drones

MDPI

*Article*

# Lightweight and Efficient Tiny-Object Detection Based on Improved YOLOv8n for UAV Aerial Images

**Min Yue [1,*], Liqiang Zhang [1], Juan Huang [2] and Haifeng Zhang [1]**

[1] School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; zhanglq@sues.edu.cn (L.Z.); hfzhang@sues.edu.cn (H.Z.)
[2] School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; huangjuan@sues.edu.cn
* Correspondence: yueminc n@sues.edu.cn

**Abstract:** The task of multiple-tiny-object detection from diverse perspectives in unmanned aerial vehicles (UAVs) using onboard edge devices is a significant and complex challenge within computer vision. In order to address this challenge, we propose a lightweight and efficient tiny-object-detection algorithm named LE-YOLO, based on the YOLOv8n architecture. To improve the detection performance and optimize the model efficiency, we present the LHGNet backbone, a more extensive feature extraction network, integrating depth-wise separable convolution and channel shuffle modules. This integration facilitates a thorough exploration of the inherent features within the network at deeper layers, promoting the fusion of local detail information and channel characteristics. Furthermore, we introduce the LGS bottleneck and LGSCSP fusion module incorporated into the neck, aiming to decrease the computational complexity while preserving the detector's accuracy. Additionally, we enhance the detection accuracy by modifying its structure and the size of the feature maps. These improvements significantly enhance the model's capability to capture tiny objects. The proposed LE-YOLO detector is examined in ablation and comparative experiments on the VisDrone2019 dataset. In contrast to YOLOv8n, the proposed LE-YOLO model achieved a 30.0% reduction in the parameter count, accompanied by a 15.9% increase in the mAP(0.5). These comprehensive experiments indicate that our approach can significantly enhance the detection accuracy and optimize the model efficiency through the organic combination of our suggested enhancements.

**Keywords:** object detection; lightweight; efficient; depth-wise separable convolution; UAV; YOLO



**Citation:** Yue, M.; Zhang, L.; Huang, J.; Zhang, H. Lightweight and Efficient Tiny-Object Detection Based on Improved YOLOv8n for UAV Aerial Images. *Drones* **2024**, *8*, 276. https://doi.org/10.3390/drones8070276

Academic Editors: Oleg Yakimenko, Dimitrios Zarpalas and Pablo Rodríguez-Gonzálvez

Received: 17 April 2024
Revised: 12 June 2024
Accepted: 20 June 2024
Published: 21 June 2024



## 1. Introduction

With the advancements in unmanned aerial vehicles (UAVs) and object detection technology, UAV object detection has emerged as an essential component in numerous domains, such as agriculture [1], surveillance [2,3], disaster response [4], infrastructure inspection [5], and many other fields [6–9]. The use of UAVs allows for efficient and flexible data collection over large or inaccessible areas, making them a valuable tool for tasks that benefit from aerial perspectives. Therefore, researchers and practitioners in computer vision seek to develop lightweight, robust, and accurate UAV object detection algorithms to enhance the capabilities of these aerial systems for diverse applications [10–12]. However, due to various factors, such as the different perspectives of unmanned aerial vehicles and the large imaging distances, small target objects in the images are represented by only a few pixels [13]. Furthermore, complex backgrounds and dynamic random noise can negatively impact the target detection accuracy. Thus, detecting tiny targets remains a significant challenge, considering the existence of these disruptive factors.

In this paper, given the high resolution of images with resolutions up to 2000 × 1500 pixels captured by drones, objects in UAV aerial images smaller than 32 × 32 pixels are categorized as tiny objects. In order to address these unique challenges for capturing tiny objects,

we propose the lightweight and efficient LE-YOLO detection algorithm derived from the YOLOv8n [14] architecture. Our approach can significantly enhance the detection accuracy and optimize the model efficiency by exploring and leveraging features in both the local detail information and channel domains at deeper levels. The principal contributions of this study can be outlined through the following three aspects.

1. To enhance the object detection accuracy for tiny targets and optimize the model efficiency, we introduce the LHGNet backbone, which is a more extensive feature extraction network. This network leverages the LHG block to exploit the advantages of depth-wise separable convolution [15] and the channel shuffle module, establishing a deep feature extraction network that comprehensively extracts and integrates local detail information and channel characteristics across different stages.
2. We present and utilize the LGS bottleneck and LGSCSP fusion module, further optimizing the model efficiency. The LGS bottleneck is accomplished by introducing an additional convolutional layer and substituting the add operation with a concatenation operation. The goal is to preserve the channel information as much as possible, facilitating more detailed feature information extraction across various scales.
3. We optimize the detection head of YOLOv8n by modifying its structure and the size of the feature maps. This adjustment substantially improves the detection accuracy in capturing tiny targets.

The subsequent sections of this paper are organized as follows. Section 2 summarizes some related works about the YOLO series algorithms and the performance improvement of detectors. Section 3 presents our proposed LE-YOLO for tiny objects based on YOLOv8n, followed by a description of each enhanced module's structure and functions. Section 4 provides a detailed overview of the experimental setup, along with the presentation and discussion of the experimental results. This paper concludes with a summary in the Section 5.

## 2. Related Works

The You Only Look Once (YOLO) series represents a single-stage object detection model, significantly contributing to the advancement of real-time computer vision detection tasks. The original YOLOv1 [16] model, introduced by Joseph Redmon et al., utilizes convolutional neural networks (CNNs) to extract features and identify objects' categories and locations. However, due to its use of only two predicted bounding boxes for each grid cell, the detection performance is not satisfactory when dealing with multiple densely packed objects or small targets. To address this problem, YOLOv2 [17] adopted Darknet19 for feature extraction and introduced the anchor structure. It utilized the K-means method to improve the anchor templates, enhancing the algorithm's recall rate and addressing issues related to insufficient localization accuracy. Building upon this foundation, YOLOv3 [18] introduced the residual learning concept to deepen the network and proposed Darknet53. Simultaneously, it drew inspiration from the feature pyramid idea, making predictions at three different scales and successfully tackling the detection of small objects. However, the complexity of its structure and the high number of parameters resulted in a reduction in detection speed. Derived from YOLOv3, YOLOv4 [19] incorporated the CSP [20] network structure into Darknet53, creating the new backbone network CSPDarknet53. It also employed the SPP [21] module to enlarge the receptive field and utilized the FPN [22] network and PAN [23] network for feature fusion. Furthermore, it introduced the mish [24] activation function to enhance the precision. Assessed on the MS COCO dataset, this algorithm attained 43.5% AP and 65.7% AP50 on an NVIDIA V100, surpassing a speed of 50 FPS. YOLOv5 [25] introduced the focus module, extended the CSP network to the neck structure, and replaced the SPP with the SPPF structure. YOLOv6 [26] introduced RepVGG and replaced the backbone network of CSPDarknet with EfficientRep. It created a Rep-PAN structure in the neck and designed an efficient decoupled head to accelerate the convergence and reduce the complexity of the detection head. YOLOv7 [27], building on

the previous versions, introduced the E-ELAN structure to enhance the learning capabilities without disrupting the original gradient path.

YOLOv8 [14] stands as the most recent release, where the C3 fusion module has been replaced with the C2f fusion module. The C2f fusion module incorporates the strengths of the ELAN and C3 fusion modules, maintaining a lightweight structure while gaining richer gradient flow information. Inspired by YOLOX, YOLOv8 employs a decoupled head to extract the target position and category information separately, further enhancing its performance. YOLOv8 deviates from the traditional anchor-based approach, embracing the anchor-free concept. It employs VFL loss for classification, while DFL loss and CIOU loss are applied for regression. These enhancements not only preserve the strengths of YOLOv5 but also significantly improve the detection accuracy and speed. It retains its lightweight characteristics and enhances its performance across different scenarios.

The YOLO series models redefine object detection as a regression problem, enabling the rapid recognition and localization of objects. This makes them highly suitable for the real-time detection of small target objects. Numerous researchers have introduced diverse techniques and algorithms to enhance the detection of small objects, particularly by refining the YOLO algorithm. In response to the performance shortcomings of YOLOv3 in remote sensing target detection, Xu et al. [28] adopted DenseNet to enhance the feature extraction capabilities. Xu et al. [29] utilized a portion of the YOLOv3 backbone to establish an auxiliary network and applied attention mechanisms for feature extraction. However, due to YOLOv3's utilization of the K-means method to derive nine specific anchors, it tends to underutilize certain scale features, resulting in the suboptimal detection of tiny objects. Guo et al. [3] introduced LMSD-YOLO, which consists of depth-wise separable convolution, Mobilenet with a stem block, and a depth-wise adaptively spatial feature fusion module. It is characterized by a reduced model size and improved detection accuracy, allowing for the precise identification of multi-scale targets in more intricate scenes. To achieve a balance between detection accuracy and efficiency, Qiu et al. [30] introduced the single-stage headless context model and GhostNet and proposed a new specialized vehicle detection algorithm named YOLO-GNS from the perspective of drones. To improve the detection accuracy and optimize the model efficiency, Cao et al. [31] proposed a GhostConv-based backbone, reconstructed a shallow feature fusion network, and proposed GCL-YOLO for UAV small-object detection. Xie et al. [32] designed the partial hybrid dilated convolution (PHDC) block module and constructed CSPPartialStage to improve the detection and reduce the computational burden. Luo et al. [33] fused the ghost convolution and efficient multi-scale attention modules into YOLOv8 to improve the detection speed while maintaining the parameter size. Despite significant advances in the field of UAV small-target detection, the prevailing methods for detection continue to encounter challenges in achieving a balance between accuracy and efficiency.

