

Article

PestLite: A Novel YOLO-Based Deep Learning Technique for Crop Pest Detection

Qing Dong ¹, Lina Sun ¹, Tianxin Han ^{1,*}, Minqi Cai ² and Ce Gao ³

¹ Department of Process Equipment and Control Engineering, School of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China; 2270324@stu.neu.edu.cn (Q.D.); linsun@mail.neu.edu.cn (L.S.)

² Institute of Insect Research, College of Animal Science and Technology, Yangzhou University, Yangzhou 225000, China; cai-minqi@foxmail.com

³ School of Economics and Business Administration, Beijing Normal University, Beijing 100875, China; 202328030253@mail.bnu.edu.cn

* Correspondence: 2270327@stu.neu.edu.cn

Abstract: Timely and effective pest detection is essential for agricultural production, facing challenges such as complex backgrounds and a vast number of parameters. Seeking solutions has become a pressing matter. This paper, based on the YOLOv5 algorithm, developed the PestLite model. The model surpasses previous spatial pooling methods with our uniquely designed Multi-Level Spatial Pyramid Pooling (MTSPPF). Using a lightweight unit, it integrates convolution, normalization, and activation operations. It excels in capturing multi-scale features, ensuring rich extraction of key information at various scales. Notably, MTSPPF not only enhances detection accuracy but also reduces the parameter size, making it ideal for lightweight pest detection models. Additionally, we introduced the Involution and Efficient Channel Attention (ECA) attention mechanisms to enhance contextual understanding. We also replaced traditional upsampling with Content-Aware ReAssembly of FEatures (CARAFE), which enable the model to achieve higher mean average precision in detection. Testing on a pest dataset showed improved accuracy while reducing parameter size. The mAP50 increased from 87.9% to 90.7%, and the parameter count decreased from 7.03 M to 6.09 M. We further validated the PestLite model using the IP102 dataset, and on the other hand, we conducted comparisons with mainstream models. Furthermore, we visualized the detection targets. The results indicate that the PestLite model provides an effective solution for real-time target detection in agricultural pests.

Keywords: pest detection; agricultural production; real-time target detection; MTSPPF; ECA; involution; CARAFE; YOLOv5; IP102



Citation: Dong, Q.; Sun, L.; Han, T.; Cai, M.; Gao, C. PestLite: A Novel YOLO-Based Deep Learning Technique for Crop Pest Detection. *Agriculture* **2024**, *14*, 228. <https://doi.org/10.3390/agriculture14020228>

Academic Editor: Bin Xie

Received: 8 November 2023

Revised: 23 December 2023

Accepted: 27 December 2023

Published: 31 January 2024



Copyright: © 2024 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/>).

1. Introduction

In modern agriculture, pest detection and management are crucial for the health and yield of crops, particularly for staple grains such as rice and maize, which are pivotal to the global food supply. According to statistics from the Food and Agriculture Organization (FAO) of the United Nations, global crop losses due to pests are estimated to reach 20–40% annually [1]. This not only leads to significant economic losses but also causes irreversible damage to the ecological environment. Therefore, for the sustainable development of global agriculture, it is of vital importance to develop efficient, accurate, and real-time methods for pest detection. Currently, with the widespread application of deep learning-based computer vision in agriculture, integrating advanced image processing techniques with practical agricultural applications to achieve large-scale, efficient, and real-time pest detection is emerging as a prominent research area [2]. However, due to the diversity and various morphologies of pests, establishing a general yet targeted pest detection model is a significant challenge researchers face. It is against this backdrop that the evolution of pest

detection methodologies becomes particularly insightful. By understanding the trajectory of these techniques, we can appreciate the nuances and challenges that the field has faced and continues to face. For these reasons, we decided to thoroughly explore pest detection technology and aim to leverage the latest deep learning methods to enhance detection accuracy and real-time performance.

The technical advancements in pest detection methodologies can be divided into several stages: firstly, the limitations of classical methods; secondly, the revolutionary progress of deep learning; and finally, the emergence of lightweight deep learning frameworks.

Historically, the inception of pest detection primarily relied on manual visual inspection, often supplemented by tools such as magnifying glasses [3]. As computer vision technology advanced, automated tools were developed, utilizing techniques like thresholding, edge detection, and contour analysis [4]. Among these, background subtraction became prevalent, effectively distinguishing moving pests from static backgrounds. Additionally, statistical methods like clustering and regression were introduced to assess insect behaviors [5]. Despite these advancements, traditional methods faced challenges in complex scenarios and when dealing with diverse insect postures. The limitations of manual inspection are evident: it is inefficient and susceptible to human error, especially under fatigue [6]. As such, manual methods become particularly cumbersome for large-scale tasks and struggle with detecting minute or hard-to-identify pests. With the increasing demand for precise detection across varied conditions and with multiple pest species, it is clear that traditional methods are not fully equipped to meet the needs of contemporary agriculture and pest management.

In recent years, Convolutional Neural Networks (CNN) have revolutionized the field of computer vision, yielding significant advancements in image recognition and object detection [7]. These networks extract hierarchical features from images via multi-layered convolution operations, leading to substantial improvements in the model performance. Notable convolution architectures such as ResNet, VGG, and particularly, the YOLO (You Only Look Once) series—including YOLOv5, YOLOv7, and YOLOv8—have become powerful tools for real-time object detection [8]. For instance, enhancements in YOLOv7 and YOLOv8 have improved detection accuracy and speed, while YOLOv5 has demonstrated its adaptability and effectiveness in various real-world applications.

Deep learning, especially via the YOLO architecture, has undeniably transformed object detection, offering transformative solutions across multiple domains, including the critical area of pest detection. Researchers have been refining these models to address the unique challenges of detecting pests. The introduction of the IP102 dataset by Wu et al. [9], which includes over 75,000 images across 102 categories, has been pivotal in this domain. Additionally, tailored solutions, like Cheng Chuanxiang and Yang Jia's adaptive YOLOv5-OB method for small pest detection and the developed PestNet model [10], with a precision score of 75.46% on the extensive MPD2018 dataset, highlight the potential of custom deep learning applications. The emphasis on image quality by Wei Dai and Daniel Berleant further indicates its critical role in enhancing model efficacy. Despite the precision of these specialized models, their computational demands underscore the ongoing need for efficient deep learning solutions in pest detection [11].

Lightweight deep learning models have become a focal point of innovation, specifically designed to meet the computational and memory constraints inherent in real-time pest detection on resource-limited devices [12]. These models prioritize a delicate balance between efficiency and accuracy, a necessity for deployment in practical scenarios. Early efforts to streamline deep learning architectures involved model compression and pruning strategies, which effectively reduced model size while maintaining performance levels. Advancements such as knowledge distillation, highlighted by the work of Redmon, J et al. [13], further refined this approach, enabling smaller models to emulate the success of significantly larger models. Architectures like ShuffleNet and Tiny-YOLO represent a leap forward in this direction, offering reduced computational demands without compromising detection effectiveness. Complementing these are hardware optimizations

and adaptive networks, as well as Neural Architecture Search (NAS) techniques which collectively enhance the operational efficiency of these models. Notably, YOLO iterations such as Tiny-YOLO and YOLOv4-tiny have specifically been optimized for the demands of pest detection, combining a smaller footprint with high accuracy. Moving forward, the challenge lies in perfecting the synergy between model compactness and predictive accuracy, a pursuit that stands to revolutionize real-time agricultural pest detection with models that are as efficient as they are precise.

As mentioned above, although models like YOLOv5 have been optimized for generic object detection tasks, the unique challenges presented by pest detection reveal certain limitations in traditional convolutional architectures. Pests, in their vast diversity of size, shape, and morphological variations, present intricate detection challenges. The often minute differences between various pests can easily escape the attention of standard convolutional architectures. The situation is further exacerbated when pests cluster. Such aggregations demand a model with high precision, capable of discerning each overlapping target distinctly. In the agricultural sector, the diversity of pests is vast, with each varying in size, ecological habits, and the impact they have on crops. From tiny aphids to larger beetles, each pest has unique characteristics that make precise detection a challenge. Specifically, certain pests target specific crops, causing varying degrees of damage. Thus, real-time and accurate detection and identification of these pests become crucial for the health of crops and the enhancement of agricultural yields. However, the myriad of pest species and the considerable variation in their sizes present a formidable challenge for detection. Achieving high-precision identification of these diverse pests, while ensuring the real-time performance and computational efficiency of the model, is undoubtedly a significant challenge. Therefore, we believe there is a pressing need to develop a new pest detection model that offers high recognition rates, low parameter count, and exceptional real-time capabilities.

In addressing the prevalent challenges in pest detection, this study aims to bridge the gap between existing methodologies and the emerging needs of the agricultural domain, particularly in relation to crops such as rice and maize. While traditional deep learning models possess significant power, they often falter when confronted with the subtle nuances associated with pests specific to these crops. Given the unique behavioral and morphological attributes of these pests, a targeted approach is necessitated. By introducing a more specialized model that undergoes in-depth optimization, we deeply understand the demands of this specific scenario. With these considerations and the unique challenges brought about by pests, this paper presents PestLite, a high-precision, lightweight detection model primarily for crop pests (focusing on rice and maize). The main innovative contributions of this paper include:

1. We have designed the MTSPPF, an advanced Multi-Level Spatial Pyramid Pooling module, which stands out as one of our most significant enhancements. Anchored by the lightweight unit, the MTSPPF efficiently amalgamates convolution, normalization, and activation to capture multi-scale features, ensuring the enriched extraction of contextual information. MTSPPF's strength lies in its dual capability: the model is streamlined by trimming the number of parameters, leading to improved detection accuracy at the same time. Balancing model efficiency with accuracy has been a core principle in our design approach. This architecture has demonstrated its effectiveness in subsequent experiments, especially in specialized tasks like high-density insect pest detection.
2. We introduced Involution: Unlike traditional convolution operations, Involution presents an adaptive weight adjustment mechanism that dynamically modifies its receptive field based on the image's content [14]. This novel approach not only ensures that the model captures vital contextual information across diverse scenarios, especially enhancing recognition accuracy for small target pests, but also streamlines the model, leading to a lightweight design without sacrificing performance.

3. We introduced the ECA Attention Mechanism: Considering the diverse morphologies and sizes of pests, we incorporated the ECA attention mechanism to intensify the model's focus on critical features. Through this mechanism, the model can more accurately capture and differentiate various pest features, enhancing detection accuracy and robustness [15].
4. We replaced the sampling pattern with CARAFE: To ensure the model retains critical feature information during upsampling, we employed the CARAFE upsampling operator [16]. This operator effectively preserves detailed information during upsampling, allowing the model to better capture features of small target pests, subsequently enhancing detection accuracy.

The structure of this manuscript is outlined as follows. Section 2 explains the origin of the dataset and provides a detailed explanation of our PestLite method. In Section 3, we present and discuss our experimental results, emphasizing the effectiveness of our approach. Finally, Section 4 concludes the paper and suggests directions for future research.

2. Materials and Methods

2.1. Mydataset

Mydataset has been meticulously curated for real-time monitoring of pests affecting key crops, primarily rice and maize. Initially, 6000 images were sourced from “The Teddy Cup Pest Dataset [17]”. To further cater to the specificities of rice and maize pests and to enrich the dataset, we captured an additional 626 images using a Sony Alpha a6400 camera (Tokyo, Japan) in our laboratory settings, bringing the total to 6626 images. For effective machine learning model training and evaluation, it is imperative to maintain a systematic distribution of data. Thus, of the total 6626 images, we allocated approximately 60% (3975 images) for training, allowing the model to learn the distinguishing features of different pests. About 20% (1325 images) were set aside for validation to fine-tune the model parameters and prevent overfitting, and the remaining 20% (1326 images) were used as a testing set, ensuring robust evaluation of the model's performance on previously unseen data.

This dataset primarily includes the following nine categories of pests. These include ‘DaoZhong’ (*Cnaphalocrocis medinalis*), which primarily targets rice plants, causing noticeable damage on rice leaves and potentially leading to significant reductions in the rice yield. ‘ErHua’ (*Spodoptera litura*) is a prolific pest affecting a range of crops, especially maize, with its larvae feeding on leaves during nighttime. The pest ‘DaMingShen’ (*Sesamia inferens*) has larvae that burrow into stalks, causing internal harm especially to maize and rice crops. ‘HeiBai’ (*Athetis lepigone*) affects a variety of crops by feeding on their leaves, while ‘DaoMinLin’ (*Naranga aenescens* Moore) predominantly affects rice, potentially leading to decreased yields. The dataset also features ‘YuMi’ (*Ostrinia furnacalis*), a major threat to maize due to its larvae that penetrate maize stalks, affecting its growth and reproduction. Additionally, ‘YangXue’ (*Lymantria dispar* or *Lymantria xyliina*) is active in forests and orchards, damaging trees, especially poplars and apple trees. The underground pest ‘LouGu’ (*Gryllotalpa orientalis*) affects plant roots, often leading to plant death. Lastly, ‘JinGui’ (*Holotrichia oblita* Faldermann) is a versatile pest with larvae in the soil and adults affecting leaves and fruits above ground. Figure 1 provides a representative sampling of the images, underscoring the variety within the dataset.

