MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

ODESA I.I. Mechnikov NATIONAL UNIVERSITY
(full name of higher education institution)

Faculty of mathematics, physics and information technologies
(full name of the institute, name of the faculty)

Department of mathematical support of computer systems
(full name of the department)

Qualification work

for obtaining the "master" higher education level
(education level)

on the topic

Object detection algorithms / Алгоритми детекції об'єктів

Performed by: a full-time student of the
specialty 126 – Information Systems & Technologies
(specialty code and name)
educational program «Information Systems & Technologies»
(educational program name)
Sun Zhihua
(Full Name)

Scientific supervisor Ph.D., Antonenko O.S.
(academic degree, academic title, surname and initials, signature)

Reviewer Sc.D., Prof. Volkov V.E.
(academic degree, academic title, surname and initials)

Reviewer Ph.D, Prof. Ruvinska V.M.
(academic degree, academic title, surname and initials)

Recommended for defense:
Minutes of the department meeting
No.___ of "___"_____ 2023

Head of Department
_____ Eugene MALAKHOV
(signature) (name, surname)

Defended at the EC No. ___ meeting
Minutes No. __ of "___" _____ 2023
Score _____/____/_____
(by national scale, ECTS scale, points)

Chairman of the EC
_____ Volodymyr VYCHUZHANIN
(signature) (name, surname)

Odesa - 2023

АНОТАЦІЯ

В останні роки технологія виявлення об’єктів стала популярною темою в дослідницькій галузі комп’ютерного зору, яка широко використовується в аналізі сцени БПЛА, відеоспостереженні, інтелектуальній медичній допомозі та інших сферах. В даний час виявлення великих і середніх об'єктів не може задовольнити реальні потреби. Люди висувають підвищені вимоги до продуктивності алгоритму. Особливо важливо покращити здатність алгоритму виявляти невеликі об’єкти. Однак малі об'єкти важко виявити через такі причини: низька роздільна здатність малих об'єктів, щільний розподіл тегів, велика різниця в масштабах між об'єктами та сприйнятливість до фонових перешкод. Зважаючи на вищезазначені проблеми, ми беремо YOLOv5s як базову модель і оптимізуємо основні блоки YOLOv5s, щоб отримати кращий ефект виявлення. Роботу можна підсумувати таким чином:

(1) З огляду на проблему відсутності виявлення через низьку роздільну здатність малих об’єктів, детектор P2 додано до голови YOLOv5s для виявлення менших об’єктів. Модель після вдосконалення змінює три-масштабне виявлення на чотири-масштабне виявлення, що покращує продуктивність багатомасштабного виявлення об’єктів. Крім того, детектор P2 поєднує в собі карти особливостей високої роздільної здатності, які можуть перетворювати більш дрібні особливості в більш глибокі. Таким чином інформація на картах функцій збагачується, що може ефективно вирішити проблему, коли невеликі об’єкти на великій відстані легко пропустити.

(2) Щоб вирішити проблему перешкод від шуму зображення та фону, модулі CBAM додано до Neck частини YOLOv5s. Механізм уваги використовується для оновлення карти функцій перед злиттям, щоб ігнорувати деяку нерелевантну інформацію та зосереджувати більше уваги на ключових функціях. Щоб карта функцій, яка використовується для виявлення об’єктів після злиття, містила ефективнішу інформацію. Завдяки додаванню

модуля CBAM посилюється захист від перешкод мережі, крім того, посилюються важливі характеристики об’єкта за рахунок розмірів каналу та простору. Це корисно підвищити увагу мережі до малих об’єктів і підвищити точність виявлення об’єктів.

(3) Щоб підвищити здатність обробки функцій YOLOv5s Neck і полегшити проблему недостатньої інформації про функції малих об’єктів, ми використовуємо ідею BiFPN для оптимізації структури об’єднання функцій YOLOv5s. На основі оригінальних шляхів об’єднання функцій «зверху вниз» і «знизу вгору» додано два горизонтальні перехресні з’єднання. Завдяки ефективним двонаправленим міжмасштабним зв’язкам високорівневі семантичні характеристики в глибокій карті функцій можуть бути повністю інтегровані з дрібнозернистими функціями в дрібній карті функцій, що може покращити передачу інформації між різними мережевими рівнями та збагатити особливість дрібних предметів.

Порівняно з YOLOv5s у наборі даних TinyPerson модель YOLOv5s-P2-CBAM-BiFPN має кращу продуктивність у P, R, mAP@.5 і mAP@.5:.95, збільшившись на 0.2%, 3.4%, 2.6% і 1.13% відповідно. У наборі даних VisDrone2019 модель YOLOv5s-P2-CBAM-BiFPN зросла на 5.1%, 5%, 5.7% і 3.8% відповідно.

Ключові слова: YOLOv5s; виявлення дрібних предметів; багатомасштабне виявлення об'єктів; CBAM; BiFPN

ABSTRACT

In recent years, object detection technology has become a hotspot in the research field of computer vision, which is widely used in UAV scene analysis, video surveillance, smart medical care and other fields. Nowadays, the detection of large and medium objects can't meet the actual needs. People put forward higher requirements for the performance of the algorithm. It is particularly important to improve the detection ability of the algorithm for small objects. However, small objects are difficult to detect because of the following reasons: poor resolution of small objects, dense distribution of tags, large scale difference between objects and susceptibility to background interference. In view of the above challenges, we take YOLOv5s as the baseline model, and optimize the Head and Neck of YOLOv5s to get better detection effect. The work can be summarized as follows:

(1) In view of the problem of missing detection due to the low resolution of small objects, a P2 detector is added to the Head of YOLOv5s to detect smaller objects. The model after improvement changes three-scale detection into four-scale detection, which improves the performance of multi-scale object detection. Moreover, the P2 detector combines high- resolution feature maps, which can transfer more shallow features to deep features. In this way, the information of feature maps is enriched, which can effectively improve the problem that small objects at a long distance are easy to miss detection.

(2) To solve the problem of interference from image noise and background, the CBAM modules are added to the Neck of YOLOv5s. The attention mechanism is used to update the feature map before fusion to ignore some irrelevant information and focus more attention on key features. So that the feature map used for object detection after fusion contains more effective information. By adding CBAM module, the anti- interference ability of the network is enhanced, moreover, the important features of the object are strengthened from the channel

and space dimensions. It is helpful to enhance the network's attention to small objects and improve the precision of object detection.

(3) In order to enhance the feature processing ability of YOLOv5s's Neck and alleviate the problem of insufficient feature information of small objects, we use the idea of BiFPN to optimize the feature fusion structure of YOLOv5s. On the basis of the original top-down and bottom-up feature fusion paths, two horizontal cross-scale connections are added. Through the way of efficient bidirectional cross-scale connections, the high-level semantic features in the deep feature map can be fully integrated with the fine-grained features in the shallow feature map, which can enhance the information transmission between different network layers and enrich the feature of small objects.

Compared with YOLOv5s, on the TinyPerson dataset, the YOLOv5s-P2-CBAM-BiFPN model has better performance in P, R, mAP@.5 and mAP@.5:.95 increased by 0.2%, 3.4%, 2.6% and 1.13% respectively. On the VisDrone2019 dataset, the YOLOv5s-P2-CBAM-BiFPN model increased by 5.1%, 5%, 5.7% and 3.8% respectively.

Key Words: YOLOv5s; small object detection; multi-scale object detection; CBAM; BiFPN

CONTENT

Page

1 INTRODUCTION

1.1 Research background and significance

Object detection is one of the important research topics in the field of computer vision, and it is also the cornerstone of tasks such as image description generation, object tracking, and scene understanding. Nowadays, with the rapid development of artificial intelligence and the upgrading of computer hardware equipment, breakthrough progress has been made in object detection algorithms, and related research results have also brought many conveniences to people's daily lives. For example, facial detection technology can strengthen monitoring of key locations and assist in epidemic prevention and control; Supermarket self-service payment and traffic violation tracking all rely on the support of object detection technology.

At present, the detection of large and medium-sized objects has achieved remarkable results. However, with the rapid development of intelligent systems and the widespread application of portable photography devices, a large number of small targets exist in videos and images. Simply detecting large and medium-sized objects is no longer sufficient to meet practical needs, and many fields require obtaining key information from small targets. This fully demonstrates that small object detection has great research value and application prospects. However, compared to large and medium-sized object detection, the performance indicators of the same detection algorithm for small object detection are often lower, making it difficult to achieve the expected results. This is because small targets have low resolution and limited information, making it easy to lose key features during downsampling, resulting in serious missed and false detections. In addition, the performance of the model is also affected by factors such as changes in lighting intensity, image noise, complex backgrounds, and target occlusion, which further increase the difficulty of detection.

Although there are many difficulties in small object detection, it has a wide range of application needs in practical production and life, which drives many

researchers to devote themselves to the research of small object detection. Its typical application scenarios are as follows:

(1) Applied to drone aerial image analysis. Drones are mainly used in high-altitude or dangerous areas that are difficult for manned aircraft to reach. They are of great significance for carrying out post disaster rescue work. They can monitor the situation of the disaster area in real-time and comprehensively, prevent collapse, aftershocks and other phenomena from endangering the safety of rescue personnel. The use of small object detection technology to analyze aerial images of disaster areas can better detect the specific location of trapped individuals, and also provide convenience for tasks such as planning rescue routes, selecting rescue material distribution sites, and determining severely affected areas. Drone aerial image analysis has also been applied in other fields. For example, analyzing urban aerial images can detect illegal buildings and facilitate better urban planning. It can also be used for forest fire monitoring, timely detecting remote forest fires that ground patrol cannot consider, and assisting in the development of forest fire prevention work. It can also be used in border monitoring and military reconnaissance, helping reconnaissance personnel detect suspicious targets in a timely manner, evaluate the threat level of unknown objects, and carry out precise military strikes.

(2) Applied to autonomous driving. Through target detection technology, pedestrians, vehicles, road obstacles, traffic signs and other information can be detected in real time to improve the auto drive system's ability to understand the vehicle driving scene, help the system judge the surrounding road conditions, traffic flow guidance and other information, and better plan the driving path. The auto drive system with superior performance can detect smaller obstacles, avoid obstacles that may cause traffic accidents in time, and ensure a safe distance from surrounding vehicles and pedestrians.

(3) Applied to smart healthcare. In order to effectively allocate medical resources, the urban smart healthcare platform has launched a series of digital and intelligent facilities. Artificial intelligence imaging diagnosis is the use of object detection technology to learn lesion features, and then locate and identify lesion

areas. The use of small object detection technology can detect small lesions that are difficult to observe with the naked eye, improving the accuracy of disease diagnosis. Due to the ability of intelligent diagnostic systems to process a large number of medical images in a short period of time, combining them with manual film reading can improve the efficiency of disease screening and reduce the workload of physicians.

(4) Applied to industrial automation. In industrial production, small object detection technology can detect small defects and cracks on the metal surface, defects on the surface of products such as fabrics and films, impurities and color differences in PVC powder, etc. By utilizing small object detection technology to promptly detect these defective products and repairing or removing them, a large number of defective products can be avoided from entering the market.

It can be seen that small object detection technology has great research value and broad application prospects. At the same time, there are also urgent problems that need to be further improved.

1.2 Current research status at home and abroad

In the process of gradually transitioning from large and medium-sized object detection to complex small object detection, scholars at home and abroad have done a lot of research work, which can be roughly divided into traditional methods based on manually designed features and deep learning algorithms using neural networks to extract deep features. This section will explore the topic of object detection from both traditional methods and deep learning based algorithms.

1.2.1 Traditional object detection algorithms

The main steps of traditional object detection algorithms are as follows: Firstly, preprocess the input image, including denoising, edge detection, and image grayscale transformation; The second step is to slide multiple small windows on the target image to construct candidate regions; Step three, extract the features of candidate regions and select representative feature vectors; Step 4, classify the features; The fifth step is to perform post-processing operations, which involves

filtering out redundant detection boxes through methods such as Non Maximum Suppression (NMS) [1] and Weighted Box Clustering (WBC) [2] .

Figure 1 1. Traditional object detection algorithm flowchart

Traditional object detection algorithms use manually designed feature extractors to extract image features. The commonly used feature extraction methods include Directional Gradient Histogram (HOG) [3] , Scale Invariant Feature Transform (SIFT) [4] , Haar Feature (Haar) [5] , Local Binary Pattern (LBP) [6] , and so on. Traditional classifiers include Support Vector Machine (SVM) [7] , AdaBoost [8] , and so on. The main detection methods during this period are as follows:

(1) VJ detector [5] . In 2001, P. Viola and M. Jones proposed the VJ detector for real-time face detection, which was a milestone event in the development history of face detection technology. In order to improve detection speed, the detector introduces integrated images to accelerate feature calculation, adopts AdaBoost algorithm to accelerate the selection of effective features, and utilizes a cascaded detection mechanism to reduce computational overhead. Although the detection speed of the VJ detector was faster than other algorithms at that time, the need to use sliding windows of different scales for global scanning of the image resulted in a large computational workload of the model, which far exceeded the computing power at that time.

(2) HOG detector [3] . In 2005, N. Dalal and B. Triggs proposed the HOG detector to solve pedestrian detection problems. The main principle of this detector is as follows: first, calculate the gradient of each pixel in the normalized image,

then divide the image into small connected regions, construct directional gradient histograms for each small region, finally merge these small regions into blocks, and normalize the gradient histogram to obtain the feature vector. The HOG detector has had a significant impact on many subsequent object detectors, especially the combination of HOG features with SVM classifiers, which has been widely used in image recognition tasks.

(3) Deformable Parts Model (DPM) [9] . The DPM algorithm was initially proposed by P. Felzenszwalb et al. in 2008, and later optimized by R. Girshick through a series of operations [10,11] . The initially proposed DPM algorithm is an extension of the HOG detector, which combines improved HOG features with SVM classifiers to achieve detection objectives. The DPM algorithm is one of the most successful detectors in traditional methods. Although it no longer has performance advantages compared to many current deep learning based object detection algorithms, the hybrid model, bounding box regression, and context priming ideas proposed in this algorithm have a significant impact on the development of subsequent algorithms. It is also used as a benchmark model for many algorithm improvements.

Traditional object detection algorithms mostly use sliding windows to generate candidate regions, which incurs significant system costs. Moreover, many rules and parameters need to be manually set to ensure algorithm performance. There are many uncontrollable factors in the training process of the model, and the stability of the algorithm is poor. Nowadays, traditional object detection algorithms are no longer capable of handling complex scene object detection tasks in terms of performance, especially for detecting small targets. This is because objects in small target images are densely arranged and have a large number of similar features, while traditional algorithms rely on the texture features of the target object to achieve detection goals, and their detection performance for small targets is poor.

1.2.2 Object detection algorithm based on deep learning

With the advancement of photography equipment and the rapid development of internet media, massive images and videos exist in the network, and traditional

algorithms based on manual feature construction can no longer meet the needs of big data environments. Therefore, researchers have gradually shifted their focus to deep neural networks, and Convolutional Neural Networks (CNN), which were once limited by hardware resources, have returned to the public eye. As shown in Figure 1.2, the birth time and latest research achievements of some classic convolutional neural networks are presented.

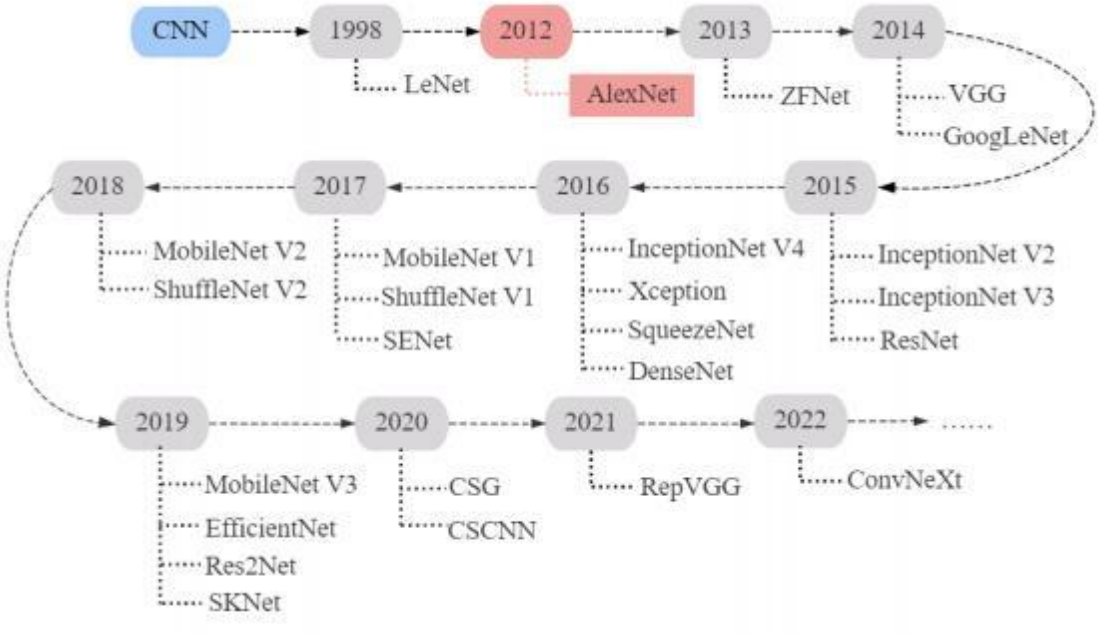


Figure 1.2 Development History of Convolutional Neural Networks

In 2012, AlexNet [12] emerged and won the championship of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) image classification track with an absolute advantage. The emergence of AlexNet is a major breakthrough in the field of computer vision, as it far surpasses traditional machine learning algorithms in performance, triggering a wave of deep learning. Subsequently, numerous deeper neural networks have been proposed. In 2014, the Oxford University team proposed VGG [13]. VGG usage 3 × Replacing the large convolution kernel with a small convolution kernel of 3 not only effectively reduces the number of parameters, but also enhances the nonlinear expression ability of the network. Compared to AlexNet, VGG networks are deeper and

perform better. Compared with GoogLeNet [14] proposed in the same year, VGG has a simpler structure, mainly composed of stacked convolutional and pooling layers, and has better portability. Due to GoogLeNet not only deepening the network layers but also expanding the network width, the network structure is relatively complex. The advantage of this design is that it can greatly reduce network parameters, accelerate model convergence speed, and prevent gradient vanishing and exploding phenomena in deep neural networks during training. GoogLeNet performs better than VGG in classification tasks. In order to further improve the algorithm performance, its team has carried out a series of optimization operations on the core module Inception of GoogLeNet and launched multiple versions of GoogLeNet [15-18].