## 3. Method

### *3.1. The Structure of YOLOv8*

YOLOv8 represents the latest advancement in the YOLO series, featuring five models tailored to varying scale factors: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. The architecture of YOLOv8 is illustrated in Figure 1 and mainly consists of three sections: the backbone, neck, and detection head. YOLOv8 employs the C2f module, which enhances the gradient flow in the backbone and neck. The C2f module utilizes more skip connections and eliminates convolution operations in its branches, inspired by the C3 module and ELAN from YOLOv5. Furthermore, YOLOv8 employs a decoupled head to extract the target position and category information separately, further enhancing its performance. Moreover, YOLOv8 has shifted from the conventional anchor-based approach to an anchor-free design. It uses VFL loss for classification, while DFL loss and CIOU loss are applied for regression tasks. In order to develop a more lightweight and efficient detection algorithm for drones, we primarily enhance the structure based on the YOLOv8n model.

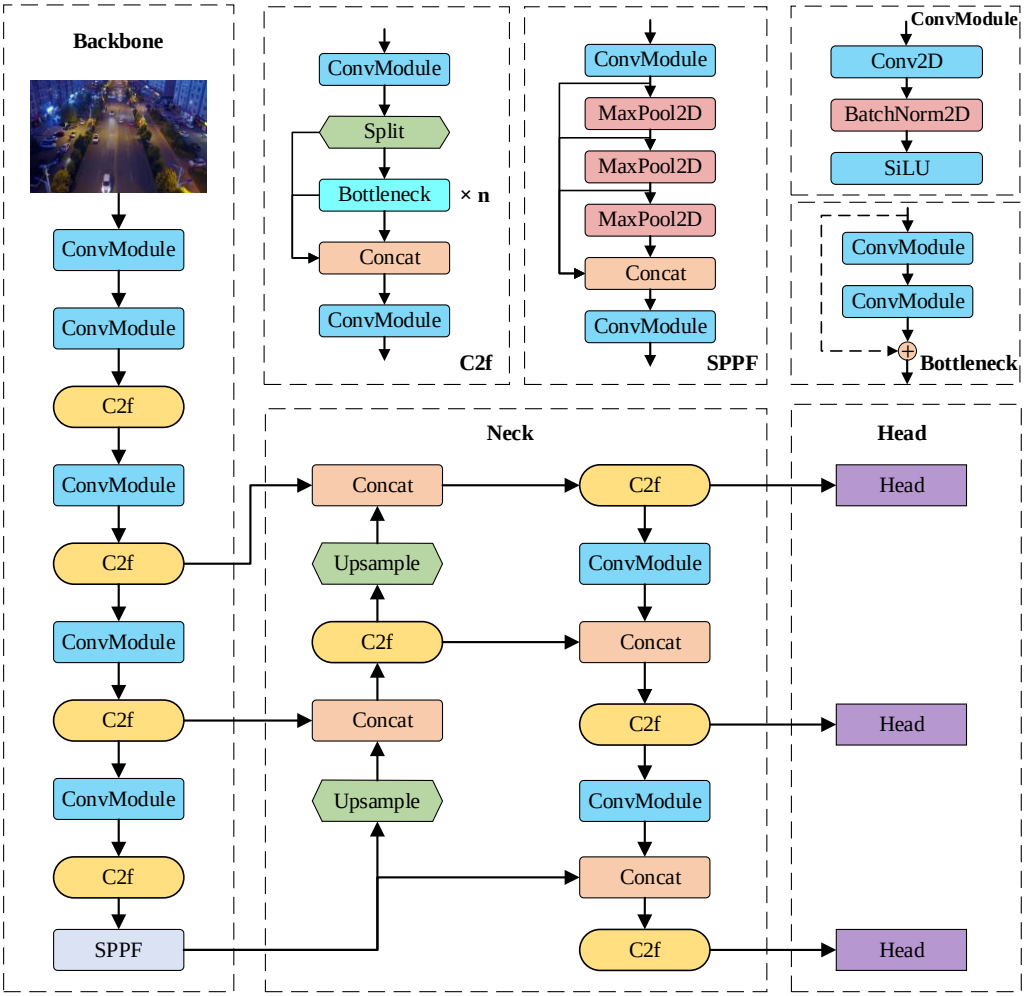



**Figure 1.** Structure of YOLOv8.

### *3.2. Improved Algorithm*

We present a lightweight and efficient framework (LE-YOLO) for the detection of tiny objects. Figure 2 illustrates the overall structure of LE-YOLO, an enhanced version of the YOLOv8n model.

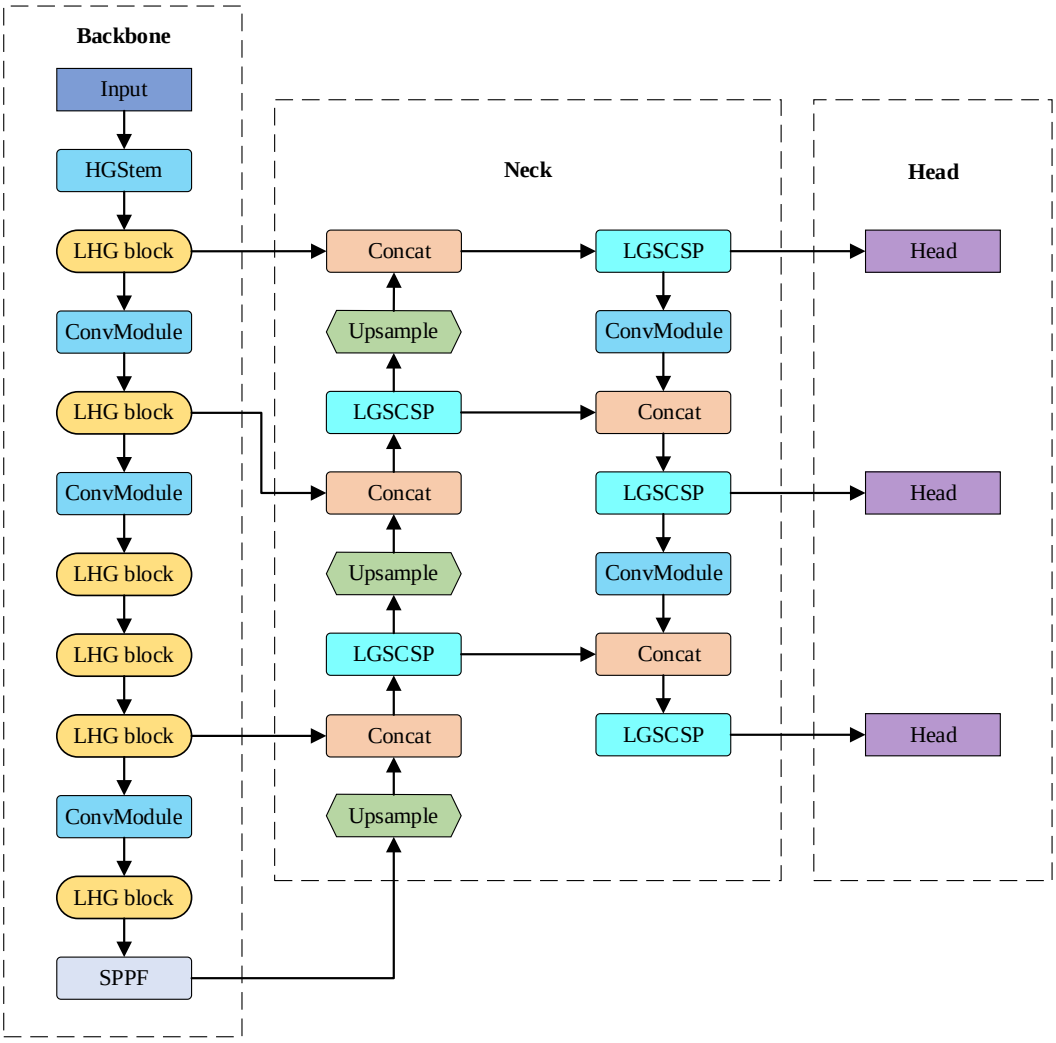



**Figure 2.** Structure of LE-YOLO.

#### 3.2.1. LHGNet Backbone

The backbone is the foundational framework in constructing a complete object detection network. It is tasked with extracting essential features from the image data for further processing. The most commonly used backbone networks include VGG [34], ResNet [35], DenseNet [36], Darknet [17], and so on, which have already demonstrated strong feature extraction capabilities in classification tasks. Among them, the VGG model originated in 2014 at the Visual Geometry Group laboratory. The authors explored the convolutional network depth, demonstrating that gradually increasing the network depth through residual networks can enhance the overall performance of the model. However, as the network depth increases, the issue of gradient vanishing emerges during the training process. With the emergence of the residual technique introduced in ResNet [35], a significant solution was provided for this issue, which can be considered a groundbreaking achievement in the field of backbone architectures. Subsequently, numerous networks with deeper and superior layers emerged, such as ResNeXt [37], DenseNet [36], CSPNet [20], VOVNet [38], etc. Influenced by this, we attempt to enhance the backbone's feature extraction abilities by introducing the residual technique. Therefore, we introduce the HGNetv2 [39] backbone from the Baidu Paddle-Paddle Vision Team, aiming to improve the YOLOv8n backbone. It has recently demonstrated superior performance and achieved a more optimal balance between performance and efficiency in the RT-DETR [40] network. Simultaneously, we incorporate depth-wise separable convolutions into the backbone to address the challenge

of the significantly increasing parameters with the growing network depth. As a result, we design an LHGNet backbone with a lightweight and efficient design.

