiency indicator under the available training records; it may depend on hardware utilization, data loading, and software implementation, and therefore should be interpreted together with parameters, GFLOPs, and inference latency.

As shown in Table 7, YOLOv8n-FEL achieves the highest precision and mAP@0.5 among the compared compact models, whereas YOLOv8n retains the best recall, mAP@0.5:0.95,

and inference latency. YOLO12n has the lowest parameter count and GFLOPs, but obtains lower mAP@0.5 and mAP@0.5:0.95 under the present KITTI training protocol. For training cost, YOLOv8n-FEL requires 190.1 min for 300 epochs, which is longer than YOLOv8n (136.0 min) but substantially shorter than YOLO11n (491.6 min) and YOLO12n (1324.4 min). These results show that YOLOv8n-FEL provides a metric-dependent accuracy–complexity trade-off: it improves precision and AP@0.5-level performance with moderate model complexity and training cost, but it does not dominate all recent YOLO-family baselines across all metrics.

4.3. Cross-Dataset Direct Evaluation on DAIR-V2X

To assess whether the KITTI-trained models retain similar behavior under domain shift, they are directly evaluated on the vehicle-side subset of DAIR-V2X using the same trained weights and inference protocol. No DAIR-V2X samples are used for training, validation-based model selection, parameter tuning, or fine-tuning. Therefore, this experiment should be interpreted as a direct cross-dataset test rather than a dataset-specific training or adaptation experiment. The results are presented in Table 8.

Table 8. Direct evaluation results of KITTI-trained models on DAIR-V2X.

Model	P/%	R/%	mAP@0.5/%	mAP@0.5:0.95/%	Params/M
YOLOv8n	69.8	48.1	55.6	35.1	2.9
+SPPF-LSKA	71.0	48.5	56.6	36.7	3.1
+C2F-Faster	69.9	48.6	55.5	34.5	2.2
+C2F-Faster-EMA	70.5	49.5	56.3	36.0	2.2
YOLOv8n-FEL	72.6	49.9	57.1	36.9	2.4

As shown in Table 8, the absolute detection accuracy on DAIR-V2X is lower than that on KITTI, which is expected under direct cross-dataset evaluation. DAIR-V2X differs from KITTI in sensor placement, scene distribution, object scale, and domain characteristics. Therefore, the DAIR-V2X results should be interpreted as evidence under domain shift rather than as same-domain validation performance.

Despite the domain gap, YOLOv8n-FEL maintains a moderate improvement over the YOLOv8n baseline. Specifically, mAP@0.5 increases from 55.6% to 57.1%, mAP@0.5:0.95 increases from 35.1% to 36.9%, and precision increases from 69.8% to 72.6%, while the parameter count is reduced from 2.9M to 2.4M. The ranking of the tested variants is broadly similar to the KITTI results: SPPF-LSKA and C2F-Faster-EMA show positive trends, C2F-Faster alone mainly reduces parameters with limited accuracy gain, and the full YOLOv8n-FEL model obtains the best mAP among the tested variants.

The qualitative examples provide illustrative evidence of detection behavior in selected DAIR-V2X scenes. Since these observations are not supported by class-wise or scale-wise recall statistics, they should be interpreted as visual examples rather than quantitative evidence of improved cross-domain performance.

Figure 5 presents representative side-by-side detection examples on DAIR-V2X, with YOLOv8n shown on the left and YOLOv8n-FEL shown on the right. The examples illustrate selected detection behaviors under roadside-view domain shift, but they are qualitative illustrations only and should be interpreted together with the quantitative results in Table 8.

Overall, the DAIR-V2X experiment provides direct cross-dataset evidence under a different sensor and scene distribution, while broader validation is still needed before making strong generalization claims.



Figure 5. Qualitative detection comparison on the DAIR-V2X vehicle-side dataset. In each paired example, the left panel shows the YOLOv8n baseline detections and the right panel shows the YOLOv8n-FEL detections. KITTI-trained models are directly evaluated without DAIR-V2X fine-tuning.