Deep networks can extract more advanced semantic information from images, and their feature extraction ability is stronger compared to shallow networks. Many researchers blindly increase the depth of the network in order to improve network performance, but as the number of network layers increases, phenomena such as model degradation and gradient disappearance actually occur. It was not until the emergence of ResNet [19] in 2015 that this challenge was overcome. The residual structure proposed by ResNet can significantly improve the fitting ability of neural networks while ensuring network depth, effectively alleviating the problem of performance degradation in deep networks. Subsequently, G. Huang et al. proposed DenseNet [20] , in which the input of the current layer comes from the outputs of all previous layers. This feature reuse approach not only reduces the number of parameters, but also effectively alleviates the problem of gradient vanishing. Afterwards, J. Hu et al. proposed SENet [21] , which does not improve network performance from the spatial dimension like other networks, but instead takes a different approach by starting from the relationship between feature channels, sparking people's attention to channel attention.

SENet generates different weights for each channel to focus on the features of important channels and suppress the features of unnecessary channels. After

2017, researchers began researching models suitable for mobile devices, and a series of high-performance lightweight models such as MobileNet [22-24], ShuffleNet [25,26], and SqueezeNet [27] emerged.

The proposal of AlexNet ushered in the era of deep learning, bringing dawn to the bottleneck stage of object detection. Due to the powerful learning and feature expression capabilities of deep neural networks, the model can automatically learn more advanced features and contextual information, effectively compensating for the shortcomings of traditional object detection algorithms that can only extract shallow features such as textures and colors. So many researchers have begun to explore the application of deep convolutional neural networks in object detection tasks, and many high-performance algorithms have emerged, ushering in a period of rapid development in the field of object detection. Deep learning based object detection algorithms can be roughly divided into two categories: two-stage algorithms based on candidate regions, and one-stage algorithms based on regression. In Figure 1.3, the development history of these two types of object detection algorithms is shown.

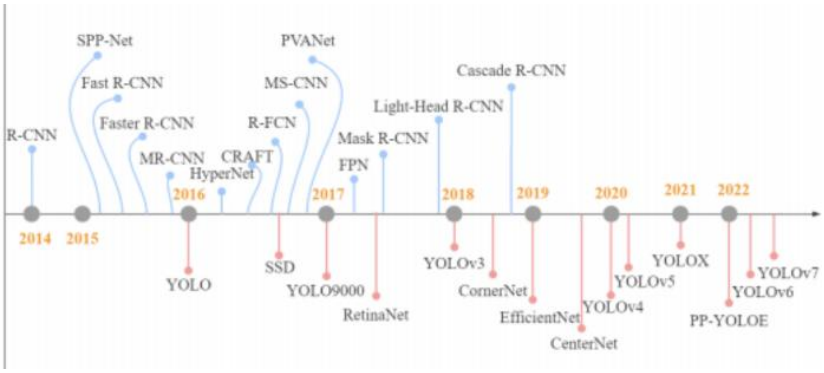


Figure 1.3 Development history of object detection algorithms based on deep learning

(1) Two stage object detection algorithm

The main representative of two-stage object detection algorithms is the R-CNN series. In 2014, R. Girshick et al. proposed the R-CNN algorithm [28], which was the first algorithm to use convolutional neural networks to solve object detection

tasks. R-CNN first generates several candidate regions that may contain targets, and then uses convolutional neural networks to extract the features of each candidate region. The features are then fed into an SVM classifier to determine the category of the target, and finally, a regressor is used to correct the position of the candidate boxes. Although R-CNN has significantly improved accuracy compared to traditional object detection algorithms, it still has the following drawbacks: the generated candidate regions will overlap, and there will be a lot of redundant calculations when performing convolution operations, resulting in long model training time, large spatial expenses, and slow detection speed. The second drawback is that the candidate regions must be scaled to the same size before being input into CNN to extract features, which can easily result in loss of image information.

He et al. proposed the Spatial Pyramid Pooling Network (SPP Net) [29] to compensate for the shortcomings of R-CNN. In order to solve the problem of feature duplication calculation, SPP Net first extracts the feature map of the entire image through CNN, and then maps the candidate boxes onto the feature map to obtain the corresponding feature maps for each candidate box. In order to overcome the drawback of fixed input image size, SPP Net utilizes Spatial Pyramid Pooling (SPP) layers to pool features, converting feature maps of different sizes output by the convolutional layer into fixed sizes, and then inputting them into a fully connected layer. The proposal of the SPP layer enables the network to support both image inputs of any size and multi-scale training. Compared with R-CNN, SPP-Net has improved detection speed, but model training is still carried out in stages, resulting in lower efficiency.

In 2015, R. Girshick et al. proposed Fast R-CNN based on R-CNN and SPP Net [30] . This model simplifies the SPP layer by using a Region of Interest Pooling (RoI Pooling) layer to obtain candidate region feature maps of specific sizes, which are then input into a fully connected layer for classification and bounding box regression. Fast R-CNN combines classification loss with bounding box regression loss for training, and uses Softmax classifier instead of SVM classifier.

The above algorithms all use Selective Search (SS) to obtain candidate regions, which is relatively time-consuming. Moreover, when generating feature maps for candidate regions, the features extracted by the feature extraction network cannot be shared, which wastes computing resources. Subsequently, R. Girshick et al. addressed the above issues in their proposed Faster R-CNN [31] . The author proposes using Region Proposal Network (RPN) combined with Anchor mechanism to generate candidate boxes, which improves the model speed. Moreover, the features extracted through the backbone network are input into both the subsequent network layers and the RPN network, fully achieving convolutional feature sharing. Faster R-CNN is the first algorithm that is closest to real-time object detection, greatly promoting the development of the field of object detection. However, this model only predicts on the last layer of feature maps, which is not effective for detecting small objects with limited information.

After Faster R-CNN, researchers mainly focused on improving the classification and localization capabilities of the model and proposed a series of optimization algorithms. In 2016, Dai et al. proposed R-FCN [32] , which is sensitive to changes in object position. This network introduces a Full Convolutional Network (FCN) [33] in Faster R-CNN to achieve convolutional operation sharing, in order to reduce computational complexity and improve model speed. In 2017, He et al. combined Faster R-CNN with FCN and proposed Mask R-CNN [34] algorithm for pixel level segmentation, which integrates classification, regression, and segmentation. The region of interest alignment layer (RoI Align) proposed in Mask R-CNN solves the pixel bias problem caused by two quantization operations in the RoI Pooling layer. In order to solve the problem of model overfitting and IoU mismatch during inference, Cai et al. proposed Cascade R-CNN in 2018 [35] . The network adopts a cascaded R-CNN structure to optimize the detector. The detection model of the network is trained in stages on positive and negative samples with different IoU thresholds, that is, the detector in each stage is trained based on the output of the previous stage.

（2）One stage object detection algorithm

One stage algorithms are more efficient than two-stage algorithms and can meet real-time requirements, but their accuracy is not as good as two-stage algorithms. Joseph Redmon et al. proposed YOLO (You Only Look Once) in 2016 [36] , which treats detection problems as regression problems, pioneering the first stage of object detection. Although YOLO runs at a fast speed, its recall rate is low and its performance in detecting small targets is not good. The following year, Joseph Redmon et al. introduced YOLO9000 [37], which, as the name suggests, can detect over 9000 different types of targets. YOLO9000 has made many improvements at the level of details, mainly including the introduction of Batch Normalization (BN) to accelerate model convergence speed, training image classifiers with high-resolution samples, proposing a new basic network DarkNet-19, introducing Anchor idea, K-means clustering to extract prior boxes, multi-scale training, and using passthrough layers to fuse feature maps of different scales. After a series of improvements, YOLO9000 has significantly improved detection speed and accuracy compared to YOLO. In 2018, Joseph Redmon et al. launched YOLOv3 [38]. This network utilizes a deeper DarkNet-53 to extract image features, and the model can detect on feature maps at multiple scales, improving the performance of small object detection. Moreover, multiple independent logistic classifiers are used instead of Softmax classifiers for multi label classification, and binary cross entropy (BCE) loss functions are used to calculate classification errors and confidence errors. In 2020, YOLOv4 [39] and YOLOv5 emerged successively. YOLOv4 replaces the backbone network with CSPParkNet53; The Neck section introduces SPP (Spatial Pyramid Pooling) [29] to expand the receptive field, and adds a PAN (Path Aggregation Network) [41] module on top of YOLOv3's FPN (Feature Pyramid Networks) [40] module to better fuse feature map information at different scales. In addition, YOLOv4 also integrates numerous optimization strategies, greatly improving detection speed while ensuring detection accuracy. Subsequently, YOLOv5 made a series of improvements to YOLOv4, launching multiple versions of the YOLOv5 model based on the YOLOv5l model. Compared to YOLOv4, YOLOv5 takes up less memory, is easy to deploy, and has a faster speed. At present, the YOLO series

algorithms are still being developed. In addition to the YOLO series algorithms, there are also some excellent one-stage object detection algorithms that have been proposed one after another.

Liu et al. proposed the Single Shot Multibox Detector (SSD) [42] , which combines the advantages of Faster R-CNN and YOLO, with high detection accuracy and fast speed. SSD draws inspiration from the design concept of Anchor in Faster R-CNN, setting anchor boxes of different sizes and aspect ratios for each cell in the feature map, and then fine-tuning the anchor box position based on the predicted offset to obtain more accurate detection results. In addition, SSD also integrates multi-scale feature maps, and the model can detect targets of different sizes on feature maps of different scales. The author of RetinaNet [43] pointed out that the main reason for the lower accuracy of one-stage object detection algorithms compared to two-stage algorithms is the imbalanced sample categories. So the author proposed the Focal Loss function to reduce the proportion of easily classified negative samples in the loss, thus balancing the problem of balanced positive and negative samples and difficult to classify samples. In 2019, Tan et al. proposed an efficient EfficientNet [44], which utilizes a composite model expansion method to scale the model from three dimensions: network depth, width, and input image resolution, effectively improving model speed and detection accuracy. Subsequently, Tan et al. proposed EfficientDet [45] based on EfficientNet, which utilizes the Weighted Bi directional Feature Pyramid Network (BiFPN) to quickly fuse multi-scale features and optimize the model scaling method in EfficientNet, further improving model performance.

The two-stage object detection algorithm and the one-stage object detection algorithm each have their own advantages. With the continuous deepening of research, researchers gradually integrate some excellent ideas of the two-stage algorithm on the basis of the one-stage algorithm, which can further improve detection accuracy while ensuring model speed.

1.2.3 Small object detection method

There are two ways to define small targets: relative scale and absolute scale. The relative scale is defined based on the proportion of target pixels in the image. According to the SPIE organization, a small target is defined as a target pixel that accounts for less than 0.12% of the total pixels in the image. In addition, it can be defined based on the ratio of the width to height of the target border to the width to height of the image. Typically, targets with a width to height ratio less than 0.1 are considered small targets. The absolute scale is defined based on the absolute pixels of the target and varies on different datasets. The COCO dataset defines targets with a resolution of less than 322 as small targets. The TinyPerson dataset defines targets between 20 and 32 pixels as small targets, and targets between 2 and 20 pixels as small targets.

There are many difficulties in detecting small targets, which leads to many algorithms having much lower detection performance than large and medium-sized targets. The main reasons that affect algorithm performance are as follows: Firstly, low resolution and insufficient information of small targets result in fewer effective features extracted by neural networks. Secondly, small targets occupy a relatively small area in the image and are susceptible to background interference, which requires high localization performance of the algorithm. Thirdly, it is difficult to annotate small objects and the training data is limited, resulting in poor generalization ability of the model. Researchers have proposed various solutions to the characteristics of small target datasets, including multi-scale learning, anchor free mechanisms, and generative adversarial learning. Research has shown that these methods can improve the detection performance of small targets to a certain extent.

Multi scale learning is a commonly used method to improve the small object detection performance of algorithms. In 2016, Tsinghua University proposed the HyperNet algorithm [46], which combines shallow features with high-level semantic features to improve the accuracy of target localization and provide research ideas for small object detection. In 2017, Lin et al. proposed Feature Pyramid Networks (FPN) [40]. FPN is an efficient feature extractor that can fuse extracted feature information

of different scales, construct feature pyramid structures of different sizes, and then predict each layer of feature map separately. This method of upsampling low resolution deep feature maps and then fusing shallow feature maps can improve the problem of insufficient small target information in deep features.

The anchor free mechanism refers to a method that directly converts object detection tasks into keypoint estimation without the need to preset prior boxes on the image. It mainly includes two methods: corner detection based and center point detection based. Due to the design of anchor boxes, algorithms are prone to ignoring small targets, so anchor free mechanism has become an important means of studying small target detection. In 2018, Law et al. proposed the CornerNet algorithm based on corner detection [47]. This algorithm predicts the upper left and lower right corners of each target through a heatmap, and then matches the upper left and lower right corners of the same target based on the embedding vector of the detected corners. Finally, the position of the corners is adjusted by offset to obtain the target bounding box. CornerNet is prone to ignoring the internal information of the target. To improve this problem, Duan et al. proposed the CenterNet algorithm [48]. The algorithm first predicts the positions of the top left corner, bottom right corner, and center point of the target, then performs corner matching to determine the bounding box, and finally uses the center key point to eliminate erroneous bounding boxes. Subsequently, Zhou et al. [49] proposed a method of modeling the target as a point for prediction, which utilizes keypoint estimation to obtain the center point of each target, and then regresses the size and direction attributes of the target at the center point.

Goodfellow et al. first proposed Generative Adversarial Networks (GANs) [50]. The generator of GAN can generate image data using random noise, while the discriminator is responsible for distinguishing between generated data and real data. The generator and discriminator mutually promote and improve during the game process, and after multiple iterations, they finally reach a balanced state. Due to its ability to generate data that is highly similar to real data, GAN can alleviate the false detection phenomenon caused by a large number of similar features in small target

images. When applied to small target detection tasks, it has achieved initial success. Subsequently, Li et al. proposed a Perceptual Generative Adversarial Network (Perceptual GAN) specifically designed for small object detection [51]. The generator of this network maps small target features to features similar to large targets, enhancing the feature expression ability of small targets. The discriminator and generator use adversarial learning to distinguish between the representations generated by the generator and the original large target features, and propose that the generated representations must be conducive to the perception requirement of small target detection. In 2018, Bai et al. proposed an end-to-end Multi Task Generative Adversarial Network (MTGAN). This network can input low resolution images into a generator, generate super-resolution images, and then input them into a multitask discriminator to simultaneously distinguish between real images and generated super-resolution images. Due to its ability to restore fuzzy small targets to clear ones, MTGAN can better extract small target features and improve the algorithm's small target detection performance.

1.3 Research contents

In small target images, objects are densely arranged, target shapes are diverse, and there is a lot of noise, which leads to poor detection performance of many algorithms for small targets. In response to this phenomenon, after comparing the performance of many classic object detection algorithms, we ultimately chose YOLOv5s, a lightweight version of YOLOv5, as the benchmark model. By optimizing its network structure, we improved the model's small object detection ability. And the improved model will be validated for performance on the TinyPerson dataset and the VisDrone2019 dataset. The main research content is as follows:

(1) In order to improve the serious problem of missed detection of small targets, P2 detection heads have been added to enable the network to detect smaller targets. The YOLOv5s-P2 model integrates a 4x downsampling feature map with rich location information, which can transfer more shallow features to deep features,

which is beneficial for enhancing the expression ability of the multi-scale feature map extracted by the network and enabling better multi-scale object detection.

(2) In order to suppress the interference of background and noise factors and improve the attention of small targets, multiple CBAM modules have been added to the Neck structure of YOLOv5s. The introduction of CBAM attention mechanism enables the Neck network to suppress the interference of irrelevant information and focus more on the key features of the target to be detected when processing a large amount of feature information, which is beneficial for enhancing the feature information of small targets and improving the accuracy of small target detection.

(3) In order to better perform feature fusion and enhance the network's feature processing ability, the feature fusion structure of YOLOv5s was optimized by drawing on the idea of bi-directional cross scale connections in BiFPN networks. The improved network performs bidirectional cross scale fusion between shallow and deep features, which enhances the connection between feature maps and makes the detected feature maps contain more information, especially low-level geometric information. It helps the network to better focus on the area where small targets are located and improve the recall rate of small targets.

2 RELATED KNOWLEDGE

2.1 Overview of Deep Learning

With the proposal of AlexNet in 2012, deep neural networks gradually emerged in various fields, and deep learning also ushered in an explosive period. Thanks to the powerful learning ability of deep neural networks, they have made significant breakthroughs in image processing and speech technology, greatly promoting the development of medical image analysis, image retrieval, recommendation systems, autonomous driving, machine translation, speech recognition and other fields. Nowadays, deep learning algorithms have become the mainstream technology for implementing machine learning.

Tracing the development history of neural networks, they were originally derived from the Artificial Neural Network (ANN). ANN is inspired by biological neurons and is used to simulate human behavioral characteristics, enabling machines to have information analysis and processing capabilities like humans. As shown in Figure 2.1, it is an artificial neuron model. Among them, xi is the input signal, wi is the weight, and f is the activation function. The current input of the neuron is $\sum_{i=}^{n}1$ wi xi + b. After the signal is processed by the neuron, the output value is$f(\sum_{i=}^{n}1$ wi xi + b).

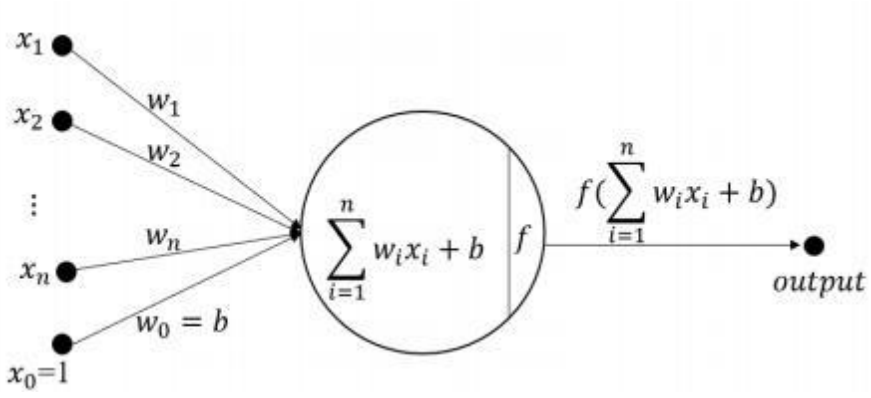


Figure 2.1 Artificial neuron model

In the biological nervous system, there are countless neuronal cells, and organisms rely on the communication between these neurons to perceive external

information. The same applies to existing artificial neural networks, which are connected by multiple artificial neurons and add multiple hidden layers between the input and output layers. As shown in Figure 2.2, it is a typical multi-layer feedforward neural network with a connection principle similar to the synaptic structure of biological neurons, which can transmit information to surrounding neurons. The result of processing in one layer of artificial neurons will serve as input for the next layer, and the processing of data varies among neurons in different layers. In order to change the strength of signals to be transmitted between neurons in different layers, different weights are usually designed for some parameters. These weight sizes will be continuously adjusted according to the training situation of the network, minimizing the error between predicted values and true values to achieve optimal network performance. The currently popular weight update method is the BP backpropagation algorithm, which includes two stages: forward propagation and backward propagation.

