

Article

A Novel Lightweight Object Detection Network with Attention Modules and Hierarchical Feature Pyramid

Shengying Yang ^{1,2,*} , Linfeng Chen ², Junxia Wang ², Wuyin Jin ¹ and Yunxiang Yu ³¹ School of Mechanical and Electrical Engineering, Lanzhou University of Technology, Lanzhou 730050, China² School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China³ Zhejiang Dingli Industry Co., Ltd., Lishui 321400, China

* Correspondence: syyang@zust.edu.cn

Abstract: Object detection methods based on deep learning typically require devices with ample computing capabilities, which limits their deployment in restricted environments such as those with embedded devices. To address this challenge, we propose Mini-YOLOv4, a lightweight real-time object detection network that achieves an excellent trade-off between speed and accuracy. Based on CSPDarknet-Tiny as the backbone network, we enhance the detection performance of the network in three ways. We use a multibranch structure embedded in an attention module for simultaneous spatial and channel attention calibration. We design a group self-attention block with a symmetric structure consisting of a pair of complementary self-attention modules to mine contextual information, thereby ensuring that the detection accuracy is improved without increasing the computational cost. Finally, we introduce a hierarchical feature pyramid network to fully exploit multiscale feature maps and promote the extraction of fine-grained features. The experimental results demonstrate that Mini-YOLOv4 requires only 4.7 M parameters and has a billion floating point operations (BFLOPs) value of 3.1. Compared with YOLOv4-Tiny, our approach achieves a 3.2% improvement in mean accuracy precision (mAP) for the PASCAL VOC dataset and obtains a significant improvement of 3.5% in overall detection accuracy for the MS COCO dataset. In testing with an embedded platform, Mini-YOLOv4 achieves a real-time detection speed of 25.6 FPS on the NVIDIA Jetson Nano, thus meeting the demand for real-time detection in computationally limited devices.

Keywords: object detection; embedded platform; attention model; feature pyramid; real-time performance



Citation: Yang, S.; Chen, L.; Wang, J.; Jin, W.; Yu, Y. A Novel Lightweight Object Detection Network with Attention Modules and Hierarchical Feature Pyramid. *Symmetry* **2023**, *15*, 2080. <https://doi.org/10.3390/sym15112080>

Academic Editors: João Ruivo Paulo, Cristina P. Santos and Gabriel Pires

Received: 10 October 2023

Revised: 12 November 2023

Accepted: 14 November 2023

Published: 17 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

AS one of the fundamental tasks in computer vision, object detection is widely used in face detection, object tracking, image segmentation, and autonomous driving [1]. The objective is to localize and classify specific objects in an image, accurately find all the objects of interest, and locate the position with a rectangular bounding box [2,3]. In recent years, in the field of computer vision, there has been a growing focus on designing deeper networks to extract valuable feature information, resulting in improved performance [4–8]. However, due to the vast number of parameters in these models, they often consume a significant amount of computational resources. For example, Khan [9] proposed an end-to-end scale-invariant head detection framework by modeling a set of specialized scale-specific convolutional neural networks with different receptive fields to handle scale variations. Wang [10] introduced a pyramid structure into the transformer framework, using a progressive shrinking strategy to control the scale of feature maps. While these models demonstrate outstanding detection accuracy, they heavily rely on powerful GPUs to achieve rapid detection speed [11]. This poses a significant challenge in achieving a balance between accuracy and inference speed on mobile devices with limited computational resources [12–14]. Currently, detection models based on deep learning often use complex

network architectures to extract valuable feature information. Although such models have a high detection accuracy, they usually rely on powerful graphics processing units (GPUs) to achieve a fast detection speed [15]. With the rapid development of technologies such as smartphones, drones, and unmanned vehicles, implementing neural networks in parallel on devices with limited storage and computing power is becoming an urgent need. Under computing power and storage space constraints, lightweight real-time networks have become popular research topics related to the application of deep learning in embedded applications [16].

Recently, some researchers have reduced the number of parameters and model size of the network by optimizing the network structure, such as SqueezeNet, MobileNetv1-v3 [17–19], ShuffleNetv1-v2 [20,21], Xception [22], MixNet [23], EfficientNet [24], etc. The MobileNet series methods replace the traditional convolution by using depth-wise separable convolutions, thus achieving a result similar to that of standard convolution but greatly reducing the number of model calculations and parameters. The ShuffleNet series networks use group convolution to reduce the number of model parameters and apply channel shuffling to reorganize the feature maps generated by group convolution. Other researchers have proposed regression-based one-stage object detectors, such as SSD [25], YOLOv1-v4 [26–29], RetinaNet [30], MimicDet [31], etc. Instead of taking two shots, as in the RCNN series, one-stage detectors predict the target location and category information directly from a network without region propositions. Based on the regression concept of YOLOv1, SSD uses predefined boxes of different scales and aspect ratios for prediction and extracts feature maps of different scales for detection. Although the SSD accuracy is better than that of YOLOv1, SSD does not perform well in small object detection. YOLOv3 uses the Darknet backbone network to mine high-level semantic information, which greatly improves the classification performance. A similar feature pyramid network is used for feature fusion to enhance the accuracy of small target detection. Since a large number of easily classified negative samples in the training phase can lead to model degradation, RetinaNet proposes focal loss based on standard cross-entropy loss to eliminate category imbalance effectively, similar to a kind of hard sample mining. To improve the accuracy of the one-stage detector, MimicDet uses the features generated by a two-stage detector to train the one-stage detector in the training phase. However, in the inference phase.

MimicDet uses a one-stage method directly for prediction to ensure that the detection speed is relatively fast. The YOLO series methods achieve an excellent balance between accuracy and speed and have become widely used for target detection in actual scenarios. Nevertheless, YOLO models have a complex network structure and a large number of network parameters, so they require vast computing resources and considerable storage space when used in embedded devices. However, the high computational cost limits the ability of YOLO models to perform multiple tasks that require real-time performance on computationally limited platforms [32]. To reduce the occupation of computing resources, lightweight YOLO methods require fewer parameters and improve the detection speed by applying a smaller feature extraction network, such as the latest YOLOv4-Tiny [33]. Therefore, when performing object detection on embedded devices, improving the detection accuracy while achieving real-time performance is a significant problem to be solved.

This is because YOLO has a faster processing speed compared to Xception, making it more suitable for real-time applications, and the widespread use of Yolov4 and its excellent performance in many object detection-related tasks. After a comprehensive literature review to identify popular methods and the existing limitations in the field of object detection, we decided to exploit multiscale feature maps to promote the extraction of fine-grained features because they can improve the detection of small and medium-sized objects. In addition, we also considered how to achieve spatial and channel attention calibration through structural optimization and applied other strategies to improve detection accuracy without increasing computational costs. To obtain an efficient object detection model that can be applied in constrained environments originating from YOLOv4-Tiny, Mini-YOLOv4 is proposed in this paper to achieve an excellent trade-off between speed and accuracy. Compared with

YOLOv4-Tiny, Mini-YOLOv4 not only improves the detection accuracy but also effectively reduces the number of model calculations and number of parameters from 5.9 M to 4.7 M, which means it can achieve efficient object detection in embedded devices. And compared to YOLOv3, YOLOv3-Tiny, YOLOv4-CSP, YOLOv4-Tiny, and YOLOv5s, Mini-YOLOv4 achieves fewer parameter sizes. To evaluate the effectiveness of our method for lightweight object detection in embedded systems, we conducted experiments on benchmark datasets, such as PASCAL VOC and MS COCO, and compared our method with other models. We also considered various metrics (e.g., parameters, BFLOPs, FPS, mAP, AP50, AP75, etc.) to measure the computational cost and detection performance of our method. We chose the NVIDIA Jetson Nano, a widely used low-cost deep learning platform, as the experimental test environment.

The main contributions of this paper are as follows:

(1) To reduce the number of parameters while achieving improved detection accuracy, we built a multibranch feature aggregation module (MFBlock) to replace the last 3×3 convolutional block in the backbone network. In MFBlock, we embed a new attention mechanism called the complete attention module (CAM) that directly explores spatial and channel clues. CAM uses the spatial structure information ignored by SENet and provides a significant increase in accuracy at a low computational cost.

(2) To exploit long-range dependencies, we design a group self-attention block (GS-Block) to replace the 3×3 convolutional block in the prediction head, consisting of a spatial group attention module (SGAM) and channel group attention module (CGAM). SGAM focuses on capturing the spatial association among feature maps, and CGAM aims to aggregate channel-wise feature information. We use SGAM and CGAM jointly to obtain comprehensive feature representations.

(3) To improve the regression accuracy for small and medium targets, we introduce a hierarchical feature pyramid network (H-FPN). In H-FPN, we use upsampling and downsampling to resize the feature maps of different stages in the network. Then, we fuse the high-level semantic features with the low-level feature representations in a hierarchical manner to obtain fine-grained features.

(4) Extensive experiments on PASCAL VOC and MS COCO datasets verify the effectiveness of each component. Moreover, we compare Mini-YOLOv4 with state-of-the-art object detection algorithms and other lightweight models. Mini-YOLOv4 achieves comparable results for mAP, lower BFLOPs value, and a real-time detection speed on an embedded platform NVIDIA Jetson Nano.