Figure 3 visually contrasts standard convolution with depth-wise separable convolution. Figure 3a illustrates the convolutional process of applying an ordinary convolution kernel to an input feature map. The standard convolution can be considered a linear model, and its abstraction capability is insufficient, requiring the introduction of more potent nonlinear functions. However, as shown in Figure 3b, depth-wise separable convolution is composed of two essential components, depth-wise convolution and pointwise convolution, both playing crucial roles in the overall architecture. Due to the addition of pointwise convolution, the output channels can be freely adjusted, and it also allows for the channel fusion of the feature map output from depth-wise convolution. Hence, it exhibits superior generalization capabilities and reduces the computational complexity compared to standard convolution.

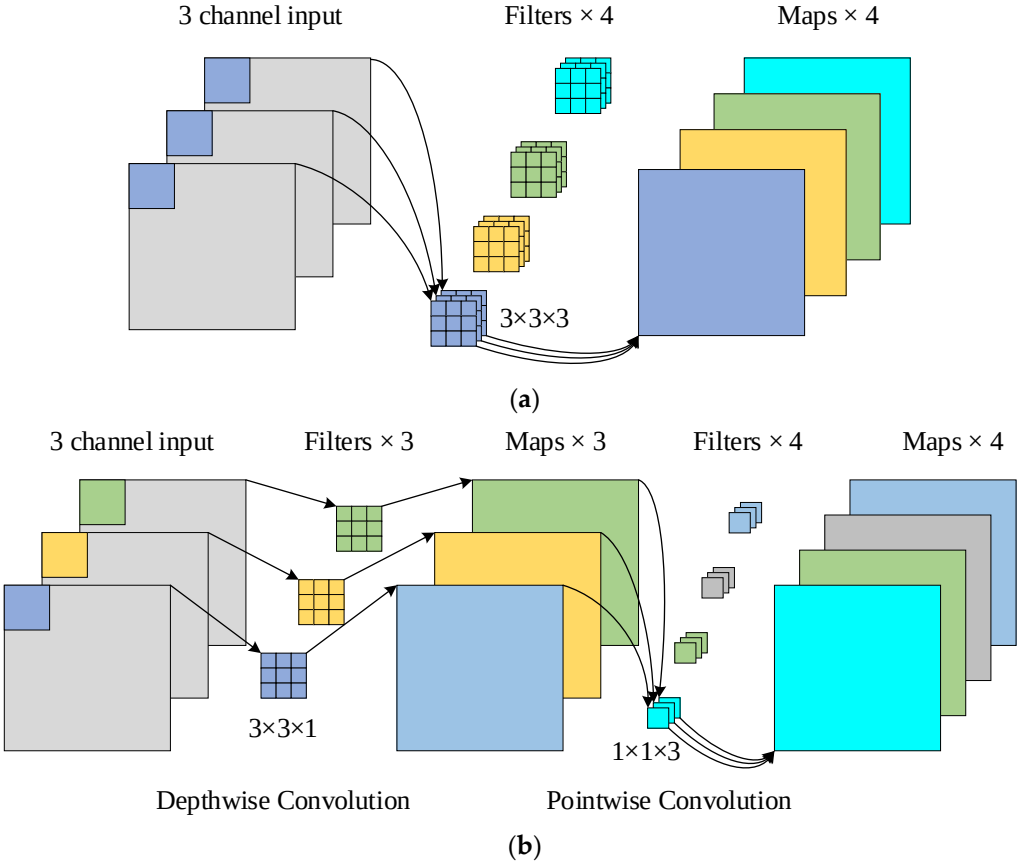



**Figure 3.** A comparison between (**a**) standard convolution and (**b**) depth-wise separable convolution.

The HGNetv2 [39] backbone developed by the Baidu Paddle-Paddle team primarily consists of the HGStem, the HG block, a learnable downsampling (LDS) layer, global average pooling (GAP), convolution, and fully connected (FC) layers, as shown in Figure 4a. Within HGNetv2, the HG block serves as a fundamental module, designed to process data hierarchically. The process involves using numerous stacked convolutional modules to extract information across various layers, particularly focusing on high-level semantic information at deeper layers. Furthermore, the integration of rich gradient information from different layers through the cross-stage partial (CSP) strategy facilitates the extraction of varied features. To simplify the computations and construct a deeper network, thereby exploring and utilizing features between the spatial and channel domains at a deeper level, we replace all $3 \times 3$ standard convolution modules with two depth-wise separable convolution modules. Depth-wise separable convolution facilitates the fusion of the channels in the feature map output by depth convolution, thereby promoting the integration of the

channel characteristics and more efficiently extracting the channel features. As shown in Figure 4b, the LHG block is constructed with two depth-wise separable convolutions arranged sequentially, followed by a concatenation operation, and is then processed by two 1 × 1 standard convolution modules. These two 1 × 1 standard convolution modules play a crucial role in our LHG block. They not only facilitate convenient adjustment of the output feature map dimensions but also play a pivotal role in extracting local information, particularly relevant for detecting tiny and sparse targets. This contributes to improving the detection accuracy of capturing tiny objects. To enhance the channel information fusion, we integrate a channel shuffle module between these two 1 × 1 convolution modules. Additionally, by incorporating residual techniques and the CSP strategy, the LHG block gains richer gradient flow information. This enables the acquisition of feature information at different depths and receptive fields, thereby further enhancing its ability to extract features from tiny and sparse targets. In this manner, the LHG block can extract more detailed features from each feature map, guided by local and channel information through two 1 × 1 standard convolution and channel shuffle modules.

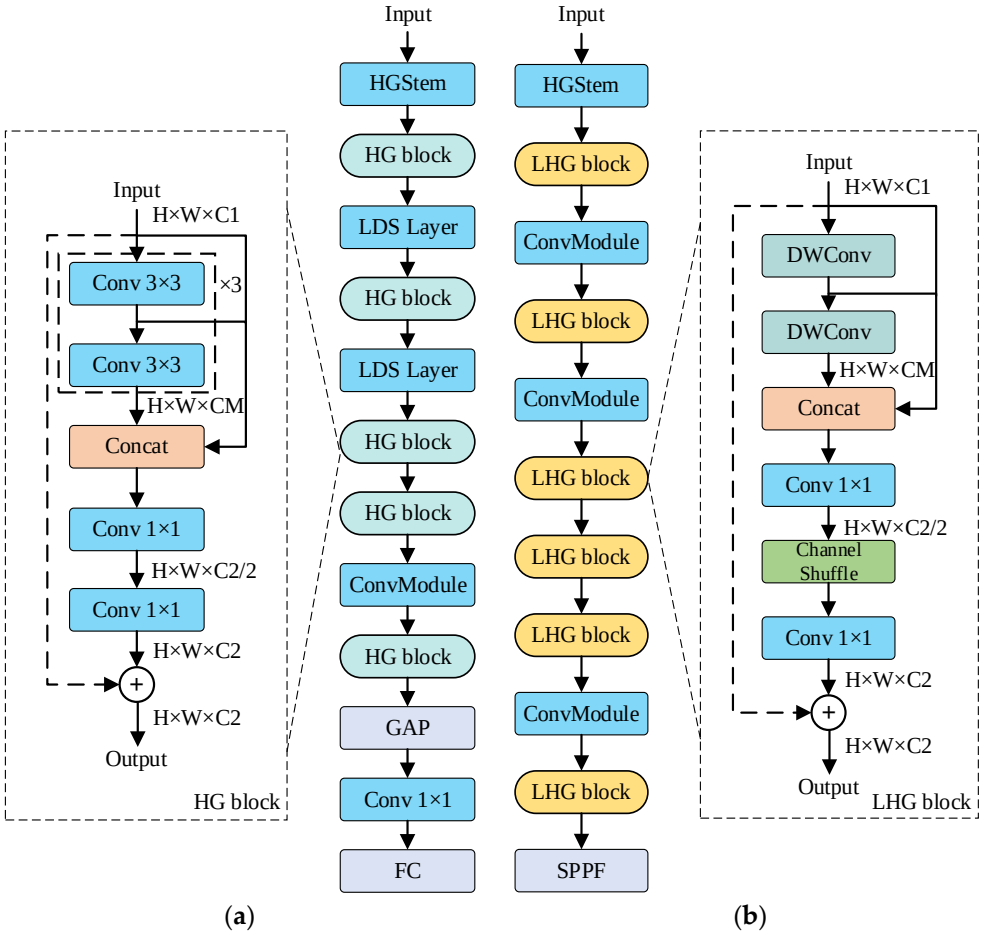



**Figure 4.** The structure of (**a**) HGNetv2 and (**b**) LHGNet. (Note: the add operation is performed when the shortcut is True and the input C1 channel is equal to the output C2 channel.).

