electronics

MDPI

Article

Fast and Accurate Detection of Dim and Small Targets for Smart Micro-Light Sight

Jia Wei [1], Kai Che [1], Jiayuan Gong [2], Yun Zhou [1,*], Jian Lv [1], Longcheng Que [1], Hu Liu [3] and Yuanbin Len [4]

[1] College of Optoelectronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; weijia@std.uestc.edu.cn (J.W.); chekaiyc@std.uestc.edu.cn (K.C.); lvjian@uestc.edu.cn (J.L.); lcque@uestc.edu.cn (L.Q.)
[2] Institute of Automotive Engineers, Hubei University of Automotive Technology No. 167, Shiyan 442000, China; jygong@huat.edu.cn
[3] Xi'an Institute of Applied Optics, Xi'an 710065, China; 13991971532@126.com
[4] Chengdu Dingyi Information Technology Co., Chengdu 611731, China; leng.yuanbin@dingytech.net
* Correspondence: zhou.yun@dingytech.net

Abstract: To deal with low recognition accuracy and large time-consumption for dim, small targets in a smart micro-light sight, we propose a lightweight model DS_YOLO (dim and small target detection). We introduce the adaptive channel convolution module (ACConv) to reduce computational redundancy while maximizing the utilization of channel features. To address the misalignment problem in multi-task learning, we also design a lightweight dynamic task alignment detection head (LTD_Head), which utilizes GroupNorm to improve the performance of detection head localization and classification, and shares convolutions to make the model lightweight. Additionally, to improve the network's capacity to detect small-scale targets while maintaining its generalization to multi-scale target detection, we extract high-resolution feature map information to establish a new detection head. Ultimately, the incorporation of the attention pyramid pooling layer (SPPFLska) enhances the model's regression accuracy. We conduct an evaluation of the proposed algorithm DS_YOLO on four distinct datasets: CityPersons, WiderPerson, DOTA, and TinyPerson, achieving a 66.6% mAP on the CityPersons dataset, a 4.3% improvement over the original model. Meanwhile, our model reduces the parameter count by 33.3% compared to the baseline model.

Keywords: dim and small target detection; micro-light sight; lightweight; task alignment

Citation: Wei, J.; Che, K.; Gong, J.; Zhou, Y.; Lv, J.; Que, L.; Liu, H.; Len, Y. Fast and Accurate Detection of Dim and Small Targets for Smart Micro-Light Sight. *Electronics* **2024**, *13*, 3301. https://doi.org/10.3390/electronics13163301

Academic Editors: Stefanos Kollias and José Carlos Castillo

Received: 30 May 2024
Revised: 24 July 2024
Accepted: 10 August 2024
Published: 20 August 2024

1. Introduction

Micro-light sight, serving as the primary equipment for night hunting, is crucial for night-time observation and targeting. This paper aims to integrate target detection technology into smart micro-light sight (see Figure 1), and addressing challenges such as low recognition accuracy and large time-consumption for dim and small targets, thereby driving the intelligent optimization of micro-light sight.

One of the main problems in the science of computer vision is object detection, which attempts to identify all items of interest by figuring out their categories and locations [1]. Nowadays, object detection is widely used in both military and civilian contexts, including autonomous driving [2–4], video surveillance [5,6], and human–computer interaction [7,8]. Dim and small target detection, as a crucial technology within object detection, has recently drawn attention from researchers. Objects at longer distances often exhibit characteristics of small objects. SPIE defines small objects as those with fewer than 80 pixels in a 256 × 256 pixel image [9]. Due to their limited pixel count and less prominent features, small objects exhibit lower detection rates and higher false alarm rates compared to larger objects. Thus, dim and small target recognition continues to be a difficult study topic. The main challenges in detecting dim and small targets [10] include:

- Insufficient features: Small-scale objects cover fewer pixels, retaining less information. The feature representation is weak, and in typical application scenarios of object detection technology, small-scale object instances are accompanied by problems such as low resolution and blurred background.
- Information loss: After multiple convolution and pooling operations in convolutional neural networks, inevitable semantic information loss of small-scale objects occurs. Meanwhile, feature maps contain more unnecessary background information in small-scale objects, weakening the feature representation of small-scale objects.
- Complex background: In urban streets, parks, and other scenes, various complex backgrounds may exist around objects, including buildings, trees, vehicles, etc. These backgrounds can interfere with object detection.
- Difficult detection and localization: Due to their small size, the position of small-scale objects has more possibilities, such as corners or overlapping areas with other objects. Additionally, it can be challenging to distinguish small-scale objects in complicated scene surfaces from noise clutter and accurately locate their boundaries, which means that higher precision is required for positioning during detection.

When tackling these issues, it is critical to take into account the limited computational resources in practical applications. Therefore, efficient algorithms need to be designed to accomplish the task of detecting dim and small targets under limited resources. In this work, we propose three lightweight and efficient modules alongside innovative architectural designs, leading to the development of a novel network, DS_YOLO. The following are this work's main contributions:

(1) We propose a lightweight dynamic task alignment detection head (LTD_Head) to address the misalignment issue between classification and localization tasks during prediction, caused by differences in feature spatial distribution. This aims to better coordinate classification and localization tasks, resulting in more accurate and consistent prediction outcomes.
(2) We introduce an additional detection layer into the network architecture to extract detailed deep features of dim and small objects, addressing the issue of information loss while improving the network's generalization capability for multi-scale object detection.
(3) We propose an adaptive channel convolution module (ACConv), which reduces parameters and computational load significantly by calculating only the channels with larger weights (selected by SE module), thereby reducing redundant feature computation and improving network efficiency.
(4) We introduce a large separable kernel attention module into the pyramid feature layer to help the model dynamically adjust the weights of feature mappings. This allows the model to focus more on the feature regions that are more important for the current task.

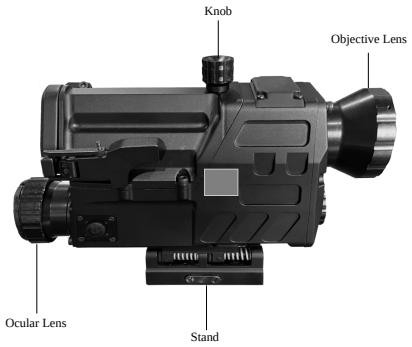


Figure 1. Self-developed micro-light sight.

2. Related Works

2.1. Traditional Object Detection Methods

The three primary steps of traditional object detection algorithms are feature extraction, classification, and region proposal (sliding window). The process of choosing a subset of data with particular area features from datasets in domains like machine learning is commonly referred to as region proposal. Feature extraction involves obtaining from initial data representative features for use in further data analysis and machine learning tasks. Classification mostly entails data analysis and learning. Decision trees are usually used in region proposal. Feature extraction is carried out using methods like the HOG (historical oriented gradient), deformable part models [11] and local binary patterns (LBPs) [12]. The HOG divides the image into small connected regions (cells), calculates the histogram of gradient directions within each cell, and then normalizes and combines these histograms to form the feature vector of the image. DPM is a part-based object detection method that captures object shape variations by decomposing the object into multiple deformable parts. Each part is represented by a linear support vector machine (SVM) model based on HOG features and is connected through elastic links. LBP is an operator for texture description that generates binary patterns by comparing each pixel with its neighboring pixels. Subsequently, classifiers like SVM [13] and AdaBoost [14] are used to classify the extracted features. SVM finds an optimal hyperplane that maximizes the margin between different classes. This hyperplane separates the data points of different classes, ensuring that the data points are as far from the hyperplane as possible. AdaBoost creates a strong classifier by combining multiple weak classifiers, such as decision stumps. Weak classifiers are those that perform slightly better than random guessing when applied individually. Felzenszwalb et al. proposed using deformable part models to handle variations in target feature. Selecting search ensures variation in the search process and boosts efficiency by combining segmentation and exhaustive search [15]. Oxford-mlk combines the benefits of cascaded support vector machines and oriented gradient feature histograms to detect [16]. Nlpr-hogllbp proposes a context-based object detection method that improves detection performance by considering the relationship between the object and its surrounding environment. Additionally, the paper introduces an enhanced HOG-LBP feature, which combines HOG and LBP to improve the accuracy of object detection [17]. A self-learning pedestrian dependability detection system was suggested by Liu et al. [18]; the main innovation lies in proposing a self-learning framework for pedestrian detection. This framework can adapt any offline-trained detector to specific scenes, achieving better performance. After locating the motion regions in the scene, features are extracted from the motion regions using support vector machines. Lastly, in the classification stage, self-learning is used to rectify targets that were incorrectly classified.