2.2. IP102

To comprehensively assess the efficacy of our PestLite algorithm under varying conditions and ensure its broad applicability, we expanded our experimental scope beyond initial trials. We conducted a series of additional tests using the IP102 dataset. The IP102 dataset serves as an extensive insect pest image dataset, curated to provide a standardized platform for agricultural pest detection and recognition. Spanning 102 unique pest categories, it covers a broad spectrum of pests commonly found across diverse crops and environments. The dataset, in its entirety, consists of 18,975 images. For a structured

evaluation, the dataset was split following the 6:2:2 ratio, allocating approximately 60% for training, 20% for validation, and the remaining 20% for testing purposes.

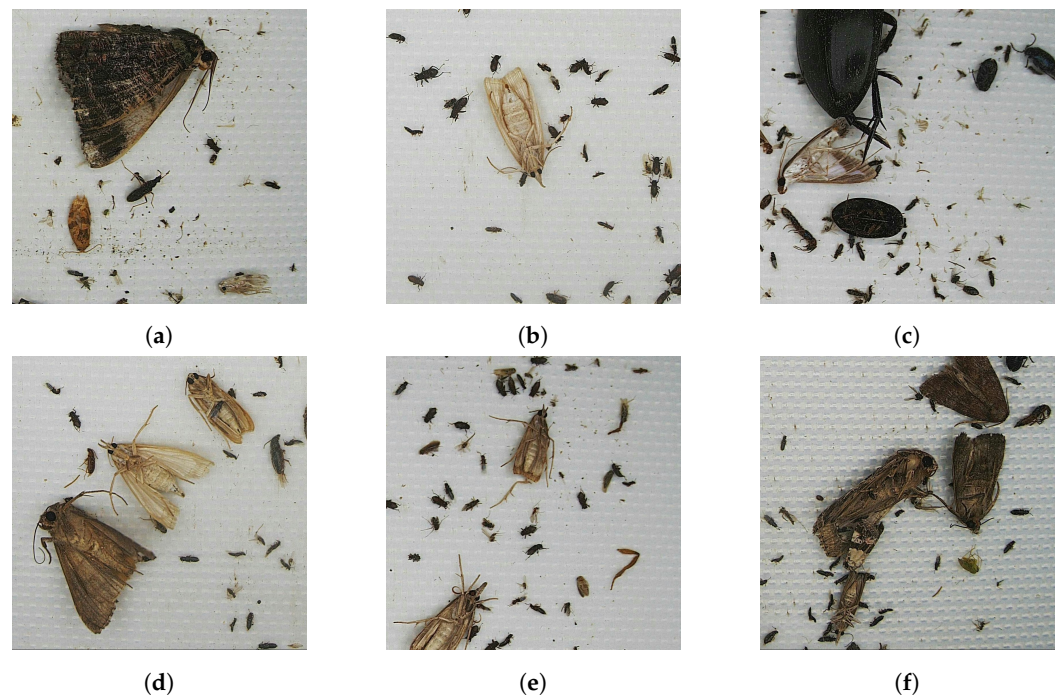


Figure 1. A selection of raw images from the datasets. (a) Sample (DaMingShen); (b) Sample (ErHua); (c) Sample (JinGui); (d) Sample (Mixed pests1); (e) Sample (Mixed pests2); and (f) Sample (Mixed pests3).

As depicted in Figure 2, the images, sourced from the IP102 dataset, encapsulate pests across varied species and contexts. Procured via diverse avenues such as internet searches, agricultural field surveys, and laboratory samplings, the dataset ensures both the authenticity and a varied representation of the species.

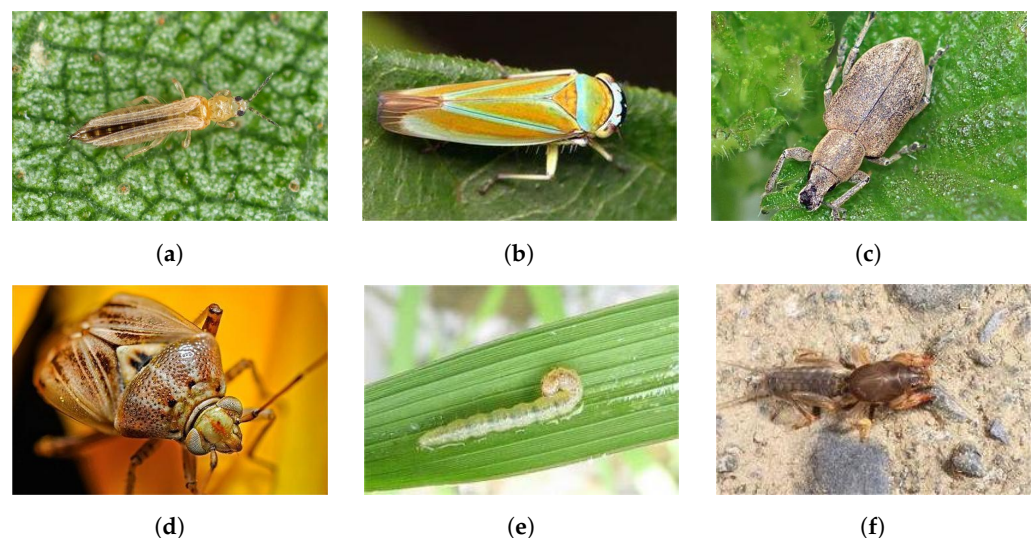


Figure 2. A selection of raw images from the IP102. (a) Sample (thrips); (b) Sample (Cicadellidae); (c) Sample (JinGui); (d) Sample (Miridae); (e) Sample (rice leaf roller); and (f) Sample (mole cricket).

The decision to incorporate IP102 was driven by multiple considerations. Its vast species diversity presents our algorithm with a rigorous testing environment, thereby pushing the boundaries of our method's adaptability across different agricultural settings.

Furthermore, the superior resolution and image quality of IP102 set a high benchmark, allowing us to gauge our model's efficiency in tasks demanding precision. This expansive validation on IP102 accentuates the generalization prowess of our model, attesting its consistent performance across multiple datasets and practical scenarios.

In essence, the IP102 dataset furnishes us with an invaluable benchmarking platform, enabling a multifaceted evaluation of the PestLite algorithm. We are optimistic that juxtaposing our results with IP102 will further substantiate the efficacy and the broad applicability of our model.

2.3. YOLO and Its Improvements

2.3.1. The YOLOv5 Framework

The YOLO (You Only Look Once) series, represented by Redmon's pioneering work, has revolutionized object detection via its fast and accurate performance. Starting with YOLOv1, the framework quickly evolved with YOLOv2 (YOLO9000) and YOLOv3, continually advancing the state of the art by simplifying object detection into a single regression task. This shift significantly reduced the complexity of object detection, previously characterized by separate stages of the region's proposal and classification. With YOLOv4, the series introduced even more efficient mechanisms such as weighted residual connections and cross-stage partial connections, improving both speed and accuracy. However, it is YOLOv5 that has become particularly prominent for its deployment readiness and suitability for real-time applications. YOLOv5's architecture, featuring automatic optimization and scalable layers, is adept at handling a variety of real-world detection scenarios efficiently. While later iterations such as YOLOv7 and YOLOv8 have continued to refine the model's capabilities, YOLOv5 remains a preferred choice for its balance between speed and performance, especially in real-time applications. YOLOv5 stands out in the YOLO series for its deployment capabilities in real-time applications. Its architecture is designed for ease of use with automatic optimization and scalability, efficiently handling a variety of real-world detection scenarios [18]. Notably, its convenient deployability has facilitated its rapid integration into detection tasks, especially on resource-constrained platforms. Although subsequent versions such as YOLOv7 and YOLOv8 offer further refined architectures, YOLOv5 remains the preferred choice due to the admirable balance it strikes between speed and performance. This has allowed it to excel in various applications, notably in the agricultural sector. For agricultural practices, especially in pest detection, YOLOv5 has emerged as the technology of choice [19]. Its structure and algorithmic design, as illustrated in Figure 3, provide a solid foundation for real-time detection and high precision. An in-depth overview of the YOLOv5 architecture follows.

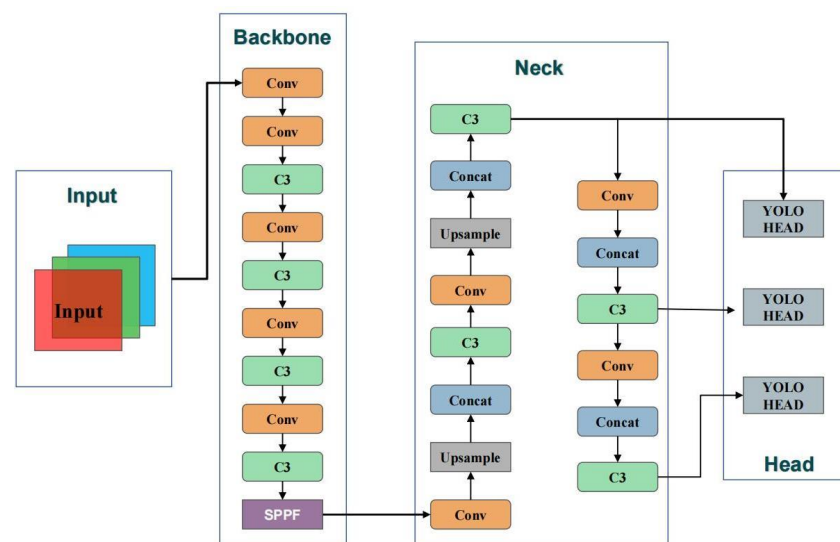


Figure 3. Architecture of the YOLOv5 model.

YOLOv5 processes image inputs across a spectrum of resolutions, tailoring itself to diverse contexts. Its configuration encompasses parameters set for class numbers, model depth, and channel multipliers per layer. Moreover, it establishes multiple anchor point clusters specific to different feature scales. The backbone of YOLOv5, dedicated to image feature extraction, is structured around an array of convolutional layers. These layers capture features ranging from elementary to intricate. It initiates by halving the dimension of the feature map via a convolutional layer, sequentially enhancing channel counts, and amalgamating features via the C3 module—a core unit of YOLOv5. This module integrates an array of convolutional layers and residual connections, optimizing feature extraction and consolidation. With an aim to capture features at varied scales, the backbone embeds the SPPF module, which elevates feature representation via max pooling across assorted sizes. The neck phase fuses feature maps of different scales, fortifying the model's capability to identify targets of varied sizes. YOLOv5 augments resolution via the Upsample module and conjoins feature maps of distinct scales via the Concat module. Conclusively, the head segment leverages integrated features for object categorization and positioning. Initially, this section refines features via a convolutional layer, subsequently amalgamating them with the C3 module. Accommodating multi-scale detection, the head integrates three distinct output layers. The Detect module eventually transmutes features into target coordinates, categorizations, and confidence scores, underpinning YOLOv5's prowess in rendering precise object detection across multifarious scenarios.

2.3.2. PestLite

Responding to the specific requirements and challenges posed by tasks such as high-density insect pest detection, we designed and refined an algorithm rooted in the YOLOv5 v6.1 framework. Our distinctive design, christened PestLite, maintains the original Backbone and Head from YOLOv5, but seamlessly introduces several innovative modules and techniques, encompassing C3ECA, CARAFE, Involution, and MTSPPF. With these advancements, PestLite, our proprietary model, showcases unparalleled performance, especially in complex scenarios like insect pest detection.

PestLite incorporates several novel modules and techniques aimed at improving pest detection in high-density scenarios and enhances the capability for real-time detection applications. A key innovation is the introduction of the MTSPPF, designed to capture features across different scales with increased precision. Unlike traditional spatial pyramid pooling structures, MTSPPF employs a multi-level structure to capture richer and more detailed feature information across different levels, this might augment the model's prowess in feature discernment and fortify its stability in multifaceted scenarios. Additionally, the integration of Involution allows for an adaptive weight adjustment mechanism, dynamically tuning the receptive field based on the image content, which may improve recognition accuracy for small target pests. The model also incorporates the Efficient Channel Attention (ECA) mechanism to potentially enhance focus on critical features, given the diverse morphologies and sizes of pests. This addition aims to enhance detection accuracy and robustness by accurately capturing and differentiating various pest features [20]. Lastly, as illustrated in Figure 4, the replacement of the traditional upsampling pattern with the CARAFE operator is aimed at preserving critical feature information during upsampling, which could be beneficial in better capturing features of small target pests and enhancing detection accuracy.