The LHGNet backbone includes the HGStem module, LHG blocks, convolution module blocks, and a spatial pyramid pooling fast (SPPF) block, as shown in Figure 4b. The initial image undergoes processing through the HGStem module, as illustrated in Figure 5, primarily serving to upsample and downsample the features of the input image. Subsequently, we replace the C2f module with the LHG block based on depth-wise separable convolution to extract features at various depths. Compared to C2f, the LHG block can extract more detailed features from each feature map through two 1 × 1 standard convolution modules. Furthermore, we downsample the feature maps from different stages using

3 × 3 convolutional modules with a stride of 2. The final-stage feature map is channeled into the SPPF module, enabling the extraction of advanced semantic information. It is worth mentioning that by adopting the lightweight LHG block to replace the C2f module, we enable the backbone network to stack more LHG blocks under the same parameter constraints. This allows the backbone network to increase the receptive field and explore the inherent features within the network at deeper layers, promoting the fusion of spatial characteristics.

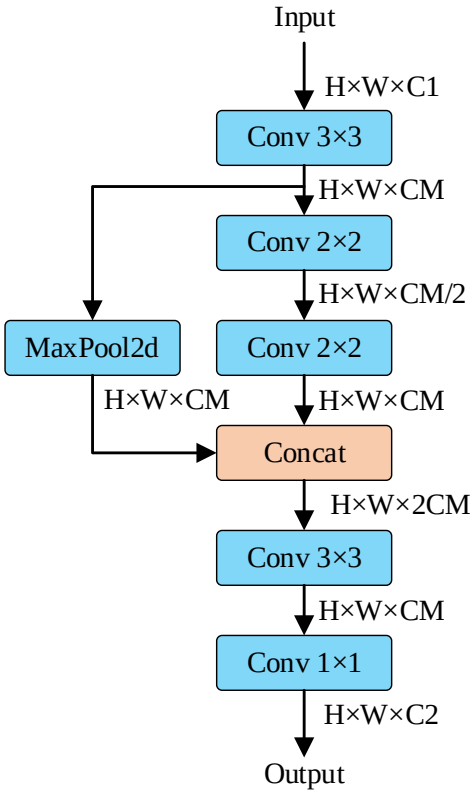



**Figure 5.** The structure of the HGStem.

3.2.2. Improvement of Neck

The neck structure is a crucial component in this model, playing a vital role in its feature extraction and fusion. Following the PAN-FPN concept in YOLOv8, the neck section integrates the output feature maps from multiple stages of the feature extraction backbone into the PAN-FPN network structure, fostering robust multi-scale feature fusion for enhanced object detection performance. Considering that the feature maps generated from the feature extraction backbone are relatively tiny and may not easily detect tiny and densely packed targets, we extend the PAN-FPN module by adding a feature fusion layer, specifically for the detection of tiny objects, after the first LHG block in the backbone. Subsequently, we remove the feature fusion layer in the neck that is responsible for detecting large objects, while still maintaining the output of three layers of feature maps for the detection head to identify tiny, small, and medium-sized targets. This facilitates feature extraction and fusion across multiple stages, thus improving the model's ability to represent features. Simultaneously, we integrate GSConv [41] and introduce a more efficient and lightweight LGS bottleneck and LGSCSP fusion module to make the model more lightweight.

Figure 6 illustrates the structure of GSConv. GSConv combines the advantages of standard convolution, depth-wise separable convolution, and shuffle operations. The shuffle operation can permeate the information generated by these two different convolutional modules. This facilitates the exchange of local feature information across various channels, without consuming any computational resources. It is important to note that the feature

maps that it produces are very close to those generated by standard convolution, but it is more lightweight than standard convolution [41].

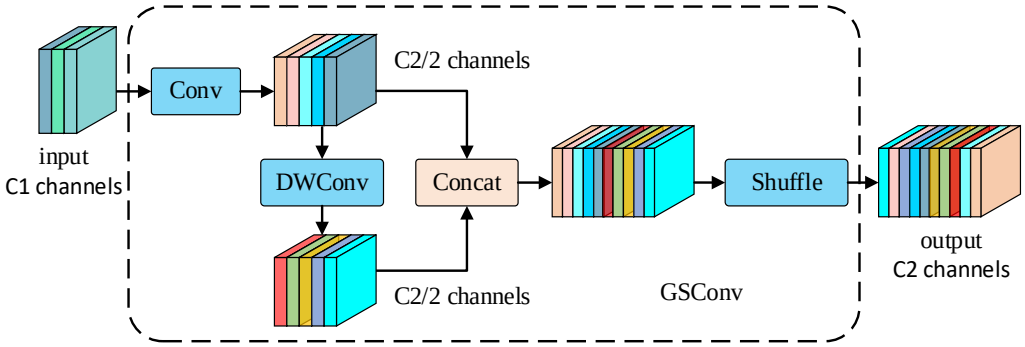



**Figure 6.** The structure of GSConv.

The introduction of GSConv [41] and the GS bottleneck [41] addresses the issue of semantic information loss in the compression and channel expansion processes of the feature maps. The GS bottleneck is illustrated in Figure 7a. It aims to minimize the computational complexity as much as possible while preserving the detection accuracy. However, in the GS bottleneck, the operation of adding the results of $1 \times 1$ standard convolution and GSConv, while increasing the description features from the feature maps, still incurs a certain loss of dimensional information. Therefore, we attempt to divide the input feature maps into two groups to perform convolution operations separately, use a concatenation instead of an addition operation, and design the LGS bottleneck to improve the GS bottleneck. Within the LGS bottleneck, the initial feature map undergoes feature extraction through standard convolution and GSConv, respectively, and then, the resulting feature maps are concatenated. As illustrated in Figure 7b, the improved LGS bottleneck not only preserves the image features and channel information as much as possible but also further reduces the computational complexity.

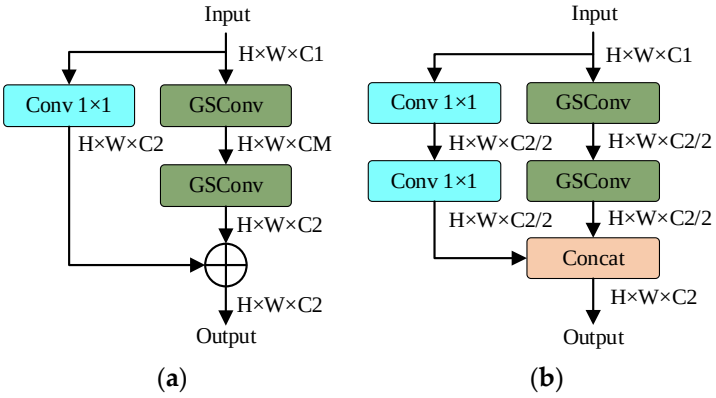



**Figure 7.** The structure of the (**a**) GS bottleneck and (**b**) LGS bottleneck.

Then, leveraging the CSPNet for branch extraction and integrating the concept of the residual structure, we design the LGSCSP fusion module as an alternative to the C2f fusion module for feature fusion, as shown in Figure 8.

Table 1 illustrates the number of model parameters for the necks of YOLOv8n and our proposed neck. As shown in Table 1, the proposed neck architecture exhibits a significant increase in network depth and width, while the computational parameters are reduced by 63% compared to YOLOv8n.

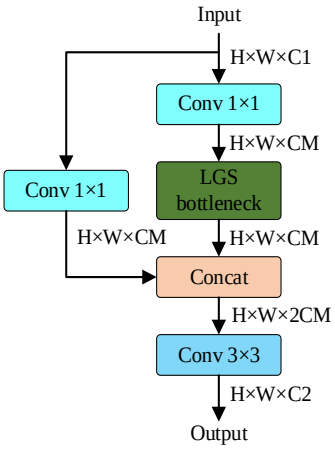



**Figure 8.** The structure of the LGSCSP fusion module.

**Table 1.** The neck architectures of YOLOv8n and the proposed neck.

| YOLOv8n Neck | | | | Proposed Neck | | | |
|---|---|---|---|---|---|---|---|
| **Layer Name** | **The Number of Input/Output Channels** | **The Number of Layers** | **The Number of Model Parameters ($\times 10^6$)** | **Layer Name** | **The Number of Input/Output Channels** | **The Number of Layers** | **The Number of Model Parameters ($\times 10^6$)** |
| 0. Upsample | 256/128 | 1 | - | 0. Upsample | 256/256 | 1 | - |
| 1. Concat | 384/384 | 1 | - | 1. Concat | 512/512 | 1 | - |
| 2. C2f block | 384/128 | 15 | 0.148 | 2. LGSCSP | 512/128 | 46 | 0.137 |
| 3. Upsample | 128/64 | 1 | - | 3. Upsample | 128/128 | 1 | - |
| 4. Concat | 192/192 | 1 | - | 4. Concat | 256/256 | 1 | - |
| 5. C2f block | 192/64 | 15 | 0.037 | 5. LGSCSP | 256/64 | 46 | 0.035 |
| 6. ConvModule | 64/64 | 3 | 0.037 | 6. Upsample | 64/32 | 3 | - |
| 7. Concat | 192/192 | 1 | - | 7. Concat | 96/96 | 1 | - |
| 8. C2f block | 192/128 | 15 | 0.124 | 8. LGSCSP | 96/32 | 46 | 0.008 |
| 9. ConvModule | 128/128 | 3 | 0.148 | 9. ConvModule | 32/64 | 3 | 0.019 |
| 10. Concat | 384/384 | 1 | - | 10. Concat | 128/128 | 1 | - |
| 11. C2f block | 384/256 | 15 | 0.493 | 11. LGSCSP | 128/64 | 46 | 0.027 |
| | | | | 12. ConvModule | 64/64 | 3 | 0.037 |
| | | | | 13. Concat | 192/192 | 1 | - |
| | | | | 14. LGSCSP | 192/128 | 46 | 0.096 |
| Total. | | 72 | 0.987 | | | 246 | 0.359 |

#### 3.2.3. Improvement of Detection Head

The inherent challenge in object detection arises from the conflicting nature of classification and regression tasks, which is widely acknowledged. As a result, decoupled heads for classification and localization are commonly utilized in the majority of detectors. In our proposed model, we continue to employ the decoupled detection head from YOLOv8n. Both parallel branches utilize two $3 \times 3$ convolution modules to extract category features and position features, followed by a $1 \times 1$ convolution layer to accomplish the classification and regression tasks separately. In terms of loss functions, we utilize the binary cross-entropy (BCE) loss for classification, while the distribution focus loss (DFL) and complete IoU (CIOU) loss are utilized for regression.

