International Journal of Engineering and Technology Innovation, vol. 16, no. 4, 2026, pp. 491-511

Efficient YOLOv11-Based Oil Palm Fresh Fruit Bunch Detection Using Structured Detection-Path Pruning and INT8 Quantization

Baizul Zaman[1], Indrabayu[2,*], Ingrid Nurtanio[2]

[1]Department of Electrical Engineering, Hasanuddin University, Makassar, Indonesia

[2]Department of Informatics, Hasanuddin University, Makassar, Indonesia

Received 27 April 2026; received in revised form 23 June 2026; accepted 25 June 2026

DOI: https://doi.org/10.46604/ijeti.2026.16411

Abstract

This study proposes an efficient you only look once (YOLO)v11s-based framework for real-time oil palm fresh fruit bunches (FFB) detection, addressing the high computational cost that typically limits YOLO-based detectors on edge devices. The framework integrates structured detection-path pruning (SDPP) and 8-bit Integer (INT8) post-training quantization. SDPP analyzes the object-scale distribution of FFBs in UAV imagery to identify the detection paths that contribute most effectively to the target object characteristics. Based on this analysis, redundant detection branches with limited contribution to detection performance are removed, reducing the computational burden while preserving accuracy. The pruned model is subsequently quantized to INT8 precision to further improve inference efficiency on resource-constrained devices. Experimental results show that SDPP reduces computational complexity by 53.5% while maintaining an mAP@0.5 of 0.987. With INT8 quantization, the optimized model achieves 30.42 FPS on the NVIDIA Jetson Orin Nano, demonstrating its potential for real-time FFB detection on edge devices.

Keywords: YOLOv11, structured pruning, INT8 quantization, oil palm FFB detection, UAV imagery

1. Introduction

The rapid development of artificial intelligence (AI) technology drives the adoption of vision-based object detection systems in the domain of precision agriculture [1]. Among the various aerial platforms available, unmanned aerial vehicles (UAVs) have gained growing prominence in monitoring large-scale plantations, primarily due to their ability to acquire high-resolution imagery directly under actual field conditions [2]. Within the context of oil palm cultivation, the precise identification of fresh fruit bunches (FFB) plays a critical role in supporting yield forecasting, optimizing harvest scheduling, and enhancing overall agricultural productivity [3]. Object detectors grounded in deep learning, most notably those belonging to the you only look once (YOLO) family, exhibit remarkable performance attributed to their end-to-end architectural design and efficient inference processing [4].

More recent iterations, including YOLOv11, advance detection performance through improved feature representation and the incorporation of multi-scale detection mechanisms [5]. Nevertheless, such architectural enhancements inevitably introduce greater computational demands, which poses considerable challenges for real-time deployment on platforms with limited processing resources [6]. To address this limitation, various optimization techniques, particularly structured pruning and low-precision quantization, are explored to reduce model complexity while improving inference performance [7]. However, most existing studies focus on earlier YOLO variants and apply these techniques independently rather than in an integrated manner. In addition, limited attention is paid to analyzing the trade-offs between detection accuracy, architectural simplification, and runtime performance within a unified framework.

* Corresponding author. E-mail address: indrabayu@unhas.ac.id

In UAV-based FFB detection, objects generally appear at medium-to large- scales, suggesting that certain fine-scale detection paths may contribute minimally to the detection performance while incurring additional computational costs. However, this structural redundancy remains under investigated within the YOLOv11 architecture. This gap highlights the need for more targeted and domain-aware optimization strategies.

To address this gap, this study proposes an optimization framework for YOLOv11 by integrating structured detection-path pruning (SDPP) and INT8 quantization. The proposed approach aims to reduce computational complexity while preserving detection accuracy for UAV-based FFB detection. Structured pruning is employed to remove redundant detection branches, whereas 8-bit Integer (INT8) quantization accelerates inference and reduces latency on edge devices. All experiments utilize UAV imagery collected from oil palm plantations. Specifically, this research aims to:

(1) Propose a domain-guided structured pruning strategy that removes redundant multi-scale detection paths in YOLOv11.

(2) Achieve significant reductions in architectural complexity and computational cost while maintaining detection accuracy.

(3) Integration of INT8 quantization using TensorRT to improve inference efficiency on resource-constrained devices.

(4) Comprehensive evaluations of both workstation GPU and NVIDIA Jetson-class hardware are provided to validate deployment feasibility.

This paper is organized as follows. Section 2 reviews related work on UAV-based FFB detection, YOLO optimization, pruning, and quantization. Section 3 presents the proposed methodology, including dataset preparation, the baseline YOLOv11s architecture, SDPP, INT8 quantization, and the experimental setup. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Related Works

The use of unmanned aerial vehicles in precision agriculture has attracted increasing research interest, particularly for tasks such as oil palm tree identification and FFB detection. Early studies employed deep learning–based detectors, including YOLOv3 and YOLOv4, demonstrating the feasibility of UAV imagery for plantation monitoring despite challenges such as illumination variation, occlusion, and complex backgrounds [8-11]. Subsequent advancements in the YOLO family, notably YOLOv5 and YOLOv7, improve detection accuracy and inference speed, enabling lightweight variants to achieve real-time performance on embedded devices such as the NVIDIA Jetson Nano [12-13]. In addition, comparative studies of YOLO-based detectors in other real-time vision domains show that newer YOLO variants provide strong detection performance under practical deployment scenarios [14].

Various model optimization approaches have been proposed to support deployment in resource-constrained environments, including channel pruning, network compression, and lightweight backbone design. Prior research shows that structured pruning and low-precision quantization can substantially reduce the model size and computational overhead while preserving accuracy [15]. These techniques have been successfully applied to earlier YOLO versions, including YOLOv3, YOLOv5, and YOLOv8, achieving notable inference speed improvements with marginal reductions in mean average precision (mAP) [16-17]. However, these methods predominantly operate at the channel or weight level without modifying the architecture of the multi-scale detection head.

AI research also explored custom, lightweight architectures tailored for embedded deployment and domain-specific training strategies, such as specialized augmentation and synthetic data generation. Despite the progress achieved in pruning and quantization, limited attention has been given to domain-guided detection-path pruning in YOLO models, particularly within the YOLOv11 framework. Most existing pruning studies focus on filter-, channel-, or weight-level compression, whereas the structural contribution of multi-scale detection heads remains less explored. In UAV-based FFB detection,

particularly for datasets dominated by large-scale instances, the removal of redundant small- and medium-scale detection paths has not been systematically examined. Moreover, the combined use of detection-path pruning and post-training INT8 quantization has not been comprehensively evaluated. In particular, their effects on detection accuracy, computational efficiency, and real-time deployability on embedded platforms such as the NVIDIA Jetson Orin Nano. These gaps motivate the development of an integrated architectural simplification and quantization strategy for efficient real-time YOLOv11-based FFB detection in precision agriculture scenarios.

To clarify the novelty and research gap of the proposed study, Table 1 summarizes representative studies related to oil palm detection, FFB ripeness detection, UAV-based object detection, YOLO model evolution, pruning, TensorRT quantization, and real-time YOLO-based deployment.

Table 1 Comparison of previous studies and research gaps

Study	Domain	Method	Main result	Research gap
S. Siang et al. [3]	Oil palm tree detection	Enhanced RetinaNet	F1-score: 0.947 (single)/0.902 (dual)	Focuses on trees; lacks FFB detection, YOLOv11 optimization, and edge deployment.
Lai et al. [8]	FFB ripeness detection	Systematic review	Identified deep learning as feasible	Review study; lacks YOLOv11-based lightweight model and pruning/quantization evaluation.
Junior and Suharjito [9]	Video-based FFB detection	YOLOv4 variants	90.56% mAP (single-class) and 70.21% mAP (multi-class)	Focused on YOLOv4; lacks YOLOv11, detection-path pruning, and edge deployment.
Wibowo et al. [10]	Oil palm tree detection	YOLOv3, v4, v5m	F1-scores: 97.28%, 97.74%, and 94.94% for YOLOv3, YOLOv4, and YOLOv5m, respectively	Focused on tree inventory; lacks FFB detection and model pruning/quantization.
Ahn et al. [15]	General object detection	SAFP-YOLO	~5x speedup on TX-2	Uses filter-level pruning; lacks UAV-based FFB detection and YOLOv11 architecture.
Zhou et al. [16]	TensorRT quantization	TensorRT workflows	Improved inference efficiency	Focused on general quantization ; lacks task-specific detection-path.
Liu et al. [17]	UAV remote sensing	FasterNet-16 backbone	Lightweight real-time detection	Focused on transportation; lacks YOLOv11, SDPP, or INT8 integration.
Solak et al. [14]	Urban surveillance	YOLOv4, v5, v8	97%–99% accuracy	Different application domain; lacks agricultural-specific FFB detection and edge optimization.
Proposed study	UAV-based FFB detection	YOLOv11s+SDPP+ INT8	0.987 mAP / 30.42 frames per second (FPS) (Jetson Orin Nano)	Integrates domain-guided detection-path pruning and INT8 for edge-based FFB detection.

As summarized in Table 1, previous studies address oil palm tree detection, FFB ripeness assessment, UAV-based object detection, YOLO model evolution, pruning, and TensorRT quantization. However, existing oil palm studies mainly focus on tree detection, ripeness classification, or earlier YOLO variants, while lightweight UAV detectors are generally designed for public UAV datasets rather than oil palm FFB detection. Existing pruning studies commonly operate at the filter, channel, or weight level, whereas TensorRT quantization studies mainly focus on inference acceleration without considering task-specific architectural simplification. Consequently, a clear research gap remains in developing a domain-guided YOLOv11 optimization framework that integrates SDPP and INT8 quantization for real-time UAV-based FFB detection on resource-constrained edge devices.