After improvements, PestLite effectively addresses the challenges encountered in high-density insect pest detection scenarios and significantly enhances real-time detection efficiency. Through algorithmic optimization, PestLite realizes superior detection precision even as it scales down the number of parameters. Thus, PestLite is better suited to meet the demands of high-precision real-time insect pest detection in agricultural production, offering an efficient solution.

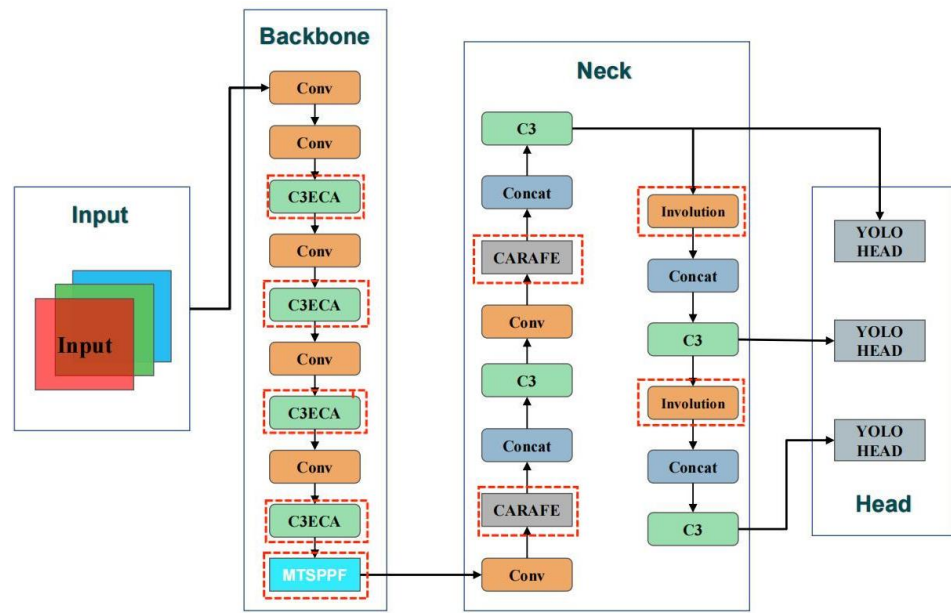


Figure 4. Architecture of the PestLite model.

2.4. Improvement of YOLOv5s Algorithm

2.4.1. Efficient Channel Attention Mechanism

Channel attention mechanisms play a vital role in emphasizing or suppressing specific channels based on their significance in representing the input data. The ECA mechanism, in particular, provides a computationally efficient method to gauge this significance without incurring a substantial computational cost [21]. At its core, the ECA module introduces a nimble technique to dynamically adjust channel-wise features according to their content. Remarkably, it achieves this without resorting to dimensionality reduction, ensuring a balance between efficiency and performance.

Given an input feature map $X \in \mathbb{R}^{H \times W \times C}$, where H , W , and C represent the height, width, and number of channels, respectively, the channel descriptor is first acquired using Global Average Pooling (GAP):

$$z_c = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W X_{ijc} \quad (1)$$

This results in a descriptor $z \in \mathbb{R}^C$. Subsequently, to determine the adaptive kernel size for the 1D convolution, denoted as $k(C)$, a simple linear relationship is postulated:

$$k(C) = \alpha C + \beta \quad (2)$$

Here, α and β are parameters that regulate the adaptive kernel size based on the number of channels C . They are crucial for tuning the channel-wise emphasis dynamically.

With this kernel, the 1D convolution generates channel weights, which are normalized using the softmax function:

$$w = \text{Conv1D}(z, k(C)) \quad (3)$$

$$\hat{w}_c = \frac{\exp(w_c)}{\sum_{c'=1}^C \exp(w_{c'})} \quad (4)$$

$$Y_{ijc} = \hat{w}_c X_{ijc} \quad (5)$$

This mechanism amplifies channels deemed crucial while concurrently suppressing the less significant ones. As depicted in Figure 5, the input feature map undergoes global average pooling, transforming the matrix from $[h, w, c]$ to $[1, 1, c]$. An adjustable

one-dimensional convolution kernel size, termed as *kernel_size*, is derived from the channel count of the feature map. Utilizing this *kernel_size*, the weights for individual channels are identified. These weights are subsequently applied element-wise to the input feature map, resulting in the weighted feature map.

In summary, the ECA mechanism stands out for its adaptive recalibration of channel-wise feature responses. By focusing on inter-channel relationships without the computational burden of traditional attention mechanisms, ECA efficiently emphasizes informative features and suppresses less relevant ones. Its unique adaptive kernel sizing captures varying dependencies, ensuring the model's heightened sensitivity to crucial features. This selective attention, as a result, significantly bolsters the model's performance, leading to notable improvements in metrics such as precision (P) and mAP50. The integration of ECA, therefore, not only refines the algorithm's attention to pivotal details, but also enhances its overall detection efficacy.

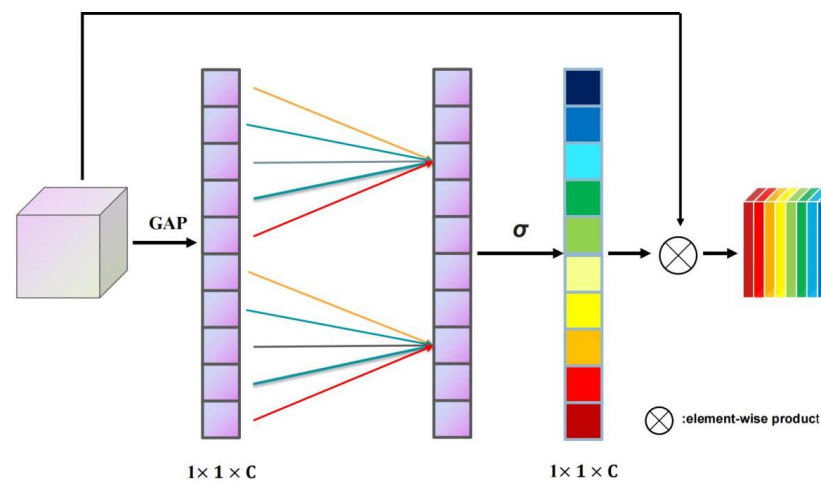


Figure 5. Structure and flow of the ECA.

2.4.2. The Advent of the Lightweight Neural Operator: Involution

Involution, as a unique neural network operation, dynamically generates a specific convolutional kernel for each spatial position, contrasting with the spatial invariance of traditional convolution. This operation effectively reallocates computational resources to the spatial positions where they are most needed, allowing for an adaptive mechanism based on the content of the input image. Through dynamic kernel generation, it facilitates fine-grained processing of input data, providing a new optimization path for deep learning models, especially in situations where computational resources are constrained. Involution paves a new efficient path for network design, enhancing model performance without increasing computational complexity.

The uniqueness of Involution lies in its channel-agnostic and spatial-specific characteristics. Essentially, it reallocates computational resources at a micro-level, aligning them to positions where significant performance improvements can be attained. This design contrasts with conventional convolution operations which are spatial-agnostic and channel-specific, thereby bringing a new perspective to the allocation of computational resources within neural networks [22]. As depicted in Figure 6, the Involution mechanism efficiently aggregates contextual information over an expansive spatial domain. By emphasizing key visual cues in the image's spatial realm, it achieves a leaner parameter structure and expedites convergence.

Mathematically, the Involution process for an input feature map F can be expressed as:

$$I(F) = \sum_{i=1}^N K(F_i) * F_i \quad (6)$$

In this context, $I(F)$ signifies the result post-involution operation. The convolution kernel, dynamically formed from the input feature map, is represented by K . The i -th channel or segment of the input feature map F is denoted by F_i , while the $*$ operation corresponds to the convolution process. For each channel F_i of the input feature map, the Involution mechanism generates a corresponding kernel. This dynamic kernel generation is further detailed as:

$$K(F_i) = W_k * F_i + b_k \quad (7)$$

where W_k and b_k are the learned weights and biases, respectively, associated with the kernel generation process. The generated kernel $K(F_i)$ is then convolved with the input feature map slice F_i to produce the Involution-transformed output.

Involution, by providing a specific convolutional kernel for each spatial position, not only enhances detection accuracy but also reduces the model's parameter count, making it more efficient compared to traditional convolutions. This innovation has piqued the interest of deep learning researchers and practitioners, showcasing outstanding performance across various computer vision tasks.

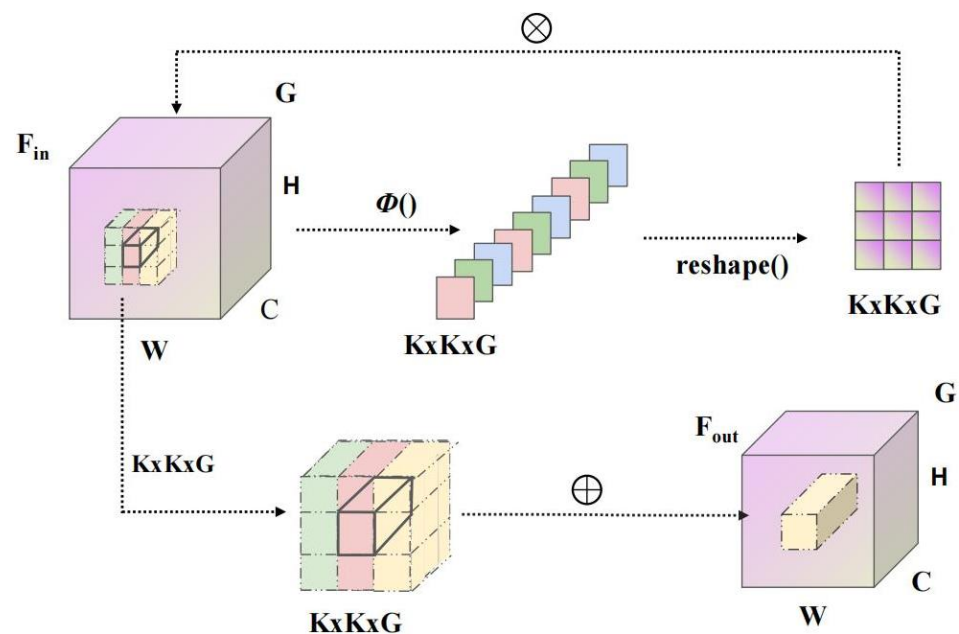


Figure 6. Fundamentals and operational mechanics of Involution.

2.4.3. A Unique Upsampling Operation: CARAFE

Content-Aware ReAssembly of FEatures (CARAFE) is a unique upsampling technique in neural networks that introduces content awareness by adaptively predicting the upsampling kernel based on the feature map's content. Diverging from conventional methods, this mechanism brings to the fore two primary components: the Kernel Prediction Module and the Content-Aware Reassembly Module [23]. As illustrated in Figure 7, CARAFE not only boasts an expansive receptive field, but also astutely leverages input features for reassembly, ensuring a harmonious and efficient upsampling process.

Given an input feature map X with dimensions $C \times H \times W$, aiming to upscale it by a factor of σ , the resultant feature map X' spans $C \times \sigma H \times \sigma W$. A pivotal aspect of CARAFE's operation involves the relationship between a target position $l' = (i', j')$ in X' and its analogous source position $l = (i, j)$ in X , encapsulated as:

$$i = \left\lfloor \frac{i'}{\sigma} \right\rfloor, \quad j = \left\lfloor \frac{j'}{\sigma} \right\rfloor \quad (8)$$

Designating $N(X_l, k)$ as the $k \times k$ subregion centered at position l in X , the Kernel Prediction Module ψ exploits this subregion to deduce the reassembly kernel $W_{l'}$ for each

target position l' , and, combined with the Content-Aware Reassembly Module ϕ , produces the upsampled feature at position l' :

$$W_{l'} = \psi(N(X_l, k_{\text{encoder}})), \quad (9)$$

$$X_{l''} = \phi(N(X_l, k_{\text{up}}), W_{l'}), \quad (10)$$

$$\psi(x) = x * w + b \quad (11)$$

Here, w symbolizes the weights and b typifies the biases, both refined during training.