YOLOv8n features three detection heads, each associated with feature maps sized at $80 \times 80$ pixels, $40 \times 40$ pixels, and $20 \times 20$ pixels, facilitating multiple-scale target detection. However, when our targets are predominantly small objects, the $20 \times 20$ pixels feature map contains limited information, significantly constraining its ability to detect tiny objects. To address the specifics of tiny target detection, a miniature target detection head is integrated into the detection head with a $160 \times 160$ pixels feature map to improve the model's capacity to detect tiny objects, as illustrated in Figure 9. Additionally, we cropped the $20 \times 20$ pixels feature map, which contained limited information on small targets, thereby further reducing the model's parameters. This integration corresponds

directly to the additional feature maps introduced in the neck structure, enhancing the accuracy in detecting tiny targets. Simultaneously, the detection branch for the detection of large targets is removed, thereby reducing the computational parameters of the model.

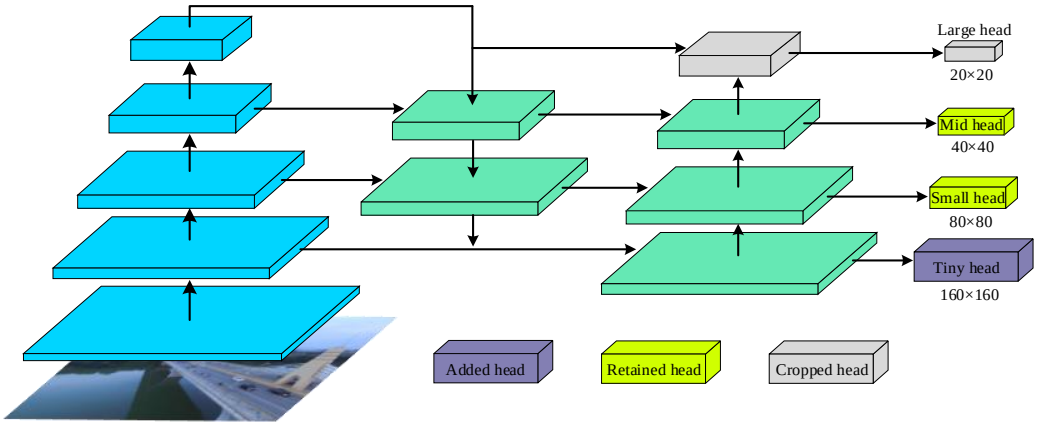



**Figure 9.** The structure of the improved detection head.

## 4. Experiments

### *4.1. Dataset and Parameter Settings*

The VisDrone2019 [42] dataset is a comprehensive benchmark particularly designed for object detection, tracking, and counting, focusing on the perspective of UAVs. Released in 2019, the dataset presents a diverse and challenging collection of high-resolution aerial images captured from various urban scenarios, including crowded streets, intersections, and public spaces. This dataset comprises 261,908 video frames and 10,209 static images captured from the perspective of the drones at various locations and altitudes. These images feature resolutions from 2000 × 1500 pixels to 480 × 360 pixels and contain 2.6 million manually annotated bounding boxes for commonly observed objects. It includes diverse object categories, such as pedestrians, vehicles, and tricycles, reflecting the diversity and intricacies of real-world urban scenes. Due to its drone perspective, the images differ significantly from those captured by ground individuals, such as MS-COCO and VOC2012, in terms of the capture angles, object scales, background, and weather conditions. These aspects enhance the complexity of the dataset, making it a valuable resource in assessing the performance and robustness of computer vision models.

In the experiment, we employed an NVIDIA RTX4090 as the GPU hardware and Ubuntu 22.04 as the operating system, and we established a computing software environment with Python 3.10, PyTorch 2.1.1, and Cuda 11.8. Given the computational capacity of the computer and the size of the dataset, we set the batch size to 8 and maintained the default training parameters of YOLOv8 for additional hyperparameters. In addition, all models were trained from the initial parameters, without loading any pre-trained weights. Finally, the training duration was configured for 200 epochs, and the input images for the network were resized to 640 × 640 pixels. The hyperparameters utilized throughout the training, testing, and validation processes remained constant. Table 2 details the crucial parameter configurations employed throughout the training process.

In the embedded systems experiments, we employed the Jetson Nano B01 platform released by NVIDIA in 2020, which is equipped with 128-core NVIDIA Maxwell™ architecture GPU and 4 GB memory, offering a peak performance of 0.5 TFLOPS. The embedded platform utilizes Ubuntu 18.04 with Python 3.8 as its software testing environment.

**Table 2.** The crucial parameter configurations.

| Parameter | Setup |
|---|---|
| Epoch | 200 |
| Image Size | 640 × 640 pixels |
| Batch Size | 8 |
| Optimizer | SGD |
| Learning Rate | 0.01 |
| Momentum | 0.937 |
| Weight Decay | 0.0005 |

### *4.2. Experimental Metrics*

The experiments assessed the proposed methodologies by analyzing the detection performance and model parameter size. The evaluation metrics encompassed the average precision (AP) and mean average precision (mAP). The precision (P) and recall (R) are calculated as shown in the following equations:

$$P = \frac{TP}{TP + FP} \times 100\%, \tag{1}$$

$$R = \frac{TP}{TP + FN} \times 100\% \tag{2}$$

where TP (true positive) signifies the number of accurately detected targets, FP (false positive) represents the number of background instances mistakenly detected as targets, and FN (false negative) refers to the number of background instances erroneously identified as targets. The average precision (AP) and mean average precision (mAP) are calculated as shown below:

$$AP = \int_0^1 P(R)dR, \tag{3}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{4}$$

where N denotes the total number of categories, and AP represents the average accuracy of each category.

In addition, to validate the detection capabilities of the proposed algorithm for tiny objects, we define objects smaller than 32 × 32 pixels as tiny objects based on the classic COCO evaluation metrics. The $mAP_{tiny}$ signifies the mean average precision of these tiny targets.

### *4.3. Ablation Experiments*

We designed two sets of ablation experiments. One set of ablation experiments was aimed at validating the effectiveness of the LGS and LGSCSP algorithms, while the other set was intended to assess the influence of different enhancements on the model. We validated the effectiveness of our method with YOLOv8n as a baseline.

To assess the performance of the introduced LGS bottleneck and LGSCSP fusion module, we designed the first set of ablation experiments by replacing these improved modules with C2f in the neck. Table 3 reports the influence of the GS and LGS bottlenecks evaluated on VisDrone2019. From the table, it can be observed that when we use YOLOv8n as the base with only the LGS bottleneck and LGSCSP fusion module, the detection results are consistent with those of the GS bottleneck. However, when we replace the backbone with the LHG backbone and modify the detection head branch, the detection results obtained with the LGS bottleneck and LGSCSP fusion module are better, and the number of parameters is also slightly smaller. Compared to row 4 in Table 3, our improvements show a 2.3% increase in both mAP(0.5) and mAP(0.5:0.95), with a slight reduction in parameter count by 4.5%. Although the improvement provided by the LGS bottleneck and the LGSCSP fusion module in the neck will slightly reduce the detection accuracy, maintaining a balance

between the detection speed and accuracy is critical in detecting tiny objects, given that the detection accuracy meets the requirements. As a result, the effectiveness of our improved LGS bottleneck and LGSCSP fusion module is successfully validated.

**Table 3.** Influence of GS and LGS bottleneck evaluated on VisDrone2019-val.

| Model | mAP(0.5) (%) | mAP(0.5:0.95) (%) | The Number of Model Parameters ($\times 10^6$) | FLOPS ($\times 10^9$) |
|---|---|---|---|---|
| Baseline (YOLOv8n) | 33.9 | 19.4 | 3.0 | 8.2 |
| Baseline + GS | 33.1 | 18.7 | 2.8 | 7.4 |
| Baseline + LGS | 33.1 | 18.7 | 2.8 | 7.3 |
| Baseline + LHGNet + GS + head | 38.6 | 22.2 | 2.2 | 13.3 |
| Baseline + LHGNet + LGS + head (LE-YOLO) | 39.3 | 22.7 | 2.1 | 13.1 |

We performed another ablation experiment to assess the performance of the three introduced enhancements. Based on YOLOv8n, we proposed LHGNet as the backbone, presented the LGS bottleneck and LGSCSP fusion module, and optimized the detection head. Table 4 reveals that each method led to enhancements in the detection accuracy and efficiency.