2. Related Work

2.1. Attention Mechanism

In recent years, attention mechanisms have been widely used in various fields of computer vision to enhance important features and suppress irrelevant noise. These mechanisms provide excellent performance in improving model accuracy, such as SENet [34], CBAM [35], No-local [36], SKNet [37], GCNet [38], NAM [39], ECANet [40], SA-Net [41], SimAM [42], GAM [43], etc. SENet explicitly models the correlation between feature channels and automatically learns the channel-wise weights for feature selection. CBAM focuses on spatial and channel attention information and concatenates the feature maps after average and maximum pooling operations to reduce feature loss, thus making the model focus on the target itself rather than the background. SKNet uses convolution kernels of different sizes to extract semantic features and dynamically adjusts the receptive field by aggregating feature information from multiple branches. Based on SENet and No-local, GCNet proposes a simple global context modeling framework to mine long-distance dependencies and reduce computational pressure. NAM applies a weight sparsity penalty to the attention module, thereby improving computational efficiency while maintaining similar performance. ECANet has mainly made some improvements to the SENet module, proposing a non-dimensional reduction local cross-channel interaction strategy and an adaptive method for selecting the size of one-dimensional convolutional kernels, thereby

achieving performance improvement. Although CBAM brings performance improvements, it increases the computational complexity to a certain extent. SA-Net introduces the channel shuffle method, which parallelizes the use of spatial and channel attention mechanisms in blocks, enabling efficient integration of the two types of attention mechanisms. Different from the common channel and spatial attention modules, SimAM introduces an attention mechanism without any trainable parameters, proposed based on neuroscience theory and the linear separability principle. GAM proposes a global attention mechanism that introduces channel attention and multi-layer perceptrons to reduce information diffusion and amplify global interactive representations, thereby improving the performance of deep neural networks. The attention mechanism we propose is different from those in previous methods in some aspects. First, Mini-YOLOv4 introduces the CAM to combine spatial and channel attention calibration, which overcomes the limitations of performing spatial or channel attention alone with a low computational cost. Second, our two self-attention modules make full use of group convolutions, thereby creating a lightweight model.

2.2. YOLOv4, YOLOv4-CSP and YOLOv4-Tiny Networks

YOLOv4 is an evolution from YOLOv3, and the purpose is to design a real-time object detection network that can be applied in the actual working environment. YOLOv4 proposes a CSPDarknet53 backbone network to reduce repeated gradient learning effectively and improve the learning ability of the network. In terms of data augmentation, YOLOv4 uses a mosaic to combine four images into one, which is equivalent to increasing the minibatch size and adds self-adversarial training (SAT), which allows the neural network to update images in the reverse direction before normal training. In addition, YOLOv4 uses modules such as ASFF [44], ASPP [45], and RFB [46] to expand the receptive field and introduce attention mechanisms to emphasize important features. Based on YOLOv4, YOLOv4-CSP is compressed in terms of network width, network depth, and image resolution to achieve the optimal trade-off between speed and accuracy. Compared with YOLOv4, YOLOv4-CSP converts the first CSP stage into the original Darknet residual layer and modifies the PAN architecture in YOLOv4 according to the CSP approach. Moreover, YOLOv4-CSP inserts an SPP module in the middle position of the modified PAN structure. To reduce the computational complexity for embedded devices, YOLOv4-Tiny is a simplified structure of YOLOv4 and YOLOv4-CSP. YOLOv4-Tiny uses a lightweight backbone network called CSPDarknet-Tiny while directly applying a feature pyramid network (FPN) [47] instead of a path aggregation network (PANet) [48] to reduce computational complexity. In the inference stage, multiscale feature maps are first fused via the FPN. Then, the category scores and offsets of each predefined anchor are predicted by a 1×1 convolution kernel, and the predicted bounding boxes are postprocessed using non-maximal suppression (NMS) [49] to obtain the final detection results. Although YOLOv4-Tiny provides a certain accuracy rate and fast detection speed, the regression accuracy for small and medium targets is relatively low, which will be improved in the proposed Mini-YOLOv4 network.

3. Method

Figure 1 illustrates our proposed Mini-YOLOv4 framework, which mainly includes three proposed modules: a multibranch feature aggregation block (MFBlock), a group self-attention block (GSBlock), and a hierarchical feature pyramid network (H-FPN). MFBlock introduces an attention module to capture spatial and channel clues directly and fuse feature information from multiple branches to expand the receptive field. Next, GSBlock explicitly models the point-to-point correlations among feature maps to mine long-range dependencies and obtain rich global information, thereby ensuring that the detection accuracy is improved without increasing the computational and parametric volume of the model. Finally, to improve the detection accuracy for small and medium-sized targets, we optimize the multiscale prediction process in YOLOv4-Tiny by fusing feature maps in different layers of the network in a hierarchical manner to obtain fine-grained feature representations.

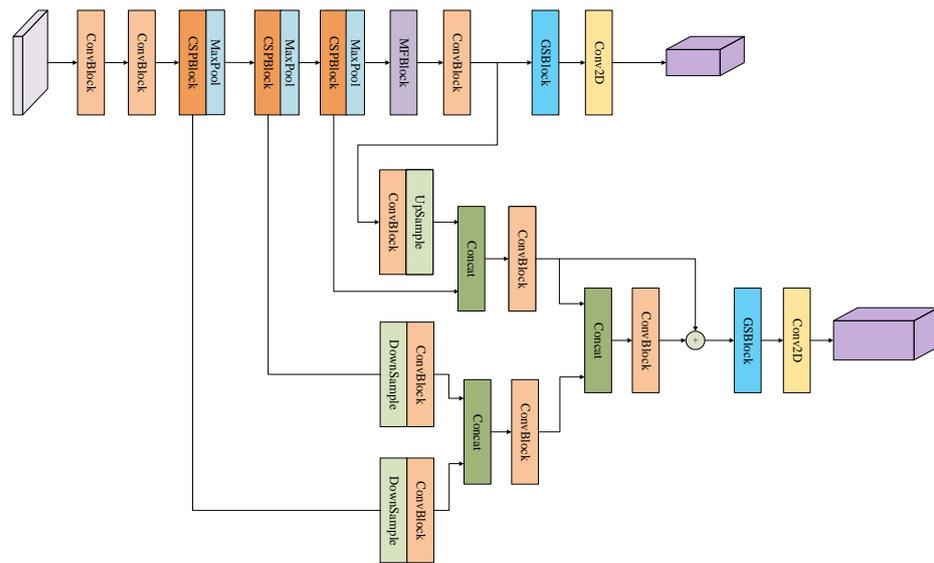


Figure 1. Architecture of Mini-YOLOv4.

3.1. Multibranch Feature Aggregation Block

To improve the detection speed, the network structure of YOLOv4-Tiny is relatively simple, so it is impossible to obtain fine-grained features. A multibranch feature aggregation block called MFBBlock is designed to replace the last 3×3 convolution operation in the backbone network of the original YOLOv4-Tiny. The objective is to reduce the computational complexity and parameter size of the network while improving the ability of network feature extraction.

Recent studies have shown that channel attention (e.g., SENet) contributes to model performance, but it ignores extremely important spatial structure information. In order to generate fine-grained features with semantic information, we directly obtain channel and spatial responses through 1×1 convolution and enlarge the receptive field by fusing multiscale features. As shown in Figure 2, the multibranch feature aggregation block consists of three branches. The first and second branches are used to extract the feature maps of different receptive fields, and the last branch is used for shortcut connections. Specifically, the first branch performs channel reduction through 1×1 convolution and then extracts semantic features through 3×3 convolution. After channel reduction, the second branch learns the weights of each channel and spatial location for feature selection through the CAM. As shown in Figure 3, in addition to calibrating the channel response of the feature map, CAM considers the spatial location information and uses two 1×1 convolutions and a Leaky Relu activation function instead of the fully connected layer used by SENet to obtain an attention mask while performing identity connection to effectively uses previous feature information. Therefore, we can obtain an attention mask the same size as the input feature map.

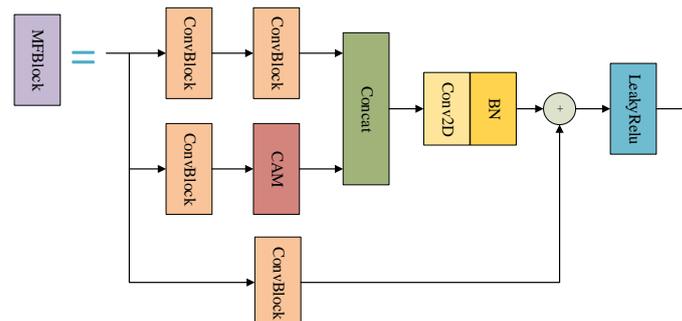


Figure 2. Multibranch feature fusion block (MFBBlock).

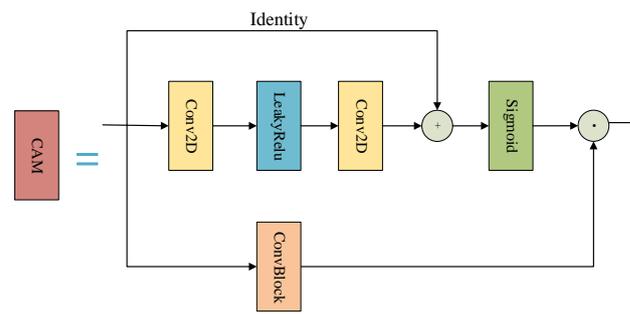


Figure 3. Complete attention module (CAM).