Incorporating CARAFE into architectures like YOLOv5 supplants traditional nearest-neighbor interpolation methods, reaping the rewards of content-aware upsampling. Specifically, within the YOLOv5 framework, the CARAFE module is affixed post the P3 and P4 pathways, effectuating a two-fold upsampling. The parameter count is articulated as:

$$\text{Parameters} = 2(C_{\text{in}} + 1)C_m + 2(C_m k_{\text{encoder}}^2 \sigma^2 + 1)\sigma^2 k_{\text{up}}^2 + 2\sigma^2 k_{\text{up}}^2 C_i \quad (12)$$

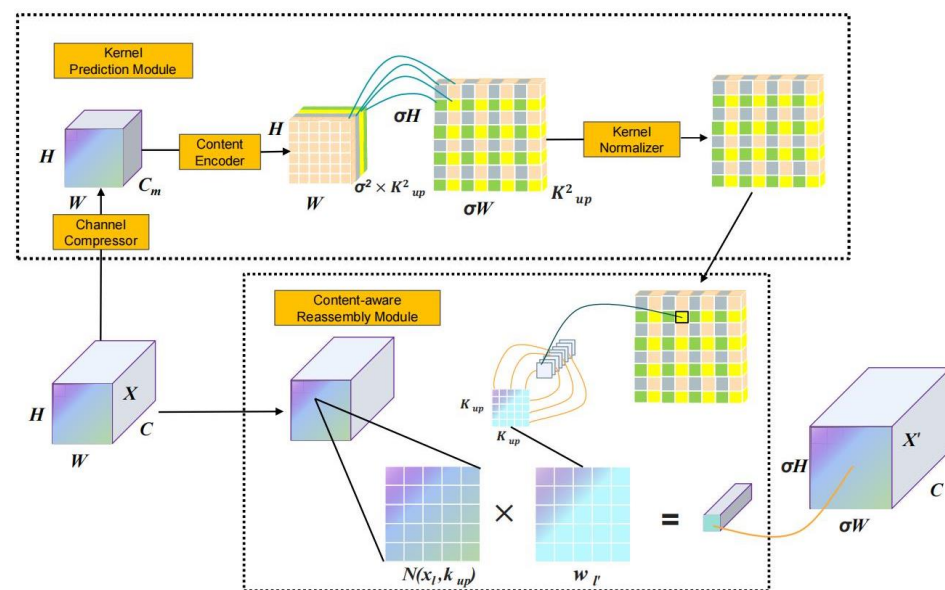


Figure 7. Structure and flow of the CARAFE.

In conclusion, CARAFE, through its adaptive upsampling approach, facilitates an efficient and content-aware feature reassembly. It demonstrates significant improvements in detection accuracy, meeting the demands for precise pest detection in contemporary agricultural production. Furthermore, its effectiveness as a neural network upsampling technique is evident across various applications.

2.4.4. Innovative Multi-Level Feature Integration: MTSPPF (Multi-Scale Spatial Pyramid Pooling)

Spatial Pyramid Pooling-Fast (SPPF) emerges as a crucial advancement in image processing, extending beyond the boundaries of the traditional bag-of-words model. Through the strategic segmentation of image features across various granularities, from the coarser to the more detailed, SPPF integrates these features to deliver a more accurate representation of the original image. A distinguishing trait of SPPF is its inherent adaptability, capable of seamlessly processing images of diverse sizes during the prediction phase while promoting multi-scale training. This dual functionality results in significant scale invariance, offering a formidable solution against the persistent challenges of overfitting in deep learning architectures [24].

Diving deeper into the mechanics of SPPF, the procedure can generally be categorized into three pivotal phases: feature segmentation, max pooling, and descriptor amalgamation. Beginning with an image feature map F of dimensions $H \times W \times C$, for each specified

scale s (typically where $s \in \{1, 2, 4\}$), F is fragmented into $s \times s$ grids. Each of these grids, of dimensions $\left\lceil \frac{H}{s} \right\rceil \times \left\lceil \frac{W}{s} \right\rceil \times C$, is subjected to max-pooling. This operation yields an output of $1 \times 1 \times C$, producing descriptors of dimensions $s \times s \times C$. In the final phase, by merging descriptors from all scales, we obtain the definitive SPPF feature map, symbolized as $F_{\text{sppf}} = [F_{s1}, F_{s2}, \dots, F_{sn}]$.

Yet, SPPF faces challenges in terms of computational efficiency and the management of highly varied feature scales. These issues pave the way for the unveiling of our groundbreaking strategy: MTSPPF, which seeks to rectify the inherent limitations of the conventional SPPF.

Object detection tasks are frequently confronted with the intricacies of discerning objects that vary in size and shape. The historical tension between model complexity and performance has long been a focal point. Against this backdrop, our MTSPPF module emerges as a noteworthy development. It has been scrupulously designed to capture multi-scale features by utilizing diverse spatial pooling scales, thus ensuring an enriched extraction of contextual information. Furthermore, the essence of MTSPPF is anchored in its capacity to assimilate features spanning a wide array of scales. By deploying a series of max-pooling layers, each uniquely scaled, it adeptly captures expansive contexts derived from larger pooling dimensions and delves into the minutiae offered by the smaller scales. This operational facet is visually elucidated in Figure 8.

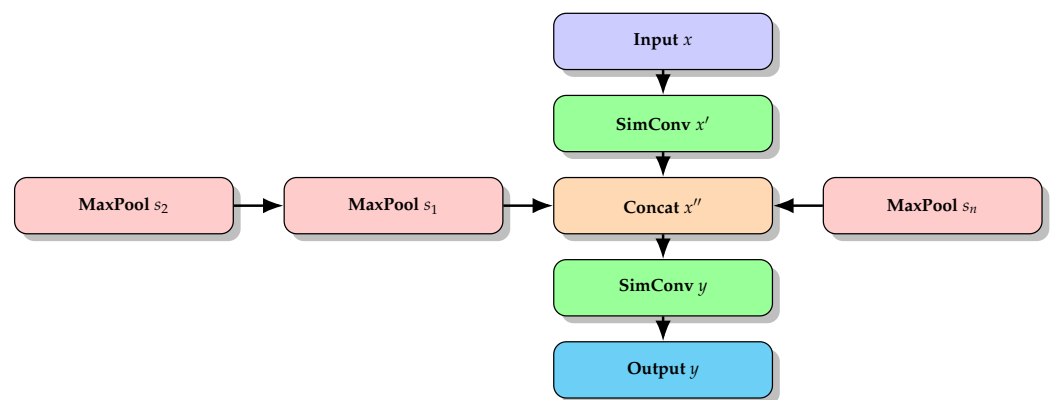


Figure 8. Structure and flow of the MTSPPF.

Integral to the MTSPPF module is its foundational convolution unit, SimConv. This component encapsulates a combination of two-dimensional convolution, batch normalization, and the ReLU activation mechanism, creating a unified mechanism that facilitates feature transformation, all the while optimizing computational efficiency. The mathematical articulation of this process can be expressed as:

$$\text{SimConv}(x) = \text{ReLU}(\text{BN}(W * x + b)) \quad (13)$$

Within this formulation, $*$ signifies convolution. Moreover, W and b outline the convolution kernel and correspondingly, the bias, while BN symbolizes batch normalization.

Detailing its operational flow, the MTSPPF module commences with an input feature map x . This input undergoes an intermediary feature extraction via SimConv, resulting in x' . Subsequent to this, a suite of max-pooling operations is applied to x' , each uniquely defined by its pooling size s_i . The resultant outputs from these pooling operations are then amalgamated, culminating in x'' . As a concluding step, x'' undergoes another transformation via SimConv, producing the final output y , as represented in Equations (14)–(17).

$$x' = \text{SimConv}(x) \quad (14)$$

$$p_i(x') = \text{MaxPool}_{s_i}(x') \quad (15)$$

$$x'' = \text{Concatenate}(p_1(x'), p_2(x'), \dots) \quad (16)$$

$$y = \text{SimConv}(x'') \quad (17)$$

The uniqueness of MTSPPF lies in its ability to effectively balance model lightness with efficiency. By reducing the number of parameters, it offers a lightweight solution for real-time applications. Notably, this reduction in parameters does not compromise its detection accuracy but, on the contrary, achieves greater recognition precision. This capability to maintain lightness while enhancing detection accuracy firmly establishes MTSPPF's pivotal role in modern object detection strategies.

2.5. Criteria for Assessing Detection Precision

The efficacy of the detection method is evaluated using three core metrics: detection precision, number of parameters, and computational complexity. Detection precision assesses the algorithm's ability to correctly identify and classify entities [25]. Herein, we employ the metric of mean average precision (mAP) in conjunction with an Intersection over Union (IoU) threshold fixed at 0.5 for our assessment standards. In mathematical terms, mAP can be articulated as:

$$mAP = \frac{1}{n} \sum_{i=1}^n AP(i) \quad (18)$$

where n represents the total number of categories. Considering our dataset comprises nine distinct classes, we assign n to 9. For every individual class, the average precision (AP) is determined by averaging the precision over varied recall thresholds:

$$P = \frac{TP}{TP + FP} \quad (19)$$

$$R = \frac{TP}{FN + TP} \quad (20)$$

Within the equations presented above, P denotes precision, which quantifies the proportion of accurate predictions among all instances classified as positive. Concurrently, R embodies recall, capturing the proportion of actual positive cases that were correctly identified by the model. These metrics are underpinned by the classifications of True Positives (TP), False Positives (FP), and False Negatives (FN). To elucidate, True Positives (TP) are instances where the model correctly predicts the positive class, aligning with the actual class. Conversely, False Positives (FP) occur when the model inaccurately predicts a positive class, despite the actual class being negative. This distinction is pivotal in model evaluation. While TP represents instances of accurate positive predictions, FP indicates instances where the model erroneously identified a condition or event as positive. Achieving an optimal balance between these classifications is paramount, as an excess of false positives can lead to misguided interventions, whereas a deficiency might result in overlooking critical events or conditions.

$$AP = \int_0^1 P(R) dR \quad (21)$$

Computational complexity is assessed in terms of Giga Floating-point Operations Per Second (GFLOPs), which reflect the model's computational demands and efficiency. This metric is especially important for applications that require real-time processing or are deployed on devices with computational constraints. GFLOPs represents the billion floating-point operations performed per second and is a primary metric for assessing the model's computational capability. The number of parameters in the model provides a measure of its size and capacity, directly correlating with the computational and storage requirements of the model [26].

2.6. Frames per Second

Frames Per Second (FPS) has emerged as a pivotal metric in the evaluation of object detection algorithms' processing velocity, especially in scenarios that demand real-time detection capabilities [27]. The efficacy of these algorithms is not solely dependent on their accuracy metrics but is also significantly influenced by their computational throughput. An augmented FPS value is indicative of an algorithm's enhanced capability to process a substantial volume of images within a singular second, thereby enabling more rapid recognition and adjustment to the dynamic alterations in the scene [28]. In contexts like real-time surveillance, autonomous vehicular navigation, and pest detection, it is imperative for object detection algorithms to swiftly process sequential video feeds. In these scenarios, a superior FPS is indispensable as it guarantees timely detection and interaction with objects within ever-changing environments. For instance, in the domain of pest detection, the rapid identification of pests is essential, necessitating an algorithm adept in brisk image processing. In essence, FPS, serving as a benchmark for the speed of object detection algorithms, not only reflects their image processing prowess but also directly influences their real-time applicability and user experience in practical deployments. Consequently, in the design and assessment of object detection algorithms, the enhancement of FPS, ensuring efficient algorithmic functionality, is as pivotal as the consideration of accuracy.

3. Results and Discussion

3.1. Experiment and Parameter Setting

To ensure the reliability and reproducibility of our experiments, we meticulously set up our experimental environment and carefully selected model parameters and hyperparameters.

3.1.1. Experimental Environment

Our experiments were conducted on a system running the Ubuntu 20.04 operating system, powered by an Intel(R) Xeon(R) Gold 6330 processor. For accelerated computations, we utilized an NVIDIA RTX 3090 graphics card. Our software stack primarily comprised Python version 3.8, with the neural network models implemented and trained using PyTorch version 1.10.0. Additionally, we used CUDA version 11.3 to leverage GPU acceleration. The detailed environment setup is presented in Table 1.

Table 1. Environment Setup for Experiments.