3. Methodology

An optimized YOLOv11-based framework is developed for FFB detection by using UAV imagery. The proposed approach integrates SDPP and INT8 quantization to reduce computational complexity while preserving detection accuracy. The detailed components of the optimization framework are described in the following sections.

3.1. System overview

The proposed system employs the YOLOv11s model as the baseline architecture for detecting FFB in UAV imagery captured from oil palm plantations. This study assesses the effects of domain-specific architectural pruning and low-precision quantization on the model complexity, detection accuracy, and inferential efficiency within computationally constrained deployment environments. The overall optimization pipeline is shown in Fig. 1.

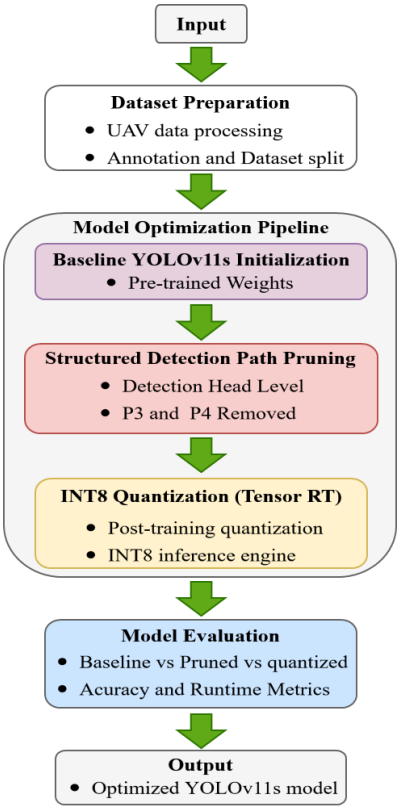


Fig. 1 Overall workflow of the proposed optimization pipeline for YOLOv11s

Fig. 1 presents the proposed workflow, which encompasses five sequential stages: Dataset Preparation, Baseline YOLOv11s Initialization, SDPP, INT8 Quantization, and Model Evaluation. First, the UAV-acquired videos undergo frame extraction, annotation, and dataset partitioning to produce the training, validation, and testing sets. Second, the baseline model is initialized with COCO-pretrained weights and is simplified using the SDPP, with P3 and P4 detection branches removed to reduce redundant computations. Consequently, fine-scale detection paths provide limited benefits while increasing the computational overhead. Third, the pruned model is subsequently optimized using TensorRT-based INT8 quantization to improve the runtime efficiency. Finally, the optimized model is evaluated on workstation GPUs and NVIDIA Jetson-class edge devices to assess the detection accuracy and real-time inference performance.

3.2. Dataset 9reparation

All image data employed in this research are gathered directly from smallholder oil palm plantation areas located in Patila Village, South Sulawesi, Indonesia. The aerial imagery is captured utilizing a DJI Mavic Air drone fitted with a high-resolution optical sensor, enabling the documentation of FFB under authentic field conditions. A comprehensive illustration of the data collection procedure is provided in Fig. 2.

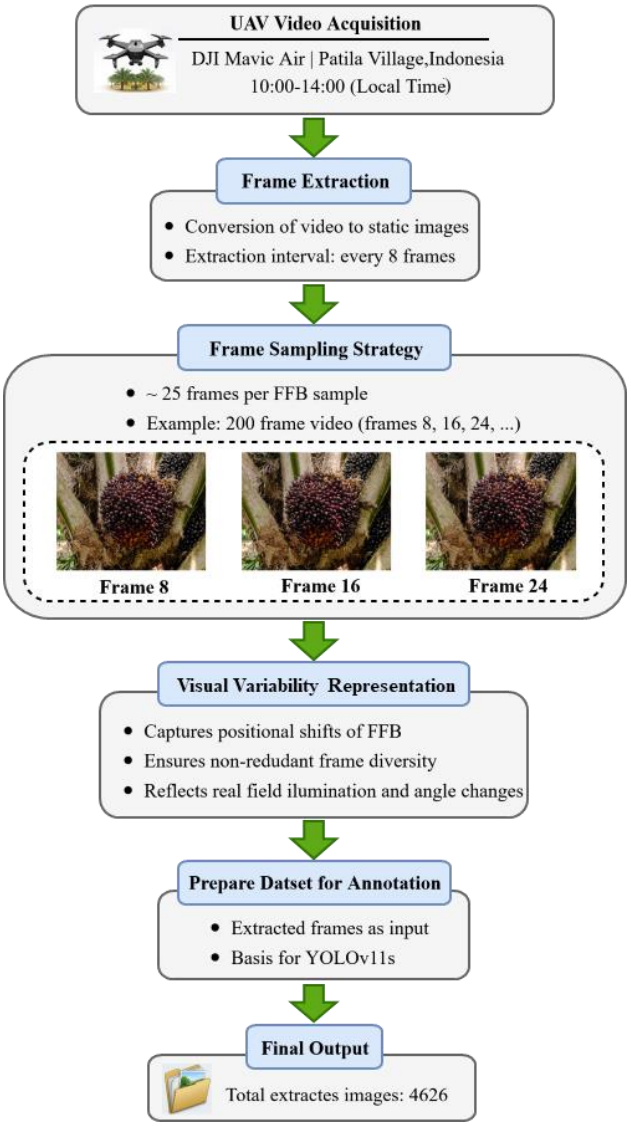


Fig. 2 Workflow of UAV-based dataset acquisition and frame extraction process

As shown in Fig. 2, after data acquisition, the recorded UAV videos are converted into static images using a frame extraction process. The frames are sampled at fixed intervals of eight frames to reduce temporal redundancy while preserving variations in the FFB appearance and camera perspective. This sampling strategy captures positional shifts and viewpoint variations commonly observed in UAV imagery. This process produces 4626 images, which are used to construct the dataset. Ground-truth annotations are then generated to prepare the dataset for model training. Each oil palm FFB in the extracted frames is manually labeled using bounding boxes on the Roboflow platform to ensure precise object localization. The annotation workflow is shown in Fig. 3.



Fig. 3 Illustration of bounding box annotation for oil palm FFB

As shown in Fig. 3, following the bounding box annotation process, the collected data are organized into three distinct partitions: training, validation, and testing. This segmentation establishes a coherent model development pipeline while enabling an unbiased evaluation of the model's ability to generalize beyond its training data. The training partition refines the internal parameters of the model iteratively. The validation partition serves as a feedback mechanism to detect and mitigate overfitting. The testing partition is kept entirely separate as the benchmark for final performance evaluation on previously unseen data. Using a random allocation approach, the data are distributed across the three partitions at ratios of 70%, 20%, and 10% for training, validation, and testing, respectively, ensuring proportional representation of FFB instances in each partition. A complete overview of the dataset distribution is presented in Table 2.

Table 2 Dataset distribution

Subset	Proportion	Number of images
Training	70%	3238
Validation	20%	925
Testing	10%	463
Total	100%	4626

In addition to the dataset distribution, the UAV imagery used in this study is collected under real plantation conditions. Based on the recorded flight interface during data acquisition, the UAV altitude is approximately 6 m, and the videos are recorded at 1080p and 30 FPS. The camera captures the oil palm fresh fruit bunches from an oblique viewpoint rather than a strictly nadir perspective. The dataset contains several visual challenges commonly encountered in UAV-based oil palm imagery, including natural illumination variation, sun glare, shadow effects, palm frond occlusion, complex foliage backgrounds, and viewpoint changes caused by UAV movement.

Higher UAV altitude reduces the apparent size and visual details of FFB objects, making small clusters or partially occluded bunches more difficult to detect. These conditions provide a realistic evaluation setting for FFB detection in plantation environments. However, detailed flight metadata, including the exact gimbal pitch angle and complete weather records for all video sequences, are not consistently logged during data acquisition. Therefore, their quantitative effects on detection accuracy are acknowledged as a limitation of this study.

To further characterize the dataset, an object-scale distribution analysis is conducted using all annotated FFB instances. Based on the bounding-box dimensions, annotated FFB instances are categorized into small-, medium-, and large-scale groups to characterize the object-scale distribution within the UAV imagery. Table 3 summarizes the distribution of annotated FFB instances across object scales. The analysis reveals that 6,738 instances (95.91%) belong to the large-scale category, while only 227 instances (3.23%) and 60 instances (0.85%) belong to the medium- and small-scale categories, respectively. These results indicate that the dataset is strongly dominated by large-scale FFB instances, with only 4.08% of the annotated instances belonging to the small- and medium-scale categories combined.

Table 3 Distribution of annotated FFB instances across object scales

Object scale	Annotated FFB instances	Percentage (%)
Small-scale	60	0.85
Medium-scale	227	3.23
Large-scale	6,738	95.91
Total	7,025	100.00

As shown in Table 3, the dataset is strongly dominated by large-scale FFB instances, which account for 95.91% of all annotated objects. In contrast, only 4.08% of the instances belong to the small- and medium-scale categories combined. This distribution indicates that most FFB targets occupy relatively large image regions in UAV imagery, providing quantitative support for the proposed Structured Detection-Path Pruning strategy.