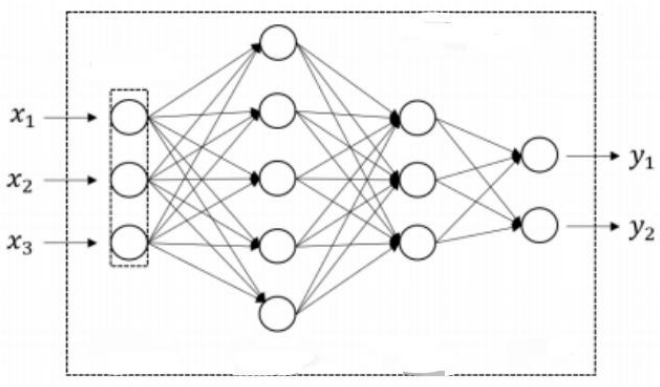


Figure 2.2 Multilayer feedforward neural network

Forward propagation refers to the transmission of data from the input layer to the hidden layer, and a series of learning and optimization operations are carried out in multiple hidden layers. After layer by layer processing, the final result is output through the output layer. If the final result is significantly different from the expected value, the loss function calculates the error between the output value and the expected value, and then enters the error backpropagation stage.

In the backpropagation process, an optimizer is usually used to update the parameters of the loss function, making the error infinitely close to the minimum value. When the error cannot be further reduced, the network learning process ends. Common optimizers include the following categories:

(1)traditional gradient descent methods, including stochastic gradient descent (SGD) [53], small batch gradient descent (MBGD), etc;

(2) Momentum descent method [54];

(3) Adaptive learning rate optimization algorithms, including AdaGrad [55], RMSProp, and Adam [56]. SGD and Adam are commonly used in these algorithms. The advantage of the SGD algorithm lies in its fast training speed even when there is a large amount of data, as it does not require traversing the entire dataset and only randomly selects a certain number of samples to update the model parameters each time. The disadvantage is that the algorithm is unstable, and each updated parameter is only related to the gradient of a small batch of samples, and it may not necessarily ensure that it is updated in the correct direction every time. The Adam algorithm combines the advantages of first-order momentum and second-order momentum, and can effectively control the gradient direction. It is also very robust in selecting hyperparameters and can dynamically update the learning rate for each parameter, greatly improving training speed.

With the continuous deepening of research on artificial neural networks, the number of layers in the network continues to deepen, gradually forming a deep neural network composed of multiple hidden layers, and machine learning based on deep neural networks is called deep learning. The core of deep learning is to repeatedly train a large amount of computational data on neural network models, and these image data will be transformed into more abstract feature expressions after undergoing a large amount of nonlinear operations. Due to the ability of deep neural networks to extract a large number of advanced semantic features from images, deep learning systems autonomously learn these key features, thus possessing strong feature expression capabilities, fully compensating for the shortcomings of traditional

algorithms that require manual feature extraction, greatly improving the robustness and generalization ability of the model.

The concept of deep learning has attracted a large number of researchers since its inception, and researchers have continuously innovated and proposed many excellent algorithms in the development process of deep learning. In terms of whether or not to interfere with the model training process, it can be roughly divided into the following three categories: supervised learning, unsupervised learning, and semi supervised learning. Typical deep learning networks include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Deep Belief Networks (DBNs), Generative Adversarial Networks (GANs), and so on. Especially represented by CNN, neural networks have strong modeling capabilities and are one of the most widely used networks for deep learning algorithms. In the past few years, CNN has made breakthrough progress in fields such as computer vision and played a crucial role in the long history of neural network development.

2.2 Convolutional neural network

As shown in Figure 2.3, it is a typical convolutional neural network structure diagram, mainly composed of input layer, convolutional layer, pooling layer, fully connected layer, and output layer. This section will provide a detailed introduction to the structure of CNN.

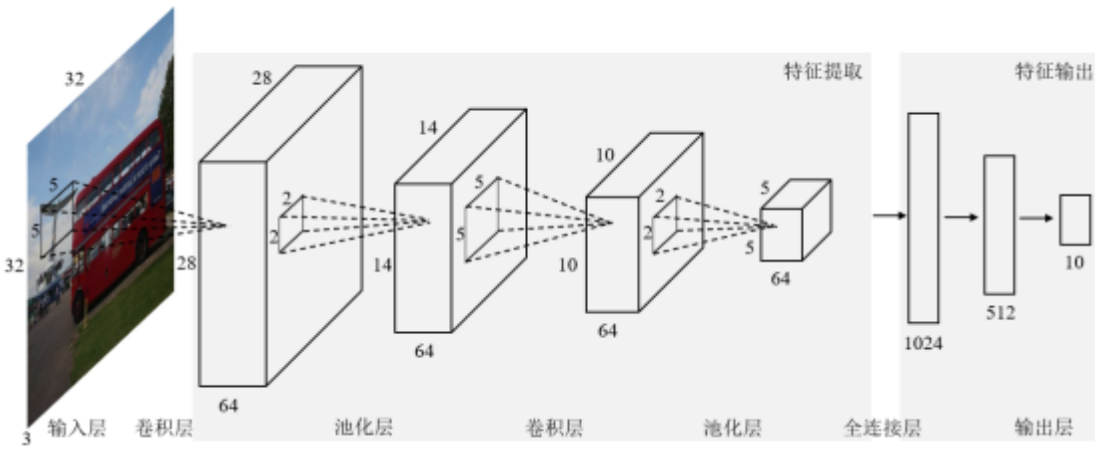


Figure 2.3 Convolutional Neural Network Structure

2.2.1 Convolutional Layer

Each convolution layer is composed of multiple filters, and each filter contains many convolution kernels inside. In CNN, image features are extracted through convolution operations, and each filter is convolved to obtain a feature map. Due to the different image features that each filter focuses on, it is necessary to use multiple filters to obtain more image feature information. When performing convolution operations, convolutional layers closer to the input end extract shallow features of the image. The resolution of the feature map is higher, the local receptive field is smaller, and it contains more detailed information such as texture, color, and edges. Convolutional layers closer to the output end can extract deep features of the image. The local receptive field of deep feature maps is larger, containing more advanced semantic information, but lacking positional information.

When the size of the convolution kernel increases, the local receptive field of the output feature map after convolution operation becomes larger, and it can pay more attention to the overall information of the image. However, a large convolution kernel increases computational complexity and reduces network performance, so the size of the convolution kernel is not necessarily better. Usually through stacking 3 × 3 and 5 × A small convolution kernel of 5 is used to extract finer features from the image, and the small convolution kernel adds more nonlinear transformations. As shown in Figure 2.4, taking two-dimensional convolution as an example, the convolution calculation process is introduced. Where input represents the input image; The kernel is a 3 × The feature extraction effect of a 3-size convolution kernel depends on the changes in parameters in the kernel; Output is a feature map generated after convolution operation, representing the local features extracted from the original matrix. The weighted convolution kernel slides on the image in order from left to right and from top to bottom, and performs multiplication and addition operations on the overlapping pixel matrices one by one. When all regions of the image have been slid, a new feature map can be generated.

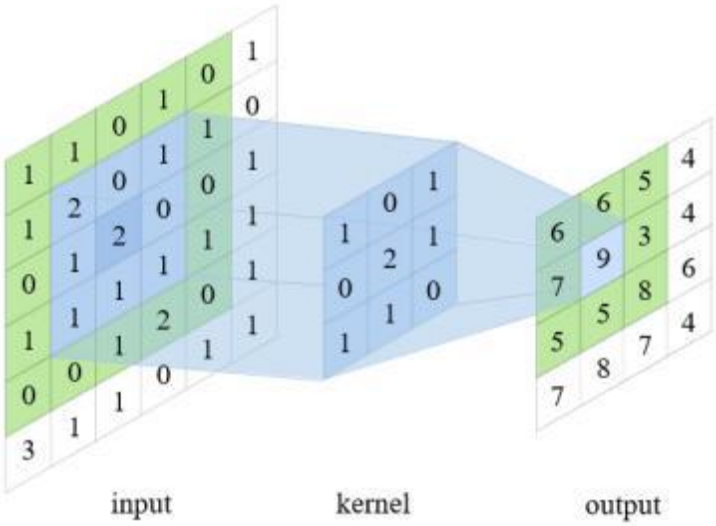


Figure 2.4 Schematic diagram of convolution calculation

2.2.2 Pooling layer

Usually, a periodic pooling layer is added after the convolutional layer of a convolutional neural network to filter the features within a filtering interval, extract the most representative features, and reduce the number of parameters input to the next network layer. Therefore, adding a pooling layer can to some extent prevent overfitting from occurring.

Pooling operations can reduce feature dimensions, essentially downsampling images. The common pooling methods include Max Pooling and Average Pooling, with the most commonly used being the max pooling operation. Maximum pooling refers to taking the maximum value within a filtering interval, which can suppress some noise interference and retain more edge and texture information in the image. Average pooling is the process of summing up the pixels within a sliding window and taking the average value. It can improve the robustness of the model and tends to preserve the background information of the image. As shown in Figure 2.5, there are schematic diagrams of maximum pooling operation and average pooling operation, respectively.

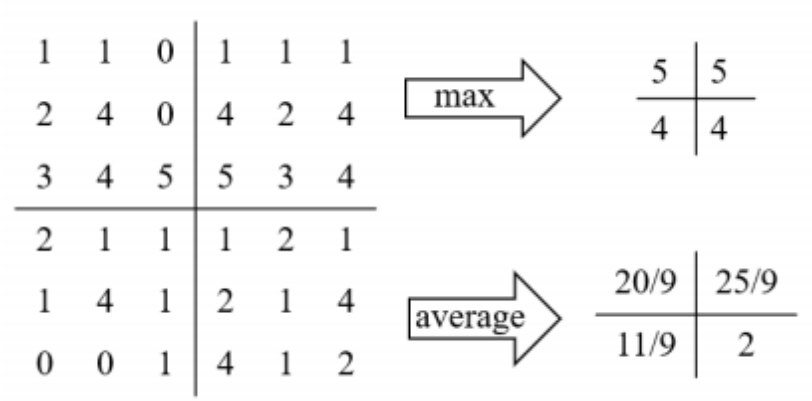


Figure 2.5 Schematic diagram of pooling operation

2.2.3 Fully connected layer

The Fully Connected (FC) layer is located in the later layers of the convolutional neural network, which integrates the local features extracted by the previous network layers by mapping high-dimensional feature maps into one-dimensional feature vectors. Fully connected neural networks generally include at least an input layer, a hidden layer, and an output layer, mainly used to achieve classification purposes. As shown in Figure 2.6, it is a fully connected neural network structure diagram.

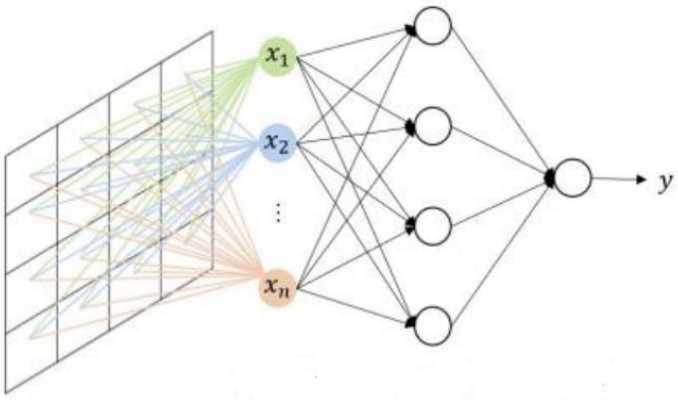


Figure 2.6 Fully Connected Neural Network Structure

As shown in Figure 2.6, each neuron in the fully connected layer is connected to all neurons in the previous layer, so the parameter count of the fully connected

layer is usually the highest. In practical applications, in order to reduce parameter redundancy, the extracted features are usually first subjected to Global Average Pooling (GAP) operation, and then input into fully connected layers, or convolutional layers are directly used instead of fully connected layers.

2.2.4 Activation function

Activation functions are mostly nonlinear, used to map the input signal of neurons to the output end. Due to the fact that convolution and pooling operations are linear operations, no matter how many layers the deep neural network is stacked, it is essentially a linear model. In order to improve the nonlinear expression ability of the network and accelerate the convergence speed of the model, it is necessary to introduce a nonlinear activation function in the hidden layer. The common activation functions are as follows: Sigmoid function, Tanh function, Softmax function, ReLU function [57], Leaky ReLU function [58], PReLU function, etc. The function graph is shown in Figure 2.7.

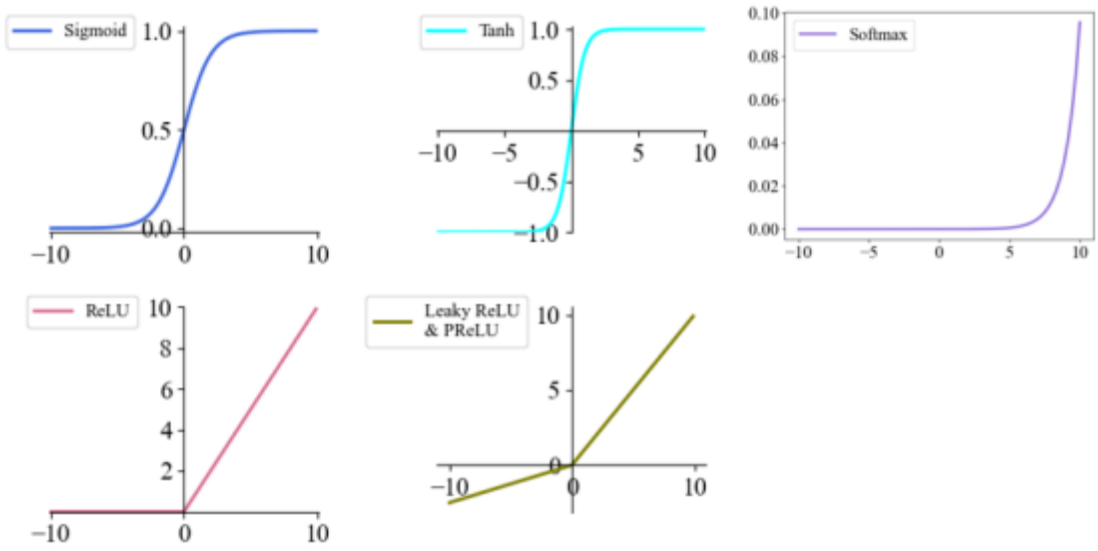


Figure 2.7 Activation function image

In binary classification problems, the Sigmoid function is generally used for the output layer, and the Tanh function is used for the hidden layer. The expressions for both are shown in formulas 2.1 and 2.2.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

The Softmax function is used in the output layer of multi classification tasks, often combined with the Cross entropy (CE) loss function. The Softmax function can normalize the data output from the previous network layer, mapping the values to the (0,1) interval, and the sum of all items is 1. The probability distribution of each category obtained is the prediction result of the multi classification task. The expression is shown in formula 2.3, where zi represents the output value of the i-th node, and C represents the total number of nodes.

$$Soft\max(z_i) = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}} \tag{2.3}$$

The ReLU function solves the problem of vanishing gradients and has a fast calculation speed, making it one of the most widely used activation functions. The expression is shown in formula 2.4.

$$f(x) = \max(0, x) \tag{2.4}$$

Although the convergence speed of ReLu function is fast, there is a phenomenon of neuronal necrosis. If the values of the input activation function are all negative, the corresponding weights and bias parameters cannot be updated, resulting in the neuron being unable to continue learning. The Leaky ReLU function and PReLU function solve the problem of neuronal necrosis in the ReLU function, and their calculation methods are shown in formula 2.5. The difference is that a in Leaky ReLU is a fixed value, while a in PReLU is variable and learned through backpropagation.

$$f(x) = \max(ax, x) \qquad (2.5)$$

2.2.5 Loss function

The loss function is used to calculate the deviation between the predicted value and the target value of the model. The smaller the error, the better the robustness of the model. The purpose of neural network training is to learn a set of parameters to make the predicted values infinitely approximate the true values. To achieve this goal, it is necessary to select a suitable loss function to calculate the difference, provide input data for backpropagation, update model parameters, and optimize model performance.

The loss function can be roughly divided into classification loss and regression loss. The classification loss function is mainly used to handle discrete variables, and the commonly used one is the cross entropy (CE) loss. The regression loss function is aimed at continuous variables, commonly including Mean Square Error (MSE), Smooth L1 Loss, etc. Below are two types of loss functions.

(1)Cross entropy loss function

The binary cross entropy (BCE) loss function is shown in formula 2.6, where y represents the true category with a value of 0 or 1; P represents the probability of being predicted as a positive sample, with a value within the (0,1) interval, and p is usually processed by the sigmoid activation function.

$$Loss_{BCE} = -[y \log(p) + (1-y)\log(1-p)] \qquad (2.6)$$

The multi class cross entropy loss function is shown in formula 2.7. Where i and M represent the number of samples and categories, respectively; Yic represents whether the true category of sample i is c, with a value of 0 or 1; P_{ic} represents the probability that sample i is predicted as class c, and P_{ic} is usually obtained by Softmax processing.

$$Loss_{CE} = -\sum_{i}\sum_{c=1}^{M} y_{ic}\log(p_{ic}) \tag{2.7}$$

(2) mean squared error

Mean squared error refers to the sum of squares of the difference between the predicted value and the target value, used to measure the average error. The smaller the calculation result, the more accurate the prediction is. The calculation method is shown in formula 2.8.