Given an input feature F_m , the attention map is calculated as:

$$A_m = \text{Sigmoid}(F_m + W_1(\text{LeakyReLU}(W_2(F_m)))) \quad (1)$$

where W_1 and W_2 are the parameter matrices of 1×1 convolution operation. The former is used for excitation, and the latter is used for squeezing. After obtaining the attention map A_m , the output feature map of \tilde{F}_m is calculated as:

$$\tilde{F}_m = \text{ConvBlock}(F_m) \odot A_m \quad (2)$$

where *ConvBlock* contains a 1×1 convolution, batch normalization, and Leaky ReLU activation function. The operator \odot is implemented in an elementwise manner. Notably, since CAM does not change the input feature-map size, CAM can be applied to any existing CNN network to emphasize discriminative features.

3.2. Group Self-Attention Block with Symmetric Structure

Although MFBlock emphasizes important features and mitigates the interference associated with redundant information, it is limited by the local receptive field and cannot obtain rich global information. Specifically, the global information mentioned here refers to obtaining the attention coefficient matrix through the attention mechanism, selecting the original input features based on the attention coefficient matrix, and choosing important features from the rich semantic information. Therefore, to efficiently capture long-range dependencies, we propose a GSBlock to replace the 3×3 convolutional block in the prediction head. GSBlock consists of two parts: (1) a spatial group self-attention module (SGAM) and (2) a channel group self-attention module (CGAM). Specifically, SGAM focuses on capturing the spatial association among feature maps. Since high-level channels tend to be strongly correlated, CGAM aims to aggregate channel-wise feature information. We jointly use SGAM and CGAM to mine contextual information and obtain a comprehensive feature representation.

Spatial group self-attention module: To capture the semantic dependencies among pixels in the spatial domain, we introduce a spatial attention module based on a self-attention mechanism. The no-local approach first performs linear mapping of input features to obtain query and key feature maps for calculating the similarity weights. Then, the attention information of the original features is corrected based on these weights. Finally, the feature channel for residual operation is expanded, but this whole process is relatively inefficient. Unlike the no-local method, our approach considers contextual information by partitioning a feature map into two symmetric groups through group convolution and computes the pairwise relations to form an affinity matrix. Then, the feature map groups are aggregated through weighted summation with the affinity matrix. The purpose of this design is to promote cross-group interactions to gain rich global information and efficiently reduce the number of self-attention calculations to achieve a lightweight network structure.

As shown in Figure 4, given the feature map $F_s \in \mathbb{R}^{C \times H \times W}$, the pairwise feature maps F_{s_1} and F_{s_2} are computed as:

$$[F_{s_1}, F_{s_2}] = W_3((W_4(F_s))) \quad (3)$$

where $F_{s_j} \in \mathbb{R}^{C' \times H \times W}$ and $C' = C/2$. W_3 and W_4 are the parameter matrices of 1×1 convolution and 3×3 group convolution, respectively. Then, we resize F_{s_1} and F_{s_2} to $\mathbb{R}^{C' \times N}$, where $N = H \times W$. The spatial affinity matrix $A_s \in \mathbb{R}^{1 \times H \times W}$, as shown below:

$$A_s = \text{Softmax}(F_{s_1}^T F_{s_2}) \quad (4)$$

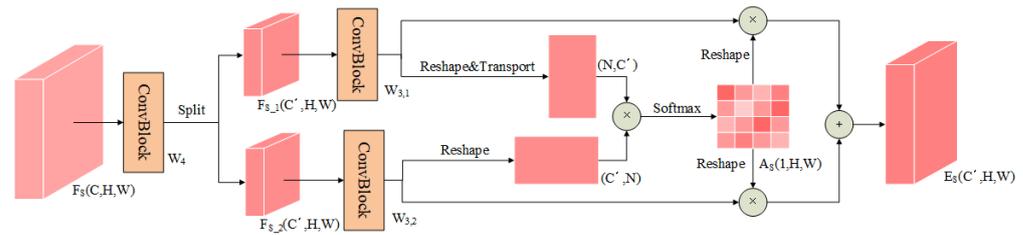


Figure 4. Spatial group self-attention module (SGAM).

Finally, we superimpose the attentional information on F_{s_1} and F_{s_2} . The final output feature map E_s is obtained with the following equation:

$$E_s = A_s^T F_{s_1} + A_s^T F_{s_2} \quad (5)$$

Channel group self-attention module: Since high-level channels tend to be strongly correlated, some channels share similar semantic information. To mine semantically related channels, we designed a new attention module called the CGAM.

Similar to the SGAM, the CGAM uses group convolution to generate query-specific attention weights. We use the output feature E_s of the SGAM module as the input features of the CGAM, and the pairwise feature maps F_{c_1} and F_{c_2} are calculated with the following equation:

$$[F_{c_1}, F_{c_2}] = W_5(E_s) \quad (6)$$

where $F_{c_1}, F_{c_2} \in \mathbb{R}^{C' \times H \times W}$. W_5 is the parameter matrix of 1×1 group convolution. Then, we resize F_{c_1} and F_{c_2} to $\mathbb{R}^{C' \times N}$ and the channel affinity matrix $A_c \in \mathbb{R}^{1 \times H \times W}$ is computed as:

$$A_c = \text{Softmax}(F_{c_1}^T F_{c_2}) \quad (7)$$

Then, the feature map with channel cues E_c is obtained based on the following equation:

$$E_c = A_c^T F_{c_1} + A_c^T F_{c_2} \quad (8)$$

Finally, the output feature representation F_{out} is obtained through a shortcut operation as

$$F_{out} = \text{LeakyReLU}(\text{BN}(W_6(E_c) + \text{ConvBlock}(F_s))) \quad (9)$$

where $F_{out} \in \mathbb{R}^{C \times H \times W}$. W_6 is the parameter matrix of 1×1 convolution.

3.3. Hierarchical Feature Pyramid Network

The feature pyramid network (FPN) improves the accuracy of object detection algorithms by fusing multiscale features. As shown in Figure 5, YOLOv4-Tiny first generates feature maps in various stages {C2, C3, C4, C5}. Then, the FPN obtains P5 from C5 through a 1×1 convolution operation and uses top-down upsampling and horizontal connection operations to generate the fusion feature P4. However, P4 fails to effectively utilize low-level feature information, leading to low accuracy in detecting small targets. To solve

this problem, we propose a new feature fusion network. Beyond the previous works, we argue that maximum pooling gathers important clues about distinct object features, and average pooling helps mitigate the loss of feature information as the network deepens. Thus, instead of using maximum pooling alone, such as in the FPN, we apply average pooling and maximum pooling operations simultaneously in downsampling to extract semantic features. Based on the multiscale prediction, we fuse high-level semantic features with low-level features in a hierarchical manner to use multiscale feature maps fully. As demonstrated in Figure 6, we concatenate the fourfold downsampling results of C2 and the twofold downsampling results of C3 to generate an efficient feature descriptor and then perform channel reduction through a 1×1 convolution operation to obtain a feature map M2. We concatenate the upsampling result of C5 with C4 and then reduce the feature dimension to generate a feature map M4. Finally, we perform channel reduction based on M2 and M4 to obtain a feature map {P4, P5} with detailed information. This structure combines low-resolution but semantically rich features with high-resolution but poor semantic features through a top-down path and hierarchical feature fusion. Without significantly increasing the computational complexity, this method further enriches the semantic information of the feature maps.

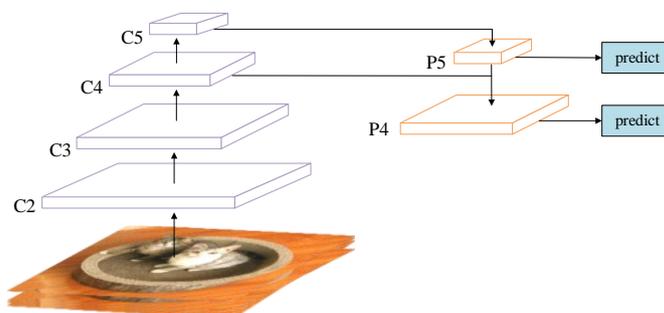


Figure 5. Feature pyramid network (FPN).

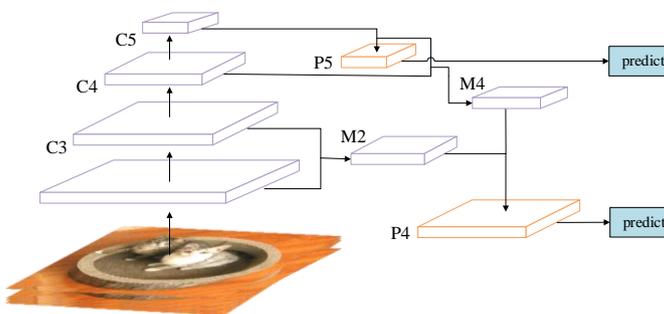


Figure 6. Hierarchical feature pyramid network (H-FPN).

3.4. Loss Function

The multitask loss function used to evaluate model performance is defined as follows:

$$Loss = L_{coord} + L_{conf} + L_{cls} \tag{10}$$

CIOU (Complete-IOU) [50] is used to describe the coordinate loss. CIOU considers three critical geometric measures, i.e., overlap area, central point distance, and aspect ratio, which make the predicted box regression stable. In this paper, CIOU is used to measure the performance of the predicted box, which is defined as follows:

$$L_{coord} = 1 - IOU + \frac{\rho^2(B, B^{gt})}{c^2} + \alpha v \tag{11}$$

where IOU is the overlap ratio between the predicted box B and the ground truth box B^{gt} , $\rho(\cdot)$ is the Euclidean distance, c is the diagonal distance for the minimum outer rectangle

between the predicted box and ground-truth box, and av is used to monitor the aspect ratio of the predicted box.