3.3. Baseline YOLOv11 architecture

The YOLOv11s architecture is adopted as the baseline model for the proposed optimization framework. A simplified structural overview of the baseline network is shown in Fig. 4, which is reconstructed based on the official YOLOv11 architectural specifications [18]. The model consists of three main components: a backbone for hierarchical feature extraction, a neck for multi-scale feature aggregation, and a detection head for object prediction. In the baseline architecture, the detection head operates at three scales, namely P3, P4, and P5. These branches are responsible for detecting small-, medium-, and large-scale objects, respectively. Specifically, P3 operates at a higher spatial resolution for small objects, P4 handles medium-scale objects, and P5 captures larger objects with stronger semantic features. Fig. 4 provides the architectural basis for analyzing the contribution of each detection path and for introducing the proposed SDPP strategy in the following section.

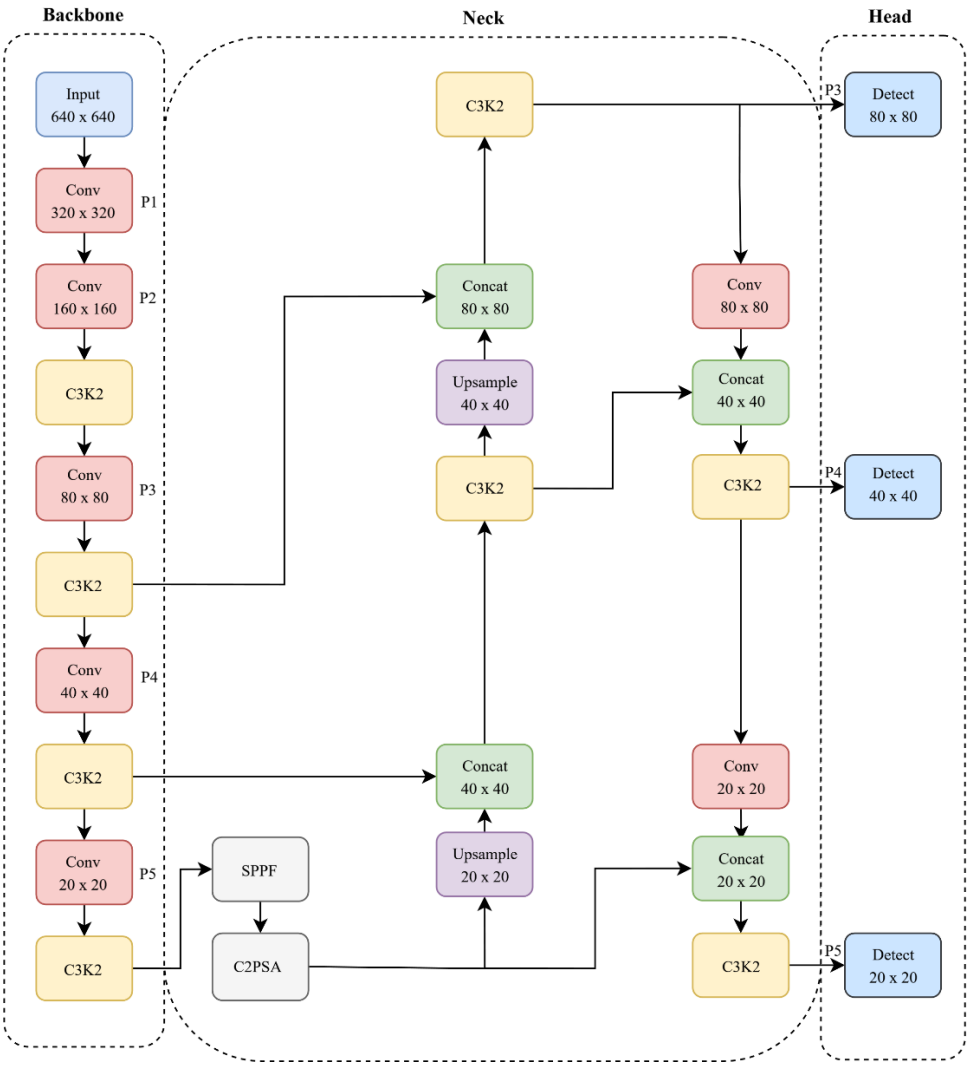


Fig. 4 Baseline architecture of YOLOv11s

As illustrated in Fig. 4, YOLOv11s adopts a three-path detection head operating at P3 (80×80), P4 (40×40), and P5 (20×20) scales, each responsible for detecting objects of different sizes. This multiscale design provides high flexibility but also increases the computational cost, particularly for smaller-scale branches. However, because UAV-captured FFB typically exhibit medium-to-large object sizes, the relevance of each detection path varies. Specifically, branches P3 and P4 contribute limited benefits relative to their computational costs.

Supporting this observation, the object-scale distribution analysis presented in Table 3 shows that 6,738 out of 7,025 annotated FFB instances (95.91%) belong to the large-scale category. In contrast, only 227 instances (3.23%) and 60 instances (0.85%) belong to the medium- and small-scale categories, respectively. These results indicate that the dataset is strongly

dominated by large-scale FFB instances, suggesting that the P5 detection branch captures the majority of target objects, whereas the P3 and P4 branches contribute less to the target detection task. This observation supports the hypothesis that fine-scale detection branches may contribute minimally to detection accuracy while introducing a significant computational overhead.

3.4. SDPP

The SDPP is applied to reduce the architectural complexity of YOLOv11s by selectively removing redundant multi-scale detection branches. The pruning workflow is shown in Fig. 5. YOLOv11s originally employs three detection paths: P3, P4, and P5. For UAV-based oil palm FFB detection, objects predominantly fall within the medium to large scale range, making fine-scale branches (i.e., P3 and P4) less relevant to the target domain. Based on this observation, the SDPP evaluated the contribution of each detection path to the FFB detection task and identified P3 and P4 as nonessential components. These branches, along with their associated upsampling operations, concatenation modules, and detection heads, are removed from the architecture. Only the P5 branch—responsible for large-scale predictions—retained.

The model is then reconstructed using the remaining detection path, resulting in a simplified single-branch detection architecture. This structured pruning approach eliminates unnecessary computations while preserving task-relevant detection performance. The reduced architectural complexity significantly lowers the overall computational workload, as reflected by the reduction in floating-point operations (FLOPs), providing an efficient foundation for the subsequent INT8 quantization stage.

As shown in Fig. 5, the SDPP operates at the detection path level of the YOLOv11s architecture. In the baseline model, the neck and head modules contain three prediction branches—P3, P4, and P5—which are responsible for small-, medium-, and large-scale object detection. Although this multiscale design benefits general object detection tasks, UAV imagery of oil palm plantations typically contains FFB as medium- to large-scale objects. Based on this observation, the fine-scale detection paths (P3 and P4) are identified as redundant for the target task and are removed, along with their associated upsampling layers, concatenation nodes, and detection heads. The model is then reconstructed using only the P5 branch, resulting in a simplified single-scale detection architecture. This pruning strategy reduces computational complexity while preserving task-relevant detection capability. The resulting pruned YOLOv11s architecture is shown in Fig. 6.

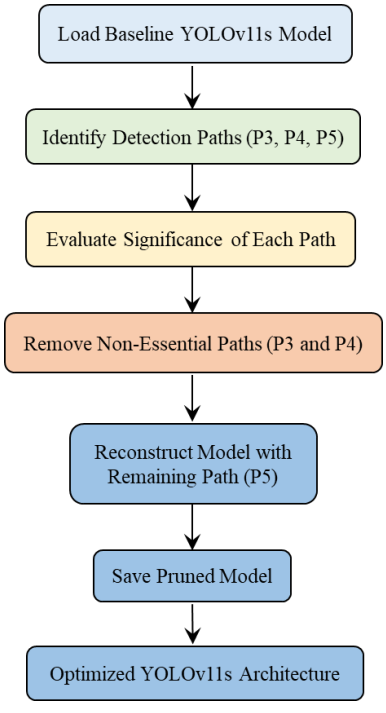


Fig. 5 Workflow of the proposed YOLOv11s pruning method

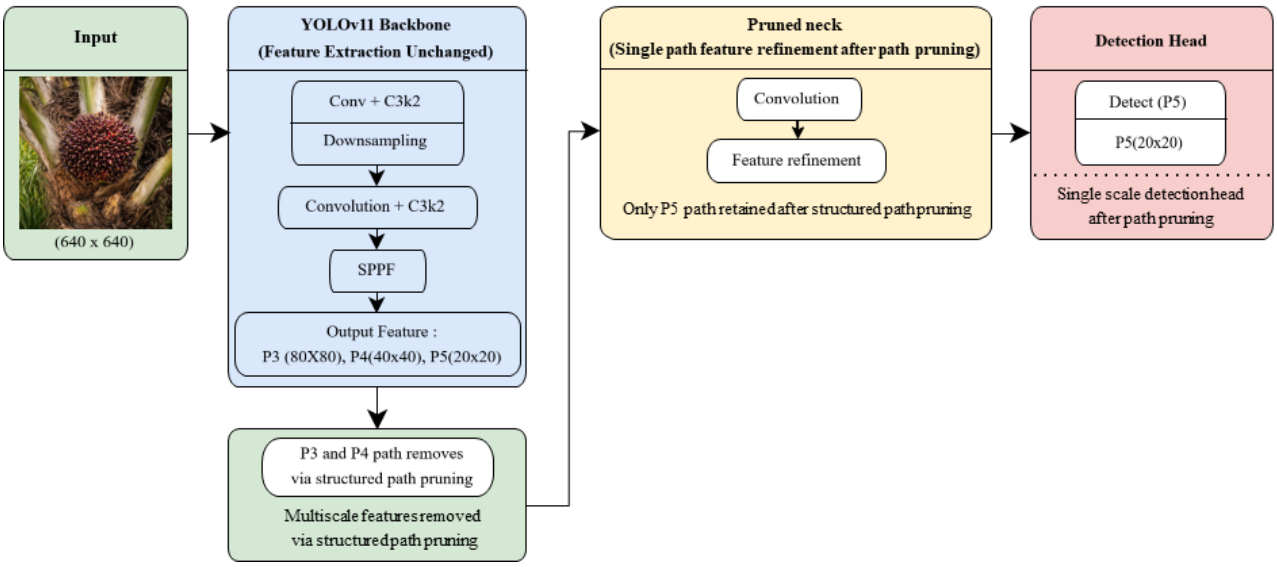


Fig. 6 Architecture of the proposed structured detection-path pruning YOLOv11s model

The pruned architecture shown in Fig. 6 results in a more computationally efficient detection network. By simplifying the original multi-scale detection structure into a streamlined single-branch architecture, the model achieves a substantial reduction in FLOPs and overall computational workload. Although the parameter count slightly increases due to architectural reconfiguration, the removal of redundant detection paths reduces computational overhead within the neck and head modules while maintaining detection performance for the target application. The resulting efficient model improved inference performance, providing a suitable foundation for subsequent INT8 quantization.