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(p_i - y_i)^2 \tag{2.8}$$

2.3 Object detection algorithm

After entering the era of deep learning, object detection algorithms have sprung up like mushrooms after rain, which can be roughly divided into two-stage algorithms represented by R-CNN series algorithms and one-stage algorithms represented by YOLO series algorithms. The two-stage object detection algorithm is completed in two steps. Firstly, candidate regions are generated, and then the candidate boxes are classified and position corrected. Although the accuracy of algorithm recognition is high and the missed detection rate is low, the speed is slow and cannot meet the needs of real-time detection. In order to compensate for the shortcomings of such algorithms, another type of one-stage object detection algorithm has emerged. The first stage object detection algorithm no longer selects candidate regions separately, but directly outputs object categories and positions. Although it improves detection speed, the accuracy is not as good as two-stage algorithms.

R-CNN is the first algorithm to use CNN for object detection, ushering in the era of deep learning based object detection. YOLO is the opening work of one-

stage object detection algorithms, and many researchers have been influenced by YOLO's ideas to propose a series of high-performance object detection algorithms. The R-CNN and YOLO series algorithms have greatly promoted the development of object detection technology, and this section will focus on introducing these two types of algorithms.

2.3.1 R-CNN series algorithms

The main differences of the R-CNN series algorithms are introduced. In R-CNN [28], selective search (SS) is used to obtain candidate boxes, and the candidate regions are cropped and scaled. The candidate region acquisition, feature extraction, classifier, and regressor of this model all work independently. Fast R-CNN [30] integrates feature extraction networks, classifiers, and regressors together. Faster R-CNN [31] integrates all the above components into one network, enabling end-to-end training of the network.

Faster R-CNN is a classic two-stage object detection algorithm, and this section will focus on introducing this model. It mainly consists of feature extraction network, region generation network (RPN), region of interest pooling layer (RoI Pooling), and classification regression layer. Figure 2.9 shows the fast_ RCNN_ The network structure of test.pt.

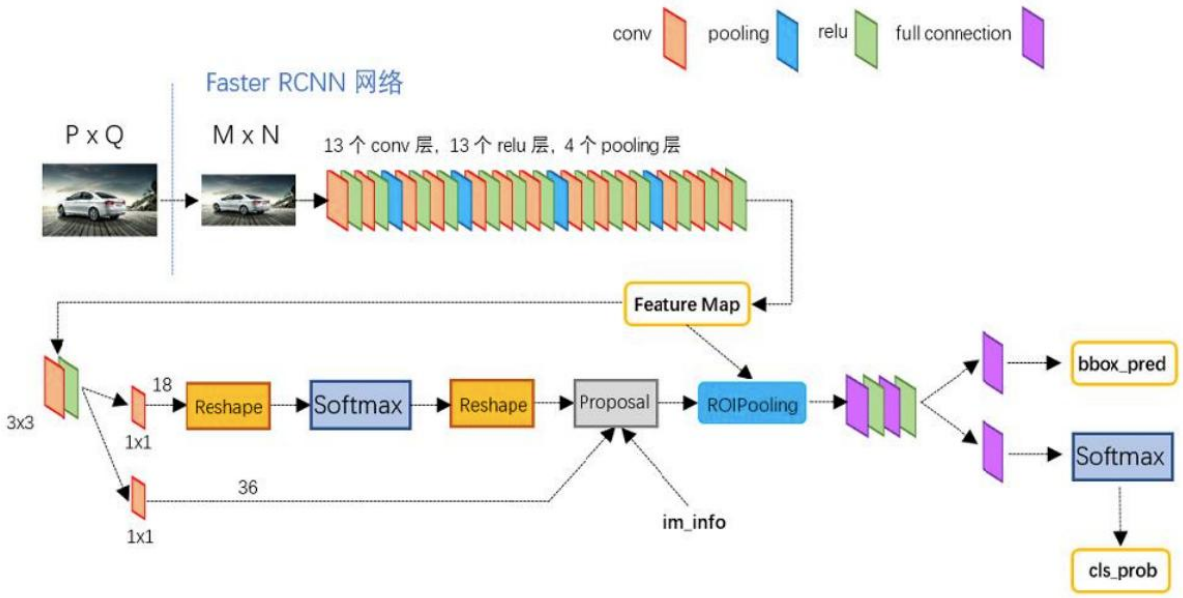


Figure 2.9 Faster R-CNN Network Structure

As shown in Figure 2.9, the Faster R-CNN algorithm process mainly consists of the following steps: first, scale the image to a certain size, and then input it into a deep neural network to obtain a convolutional feature map; The second step is to input this feature map into the RPN module to generate a series of candidate region positions; Step three, input the candidate regions output by the RPN network and the feature maps output by the feature extraction network into the RoI Pooling layer to obtain a fixed size feature map of the region of interest; Step 4, classify the candidate boxes and perform border regression. Below is a detailed introduction to the structural composition of each part of the model.

(1) Feature extraction network

In Faster R-CNN, the ZF model [59] and VGG-16 [13] are used to extract image features. The ZF model is relatively shallow with only 5 shared convolutional layers, while VGG-16 is relatively deep with 13 shared convolutional layers. Figure 2.9 shows the network constructed with VGG-16 as the backbone network.

(2) RPN network

R-CNN, SPP Net [29], and Fast R-CNN all use selective search on the original input image to generate candidate regions, which is relatively time-consuming. Faster R-CNN generates the positions of candidate regions on shared convolutional feature maps through an RPN module, greatly improving the speed of candidate box generation, thus establishing the important position of Faster R-CNN in the field of object detection.

The Anchor mechanism is the core of the RPN network, where each pixel on the feature map corresponds to 9 candidate boxes, which are calculated from anchor boxes of different scales and aspect ratios. As shown in Figure 2.10, it is a schematic diagram of the generated anchor box. The RPN network defaults to using anchor boxes with scales of 1282, 2562, and 5122, and each scale corresponds to three different ratios of 1:1, 1:2, and 2:1, respectively. Therefore, a total of nine different anchor boxes are generated, which can basically cover targets of different shapes and scales in the dataset. Specifically, when using the MS COCO dataset, an additional

anchor box with a scale of 642 was added due to the large number of small targets. So setting the anchor box scale reasonably can help improve the detection performance of the algorithm.

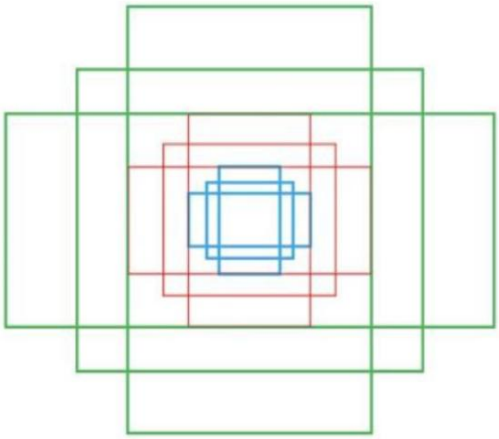


Figure 2.10 Anchor diagram

After generating candidate boxes for each pixel on the feature map, it is necessary to further determine whether each candidate area is the target area, and if it is the target area, then determine its border position. As shown in Figure 2.11, the operational mechanism of the RPN network is demonstrated. Due to the fact that the feature map of the input RPN network has multiple channels, it is necessary to use 3 × The sliding window of 3 performs convolution operations on pixels at the same position in each channel. Each sliding window area is mapped into a low dimensional feature through an intermediate layer, and the 9 candidate boxes corresponding to this pixel share this low dimensional feature. Then pass this low dimensional feature to the classification layer to determine whether the candidate region is foreground or background. If it is the foreground, enter the regression layer to position the candidate boxes.

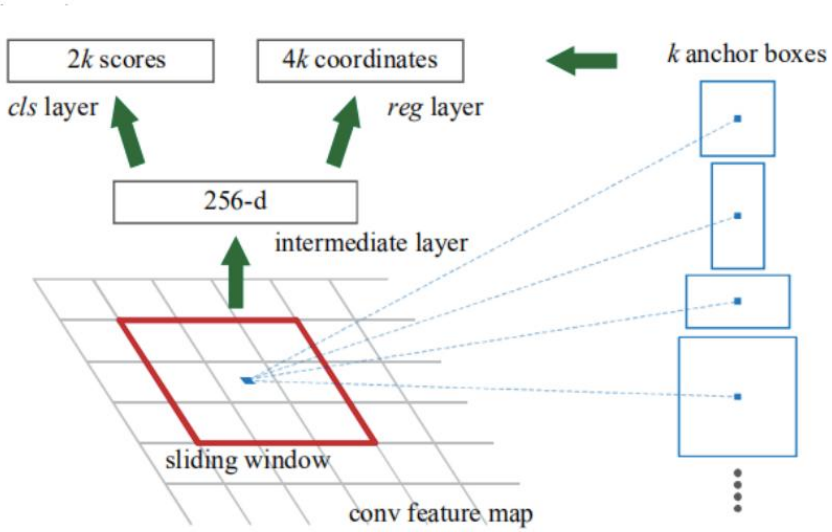


Figure 2 11 RPN Network Operation Mechanism[31]

Using an RPN network will generate many candidate regions, some of which will highly overlap. To reduce redundancy, the Non Maximum Suppression (NMS) algorithm is used for filtering. Finally, sort the candidate regions according to their scores and input them into the next network layer for further processing.

(3) RoI Pooling layer

Due to the fact that RPN networks are candidate regions generated based on the Anchor mechanism and have multiple scales, they cannot be directly inputted into fully connected layers. In order to solve the problem of scale inconsistency, the RoI Pooling layer of Fast R-CNN is adopted in Faster R-CNN, which is used to convert the input multi-scale feature map into a fixed size before outputting. This method was originally inspired by the Spatial Pyramid Pooling layer (SPP) of SPP-Net. The RoI Pooling layer in Faster R-CNN combines the candidate boxes output by the RPN network with the feature maps output by the backbone network to obtain the feature maps of the candidate regions, which are then input into the fully connected layer for subsequent operations.

(4) Classification regression layer

Candidate box classification and regression are the final steps of Faster R-CNN. Use Softmax classifier to obtain the category of candidate boxes, and use bounding box regression to obtain the offset of each candidate box relative to its true position,

which is used to correct the position of the candidate boxes and make the detection results more accurate.

2.3.2 YOLOv1 algorithm

Although the introduction of Fast R-CNN into RPN networks has improved algorithm performance, the step of generating candidate regions is still time-consuming and requires repeated training of both RPN and Fast R-CNN networks until the emergence of YOLO [36] broke through the speed bottleneck of object detection algorithms. YOLO transforms detection tasks into regression problem processing, truly achieving end-to-end detection with very fast speed, which can be used for real-time detection tasks such as video surveillance. Below is a detailed introduction to the network structure, basic principles, and loss function composition of the YOLO algorithm.

(1) Network structure

The network structure of YOLO is relatively simple, first extracting image features through convolutional and pooling layers, and then using fully connected layers to predict object class probabilities and regress bounding box positions. Due to the addition of two fully connected layers at the end of the network, the size of the input image must be fixed. Add 448 × A 448 sized RGB image input network, after a series of convolution and pooling operations, yields 7 × seven × A tensor of 1024 is inputted into the fully connected layer for linear regression, and finally output 7 × seven × A feature matrix of size 30.

(2) Basic Principles

The basic principle of YOLO algorithm is as follows: as shown in Figure 2.12, the input image is divided into S × S grids, each responsible for detecting objects falling into the center, the network predicts the target bounding box, confidence level, and object category probability contained in all grids at once. In this S × In S grids, each grid generates B bounding boxes (Bboxes), each containing 4 positional offsets (x, y, w, h) and 1 confidence score. These four position coordinates are the center point coordinates of the bounding box, as well as the width and height, which are normalized relative values. The confidence score is used to indicate the likelihood of

the bounding box containing the target. In addition, each grid also predicts probability scores for C categories.

In the PASCAL VOC dataset, YOLO divides the image into 7 × A grid of size 7, with a value of 2 for B, has a total of 20 categories in this dataset. Therefore, each grid predicts 30 values, and the network ultimately outputs 7 × seven × A tensor of 30 dimensions. The value of 30 is determined by (4+1) × Calculated by 2+20, it represents that each grid's two bounding boxes generate 4 positional offsets and 1 confidence score, respectively. At the same time, the grid also predicts probability values for 20 categories.

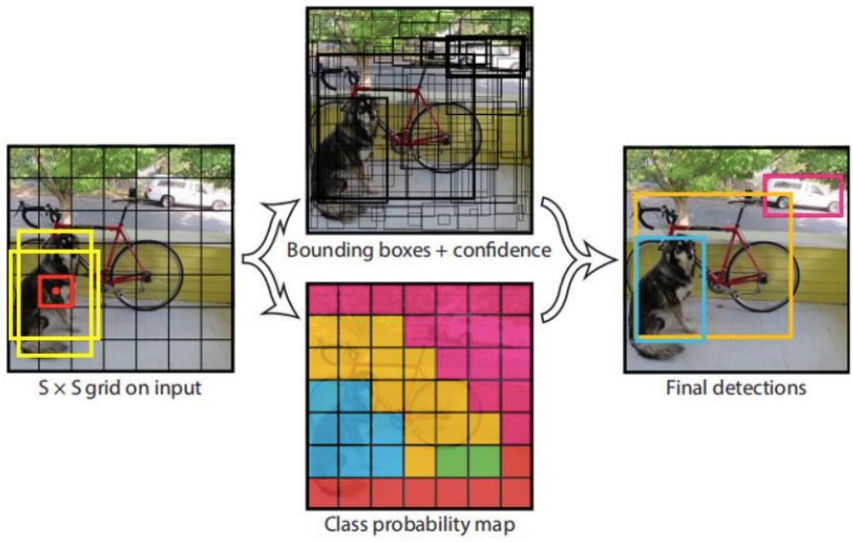


Figure 2 12 YOLO algorithm flowchart[36]

As shown in Figure 2.12, the detection process of YOLO algorithm can be roughly divided into the following steps: first, divide the input image into 7 × 7 cells, each generating 2 bounding boxes, resulting in a total of 98 bounding boxes. Then calculate the confidence level for each bounding box. If there are no objects inside the box, the confidence level is 0; If the target center is within the box, the confidence level is the IoU between the true box and the bounding box. If the confidence level is less than the set threshold, ignore this box and use the NMS algorithm to eliminate a large number of redundant prediction boxes. When a true bounding box overlaps with

multiple bounding boxes, only the box with the highest IoU value is taken. Finally, fine tune the corresponding offset of the bounding box and map it to the original image to obtain the final detection result.

Although YOLO has a fast detection speed, it has the following problems: each cell can only predict a maximum of one target, making it difficult to detect overlapping targets; There are only two types of prior boxes and detection is only performed on the top-level feature map, which results in poor detection performance for small targets; Cannot perform multi label classification, etc.

(3) Loss function

The loss function of YOLO consists of the weighted sum of position error, confidence error, and classification error. The following will introduce these loss functions separately. Formula 2.10 is used to calculate positional errors. In order to mitigate the impact of target scale differences, the root sign and variance function are introduced to calculate the loss of width and height. Where x, y, w, h represents the bounding box coordinates, S2 represents the total number of cells into which the image is divided, and B represents the number of bounding boxes generated by each cell,L_{ij}^{obj} represents the weight of the position error and whether there is a target in the jth bounding box generated by the i-th grid λ The value of coord is 5.

$$\begin{aligned} \mathrm{L}_1 &= \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} L_{ij}^{obj}\left[\left(x_i-\hat{x}_i\right)^2+\left(y_i-\hat{y}_i\right)^2\right] \\ &+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} L_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \end{aligned} \tag{2.10}$$

Formula 2.11 represents the confidence loss of bounding boxes with and without targets. Because there are many bounding boxes that do not include the center position of the target, it is necessary to reduce the weight of their confidence loss, so the value of λnoobj is set to 0.5.

$$L_2 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 \quad (2.11)$$

Formula 2.12 is used to calculate the classification error, where represents whether there is a target in the i-th grid. Finally, calculate the final Loss value using formula 2.13.

$$L_3 = \sum_{i=0}^{S^2} L_i^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2 \quad (2.12)$$

$$Loss = L_1 + L_2 + L_3 \quad (2.13)$$

2.3.3 YOLOv2 algorithm

YOLOv2 [37] has made many improvements in the details of network implementation, as shown in Table 2.1, demonstrating the improvement techniques adopted by YOLOv2 and its effects on the VOC2007 dataset.

Table 2 1 YOLOv2 Technology Improvement List

	YOLO								YOLOv2
Batch standardization		√	√	√	√	√	√	√	√
High resolution classifier			√	√	√	√	√	√	√
convolution				√	√	√	√	√	√
Anchor Thought				√	√				
DarkNet network					√	√	√	√	√
Cluster extraction prior box						√	√	√	√
position prediction						√	√	√	√
Passthrough fusion features							√	√	√
Multiscale training								√	√
High resolution detector									√
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

(1) Batch standardization (BN). Dropout usually appears in the fully connected layer of CNN, used to randomly discard a portion of neurons to prevent overfitting of the network. In YOLOv2, BN is used instead of Dropout to normalize the input data of each convolutional layer, in order to improve the learning ability of the model and accelerate its convergence speed. According to Table 2.1, the mAP of the model increased by 2.4% after adding BN operation. Nowadays, adding BN operations after convolutional layers has become a necessary processing method for building CNN models.

(2) High resolution image classifier. YOLOv1 uses a resolution size of 224 × Use image 224 to pre train the classification model, then transfer the classification model to the detection model for initializing the parameter weights of the detection model, and then use 448 × Train the detection model with an image size of 448. To reduce the impact of resolution changes during model migration, YOLOv2 first uses a resolution of 224 × Pre training the classification model with samples of 224, and then utilizing 448 × A sample size of 448 underwent 10 iterations of training to fine tune the classification model. After using a high-resolution classifier, the mAP of the model increased by 3.7%.

(3) Network structure. The basic network of YOLOv2 is DarkNet 19, located in two 3 × Added 1 between the convolutional layers of 3 × 1 convolution effectively reduces the number of feature map channels and reduces the number of model parameters. And also remove the fully connected layer in YOLOv1, using 1 × 1's convolutional layer and global average pooling (GAP) are used instead. After removing the fully connected layer, the network can receive input images with a resolution of 32 multiples and a maximum size of 608 × 608. After several iterations, the network randomly changes the size of the input image and performs multi-scale training.

(4) Cluster extraction prior box. YOLOv2 was inspired by Faster R-CNN and introduced the Anchor idea, which increased the number of prior boxes generated by the network and effectively improved the recall rate. In Faster R-CNN, the proportions of the 9 artificially set prior boxes are all conventional,

making it difficult to fully adapt to different datasets. YOLOv2 uses the K-means method to cluster the true labeled bounding box positions in the current training set. The prior box size selected through this method matches the true size of the sample more closely. After comprehensive evaluation, the value of K is selected as 5.