BCE (BinaryCrossEntropyLoss) is used to describe the confidence loss and is defined as follows:

$$L_{conf} = - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{obj} l(C_i^j, \hat{C}_i^j) - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{noobj} l(C_i^j, \hat{C}_i^j) \quad (12)$$

where S^2 is the number of cells and B is the number of predicted boxes per cell. I_{ij}^{obj} and I_{ij}^{noobj} indicate whether the j -th anchor in the i -th cell is responsible for predicting. \hat{C}_i^j is the object confidence of the ground truth box, and C_i^j is the object confidence of the predicted box. The binary cross-entropy loss $l(C_i^j, \hat{C}_i^j)$ is defined as follows:

$$l(C_i^j, \hat{C}_i^j) = \hat{C}_i^j \ln \hat{C}_i^j + (1 - \hat{C}_i^j) \ln(1 - \hat{C}_i^j) \quad (13)$$

CE (CrossEntropyLoss) is used to describe the classification loss, as described below:

$$L_{cls} = - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{obj} \left(\log \frac{P_i^j(\hat{c})}{\sum_{c \in \text{classes}} P_i^j(c)} \right) \quad (14)$$

where $P_i^j(\hat{c})$ is the probability of the true target; $P_i^j(c)$ is the probability of other targets.

4. Experiment

In this paper, we use mAP, BFLOPs, FPS, and parameter size as performance metrics for the proposed model. The mean average precision (mAP) evaluates the detection effect of multicategory objects by calculating the area under the PR curve. This paper uses the mAP formula proposed in VOC2012. The AP is calculated by the following equation:

$$AP = \sum_{r=0}^1 (r_{n+1} - r_n) \max_{r' \geq r_{n+1}} (P_{r'}) \quad (15)$$

where P_r is the precision at recall level r . The mAP is the average of the AP values for each category of samples and is defined as follows:

$$mAP = \frac{\sum_{k=1}^M AP(k)}{M} \quad (16)$$

where M is the number of classes in the test set.

BFLOPs (Billion Floating Point Operations) is used to measure the computational complexity of a model. FPS (frames per second) is used to evaluate the inference time of the model to process each image and analyze real-time performance. Parameter size is used to reflect the size of the model and determines the model occupies in memory.

4.1. Datasets

The PASCAL VOC [51] and MS COCO [52] datasets are used to evaluate the performance of the proposed model. These two datasets are authoritative datasets widely used in the fields of image classification, object detection, and semantic segmentation. Detailed information on the two datasets is shown in Table 1. We believe that using mainstream PASCAL VOC and MS COCO as training and testing sets to compare evaluation indicators with advanced models can fairly reflect the quality of the model.

The PASCAL VOC datasets consist of 20 categories of objects, including PASCAL VOC 2007 and 2012. We use the mixed dataset composed of PASCAL VOC 2007 and 2012 for training and testing. The training set consisted of 16,511 images, and the test set consisted of 4592 images. The MS COCO dataset contains 80 categories of objects with 118,287 images

in the training set. There are 40,670 images in the MS COCO test dataset. Compared with the PASCAL VOC dataset, the MS COCO dataset mainly involves complex daily scenes, more ground truth boxes, and small targets in a single image. The main evaluation criterion is the average of 10 values of IOU from 0.5 to 0.95, which can comprehensively reflect the performance of the algorithm.

Table 1. Detailed comparison between PASCAL VOC and MS COCO.

Dataset	Training Images	Testing Images
PASCAL VOC	16,551	4952
MS COCO	118,287	40,670

4.2. Implementation Details

We use the public PyTorch framework to implement our approach and utilize transfer learning in our experiments to improve performance. In the experiment, we use the same parameter configuration for the Pascal VOC and COCO datasets. We repeat each experiment three times and calculate the average value as the result. The batch size is set to 16, and the Adam gradient optimizer is used. Data enhancement methods include angle rotation, saturation change, exposure change, and hue change. We use a two-stage transfer learning algorithm to train the network parameters. First, we freeze the backbone weights and train other network parameters with an initial learning rate of 0.001 and decay 10 times in the 60th epoch for a total of 150 epochs of training. Then, we train the entire network for another 350 epochs with an initial learning rate of 0.0001 and decay 10 times in the 210th epoch until the end of the training period is reached. We train and verify the proposed model on a desktop PC. In addition, NVIDIA Jetson Nano [53] is used as an embedded device for performance testing. The configuration details of the intelligent devices are reported in Table 2.

Table 2. Hardware environment configuration information.

Platform	CPU	GPU	Memory
Desktop PC	Intel Core i9-9900 KF 3.60 GHz	NVIDIA TITAN RTX	16 GB
Jetson Nano	Quad-Core ARM Cortex-A57 MPCore	128-Core Maxwell	4 GB

5. Discussions

To evaluate the performance of the proposed network compared to that of other YOLO methods, Mini-YOLOv4 is compared with YOLOv3, YOLOv3-Tiny, YOLOv4-CSP, and YOLOv4-Tiny. The PASCAL VOC dataset introduced in Section 4 is used.

As shown in Table 3, YOLOv4-CSP achieves the best detection accuracy but requires the largest number of parameters and has a slow detection speed. In contrast, Mini-YOLOv4 requires fewer parameters and computations than YOLOv4-CSP and is able to obtain suboptimal detection results compared to those of YOLOv3. Mini-YOLOv4 achieves the mAP of 79.0%, which is 17.8% and 3.2% higher than those of YOLOv3-Tiny and YOLOv4-Tiny, respectively. Such results imply that Mini-YOLOv4 provides higher classification accuracy for multcategory objects. Compared with the original model YOLOv4-CSP with a BFLOPs value of 29.9, the BFLOPs value of Mini-YOLOv4 is approximately 10 times smaller. In terms of the parameter size, Mini-YOLOv4 requires only 4.7 M parameters, which is 15 times fewer than that of YOLOv3 and YOLOv4-CSP. In addition, compared to YOLOv4-Tiny, Mini-YOLOv4 requires 20.3% fewer parameters. The FPS of our network is 172.6, which is 3.1 times as many as obtained with YOLOv4-CSP. Compared to YOLOv4-Tiny, the real-time performance of Mini-YOLOv4 is reduced by 16.1%. Figure 7 shows some qualitative examples of object detection results for different detection models on the PASCAL VOC 2007 test dataset. Each rectangular bounding box shown in the image contains information

about the predicted category as well as the confidence score. Obviously, our detection model can find objects of interest more comprehensively and locate positions more accurately than YOLOv4-Tiny. That is because MFBlock and GSBlock in the proposed network increase the amount of contextual information by expanding the receptive field. Moreover, with hierarchical feature fusion in the H-FPN, the proposed network can accurately locate small and medium-sized targets. This is because MFBlock replaces the last 3×3 convolutional layer and focuses the network's attention on regions of interest, capturing important features through attention mechanisms and enhancing the expression ability of features. Next is the group attention module, which proposes a spatial group attention module and a channel group attention module, focusing on calculating the semantic correlation of the spatial domain and channel domain to capture rich global information and obtain comprehensive feature representations. Finally, the improved H-FPN focuses on the fusion of high-level semantic features and low-level semantic features, compensating for the accumulation of lost feature information in the downsampling stage and enhancing target detection capabilities. The experimental results on PASCAL VOC show that Mini-YOLOv4 has excellent feature extraction capability and effectively reduces the number of model parameters and the number of calculations.

Table 3. Performance comparison of different models on the PASCAL VOC dataset.

Method	Backbone	Size	mAP (%)	BFLOPs	FPS	Parameter Size (M)
YOLOv3	Darknet53	416	79.2	33.0	53.8	61.9
YOLOv3-Tiny	Darknet19	416	61.2	2.8	277.7	8.8
YOLOv4-CSP	CSPDarknet-53s	416	85.7	29.9	58.2	64.0
YOLOv4-Tiny	CSPDarknet-Tiny	416	75.8	3.4	205.8	5.9
PPYOLO-Tiny [54]	MobileNetv3	416	76.2	0.29	467.4	1.3
YOLOv5s [55]	CSPDarknet-Tiny	416	78.7	3.5	106.7	7.1
YOLOX-Tiny [56]	CSPDarknet-Tiny	416	79.4	3.2	150.3	5.0
Mini-YOLOv4	CSPDarknet-Tiny	416	79.0	3.1	172.6	4.7

In order to further evaluate the superiority of Mini-YOLOv4 compared to other lightweight object detection network architectures, we take MobileNetv1-YOLOv4, MobileNetv2-YOLOv4, MobileNetv3-YOLOv4, ShuffleNetv1-YOLOv4, and ShuffleNetv2-YOLOv4 into comparison. All models are trained on the PASCAL VOC 2007 + 2012 training dataset and tested on the PASCAL VOC 2007 test dataset.