3.5. INT8 quantization

After applying the SDPP, the simplified YOLOv11s model is further optimized through post-training INT8 quantization using NVIDIA TensorRT. This optimization stage accelerates inference by converting floating-point operations into 8-bit integer representations, thereby reducing computational cost, memory bandwidth consumption, and inference latency on resource-constrained hardware. The overall quantization workflow is illustrated in Fig. 7.

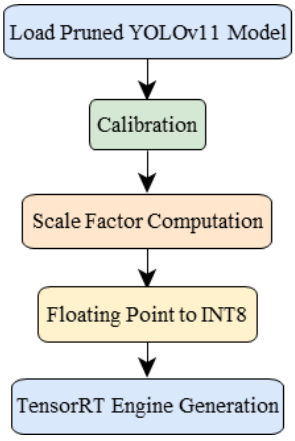


Fig. 7 Workflow of the proposed INT8 quantization pipeline for the pruned YOLOv11s model

Fig. 7 illustrates the INT8 quantization pipeline applied to the pruned YOLOv11s model to further enhance computational efficiency. The process begins with a calibration phase, in which a representative subset of the dataset is used to estimate the dynamic activation range for each network layer. The minimum and maximum activation values are obtained as follows:

$$x_{\min} = \min(x), x_{\max}(x) \tag{1}$$

where x denotes the floating-point activation values in a network layer, while x_{min} and x_{max} denote the minimum and maximum activation values obtained from the calibration dataset, respectively. This step ensures that the quantization process is aware of the smallest and largest values that the layer activations can produce.

Once this dynamic range is known, it becomes necessary to map it into the limited space available in the INT8 representation. Since 8-bit integers can represent ($2^8 = 256$) discrete levels, the effective step size between each quantized level is obtained by dividing the total floating-point range by (2^b - 1), where (b = 8). This leads to the scale factor:

$$s = \frac{x_{\max} - x_{\min}}{2^b - 1} \tag{2}$$

where s denotes the quantization scale factor and b denotes the quantization bit width.

The scale factor (s) defines the mapping between floating-point values and their corresponding INT8 representation. Given this scaling factor, a floating-point activation (x) is transformed into the integer domain by scaling, rounding, and shifting with a zero-point offset (z), which aligns the integer range with the original data distribution. The quantized value is obtained as follows:

$$q = round\left(\frac{x}{s}\right) + z \tag{3}$$

where q denotes the quantized integer value, z denotes the zero-point offset, and *round* denotes rounding to the nearest integer.

Eqs. (1-3) describe the core quantization mechanism, where the activation range is first identified, discretized into uniform quantization steps, and subsequently mapped into the INT8 domain. This process allows the model to preserve its numerical characteristics while operating with reduced precision. Following calibration and quantization, TensorRT converts the pruned YOLOv11s model into an optimized inference engine (*.engine* file) by replacing floating point operations with integer arithmetic. This conversion significantly improves the inference speed and reduces the latency while maintaining detection performance comparable to that of the original floating-point model. The resulting quantized model serves as the basis for the for runtime performance evaluation discussed in the subsequent section.

3.6. Experimental setup

All experiments are conducted to evaluate the performance of the proposed YOLOv11s optimization pipeline for FFB detection. The hardware platform consists of a development laptop equipped with an AMD Ryzen 7 4800H processor, 16 GB RAM, and an NVIDIA GTX 1650 Ti GPU with 4 GB VRAM running Windows 11. This system is used for model optimization and inference evaluation. The UAV platform is solely used for data acquisition and is not involved in onboard inference or real-time processing. On the software side, Python 3.10 and the YOLOv11s framework are used.

Dataset annotation and augmentation are performed using the Roboflow platform. Model training is conducted in a Google Colab environment to utilize cloud-based GPU resources. For inference optimization, post-training INT8 quantization is implemented using NVIDIA TensorRT with CUDA Toolkit 12.3, enabling GPU acceleration. Additional libraries, including OpenCV, were used for image and video processing during the evaluation. To ensure a fair comparison, the baseline YOLOv11s, SDPP-pruned, and INT8-quantized models are evaluated using the same dataset splits and evaluation protocols. This experimental setup enables a systematic analysis of the trade-offs among detection accuracy, computational complexity, and inference performance.

After the pruning process, the SDPP-YOLOv11 model is fine-tuned using the same training dataset to adapt the remaining detection branch and recover potential performance degradation caused by pruning. The model is trained for a maximum of 50 epochs with an early stopping patience of 25 epochs. Training is performed using a batch size of 8 and learning rate of

0.0005 with cosine learning rate scheduling. Data augmentation techniques—including HSV augmentation, rotation, translation, scaling, mosaic augmentation, horizontal flipping, shear transformation, and perspective distortion—are applied during training. The detailed hyperparameter settings are summarized in Table 4.

Table 4 Training hyperparameters used in this study

Parameter	Value
Epoch	50
Early stopping patience	25
Batch size	8
Learning rate (lr0)	0.0005
Learning rate final (lrf)	0.1
Scheduler	Cosine Annealing (cos_lr)
Confidence threshold	0.35
IoU threshold	0.5
Cache dataset	Yes
HSV-H, S, V	0.01, 0.5, 0.3
Rotation	10
Translation & scale	0.05, 0.3
Mosaic	0.3
Mixup	0.1
Flip horizontal	0.3
Shear and perspective	3, 0.001
Close Mosaic	15

The selected hyperparameter configuration is designed to balance detection performance and training stability. Cosine learning rate scheduling is employed to facilitate smooth convergence, while data augmentation techniques are applied to improve model robustness against variations in object appearance, viewing angle, and illumination conditions. Early stopping is adopted to mitigate overfitting and ensure efficient utilization of computational resources.

4. Results and Discussion

This section evaluates the proposed YOLOv11s optimization pipeline in terms of training behavior, validation performance, ablation analysis, comparative YOLO evaluation, precision–recall analysis, INT8 quantization, and runtime deployment. The analysis focuses on the trade-off among detection accuracy, computational complexity, and inference efficiency on both workstation GPU and NVIDIA Jetson-class hardware.

4.1. Model training result

The training pipeline is built on the YOLOv11s framework using Python 3.10 and is executed within a Google Colaboratory environment using an NVIDIA Tesla T4 GPU (16 GB VRAM) with CUDA acceleration. All experiments adopt an input resolution of 640 × 640 pixels, with training conducted for 50 epochs at a batch size of 8. Two model variants are developed: the baseline YOLOv11s and SDPP-YOLOv11s, in which the P3 and P4 detection paths are removed prior to fine-tuning. Both variants are optimized using identical hyperparameters to ensure a fair comparison between the original and pruned architectures. After the removal of the P3 and P4 detection paths, the SDPP-YOLOv11s model is fine-tuned for 50 epochs using the same training configuration as the baseline model. No extensive additional fine-tuning is required, as the training and validation losses converge stably and the final detection performance remains comparable to that of the baseline model.

Throughout the training phase, all loss components—including box, classification, and distribution focal losses—show a steady decline, which reflects stable convergence behavior. The validation losses follow a similar trend, indicating that neither model experiences overfitting. Key detection metrics, such as Precision, Recall, mAP@0.5, and mAP@0.5:0.95, show progressive improvement and converge before epoch 50. The corresponding training curves for the baseline YOLOv11s model are presented in Fig. 8.