Experimental Component	Version/Model
OS	ubuntu20.04
Graphics Card	RTX 3090
Processor	Intel(R) Xeon(R) Gold 6330
Python Version	3.8
CUDA Version	11.3
PyTorch Version	1.10.0

3.1.2. Model Training Parameters

For the training process, we standardized the input image dimensions to 640×640 . We utilized a batch size of 36 and trained the model for a total of 500 epochs. Data augmentation, a crucial step for improving model generalization, was performed using the Mosaic method. Specific details regarding these parameters can be found in Table 2.

3.1.3. Hyperparameter Settings

In the optimization process of our deep learning model, Stochastic Gradient Descent (SGD) was chosen as the primary optimization technique. To enhance the efficiency and stability of the model during parameter updates, a momentum parameter was set to 0.937. This approach not only accelerated the convergence of SGD but also helped in reducing oscillations within the parameter space. Additionally, considering the risk of overfitting, a modest weight decay coefficient of 5×10^{-4} was implemented to improve the model's

generalization capabilities across different datasets. In this study, a dynamic strategy for setting the learning rate was employed. Initially, the learning rate was set at 1×10^{-2} , and a learning rate decay mechanism was adopted, ultimately reducing it to 1×10^{-4} . These settings aimed to balance the speed and stability of learning, thus aiding in the selection of the optimal model.

Table 2. Parameters for Model Training.

Parameter Label	Selected Configuration
Number of Epochs	500
Image Dimensions	640×640
Batch Size	36
Data Augmentation Method	Mosaic

Beyond these optimization strategies, we also implemented a range of data augmentation techniques, including but not limited to image scaling, horizontal image flipping, mosaic effects, and image translation. The introduction of these techniques was intended to simulate various changes that might occur in real-world settings, thereby enhancing the model's adaptability and robustness to new scenarios. The probabilities of applying these augmentations during training were respectively 0.5, 0.5, 1.0, and 0.1. Through this approach, we were able to maintain model performance while augmenting its capability to handle unfamiliar data. The selected hyperparameters for the training are elaborated in Table 3.

Table 3. Hyperparameters for Training.

Hyperparameter	Setup
Optimizer	SGD
Initial Learning Rate (Lr0)	0.01
Final Learning Rate (Lrf)	0.01
Momentum	0.937
Coefficient of Weight Deterioration	0.0005
Image Scale	0.5
Image Flip Left-Right	0.5
Image Translation	0.1
Mosaic	1.0

3.2. Examination of Experiment Results

This research rigorously explores four essential experimental objectives that underpin the study. Firstly, it seeks to validate different mechanisms, ensuring their effectiveness in the given application. Secondly, the study evaluates the relative performance of the improved algorithm, gauging its advancements in comparison to existing benchmarks. Thirdly, an in-depth comparative analysis of various attention mechanisms is undertaken, with the intent to identify their individual merits and potential avenues for enhancement. Lastly, the research delves into comparison experiments for SPPF and MTSPPF, demonstrating the innovation and effectiveness of our proposed MTSPPF. By juxtaposing MTSPPF with the traditional SPPF, we aim to highlight the distinct advantages and improvements our approach brings, emphasizing its potential for widespread applications and setting a new benchmark in the field.

1. **Validation of Different Mechanisms:** The validation process involves examining the efficacy of the proposed improvements. The outcomes indicate that the improved method achieved significant improvements in multiple key metrics compared to the baseline approach.
2. **Comparative Analysis of Various Attention Mechanisms:** We conducted an in-depth investigation into the performance of various attention mechanisms in the pest detection task. Through a series of comparative experiments, certain mechanisms exhibited

exceptional performance in specific scenarios, such as detecting small or overlapping targets. Identifying which mechanisms are most suitable for the current task provided insights into the subtle differences and complexities of different attention models in this research context.

3. Comparison Experiments for SPPF and MTSPPF: In addition to the above objectives, we conducted comparison experiments to evaluate the performance of Single-Level Spatial Pyramid Pooling (SPPF) and Multi-Level Spatial Pyramid Pooling (MTSPPF). This task involved contrasting their performance and efficiency, shedding light on the potential benefits of MTSPPF in enhancing object detection performance while maintaining computational efficiency.
4. Relative Performance Evaluation of the Improved Algorithm: To gauge the efficacy of our advanced algorithmic proposal, we engaged in an in-depth comparison with the baseline method. The results from this section reveal that, in comparison to the baseline, our approach demonstrates notable improvements in accuracy, speed, and robustness.

3.2.1. Validation of Different Mechanisms

To validate the effectiveness of the proposed modifications on the YOLOv5 architecture, we conducted a series of experiments [29]. The detailed experimental results and analysis are presented below. We selected YOLOv5 as the baseline model and introduced four distinct modifications: MTSPPF, Involution, ECA, and CARAFE. All experiments were conducted on the same dataset to ensure consistency and comparability of the results. As inferred from Table 4, we deduce the subsequent observations:

Table 4. Assessment Test of Various Mechanisms.

Procedure	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)
YOLOv5 (baseline)	7.03	15.8	91.3	81.9	87.9
YOLOv5+ECA	7.03	15.8	93.1	80.8	90.1
YOLOv5+Involution	6.31	14.9	88.7	83.3	89.9
YOLOv5+CARAFE	7.16	16.3	88.3	82.4	88.5
YOLOv5+MTSPPF	6.64	15.5	93.7	78.1	89.0

Note: GFLOPs refers to Giga Floating Point Operations Per Second. P represents Precision. R stands for Recall. mAP50% denotes the mean Average Precision (mAP) evaluation metric with an Intersection over Union (IoU) threshold set at 0.5.

The YOLOv5 with the Efficient Channel Attention (ECA) mechanism achieves a significant increase in mAP from 87.9% to 90.1%, even without an increase in the number of parameters and computational requirements (both at 7.03 M and 15.8 GFLOPs). This improvement is attributed to the ECA mechanism's effective capture of dependencies between channels, which enhances the model's ability to capture key features, thereby improving the accuracy of detection. Incorporating the Involution mechanism into YOLOv5, the model experienced a reduction in parameters to 6.31 M while achieving an increase in mAP to 89.9%. This reduction in parameters facilitates a lightweight model that maintains high performance, underscoring the mechanism's dual benefits of efficiency and effectiveness. The introduction of CARAFE, which substitutes the conventional sampling pattern, leads to an improvement in the model's feature extraction capabilities, as evidenced by the increase in mAP to 88.5%. This enhancement is particularly significant for pest detection in complex scenarios, where CARAFE's advanced upsampling ability plays a crucial role. Lastly, the Multi-Level Spatial Pyramid Pooling (MTSPPF) mechanism, while decreasing the number of parameters to 6.64 M, accomplished a considerable rise in mAP to 89.0%. This indicates that multi-level feature aggregation aids in boosting the model's comprehensive detection capabilities, particularly critical for tasks requiring the detection of pests across various sizes and shapes.

Overall, these mechanisms demonstrate how enhancements in model efficiency can concurrently amplify precision in pest detection. Especially in real-time pest detection

applications with strict requirements for model size and computational efficiency, these improvements offer new avenues for achieving effective and accurate pest monitoring.

3.2.2. Contrast Tests for Various Attention Procedures

For a thorough assessment of the ECA attention mechanism in object detection tasks, we conducted experiments comparing ECA with several other renowned attention mechanisms [30]. These attention mechanisms were individually integrated into the YOLOv5 framework, and their performance metrics were juxtaposed with the unmodified YOLOv5 (baseline). The experimental results are tabulated below: (Table 5).

Table 5. Assessment Trial of Various Attention Procedures.

Procedure	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)
YOLOv5+CA	7.05	15.9	90.2	81.4	89.3
YOLOv5+CBAM	7.05	15.9	91.6	82.0	89.9
YOLOv5+SE	7.05	15.8	88.7	82.5	89.3
YOLOv5+ECA	7.03	15.8	93.1	80.8	90.1

Note: GFLOPs refers to Giga Floating Point Operations Per Second. P represents Precision. R stands for Recall. mAP50% denotes the mean Average Precision (mAP) evaluation metric with an Intersection over Union (IoU) threshold set at 0.5.

Table 5 provides a comparative analysis of YOLOv5 when integrated with various attention mechanisms. Among these, YOLOv5, enhanced with the ECA attention mechanism (YOLOv5+ECA), distinctly stands out. It demonstrates a notable precision of 93.1%, which is the highest among all models, indicating its adeptness in correctly identifying positive samples. Furthermore, this model achieves the top mAP50 score of 90.1%, signifying its superior accuracy in detecting objects with a 50% intersection over union.

Interestingly, while enhancing the model's precision and accuracy, the ECA mechanism does not inflate the model's size. It maintains a competitive parameter count of 7.03 M, comparable to other configurations. The consistent computational cost across models, especially with YOLOv5+ECA at 15.8 GFLOPs, suggests that integrating the ECA attention mechanism offers enhanced performance without compromising efficiency. Thus, given these merits, the ECA mechanism emerges as the most suitable choice, seamlessly amalgamating high precision, accuracy, and resource efficiency.

3.2.3. Comparison Experiments for SPPF and MTSPPF

The task involves contrasting Single-Level Spatial Pyramid Pooling (SPPF) and Multi-Level Spatial Pyramid Pooling (MTSPPF) in terms of performance and efficiency. Through this comparison, insights are gained into the potential benefits of MTSPPF in enhancing object detection performance while maintaining computational efficiency, as illustrated in the provided table, Table 6.

Table 6. Performance Metrics-Based Analysis between SPPF and MTSPPF.

Framework	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)
SPPF	7.03	15.8	91.3	81.9	87.9
MTSPPF	6.64	15.5	93.7	78.1	89.0

Note: GFLOPs refers to Giga Floating Point Operations Per Second. P represents Precision. R stands for Recall. mAP50% denotes the mean Average Precision (mAP) evaluation metric with an Intersection over Union (IoU) threshold set at 0.5.

The performance comparison between Spatial Pyramid Pooling-Fast (SPPF) and Multi-Level Spatial Pyramid Pooling (MTSPPF) is presented in Table 6. A salient advantage of MTSPPF is its efficient parameterization, reducing the model size from 7.03 M to a lean 6.64 M. Such optimization not only highlights the architectural dexterity of MTSPPF but also showcases its suitability for deployment in resource-limited environments—crucial for

real-time applications. While there is a minor decline in recall for MTSPPF at 78.1% compared to SPPF's 81.9%, MTSPPF surpasses SPPF in terms of precision (93.7% against 91.3%). Notably, the mAP50 metric—which serves as a robust measure of the balance between precision and recall—rises to 89.0% from SPPF's 87.9%. These findings underscore MTSPPF's enhanced precision and harmonized mAP50, positioning it as a potent solution for refining the object detection performance without compromising the computational efficiency.

3.2.4. Relative Performance Evaluation of the Improved Algorithm

Compared to the baseline, delineated in Table 7, the YOLOv5 model, integrated with Involution, ECA, MTSPPF, and CARAFE and named as PestLite, exhibited significant enhancements in the performance metrics. Specifically:

Table 7. Performance Metrics-Based Algorithm Analysis.

Model	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)	FPS
Baseline	7.03	15.8	91.3	81.9	87.9	109.46
PestLite	6.09	15.5	91.9	82.7	90.7	112.56

Note: GFLOPs refers to Giga Floating Point Operations Per Second. P represents Precision. R stands for Recall. mAP50% denotes the mean Average Precision (mAP) evaluation metric with an Intersection over Union (IoU) threshold set at 0.5.

In terms of Parameters (M), our enhanced model, PestLite, exhibits a parameter count of 6.09 M, which is notably less than the baseline's 7.03 M. This decrease in parameters is largely due to the incorporation of cutting-edge modules like Involution and MTSPPF, which are inherently more parameter efficient.

In the evaluation of Precision (P) and Recall (R), the PestLite model demonstrates a precision rate of 91.9%, closely mirroring the 91.3% precision of the baseline. More notably, there is an improvement in recall when utilizing PestLite, rising from the baseline's 81.9% to 82.7%. This increase in recall can be largely ascribed to the incorporation of the ECA module, which, by effectively modulating channel weights, sharpens the model's ability to discern essential feature channels, resulting in enhanced recall.

For the mAP50 (%) metric, there is a notable improvement from 87.9% in the baseline to 90.7% in PestLite. This advancement is largely credited to the adoption of the CARAFE module. As an advanced feature reconstruction methodology, CARAFE learns to optimally reconstruct features, ensuring the better retention of details for small objects. This inherent capability strengthens the model's prowess in identifying smaller targets by preserving their intricate information.