**Table 4.** Influence of different enhancements evaluated on VisDrone2019-val.

| Model | mAP(0.5) (%) | mAP(0.5:0.95) (%) | The Number of Model Parameters ($\times 10^6$) | FLOPS ($\times 10^9$) |
|---|---|---|---|---|
| Baseline (YOLOv8n) | 33.9 | 19.4 | 3.0 | 8.2 |
| Baseline + LHGNet | 35.6 | 20.7 | 3.2 | 10.5 |
| Baseline + LGS | 33.1 | 18.7 | 2.8 | 7.3 |
| Baseline + head | 38.2 | 22.1 | 2.3 | 16.7 |
| Baseline + LGS + head | 36.6 | 20.9 | 1.9 | 10.6 |
| Baseline + LHGNet + LGS + head (LE-YOLO) | 39.3 | 22.7 | 2.1 | 13.1 |

To validate the performance of the LHG backbone, YOLOv8n was chosen as the baseline, and the LHG backbone was substituted for the original CSPDarkNet backbone. From rows 1 and 2 in Table 4, it can be observed that, compared to YOLOv8n, the mAP(0.5) is increased by 5.0% after implementing the LHG backbone. Considering the dominant presence of small-scale objects in UAV image detection, the proposed LHG backbone strategy achieves the desired improvement in accuracy.

To improve the efficiency in capturing tiny targets, we improved the detection head by modifying the location of the detection head branch. As can be seen from row 4 in Table 4, by adjusting the location of the detection head branch, we achieved improvements of 12.7% and 13.9% in the mAP(0.5) and mAP(0.5:0.95), respectively. Simultaneously, there was a 23% reduction in the model parameters. Therefore, it can be concluded that our approach of adjusting the detection head significantly improves the accuracy in detecting tiny objects and reduces the model parameters.

To validate the interaction of the various improvements, we replaced the LHG backbone, improved the C2f with the LGS bottleneck and LGSCSP fusion module in the neck, and made modifications to the detection head. As shown in rows 1 and 6 of Table 4, we were able to achieve gains of 15.9% and 15.5% in the mAP(0.5) and mAP(0.5:0.95), respectively. Simultaneously, our proposed LE-YOLO model yielded a 30% reduction in the model parameters, yet it had the highest mAP(0.5), thereby confirming that we can enhance the detection accuracy by employing our LE-YOLO algorithm to explore high-level semantic information.

Table 5 presents the comparison results for ten categories evaluated on VisDrone2019-val. The ten categories are PED, PER, BC, Car, Van, TR, TRI, ATR, Bus, and MO, representing pedestrians, people, bicycles, cars, vans, trucks, tricycles, awning tricycles, buses, and motors, respectively. The comparison results demonstrate the strong performance of the interactions among the various improvement methods on the VisDrone2019 dataset. Specifically, they significantly enhance the detection performance for tiny objects such as motors, pedestrians, and people.

**Table 5.** The comparison results of ten categories evaluated on VisDrone2019-val.

| Model | PED | PER | BC | Car | Van | TR | TRI | ATR | Bus | MO | mAP(0.5) (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baseline (YOLOv8n) | 35.6 | 29.1 | 7.9 | 76.2 | 39.2 | 31.2 | 21.6 | 11.0 | 48.9 | 38.4 | 33.9 |
| Baseline + LHGNet | 39.1 | 31.0 | 9.7 | 78.0 | 41.7 | 31.4 | 23.6 | 13.7 | 47.2 | 40.5 | 35.6 |
| Baseline + LGS | 35.4 | 28.1 | 7.5 | 75.7 | 38.3 | 27.5 | 21.7 | 12.4 | 47.3 | 37.1 | 33.1 |
| Baseline + head | 45.6 | 38.1 | 11.4 | 81.1 | 42.6 | 29.2 | 24.5 | 13.3 | 50.5 | 45.1 | 38.2 |
| Baseline + LGS + head | 43.6 | 37.0 | 11.4 | 80.4 | 40.9 | 28.6 | 21.8 | 13.5 | 45.1 | 43.9 | 36.6 |
| Baseline + LHGNet + LGS + head (LE-YOLO) | 46.7 | 39.1 | 12.0 | 82.3 | 43.1 | 31.4 | 24.5 | 13.8 | 52.9 | 46.8 | 39.3 |

### *4.4. Comparison Experiments*

To further assess its effectiveness, we compared LE-YOLO with other state-of-the-art detectors on the VisDrone2019 dataset. The detectors under comparison consisted of single-stage detectors, namely SSD [43], YOLOv5n, YOLOv6n, YOLOv7, YOLOv8n, and Drone-YOLO-n [44], along with the two-stage detector Faster RCNN [45]. Among them, YOLOv6n, YOLOv8n, and Drone-YOLO-n are anchor-free detectors [46], while the others are anchor-based detectors. All algorithms shared the same training parameters and testing parameters. The evaluation results for LE-YOLO and the other object detection models on VisDrone2019-val are presented in Table 6.

**Table 6.** Performance of LE-YOLO and other object detection models evaluated on VisDrone2019-val.

| Model | Backbone | mAP(0.5) (%) | mAP(0.5:0.95) (%) | $mAP_{tiny}$ (%) | The Number of Model Parameters ($\times 10^6$) | FLOPS ($\times 10^9$) |
|---|---|---|---|---|---|---|
| SSD | VGG16 | 9.2 | 5.0 | 0.5 | 58.0 | - |
| Faster RCNN | ResNet50 | 29.0 | 17.8 | 7.2 | 165.6 | - |
| YOLOv5n | CSPDarkNet53 | 32.9 | 18.6 | 9.0 | 2.5 | 7.2 |
| YOLOv6n | EfficientRep | 30.6 | 17.4 | 7.6 | 4.2 | 11.9 |
| YOLOv7 | Extended-ELAN | 29.1 | 17.9 | 8.6 | 37.3 | 105.3 |
| YOLOv8n | CSPDarkNet53 | 33.9 | 19.4 | 9.4 | 3.0 | 8.2 |
| Drone-YOLO-n | RepVGG | 38.1 | 22.7 | - | 3.1 | - |
| LE-YOLO | LHGNet | 39.3 | 22.7 | 13.5 | 2.1 | 13.1 |

For the VisDrone2019-val dataset, our LE-YOLO achieved a score of 39.3% for the mAP(0.5) at the model parameter number of 2.1 M, outperforming all compared models. In contrast to YOLOv8n, the introduced LE-YOLO model achieved a 30.0% reduction in the parameter count, accompanied by a 15.9% increase in the mAP(0.5). Additionally, compared to the YOLOv6n model, our LE-YOLO halved the parameter count, yet it achieved a 28.4% increase in the mAP(0.5). Although our approach was equal to Drone-YOLO-n in terms of the mAP(0.5:0.95), the mAP(0.5) was improved by 1.8%, and the model parameters were reduced by 32.3%, which was significantly better than in other methods. In terms of validating the model's ability to detect tiny objects, our LE-YOLO achieved a precision of 13.2% in terms of the $mAP_{tiny}$, significantly surpassing other models. Compared to the baseline YOLOv8n model, it exhibited a 43.6% enhancement in detecting tiny objects. The

comparison experiments demonstrate that our proposed model achieves superior detection accuracy in detecting various objects and tiny objects, while significantly reducing the number of model parameters.

To validate the model's detection capabilities for various categories, we conducted comparative experiments on the VisDrone2019-test dataset. The networks compared included Faster RCNN, RetinaNet [47], Cascade RCNN [48], and Improved-yolov4 [49], along with YOLOv5n, YOLOv6n, YOLOv8n, and Drone-YOLO-n [44]. Figure 10 presents the comparison results for the ten categories evaluated on the VisDrone2019-test dataset.