4.4. Controlled Evaluation in Carla Simulation

To complement the real-world KITTI and DAIR-V2X evaluations, the models trained on KITTI are further tested in Carla simulation without additional training or fine-tuning. The evaluation is conducted using the Carla 0.9.15 simulation platform [21]. Carla provides controllable scenario variation and ground-truth annotations, making it suitable for examining detection behavior under distance variation and occlusion-rich simulated conditions. The quantitative comparison focuses on three representative scenarios: a near-range scenario, a mid-to-far-range scenario, and a complex scenario involving occlusion and varying scene conditions. Unless otherwise specified, the implementation environment and hardware platform follow the experimental setup described in Section 4.1. The results are evaluated using mAP@0.5, inference time, FPS, and scenario-wise detection rate (DR). DR is defined as the ratio of correctly detected objects to the total number of ground-truth objects in each scenario.

For fair comparison, all models are evaluated under the same Carla scenario definitions, annotation mapping, and inference settings. The Carla protocol is used as a controlled

simulation-based assessment rather than a substitute for real-world deployment validation. It is also not intended to replace the KITTI per-class AP analysis for Pedestrian, Cyclist, and Car; instead, it provides complementary scenario-level evidence for distance and scene-complexity effects.

4.4.1. Scene-Level Detection Performance

Table 9 presents the scene-level comparison between YOLOv8n and YOLOv8n-FEL in terms of detection accuracy (mAP@0.5), inference time (ms), and frame rate (FPS).

Table 9. Scene-level direct evaluation results in Carla simulation.

Scenario	YOLOv8n			YOLOv8n-FEL		
	mAP@0.5/%	ms	FPS	mAP@0.5/%	ms	FPS
Near-range	93.9	22.5	44.4	95.1	19.1	52.4
Mid-to-far-range	89.5	25.6	39.1	91.3	22.6	44.2
Complex	84.4	28.6	35.9	88.2	27.2	36.8
Average	89.3	25.6	39.1	91.5	23.0	43.5

To reduce reliance on qualitative Carla examples, Table 10 further reports the scenario-wise DR comparison. YOLOv8n-FEL improves DR from 95.5% to 96.5% in the near-range scenario, from 89.1% to 91.1% in the mid-to-far-range scenario, and from 81.5% to 88.3% in the complex scenario. The absolute DR gain is therefore +1.0, +2.0, and +6.8 percentage points, respectively, with the largest gain appearing in the complex scene.

Table 10. Scenario-wise detection rate (DR) comparison in Carla simulation.

Model	Near-Range	Mid-to-Far-Range	Complex Scene
YOLOv8n	95.5%	89.1%	81.5%
YOLOv8n-FEL	96.5%	91.1%	88.3%
Δ DR	+1.0	+2.0	+6.8

Several observations can be drawn from Tables 9 and 10. First, YOLOv8n-FEL obtains higher mAP@0.5 and DR than the YOLOv8n baseline across all three Carla scenarios. The largest absolute gains appear in the complex scenario, where mAP@0.5 increases from 84.4% to 88.2% (+3.8 percentage points) and DR increases from 81.5% to 88.3% (+6.8 percentage points). This trend is consistent with the design motivation of combining lightweight feature extraction with attention-enhanced feature refinement, although the Carla experiment does not isolate the individual contribution of each module. Second, the improvement over the baseline becomes more visible as the scene becomes more challenging. This result suggests that the full YOLOv8n-FEL architecture may provide additional benefit under occlusion and distance variation in controlled simulation. Nevertheless, because Carla is used only for direct testing of KITTI-trained models, these results should be interpreted as scenario-level supporting evidence rather than class-wise or deployment-level validation.

Figure 6 presents representative side-by-side detection renderings of YOLOv8n and YOLOv8n-FEL under the Carla evaluation scenarios. The examples illustrate selected detection behaviors under distance variation, occlusion, and cluttered simulated scenes. They should be interpreted together with the scene-level quantitative results in Tables 9 and 10, rather than as independent evidence of scenario-level improvement.

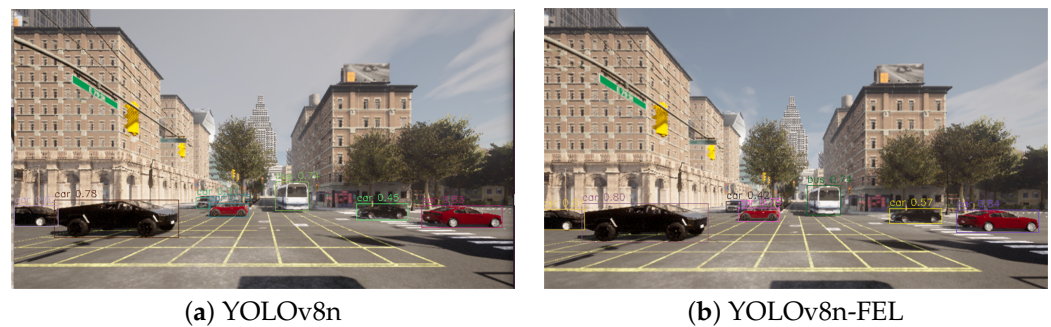


Figure 6. Qualitative detection comparison between YOLOv8n and YOLOv8n-FEL in Carla simulation. The same near-range, mid-to-far-range, and complex scenarios are shown for direct comparison. The examples are illustrative and should be interpreted together with the scene-level mAP and DR results in Tables 9 and 10.