In our endeavor to emphasize the efficacy of PestLite in comparison to the baseline algorithm, we present three distinct graphical analyses over training epochs in Figure 9. Specifically, Figure 9a–c consecutively illustrates the comparative mAP, precision, and recall curves of the original and improved algorithms. Across all these metrics, once stabilized, PestLite consistently outperforms the baseline algorithm on the test data. This comparative visualization underscores PestLite's enhanced capabilities in terms of accuracy and reliability.

Frames Per Second (FPS) represents the number of images processed per second and reflects the processing speed of a model. It is a critical indicator for measuring the real-time inference capability. As shown in the table, the improved PestLite model can achieve up to 112.56 FPS. This demonstrates that PestLite effectively balances detection accuracy with speed, making the model suitable for real-time detection tasks in the complex conditions of agricultural pest monitoring.

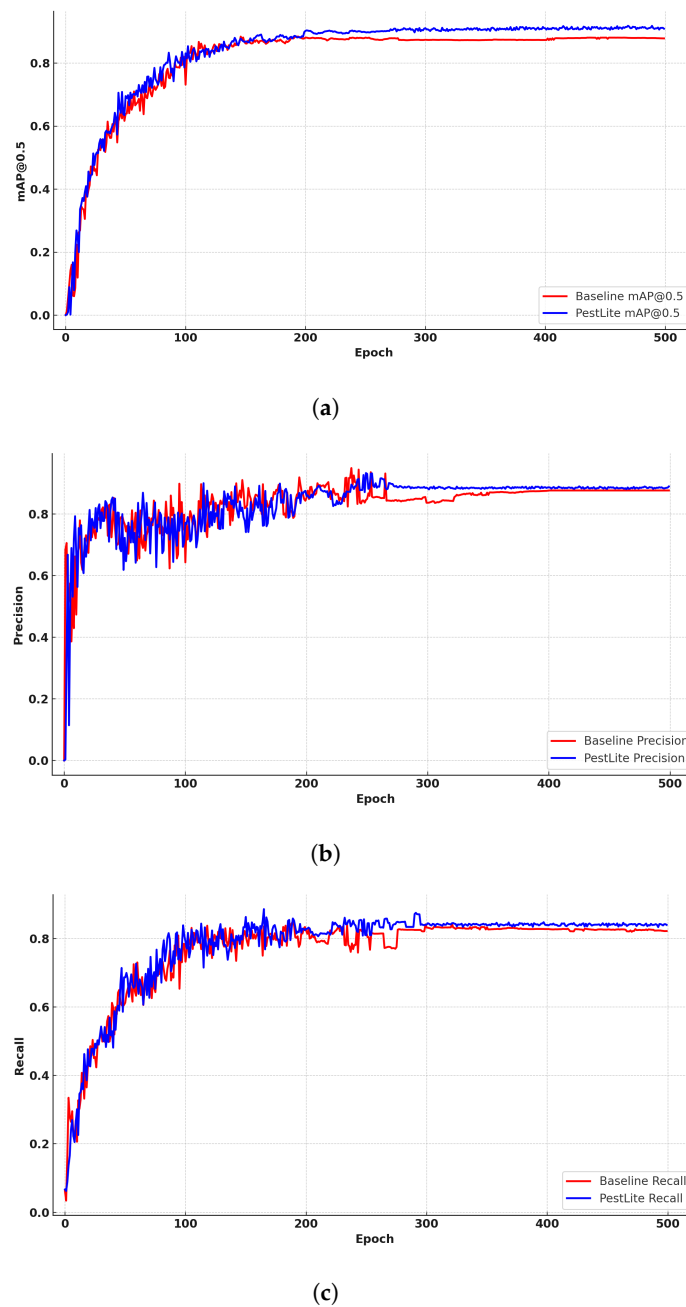


Figure 9. Comparative Metrics Analysis over Epochs: Baseline vs. PestLite on the Datasets. (a) Fluctuation of mAP50 via Epochs; (b) Shift in Precision over Epochs; and (c) Change in Recall throughout Epochs.

In conclusion, each improvement method optimized a specific aspect of YOLOv5, resulting in enhancements across various performance metrics. The amalgamation of these methods into PestLite not only refined the model but also significantly advanced its performance in pest detection tasks. These results validate the effectiveness of our improvement strategies and provide valuable insights for subsequent research [31].

3.3. Performance in IP102

To comprehensively assess the efficacy of our PestLite algorithm under varying conditions and ensure its broad applicability, we expanded our experimental scope beyond initial trials. We conducted a series of additional tests using the IP102 dataset. IP102 is a challenging dataset that encompasses 102 distinct pest categories [32]. Such diversity positions it as an ideal platform for evaluating the broad applicability of a model. As indicated in Table 8,

the parameter count of PestLite reduced from the baseline's 7.28 M to 6.34 M. On key performance indicators, such as Precision (P), Recall (R), and mAP50%, PestLite achieved notable improvements. In the domain of pest identification, achieving real-time monitoring and rapid response is crucial, making model lightweighting imperative. Models with fewer parameters not only help in reducing computational and storage demands but also facilitate easier deployment on resource-constrained devices. Notably, while the number of parameters was reduced, the computational complexity, represented by GFLOPs, also diminished, further highlighting our lightweighting strategy in model design. PestLite's precision rose from the baseline's 54.9% to 57.2%, and recall also increased from 54.6% to 56.4%. Most critically, as a comprehensive evaluation metric, the mAP50% improved from the baseline's 54.5% to 57.1%. These results consistently demonstrate that PestLite can achieve exemplary performance across a broad range of pest categories. In summary, PestLite not only showcases superior performance in the realm of pest detection but also successfully achieves model lightweighting, making it especially suitable for real-time applications and deployment on resource-limited devices. The validations on the IP102 dataset further attest to the wide applicability of our model.

Table 8. Performance in IP102.

Model	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)
Baseline	7.28	16.6	54.9	54.6	54.5
PestLite	6.34	16.3	57.2	56.4	57.1

Note: GFLOPs refers to Giga Floating Point Operations Per Second. P represents Precision. R stands for Recall. mAP50% denotes the mean Average Precision (mAP) evaluation metric with an Intersection over Union (IoU) threshold set at 0.5.

3.4. Robustness Analysis

In the intricate realm of deep learning, a model's robustness—resilient to adversarial attacks or anomalous data—is paramount, especially for applications in dynamic real-world environments [33]. During the development of our model, our primary aim was to enhance its resistance to adversarial perturbations using YOLOv5.

Data Augmentation for Robustness: A pivotal element of our robustness strategy was the Mosaic technique, which merges four distinct images into a single training instance. This not only amplifies the variety within our training set but also strengthens the model's adaptability across a broad range of visual content.

Regularization Techniques: Dropout, with a rate of 0.5 (50%), was employed as a defense against overfitting and adversarial vulnerabilities. By intermittently deactivating certain neurons during training, the model diversifies its dependency on various features, thereby reducing its susceptibility to specific adversarial attacks.

We rigorously evaluated our model's resistance to overfitting during its training phase. As depicted in Figure 10, Figure 10a showcases the model's capability to reduce prediction discrepancies between the predicted and actual object locations, as evidenced by the converging training and validation box loss over epochs. Meanwhile, Figure 10b emphasizes the model's proficiency in accurately predicting object presence, evident from the consistent object loss during both training and validation phases. The convergence pattern for classification loss over epochs in Figure 10c signifies the model's strength in categorizing objects into their respective classes. The consistent performance presented in these figures, especially the converging loss curves, confirms the model's resistance to overfitting. These graphs further underscore the model's unwavering learning trajectory throughout its training phase.

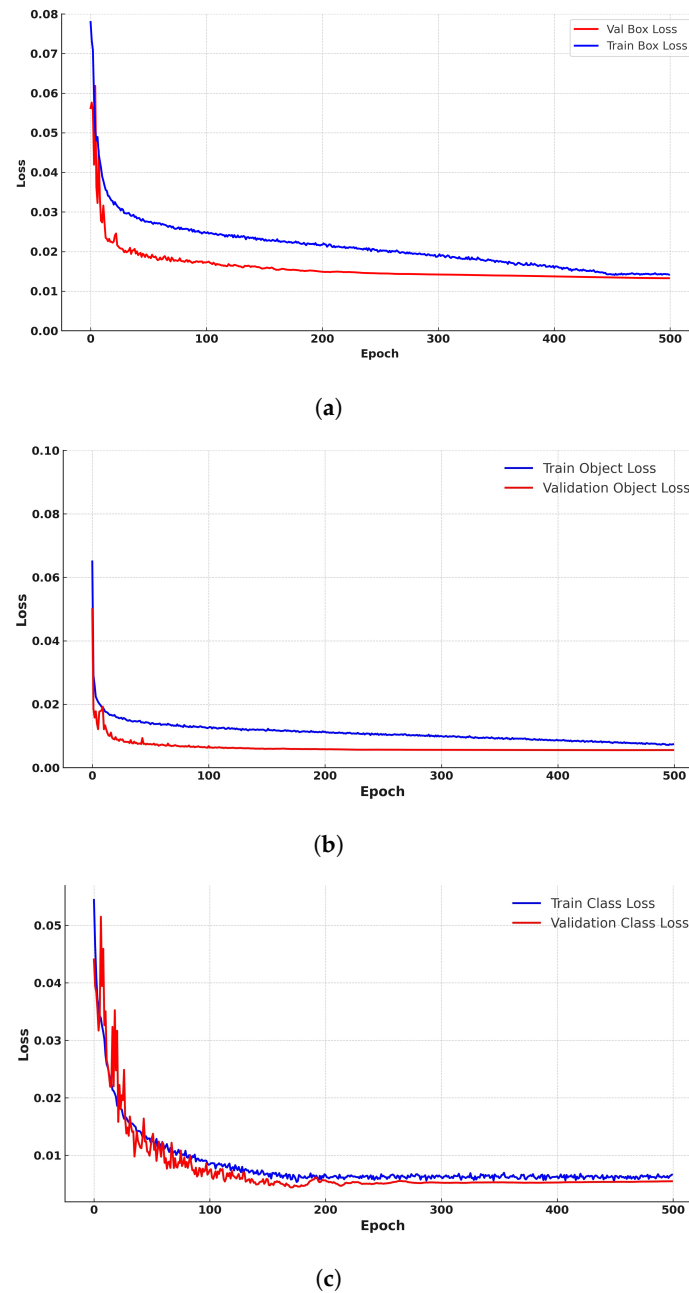


Figure 10. Robustness Comparative Analysis. (a) Comparison of Train and Validation box Loss over Epochs; (b) Comparison of Train and Validation Object Loss over Epochs; and (c) Comparison of Train and Validation cls Loss over Epochs.

In conclusion, the rigorous evaluation of our model's resistance to overfitting, coupled with our strategic modifications, emphasizes its robust capabilities. The visual representations in the figures serve as a testament to the model's promising learning journey. The close convergence of training and validation losses, particularly evident in the box losses, reveals a well-adjusted learning behavior, free from the common challenges of overfitting. This trend is consistently reflected across the object (Obj Loss) and class losses (Cls Loss) as both sets of losses decrease with successive epochs.

Taken together, these graphical analyses reinforce the model's consistent and balanced performance across different datasets, effectively dispelling concerns about overfitting tendencies.

3.5. Ablation Experiment

To understand the impact of the improvements proposed in this research on pest target identification, we set up ablation tests [34]. The experimental approaches are (1) Baseline (YOLOv5): The fundamental YOLOv5 structure without any supplementary modules; (2) YOLO with ECA: A version of YOLOv5 supplemented with the ECA module to fine-tune channel attention, aiming for better detection precision; (3) YOLO incorporated with Involution: This YOLOv5 variant is merged with the Involution procedure, enabling the assimilation of deeper contextual insights over a broader spatial range; (4) YOLO combined with CARAFE: An iteration of YOLOv5 that integrates the CARAFE method, boosting feature rebuilding, particularly during upsampling phases, for enhanced object pinpointing; (5) YOLO integrated with MTSPPF: In this YOLOv5 adaptation, the MTSPPF module is introduced to adeptly capture multi-scale details. With MTSPPF's integration, the goal is to amplify the model's efficacy in detecting petite insect targets; (6) YOLO merging Involution and ECA: This YOLOv5 experiment fuses both the Involution and ECA modules. Such a combination streamlines intricate feature discernment and fine-tunes channel attention, elevating the model's prowess in recognizing minute insect entities; (7) YOLO with CARAFE and Involution: This YOLOv5 iteration marries CARAFE's adept feature reassembly with Involution's targeted feature recognition, targeting sharp and streamlined object pinpointing; (8) YOLO amalgamating Involution, ECA, and CARAFE: This enriched YOLOv5 setup combines the granularity of Involution, the channel-centricity of ECA, and the upscaling finesse of CARAFE, presenting a holistic methodology for meticulous object detection; and (9) PestLite: A holistic YOLOv5 model that combines Involution's localized feature extraction, ECA's channel attention, CARAFE's refined upsampling, and the innovative MTSPPF for multi-level spatial pyramid pooling, delivering state-of-the-art precision in object detection tasks.