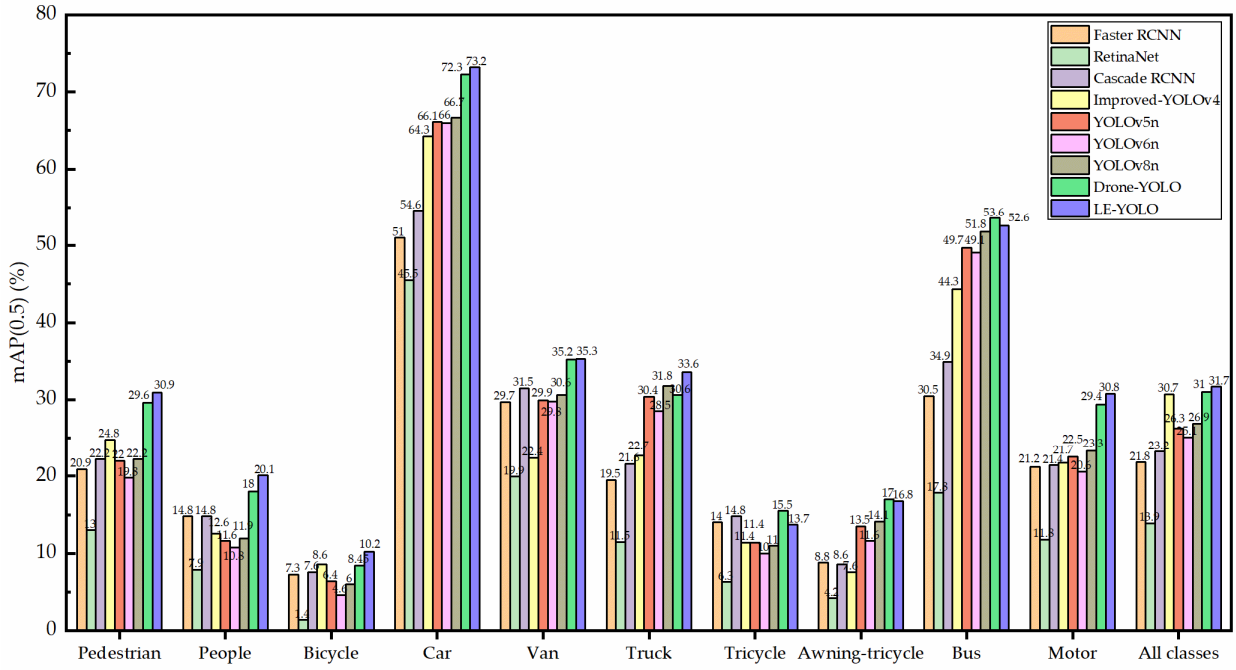
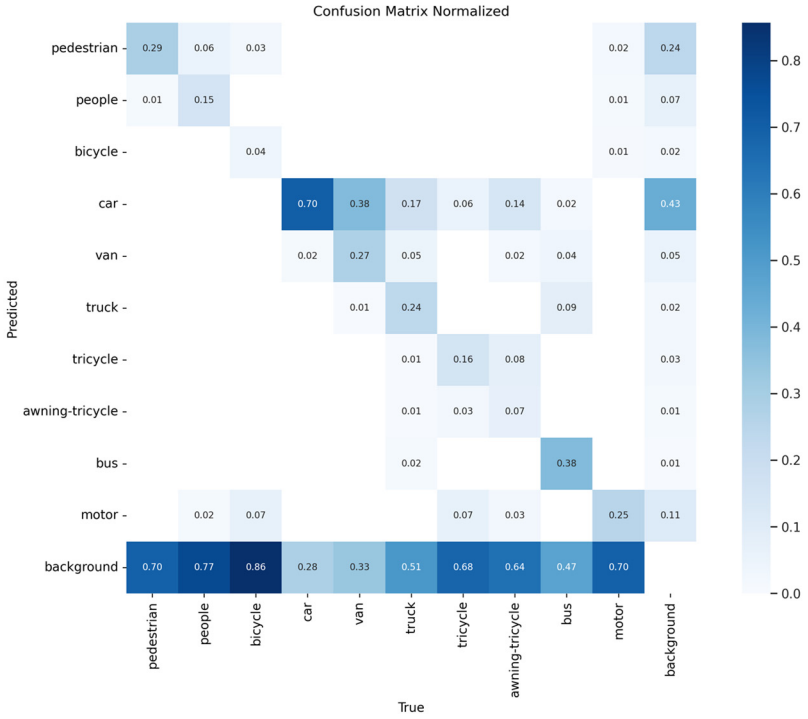


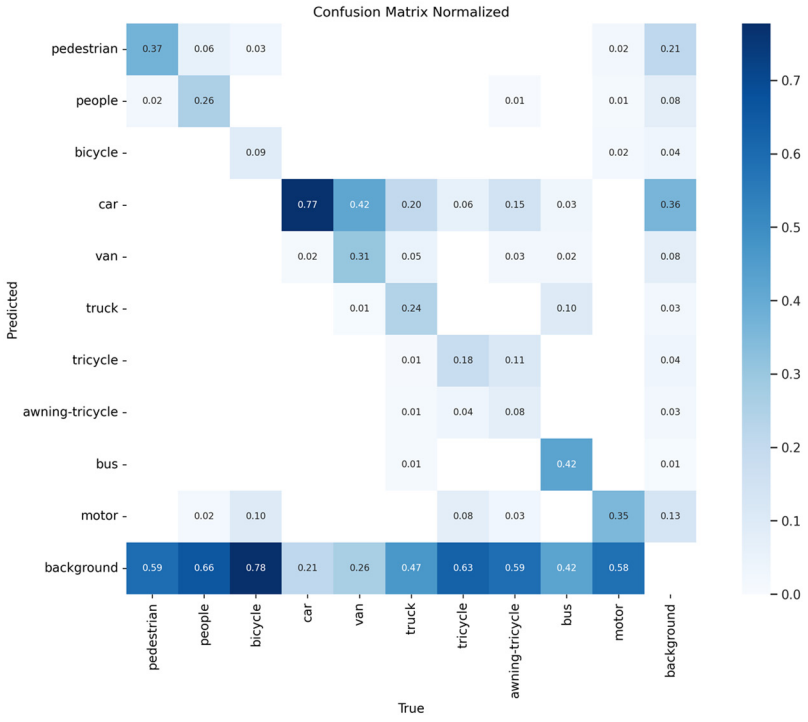

**Figure 10.** Comparison of different algorithms evaluated on VisDrone2019-test.

It can be seen from Figure 10 that our LE-YOLO outperformed all models except Drone-YOLO-n in terms of detection accuracy. Compared to Drone-YOLO-n, our LE-YOLO achieved superior detection accuracy across seven object categories, including pedestrians, people, bicycles, cars, and others. Most of these object categories consisted of tiny objects, confirming our model's proficiency in detecting such tiny objects. However, for tricycles, awning tricycles, and buses, our model's detection performance was slightly poorer than that of Drone-YOLO-n. This may have been due to the removal of the detection head for large objects, but this minor loss is acceptable considering the significant reduction in the detection parameters.

Figure 11 shows the confusion matrix diagrams among the ten categories of YOLOv8n and LE-YOLO. Our algorithm performs better in the recognition of all categories compared to YOLOv8n, especially showing significant improvements in the recognition of tiny targets such as motors, pedestrians, and people. Furthermore, despite both algorithms exhibiting relatively high rates of missed detection, our algorithm reduces the probability of identifying each category as the background.




(**a**)




(**b**)

**Figure 11.** Confusion matrix diagram of (**a**) YOLOv8n and (**b**) LE-YOLO.

In order to ascertain whether the LE-YOLO algorithm applies to low-power edge devices, we conducted experiments on the Jetson Nano B01 platform, with the results shown in Table 7. As the images captured by current drone cameras were similar to and compatible with those in the dataset, we continued to conduct the embedded experiments on VisDrone2019-val. As shown in Table 7, although our algorithm achieved the highest detection accuracy in both mAP(0.5) and mAP(0.5:0.95), it was relatively slower in average

inference time per image. This is primarily because our algorithm has a slightly higher FLOPS, requiring more floating-point computations. However, it remains suitable for use on low-power edge devices and fulfills the requirements for regular inspections.

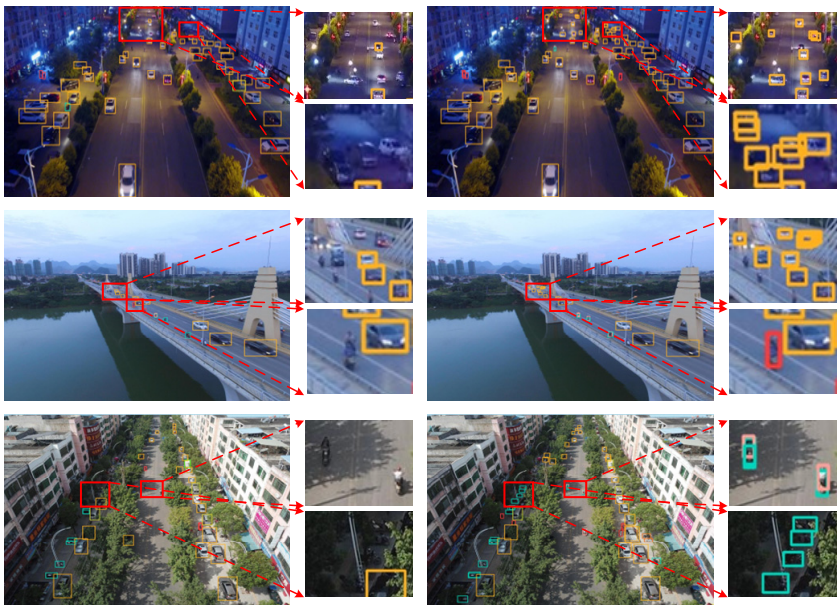
**Table 7.** Embedded experiments performance of LE-YOLO and other object detection models evaluated on VisDrone2019-val.

| Model | mAP(0.5) (%) | mAP(0.5:0.95) (%) | Average Inference Time per Image (ms) | The Number of Model Parameters ($\times 10^6$) | FLOPS ($\times 10^9$) |
|---|---|---|---|---|---|
| YOLOv5n | 32.9 | 18.6 | 127.5 | 2.5 | 7.2 |
| YOLOv6n | 30.6 | 17.4 | 174.8 | 4.2 | 11.9 |
| YOLOv8n | 33.9 | 19.4 | 141.7 | 3.0 | 8.2 |
| LE-YOLO | 39.3 | 22.7 | 338.6 | 2.1 | 13.1 |

### *4.5. Visualization*

To fully showcase the applicability of LE-YOLO in diverse aerial scenarios and perspectives, we provide object detection results obtained in typical environmental conditions from the VisDrone2019-test dataset. In the VisDrone2019-test dataset, there are considerable differences in the sizes of the targets, most of which are tiny objects. Simultaneously, the image background is intricate and diverse, making it challenging to distinguish objects from their surroundings. Furthermore, some objects are occluded, posing difficulties for detection.

The detection results of YOLOv8n and LE-YOLO are illustrated in Figure 12. Column (a) displays the detection results generated by the YOLOv8n network, while column (b) presents the detection results generated by our proposed LE-YOLO network. The comparison shows that LE-YOLO performs better in identifying occluded objects without introducing false positives, distinguishing itself from YOLOv8n, which may exhibit incorrect detection results and overlook certain objects.