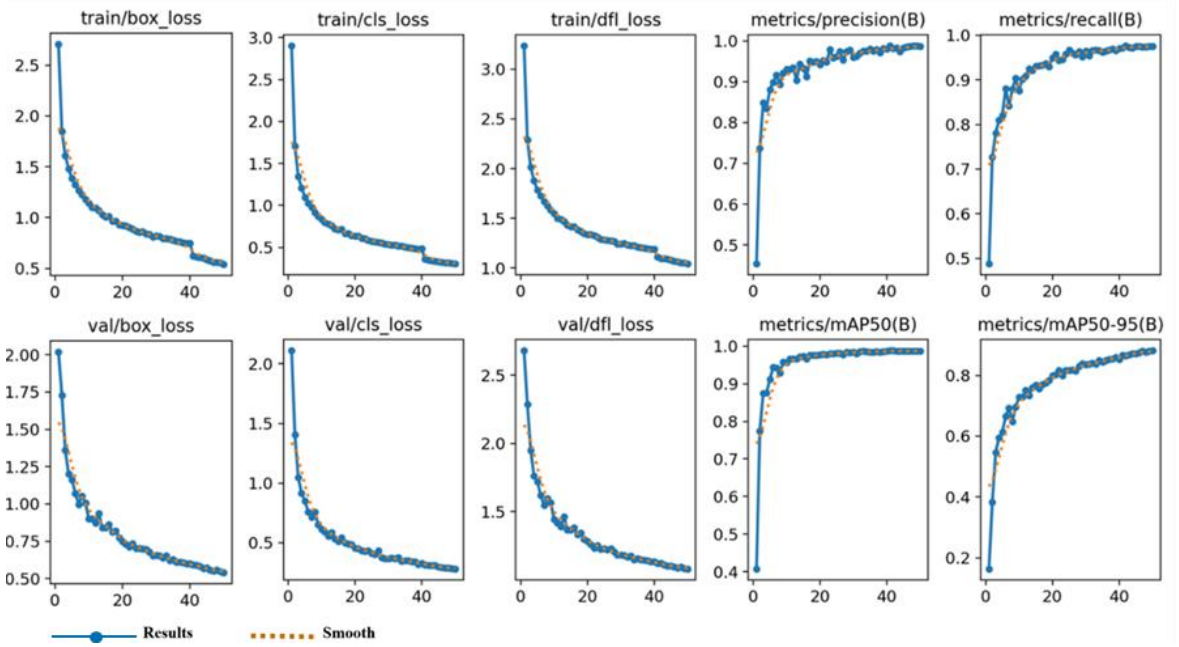


Fig. 8 Training and validation curves of the baseline YOLOv11s model

To further examine the effect of the structured detection-path pruning on model optimization, the same training procedure is applied to the SDPP-YOLOv11s model. In this configuration, the P3 and P4 detection branches are disabled, and operated solely on the P5 detection path. Owing to this architectural simplification, the learning dynamics of the pruned model may differ from those of the baseline model, particularly in terms of convergence speed and metric stability. Therefore, analyzing the training curves of the SDPP-YOLOv11s model is important to verify whether the pruning strategy introduces instability or negatively affects optimization. The training curves of the pruned model are shown in Fig. 9.

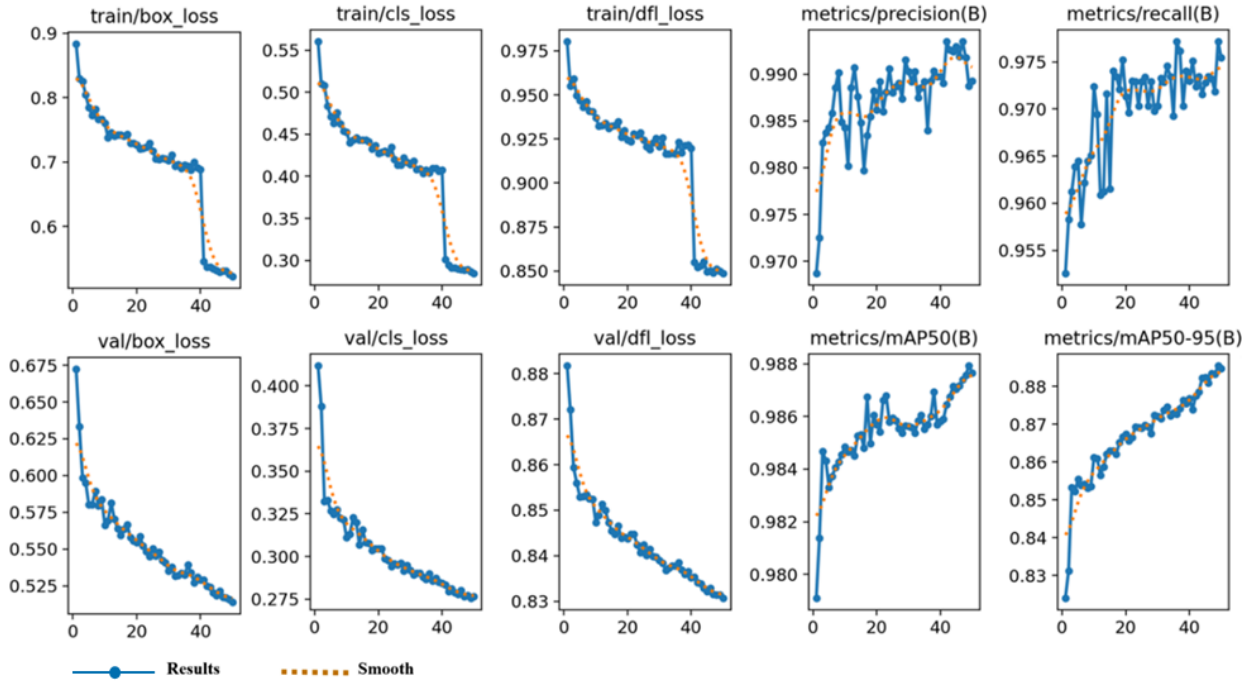


Fig. 9 Training and validation curves of the SDPP-YOLOv11s model

Furthermore, the behavior of the training and validation losses provides insight into the optimization stability of the proposed architecture. Similar to the baseline model, the loss components, including the box loss, classification loss, and distribution focal loss—exhibit a consistent downward trend during the training process. The validation losses decrease progressively, indicating that the SDPP-based structural modification does not introduce significant instability during optimization. Despite the removal of the P3 and P4 detection branches, the model continues to learn meaningful feature representations using the remaining P5 detection path. In addition to loss convergence, detection performance metrics such as Precision, Recall, mAP@0.5, and mAP@0.5:0.95 gradually improved during training.

Although the pruned architecture reduces the number of detection paths, the model maintains a stable metric growth across epochs. This observation suggests that the SDPP pruning strategy effectively simplifies the detection architecture while preserving the network's essential learning capability. Consequently, the pruned model achieves a reliable convergence behavior comparable to that of the baseline model.

4.2. Validation result

The validation performance of both the baseline YOLOv11s and SDPP-YOLOv11s models are evaluated using standard object detection metrics, including Precision, Recall, F1-score, mAP@0.5, and mAP@0.5:0.95. In addition, model complexity indicators, such as the number of layers and Giga Floating-Point Operations (GFLOPs), are analyzed to assess computational efficiency. A comparison of the detection performance and computational cost is summarized in Table 5.

Table 5 Validation performance and model complexity of baseline vs. SDPP-YOLOv11s

Model	P	R	F1 Score	mAP @0.5	mAP @0.5-0.95	Layer	GFLOPs
Baseline YOLOv11s	0.987	0.975	0.988	0.988	0.883	265	35.3
SDPP-YOLOv11s (Pruned)	0.991	0.972	0.987	0.987	0.883	177	16.4

As shown in Table 5, the SDPP-YOLOv11s model preserves detection performance that is highly comparable to the baseline architecture, despite the structural simplifications. The pruned model achieves a precision of 0.991 and a recall of 0.972, indicating that the removal of the P3 and P4 detection branches does not substantially affect the model's ability to detect FFB objects in UAV imagery. Furthermore, the mAP@0.5 and mAP@0.5:0.95 values remain at 0.987 and 0.883, respectively, closely matching those of the baseline model. These results suggest that the contributions of the fine- and medium-scale detection paths to the overall detection performance are relatively limited for this dataset. In contrast, the retained P5 detection path is sufficient to capture the dominant spatial characteristics of FFB objects in UAV imagery.

From a computational perspective, the proposed SDPP is considerably more significant. The pruning process reduces the total number of layers from 265 to 177, representing a structural reduction of approximately 33.2%. More importantly, the computational complexity decreases from 35.3 to 16.4 GFLOPs, corresponding to a reduction of approximately 53.5%. This substantial decrease indicates that the removed detection branches account for a large portion of the computational cost while contributing minimally to detection accuracy. By eliminating redundant paths, the model becomes significantly more efficient without sacrificing its predictive ability.

These findings highlight an important trade-off in multi-scale object detection architectures. Although multi-scale detection heads are designed to improve robustness across varying object sizes, they can introduce redundant computations when target objects appear at relatively consistent spatial scales. The SDPP strategy takes advantage of this characteristic by selectively removing detection paths with a lower contribution to performance, resulting in a more compact architecture. Overall, the SDPP-YOLOv11s model demonstrates a good compromise between detection accuracy and computational efficiency. The results indicate that the proposed pruning strategy effectively reduces architectural complexity while maintaining the core detection capability of the original YOLOv11s model. Consequently, the pruned model a suitable basis for subsequent INT8 quantization and deployment in resource-constrained edge environments.

4.3. Ablation study

To further justify the proposed SDPP strategy, an ablation study is conducted to evaluate the contribution of each detection head in the YOLOv11s architecture. The baseline model originally uses three detection heads, namely P3, P4, and P5, which are responsible for small-, medium-, and large-scale object detection, respectively. Since the proposed dataset is dominated by large-scale FFB instances, this study progressively evaluates different detection-head configurations by removing P3 only, removing P4 only, and removing both P3 and P4. This study aims to verify whether the removal of these detection paths reduces computational complexity while maintaining competitive box detection performance. The quantitative results of the evaluated detection-head configurations are summarized in Table 6.