Table 9 presents the results from our ablation studies, elucidating the impact of distinct components on YOLOv5's performance. Initially, the baseline YOLOv5 exhibited an mAP50 of 87.9%. The ECA module integration notably improved the mAP50 to 90.1%, underscoring its efficacy in enhancing model precision with no additional computational cost [35].

Table 9. Outcome Analysis of Ablation Experiments.

Algorithm	ECA	CARA	In	MTSPPF	Parameters (M)	GFLOPs	P (%)	R (%)	mAP50 (%)
YOLOv5 (baseline)					7.03	15.8	91.3	81.9	87.9
YOLOv5+ECA	✓				7.03	15.8	93.1	80.8	90.1
YOLOv5+Involution			✓		6.31	14.9	88.7	83.3	89.9
YOLOv5+CARAFE		✓			7.16	16.3	88.7	82.4	88.5
YOLOv5+MTSPPF				✓	6.64	15.5	93.7	78.1	89.0
YOLOv5+In+ECA	✓		✓		6.32	14.9	90.2	82.2	90.2
YOLOv5+In+CARA		✓	✓		6.48	15.8	86.8	82.4	90.1
YOLOv5+In+CARA+ECA	✓	✓	✓		6.48	15.8	89.6	83.7	90.5
PestLite	✓	✓	✓	✓	6.09	15.5	91.9	82.7	90.7

Note: Each checkmark symbolizes the types of modules integrated into the model under this condition.

The introduction of the Involution (In) module brought forth notable improvements: This led to a boost in the mAP50, reaching 89.9%, and concurrently, the model parameters were minimized to 6.31 million. Additionally, the computational demand was reduced to 14.9 GFLOPs. This demonstrates the module's remarkable ability to enhance both performance and efficiency. On the other hand, our integration of the CARAFE (CARA) module further advanced the model's capability, achieving an mAP50 of 88.5%, surpassing the baseline of 87.9%. This result underscores the effectiveness of CARAFE as one of our significant innovations, contributing positively to the model's overall performance.

Moreover, the Multi-Level Spatial Pyramid Pooling (MTSPPF) module boosted the mAP50 to 89.0% while slightly reducing both the model parameters and computational complexity. The combined integration of In, CARAFE, and ECA modules achieved an mAP50 of 90.5%, whereas the PestLite model, encompassing all four modules, achieved the highest mAP50 of 90.7%, with a reduced model parameter count of 6.09 million and maintained GFLOPs at 15.5, illustrating a balanced performance boost and computational

efficiency. On the one hand, the integration and synergistic operation of various modules are pivotal in the optimization of the overall model architecture. Specifically, the Involution and Multi-Level Spatial Pyramid Pooling (MTSPPF) modules, characterized by their lightweight design, are instrumental in delivering efficient functionality without incurring additional computational overhead. The concurrent utilization of these modules engenders a complementary effect, notably enhancing the efficacy of feature extraction and representation processes. This synergistic interaction induces a reevaluation of the necessity of certain components in the original YOLO architecture, thus enabling a more streamlined model configuration conducive to a reduction in parameter count.

Critically, the MTSPPF module introduces an innovative paradigm for the capturing of multi-scale features of the input. This is achieved via a dual-path processing mechanism, wherein features are concurrently extracted from both the unaltered and transformed forms of the input. This dual-path approach not only facilitates a reduction in the overall number of parameters but also significantly augments the diversity and richness of the feature set extracted. Leveraging the capabilities of the MTSPPF module, the model acquires the ability to discern and prioritize more salient features across varying scales, while simultaneously curtailing the incorporation of superfluous, redundant features.

Consequently, when the MTSPPF module is integrated in tandem with other modules, such as the Efficient Channel Attention (ECA) and Content-Aware ReAssembly of FEatures (CARAFE), it exploits its unique functionality to an enhanced degree. This integration allows for a more effective processing of features, which in turn contributes to a more pronounced reduction in the overall parameter requirements of the model. This collaborative effect culminates in a substantial decrement in the parameter count to 6.09 M, representing a significant reduction from the initial parameter estimate.

To provide a clearer insight into how different modules affect model performance, we graphically represented the results of our ablation studies. In this graph, the x-axis denotes the training epochs, while the y-axis signifies the mAP (Mean Average Precision). As depicted in Figure 11, we used mAP as the metric to assess the effectiveness of the various modules integrated into the algorithm. This visualization clearly demonstrates the enhancements each module brings to PestLite, which significantly surpasses the baseline algorithm in terms of mAP.

In summary, PestLite represents a significant advancement in the domain of real-time pest detection. By skillfully integrating multiple modules, PestLite not only significantly improved its performance metrics compared to the baseline YOLOv5, but also achieved a more streamlined architecture, further reducing its parameter count. This optimized design, coupled with superior computational efficiency, establishes PestLite as a benchmark model for real-time applications. Collectively, these improvements highlight PestLite's unparalleled precision and efficiency, solidifying its position as a pioneering solution in pest detection.

3.6. Benchmarking against Leading Algorithms

In our pursuit of identifying an object detection model that harmonizes precision with computational frugality, we conducted a thorough comparative analysis against leading algorithms such as Faster R-CNN and a range of YOLO variants from YOLOv3 to YOLOv8. These models were chosen for their notable contributions to advancements in the field, thereby providing a robust benchmark for assessing the efficacy of PestLite.

PestLite distinguishes itself in parameter efficiency, with a mere 6.09 M parameters, markedly less than its counterparts. For instance, YOLOv5L and YOLOv3 report 46.3 M and 61.5 M parameters, respectively, illustrating a significant reduction with PestLite. PestLite emerges as a model of exemplary lightness and efficiency. This parameter efficiency originates from the innovative use of PestLite's original Multi-Level Spatial Pyramid Pooling (MTSPPF) and the introduction of lightweight operations such as Involution. These innovations consolidate the model's capability to process features adeptly without

the need for superfluous parameter overhead, a quality imperative for deployment on resource-constrained devices.

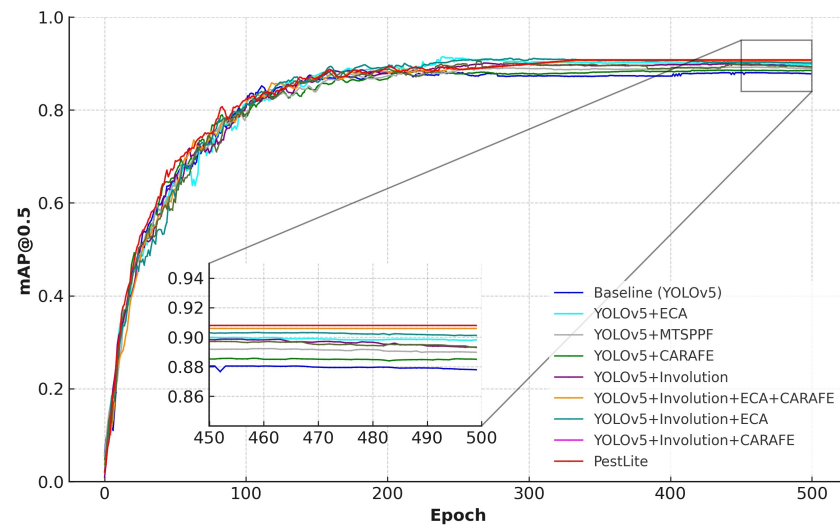


Figure 11. mAP for different module.

In the dimension of computational complexity gauged by GFLOPs, PestLite demonstrates superiority with only 15.5 GFLOPs required, in stark contrast to the more demanding 101.4 GFLOPs for YOLOv7 and 107.6 GFLOPs for YOLOv5L. Moreover, this significant efficiency ensures rapid inference times, indispensable in real-time application scenarios, meeting the demands for immediacy and ease of deployment in the pest detection domain. PestLite's calculated balance between lean design and operational swiftness provides a solution that is both nimble and robust, ready to be deployed in dynamic agricultural settings where time and accuracy are of the essence.

PestLite's mAP50% reaches 90.7%, surpassing YOLOv8 and YOLOv7, which stand at 88.1% and 88.5%, respectively. This enhanced mAP50% is a direct consequence of PestLite's improved feature extraction techniques and advanced anchor box optimization strategies. The integration of novel data augmentation and innovative optimization strategies coalesce to boost the model's proficiency in detecting pests with high precision within diverse and complex agricultural landscapes.

Moreover, PestLite's performance is not limited to the primary metrics; it also excels in Precision and Recall, with scores of 91.9% and 82.7%, respectively. These metrics reflect the model's nuanced capability to detect pests accurately (high precision) while also ensuring minimal missed detections (high recall). The integration of advanced data augmentation and specialized loss functions contribute to these results, allowing PestLite to recognize and accurately classify a high proportion of pest instances within the dataset.

In summary, the benchmarking results not only favorably position PestLite among advanced pest detection algorithms but also highlight its potential as a comprehensive solution. The balance of high accuracy, a lightweight model structure, and reduced computational demand underscores its suitability for real-time pest detection applications, marking it as a significant contribution to the field.

3.7. Visual Interpretation of Experiment Findings

As part of our efforts to highlight the practical advantages of our developed model, PestLite, we engaged in an extensive visual comparison using four sets of images. These sets are particularly chosen as they encompass a wide variety of pests, both in terms of species and size. Within each set, there are two contrasting images: the first was detected using the conventional YOLOv5, and the second demonstrated the capabilities of PestLite.

The detailed outcomes of this comparative analysis are adeptly presented in Figure 12 and Table 10.

Table 10. Advanced algorithm comparison.

Method	mAP50%	Precision%	Recall%	GFLOPs	Parameters
Faster R-CNN	62.6	76.7	58.3	91.03	41.2
YOLOv8	88.1	91.3	81.9	28.9	11.2
YOLOv7	88.5	91.3	81.9	101.4	36.4
YOLOv5L	88.1	89.3	82.1	107.6	46.3
YOLOv5	87.9	91.3	81.9	15.8	7.03
YOLOv4	84.7	88.9	80.6	52.9	24.6
YOLOv3	84.2	87.4	78.3	77.6	61.5
PestsLite	90.7	91.9	82.7	15.5	6.09



(a)



(b)



(c)



(d)



(e)



(f)



Figure 12. Visual comparison between traditional YOLOv5 results and our enhanced PestLite identifications. (a) Recognition results of pest images by the unimproved algorithm. (b) “JinGui” Identification: Advancements with PestLite. (c) Recognition results of pest images by the unimproved algorithm. (d) Improvements in “DaoZhong” Recognition and Rectification of “ErHua” Omissions with PestLite. (e) Recognition results of pest images by the unimproved algorithm. (f) PestLite’s Enhanced Identification of Previously Missed “DaMingShen”. (g) Recognition results of pest images by the unimproved algorithm. (h) PestLite’s Enhanced Identification of Previously Missed “HeiBai”.

As shown in Figure 12, in the first set of images, we observe that under such conditions, the YOLOv5 image displays certain limitations in pest detection. In contrast, the PestLite image clearly offers enhanced detection. The recognition rate for the pest labeled as “JinGui” improved from 0.6 to 0.9, surpassing the accuracy of the original YOLOv5 algorithm. Transitioning to the second set of images, where multiple pests are present in a single frame, the PestLite image distinctly showcases superior multi-object detection capabilities. It effectively identifies and differentiates the types of pests encountered in the image. In contrast, YOLOv5 missed the pest labeled as “ErHua”, while PestLite successfully detected it and improved the accuracy of the pest labeled “DaoZhong” from 0.7 to 0.9. In the remaining sets of photos, YOLOv5 showed varying degrees of misses, whereas PestLite consistently excelled, successfully detecting pests labeled as “DaMingShen” and “HeiBai”. While YOLOv5 might miss certain pests, PestLite can accurately identify them. This enhanced performance can be attributed to PestLite’s integration of Involution, MTSPPE, CARAFE, and ECA, leading to improved capabilities in handling such pest detections.

The selected image sets from our dataset represent a variety of conditions encountered in pest imagery. Through this in-depth visual comparison, our aim is to clearly delineate the advantages PestLite offers over the standard YOLOv5 [36]. By highlighting instances where the conventional algorithm either inaccurately identified or missed a pest, the enhanced performance of PestLite becomes evident.