Table 3 summarizes the comparative results for the models. Compared to the other lightweight YOLOv4 methods, since Mini-YOLOv4 uses only two feature map scales for prediction, it requires fewer parameters and calculations and provides a faster detection speed. PPYOLO-Tiny uses MobileNetv3 as the backbone network, benefiting from the use of depthwise separable convolution instead of traditional convolution operations and post-quantization strategy. The FPS value and BLOPs results of the model are excellent. However, due to the pursuit of extreme speed optimization, it reduces feature extraction ability, and the mAP value is 2.8% lower than MiniYOLOv4. It is worth noting that YOLOv5s, which is also based on CSPDarknet-Tiny as the backbone network, uses three prediction heads for detection. Compared to our proposed MiniYOLOv4, which uses two prediction heads for detection, YOLOv5s requires more inference time and post-processing time to generate the final detection results. Therefore, it has almost half the FPS value and double the BLOPs compared to the model proposed in this paper.

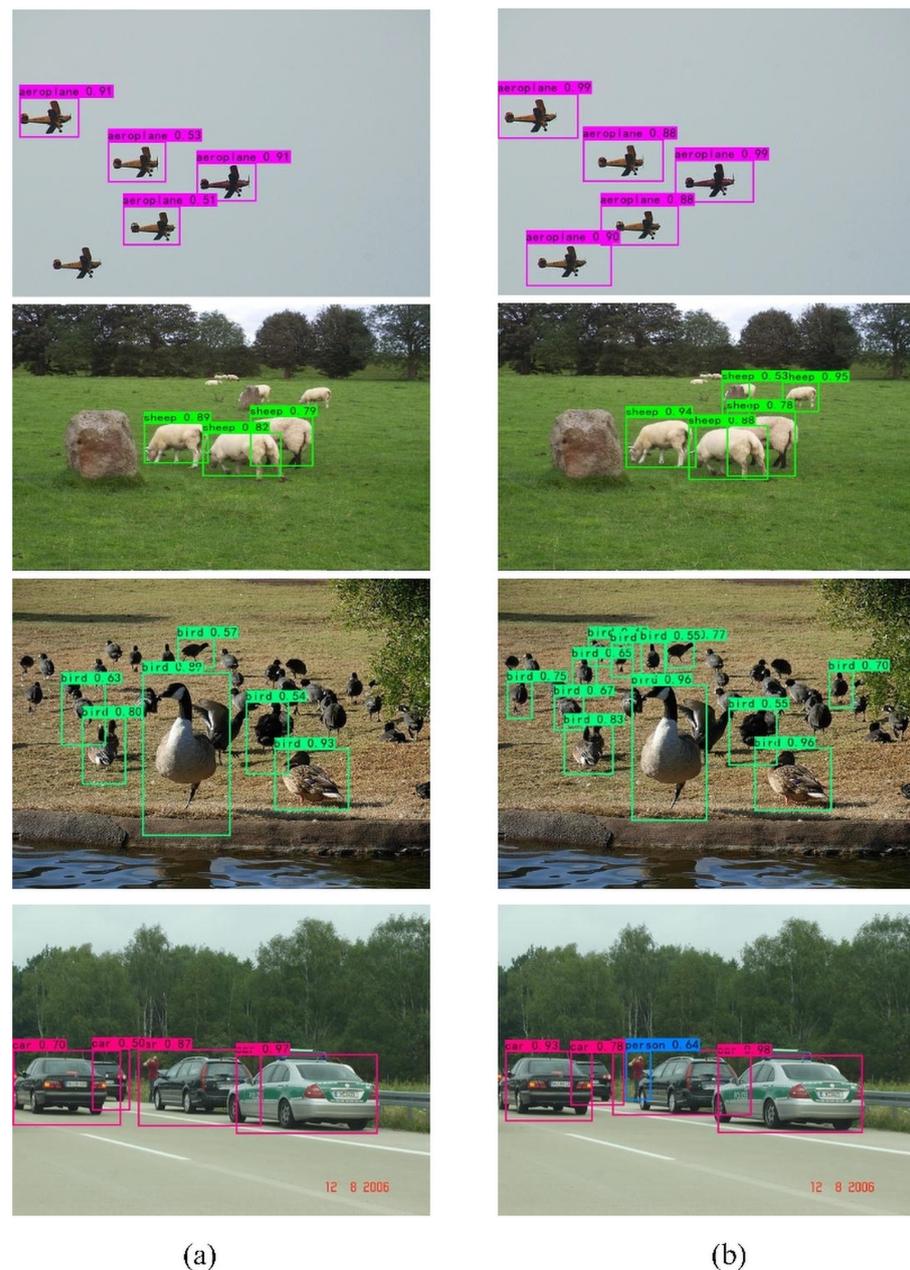


Figure 7. Comparison of different algorithms on PASCAL VOC 2007 test dataset. (a) YOLOv4-Tiny. (b) Mini-YOLOv4.

As shown in Table 4, MobileNetv3-YOLOv4 outperforms Mini-YOLOv4 by 1.4% mAP due to the complex feature extraction network, but Mini-YOLOv4 achieves better results in terms of BFLOPs, FPS, and parameter size. Compared with MobileNetv2-YOLOv4, Mini-YOLOv4 has similar accuracy, but the BFLOPs value and parameter size are reduced by 38.0% and 64.7%, respectively. For real-time performance, MobileNetv1-YOLOv4 achieves 77.6 FPS, ShuffleNetv1-YOLOv4 achieves 89.6 FPS, and ShuffleNetv2-YOLOv4 achieves 120.3 FPS. Compared to these three models, Mini-YOLOv4 improves the real-time performance by 122.4%, 92.6%, and 43.5%, respectively. In terms of parameter size, Mini-YOLOv4 requires approximately 3 times fewer parameters than the other models, and the number of parameters is reduced by 57.3% and 54.4% compared to those required by ShuffleNetv1-YOLOv4 and ShuffleNetv2-YOLOv4. It can be seen that although we use group convolution to reduce the amount of calculation like the ShuffleNet series of networks, our model creatively combines it with the attention mechanism to reduce the

model size while improving the feature extraction capability of the model. According to the experimental results, it can be concluded that Mini-YOLOv4 achieves an excellent trade-off between accuracy and speed.

Table 4. Performance comparison of lightweight models on PASCAL VOC dataset.

Method	Backbone	Size	mAP (%)	BFLOPs	FPS	Parameter Size (M)
MobileNetv1-YOLOv4	MobileNetv1-1.0	416	79.6	5.0	97.9	13.3
MobileNetv2-YOLOv4	MobileNetv2-1.0	416	80.1	3.8	77.6	12.1
MobileNetv3-YOLOv4	MobileNetv3-large	416	80.4	3.6	65.5	14.0
ShuffleNetv1-YOLOv4	ShuffleNetv1-1.0	416	77.6	3.5	89.6	11.0
ShuffleNetv2-YOLOv4	ShuffleNetv2-1.0	416	78.8	3.4	120.3	10.3
Mini-YOLOv4	CSPDarknet-Tiny	416	79.0	3.1	172.6	4.7

We compare Mini-YOLOv4 with state-of-the-art object detection networks on the MS-COCO test-dev dataset. As shown in Table 5, most object detectors achieve excellent real-time performance and high detection accuracy. Notably, our proposed model enlarges the receptive field and enhances the feature extraction capability of the network, yielding significant improvements of 10.2% and 3.5% in the overall detection accuracy relative to that of YOLOv3-Tiny and YOLOv4-Tiny, respectively. Mini-YOLOv4 facilitates the interaction of low-level feature maps with high-level feature maps and enriches the semantic information of different scales. Compared to YOLOv4-Tiny, Mini-YOLOv4 achieves notable increases of 4.9% and 1.4% in detecting medium and small objects, respectively. It can be found that ASFF provides the highest overall detection accuracy of 38.1 but achieves poor real-time performance for an input size of 320×320 . In comparison, our method is 4.2 times faster and achieves a slightly lower detection accuracy. For 512×512 input size, YOLOv4-CSP provides the best detection results and achieves real-time detection speed. In contrast, our method, with better accuracy than YOLOv4-Tiny, is 3.1 times faster than YOLOv4-CSP. Compared to PPYOLO-Tiny, Mini-YOLOv4 achieves improvements of 3.7%, 2.9%, and 3.4% in overall detection accuracy at the three different resolutions, respectively. Although YOLOX-Tiny outperforms Mini-YOLOv4 in terms of detection accuracy, Mini-YOLOv4's rapid inference capability allows it to handle more deep learning tasks compared to YOLOX-Tiny. These results emphasize the robust performance and versatility of Mini-YOLOv4 as an efficient object detection model for various resolution settings and real-time applications.

Table 5. Comparison with state-of-the-art object detection algorithms on COCO test-dev.

Method	Backbone	Size	FPS	AP	AP50	AP75	APS	APM	APL
SSD [25]	VGG-16	300	64.5	25.1	43.1	25.8	6.6	25.9	41.4
DSSD [57]	ResNet-101	320	13.8	28.0	46.1	29.2	7.4	28.1	47.6
RefineDet [58]	VGG-16	320	55.1	29.4	49.2	31.3	10.0	32.0	44.4
RFBNet [45]	VGG-16	300	84.6	30.3	49.3	31.8	11.8	31.9	45.9
M2det [59]	VGG-16	320	52.8	33.5	52.4	35.6	14.4	37.6	47.6
ASFF [44]	Darknet-53	320	42.6	38.1	57.4	42.1	16.1	41.6	53.6
LRF [60]	VGG-16	320	92.7	32.0	51.5	33.8	12.6	34.9	47.0
YOLOv4-Tiny	CSPDarknet-Tiny	320	218.3	20.5	38.5	20.5	7.0	24.1	28.4
PPYOLO-Tiny	MobileNetv3	320	486.8	20.6	39.0	21.2	8.2	25.7	29.3
YOLOv5s	CSPDarknet-Tiny	320	115.4	27.2	45.6	28.1	10.1	32.4	36.2
Mini-YOLOv4	CSPDarknet-Tiny	320	181.4	24.3	43.4	24.1	8.5	29.4	33.8

Table 5. Cont.