Table 6 Ablation study of YOLOv11s detection-head configurations

Configuration	Active head detection	Layers	Parameters	GFLOPs	P	R	mAP@0.5	mAP@0.5:0.95
Baseline YOLOv11s	P3+P4+P5	265	10,067,203	35.3	0.987	0.975	0.988	0.883
P3 removed	P4+P5	101	9,413,187	21.3	0.990	0.975	0.983	0.866
P4 removed	P3+P5	93	9,171,202	20.5	0.986	0.976	0.983	0.866
P3+P4 removed / SDPP-YOLOv11s	P5	177	10,363,937	16.4	0.991	0.972	0.987	0.883

The ablation results show that removing individual detection heads substantially reduces computational complexity. Removing P3 reduces the GFLOPs from 35.3 to 21.3, corresponding to a 39.7% reduction, while maintaining a high mAP@0.5 of 0.983. Similarly, removing P4 reduces the GFLOPs to 20.5, corresponding to a 41.9% reduction, with the same mAP@0.5 of 0.983 and mAP@0.5:0.95 of 0.866. These results indicate that both P3 and P4 provide relatively limited improvements in box detection accuracy compared with their computational cost in the proposed UAV-based FFB dataset.

The SDPP-YOLOv11s configuration, which removes both P3 and P4 and retains only P5, achieves the largest computational reduction, decreasing GFLOPs from 35.3 to 16.4, or approximately 53.5%. Despite this, the model maintains a high mAP@0.5 of 0.987 and preserves the mAP@0.5:0.95 value of 0.883, which is comparable to the baseline. Although the parameter count slightly increases due to architectural reconfiguration, the overall computational workload is substantially reduced. This finding confirms that the P5 detection head provides sufficient high-level semantic representation for detecting the dominant large-scale FFB objects in the dataset. Therefore, the P5-only configuration is selected as the final SDPP architecture because it provides the best computational efficiency while preserving reliable box detection performance.

4.4. Comparative evaluation with recent YOLO architectures

To further highlight the advantage of using YOLOv11s as the baseline architecture, a comparative evaluation is conducted against recent small-scale YOLO variants, including YOLOv8s, YOLOv9s, and YOLOv10s. All models are evaluated using the same UAV-based FFB dataset and validation protocol. The small-scale variants are selected to ensure a fair comparison with the proposed YOLOv11s-based framework, allowing the evaluation to focus on architectural differences rather than model-scale differences.

As shown in Table 7, YOLOv8s and YOLOv9s achieve competitive detection performance, with mAP@0.5 values of 0.984 and 0.983, respectively. YOLOv10s has lower computational complexity than YOLOv8s and YOLOv9s, with 21.4 GFLOPs, but its detection accuracy decreases to 0.973 mAP@0.5 and 0.872 mAP@0.5:0.95. The baseline YOLOv11s attains the highest mAP@0.5 of 0.988, confirming its strong detection capability for UAV-based FFB detection. However, it also requires the highest computational cost, with 35.3 GFLOPs.

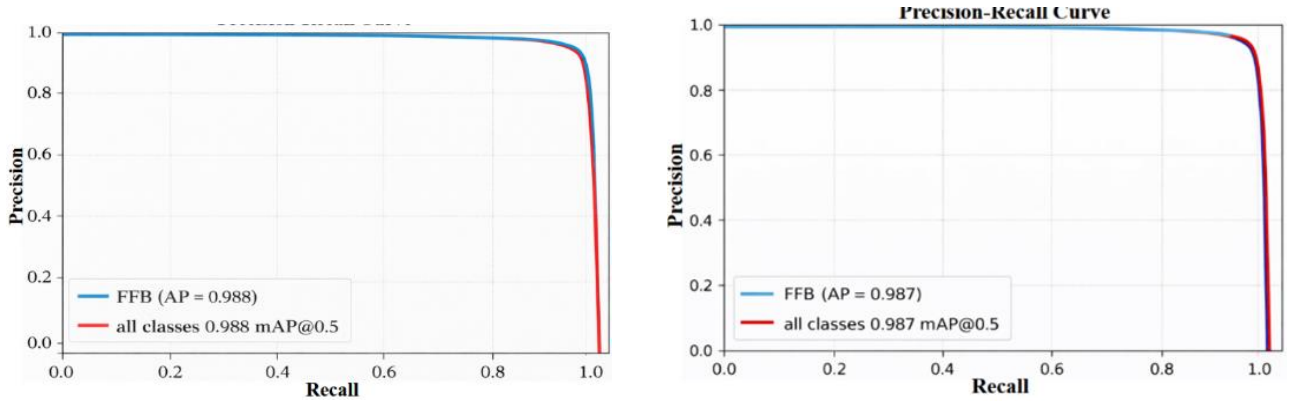
Table 7 Comparative performance of recent YOLO architectures on the UAV-based FFB dataset

Model	Layers	Parameters	GFLOPs	P	R	mAP@0.5	mAP@0.5:0.95
YOLOv8s	73	11,125,971	28.4	0.987	0.982	0.984	0.883
YOLOv9s	197	7,167,475	26.7	0.984	0.982	0.983	0.881
YOLOv10s	106	7,218,387	21.4	0.980	0.974	0.973	0.872
YOLOv11s	265	10,067,203	35.3	0.987	0.975	0.988	0.883
SDPP-YOLOv11s	177	10,363,937	16.4	0.991	0.972	0.987	0.883

The proposed SDPP-YOLOv11s achieves a more favorable balance between detection accuracy and computational efficiency. Although its mAP@0.5 slightly decreases from 0.988 to 0.987 compared with the baseline YOLOv11s, it substantially reduces the computational complexity from 35.3 to 16.4 GFLOPs. When compared with YOLOv8s, YOLOv9s, and YOLOv10s, the proposed SDPP-YOLOv11s maintains competitive detection accuracy while requiring the lowest GFLOPs. These results demonstrate that the proposed detection-path pruning strategy effectively improves the efficiency of YOLOv11s while preserving its detection capability, making it suitable for real-time UAV-based FFB detection on resource-constrained devices.

4.5. Precision–recall curve analysis

To further analyze the detection reliability of the proposed pruning strategy, precision–recall (PR) curves are generated for both the baseline YOLOv11s model and the SDPP-YOLOv11s model. Fig. 10 presents the PR curves for these models, which enables a direct comparison between the original architecture and the pruned variant. The overall curve shape and the area under the curve provide insight into how consistently each model maintains high precision while achieving higher recall during detection.



(a) Baseline YOLOv11s model (b) SDPP-YOLOv11s model

Fig. 10 Precision–recall curves

As shown in Fig. 10 (a)-(b), both the baseline YOLOv11s and SDPP-YOLOv11s models maintain high precision across nearly the entire recall range. The PR curve of the SDPP-YOLOv11s model in Fig. 10(b) closely overlaps with the baseline curve in Fig. 10(a), indicating that the proposed pruning strategy does not significantly degrade detection reliability. Although a minor reduction in precision is observed at very high recall levels, the overall curve shape and AP values remain nearly identical, with AP values of 0.988 for the baseline model and 0.987 for the SDPP-YOLOv11s model.

These results confirm that removing the P3 and P4 detection branches does not significantly affect the model's ability to detect FFB objects in UAV imagery. Therefore, the PR curve analysis is consistent with the quantitative results presented in Tables 5 and 6, demonstrating that SDPP preserves the essential detection characteristics of YOLOv11s while significantly reducing computational complexity.

4.6. Runtime performance analysis after INT8 quantization

After applying Structured Detection-Path Pruning, post-training INT8 quantization is introduced to further enhance the computational efficiency of the YOLOv11s model. While pruning reduces architectural complexity by eliminating redundant multi-scale detection paths, quantization improves inference speed by transforming floating-point (FP32) computations into INT8 operations. This lower-precision representation decreases memory bandwidth requirements and facilitates faster execution on hardware platforms that support INT8 optimization.

To assess the effect of quantization on runtime performance, inference experiments are conducted using UAV video datasets. The evaluation emphasizes processing speed, measured in frames per second (FPS), as well as per-frame inference time in milliseconds. In this work, inference time is calculated on a per-frame basis, encompassing preprocessing, model inference, and post-processing stages. All experiments are carried out under consistent hardware settings and identical input resolutions to ensure a fair comparison among the optimized models. The reported FPS values derive from the complete video processing pipeline, including frame decoding, preprocessing, inference, and post-processing. The results of the runtime evaluation are presented in Fig. 11.

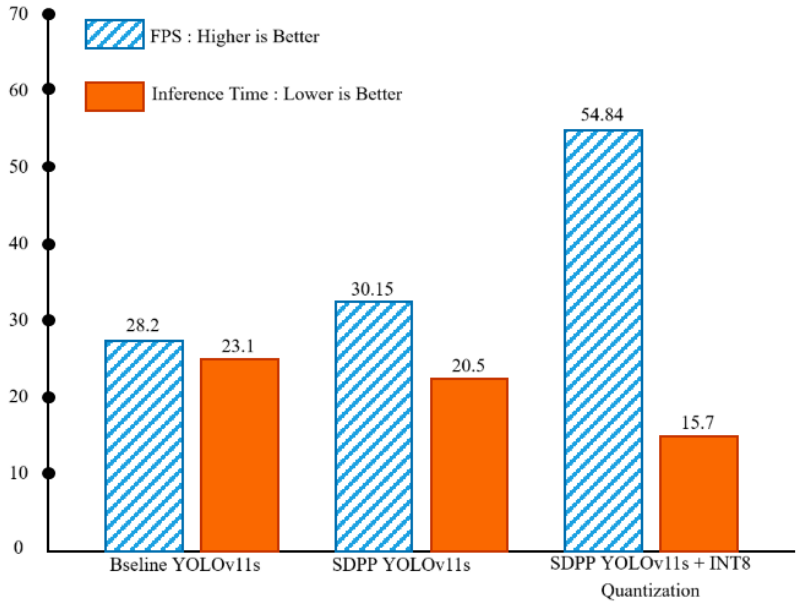


Fig. 11 Comparison of YOLOv11s in terms of FPS and inference time

As illustrated in Fig. 11, the baseline YOLOv11s model achieves an inference speed of 28.2 FPS, corresponding to an average inference time of 23.1 ms per frame. After applying SDPP, the inference speed increases to 30.15 FPS, while the inference time decreases to 20.5 ms per frame. This represents an improvement of approximately 6.9% in inference speed compared to the baseline model. This improvement is primarily due to the removal of redundant detection paths, which reduces the overall computational workload of the network. Note that the reported FPS values are measured across the entire video processing pipeline, including frame decoding, preprocessing, inference, and post-processing.