(5) Predict the position of the bounding box. YOLOv2 does not directly predict the offset, but instead predicts the offset of the center point of the bounding box relative to the upper left corner of the current grid, and normalizes this offset. In this way, the center point of the bounding box is constrained within a specific grid, which is beneficial for enhancing the stability of the model. By using clustering to extract prior boxes and a new bounding box position prediction method, Anchor Box improved YOLOv2's mAP value on the VOC2007 dataset by 4.8%.

(6) Use passthrough layers to fuse features. Due to the large receptive field of the feature map output by the last pooling layer, which is not conducive to small object detection, YOLOv2 proposes to use a passthrough layer to connect the high and low resolution feature maps to obtain fine-grained features, further improving the detection effect. As shown in Figure 2.13, the 26 output from the previous layer × twenty-six × Split the feature map of 512 into four 13 × thirteen × The feature map of 512 can be used to match 13 × thirteen × Splicing 1024 feature maps into a new 13 × thirteen × The feature map of 3072 is then convolved on the new feature map to obtain the final detection result.

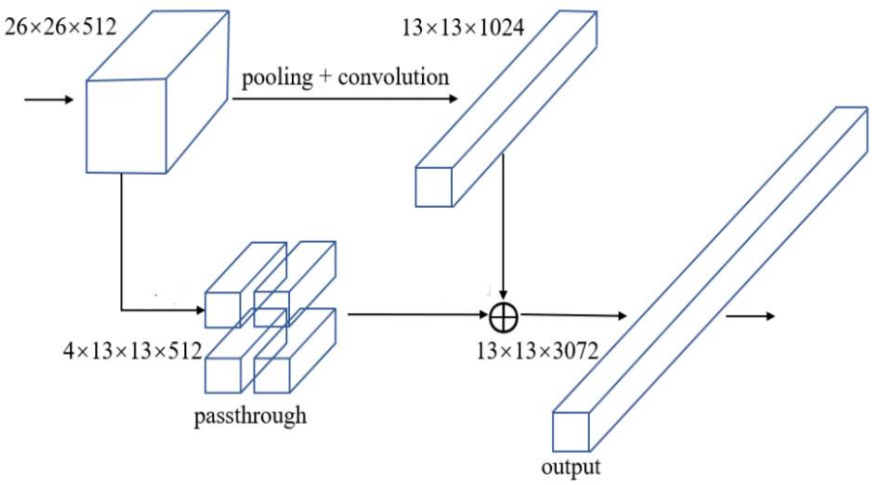


Figure 2 13 Implementation of Passthrough Layer

2.3.4 YOLOv3 algorithm

YOLOv3 [38] has made some improvements on the basis of YOLOv2, making the network more conducive to small object detection and further improving algorithm performance. It is now widely used in industrial production. The network structure of YOLOv3 is shown in Figure 2.14, and its improvement points can be summarized in the following three aspects: proposing a new backbone network DarkNet-53, using multi-scale prediction in the Neck network, and using logical classifiers for multi label prediction.

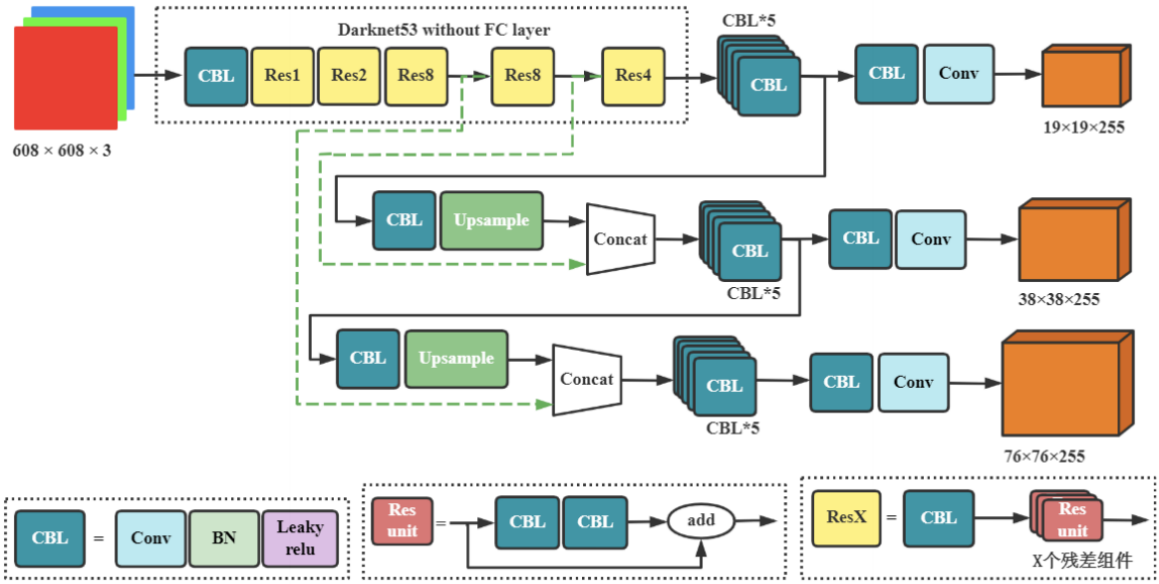


Figure 2.14 YOLOv3 Network Architecture

(1) DarkNet-53 Network

The network structure of DarkNet-53 is shown in Figure 2.14. It can be seen from the figure that YOLOv3 draws inspiration from the idea of residual blocks in ResNet [19], uses a large number of shortcut connections, and improves the network's feature extraction ability by stacking more network layers. Moreover, YOLOv3 is a fully convolutional network that eliminates the max pooling layer in YOLOv2 and implements downsampling operations through convolution operations with a stride of 2.

(2) Multiscale prediction

YOLOv3 designed feature maps of three scales to detect targets of different sizes, and set three different specifications of prior boxes for each feature map, resulting in a total of nine types of prior boxes being clustered. If the input image scale is 608 × 608, obtained 19 after downsampling at 32x, 16x, and 8x, respectively × 19. 38 × 38 and 76 × Feature map of 76. Among them, 19 × The 19 feature map has the largest receptive field and uses the largest prior box scale, making it suitable for detecting large targets; thirty-eight × The feature map of 38 takes second place, used for detecting medium-sized targets; seventy-six × The feature map of 76 has the smallest receptive field and is used for detecting small

targets. As shown in Figure 2.14, YOLOv3 performs detection on feature maps downsampled at 32x, 16x, and 8x, respectively. After the 79th layer, the network undergoes corresponding convolution operations to obtain a 32 fold downsampled feature map. Due to the large receptive field of this feature map, the captured image information is more comprehensive and can be directly used to detect large targets. Then, the 79th layer feature map is upsampled and fused with the 61st layer feature map to generate the 91st layer feature map. After a series of convolution operations, a 16x downsampled feature map is obtained, which is suitable for detecting medium-sized targets. Similarly, the upsampling of the 91st layer feature map is fused with the 36th layer feature map to obtain an 8-fold downsampling feature map for detecting small targets.

(3) Multi label prediction

YOLOv1 and YOLOv2 use Softmax classifiers and can only perform single label classification, meaning that a target can only belong to one category. In reality, a goal often belongs to multiple categories. In order to achieve multi label classification, YOLOv3 uses a logistic regression layer to perform binary classification on each category, mainly using the sigmoid function to constrain the input data within the (0,1) interval. If the result is greater than the set threshold, it indicates that the target belongs to a certain category.

2.3.5 YOLOv4 algorithm

YOLOv4 [39] has made improvements to both the Backbone and Neck structures of the network and adopted a series of optimization strategies. Compared with YOLOv3, it has greatly improved in performance. The main work of YOLOv4 is introduced below.

(1) Input end. The improvement of YOLOv4 input mainly involves proposing Mosaic data augmentation to expand the sample.

(2) Backbone network. Backbone adopts CSPDarknet53, which integrates the CSP module on the basis of Darknet-53 [38]. The CSP module draws inspiration from the design of Cross Stage Partial Network (CSPNet) [60]. In

addition, Mish activation function and DropBlock regularization method are also used in the backbone network.

(3)Neck network. The Neck section introduces the SPP module and FPN+PAN structure. YOLOv4 was inspired by the SPP Net [29] network and proposed the SPP module. The use of SPP can increase the receptive field of feature maps, which is beneficial for multi-scale object detection.

The FPN+PAN structure draws inspiration from the idea of Path Aggregation Network (PANet) [41] and adds two PAN structures on the basis of YOLOv3's FPN module. YOLOv4 adopts the feature fusion method of FPN+PAN, which is conducive to parameter aggregation between different backbone layers and different detection layers, and improves detector performance.

(4)Output terminal. Propose the bounding box regression loss function CIoU in YOLOv4_ Loss and utilize DIoU_ NMS filter prediction box.

2.4 Summary of this chapter

The relevant knowledge introduced in this chapter is the theoretical cornerstone for solving object detection tasks. Firstly, a systematic introduction was given to the theories related to deep learning, including the origin and structural composition of artificial neural networks, weight updating methods, optimizers, advantages of deep neural networks, and classification of deep learning algorithms. Secondly, the structural composition of convolutional neural networks was emphasized, including convolutional layers, pooling layers, fully connected layers, activation functions, and loss functions. Finally, the most representative algorithms among the two types of object detection algorithms were introduced. In the two-stage object detection algorithms, the Faster R-CNN algorithm was mainly introduced, while in the first stage, the YOLOv1, YOLOv2, YOLOv3, and YOLOv4 algorithms were introduced.

3 SMALL OBJECT DETECTION BASED ON IMPROVED YOLOV5

3.1 Introduction

The YOLO series algorithms are known for their fast speed, small model size, and simple structure, making them widely used in industrial production and daily life. Especially YOLOv3 and subsequent algorithms significantly improve detection accuracy while maintaining high detection speed. As shown in Table 3.1, the performance comparison of YOLOv1 to YOLOv4 algorithms is presented. From the table, it can be seen that with each new version released, the overall performance of the algorithm improves accordingly. Although YOLOv4 has lower detection accuracy than YOLOv3, its speed has been greatly improved, making it very suitable for real-time object detection. YOLOv5 combines some features of YOLOv3 SPP and YOLOv4 for faster speed. After comprehensive evaluation, we chose to optimize the YOLOv5 model to further improve algorithm performance.

Table 3.1 Performance Comparison of YOLO Series Algorithms

algorithm	mAP(%)			FPS	advantage	disadvantage
	COCO	VOC 2007	VOC 2012			
YOLOv1	-	63.4	57.9	45.0	Pioneered the first stage of object detection	Inaccurate positioning and low recall rate
YOLOv2	21.6	78.6	73.5	40.0	Cluster extraction prior box	Difficulty in migration
YOLOv3	57.9	-	-	51.0	Multiscale prediction	The effectiveness of large object detection needs to be improved
YOLOv4	43.5	-	-	23.0	Balancing accuracy and speed	The detection accuracy needs to be improved

Although YOLOv5 has attracted attention for its excellent detection speed, its performance on small target datasets is unsatisfactory. This is due to the low resolution of small targets and the interference of noise and complex backgrounds, which makes small target detection more difficult. In response to the above difficulties, the research mainly focuses on the following aspects to improve the small object detection performance of YOLOv5.

(1) To address the issues of low resolution and insufficient information for small targets, the following two solutions are proposed. The first approach is to use larger feature maps to detect small targets. If the resolution of the feature map used for small object detection increases, the local receptive field of the feature map will correspondingly shrink, and the network can detect more small objects with lower resolution. The second approach is multi-scale feature fusion. Due to multiple downsampling operations, deep neural networks lose a lot of feature information, exacerbating the problem of insufficient information for small targets. Shallow feature maps contain more fine-grained information such as color, texture, and edges, so in shallow feature maps, the detailed information of small targets is more abundant. Therefore, it can be considered to optimize the feature fusion structure of YOLOv5, fully integrating more shallow and deep features to enrich the effective information of small targets.

(2) To address the issue of interference from factors such as image noise and background, it is proposed to adopt the method of increasing attention mechanism. Using attention mechanisms to enhance key features and suppress noise interference can improve the model's attention to small targets and enable the network to better recognize occluded targets.

3.2 YOLOv5 Algorithm principle

YOLOv5 has been optimized based on YOLOv4, with varying degrees of improvements made to the input, backbone, Neck, and output layers of the model, further enhancing the algorithm performance. Due to the continuous iterative updating of the YOLOv5 model, the network structures of each version are slightly different, but overall tend to be consistent. All are based on the YOLOv5l model,

with depth_ Multiple and width_ Multiple sets different values to change the depth and width of the network. Among them, depth_ Multiple is used to control the number of submodules, width_ Multiple is used to control the number of convolution kernels in each layer.

Among the various models in this series, YOLOv5n has the smallest volume, fastest speed, and lowest accuracy. YOLOv5s comes second, with depth and width coefficients of 0.33 and 0.5, respectively. After comprehensively measuring the model speed and detection accuracy, YOLOv5s was ultimately chosen as the benchmark model to optimize the algorithm performance by adjusting the network structure. The network structure of YOLOv5s is shown in Figure 3 As shown in Figure 1, modules such as CSP and SPP will be introduced in detail in subsequent sections, and their structural composition will not be listed here.

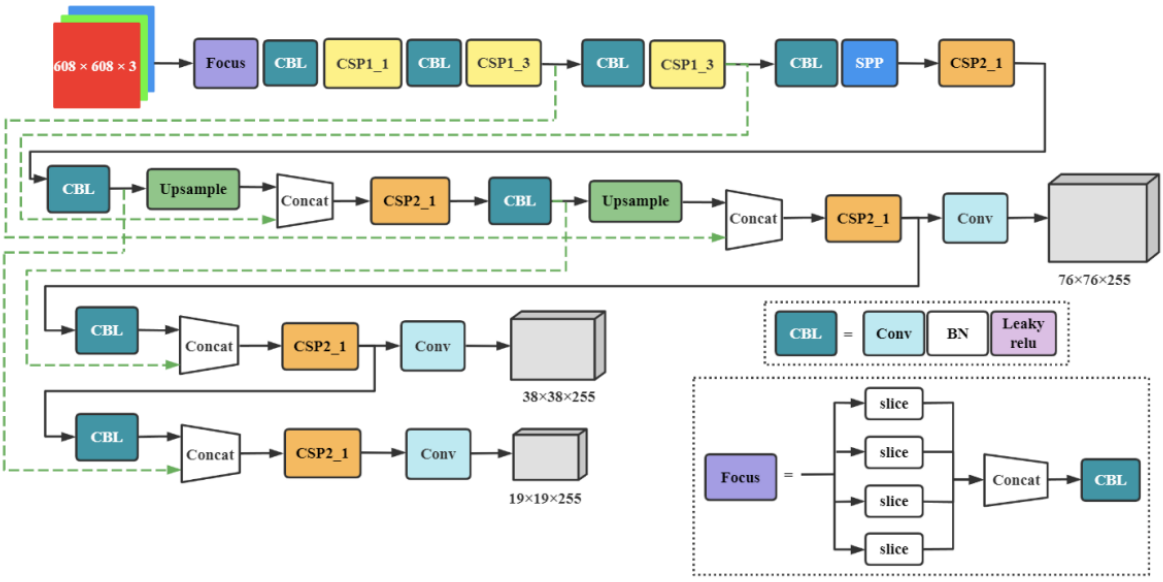


Figure 3.1 YOLOv5s Network Architecture

3.2.1 Input

YOLOv5 has made the following improvements during the model training phase:

(1) Use Mosaic data augmentation. In YOLOv4, the Mosaic data augmentation method was proposed to expand the dataset, and YOLOv5 continued

to use this method. This method concatenates four randomly cropped, scaled, and arranged images into a new image, and then inputs the new image into the network for training. This data augmentation method is beneficial for small object detection.

(2) Adaptive anchor box calculation. YOLOv5 embeds the calculation of anchor box size into the model training process, enabling the model to automatically calculate the optimal anchor box, eliminating the need to separately calculate anchor box size when training on different datasets.

(3) Adaptive image scaling. By using this method to fill the original image with the least number of pixels, it can be scaled to the standard size, reducing redundant information and improving network inference speed.

3.2.2 Backbone

Backbone is a backbone network used to extract image features for use by subsequent network layers. YOLOv5 before version 6.0 used the Focus structure in the backbone network and proposed two CSP structures. Below is a specific introduction to the Focus module and CSP module.

(1) Focus module

Before inputting the image into the backbone network, the Focus module first performs slicing operations on it, splitting the high-resolution image into multiple low resolution images. As shown in Figure 3.2, a 4 × four × Taking the input image of 3 as an example, performing interval sampling on it and concatenating the sampling results in the channel dimension can expand the channel to four times its original size, generating 2 × two × The feature map of 12 is then convolved by the network on the new feature map.

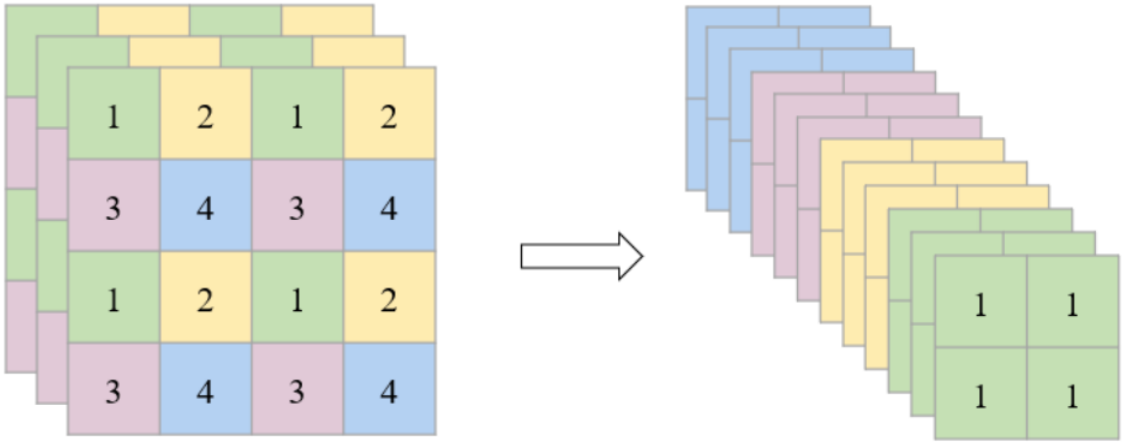


Figure 3.2 Focus Slicing Operation

Scale to 608 × six hundred and eight × The image of 3 is input into YOLOv5s, and after slicing through the Focus layer, it becomes 304 × three hundred and four × 12 feature maps, then input 32 convolutional layers for convolution operation, and finally obtain 304 × three hundred and four × 32 size feature map. By using the Focus slicing operation to convert the planar information of the image to the channel dimension, double downsampling can be achieved while ensuring lossless image information, which is beneficial for improving network speed.