Method	Backbone	Size	FPS	AP	AP50	AP75	APS	APM	APL
RetinaNet [30]	ResNet-101	400	17.3	31.9	49.5	34.1	11.6	35.8	48.5
YOLOv3 [28]	Darknet-53	416	53.8	31.0	55.3	32.3	15.2	33.2	42.8
YOLOv3-Tiny [28]	Darknet-19	416	277.8	15.3	33.1	12.4	4.4	15.2	25.1
ASFF [44]	CSPDarknet-Tiny	416	37.4	40.6	60.6	45.1	20.3	44.2	54.1
YOLOv4 [29]	CSPDarknet-53	416	57.2	41.2	62.8	44.3	20.4	44.4	56.0
YOLOv4-Tiny [33]	CSPDarknet-Tiny	416	205.8	22.0	40.2	21.7	8.4	26.3	29.2
PPYOLO-Tiny	MobileNetv3	416	467.4	22.6	41.2	23.1	8.9	28.8	31.4
YOLOv5s	CSPDarknet-Tiny	416	106.7	28.1	46.8	30.4	11.4	34.1	38.7
Mini-YOLOv4	CSPDarknet-Tiny	416	172.6	25.5	44.9	25.2	9.8	31.2	34.1
CornerNet [61]	Hourglass	512	6.9	40.5	57.8	45.3	20.8	44.8	56.7
PFPNet-R [62]	VGG-16	512	32.8	35.2	57.6	37.9	18.7	38.6	45.9
HSD [63]	VGG-16	512	30.8	38.8	58.2	42.5	21.8	41.9	50.2
EFGRNet [64]	VGG-16	512	33.5	37.5	58.8	40.4	19.7	41.6	49.4
EfficientDet-D0 [65]	EfficientDet-B0	512	45.6	33.8	52.2	35.8	12.0	38.3	51.2
YOLOv4 [29]	CSPDarknet-53	512	50.3	43.0	64.9	46.5	24.3	46.1	55.2
YOLOv4-CSP	CSPDarknet-53s	512	52.6	46.2	64.8	50.2	24.6	50.4	61.9
YOLOv4-Tiny	CSPDarknet-Tiny	512	190.2	23.5	43.0	23.5	10.1	28.4	29.9
PPYOLO-Tiny	MobileNetv3	512	431.7	23.9	44.1	24.2	10.4	30.2	31.3
YOLOv5s	CSPDarknet-Tiny	512	94.6	30.6	48.7	32.4	13.2	36.4	40.3
Mini-YOLOv4	CSPDarknet-Tiny	512	163.3	27.3	47.8	27.1	11.6	33.7	34.8

To illustrate why the proposed model can improve detection accuracy, we use Grad-CAM [66] as an attention extraction tool to visualize YOLOv3-Tiny, YOLOv4-Tiny, and Mini-YOLOv4 on the MS COCO test dataset. In Figure 8, we select output feature maps with sizes of 13×13 and 26×26 for the comparison of attention maps. For the 13×13 size, the attention map generated by Mini-YOLOv4 can locate large target objects more accurately and does not include much of the background area. For the size of 26×26 , Mini-YOLOv4 effectively confines attention to the semantic area of small and medium targets. It is clear that our proposed network can locate foreground objects more accurately than the other networks, regardless of the size and shape. This is because MFBlock enhances the expression ability of features through attention mechanism and feature fusion, providing a solid foundation for subsequent feature extraction operations. In addition, before the final convolution operation generates feature vectors, a spatial group self-attention module and a channel group self-attention module are inserted, which are used to model feature correlations in the spatial domain and channel domain. Based on the above method, the attention module effectively suppresses redundant information and captures rich global information to enhance feature representation capabilities. Finally, the proposed H-FPN utilizes the fusion of high-level semantic features and low-level semantic features for prediction, further enhancing the interaction of low-level semantic information, compensating for the accumulation of lost feature information in the downsampling stage, and enhancing target detection capabilities. This result indicates that the proposed model further enhances the detection accuracy for small and medium targets.

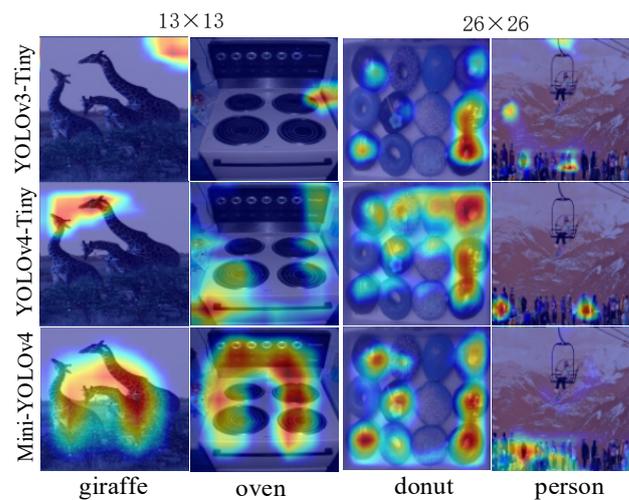


Figure 8. Visualization of attention maps.

In order to solve the problems of poor model framework compatibility and slow model running speed, the model reasoning deployment framework comes into being. It unifies the model format and greatly accelerates the inference speed. TensorRT is one of the mainstream model deployment frameworks, and it performs well on NVIDIA Jetson series devices. This TensorRT deployment adopts a two-stage process. In the first stage, the Pytorch model is converted into an FP32-precision ONNX model file using the Pytorch API. In the second stage, the ONNX file is parsed through the TensorRT API, the TensorRT engine with FP16 precision is built, and the final deployment is completed. Compared with directly deploying the original model, this process solves the compatibility problem of the Pytorch model on the Jetson device and also reduces the model accuracy from FP32 to FP16, accelerating the inference speed. Through the introduction of the deployment framework, model training and deployment are decoupled, and the performance of the model is optimized, providing a guarantee for wide use in practical application scenarios.

To verify the detection speed of the proposed model on embedded platforms, we use the above method to compare Mini-YOLOv4 with YOLOv4-CSP and YOLOv4-Tiny on a Jetson Nano device [53]. We believe that the Jetson Nano device is currently the mainstream embedded device, and it is reasonable to use for speed standard measurement. The comparison results are shown in Table 6. Limited by the computing power of the embedded platform, YOLOv4-CSP, with high detection accuracy, can only achieve a maximum detection speed of 3.7 FPS on the Jetson Nano. The proposed Mini-YOLOv4 achieves detection speeds of 13.7 FPS, 16.4 FPS, and 25.6 FPS for the three input sizes considered. In order to improve the detection accuracy, we optimize the original YOLOv4-Tiny network structure so that the detection speed of Mini-YOLOv4 is slightly inferior to that of YOLOv4-Tiny, but it still achieves a real-time detection speed. The experiments show that Mini-YOLOv4 can achieve real-time performance on the embedded device, and the detection accuracy is greatly improved compared with that of YOLOv4-Tiny. This finding suggests that Mini-YOLOv4 is suitable for target detection on embedded devices.

The performance of these three algorithms is also measured in terms of resource usage, such as GPU and memory usage on the Jetson Nano. We monitor for 60 min, of which the machine is idle for the first 30 min (only background tasks are running), and the detection task is executed in the last 30 min. Figure 9 shows the comparison result of memory usage. Notably, the initial memory usage of the Jetson Nano is approximately 32%. At about 30 min, the memory usage of YOLOv4-CSP jumps to 88%, the memory usage of YOLOv4-Tiny jumps to 51%, and the memory usage of Mini-YOLOv4 jumps to 45%. This phenomenon is mainly due to the small number of parameters in the proposed network, which is extremely lightweight. In Figure 10, when the detection model starts to work, the GPU usage rate of YOLOv4-CSP far exceeds that of the other two models, remaining at

around 88%. Compared with YOLOv4-Tiny, the proposed model has a similar usage rate. This is because the number of model calculations is reduced in the proposed approach while the detection accuracy is greatly enhanced.

Table 6. Performance comparison of different network architectures on Jetson Nano.

Model	Size	FPS
YOLOv4-CSP	320	3.7
YOLOv4-Tiny	320	28.4
Mini-YOLOv4	320	25.6
YOLOv4-CSP	416	2.4
YOLOv4-Tiny	416	17.8
Mini-YOLOv4	416	16.4
YOLOv4-CSP	512	1.3
YOLOv4-Tiny	512	14.5
Mini-YOLOv4	512	13.7

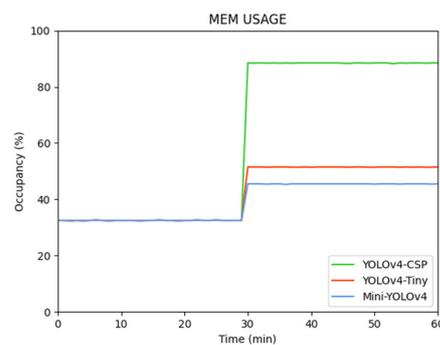


Figure 9. Memory usage vs execution time.