A significantly larger performance gain is observed after applying INT8 quantization. The pruned and quantized model achieves 54.84 FPS, with an average inference time of 15.7 ms per frame, representing a 94.5% increase in speed over the baseline model. This result demonstrates the effectiveness of low-precision integer arithmetic in accelerating neural network inference, particularly when applied to an already simplified architecture[19]. Similar improvements in computational efficiency through model quantization for edge inference deployment are reported in recent studies[20-21].

Importantly, the improvement in runtime efficiency does not significantly compromise detection accuracy. As demonstrated by the validation results, the accuracy reduction after pruning and quantization remains below 0.3%. This indicates that the proposed optimization pipeline successfully enhances computational efficiency while preserving reliable

detection performance. Overall, the combination of SDPP and INT8 quantization forms an effective optimization strategy: pruning reduces architectural redundancy and computational complexity, whereas quantization accelerates inference through low-precision computations. Together, these optimizations enable the YOLOv11s model to achieve real-time performance suitable for UAV-based FFB detection. A detailed comparison of the baseline, pruned, and quantized models in terms of model complexity is presented in Table 8.

Table 8 Complexity comparison of baseline, pruned, and INT8-quantized YOLOv11 models

Model variant	Precision	Parameters	GFLOPs	Model size
Baseline YOLOv11s	FP32	10,067,203	35.3	19.9 MB
SDPP-YOLOv11s	FP32	10,363,937	16.4	19.5 MB
SDPP-YOLOv11s + INT8 quantized	INT8	10,363,937	16.4	12.6 MB

Table 8 presents a comparison of the structural complexities of the baseline YOLOv11s model, the pruned SDPP-YOLOv11s model, and INT8-quantized variants. After applying SDPP, the computational complexity decreases significantly from 35.3 GFLOPs to 16.4 GFLOPs, corresponding to a reduction of approximately 53.5%. This reduction confirms that removing the P3 and P4 detection branches effectively eliminates redundant multi-scale detection computations.

Notably, the parameter count of the pruned model slightly increases from 10.07M to 10.36M. This suggests that the architectural reconfiguration of the detection paths after pruning introduces additional parameters. However, despite this marginal increase, the overall computational workload is substantially reduced, as reflected by the significant decrease in GFLOPs and improved inference speed observed in the runtime evaluation.

The impact of INT8 quantization primarily affects model storage size rather than network structure. While the number of parameters and GFLOPs remain unchanged, the model size reduced from 19.5 MB to 12.6 MB, corresponding to an approximate 35% reduction compared to the pruned FP32 model. This reduction is achieved by converting the weights from FP32 to INT8 representations, which significantly reduces the memory footprint of the model.

Overall, these results highlight the complementary roles of pruning and quantization in the proposed optimization pipeline: pruning reduces computational complexity, whereas quantization compresses the model representation and improves runtime efficiency. To illustrate the combined impact of these optimizations on detection accuracy, computational complexity, and runtime performance, a consolidated comparison across all model variants is presented in Table 9.

Table 9 Comprehensive performance comparison of baseline, pruned, and INT8-quantized YOLOv11 models

Model variant	Data precision	mAP@0.5	Parameters	GFLOPs	Model size(MB)	FPS	Speed-up
Baseline YOLOv11s	FP32	0.988	10,067,203	35.3	19.9	28.2	–
SDPP-YOLOv11s (Pruned)	FP32	0.987	10,363,937	16.4	19.5	30.15	1.07×
SDPP-YOLOv11s + INT8 quantized	INT8	0.987*	10,363,937*	16.4*	12.6	54.84	1.94×

* Values remain unchanged after INT8 quantization except model size.

The optimization pipeline introduced in this study successfully maintains a well-balanced trade-off between detection precision and processing efficiency across various model configurations. The baseline YOLOv11s model records an mAP@0.5 of 0.988 at an inference speed of 28.20 FPS. Upon the implementation of SDPP, the resulting SDPP-YOLOv11s model retained nearly equivalent detection performance, with the mAP@0.5 marginally declining to 0.987. Notably, the computational burden is substantially lowered from 35.3 GFLOPs to 16.4 GFLOPs, which contributes to a slight improvement in processing speed, with the model reaching 30.15 FPS.

A considerably greater performance gain is achieved following the application of INT8 quantization. The quantized SDPP-YOLOv11s model demonstrates an inference speed of 54.84 FPS, reflecting an approximately 1.94× acceleration relative to the original baseline model. Furthermore, the model's memory footprint is reduced from 19.9 MB to 12.6 MB as a result of adopting lower-precision INT8 numerical representations.

Importantly, detection accuracy remains largely unaffected despite this substantial compression and the reduction in numerical precision, confirming that the optimized model retains robust detection capability. Collectively, these findings highlight that SDPP and INT8 quantization offer complementary advantages: the former eliminates redundant computations across multi-scale detection layers, while the latter compresses the model's internal representation and markedly enhances inference throughput. The combined effect of these two techniques enables the YOLOv11s model to operate efficiently in real-time, rendering it well-suited for deployment within UAV-base FFB detection frameworks in hardware-constrained environments.

4.7. Runtime performance analysis on Jetson

To evaluate the real-time deployment capability of the optimized SDPP-YOLOv11s model, experiments are conducted using UAV-captured video streams on an NVIDIA Jetson Orin Nano. Video-based evaluation is essential because it reflects realistic operational conditions in plantation environments, including camera motion, frame-to-frame variability, illumination changes, and dynamic FFB appearance. Compared to static image inference, this setup provides a more representative assessment of runtime performance during actual UAV operations.

The Jetson Orin Nano is equipped with a 1024-core Ampere GPU, six ARM Cortex-A78AE CPU cores, and 8 GB LPDDR5 memory. The device ran JetPack 6.2, which included CUDA, cuDNN, TensorRT, and an Ubuntu-based operating system. All deployment routines are implemented in Python 3.10.12 using PyTorch 2.5.0a0+, Ultralytics 8.3.202, OpenCV 4.11.0.86, NumPy 1.23.5, and ONNX Runtime GPU 1.20.0. TensorRT 10.3.0 is used to generate an INT8 inference engine from the pruned ONNX model. During the evaluation, the Jetson Orin Nano processed continuous UAV video streams. The reported inference time represents the per-frame processing time of the complete pipeline under real-time video conditions. The runtime performance results are presented in Table 10.

Table 10 Runtime performance on Jetson Orin Nano (video-based evaluation)

Model variant	Data precision	Average FPS	Inference time (ms)
Baseline YOLOv11s	FP32	16.5	52.24
SDPP-YOLOv11s	FP32	22.08	30.25
SDPP-YOLOv11s + INT8 quantized	INT8	30.42	25.85

As shown in Table 10, the baseline FP32 YOLOv11s model achieves an average processing speed of 16.50 FPS during UAV video inference on the Jetson Orin Nano, with an average inference time of 52.24 ms per frame. This performance indicates that real-time deployment is challenging for an unoptimized model when running on embedded edge hardware. After applying SDPP, the SDPP-YOLOv11s model improves the runtime performance to 22.08 FPS while reducing the inference time to 30.25 ms per frame. This corresponds to an increase of approximately 33.8% in processing speed and a reduction of 42.1% in per-frame processing time compared to the baseline model. This improvement is mainly attributed to the removal of redundant multi-scale detection branches, which reduces the computational workload of the detection head during inference.

The highest runtime performance is achieved by the INT8-quantized SDPP-YOLOv11s model, which reaches 30.42 FPS with an average inference time of 25.85 ms per frame. This represents a substantial improvement over both the baseline and pruned FP32 models, resulting in an approximately 84.4% higher FPS and a 50.5% reduction in inference time relative to the baseline model. The results demonstrate that INT8 quantization significantly accelerates inference on embedded GPU

architectures by reducing numerical precision from FP32 to INT8 and enabling more efficient hardware utilization. It should be noted that the reported FPS values represent the average processing speed measured during continuous UAV video playback, including frame decoding, preprocessing, inference, and post-processing. Therefore, the average FPS does not directly correspond to the model inference time.

A qualitative inspection of detection results during continuous UAV video playback shows that the SDPP-YOLOv11s INT8 model maintains stable and consistent detections under dynamic field conditions. Bounding boxes remain temporally stable despite rapid camera motion, illumination changes, and varying foliage density. The detector also preserves accurate localization, even under partial occlusions caused by palm fronds covering portions of the fruit bunch. Although occasional detection errors may appear in heavily cluttered scenes, these instances are relatively infrequent and do not significantly affect overall detection reliability during UAV flights.