(2) CSP module

Due to the problem of repeated gradient calculations during network optimization, the inference process requires too much computation. To solve this problem, as early as YOLOv4, the CSP module was proposed by drawing on the ideas of CSPNet [60]. Research has shown that introducing CSP modules is beneficial for enhancing the learning ability of networks, ensuring accuracy while reducing computational complexity, and also reducing memory consumption.

The backbone networks of YOLOv4 and YOLOv5 both use CSP structures, with the difference being that YOLOv5 designs two types of CSP modules, as shown in Figure 3.3, where CSP1_ X is used for the backbone network, while

CSP2_ X is applied to the Neck network. CSP1_ X first divides the feature map into two parts according to channels, one part performs regular convolution operations, and the other part constructs residual components using the idea of residual networks. Finally, these two parts are merged to obtain a new feature map. This design can avoid duplicate calculation of gradient values and improve the inference speed of the model. Moreover, the CSP structure with residual components can enhance gradient values during backpropagation. When the number of layers in the backbone network is deep, it can alleviate the problem of gradient disappearance and enhance the network's feature extraction ability. And CSP2_ X uses convolutional layers instead of residual components to divide the input feature map into two parts, calculate them separately, and then fuse them, which can retain more image information.

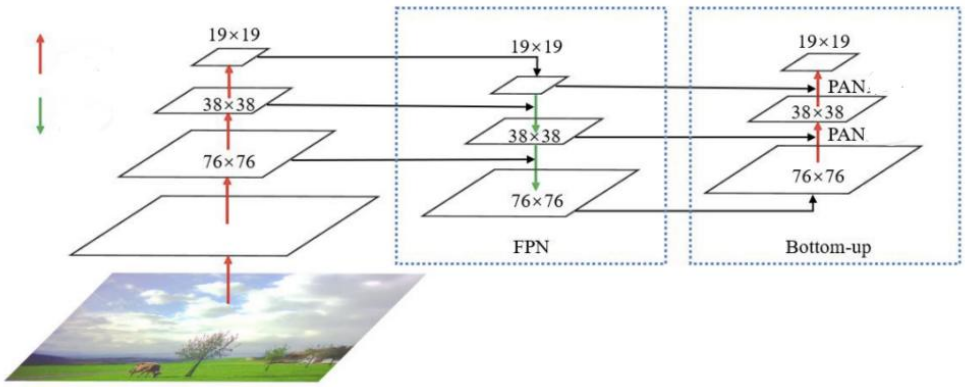


Figure 3.3 CSP module structure diagram

The backbone network of YOLOv5 model v6.0 version is different from previous versions, mainly manifested in the following aspects:

(1) In order to facilitate model deployment, the Focus module has been replaced with 6 × 6 convolutional layers.

(2) Using C3 module instead of CSP module, and introducing residual component into the last C3 module of the backbone network. C3, as the name suggests, is a module that includes three standard convolutional layers and several Bottleneck layers.

(3) In order to improve speed, the number of repetitions of C3 modules in the P3 layer of the backbone network has been reduced.

(4) Replace the SPP module with the SPPF module and place it at the end of the backbone network.

The structure of SPP module and SPPF module is shown in Figure 3.4. As shown in the figure, the SPP module inputs feature maps in parallel into a convolution kernel size of 5 × 5. 9 × 9 and 13 × The maximum pooling layer is 13, and then the feature maps of four different receptive fields are concatenated together. The SPPF module, on the other hand, serially inputs feature maps into a convolution kernel with a size of 5 × Maximum pooling layer of 5, followed by feature fusion. Due to two 5 × The maximum pooling layer of 5 and a 9 × The maximum pooling layer of 9 has the same effect, and three 5 × The maximum pooling layer of 5 and a 13 × The operation result of the maximum pooling layer of 13 is also the same. Therefore, using the SPPF structure can achieve the same effect, and the model has smaller computational complexity and higher efficiency.

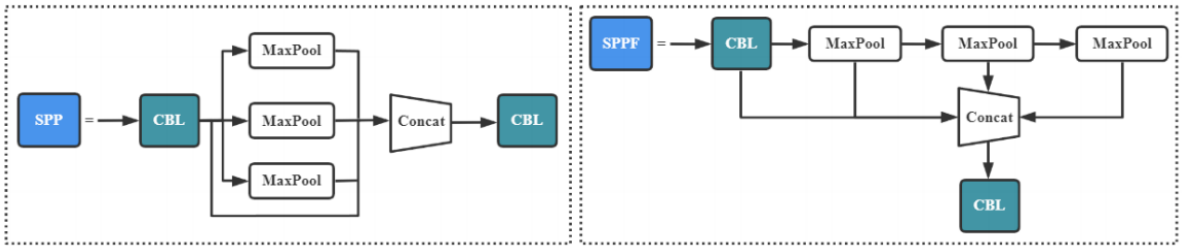


Figure 3.4 Comparison of SPP and SPPF Structures

3.2.3 Neck

As feature extraction deepens, some local information in the image may disappear. By using the Neck network to fuse feature maps from different network levels, richer feature information can be obtained in the image. These processed features can be input into the Head layer for better classification and localization.

YOLOv3 uses an FPN structure to upsample deep feature maps and fuse them with shallow feature maps, and then predicts each layer of feature maps in the feature pyramid separately. On the basis of YOLOv3, YOLOv5 has added a

bottom-up feature pyramid, which includes two PAN structures that can downsample shallow feature maps and fuse them with deep feature maps. As shown in Figure 3.5, the structure diagram of FPN and PAN is presented. This dual pronged design approach can transmit strong semantic information from top to bottom and strong localization information from bottom to top, effectively improving algorithm performance.

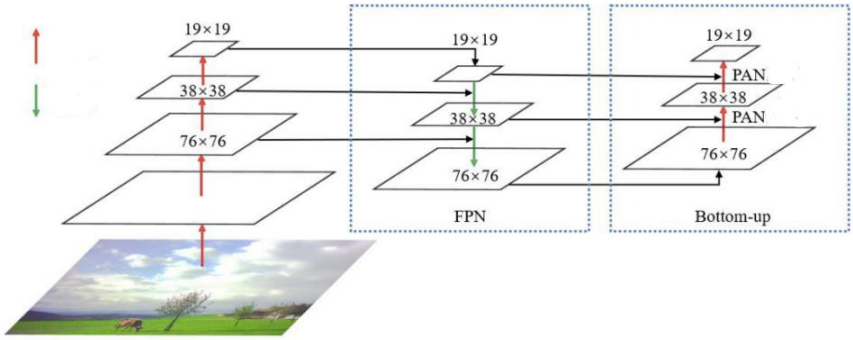


Figure 3.5 FPN+PAN Structure

The Neck network of YOLOv4 and YOLOv5 both adopt FPN+PAN structure, although the idea is the same, the network structure is different. YOLOv5 introduces the CSP module into the Neck network, replacing the ordinary convolution operation in YOLOv4, which can promote better feature fusion in the network.

3.2.4 Head

When using YOLOv5 to process object detection tasks, the image is first input into the backbone network to extract features, and then the features extracted by the backbone network are input into the Neck network for processing. Finally, the Head layer outputs the target category and corrects the position of the candidate boxes based on the position offset, thereby obtaining more accurate detection results.

(1) Loss function

The loss function of YOLOv5 consists of three parts: bounding box loss, confidence loss, and classification loss. The bounding box regression loss uses

CIoU_ Loss calculation, this loss function combines the overlap area, center point distance, and aspect ratio information between the predicted box and the target box, improving the regression speed and accuracy of the detection box. CIoU_ The calculation method for Loss is as follows.

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \tag{3.1}$$

$$\alpha = \frac{\mathrm{v}}{1 - IoU + v} \tag{3.2}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} + \alpha v \tag{3.3}$$

Both confidence loss and classification loss default to using the BCEWithLogitsLoss loss function, which combines the binary cross entropy (BCE) loss function and the sigmoid activation function.

(2) NMS non maximum suppression

In object detection tasks, the network outputs a large number of detection boxes, and many detection boxes contain the same target, but each target ultimately only requires one detection result. Usually, during the post-processing process, the NMS algorithm is used to filter out the best result from multiple prediction boxes. The NMS operation obtains all local maximum elements through search and suppresses non local maximum elements to achieve the filtering purpose. The specific operation steps are as follows: Sort the scores of all detection boxes, and then select the box with the highest score. Calculate the IoU value between the remaining detection boxes and that detection box. If it exceeds the set threshold, remove it from the remaining detection boxes. From the unprocessed detection boxes, select the one with the highest score and repeat this process. The IoU values between the remaining detection boxes are all very low, and it is basically possible to achieve one target with only one detection result.

3.3 Multi scale object detection

3.3.1 Motivation for improvement

When using convolutional neural networks to extract image features, as the number of convolutional layers deepens, the resolution of the feature map gradually decreases, and the local receptive field can perceive the range of the original image continuously expanding. Moreover, feature maps closer to the top level tend to focus more on the global information of the image, so the deep features extracted by deep neural networks are very unfavorable for small object detection. Although the local receptive field of shallow feature maps is smaller and contains more positional and detailed information, which can to some extent compensate for the shortcomings of deep features, if only shallow features are used to detect targets, it will cause serious false positives and missed detections due to the lack of guidance from advanced semantic information. So some researchers propose to fuse shallow features with deep features, and then predict object positions on fused feature maps of different scales. Research has shown that this multi-scale detection method can effectively improve algorithm performance.

In YOLOv3, multi-scale object detection technology was used to predict targets of different sizes. YOLOv4 and YOLOv5 adopted this idea and made a series of improvements to the network structure. In YOLOv5, if the size of the input image is 640 × six hundred and forty × 3. After down sampling by 8x, 16x, and 32x, 80 was obtained, respectively × eighty × 255, 40 × forty × 255 and 20 × twenty × The network ultimately performs object detection on the feature maps of 255 at three different scales. Among these three scale feature maps, the one with the smallest local receptive field is the 8-fold down sampling feature map. That is, if this feature map is mapped to the original input image, each grid corresponds to the original image 8 × Area 8. For targets with smaller resolutions, the receptive field of the feature map obtained by 8-fold downsampling is still too large, which can easily lead to the loss of position and detail information of some small targets.

From the above analysis, it can be seen that YOLOv5 has difficulty learning resolutions below 8 × The feature information of the small target is 8, while in a

typical small target dataset like TinyPerson [61], the average absolute size of the target to be detected is only 18 pixels. Due to YOLOv5 failing to detect a large number of small targets with resolutions below 64 pixels, its performance on small target datasets is poor. In order to improve the current situation of missed target detection, we optimized the Head structure of YOLOv5s and added a detection head for detecting small targets on the basis of the original three-scale detection head. By adding small-scale detection heads, the network can detect small targets with a resolution of no less than 16 pixels, effectively increasing the number of small targets detected by the network.

3.3.2 Design of multi-scale detection head

The original YOLOv5 model performs detection on feature maps at three different scales, and sets three different specifications of prior boxes for each scale feature map to detect large, medium, and small targets. The model calculates 9 optimal anchor box values on different datasets through adaptive anchor box calculation.

In YOLOv5, the feature map used for object detection can perceive a minimum range of 8 from the original image × The area of 8 makes it easy for the model to miss small targets with a width and height less than 8 pixels. To improve this issue, we added a 4x downsampling detection head to the YOLOv5s model and named it YOLOv5s-P2. The minimum target resolution that YOLOv5s-P2 can detect is 4 × 4. It is more powerful than the original network, generating 12 prior boxes of different scales through co aggregation. It can predict targets of different scales on downsampled feature maps at 4x, 8x, 16x, and 32x, greatly improving the multi-scale object detection performance of the algorithm. Due to the addition of a small-scale detection head in the improved model, the feature fusion method at the Neck end has also undergone corresponding changes, but overall it still follows the FPN+PAN structure. As shown in Figure 3.6, it is the feature fusion structure of the YOLOv5s P2 model. C2, C3, C4, and C5 in the figure correspond to the downsampling feature maps extracted by the backbone network at 4 times, 8 times,

16 times, and 32 times, respectively. F3 and F4 are feature fusion layers, and P2, P3, P4, and P5 are detection layers.

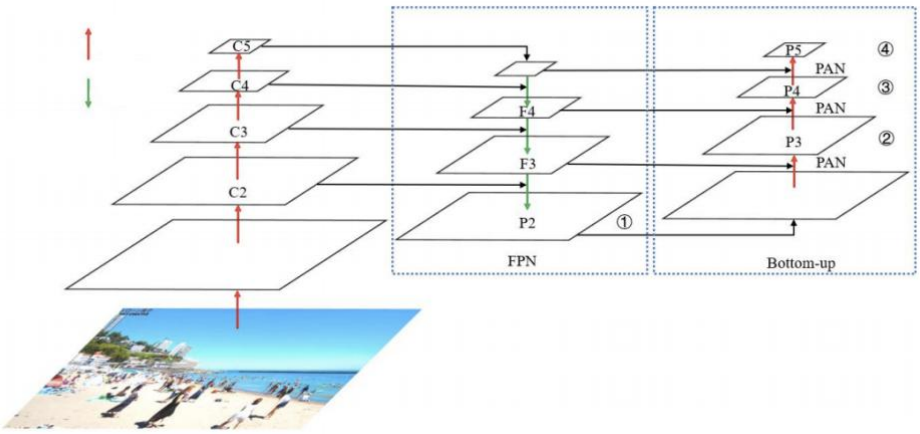


Figure 3.6 YOLOv5s-P2 Feature Fusion Structure

As shown in Figure 3.6, the Neck network performs upsampling fusion on the input feature map, followed by downsampling fusion, and then predicts the fused feature map separately. The specific process is as follows: (1) Upsampling fusion. In order to achieve fine-grained feature detection, the C5 feature map input from the backbone network is first upsampled, and the C4 feature map is fused to obtain the F4 feature map. After performing convolution and other operations on F4, upsampling is performed and fused with C3 to obtain fine-grained feature maps for the F3 layer. Continue upsampling F3 to obtain a larger scale feature map, then fuse it with the C2 feature map to perform small object detection on the fused large-scale feature map. Finally, the detection result is output by the P2 detection layer. (2) Downsampling fusion. First, perform downsampling on the feature map of the P2 detection layer, fuse it with the F3 feature map at the same scale, and then output the detection result of small targets by the P3 detection layer. The network continues downsampling and then fuses with F4 and C5 respectively to detect medium and large targets on the obtained 16x and 32x downsampling feature maps. Finally, the prediction results are output by the P4 and P5 detection layers.

The original YOLOv5 model had three output layers: P3, P4, and P5, which were used to detect small, medium, and large targets, respectively. After adding the P2 detection layer, the network can perform detection on four different scale feature maps, and its network structure is shown in Figure 3.7. The improved network transforms the P3 detection layer into an F3 feature fusion layer, which continues to upsample the feature map after F3 and fuses the 4x downsampled feature map extracted by the backbone network. Then, the network detects small targets on the fused feature map, and the detection result is finally output by the P2 detection layer. As shown by the red dashed line in the figure, after the first C3 module in the backbone network, a feature output is added and connected to the Neck end for feature fusion at the same scale. After C3 module and convolution operations, it is input into the P2 detection layer for prediction. Due to the fusion of shallow and high-resolution feature maps by the P2 detection head, more shallow feature information can be transmitted to deep feature maps during the downsampling process. Therefore, the introduction of P2 detection heads makes the network particularly sensitive to small targets, improving the algorithm's small target detection performance.

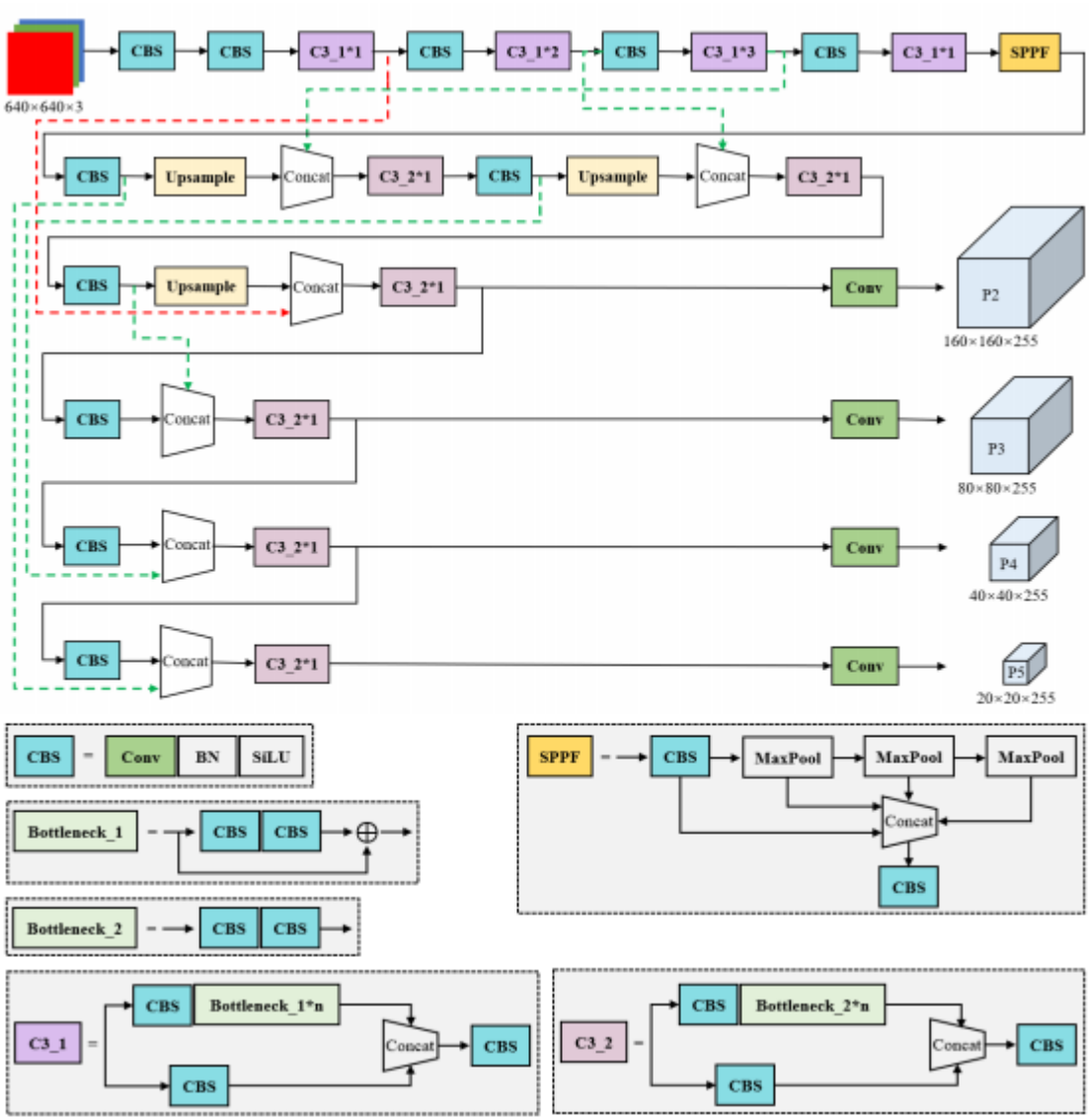


Figure 3.7 YOLOv5s-P2 Network Structure

According to the network structure shown in Figure 3.7, the detection process of the YOLOv5s-P2 model can be roughly divided into the following steps:

(1) The backbone network extracts image features. The first convolutional layer performs a 2x downsampling operation on the input image, and the convolutional layers in front of the four C3 modules also perform a stride of 2 by 3 × 3 convolutions, so a total of 5 downsampling occurs in the backbone network, and finally a 32 fold downsampling feature map is input into the Neck end.