2.2. Deep Learning Object Detection Methods

In general, there are two types of neural network object detection algorithms. One is a one-stage algorithm that directly outputs the target location and category. The YOLO, first proposed by Redmon et al. [19], is a typical representative of the one-stage algorithm. Since then, the algorithm has evolved continuously, and has successively derived several versions such as YOLOv2 [20], YOLOv3 [21], YOLOv4 [22], YOLOv5 [23], YOLOv7 [24], etc.; each of them have different optimizations and enhancements in detection accuracy and speed. The second sort is the two-stage algorithm, which produces region suggestions initially followed by performing classification and localization. Representative examples of this category include region-based Faster R-CNN and Fast R-CNN [25].

The YOLO networks use the whole picture as input, and output the regression target box's class and location. This significantly improves detection speed. YOLOv1 pioneered the transformation of object detection into a regression problem, predicting bounding boxes and class probabilities directly through a single neural network model. YOLOv2 introduced anchor boxes and passthrough layers, enhancing the detection capability for small objects. YOLOv3 adopted multi-scale prediction and feature pyramid network (FPN),

improving detection accuracy for objects of different sizes. YOLOv4 made multiple improvements in data augmentation, loss functions, and network architecture, enhancing performance and generalization ability. YOLOv5 primarily optimized network structure and loss functions, introducing new techniques such as adaptive anchor box calculation and positive sample matching strategies. YOLOv6 added new strategies for handling large-scale data processing and further refined anchor box handling. YOLOv7 introduced advanced technologies including the efficient latent attention network (ELAN), reparameterized convolution, and dynamic label assignment. By combining the outputs from several layers, Gong et al. [26] presented an enhanced feature-fusion-based detection approach that improves both recognition speed and accuracy. The convolutional block attention module, that Woo et al. [27] developed, improves channel and spatial information and, hence, increases the feature extraction capacity. Hu et al. [28] proposed the squeeze-and-excitation (SE) module, which explicitly models the interdependence between convolutional feature channels. The Swin-Transformer module, which outputs multi-scale feature information by using distinct downsampling feature maps from successive stages, was introduced by Liu et al. [29]. Chen et al. [30] constructed hierarchical residual connectivity within residual blocks to express multi-scale features at finer granularity levels. A pedestrian detection algorithm based on small sample datasets was proposed by Xu et al. [31]. They increased the dataset size using patch data augmentation methods. This method improved the model's overall performance without changing the existing dataset and network size. A transformer-based multi-scale feature fusion detection network was created by Chen and Guo [32]. The transformer improves pedestrian detection during the detection stage by capturing global information and successfully addresses long-distance reliance mechanisms among picture pixels. The attained detection accuracy was 78.5%.

2.3. Small Object Detection Methods

One common issue with existing detection frameworks is the recognition of small targets. Within the domain of detecting tiny faces, Bai et al. [33] proposed using a super-resolution network to upsample blurry low-resolution images into finely detailed high-resolution images, aiming to enhance spatial information in advance. Later, a multi-task generative adversarial network was suggested by Bai et al. [34] to restore detailed information for more precise detection. Despite their impressive performance, they experience a heavy computational load because of the introduction of additional super-resolution networks.

The existing models either exhibit low recognition rates for dim and small targets or cannot be effectively deployed due to the large computational complexity and parameter size.

3. Methodology

3.1. Overall Architecture

The four primary parts of the DS_YOLO algorithm are the input, backbone, neck, and head. The DS_YOLO is shown in Figure 2. The feature extraction and feature fusion networks were redesigned based on the characteristics of small targets. Specifically, we developed a dedicated detection layer tailored for dim and small objects, enhancing the fusion of information across different stages. To ensure the model's generalization capability and robustness, we retained the detection layer for large objects. This method preserves the accuracy of general size object detection while simultaneously increasing the accuracy of small object detection. Next, a novel convolutional module, ACConv, is designed to extract representative channel features for deeper computation, with the remaining channel features serving as supplementary information for subtle details. This not only ensures detection accuracy but also drastically lowers the computing burden and quantity of parameters. For the misalignment between target classification and detection tasks, the LTD_Head is proposed as a lightweight dynamic task alignment detection head. It utilizes interactive features to align classification and detection tasks, thereby improving detection accuracy. The model becomes lighter by dramatically reducing the number of parameters through the utilization of shared convolution. Lastly, an innovative design is applied to the

SPPF module to create the SPPFLska module, addressing semantic conflicts at the pixel level that arise when combining different-sized pooling layers. This prevents the loss of important feature information.