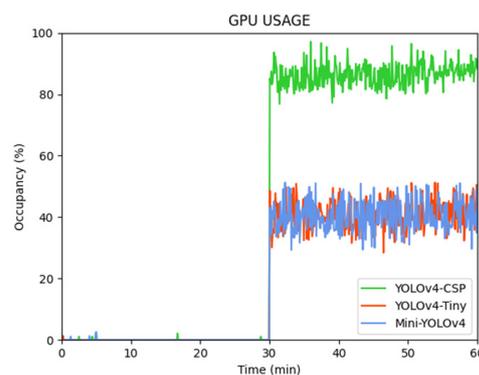


Figure 10. GPU usage vs execution time.

Ablation Study

To verify the impact of the proposed modules on the final performance, we performed ablation experiments by gradually adding MFBlock, GSBlock, and H-FPN to the baseline YOLOv4-Tiny. Table 7 shows the detection performance results for each model on the MS COCO test-dev dataset. In the case of tests under IOU = 0.5 and IOU from 0.50 to 0.95, the mAP values improve by 1.8% and 1.3% after adding MFBlock. GSBlock makes a significant contribution to improving detection accuracy by effectively mining contextual information. Additionally, SGAM improves overall performance by 0.9%, and CGAM improves overall performance by 1.2%. When these two modules are jointly integrated into the baseline, the overall detection accuracy is significantly improved by 1.8%. Based on

the introduction of MFBlock and GSBlock, the H-FPN can further improve the detection performance for small and medium targets by 0.5% and 0.6%, respectively. It is evident that the H-FPN enriches the semantic information by effectively fusing multiscale features. As demonstrated in Section 3.1, we learn from the idea of squeezing and activating to achieve attention calibration, which has proven to be useful for improving feature representation. Here, we study how the channel reduction rate in the CAM affects detection performance. In Table 8, we conduct experiments involving a series of channel reduction rates. It can be seen that a smaller ratio slightly increases the number of parameters of the model, but it does not bring about a performance improvement. As the reduction rate increases, the trend of the mAP value is to gradually increase and then decrease. Particularly, when the channel reduction rate is set to 8, the mAP value is 79.0%, which can achieve a superior balance between accuracy and complexity. In addition, we find that taking the feature representations before the cross-channel interaction as residuals by adding an identity connection, as shown in Figure 3, can further improve the detection accuracy. When the extra identity connection is removed, the detection accuracy of the model drops by 0.3%.

Table 7. Ablation experiments on MS COCO test-dev.

MFBlock	Module			AP	AP50	AP75	APS	APM	APL
	GSBlock		H-FPN						
	SGAM	CGAM							
				22.0	40.4	21.7	8.4	26.3	29.2
✓				23.3	42.2	23.1	8.8	27.8	30.9
✓	✓			24.2	43.4	24.0	9.0	29.2	32.2
✓		✓		24.5	43.6	24.2	9.2	29.5	32.8
✓	✓	✓		25.1	44.3	24.9	9.3	30.6	33.8
✓	✓	✓	✓	25.5	44.9	25.2	9.8	31.2	34.1
				22.0	40.4	21.7	8.4	26.3	29.2
✓				23.3	42.2	23.1	8.8	27.8	30.9
✓	✓			24.2	43.4	24.0	9.0	29.2	32.2
✓		✓		24.5	43.6	24.2	9.2	29.5	32.8

Table 8. Ablation experiments about the design choices of CAM on the PASCAL VOC dataset. ‘CR Rate’ is the channel reduction rate. ‘Identity’ refers to the corresponding component with the same name as in Figure 3.

Model	CR Rate	Identity	BFLOPs	Parameter Size (M)	mAP (%)
Mini-YOLOv4	2	✓	3.16	4.83	78.7
Mini-YOLOv4	4	✓	3.15	4.80	78.8
Mini-YOLOv4	8	×	3.15	4.78	78.7
Mini-YOLOv4	8	✓	3.15	4.78	79.0
Mini-YOLOv4	16	✓	3.15	4.77	78.8
Mini-YOLOv4	32	✓	3.15	4.77	78.5

To demonstrate the advantages of the proposed CAM over other powerful attention modules, we investigate the performance of various attention modules, including SENet, CBAM, and SKNet. In the experiments, we replaced SENet, CBAM, and SKNet with the CAM in our network to verify its efficiency. The metrics of BFLOPs, FPS, parameter size, and mAP on the PASCAL VOC dataset are shown in Table 9. Compared with SENet,

CAM focuses on both channel and spatial attention information, which improves the mAP by 0.3%. CBAM calibrates the feature distribution by considering soft attention mechanisms in the sequential channel-spatial structure, but it does not perform well on this dataset. The mAP value of Mini-YOLOv4-CBAM is 78.1%, which is lower than that of Mini-YOLOv4-CAM by 0.9%. SKNet obtains the features of different receptive fields through various convolution kernels of different sizes and aggregates information from multiple paths to obtain a global and comprehensive representation. In terms of BFLOPs, FPS, and parameter size, Mini-YOLOv4-CAM obviously outperforms Mini-YOLOv4-SKNet. These results demonstrate that CAM has powerful feature extraction capabilities and excellent calculation efficiency.

Table 9. Performance comparison of attention modules on the PASCAL VOC dataset.

Model	BFLOPs	FPS	Parameter Size (M)	mAP (%)
Mini-YOLOv4-SENet	3.12	184.2	4.64	78.6
Mini-YOLOv4-CBAM	3.12	179.1	4.64	78.1
Mini-YOLOv4-SKNet	3.17	160.4	5.01	79.2
Mini-YOLOv4-CAM	3.15	172.6	4.78	79.0

To further explore the impact of the network structure of the SGAM and CGAM on detection accuracy, we investigate different combinations of the SGAM and CGAM to achieve optimal performance. We study three ways of combinations: parallel with a fusion (Mini-YOLOv4-S//C), sequential spatial-channel (Mini-YOLOv4-SC), and sequential channel-spatial (Mini-YOLOv4-CS). Table 10 shows that Mini-YOLOv4-SC achieves the best performance, with an overall detection accuracy 0.9% and 0.3% higher than that of Mini-YOLOv4-S//C and Mini-YOLOv4-CS, respectively, on MS COCO. In the case of the test under IOU = 0.5, Mini-YOLOv4-SC gains 1.4% and 0.3% accuracy improvement over the other two combinations. Sequential architecture first generates refined features through the previous attention module and then allows the latter modules to learn attention information, which promotes optimization.

Table 10. Performance comparison of combinations of self-attention modules on MS COCO dataset.

Model	AP	AP50	AP75
Mini-YOLOv4-S//C	24.6	43.5	24.3
Mini-YOLOv4-CS	25.2	44.6	25.0
Mini-YOLOv4-SC	25.5	44.9	25.2

In addition to the above ablation experiments, the influence of different downsampling methods in H-FPN is assessed, and a comparison of the results is given in Table 11. We resize the feature maps of different layers in the network to the same scale by downsampling and then perform a fusion operation to obtain a comprehensive feature representation. In our experiments, we use different combinations of average pooling and maximum pooling for downsampling and observe the corresponding performance differences. As shown in Table 11, GAP + GMP provides the best detection results for the AP, AP50, and AP75 metrics, which significantly outperforms GAP or GMP alone. This result is because, potentially, using both pooling methods together not only extracts key features but also establishes connections among locations within the entire pooling window, allowing for local contextual information to be effectively captured.

Table 11. Performance comparison of different pooling methods on MS COCO dataset.

Pooling Method		AP	AP50	AP75
GAP	GMP			
✓		24.5	43.3	24.3
	✓	25.3	44.5	25.0
✓	✓	25.5	44.9	25.2

6. Conclusions

In this paper, we propose a lightweight real-time object detection method called Mini-YOLOv4. First, we propose a multibranch feature aggregation block to enlarge the receptive field and enhance the network feature extraction capability. Second, we design a group self-attention block (GSBlock) to capture long-range dependencies and explore contextual information. In the GSBlock, the spatial group self-attention module and the channel group self-attention module focus on capturing semantically relevant feature information. To improve the model detection accuracy for small and medium targets, a hierarchical feature pyramid network is proposed to fully exploit multiscale feature maps and fuse high-level semantic features with low-level feature representations in a hierarchical manner. Experiments conducted on the PASCAL VOC and MS COCO datasets indicate that Mini-YOLOv4 yields a higher detection accuracy than YOLOv4-Tiny. Specifically on the PASCAL VOC dataset, the mAP of Mini-YOLOv4 reaches 79.0%, which is 3.2% higher than YOLOv4-Tiny. Compared with other lightweight models (e.g., MobileNet-YOLOv4 series and ShuffleNet-YOLOv4 series), Mini-YOLOv4 achieves comparable results in mAP with lower BFLOPs and a real-time detection speed on the embedded platform NVIDIA Jetson Nano. Moreover, although MobileNetv3-YOLOv4 outperforms Mini-YOLOv4 by 1.4% mAP, Mini-YOLOv4 achieves better results in terms of BFLOPs, FPS, and parameter size. And it achieves 79.0% mAP, which is 0.2% higher than ShuffleNetv2-YOLOv4. Our model achieves a marvelous trade-off between accuracy and speed, enabling excellent performance in resource-constrained environments.

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

Funding: This work was supported by the National Natural Science Foundation of China (No. 12062009) and the Scientific Research Fund of Zhejiang Provincial Education Department (No. Y202352263).

Data Availability Statement: The data are available upon request.

Acknowledgments: We would like to give thanks to Wanfa Sun for his help in this paper.