(2) Neck network for feature fusion. The Neck end includes two feature fusion paths: upsampling fusion and downsampling fusion. The Neck network first continuously upsamples the input features to fuse the shallow features extracted by

the backbone network, achieving the goal of transmitting deep features to the shallow feature map. After the upsampling operation is completed, perform a downsampling operation to transfer shallow features to deep feature maps.

(3) The detection head outputs prediction results. The P2, P3, P4, and P5 detection layers output classification and localization results of targets at different scales on the fused feature map.

Based on the above description, it can be seen that adding a 4x downsampling detection head not only increases the network depth, but also enables the network to better learn multi-level feature information of the target, enhancing the network's detection ability for multi-scale targets in complex environments. On the other hand, it can transfer more shallow features to deep features, enabling the network to obtain more information about small targets and improving the model's ability to detect small targets.

3.4 Object detection based on attention mechanism

3.4.1 Attention mechanism

Human visual attention is a unique signal processing technique of the human brain, which is a survival and development mechanism gradually formed during the long process of human evolution. Scientists have found through their research on the human visual system that when processing large amounts of information, people automatically ignore secondary information and focus more on the key elements that require attention, thus quickly filtering out high-value information within a limited time. In recent years, researchers have applied attention mechanisms to neural network models by imitating human visual cognition, enabling the network to automatically focus on and learn important feature information of images. Research has shown that introducing attention mechanisms can effectively improve the network's ability to extract key information and enhance model performance.

Attention mechanisms mainly include three types: spatial domain, channel domain, and mixed domain. The spatial domain is represented by spatial attention [62] . This type of model reduces dimensionality along the channel dimension of

the feature map, attaching different weights to different regions of the feature map to enhance the features of key regions and suppress the features of non important regions such as background. The channel domain is represented by SENet [21] . The model compresses the feature map along the spatial dimension and generates different weights for each feature channel to represent the importance of each channel. Then, based on the feature weights, the original feature map is recalibrated to make the model focus more on key channels and suppress non key channels. The hybrid domain is represented by BAM (Bottleneck Attention Module) [63] and CBAM (Convolutional Block Attention Module) [64] , which focus on both channel and spatial features, resulting in better algorithm performance.

SENet improves model performance by assigning greater weights to channels containing important information and embedding them into classification or detection models, achieving good results. However, the drawback of this module is that it loses some information when embedding spatial information into channels through Global Average Pooling (GAP). The CBAM model utilizes both global average pooling and global maximum pooling (GMP), and combining these two pooling strategies can prevent information loss. So we apply the CBAM attention mechanism to the YOLOv5s model to retain more effective information.

The information carried by different dimensions of feature maps varies. The channel dimension focuses on the abstract expression of features, while the spatial dimension focuses more on the positional information of objects. CBAM includes Channel Attention Module (CAM) and Spatial Attention Module (SAM), which are combined in a serial manner to serialize attention feature map information in both channel and spatial dimensions. Its network structure is shown in Figure 3.8.

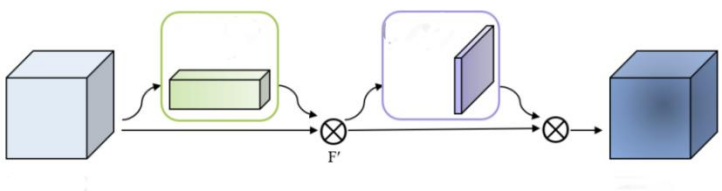


Figure 3.8 CBAM Network Architecture [64]

As shown in Figure 3.8, the CBAM process is as follows: first, the channel attention module adaptively corrects the input feature map F to obtain the feature map F ', then the spatial attention module corrects F', and finally obtains the feature map F " processed by the CBAM module. The algorithm process can be expressed using the following formula, where "⊗" represents element level multiplication, and the input feature map $F \in \mathbb{R}^{C\times H\times W}$. Channel attention feature map $M_C \in \mathbb{R}^{C\times 1\times 1}$. Spatial attention feature map $M_S \in \mathbb{R}^{1\times H\times W}$.

$$F' = M_C(F) \otimes F \tag{3.4}$$

$$F'' = M_S(F') \otimes F' \tag{3.5}$$

The structure of the channel attention module is shown in Figure 3.9. In order to efficiently compute channel attention features, GAP and GMP pooling methods are used to compress the feature map in the spatial dimension, resulting in two Cs × one × Feature map of 1. Then, these two feature maps are input into a shared multi-layer perceptron (MLP), which is a two-layer artificial neural network. The first network layer is used to reduce the dimensionality of the feature map, with the number of neurons being C/r and r being the dimensionality reduction coefficient. Next, the two channel attention vectors output by MLP are fused, and then input into the Sigmoid activation function to obtain the channel attention vector Mc. Mc can be calculated using formula 3.6, where Facvg and Fmcax represent the features obtained after average pooling and maximum pooling operations, respectively.

$$\begin{aligned} M_C(F) &= \sigma(MLP(Avgpool(F)) + MLP(Maxpool(F))) \\ &= \sigma(w_1(w_0(F_{avg}^{c})) + w_1(w_0(F_{max}^{c}))) \end{aligned} \tag{3.6}$$

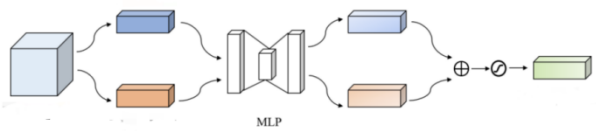


Figure 3.10 Structure diagram of spatial attention module [64]

3.4.2 Model improvement based on CBAM

The main reasons for introducing CBAM attention mechanism into the Neck structure of YOLOv5s are as follows:

(1) There are generally phenomena such as small target scale, dense arrangement, high noise, and background interference on small target datasets. Adding attention mechanism can suppress the interference of irrelevant information, retain more key features of the target to be detected, make the network focus on more small targets, and improve detection accuracy.

(2) The use of spatial attention mechanism helps to clarify the position of small targets, while channel attention mechanism can model the importance of feature channels. CBAM attention mechanism focuses on both spatial and channel dimensional features, resulting in better performance.

(3) CBAM is a simple, efficient, plug and play lightweight attention module that can be integrated into any CNN for end-to-end training along with the underlying model, with negligible computational overhead.

(4) The biggest role of using attention mechanism is to enable the model to selectively ignore irrelevant information and focus on useful features for the target task. Therefore, introducing attention mechanism can improve the network's feature processing ability. In the YOLOv5 network, the most crucial part of feature processing lies in the Neck structure. Because the Neck structure is located between the backbone network and the output layer, playing a bridging role, it is used to fully aggregate the features extracted by the backbone network and then send them to the output layer for prediction. Therefore, the ability of the Neck network to process features is the key factor affecting algorithm performance.

Based on the above reasons, we have added CBAM attention mechanism to the Neck network of YOLOv5s, enabling the network to fully utilize computing resources and filter out more useful feature information for the target task. By adding CBAM, it is possible to learn more effective features and improve network performance without increasing network depth, width, and input image resolution.

The attention mechanism distinguishes the importance of feature information by adjusting the weight size. Considering that the fusion of secondary features does not contribute much to the detection results and also increases computational complexity, it is necessary to reduce the weight of secondary features and assign higher weights to important features. Due to the fact that YOLOv5 first fuses features to varying degrees at the Neck end, and then the detection head directly outputs prediction results on the fused feature map, attention mechanisms need to be added before the next feature fusion at the Neck end to enhance key features and suppress irrelevant features.

The network structure after adding the CBAM module is shown in Figure 3 11. The YOLOv5s-P2-CBAM model adds a CBAM module after each C3 module on the Neck end and before the convolution operation. Adding an attention mechanism between two feature fusions to enhance the network's attention to small targets. After adding the CBAM module to the network, perform feature enhancement on the feature map before the next feature fusion. Make the network ignore the interference of irrelevant information, focus on key features, and fuse relatively important features. This not only makes the fused feature map contain more effective information, improves the accuracy of small target localization, but also achieves the goal of reducing computational complexity and improving model speed.

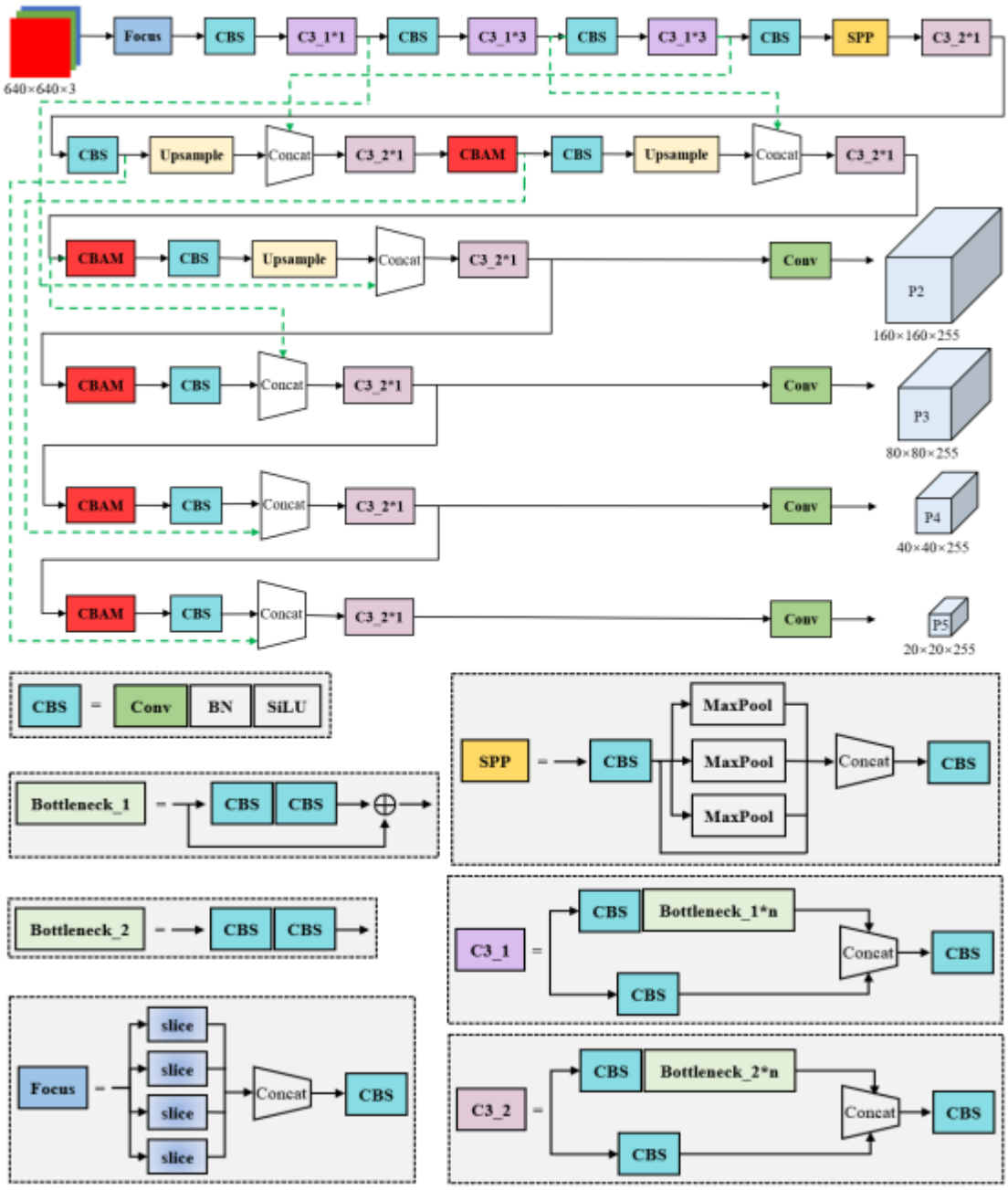


Figure 3 .11 YOLOv5s-P2-CBAM Network Architecture

From Figure 3 11. It can be seen that after adding the CBAM module, the feature processing process on the Neck end is as follows:

(1) In the FPN structure, deep feature maps are continuously upsampled to fuse larger scale shallow feature maps. For each fused feature map, perform C3 operation first to reduce the computational complexity of the network. Add CBAM attention mechanism to enhance the network's ability to focus on key channels and regions in

the feature map. Then perform convolution and upsampling operations for the next feature fusion.

(2) Introducing the CBAM attention mechanism in the PAN structure to update the previously fused feature map, enhancing the network's attention to important features, and incorporating more useful shallow geometric features into deep features. Then perform downsampling on the feature map and fuse it with the low resolution feature map.

From the above analysis, it can be seen that adding the CBAM module is beneficial for enhancing the feature fusion effect on the Neck end and improving the model's ability to analyze complex scenes. Enable the model to automatically ignore the interference of background and other factors, and focus more attention on the areas that need to be focused on. Thus, limited computing resources can be utilized to obtain more effective information and improve the accuracy of object detection.

3.5 Feature fusion network based on BiFPN

3.5.1 Multi scale feature fusion network

With the deepening of network layers, the semantic information of feature maps becomes more and more abundant, but the detailed information gradually decreases, which is extremely unfavorable for small object detection. But if only shallow features with rich details are used for detection, it will also reduce the performance of the detector. In order to overcome the shortcomings of deep and shallow features, the fusion of deep and shallow features is generally adopted to obtain more comprehensive and rich feature information. As shown in Figure 3 As shown in Figure 12, three typical design forms of multi-scale feature fusion networks are presented, and these three structures to some extent also represent the development process of feature fusion networks. Below, we will introduce the characteristics of each network structure one by one.

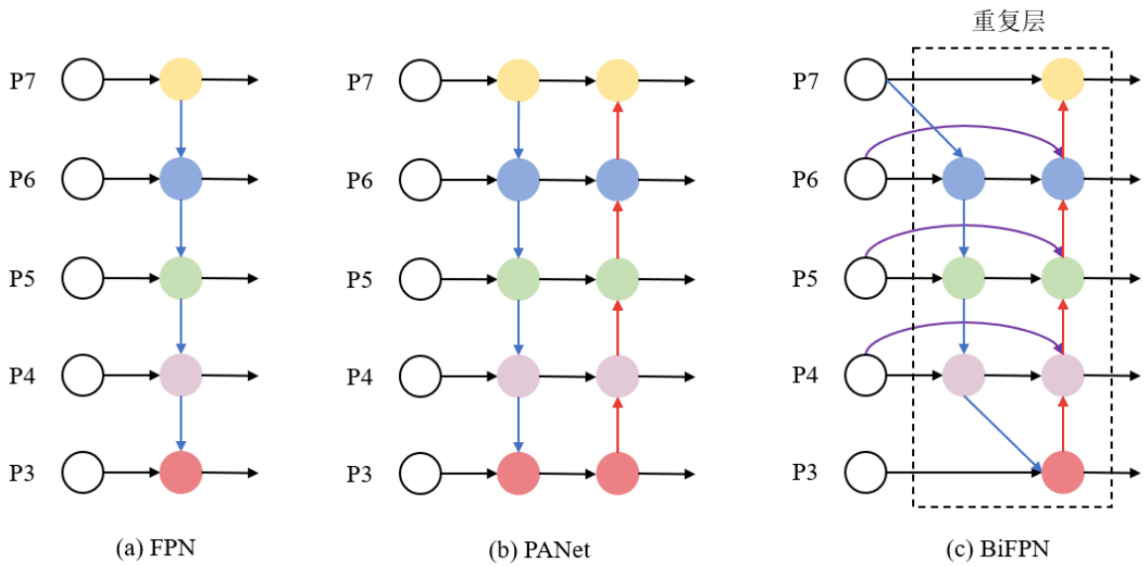


Figure 3.12 Variations of Multi scale Feature Fusion Network[45]

The FPN [40] network structure is shown in Figure 3 As shown in 12 (a). As shown in the figure, FPN can only convey strong semantic information from top to bottom, by upsampling deep feature maps and fusing them with shallow feature maps. However, in deep neural networks, the path to transfer shallow features to deep features is generally long, and most of the target information may have been lost during the downsampling process, which fails to perform feature fusion well.

The network structure of PANet [41] is shown in Figure 3 As shown in 12 (b). This network includes two feature fusion paths, top-down and bottom-up, which shorten the distance from shallow features to deep features, optimize the feature fusion method of FPN network to a certain extent, improve the target detection effect, but also increase the number of network parameters and computation.

The structure of the Weighted Bidirectional Feature Pyramid Network (BiFPN) [45] is shown in Figure 3 As shown in 12 (c). The effectiveness of bidirectional feature fusion has been verified in the PANet network, but the structure is relatively simple. In 2020, Tan et al. proposed BiFPN in EfficientDet network, further optimizing the PANet network. Compared with PANet, BiFPN has made the following improvements:

(1) removing nodes with only one input. Because only one input node carries less information and does not participate in feature fusion, retaining this node not only benefits feature fusion but also increases computational complexity.

(2) Increase jump connections. Adding skip connections to the features extracted from the backbone network and the features involved in downsampling fusion at the same scale can fuse more features without increasing costs, effectively alleviating the phenomenon of feature loss caused by too many network layers.

(3) Consider bidirectional paths as a unit. The BiFPN network considers a set of feature fusion paths from top to bottom and bottom-up as a basic network layer, which is repeated and stacked multiple times to fuse more advanced features.