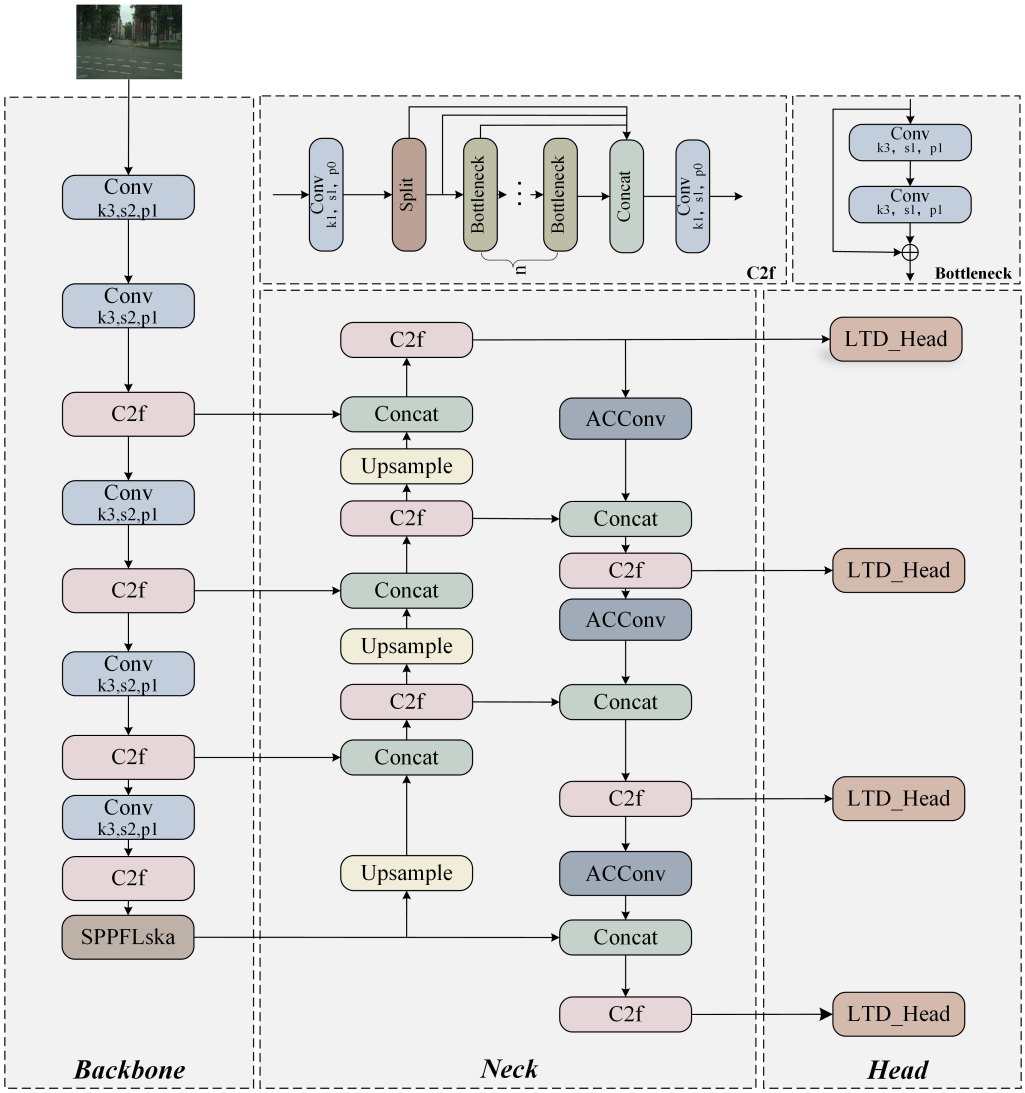


Figure 2. The structure of DS_YOLO.

3.2. Adaptive Channel Convolution (ACConv)

We observed a high degree of similarity between feature maps from different channels. However, it is challenging to determine whether these highly similar features are redundant or contain important details. Therefore, we designed a plug-and-play segmentation-based convolution operation module called ACConv, which selects representative channel features for deeper computation, instead of directly removing uncertain redundant features. This method lowers memory usage and processing burden in addition to effectively preventing the model from overfitting due to learning redundant information. Specifically, we compute the weights of each channel, where F represents the input features. After computing the channel weights through the SE module, we multiply these weights by the original input features:

$$F' = SE(F) \bigotimes F \tag{1}$$

We sort the input feature maps based on the channel weights, dividing them into representative part C_p channels and uncertain redundant part $(C - C_p)$ channels, where the size of C_p is determined according to the specific application requirements:

$$C_p = C/div \tag{2}$$

The only FLOPs of ACConv are

$$FLOPs = h \times w \times k_c^2 \times C_p^2 + (k_p - 1) \times (w/stride)^2 + C_p^2 \tag{3}$$

Through deeper computation, intrinsic information is extracted from the representative part, while the uncertain redundancy remains untouched, preserving subtle hidden details, as shown in Figure 3. Simply removing the remaining $(C - C_p)$ channels would deviate from our goal of reducing redundant features, turning ACConv into a convolution with fewer channels. By only selecting partial channels for convolution, the number of parameters in each convolutional kernel can be greatly decreased. This implies that the number of weights that must be kept in the model will be significantly decreased, thereby saving storage space, reducing the quantity of parameters, and lowering computational costs. This implies that we can deploy the model on smaller devices or run the model more efficiently in resource-constrained environments, which is particularly useful for mobile devices, embedded systems, or edge computing scenarios.

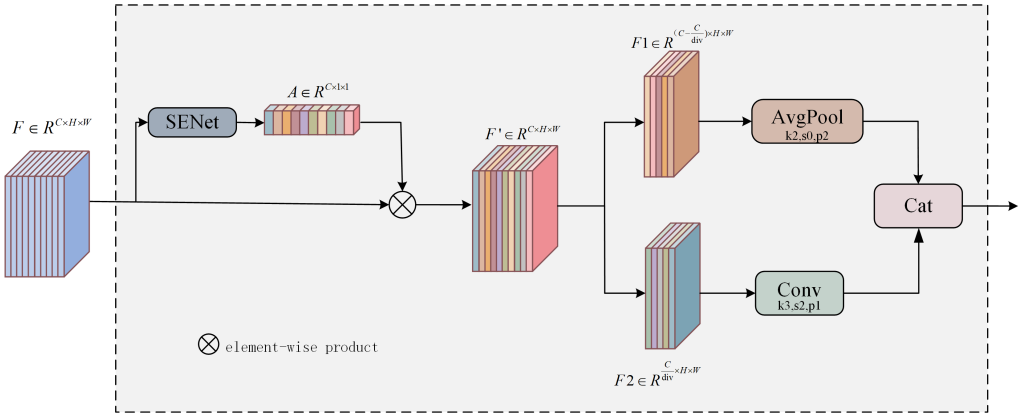


Figure 3. The structure of the ACConv module. SENet utilizes a channel attention mechanism. After computing the weights for each channel to obtain F', it selects the top-ranked channels $F2$ for convolutional operations, while the remaining channels $F1$ undergo average pooling. Subsequently, the features from these two parts are separated, processed, and concatenated to generate a new feature representation.

3.3. Lightweight Task Align Dynamic Detect Head (LTD_Head)

The classification task focuses on learning to discriminate features, while the localization task works on accurately locating the whole object and its boundaries. The classification and localization tasks' distinct learning methods mean that when predicting through two independent branches, there may be some misalignment because of differences in the geographical distribution of the traits that were learned from the two tasks. To address this issue, we drew inspiration from the task-oriented object detection (TOOD) [35] concept and designed the LTD_Head.

Specifically, each element $Input_i$ from the feature list Input is sequentially passed into the LTD_Head module. Firstly, through the feature extractor, the module uses several convolutional layers to learn task-interaction characteristics, resulting in the interaction feature $Feat_i$. To better decompose the tasks of target localization and target classification, the decomposition weights w_i for each task are computed based on $Feat_i$:

$$w_i^{task} = \sigma(sconv2(\delta(sconv1(Feat_i)))), \forall i \in \{1, 2, ..., N\} \tag{4}$$

Here, $sconv1$ and $sconv2$ represent stacked convolutions, mapping features to a lower-dimensional feature space to enhance the model's nonlinear expressive capability. δ represents ReLU, σ represents sigmoid, and N denotes the incoming feature list's size, i.e., the number of input features. Interacting features inevitably introduce certain feature conflicts between two different tasks. Therefore, specific features for each classification or localization task are calculated separately, by multiplying their respective weights and interacting features $Feat_i$, to obtain the decomposed features for each task:

$$Feat_i^{task} = Feat_i \bigotimes w_i^{task} \tag{5}$$

Next, we compute the spatial probability map (prob) for performing alignment in the classification task:

$$Prob_i = \sigma(conv2(\delta(conv1(Feat_i^{cls})))) \tag{6}$$

where $conv1$ and $conv2$ represent regular 2D convolutions, followed by adjusting the classification prediction task using the spatial probability map to obtain the aligned results:

$$Z_i^{cls} = Prob_i * Feat_i^{task} \tag{7}$$

For the localization task, we calculate the spatial offset map for the interaction feature $Feat_i$, resulting in the localization task result as shown in Equation (8):

$$Z_i^{box} = s(conv3(dy(Feat_i^{box}, offset))) \tag{8}$$

dy represents dynamic convolution, $conv3$ represents ordinary 2D convolution, s represents scaling, the ultimate outcome is derived by concating the outputs from these two distinct tasks. This module reduces the model parameter volume by using multiple shared convolutions, making the model lighter, and adopts group normalization to enhance the performance of detection head localization and classification. Then, in the localization branch, LTD_Head uses DyDCNV2 and interactive feature generation to obtain offsets and masks to achieve accurate object localization. Meanwhile, in the classification branch, LTD_Head utilizes interactive features for dynamic feature selection to enhance the manifestations of the classification task, making the model more efficient and accurate in practical applications. The structure is shown in Figure 4.