Conflicts of Interest: The authors declare that they have no conflict of interest to report regarding the present study.

Abbreviations

YOLO	You Only Look Once
MFBBlock	Multibranch Feature Aggregation Module
CAM	Complete Attention Module
GSBlock	Group Self-attention Block
SGAM	Spatial Group Attention Module
CGAM	Channel Group Attention Module

H-FPN	Hierarchical Feature Pyramid Network
SENet	Squeeze Excitation Networks
CBAM	Convolutional Block Attention Module
SKNet	Selective Kernel Networks
NAM	Normalization-based Attention Module
ECA	Efficient Channel Attention
SA-Net	Shuffle Attention Networks
GAM	Global Attention Mechanism
SAT	Self-Adversarial Training
RFB	Receptive Field Block
PANet	Path Aggregation Network
FPN	Feature Pyramid Network
NMS	Non-Maximal Suppression
CIOU	Complete-IOU
BFLOPs	Billion Floating Point Operations
FPS	Frames Per Second

References

- Chen, R.; Liu, Y.; Zhang, M.; Liu, S.; Yu, B.; Tai, Y.-W. Dive deeper into box for object detection. In Proceedings of the 2020 European Conference on Computer Vision (ECCV): 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 412–428.
- Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; Fu, Y. Rethinking classification and localization for object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10186–10195.
- Qiu, H.; Li, H.; Wu, Q.; Shi, H. Offset bin classification network for accurate object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 13185–13194.
- Shi, H.; Zhou, Q.; Ni, Y.; Wu, X.; Latecki, L.J. DPNET: Dual-path network for efficient object detection with Lightweight Self-Attention. In Proceedings of the 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 16–19 October 2022; pp. 771–775.
- Termritthikun, C.; Jamtsho, Y.; Ieamsaard, J.; Muneesawang, P.; Lee, I. EEEA-Net: An early exit evolutionary neural architecture search. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104397. [[CrossRef](#)]
- Sun, Z.; Lin, M.; Sun, X.; Tan, Z.; Li, H.; Jin, R. MAE-DET: Revisiting maximum entropy principle in zero-shot nas for efficient object detection. In Proceedings of the 39th International Conference on Machine Learning (PMLR), Virtual, 17–23 July 2022; pp. 20810–20826.
- Chen, T.; Saxena, S.; Li, L.; Fleet, D.J.; Hinton, G. Pix2seq: A language modeling framework for object detection. *arXiv* **2022**, arXiv:2109.10852.
- Du, X.; Zoph, B.; Hung, W.-C.; Lin, T.-Y. Simple training strategies and model scaling for object detection. *arXiv* **2021**, arXiv:2107.00057.
- Khan, S.D.; Basalamah, S. Disam: Density independent and scale aware model for crowd counting and localization. *Vis. Comput.* **2021**, *37*, 2127–2137. [[CrossRef](#)]
- Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. PVT v2: Improved baselines with pyramid vision transformer. *Comput. Vis. Media* **2022**, *8*, 415–424. [[CrossRef](#)]
- Xin, Y.; Wang, G.; Mao, M.; Feng, Y.; Dang, Q.; Ma, Y.; Ding, E.; Han, S. PAFNet: An efficient anchor-free object detector guidance. *arXiv* **2021**, arXiv:2104.13534.
- Yang, J.; Liu, J.; Han, R.; Wu, J. Generating and restoring private face images for internet of vehicles based on semantic features and adversarial examples. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 16799–16809. [[CrossRef](#)]
- Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 568–578.
- Ke, W.; Zhang, T.; Huang, Z.; Ye, Q.; Liu, J.; Huang, D. Multiple anchor learning for visual object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10206–10215.
- Wang, J.; Zhang, W.; Cao, Y.; Chen, K.; Pang, J.; Gong, T.; Shi, J.; Loy, C.C.; Lin, D. Side-Aware boundary localization for more precise object detection. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 403–419.
- Cao, J.; Cholakal, H.; Anwer, R.M.; Khan, F.S.; Pang, Y.; Shao, L. D2Det: Towards high quality object detection and Instance Segmentation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11485–11494.
- Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.

18. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
19. Howard, A.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q.V.; Adam, H. Searching for MobileNetV3. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
20. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An extremely efficient Convolutional Neural Network for mobile devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
21. Ma, N.; Zhang, X.; Zheng, H.-T.; Sun, J. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In Proceedings of the 2018 European Conference on Computer Vision (ECCV): 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 116–131.
22. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
23. Tan, M.; Le, Q.V. MixConv: Mixed depthwise convolutional kernels. *arXiv* **2019**, arXiv:1907.09595.
24. Tan, M.; Le, Q. EfficientNet: Rethinking model scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
25. 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 2016 European Conference on Computer Vision (ECCV): 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
26. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
27. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
28. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:804.02767.
29. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
30. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
31. Lu, X.; Li, Q.; Li, B.; Yan, J. MimicDet: Bridging the gap between one-stage and two-stage object detection. In Proceedings of the 2020 European Conference on Computer Vision (ECCV): 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 541–557.
32. Mao, Q.-C.; Sun, H.-M.; Liu, Y.-B.; Jia, R.-S. Mini-YOLOv3: Real-Time object detector for embedded applications. *IEEE Access* **2019**, *7*, 133529–133538. [[CrossRef](#)]
33. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. Scaled-YOLOv4: Scaling cross stage partial network. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13029–13038.
34. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
35. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional block attention module. In Proceedings of the 2018 European Conference on Computer Vision (ECCV): 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 3–19.
36. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local neural networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7794–7803.
37. Li, X.; Wang, W.; Hu, X.; Yang, J. Selective kernel networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 510–519.
38. Cao, Y.; Xu, J.; Lin, S.; Wei, F.; Hu, H. GCNet: Non-local networks meet squeeze-excitation networks and beyond. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019; pp. 1971–1980.
39. Liu, Y.; Shao, Z.; Teng, Y.; Hoffmann, N. NAM: Normalization-based attention module. *arXiv* **2021**, arXiv:2111.12419.
40. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient channel attention for deep convolutional neural networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11531–11539.
41. Zhang, Q.-L.; Yang, Y.-B. SA-Net: Shuffle Attention for Deep Convolutional Neural Networks. In Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 2235–2239.
42. Yang, L.; Zhang, R.; Li, L.; Xie, X. SimAM: A Simple, Parameter-Free Attention Module for Convolutional Neural Networks. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 11863–11874.
43. Liu, Y.; Shao, Z. Hoffmann, Global attention mechanism: Retain information to enhance channel-spatial interactions. *arXiv* **2021**, arXiv:2112.05561.
44. Liu, S.; Huang, D.; Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv* **2019**, arXiv:1911.09516.

45. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [[CrossRef](#)]
46. Liu, S.; Huang, D.; Wang, Y. Receptive field block net for accurate and fast object detection. In Proceedings of the 2018 European Conference on Computer Vision (ECCV): 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 385–400.
47. Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
48. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for Instance segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768.
49. Huang, X.; Ge, Z.; Jie, Z.; Yoshie, O. NMS by representative region: Towards crowded pedestrian detection by proposal pairing. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10750–10759.
50. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 12993–13000.
51. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [[CrossRef](#)]
52. TLin, Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common objects in context. In Proceedings of the 2014 European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
53. Cass, S. Nvidia makes it easy to embed AI: The Jetson nano packs a lot of machine-learning power into DIY projects—[Hands on]. *IEEE Spectr.* **2020**, *57*, 14–16. [[CrossRef](#)]
54. Long, X.; Deng, K.; Wang, G.; Zhang, Y.; Dang, Q.; Gao, Y.; Shen, H.; Ren, J.; Han, S.; Ding, E.; et al. PP-YOLO: An effective and efficient implementation of object detector. *arXiv* **2020**, arXiv:2007.12099.
55. Couturier, R.; Noura, H.N.; Salman, O.; Sider, A. A deep learning object detection method for an efficient clusters initialization. *arXiv* **2021**, arXiv:2104.13634.
56. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
57. Fu, C.-Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional single shot detector. *arXiv* **2018**, arXiv:1701.06659.
58. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-Shot Refinement Neural Network for Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4203–4212.
59. Zhao, Q.; Sheng, T.; Wang, Y.; Tang, Z.; Chen, Y.; Cai, L.; Ling, H. M2det: A single-shot object detector based on multi-level feature pyramid network. In Proceedings of the 2019 AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 29–31 January 2019; pp. 9259–9266.
60. Wang, T.; Anwer, R.M.; Cholakkal, H.; Khan, F.S.; Pang, Y.; Shao, L. Learning rich features at high-speed for single-shot object detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1971–1980.
61. Law, H.; Deng, J. CornerNet: Detecting objects as paired keypoints. *Int. J. Comput. Vis.* **2020**, *128*, 642–656. [[CrossRef](#)]
62. Kim, S.-W.; Kook, H.-K.; Sun, J.-Y.; Kang, M.-C.; Ko, S.-J. Parallel feature pyramid network for object detection. In Proceedings of the European Conference on Computer Vision (ECCV): 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 234–250.
63. Cao, J.; Pang, Y.; Han, J.; Li, X. Hierarchical shot detector. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9705–9714.
64. Nie, J.; Anwer, R.M.; Cholakkal, H.; Khan, F.S.; Pang, Y.; Shao, L. Enriched feature guided refinement network for object detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9537–9546.
65. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and efficient object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10781–10790.
66. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 618–626.

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