4.4.2. Overall Computational Efficiency

Table 11 summarizes the overall efficiency comparison between YOLOv8n and YOLOv8n-FEL on the RTX 3080 platform at 640×640 input resolution. Different from the unweighted three-scenario average reported in Table 9, the mAP@0.5 values in Table 11 are computed as aggregate results over all evaluated Carla frames.

Table 11. Overall computational efficiency comparison in Carla simulation.

Model	mAP@0.5/%	Params/M	Inference/ms	FPS
YOLOv8n	91.1	2.9	25.1	39.8
YOLOv8n-FEL	92.5	2.4	23.3	42.9

In the overall Carla evaluation setting, YOLOv8n-FEL improves mAP@0.5 by 1.4 percentage points (91.1%→92.5%), while reducing inference latency by 7.2% (25.1 ms→23.3 ms) and increasing throughput by 7.8% (39.8 FPS→42.9 FPS). The parameter count is reduced from 3.006 M to 2.582 M. These results indicate that the PConv-based lightweight modification can provide practical latency benefits under the tested Carla GPU setting, although the KITTI latency results and future embedded-device validation are still needed for deployment-level efficiency assessment.

The mAP values in Table 11 are not expected to be identical to the arithmetic averages in Table 9. Table 9 reports scene-level behavior under different simulated conditions, whereas Table 11 reports aggregate frame-level evaluation and computational efficiency over the full Carla test set.

Overall, the Carla results provide controlled simulation-based support for evaluating YOLOv8n-FEL under distance variation, occlusion, and scene complexity. The added DR analysis provides a quantitative supplement to the visual renderings, but the Carla experiment remains a scenario-level direct test of KITTI-trained models rather than a Carla-specific class-wise benchmark.

In summary, the KITTI, DAIR-V2X, and Carla experiments indicate that YOLOv8n-FEL provides a lightweight accuracy–complexity trade-off relative to YOLOv8n. Its advantages are metric-dependent: the model reduces parameters and GFLOPs and improves precision and mAP@0.5 on KITTI, but it does not improve KITTI recall, mAP@0.5:0.95, or measured inference latency. These findings motivate the following discussion.

5. Discussion

5.1. Interpretation of the Accuracy–Efficiency Trade-Off

The KITTI ablation results show a clear reduction in model complexity after replacing the original C2f modules with C2F-Faster, followed by a partial increase in parameters when EMA and SPPF-LSKA are introduced. This pattern is consistent with the intended compress-then-enhance design: lightweight feature extraction first reduces redundant computation, and attention-based modules are then added for feature refinement and spatial-context modeling. However, the benefit is metric-dependent rather than uniform. YOLOv8n-FEL reduces parameters and GFLOPs and slightly improves precision and mAP@0.5, but it does not improve recall, mAP@0.5:0.95, or KITTI inference latency relative to YOLOv8n.

The DAIR-V2X and Carla evaluations provide complementary evidence beyond the KITTI validation split. DAIR-V2X reflects a real-world domain shift in sensor placement and scene distribution, whereas Carla provides controlled simulated scenarios involving distance variation, occlusion, and scene complexity. The added Carla DR results reduce reliance on selected visual examples by providing scenario-wise quantitative evidence. However, because the Carla experiment directly tests KITTI-trained models and is not a Carla-specific class-wise benchmark, these results are not sufficient to support deployment-level robustness claims. Such claims require additional validation on real-world adverse-condition data and embedded hardware.

5.2. Limitations and Future Work

Several limitations remain and define directions for future work.

Metric-dependent performance. The proposed model should be interpreted as a lightweight accuracy–complexity trade-off. Future work should investigate whether the loss in high-IoU localization quality can be reduced through improved feature fusion, loss design, or localization-aware lightweight modules.