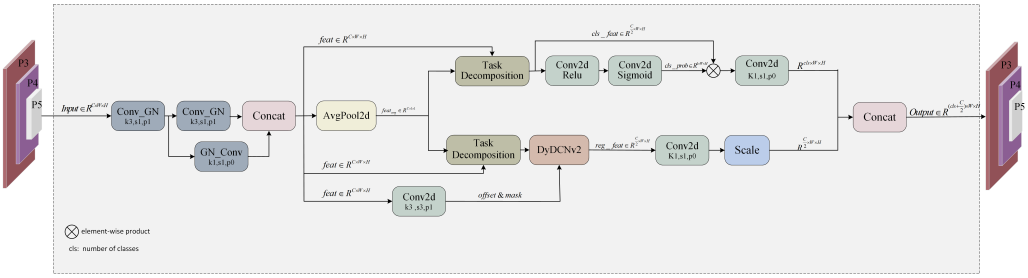


Figure 4. The structure of the LTD_Head module.

3.4. Spatial Pyramid Pooling-Faster with Large Separable Kernel (SPPFLska)

In traditional convolutional neural networks, pooling layers typically use fixed pooling levels and fixed pooling sizes. This approach leads to information loss for images of different sizes, thus affecting the model's accuracy. The SPPF spatial pyramid pooling method can adaptively pool input images of different sizes, thereby better preserving image information. However, during the pyramid pooling process, reducing computational complexity comes at the cost of inadequate capture of fine details in images. Combining different sizes of pooling levels can also lead to semantic conflicts at the pixel level, thereby erasing some important feature information.

Therefore, this method introduces the large separable kernel attention module [36], which splits the depth convolutional layer's 2D convolution kernel into cascaded one-

dimensional convolution kernels in the horizontal and vertical directions. This eliminates the need for extra blocks and permits the use of deep convolutional layers with huge convolution kernels in the attention module, and reduces computational complexity and memory usage. This module helps the model dynamically adjust the weights of feature maps, facilitating the model to focus more on feature regions that are more crucial for the job at hand. Combining the spatial pyramid pooling method allows for feature extraction on various scales, increasing the model's capacity for perception. This enables it to better capture important information at different scales and positions within images, thus raising the accuracy and robustness of the model. The computation for the parameter volume of the attention calculation module LSKA is Equation (9), indicating that this is a low-cost attention mechanism.

$$Param = C \times (2d-1) \times 2 + C \times [\frac{k}{d}] \times 2 + C \times C \tag{9}$$

As the convolution kernel's size grows, SPPFLska tends to focus more on the shape of objects rather than the texture. Figure 5 provides an illustration of the SPPFLska module.

The final output feature F'' is represented by the following equation, where cat stands for concatenation, dwd for dilated depthwise convolution, dw for standard depthwise convolution, and sp for SPPF:

$$F' = conv(dwd^2(dw^2(sp(F)))) \tag{10}$$

$$F'' = conv(cat(F', sp(F))) \tag{11}$$

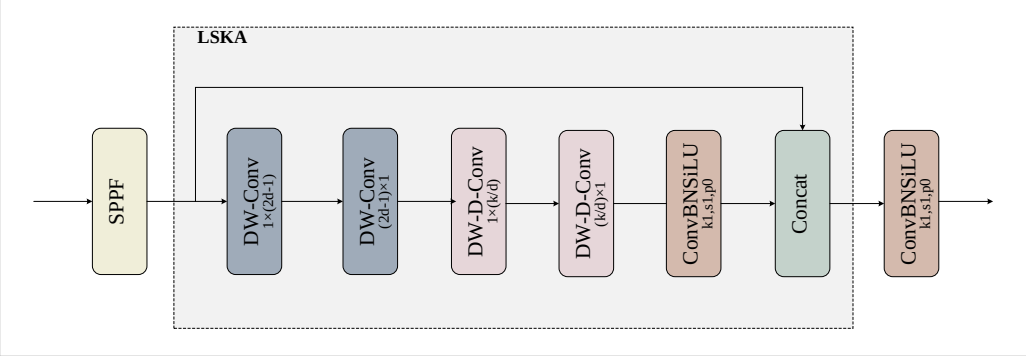


Figure 5. The structure of the SPPFLska module.

4. Experimental Results and Analysis

4.1. Experimental Settings and Dataset

The experimental setup consists of the PyTorch neural network framework and the Ubuntu 20.04 operating system. Refer to Table 1 for further details about the experimental setup in detail. Table 2 contains the parameters that were used in the training process.

The experiment is conducted on the CityPersons, WiderPerson, DOTA, and TinyPerson datasets. CityPersons [37] is a subset of the Cityscape dataset, and for each frame in the finely annotated subset of 5000 frames, CityPersons has created high-quality bounding box annotations for pedestrians. A total of 1575 photos are used for testing, 500 for validation, and 2975 for training. The CityPersons dataset documents various scenes from several cities and countries in Europe. The pictures are 2048 × 1024 in resolution. Additionally, the dataset includes labels for four categories: pedestrian, rider, sitting person, and other person. These are uniformly categorized as the "person" class for training purposes. WiderPerson [38] images are annotated with five types of annotations. This dataset contains densely populated pedestrians with varying degrees of occlusion, challenging model detection performance in complex outdoor backgrounds. The WiderPerson dataset, which includes photos chosen from a variety of scenes rather than just traffic situations, is the benchmark dataset for recognizing pedestrians in outdoor environments. In this paper,

9000 publicly annotated images are proportionally divided into 1800 photos for validation and 7200 images for training. Before conducting experiments, irrelevant data that may interfere with person detection in the original dataset are removed, and all remaining person targets are classified into the same category. A comprehensive benchmark for object detection in aerial photos is the DOTA [39] dataset, containing thousands of images and annotated instances across 15 categories. It is intended to aid in the study and advancement of object detection techniques from an aerial perspective, which are frequently employed in monitoring and remote sensing applications. TinyPerson is a small object detection dataset proposed in the context of maritime rapid rescue. TinyPerson contains 1610 labeled images and 759 unlabeled images (both primarily from the same video set), with a total of 72,651 annotations.

Table 1. Experimental configuration.

Name	Parameter
Operating system	Linux
Programing language	Python3.8
CPU	Intel(R) Xeon(R) Silver 4210 CPU @ 2.20 GHz
GPU	NVIDIA GeForce RTX 2080
Pytorch	1.11.0
CUDA	11.6

Table 2. The training hyperparameters in the experiment.

Hypeparameters	Value
Epoch	300
Batch size	8
Initial learning rate	0.01
Optimizer	SGD

4.2. Evaluation Metrics

The tests assess the models' detection performance using mean mAP, precision, recall, parameters, and GFLOPs.