(**a**) (**b**)

**Figure 12.** The detection results of (**a**) YOLOv8n and (**b**) LE-YOLO. In the figure, the yellow, red, grass green, pink, and cyan prediction boxes represent the predicted objects as cars, pedestrians, vans, people, and motors respectively.

## 5. Conclusions

To address the challenges of detecting tiny objects for UAV images using onboard edge devices, this research introduces the LE-YOLO framework for tiny-object detection based

on YOLOv8n. Specifically, this model constructs the LHGNet backbone by incorporating depth-wise separable convolution and channel shuffle modules. The aim is to explore and leverage the local detail information and channel features in the upper and lower spaces, enhancing the detection accuracy for tiny targets and achieving a lightweight model. Meanwhile, the LGS bottleneck and LGSCSP fusion module are designed to further optimize and lighten the model. Given the characteristics of multi-scale small targets in UAV images, we modify the structure and the size of the feature maps to enhance the detection accuracy in capturing tiny targets. The comprehensive experiments indicate that our approach can significantly enhance the detection accuracy and make the model lightweight by exploring and leveraging the features in the local detail information and channel domains through the organic combination of depth-wise separable convolution and standard convolution. Additionally, LE-YOLO is composed entirely of convolutional operations, making it highly suitable for application on edge devices equipped with GPUs.

In future work, we plan to deploy LE-YOLO on other advanced low-power edge devices and optimize the model's algorithm by employing attention mechanisms and model compression techniques.

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

**Funding:** This research was supported by the National Natural Science Foundation of China, grant number 52375503.

**Data Availability Statement:** The data presented in this study are available upon request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Xue, Z.; Xu, R.; Bai, D.; Lin, H. YOLO-Tea: A Tea Disease Detection Model Improved by YOLOv5. *Forests* **2023**, *14*, 415. [CrossRef]
2. Zhou, X.; Xu, X.; Liang, W.; Zeng, Z.; Shimizu, S.; Yang, L.T.; Jin, Q. Intelligent Small Object Detection for Digital Twin in Smart Manufacturing With Industrial Cyber-Physical Systems. *IEEE Trans. Ind. Inform.* **2022**, *18*, 1377–1386. [CrossRef]
3. Guo, Y.; Chen, S.; Zhan, R.; Wang, W.; Zhang, J. LMSD-YOLO: A Lightweight YOLO Algorithm for Multi-Scale SAR Ship Detection. *Remote Sens.* **2022**, *14*, 4801. [CrossRef]
4. Yang, Y.; Miao, Z.; Zhang, H.; Wang, B.; Wu, L. Lightweight Attention-Guided YOLO With Level Set Layer for Landslide Detection From Optical Satellite Images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 3543–3559. [CrossRef]
5. Zhu, Y.; Tang, H. Automatic Damage Detection and Diagnosis for Hydraulic Structures Using Drones and Artificial Intelligence Techniques. *Remote Sens.* **2023**, *15*, 615. [CrossRef]
6. Zhou, K.; Zhang, M.; Wang, H.; Tan, J. Ship Detection in SAR Images Based on Multi-Scale Feature Extraction and Adaptive Feature Fusion. *Remote Sens.* **2022**, *14*, 755. [CrossRef]
7. Zhu, J.; Pang, Q.; Li, S.; Tian, S.; Li, J.; Li, Y. ADDet: An Efficient Multiscale Perceptual Enhancement Network for Aluminum Defect Detection. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 5004714. [CrossRef]
8. Yuan, M.; Zhou, Y.; Ren, X.; Zhi, H.; Zhang, J.; Chen, H. YOLO-HMC: An Improved Method for PCB Surface Defect Detection. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 2001611. [CrossRef]
9. Xu, H.; Zhong, S.; Zhang, T.; Zou, X. Multiscale Multilevel Residual Feature Fusion for Real-Time Infrared Small Target Detection. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5002116. [CrossRef]
10. Liu, C.; Yang, D.; Tang, L.; Zhou, X.; Deng, Y. A Lightweight Object Detector Based on Spatial-Coordinate Self-Attention for UAV Aerial Images. *Remote Sens.* **2023**, *15*, 83. [CrossRef]
11. Yue, X.; Meng, L. YOLO-SM: A Lightweight Single-Class Multi-Deformation Object Detection Network. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 2467–2480. [CrossRef]
12. Yang, X.; Zhang, J.; Chen, C.; Yang, D. An Efficient and Lightweight CNN Model With Soft Quantification for Ship Detection in SAR Images. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5230713. [CrossRef]
13. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
14. YOLOv8. Available online: https://github.com/ultralytics/ultralytics (accessed on 15 November 2023).
15. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2016**, arXiv:1610.02357. [CrossRef]

16. 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), Seattle, WA, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA; pp. 779–788.
17. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Piscataway, NJ, USA; pp. 6517–6525.
18. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
19. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
20. Wang, C.-Y.; Liao, H.-Y.M.; Yeh, I.-H.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *arXiv* **2019**, arXiv:1911.11929. [CrossRef]
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *arXiv* **2014**, arXiv:1406.4729. [CrossRef]
22. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. *arXiv* **2016**, arXiv:1612.03144. [CrossRef]
23. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. *arXiv* **2018**, arXiv:1803.01534. [CrossRef]
24. Misra, D. Mish: A Self Regularized Non-Monotonic Activation Function. *arXiv* **2019**, arXiv:1908.08681. [CrossRef]
25. Yolov5. Available online: https://github.com/ultralytics/yolov5 (accessed on 15 March 2023).
26. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976. [CrossRef]
27. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696. [CrossRef]
28. Xu, D.; Wu, Y. Improved YOLO-V3 with DenseNet for Multi-Scale Remote Sensing Target Detection. *Sensors* **2020**, *20*, 4276. [CrossRef] [PubMed]
29. Xu, Q.; Lin, R.; Yue, H.; Huang, H.; Yang, Y.; Yao, Z. Research on Small Target Detection in Driving Scenarios Based on Improved Yolo Network. *IEEE Access* **2020**, *8*, 27574–27583. [CrossRef]
30. Qiu, Z.; Bai, H.; Chen, T. Special Vehicle Detection from UAV Perspective via YOLO-GNS Based Deep Learning Network. *Drones* **2023**, *7*, 117. [CrossRef]
31. Cao, J.; Bao, W.; Shang, H.; Yuan, M.; Cheng, Q. GCL-YOLO: A GhostConv-Based Lightweight YOLO Network for UAV Small Object Detection. *Remote Sens.* **2023**, *15*, 4932. [CrossRef]
32. Xie, S.; Zhou, M.; Wang, C.; Huang, S. CSPPartial-YOLO: A Lightweight YOLO-Based Method for Typical Objects Detection in Remote Sensing Images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 388–399. [CrossRef]
33. Luo, Y.; Ci, Y.; Jiang, S.; Wei, X. A novel lightweight real-time traffic sign detection method based on an embedded device and YOLOv8. *J. Real-Time Image Process.* **2024**, *21*, 24. [CrossRef]
34. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556. [CrossRef]
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385. [CrossRef]
36. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2016**, arXiv:1608.06993. [CrossRef]
37. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. *arXiv* **2016**, arXiv:1611.05431. [CrossRef]
38. Lee, Y.; Hwang, J.-W.; Lee, S.; Bae, Y.; Park, J. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. *arXiv* **2019**, arXiv:1904.09730. [CrossRef]
39. PaddlePaddle. HGNetv2. Available online: https://github.com/PaddlePaddle/PaddleDetection/blob/develop/ppdet/modeling/backbones/hgnet_v2.py (accessed on 15 November 2023).
40. Lv, W.; Zhao, Y.; Xu, S.; Wei, J.; Wang, G.; Cui, C.; Du, Y.; Dang, Q.; Liu, Y. DETRs Beat YOLOs on Real-time Object Detection. *arXiv* **2023**, arXiv:2304.08069. [CrossRef]
41. Li, H.; Li, J.; Wei, H.; Liu, Z.; Zhan, Z.; Ren, Q. Slim-neck by GSConv: A better design paradigm of detector architectures for autonomous vehicles. *arXiv* **2022**, arXiv:2206.02424. [CrossRef]
42. Du, D.; Zhu, P.; Wen, L.; Bian, X.; Lin, H.; Hu, Q.; Peng, T.; Zheng, J.; Wang, X.; Zhang, Y.; et al. VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019; pp. 213–226.
43. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference 2016, Part I 14, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
44. Zhang, Z. Drone-YOLO: An Efficient Neural Network Method for Target Detection in Drone Images. *Drones* **2023**, *7*, 526. [CrossRef]
45. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]

46. Fu, J.; Sun, X.; Wang, Z.; Fu, K. An Anchor-Free Method Based on Feature Balancing and Refinement Network for Multiscale Ship Detection in SAR Images. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 1331–1344. [CrossRef]
47. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002. [CrossRef]
48. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into High Quality Object Detection. *arXiv* **2017**, arXiv:1712.00726. [CrossRef]
49. Ali, S.; Siddique, A.; Ates, H.F.; Gunturk, B.K. Improved YOLOv4 for Aerial Object Detection. In Proceedings of the 29th IEEE Conference on Signal Processing and Communications Applications (SIU), Electr Network 2021, Istanbul, Turkey, 9–11 June 2021; IEEE: Piscataway, NJ, USA.