Further data analysis shows that compared to YOLOv5, PestLite can more consistently identify small or detail-rich targets across different experimental environments. PestLite demonstrates its ability for precise identification and lowering error rates. Additionally, PestLite maintains its robust and outstanding performance across different backgrounds, while also retaining good real-time performance, making it an ideal choice for scenarios requiring instantaneous responses.

Through this in-depth analysis and visual validation, we confirm that our PestLite surpasses the conventional YOLOv5 in multiple core aspects. Whether it is accuracy, flexibility, or response efficiency, PestLite demonstrates superior performance, attesting to the success of our improvements.

4. Conclusions

In this study, we grapple with the intricate challenge of detecting pests within multifarious agricultural environments. To this end, we present “PestLite”, a groundbreaking, lightweight, and potent algorithm tailored for this challenge. Our methodology is rooted in a meticulous refinement of the YOLO framework. This is complemented by the seamless amalgamation of several state-of-the-art modules, notably our bespoke Multi-Level Spatial Pyramid Pooling model (MTSPPF), the Involution operator, ECA attention mechanism, and CARAFE upsampling.

The inherent prowess of PestLite in addressing real-time pest detection emanates from a confluence of meticulously designed components. Our research presents a pioneering approach in the domain of pest detection. A hallmark of our innovation is the independently developed Multi-Level Spatial Pyramid Pooling (MTSPPF), which is specifically tailored for pest detection. This mechanism seamlessly integrates convolution, normalization, and activation operations, promoting enhanced feature extraction across various scales while simultaneously optimizing the model parameters. Furthermore, by integrating cutting-edge Involution and ECA attention mechanisms, our model demonstrates a nuanced contextual understanding, which is paramount for distinguishing pests from their detailed backgrounds. The adoption of the CARAFE method for upsampling emphasizes the model’s capacity to retain intricate nuances, making it particularly adept at detecting minute target pests. Empirical evaluations on a dedicated pest dataset reaffirm the efficacy of our approach, with PestLite outperforming its peers in both accuracy and computational efficiency. In summary, our research not only introduces an innovative module (MTSPPF) but also lays the groundwork for future research that applies advanced deep learning techniques to practical agricultural challenges. Most importantly, PestLite not only enhances detection accuracy but also reduces the model’s parameter count. This balance ensures that the model can operate efficiently in practical applications while maintaining a high level of accuracy. This unique combination positions the PestLite model as a standout in the field of agricultural pest detection, providing a potent tool for future research and applications.

Our meticulous head-to-head comparison with state-of-the-art algorithms, reinforced by extensive validations using the IP102 public pest dataset, has resoundingly confirmed PestLite’s expansive applicability and superior adaptability. The empirical data harvested from these stringent evaluations accentuate the algorithm’s proficiency in conforming to an extensive spectrum of pest species and classifications, cementing its leading status in pest detection technologies. Significantly, the comparative analyses have demonstrated PestLite’s distinguished performance, marked by its streamlined architecture and exceptional accuracy—a synergy that is particularly coveted yet challenging to attain in real-time pest surveillance. PestLite’s exemplary performance not only meets but consistently exceeds existing industry benchmarks. This fusion of instantaneous processing efficacy and meticulous detection precision heralds PestLite as the optimal solution for contemporary agricultural surveillance infrastructure, heralding an upcoming epoch of swift and dependable pest management strategies.

Looking ahead, we plan to explore more architectures and techniques to further optimize PestLite to adapt to a variety of agricultural scenarios. Specifically, we are interested in researching how to better handle small targets and complex background detection issues, as well as how to further reduce computational demands to achieve genuine real-time applications. We also plan to study the integration of PestLite with drones and other smart agricultural technologies to realize large-scale, real-time pest monitoring and management. Through exploring different model variants and optimization strategies, we aspire to overcome the limitations of existing methods and make significant contributions toward sustainable agriculture and global food security.

Author Contributions: Conceptual Framework, Q.D.; manuscript drafting, Q.D. and T.H.; data management, T.H.; professional expertise in data analysis, M.C.; English language editing, L.S. and

C.G.; manuscript review and modifications, Q.D., T.H., M.C. and C.G.; project oversight, L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Association of the Chinese Academy of Sciences (NO. 2022199).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data utilized in this study comprises two parts: the publicly available IP102 detection dataset and our custom-made pest detection dataset. The IP102 dataset is accessible at <https://github.com/xpwu95/IP102> (accessed on 28 December 2023). The data provided in this study can be obtained from the corresponding author upon reasonable request.

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

Abbreviations

The following abbreviations are used in this manuscript:

Abbreviations	Full Name
MTSPPF	Multi-Level Spatial Pyramid Pooling
CARAFE	Content-Aware ReAssembly of FEatures
FAO	Food and Agriculture Organization
CNN	Convolutional Neural Networks
ECA	Efficient Channel Attention
mAP	Mean Average Precision
AP	Average Precision
IoU	Intersection over Union
P	Precision
R	Recall
TP	True Positives
FP	False Positives
FN	False Negatives
GFLOPs	Giga Floating-point Operations Per Second
IN	Involution
CARA	CARAFE
FPS	Frame Per Second

References

1. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
2. Li, W.; Zheng, T.; Yang, Z.; Li, M.; Sun, C.; Yang, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecol. Inform.* **2021**, *66*, 101460. [CrossRef]
3. Kujawińska, A.; Vogt, K. Human Factors in Visual Quality Control. *Manag. Prod. Eng. Rev.* **2015**, *6*, 25–31. [CrossRef]
4. Thenmozhi, K.; Reddy, U.S. Image processing techniques for insect shape detection in field crops. In Proceedings of the 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, India, 23–24 November 2017; pp. 699–704. [CrossRef]
5. Zhang, J.; Wang, Y.; Zhao, Y.; Cai, X. Applications of inferential statistical methods in library and information science. *Data Inf. Manag.* **2018**, *2*, 103–120. [CrossRef]
6. Agarwal, N.; Kalita, T.; Dubey, A.K. Classification of Insect Pest Species using CNN based Models. In Proceedings of the 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), Greater Noida, India, 28–30 April 2023; pp. 862–866. [CrossRef]
7. Oprea, S.; Martinez-Gonzalez, P.; Garcia-Garcia, A.; Castro-Vargas, J.A.; Orts-Escolano, S.; Garcia-Rodriguez, J.; Argyros, A. A Review on Deep Learning Techniques for Video Prediction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 2806–2826. [CrossRef] [PubMed]
8. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.W.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2018**, *128*, 261–318. [CrossRef]
9. Wu, X.; Zhan, C.; Lai, Y.K.; Cheng, M.M.; Yang, J. IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8779–8788. [CrossRef]

10. Liu, L.; Wang, R.; Xie, C.; Yang, P.; Wang, F.; Sudirman, S.; Liu, W. PestNet: An End-to-End Deep Learning Approach for Large-Scale Multi-Class Pest Detection and Classification. *IEEE Access* **2019**, *7*, 45301–45312. [\[CrossRef\]](#)
11. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7794–7803. [\[CrossRef\]](#)
12. Khan, A.R.; Yasin, A.; Usman, S.M.; Hussain, S.; Khalid, S.; Ullah, S.S. Exploring Lightweight Deep Learning Solution for Malware Detection in IoT Constraint Environment. *Electronics* **2022**, *11*, 4147. [\[CrossRef\]](#)
13. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [\[CrossRef\]](#)
14. Li, D.; Hu, J.; Wang, C.; Li, X.; She, Q.; Zhu, L.; Zhang, T.; Chen, Q. Involution: Inverting the Inherence of Convolution for Visual Recognition. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 12316–12325. [\[CrossRef\]](#)
15. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11531–11539. [\[CrossRef\]](#)
16. Wang, J.; Chen, K.; Xu, R.; Liu, Z.; Loy, C.C.; Lin, D. CARAFE: Content-Aware ReAssembly of FEatures. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3007–3016. [\[CrossRef\]](#)
17. Xiang, Q.; Huang, X.; Huang, Z.; Chen, X.; Cheng, J.; Tang, X. Yolo-Pest: An Insect Pest Object Detection Algorithm via CAC3 Module. *Sensors* **2023**, *23*, 3221. [\[CrossRef\]](#)
18. Xie, L.; Ahmad, T.; Jin, L.; Liu, Y.; Zhang, S. A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 507–517. [\[CrossRef\]](#)
19. Han, T.; Sun, L.; Dong, Q. An Improved YOLO Model for Traffic Signs Small Target Image Detection. *Appl. Sci.* **2023**, *13*, 8754. [\[CrossRef\]](#)
20. Kim, M.; Jeong, J.; Kim, S. ECAP-YOLO: Efficient Channel Attention Pyramid YOLO for Small Object Detection in Aerial Image. *Remote Sens.* **2021**, *13*, 4851. [\[CrossRef\]](#)
21. Qin, Z.; Zhang, P.; Wu, F.; Li, X. FcaNet: Frequency Channel Attention Networks. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 763–772. [\[CrossRef\]](#)
22. Yazdanbakhsh, A.; Falahati, H.; Wolfe, P.J.; Esmailzadeh, H.; Samadi, K. GANAX: A Unified MIMD-SIMD Acceleration for Generative Adversarial Networks. In Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, 2–6 June 2018; pp. 650–661. [\[CrossRef\]](#)
23. Chen, S.; Liao, Y.; Lin, F.; Huang, B. An Improved Lightweight YOLOv5 Algorithm for Detecting Strawberry Diseases. *IEEE Access* **2023**, *11*, 54080–54092. [\[CrossRef\]](#)
24. Han, X.; Zhong, Y.; Cao, L.; Zhang, L. Pre-Trained AlexNet Architecture with Pyramid Pooling and Supervision for High Spatial Resolution Remote Sensing Image Scene Classification. *Remote Sens.* **2017**, *9*, 848. [\[CrossRef\]](#)
25. Zhang, L.; Zhao, C.; Feng, Y.; Li, D. Pests Identification of IP102 by YOLOv5 Embedded with the Novel Lightweight Module. *Agronomy* **2023**, *13*, 1583. [\[CrossRef\]](#)
26. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.C.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for MobileNetV3. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324. [\[CrossRef\]](#)
27. Sobti, A.; Arora, C.; Balakrishnan, M. Object Detection in Real-Time Systems: Going Beyond Precision. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1020–1028. [\[CrossRef\]](#)
28. Namiki, S.; Yokoyama, K.; Yachida, S.; Shibata, T.; Miyano, H.; Ishikawa, M. Online Object Recognition Using CNN-based Algorithm on High-speed Camera Imaging: Framework for fast and robust high-speed camera object recognition based on population data cleansing and data ensemble. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 2025–2032. [\[CrossRef\]](#)
29. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141. [\[CrossRef\]](#)
30. Brauwiers, G.; Frasincar, F. A General Survey on Attention Mechanisms in Deep Learning. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 3279–3298. [\[CrossRef\]](#)
31. Liu, X.; Gong, W.; Shang, L.; Li, X.; Gong, Z. Remote Sensing Image Target Detection and Recognition Based on YOLOv5. *Remote Sens.* **2023**, *15*, 4459. [\[CrossRef\]](#)
32. Yang, S.; Xing, Z.; Wang, H.; Dong, X.; Gao, X.; Liu, Z.; Zhang, X.; Li, S.; Zhao, Y. Maize-YOLO: A New High-Precision and Real-Time Method for Maize Pest Detection. *Insects* **2023**, *14*, 278. [\[CrossRef\]](#)
33. Tran, Q.; Shpileuskaya, K.; Zaunseder, E.; Putzar, L.; Blankenburg, S. Comparing the Robustness of Classical and Deep Learning Techniques for Text Classification. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–10. [\[CrossRef\]](#)

34. Gallo, I.; Rehman, A.U.; Dehkordi, R.H.; Landro, N.; La Grassa, R.; Boschetti, M. Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images. *Remote Sens.* **2023**, *15*, 539. [[CrossRef](#)]
35. Chen, G.; Cheng, R.; Lin, X.; Jiao, W.; Bai, D.; Lin, H. LMDFS: A Lightweight Model for Detecting Forest Fire Smoke in UAV Images Based on YOLOv7. *Remote Sens.* **2023**, *15*, 3790. [[CrossRef](#)]
36. Ahmad, I.; Yang, Y.; Yue, Y.; Ye, C.; Hassan, M.; Cheng, X.; Wu, Y.; Zhang, Y. Deep Learning Based Detector YOLOv5 for Identifying Insect Pests. *Appl. Sci.* **2022**, *12*, 10167. [[CrossRef](#)]

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