(1) mAP represents the average precision for each category of detection, calculated by the following formula:

$$mAP = \frac{1}{c}\sum_{i=1}^{c}\int_{0}^{1} P_i(R_i)dR_i \tag{12}$$

In this case, c stands for the total count of detection categories. i is a representation of the number of findings. mAP is the mean of AP across all categories. mAP@0.5 indicates the average precision at an IoU threshold of 0.5, and mAP@0.5:0.95 represents the average precision across IoU thresholds from 0.5 to 0.95 in steps of 0.05.

(2) *Precision* represents the accuracy of the model in detecting targets:

$$P = \frac{TP}{TP + FP} \times 100\% \tag{13}$$

FP denotes the number of false positive samples that were mistakenly projected as negative, and TP is the number of true positive samples that were anticipated as positive.

(3) *Recall* evaluates the model's capacity to accurately recognize genuine positives:

$$R = \frac{TP}{TP + FN} \times 100\% \tag{14}$$

(4) *Params* indicates how many parameters the model has in memory (unit: M).

(5) *GFLOPs* is short for "Giga Floating Point Operations Per Second", signifying the quantity of floating-point operations carried out in a second, typically in the billions. This parameter is generally used to measure the performance of computing systems.

4.3. Experimental Results and Analysis

4.3.1. Ablation Experiment

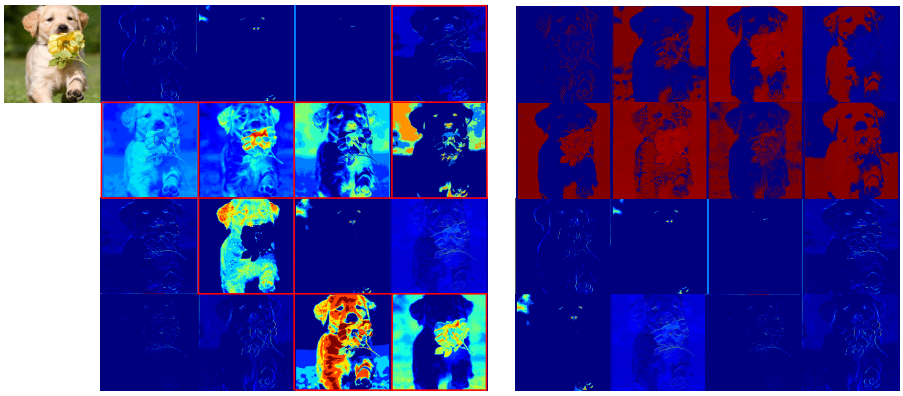
The YOLOv8n algorithm serves as the baseline in this section. Several improvement strategies proposed in this paper are evaluated through ablation experiments, aimed at gaining a deeper understanding of their effects on detection performance. There are six groups of ablation experiments carried out, using *mAP*, *Params*, *Precision*, *GFLOPs*, and *Recall* as metrics for evaluation. In these experiments, ✓ indicates the adoption of the respective method. The results of the ablation experiments performed on the CityPersons dataset are shown in Table 3.

Table 3. Ablation experiment.

Model	SPPFLska	ACConv	LTD_HEAD	Add One Head	mAP@0.5/%	P	R	Params/M
Baseline					62.28	77.34	53.51	3.01
Exp1	✓				63.29	82.13	50.76	3.21
Exp2		✓			62.46	78.49	53.14	2.82
Exp3			✓		63.11	77.86	52.18	1.94
Exp4				✓	64.30	78.72	54.28	2.92
Exp5	✓	✓	✓	✓	66.60	79.5	56.71	2.03

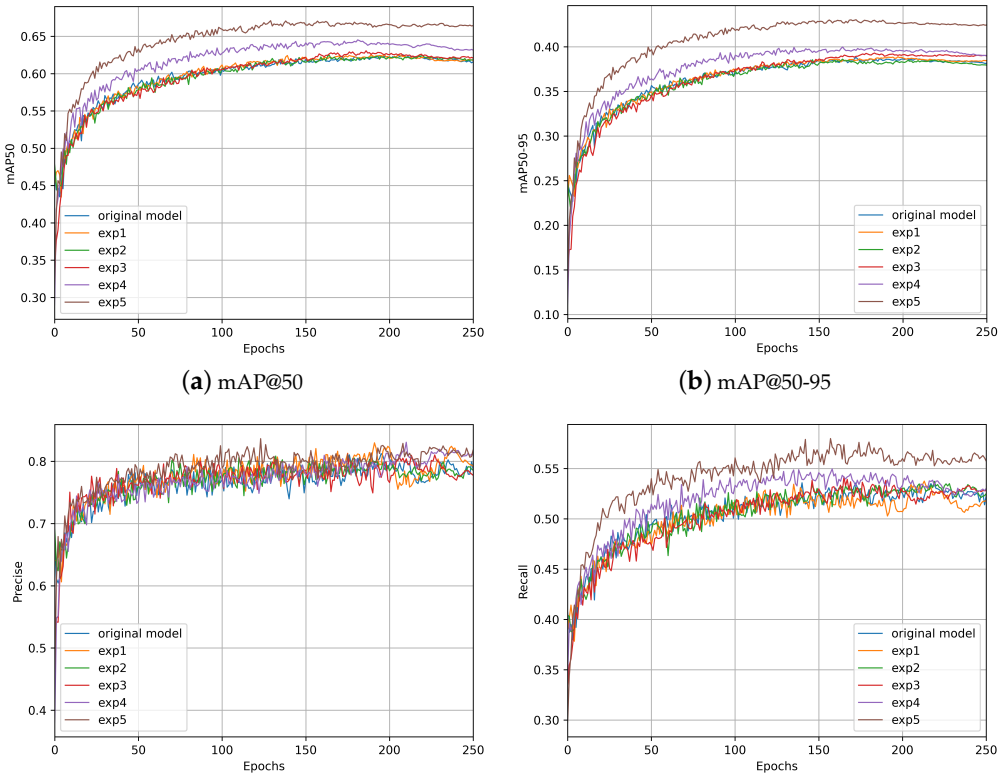
The data in Table 3 indicate that on the CityPersons dataset, Exp1 improves the average precision mean *mAP*@0.5 by 1.01% by incorporating a large separable kernel attention mechanism into the SPPF module. By incorporating large kernels, the SPPFLska module can successfully capture long-range dependencies while maintaining a reduced computational footprint. On the baseline model, Exp2 adds an adaptive channel convolution module, resulting in a notable decline in computational complexity and parameter count, with a 0.18 M parameter reduction in comparison to the baseline model. After computing the features from the input, weights are assigned to each feature, selecting those deemed most significant as representative features for convolutional computation. The remaining features are utilized as supplementary detail features, as illustrated in Figure 6. The *mAP*@0.5 value increases by 0.2%, indicating a slight improvement in detection accuracy while lightening the network. Task alignment detection is a method employed in the domain of recognizing objects to optimize two subtasks: target classification and localization. In one-stage object detection algorithms, heads with two parallel branches are typically used, which could lead to some level of spatial misalignment between the two tasks' results. Exp3 introduces a lightweight task-aligned detection head, leading to a nearly 1% increase in *mAP*@0.5 value, highlighting the importance of alignment between classification and localization tasks in detection tasks. Exp4 incorporates a tiny object detecting layer for detecting small objects, showing significant improvement with a 2.02% increase when comparing in *mAP*@0.5 to the baseline model. Exp5, the DS_YOLO, integrates all the aforementioned improvements, resulting in a gain of 2.16% in precision, 3.2% in recall, and 4.32% in *mAP*@0.5. The above results show that combining the four improvement modules can significantly increase detection precision while lowering the number of parameters and processing complexity, and the combined effect is superior to individual use. Figure 7 illustrates the training process visualization, with training epochs represented by the horizontal axis. The recall curve and *mAP*@0.5 curve of Exp5 are consistently above the other curves, and the parameter count of the network has been significantly reduced to 2.03M, confirming the effectiveness of the DS_YOLO network. Since the dataset does not include annotations for the background and only has annotations for the human category, the FN and TN metrics will be artificially high and low, respectively. Therefore, we mainly

compare the TP and FP metrics. As shown in Figure 8, compared to the baseline model, our model shows an increase in TP by 0.05 and a decrease in FP by 0.05. This indicates an improvement in the model's ability to correctly identify positive samples and a reduction in the rate of incorrectly identifying positive samples.