Overall, the results indicate that the optimized SDPP-YOLOv11s INT8 model achieves a favorable balance between detection accuracy and runtime efficiency on the Jetson Orin Nano, This balance makes it suitable for real-time UAV-based FFB detection applications in plantation environments. To further evaluate the consistency of the optimization effects across different hardware platforms, a cross-device runtime comparison is conducted between the development laptop (PC) and the Jetson Orin Nano. The results are summarized in Table 11.

Table 11 Cross-device runtime comparison between PC and Jetson Orin Nano

Model variant	Data precision	FPS (PC)	FPS (Jetson)	Inference time PC (ms)	Inference time Jetson (ms)
Baseline YOLOv11s	FP32	28.2	16.5	23.1	52.24
SDPP-YOLOv11s	FP32	30.15	22.08	20.5	30.25
SDPP-YOLOv11s + INT8 quantized	INT8	54.84	30.42	15.7	25.85

Table 11 presents a cross-device runtime comparison between the development PC and the NVIDIA Jetson Orin Nano. Instead of reiterating the raw values reported in the table, this section highlights the substantial performance improvements achieved using the proposed optimization pipeline. Compared to the baseline YOLOv11s FP32 model, the INT8-quantized SDPP-YOLOv11s model significantly improves inference speed on both platforms. On the Jetson Orin Nano, the frame rate increases from 16.5 to 30.42 FPS—representing an approximately 84% improvement—while the inference time is reduced from 52.24 ms to 25.85 ms per frame. A similar trend is observed on the PC, further confirming that the optimization consistently enhances runtime performance across different hardware environments.

Despite the limited computational capacity of the Jetson device, the optimized model maintains a clear performance advantage over the baseline. This demonstrates that the combination of SDPP and INT8 quantization is both effective and relatively hardware-agnostic. Overall, these findings confirm that the proposed approach successfully balances detection accuracy and computational efficiency. This making it well-suited for real-time UAV-based FFB detection on resource-constrained edge devices, in line with recent studies on lightweight object detection for UAV deployment.

Although the proposed SDPP-YOLOv11s model achieves strong detection performance, several failure cases may still occur under challenging UAV imaging conditions. False negatives can appear when FFB objects are partially occluded by palm fronds, have low visual contrast, or are only partially visible. Conversely, false positives occur when vegetation, dry fronds, or shadow regions exhibit visual patterns similar to FFB objects. These cases indicate that occlusion, cluttered backgrounds, illumination variation, and partially visible FFB instances remain challenging factors that should be further evaluated using more diverse UAV altitudes, plantation locations, and external UAV datasets.

5. Conclusion and Future Work

This study proposed a two-stage optimization framework that combines SDPP and post-training INT8 quantization to improve the efficiency of the YOLOv11s model for oil palm FFB detection using UAV imagery. The proposed approach effectively reduces computational complexity while preserving detection accuracy, enabling real-time deployment on resource-constrained edge devices. The main findings are summarized as follows:

(1) SDPP significantly reduced architectural complexity by decreasing the number of layers from 265 to 177 (33.2%) and reducing computational cost from 35.3 to 16.4 GFLOPs (53.5%).

(2) The pruned model maintained comparable detection performance, with mAP@0.5 decreasing only slightly from 0.988 to 0.987, demonstrating that the removed detection branches contributed minimally to the target detection task.

(3) INT8 quantization further improved inference efficiency, increasing the inference speed from 30.15 FPS to 54.84 FPS on the desktop evaluation platform, while achieving 30.42 FPS during real-time deployment on the NVIDIA Jetson Orin Nano

(4) The integration of SDPP and INT8 quantization provides an effective balance between computational efficiency and detection accuracy, making the proposed framework suitable for UAV-based precision agriculture applications.

The findings demonstrate that task-specific detection-path pruning combined with lightweight inference optimization can effectively improve the deployability of deep learning-based FFB detection systems. Future work will investigates quantization-aware training (QAT), cross-location validation, multi-altitude UAV acquisition, gimbal-angle variation, external UAV datasets, and evaluations under diverse weather and illumination conditions to further assess model robustness and generalization capability.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Z. Khan, Y. Shen, and H. Liu, “Object Detection in Agriculture: A Comprehensive Review of Methods, Applications, Challenges, and Future Directions,” Agriculture, vol. 15, no. 13, article no. 1351, 2025.

[2] R. Guebsi, S. Mami, and K. Chokmani, “Drones in Precision Agriculture: A Comprehensive Review of Applications, Technologies, and Challenges,” Drones, vol. 8, no. 11, article no. 686, 2024.

[3] S. Siang, L. G. Lim, S. Palaiahnakote, J. X. Cheong, S. S. M. Lock, and M. N. B. Ayub ., “Oil Palm Tree Detection in UAV Imagery Using an Enhanced RetinaNet,” Computers and Electronics in Agriculture, vol. 227, part 1, article no. 109530, 2024.

[4] C. M. Badgujar, A. Poulose, and H. Gan, “Agricultural Object Detection with You Only Look Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review,” Computers and Electronics in Agriculture, vol. 223, article no. 109090, 2024.

[5] Y. Luo, A. Wu, and Q. Fu, “MAS-YOLOv11: An Improved Underwater Object Detection Algorithm Based on YOLOv11,” Sensors, vol. 25, no. 11, article no. 3433, 2025.

[6] R. C. C. D. M. Santos, M. Coelho, and R. Oliveira, “Real-Time Object Detection Performance Analysis Using YOLOv7 on Edge Devices,” IEEE Latin America Transactions, vol. 22, no. 10, pp. 799-805, 2024.

[7] P. V. Dantas, W. S. da S. Jr, L. C. Cordeiro, and C. B. Carvalho , “A Comprehensive Review of Model Compression Techniques in Machine Learning,” Applied Intelligence, vol. 54, pp. 11804-11844, 2024.

[8] J. W. Lai, H. R. Ramli, L. I. Ismail, and W. Z. W. Hasan, “Oil Palm Fresh Fruit Bunch Ripeness Detection Methods: A Systematic Review,” Agriculture, vol. 13, no. 1, article no. 156, 2023.

[9] F. A. Junior and Suharjito, “Video Based Oil Palm Ripeness Detection Model Using Deep Learning,” Heliyon, vol. 9, no. 1, article no. e13036, 2023.

[10] H. Wibowo, I. S. Sitanggang, M. Mushthofa, and H. A. Adrianto, “Large-Scale Oil Palm Trees Detection from High-Resolution Remote Sensing Images Using Deep Learning,” Big Data and Cognitive Computing, vol. 6, no. 3, article no. 89, 2022.

[11] C. Chang, R. Parthiban, V. Kalavally, Y. M. Hung, and X. Wang, "Unharvested Palm Fruit Bunch Ripeness Detection with Hybrid Color Correction," Smart Agricultural Technology, vol. 9, article no. 100643, 2024.

[12] S. Suharjito, M. G. Naftali, G. Hugo, M. R. A. Priyadi, M. Asrol, and D. N. Utama, "Oil Palm Fruits Dataset in Plantations for Harvest Estimation Using Digital Census and Smartphone," Scientific Data, vol. 12, article no. 1, 2025.

[13] A. Zagitov, E. Chebotareva, A. Toschev, and E. Magid, "Comparative Analysis of Neural Network Models Performance on Low-Power Devices for a Real-Time Object Detection Task," Computer Optics, vol. 48, no. 2, pp. 242-252, 2024.

[14] S. Solak, B. Cetin, M. Hikmet, and B. Ucar, "Real-time theft detection in urban surveillance : A comparative analysis of YOLO-based approach," Journal of Engineering Research, vol. 14, no. 2, pp. 1531-1547, 2026.

[15] H. Ahn, S. Son, J. Roh, H. Baek, S. Lee, Y. Chung, et al., "SAFP-YOLO: Enhanced Object Detection Speed Using Spatial Attention-Based Filter Pruning," Applied Sciences, vol. 13, no. 20, article no. 11237, 2023.

[16] Y. Zhou, Z. Guo, Z. Dong, and K. Yang, "TensorRT Implementations of Model Quantization on Edge SoC," Proceedings of the 2023 IEEE 16th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC), pp. 486-493, 2023.

[17] Z. Liu, C. Chen, Z. Huang, Y. C. Chang, L. Liu, and Q. Pei, "A Low-Cost and Lightweight Real-Time Object Detection Method Based on UAV Remote Sensing in Transportation Systems," Remote Sensing, vol. 16, no. 19, article no. 3712, 2024.

[18] M. L. Ali and Z. Zhang, "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection," Computers, vol. 13, no. 12, article no. 336, 2024.

[19] L. Wei, Z. Ma, C. Yang, and Q. Yao, "Advances in the Neural Network Quantization: A Comprehensive Review," Applied Sciences, vol. 14, no. 17, article no. 7445, 2024.

[20] Q. Fan, Y. Li, M. Deveci, K. Zhong, and S. Kadry, "LUD-YOLO: A Novel Lightweight Object Detection Network for Unmanned Aerial Vehicle," Information Sciences, vol. 686, article no. 121366, 2025.

[21] Z. Li, H. Li, and L. Meng, "Model Compression for Deep Neural Networks: A Survey," Computers, vol. 12, no. 3, article no. 60, 2023.



Copyright© by the authors. Licensee TAETI, Taiwan. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC) license (https://creativecommons.org/licenses/by-nc/4.0/).