Class-level challenges. The per-class KITTI results show that the improvement is most evident for Cyclist AP@0.5, while pedestrian detection and stricter AP@0.5:0.95 metrics remain challenging. This suggests that small and partially occluded traffic participants still require more targeted analysis. The Carla DR analysis is reported at the scenario level and should not be interpreted as a substitute for Pedestrian/Cyclist class-wise evaluation. Future work will include scale-wise and class-wise recall evaluation, especially for pedestrians, cyclists, and distant objects.

Runtime and embedded deployment. Although the model reduces theoretical complexity, the KITTI inference results show that attention modules can increase measured latency. All current experiments are conducted on desktop GPUs, and therefore do not fully reflect in-vehicle deployment constraints. Future work should evaluate TensorRT acceleration, quantization, memory bandwidth, energy consumption, and thermal behavior on embedded platforms such as NVIDIA Jetson Orin.

Cross-domain and simulation validation. DAIR-V2X and Carla provide useful complementary evaluation under domain shift and controlled simulated scenarios, but they are not substitutes for real-world adverse-condition validation. Future work should include real-world videos under rain, fog, low light, occlusion, and dense traffic conditions.

Extended benchmark evaluation. The present study includes matched comparisons with representative compact YOLO-family baselines, including YOLO11n and YOLO12n. Future work may further extend this evaluation to additional lightweight detectors under unified data splits, augmentation schedules, training settings, and inference protocols.

6. Conclusions

This paper presented YOLOv8n-FEL, a lightweight visual detector based on a progressive compress-then-enhance architecture for traffic-object detection. The model replaces the original C2f modules in YOLOv8n with C2F-Faster-EMA for lightweight feature extraction and attention-based refinement, and introduces SPPF-LSKA at the SPPF stage for spatial-context modeling. Rather than proposing new basic modules, this work focuses on the architecture-level placement of lightweight compression and selective feature enhancement within a compact YOLOv8n-based detector.

Experiments on KITTI show that YOLOv8n-FEL reduces parameters from 3.006 M to 2.582 M and GFLOPs from 8.1 to 6.7, corresponding to reductions of approximately 14.1% and 17.3%, respectively. Under this reduced-complexity setting, precision increases from 0.888 to 0.906 and mAP@0.5 increases from 0.891 to 0.895. The per-class results further show that Cyclist AP@0.5 improves from 0.899 to 0.926. However, the gains are metric-dependent: recall, mAP@0.5:0.95, and measured KITTI inference latency do not outperform the YOLOv8n baseline. The matched YOLO11n and YOLO12n comparisons further indicate that YOLOv8n-FEL provides higher precision and mAP@0.5 than the compared recent compact baselines, while YOLOv8n retains advantages in inference latency and mAP@0.5:0.95, and YOLO12n has lower parameters and GFLOPs.

Direct evaluation on DAIR-V2X and controlled Carla simulation provides additional evidence under domain shift and scenario variation. Overall, YOLOv8n-FEL is best understood as a compact detector that improves selected AP@0.5-level accuracy metrics while reducing model complexity, rather than as an across-metric superior detector. Future work will focus on improving high-IOU localization quality, reducing attention-related latency, strengthening class-wise analysis for pedestrians and distant objects, and validating embedded deployment under real-world adverse driving conditions.

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

Funding: This research was funded by the Science and Technology Department of Sichuan Province, grant number 2024NSFSC0015; the Yibin Industry–Education Integration Project, grant number ZZ20240045; and the Ministry of Education Industry–University Cooperative Education Program, grant number 250700595164008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The KITTI dataset is publicly available at <https://www.cvlibs.net/datasets/kitti/> (accessed on 13 July 2026). The DAIR-V2X dataset is publicly available at <https://github.com/AIR-THU/DAIR-V2X> (accessed on 13 July 2026). The Carla simulation platform is available at <https://carla.org/> (accessed on 13 July 2026).