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

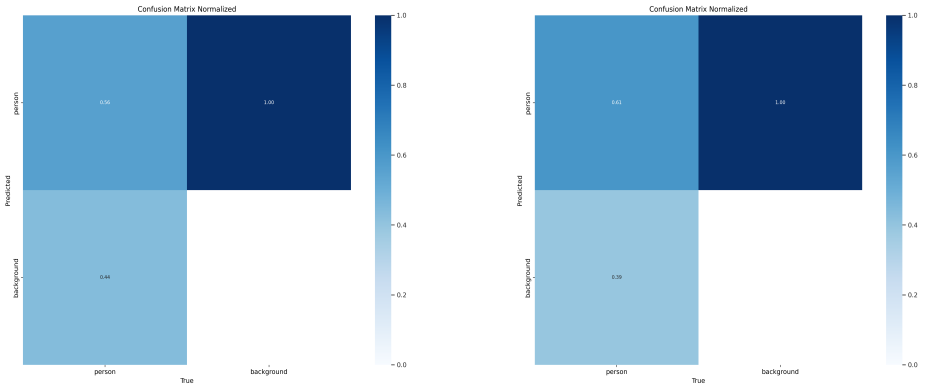
Figure 6. The top-left image represents the original picture. (**a**) The feature maps inputted into ACConv, with the feature maps encircled in red boxes selected as representative features. (**b**) The feature maps outputted by the ACConv module.



(**a**) mAP@50 (**b**) mAP@50-95

(**c**) Precise (**d**) Recall

Figure 7. Parameter curve graph.



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

Figure 8. Panel (**a**) shows the confusion matrix of the baseline model, while panel (**b**) shows the confusion matrix of our model. In the matrices, the top-left corner represents the TP metric, the bottom-left corner represents the FP metric, the top-right corner represents the FN metric, and the bottom-right corner represents the TN metric.

4.3.2. Comparison of Different Detection Algorithms

Comparative studies were performed by comparing the DS_YOLO with other popular YOLO object identification algorithms to more thoroughly assess the performance of the DS_YOLO such as Faster-RCNN [40] and SSD [41], as well as YOLOv3 [21], YOLOv4-Tiny [42], YOLOv5s [23], YOLOv7-Tiny [43], and YOLOv8n. In order to ensure fairness in detection performance, all experiments were conducted with the same number of epochs on the same device. Table 4 displays the outcomes of the comparisons conducted on the CityPersons dataset.

Table 4. Comparative experiments on the CityPersons dataset.

Model	mAP@0.5/%	P/%	R/%	Params	GFLOPs
Faster-RCNN	39.5	69.4	63.6	136.8	370.0
RepLoss	39.1	50.3	46.9	112.2	183
ALFNet	43.1	55.9	48.7	23.5	171
GoogleNet	42.63	63.6	47.3	5	2
SSD	33.3	51.7	56.8	26.3	8.5
RetinaNet	41.3	52.1	47.2	34.3	37.4
DETR	43.4	56.0	48.2	41	86
YOLOv3	40.6	49.7	50.6	61.9	66.3
YOLOv4-tiny	30.2	65.4	51.3	6.4	21.8
YOLOv5s	53.4	67.7	50.3	7.2	16.6
YOLOv7-tiny	54.6	71.2	48.9	6.0	13.2
YOLOv8n	62.3	77.3	53.5	3.0	8.9
DS_YOLO(ours)	66.6	79.5	56.7	2.0	6.4

From Table 4, it is evident that Faster-RCNN, SSD, YOLOv3, and YOLOv4-Tiny have lower detection accuracy compared to other networks, failing to meet the accuracy requirements for detection. Additionally, these models have higher computational complexity and longer inference times, making them unsuitable for rapid detection tasks. In contrast, the improved DS_YOLO algorithm strikes a more suitable equilibrium between detection speed, parameter count, and model accurateness when compared to YOLOv5s, YOLOv7-tiny, and YOLOv8n. This demonstrates its advantage in detecting dim and small objects, as targeted in this study. Compared to the baseline models, the DS_YOLO achieves a 4.32% increase in accuracy on the CityPersons dataset, while reducing parameter count by 0.97 M. The GFLOPs are also reduced to 6.4, with a marginal improvement in accuracy and speed

of recognition. This indicates the algorithm's strong feature perception and extraction capabilities for dim and small targets.

4.3.3. Generalization Experiment

This paper also conducts comparative experiments on the WiderPerson, DOTA, and TinyPerson dataset to verify the generalization of the model in detecting dim and small targets in complex environments. As Table 5 shows, the DS_YOLO continues to demonstrate excellent performance on the WiderPerson and DOTA datasets, achieving a 1.01% improvement over the baseline model on WiderPerson and a 1.69% improvement on the DOTA dataset. It outperforms other mainstream detection algorithms and exhibits strong generalization capabilities. In summary, the results analysis indicates that our model, DS_YOLO, achieves a balance between detection performance and model computational complexity in dim and small object detection tasks, demonstrating overall excellent performance.

Table 5. Comparative experiments on the WiderPerson, DOTA, and TinyPerson datasets.

Model	WiderPerson Dataset			DOTA Dataset			TinyPerson Dataset		
	mAP@0.5/%	**Params**	**GFLOPs**	**mAP@0.5/%**	**Params**	**GFLOPs**	**mAP@0.5/%**	**Params**	**GFLOPs**
Faster-RCNN	61.5	136.8	370.0	54.1	136.8	370.0	5.1	136.8	370.0
RepLoss	61.2	112.2	183	65.2	112.2	183	3.6	112.2	183
ALFNet	60.1	23.5	171	72.1	23.5	171	44	23.5	171
GoogleNet	59	5	2	49.5	5	2	6.2	5	2
SSD	57.4	26.3	8.5	50.1	26.3	8.5	3.7	26.3	8.5
RetinaNet	52.0	34.3	37.4	29.9	34.3	37.4	3.9	34.3	37.4
DETR	49.8	41	86	53.5	41	86	6.3	41	86
YOLOv3	60.3	61.9	66.3	60.0	61.9	66.3	16.3	61.9	66.3
YOLOv4-tiny	64.1	6.4	21.8	64.4	6.4	21.8	10.9	6.4	21.8
YOLOv5s	68.2	7.2	16.6	67.9	7.2	16.6	11.3	7.2	16.6
YOLOv7-tiny	70.6	6.0	13.2	70.1	6.0	13.2	13.2	6.0	13.2
YOLOv8n	74.6	3.0	8.9	81.2	3.0	8.9	18.1	3.0	8.9
DS_YOLO	**75.57**	**2.0**	**6.4**	**82.8**	**2.0**	**6.4**	**18.9**	**2.0**	**6.4**

4.4. Detection Result

4.4.1. Detection Result Comparison on Datasets

A contrast of the detection performance of the initial model and the model suggested in this work is shown in Figure 9. The input images for detecting are displayed in the first row, the original model's detection results are shown in the second row, and the suggested model's recognition results are shown in the third row. It is evident from the detection images that the first baseline network model has low confidence ratings and missed detections. In contrast, our proposed DS_YOLO model can accurately identify and predict bounding boxes for weak pedestrians missed by the baseline network, resulting in relatively more accurate detections and improved confidence scores. This visualization demonstrates that the improved network enhances the detection performance of weak pedestrian targets on complex road surfaces.

4.4.2. Detection Result on Micro-Light Sight

The micro-light data are collected by a micro-light detector in the smart micro-light sight. The response wavelength range of this detector is 380–940 nm, encompassing a broad spectrum from visible to near-infrared light. This allows the images to retain rich detail and contrast even under micro-light conditions, although the images may still appear somewhat dim. Additionally, due to the long focal length and narrow field of view of the smart micro-light sight, the targets in the collected images tend to be small and dim. In this section, we used our models trained on the previously mentioned public datasets to detect images collected by the smart micro-light sight. Figure 10 shows the detection results of DS_YOLO on images captured by the smart micro-light sight. It is evident that

the ability of detection for small and dim targets against complex backgrounds is excellent, with virtually no missed detections, even in densely populated scenes.

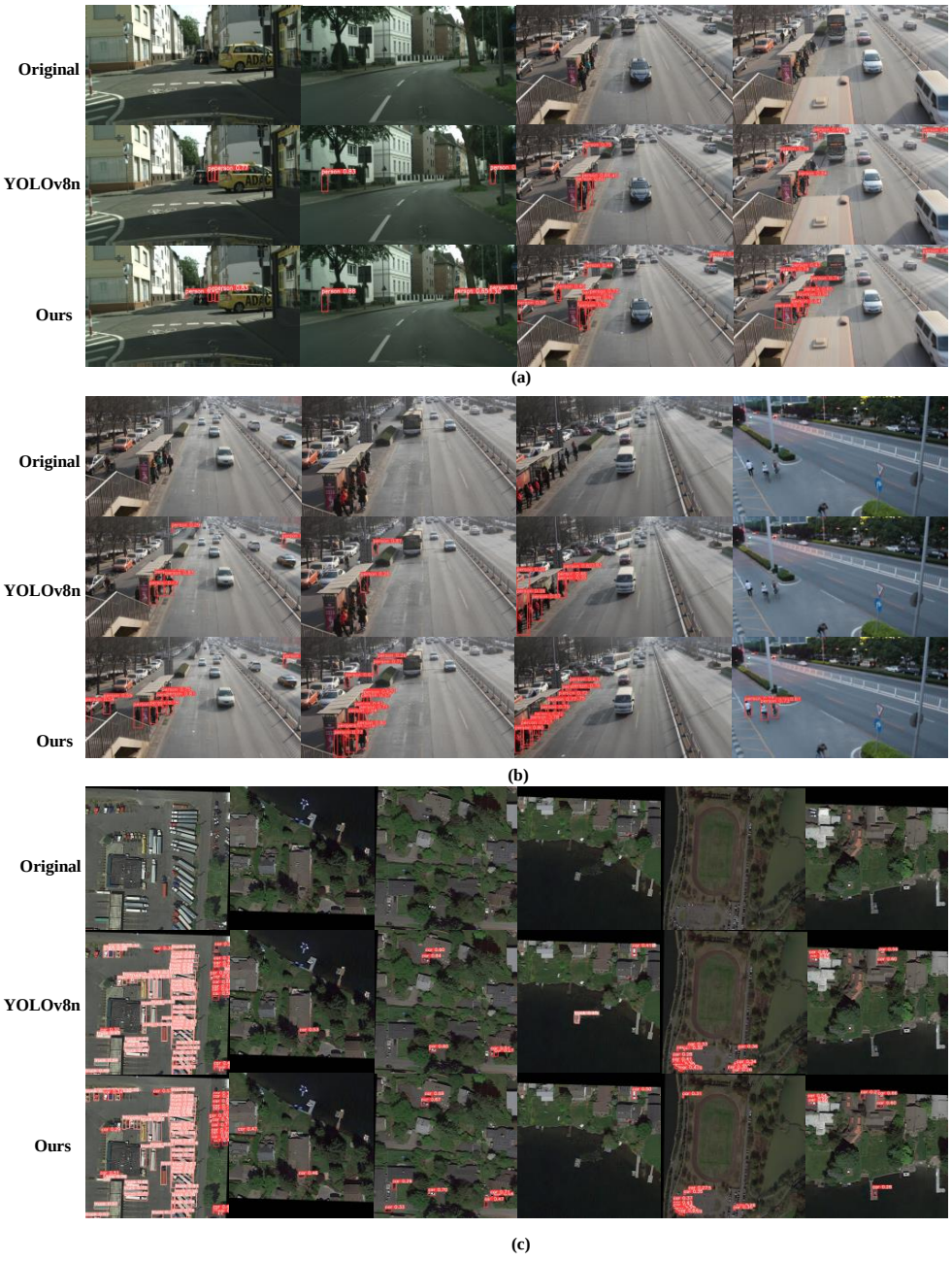


Figure 9. (**a**,**b**) Comparison of dim and small pedestrian detection results. (**c**) Comparison of remote sensing small object detection results.

The base model was successfully deployed on the RK3588 chip, with an inference time of approximately 40 ms. Our subsequent task is to optimize the model deployment to achieve real-time performance, which appears to be highly feasible. When deploying a trained model onto hardware, the supported operators of the hardware significantly impact the deployment workload and time. Some operators in this model are not supported by Rockchip's chips, necessitating the creation or optimization of custom operators. This situation is similar for other chips, where the deployment workload and final model performance are significantly impacted by the operators supported by the specific hardware.

Figure 10. Detection result for micro-light sight every 50 frames.

5. Conclusions

In this work, we presented a detection method that is lightweight and designed to identify small and dim objects with complicated backgrounds, named DS_YOLO. We innovatively improved the network architecture to address small-scale targets, and introduced large-kernel convolution attention into the pyramid pooling module, to avoid the information about small things from gradually disappearing during computation, thereby better capturing vital information about small-scale targets in images. Additionally, the proposed lightweight task alignment detection head (LTD_Head) and adaptive channel convolution (ACConv) increased task identification accuracy while drastically lowering the model's computational overhead and parameter count. Our methodology beats previous state-of-the-art methods in terms of both quantity and quality, as demonstrated by experimental data, while maintaining comparable parameters, thus alleviating the trade-off between performance improvement and parameter reduction. Our ultimate goal is to deploy the model on hardware for practical use, so the core of our future work will focus on addressing issues related to the deployment process. Given that the current base model deployment is successful but slightly falls short of achieving real-time performance, it is highly likely that our model can achieve real-time capabilities. When deploying the trained model onto hardware, the operators supported by the hardware will significantly impact the deployment workload and time. Some operators in this model are not supported by the Rockchip chip, necessitating the creation or optimization of custom operators. This situation is similar with other chips, where the deployment workload and final model performance are significantly affected by the operators supported by the specific hardware. Therefore, in our future work, we will convert the trained model into a format that is readable and executable by the hardware, followed by continuous optimization based on practical considerations.

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

Funding: This research was funded by Natural Science Foundation of Hubei Province of China (Joint Fund for Innovation and Development of Shiyan), grant number 2024AFD116 and Key Project of Science and Technology Research Plan of Hubei Provincial Department of Education, Grant NO. D20231805.