Acknowledgments: The authors thank the editors and anonymous reviewers for their constructive comments during the review and production process.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BiFPN	Bidirectional Feature Pyramid Network
CBAM	Convolutional Block Attention Module
CSP	Cross Stage Partial
DWConv	Depth-wise Convolution
EMA	Efficient Multi-Scale Attention
GFLOPs	Giga Floating Point Operations
IoU	Intersection over Union
LSKA	Large Separable Kernel Attention
mAP	mean Average Precision
PANet	Path Aggregation Network
PConv	Partial Convolution
RT-DETR	Real-Time Detection Transformer
SENet	Squeeze-and-Excitation Network
SOD	Small Object Detection
SPPF	Spatial Pyramid Pooling Fast
YOLO	You Only Look Once

References

1. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [\[CrossRef\]](#)
2. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [\[CrossRef\]](#)
3. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [\[CrossRef\]](#)
4. Jocher, G.; Chaurasia, A.; Qiu, J. *YOLO by Ultralytics*, version 8.0.0; Ultralytics: Frederick, MD, USA, 2023. Available online: <https://github.com/ultralytics/ultralytics> (accessed on 13 July 2026).
5. Chen, J.; Kao, S.-H.; He, H.; Zhuo, W.; Wen, S.; Lee, C.-H.; Chan, S.-H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 12021–12031. [\[CrossRef\]](#)
6. Aldubaikhi, A.; Patel, S. Advancements in small-object detection (2023–2025): Approaches, datasets, benchmarks, applications, and practical guidance. *Appl. Sci.* **2025**, *15*, 11882. [\[CrossRef\]](#)
7. Yi, X.; Chen, H.; Wu, P.; Wang, G.; Mo, L.; Wu, B.; Yi, Y.; Fu, X.; Qian, P. Light-FC-YOLO: A lightweight method for flower counting based on enhanced feature fusion with a new efficient detection head. *Agronomy* **2024**, *14*, 1285. [\[CrossRef\]](#)
8. Khalili, B.; Smyth, A.W. SOD-YOLOv8: Enhancing YOLOv8 for small object detection in aerial imagery and traffic scenes. *Sensors* **2024**, *24*, 6209. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Wang, H.; Liu, C.; Cai, Y.; Chen, L.; Li, Y. YOLOv8-QSD: An improved small object detection algorithm for autonomous vehicles based on YOLOv8. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 2513916. [\[CrossRef\]](#)
10. Zhao, Y.; Sun, F.; Wu, X. FEB-YOLOv8: A multi-scale lightweight detection model for underwater object detection. *PLoS ONE* **2024**, *19*, e0311173. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Tao, H.; Paul, A.; Wu, Z. Infrared image detection and recognition of substation electrical equipment based on improved YOLOv8. *Appl. Sci.* **2025**, *15*, 328. [\[CrossRef\]](#)
12. Zhu, J.; Hu, T.; Zheng, L.; Zhou, N.; Ge, H.; Hong, Z. YOLOv8-C2f-Faster-EMA: An Improved Underwater Trash Detection Model Based on YOLOv8. *Sensors* **2024**, *24*, 2483. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Zhang, Y.; Zhang, J.; Du, F.; Kang, W.; Wang, C.; Li, G. Real-time lightweight vehicle object detection via layer-adaptive model pruning. *Electronics* **2025**, *14*, 4149. [\[CrossRef\]](#)
14. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861. [\[CrossRef\]](#)
15. Ouyang, D.; He, S.; Zhang, G.; Luo, M.; Guo, H.; Zhan, J.; Huang, Z. Efficient Multi-Scale Attention Module with Cross-Spatial Learning. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [\[CrossRef\]](#)
16. Lau, K.W.; Po, L.-M.; Rehman, Y.A.U. Large Separable Kernel Attention: Rethinking the Large Kernel Attention Design in CNN. *Expert Syst. Appl.* **2024**, *236*, 121352. [\[CrossRef\]](#)

17. Jiang, X.; Zhuang, X.; Chen, J.; Zhang, J.; Zhang, Y. YOLOv8-MU: An improved YOLOv8 underwater detector based on a large kernel block and a multi-branch reparameterization module. *Sensors* **2024**, *24*, 2905. [[CrossRef](#)] [[PubMed](#)]
18. Sabaghian, M.; Keyvanrad, M.A.; Moghadami, S.M. A novel compression framework for YOLOv8: Achieving real-time aerial object detection on edge devices via structured pruning and channel-wise distillation. *arXiv* **2025**, arXiv:2509.12918.
19. Yu, H.; Luo, Q.; Peng, W.; Zheng, L.; Ju, J.; Zhuo, H. PKD-YOLOv8: A collaborative pruning and knowledge distillation framework for lightweight rapeseed pest detection. *Sensors* **2025**, *25*, 5004. [[CrossRef](#)] [[PubMed](#)]
20. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-time Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 16965–16974. [[CrossRef](#)]
21. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning (CoRL), Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.

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