Data Availability Statement: Data are contained within the article. One can send an email to the first author and corresponding author to request the data and code.

Conflicts of Interest: Author Yuanbin Len was employed by the company Chengdu Dingyi Information Technology Co. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Xiao, Y.; Zhou, K.; Cui, G.; Jia, L.; Fang, Z.; Yang, X.; Xia, Q. Deep learning for occluded and multi-scale pedestrian detection: A review. *IET Image Process.* **2021**, *15*, 286–301. [CrossRef]
2. Sun, J.; Wang, Z. Vehicle And Pedestrian Detection Algorithm Based on Improved YOLOv5. *IAENG Int. J. Comput. Sci.* **2023**, *50*, 1401–1409.
3. Liu, L.; Lu, S.; Zhong, R.; Wu, B.; Yao, Y.; Zhang, Q.; Shi, W. Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet Things J.* **2020**, *8*, 6469–6486. [CrossRef]
4. Levinson, J.; Askeland, J.; Becker, J.; Dolson, J.; Held, D.; Kammel, S.; Kolter, J.Z.; Langer, D.; Pink, O.; Pratt, V.; et al. Towards fully autonomous driving: Systems and algorithms. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden, Germany, 5–9 June 2011; pp. 163–168.
5. Gawande, U.; Hajari, K.; Golhar, Y. Pedestrian detection and tracking in video surveillance system: Issues, comprehensive review, and challenges. In *Recent Trends in Computational Intelligence*; Intech Open Publ.: London, UK, 2020; pp. 1–24. [CrossRef]
6. Khan, M.A.; Javed, K.; Khan, S.A.; Saba, T.; Habib, U.; Khan, J.A.; Abbasi, A.A. Human action recognition using fusion of multiview and deep features: An application to video surveillance. *Multimed. Tools Appl.* **2024**, *83*, 14885–14911. [CrossRef]
7. Alfred Daniel, J.; Chandru Vignesh, C.; Muthu, B.A.; Senthil Kumar, R.; Sivaparthipan, C.; Marin, C.E.M. Fully convolutional neural networks for LIDAR–camera fusion for pedestrian detection in autonomous vehicle. *Multimed. Tools Appl.* **2023**, *82*, 25107–25130. [CrossRef]
8. Ahmed, S.; Kallu, K.D.; Ahmed, S.; Cho, S.H. Hand gestures recognition using radar sensors for human-computer-interaction: A review. *Remote Sens.* **2021**, *13*, 527. [CrossRef]
9. Schmid, C.; Soatto, S.; Tomasi, C. *Conference on Computer Vision and Pattern Recognition*; IEEE Computer Society: Washington, DC, USA, 2005.
10. Huang, L.; Chen, C.; Yun, J.; Sun, Y.; Tian, J.; Hao, Z.; Yu, H.; Ma, H. Multi-scale feature fusion convolutional neural network for indoor small target detection. *Front. Neurorobotics* **2022**, *16*, 881021. [CrossRef] [PubMed]
11. Mordan, T.; Thome, N.; Henaff, G.; Cord, M. End-to-end learning of latent deformable part-based representations for object detection. *Int. J. Comput. Vis.* **2019**, *127*, 1659–1679. [CrossRef]
12. Wang, X.; Han, T.X.; Yan, S. An HOG-LBP human detector with partial occlusion handling. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 32–39.
13. Chen, P.H.; Lin, C.J.; Schölkopf, B. A tutorial on ν-support vector machines. *Appl. Stoch. Models Bus. Ind.* **2005**, *21*, 111–136. [CrossRef]
14. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
15. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
16. Vedaldi, A.; Gulshan, V.; Varma, M.; Zisserman, A. Multiple kernels for object detection. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 606–613.
17. Yu, Y.; Zhang, J.; Huang, Y.; Zheng, S.; Ren, W.; Wang, C.; Huang, K.; Tan, T. Object detection by context and boosted HOG-LBP. In Proceedings of the ECCV Workshop on PASCAL VOC, Crete, Greece, 5–11 September 2010.
18. Liu, T.; Cheng, J.; Yang, M.; Du, X.; Luo, X.; Zhang, L. Pedestrian detection method based on self-learning. In Proceedings of the 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 20–22 December 2019; Volume 1, pp. 2161–2165.
19. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

20. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
21. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
22. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
23. Wu, W.; Liu, H.; Li, L.; Long, Y. Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image. *PLoS ONE* **2021**, *16*, e0259283.
24. 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. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
25. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Piscataway, NJ, USA, 11–18 December 2015; Volume 2, pp. 1440–1448.
26. Gong, H.; Li, H.; Xu, K.; Zhang, Y. Object detection based on improved YOLOv3-tiny. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; pp. 3240–3245.
27. Woo, S.; Park, J.; Lee, J.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
28. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
29. Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y.; Ning, J.; Cao, Y.; Zhang, Z.; Dong, L.; et al. Swin transformer v2: Scaling up capacity and resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12009–12019.
30. Yin, Y.; Zhang, Z.; Wei, L.; Geng, C.; Ran, H.; Zhu, H. Pedestrian detection algorithm integrating large kernel attention and YOLOV5 lightweight model. *PLoS ONE* **2023**, *18*, e0294865. [CrossRef] [PubMed]
31. Xu, Z.; Pan, S.; Ma, X. A Pedestrian Detection Method Based on Small Sample Data Set. In Proceedings of the 2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA), Changchun, China, 11–13 August 2023; pp. 669–674.
32. Chen, H.; Guo, X. Multi-scale feature fusion pedestrian detection algorithm based on Transformer. In Proceedings of the 2023 4th International Conference on Computer Vision, Image and Deep Learning (CVIDL), Zhuhai, China, 12–14 May 2023; pp. 536–540.
33. Bai, Y.; Zhang, Y.; Ding, M.; Ghanem, B. Finding tiny faces in the wild with generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 21–30.
34. Bai, Y.; Zhang, Y.; Ding, M.; Ghanem, B. Sod-mtgan: Small object detection via multi-task generative adversarial network. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 206–221.
35. Feng, C.; Zhong, Y.; Gao, Y.; Scott, M.R.; Huang, W. Tood: Task-aligned one-stage object detection. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, Montreal, BC, Canada, 11–17 October 2021; pp. 3490–3499.
36. Lau, K.W.; Po, L.M.; Rehman, Y.A.U. Large separable kernel attention: Rethinking the large kernel attention design in cnn. *Expert Syst. Appl.* **2024**, *236*, 121352. [CrossRef]
37. Zhang, S.; Benenson, R.; Schiele, B. Citypersons: A diverse dataset for pedestrian detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3213–3221.
38. Zhang, S.; Xie, Y.; Wan, J.; Xia, H.; Li, S.Z.; Guo, G. Widerperson: A diverse dataset for dense pedestrian detection in the wild. *IEEE Trans. Multimed.* **2019**, *22*, 380–393. [CrossRef]
39. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A large-scale dataset for object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3974–3983.
40. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef] [PubMed]
41. 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, Part I 14, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
42. Jiang, Z.; Zhao, L.; Li, S.; Jia, Y. Real-time object detection method based on improved YOLOv4-tiny. *arXiv* **2020**, arXiv:2011.04244.
43. Ma, L.; Zhao, L.; Wang, Z.; Zhang, J.; Chen, G. Detection and counting of small target apples under complicated environments by using improved YOLOv7-tiny. *Agronomy* **2023**, *13*, 1419. [CrossRef]