(4) Weighted feature map fusion. In the past, feature fusion networks did not differentiate features and directly summarized and added them up. And BiFPN proposes a weighting strategy, which adjusts the importance of features by assigning learnable weights to features of different scales participating in fusion.

3.5.2 Model improvement based on BiFPN

In the YOLOv3 and subsequent versions of the YOLO series algorithms, before inputting features into the detection layer to predict object category and position information, it is necessary to fully fuse the features extracted by the backbone network on the Neck end. So the feature fusion effect on the Neck end is very important for the detection results.

As early as YOLOv3, the idea of FPN network was borrowed, proposing the fusion of deep and shallow features for multi-scale object detection. However, YOLOv3 only includes a top-down feature fusion path and cannot fully fuse features. Subsequently, YOLOv4 and YOLOv5 were inspired by PANet and proposed the use of FPN+PAN structure for multi-scale feature fusion. On the basis of FPN, a bottom-up feature fusion path has been added. It should be noted that the PAN structure in YOLOv4 and YOLOv5 networks is different from that in PANet networks. PANet network overlays feature information, while YOLOv4 and YOLOv5 concatenate feature maps in the channel dimension. Compared with YOLOv3, the introduction of PAN structure can transfer more shallow features such as position information to deep

features, which to some extent improves the effectiveness of object detection. However, this feature fusion method is relatively simple and there is still some room for optimization. In order to compensate for the shortcomings of the FPN+PAN structure, we borrowed the ideas of the BiFPN network and optimized the Neck structure of YOLOv5s, enabling the network to perform better feature fusion.

The improved model is named YOLOv5s-P2-CBAM-BiFPN, which performs multi-layer feature fusion. The fusion method is shown in Figure 3.13. Deep features are continuously upsampled and fused with shallow features. After upsampling is completed, shallow features are continuously downsampled and fused with deep features. And on the basis of the original FPN+PAN structure, the network has added two horizontal connection paths to integrate the features extracted by the backbone network into the feature map to be detected. This improvement method can fuse more feature information in the detected feature map, enhance the expression ability of the feature map, and improve the performance of the detector without increasing too much computational complexity.

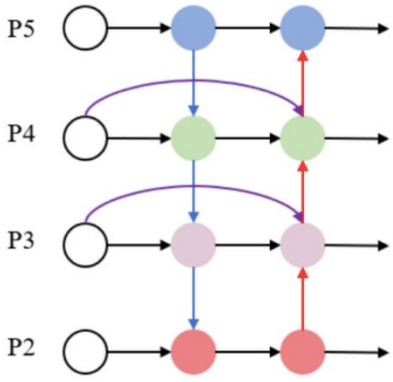


Figure 3.13 Feature fusion structure based on BiFPN

From Figure 3 13. It can be seen that the Neck network of the YOLOv5s-P2-CBAM-BiFPN model forms a bidirectional cross scale connection structure, which integrates deep and shallow features across scales through two feature fusion paths: top-down and bottom-up. This design approach enhances the information transfer between feature maps at different network levels, effectively improving the problem

of image feature information loss caused by changes in feature map resolution. The improved Neck network can integrate more location and detail information, making the network better focus on the area where small targets are located.

As shown in Figure 3 As shown in Figure 14, the network structure of YOLOv5s P2 CBAM BiFPN is demonstrated. Figure 3 In Figure 14, the second and third C3 modules of the backbone network correspond to 8 times and 16 times downsampling feature maps, respectively. The improvement method is shown by the red dashed line in the figure. First, connect the feature map output by the second C3 module to the Neck end and fuse it with the feature map of the P3 detection layer at the same scale. Similarly, fuse the feature map output by the third C3 module with the feature map of the P4 detection layer. The YOLOv5s-P2-CBAM-BiFPN model has an additional feature fusion path in both the P3 and P4 detection layers. For the two scales of P3 and P4, the feature maps to be detected are obtained by fusing three parts of features from different network levels at the same scale. These three parts of features are obtained by downsampling from the previous network layer, feature maps generated by upsampling fusion at the Neck end, and feature maps extracted from the backbone network.

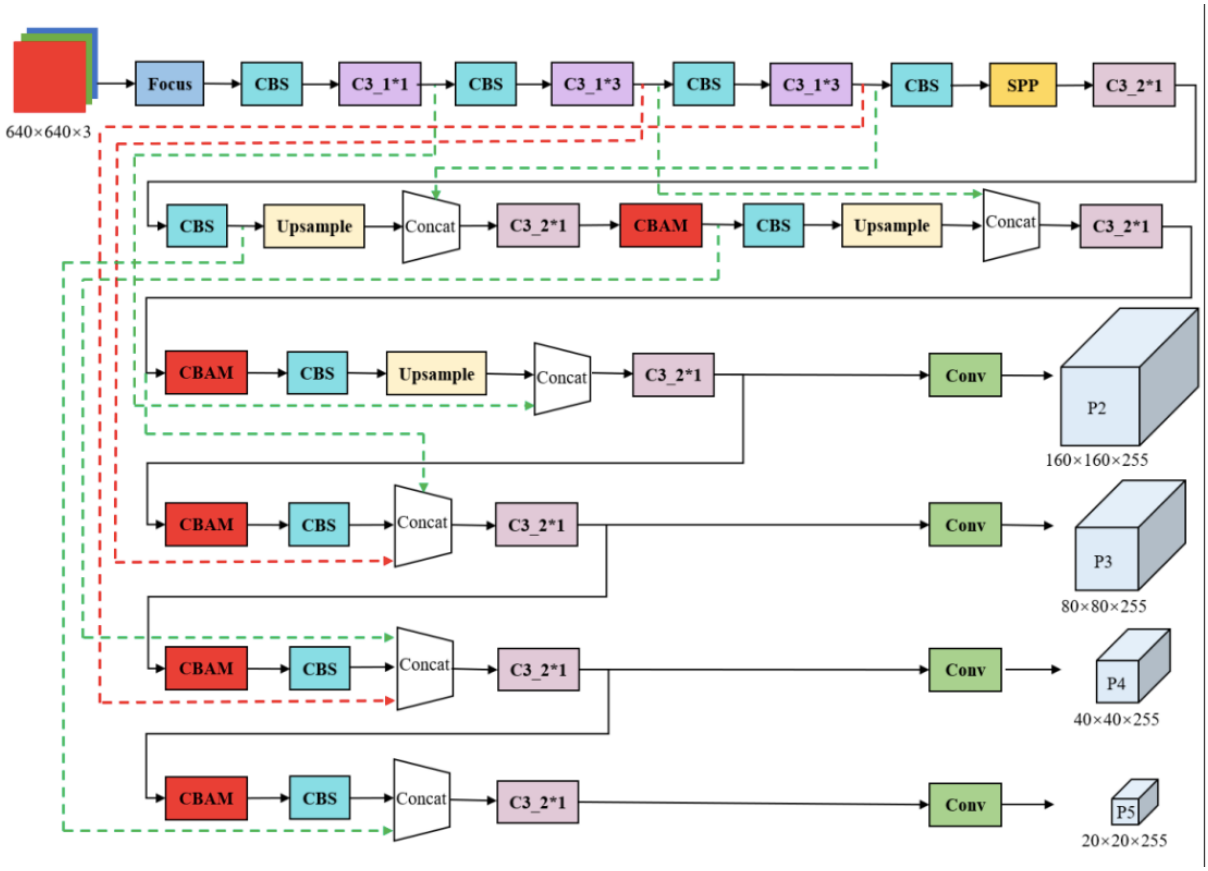


Figure 3. 14 YOLOv5s-P2-CBAM-BiFPN Network Architecture

During the research process, inspired by the BiFPN network, the feature fusion structure of YOLOv5s was optimized. By increasing the information coupling between feature maps of different scales, the impact of feature information loss caused by upsampling and downsampling is reduced, effectively improving the feature fusion ability of the original YOLOv5. It should be noted that this study focuses on improving the small object detection performance of YOLOv5. The improved model was validated on a small object dataset, and it was found that weighting the features involved in fusion actually reduces algorithm performance. So we only borrowed the idea of bi-directional cross scale connections from BiFPN networks and did not adopt a weighted feature fusion strategy.

3.6 Overall architecture of the model

In order to improve the small object detection capability of YOLOv5, three improvement methods have been proposed for its Head layer and Neck network: adding P2 detection heads, introducing CBAM attention mechanism in Neck network,

and optimizing the feature fusion method of Neck network. The improved network performance has been significantly improved, and its network structure diagram and parameter table are shown below.

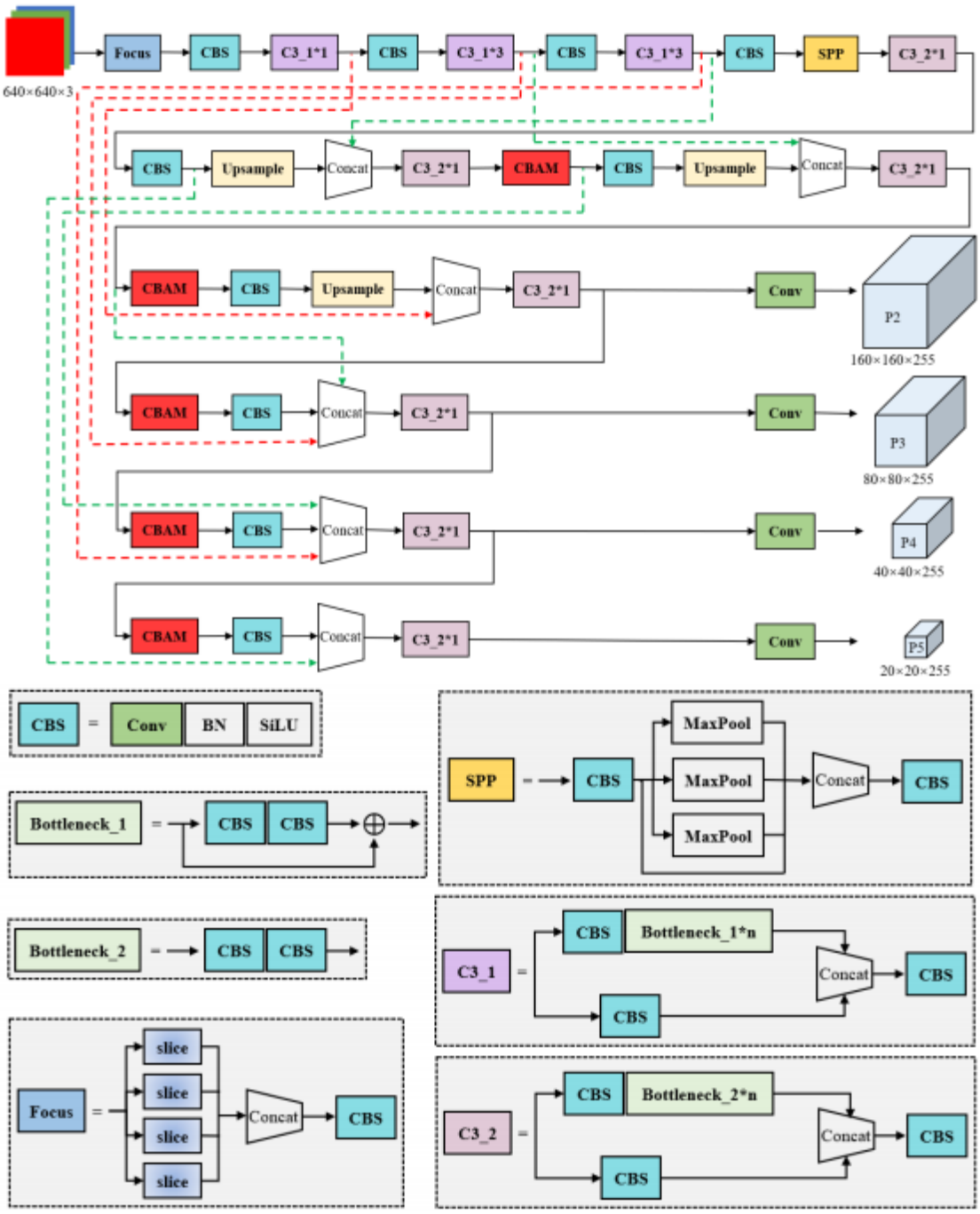


Figure 3.15 Model Overall Architecture

Table 3.2 Network Structure and Parameters of YOLOv5s-P2-CBAM-BiFPN

Name	Layer	Output	Param
Input	-	(3, 640, 640)	-
Focus_1	Conv	(32, 320, 320)	3488
Conv_2	Conv2d, BN, SiLU	(64, 160, 160)	18496
C3_3	Conv, Conv	(32, 160, 160)	2080
	Conv	(64, 160, 160)	4160
Conv_4	Conv2d, BN, SiLU	(128, 80, 80)	73856
C3_5	Conv, Conv	(64, 80, 80)	8256
	Conv	(128, 80, 80)	16512
Conv_6	Conv2d, BN, SiLU	(256, 40, 40)	295168
C3_7	Conv, Conv	(128, 40, 40)	32896
	Conv	(256, 40, 40)	65792
Conv_8	Conv2d, BN, SiLU	(512, 20, 20)	1180160
SPP_9	Conv	(256, 20, 20)	131328
	Conv	(512, 20, 20)	524800
C3_10	Conv, Conv	(256, 20, 20)	131328
	Conv	(512, 20, 20)	262656
Conv_11	Conv2d, BN, SiLU	(256, 20, 20)	131328
Upsample_12	-	(256, 40, 40)	-
Concat_13	-	(512, 40, 40)	-
C3_14	Conv, Conv	(128, 40, 40)	65664
	Conv	(256, 40, 40)	65792
CBAM_15	Channel Attention	(256, 1, 1)	8192
	Spatial Attention	(1, 40, 40)	98
Conv_16	Conv2d, BN, SiLU	(128, 40, 40)	32896
Upsample_17	-	(128, 80, 80)	-
Concat_18	-	(256, 80, 80)	-
C3_19	Conv, Conv	(64, 80, 80)	16448

Continued Table 3.2 Network Structure and Parameters of YOLOv5s-P2-CBAM-BiFPN

Name	Layer	Output	Param
C3_19	Conv	(128, 80, 80)	16512
CBAM_20	Channel Attention	(128, 1, 1)	2048
	Spatial Attention	(1, 80, 80)	98
Conv_21	Conv2d, BN, SiLU	(128, 80, 80)	16512
Upsample_22	-	(128, 160, 160)	-
Concat_23	-	(192, 160, 160)	-
C3_24	Conv, Conv	(64, 160, 160)	12352
	Conv	(128, 160, 160)	16512
CBAM_25	Channel Attention	(128, 1, 1)	2048
	Spatial Attention	(1, 160, 160)	98
Conv_26	Conv2d, BN, SiLU	(128, 80, 80)	147584
Concat_27	-	(384, 80, 80)	-
C3_28	Conv, Conv	(64, 80, 80)	24640
	Conv	(128, 80, 80)	16512
CBAM_29	Channel Attention	(128, 1, 1)	2048
	Spatial Attention	(1, 80, 80)	98
Conv_30	Conv2d, BN, SiLU	(128, 40, 40)	147584
Concat_31	-	(640, 40, 40)	-
C3_32	Conv, Conv	(128, 40, 40)	82048
	Conv	(256, 40, 40)	65792
CBAM_33	Channel Attention	(256, 1, 1)	8192
	Spatial Attention	(1, 40, 40)	98
Conv_34	Conv2d, BN, SiLU	(256, 20, 20)	590080
Concat_35	-	(512, 20, 20)	-
C3_36	Conv, Conv	(256, 20, 20)	131328
	Conv	(512, 20, 20)	262656
Detect_37	-	(102000, 15)	-

3.7 Summary of this chapter

In this chapter, the relevant research content of small object detection algorithm based on YOLOv5 improvement is systematically introduced, including research

background, improvement motivation, and improvement strategies. The article first introduces the network structure of YOLOv5, then proposes three improvement methods to optimize the Head layer and Neck structure of YOLOv5s. Finally, the overall architecture of the improved model is presented, and the parameters of each network layer are listed.

The three improvement measures proposed in this chapter are as follows:

(1) Add P2 detection heads to detect smaller targets.

(2) Add CBAM attention mechanism to the Neck network to improve the network's attention to small targets.

(3) Learn from the ideas of BiFPN network and optimize the feature fusion structure of YOLOv5s.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental environment and hyperparameter settings

The experimental environment and hyperparameter settings used in this experiment are shown in Table 4 As shown in Table 4.2.

Table 4 1 Experimental environment

Environmental configuration	Name	Value
hardware configuration	GPU	NVIDIA GeForce RTX 3080
	CPU	Intel Core i7-11800H
	Memory	16G
	Graphics memory	10G
software environment	operating system	Windows11
	Python	3.7.0
	Pytorch	1.8.0
	CUDA	11.1
	cuDNN	8.1.0

Table 4.2 Hyperparameter settings

Name	Value
image_size	640×640×3
epochs	300
batch_size	4
optimizer	SGD
lr0	0.01
lrf	0.01
momentum	0.937
weight_decay	0.0005

4.2 Datasets and Data Preprocessing

4.2.1 Dataset Introduction

To verify the detection performance of the improved model based on YOLOv5 for small targets, we trained and tested the model on the TinyPerson [61] and VisDrone2019 [65] datasets. The following is a detailed introduction to these two small target datasets.

(1) TinyPerson dataset

TinyPerson is an aerial image dataset specifically designed for human target detection, shot along the coastline to detect pedestrians on the beach for rapid maritime rescue operations. Due to the broad perspective of aerial photography, some images contain a large number of targets, and most targets have a resolution below 20 pixels. This dataset can fully reflect the model's ability to detect people in long-distance video surveillance.

This dataset contains 1610 labeled images, including 794 in the training set and 816 in the testing set. There are a total of 72651 manually annotated instances, with two categories: offshore personnel and onshore personnel, labeled as sea_ Person and Earth_ Person. Figure 4 1 (a) shows the distribution of categories on the training set. It can be observed that the label is earth_ Person has the most instances, almost all of which are sea_ Double the number of person instances. The prior box scales generated by clustering these two types of targets are shown in Figure 4 As shown in 1 (b), it can be intuitively seen from the figure that the scale of the prior box is generally very small, and there are significant differences in the aspect ratio of the characters. The position offset of the final output detection box of the network is shown in Figure 4 As shown in 1 (c), x, y, width, and height represent the center point coordinates, width, and height of the predicted box, respectively. As shown in Figure 4.1 (c), there are the most prediction boxes near the origin, indicating that the detected objects are mostly small targets.

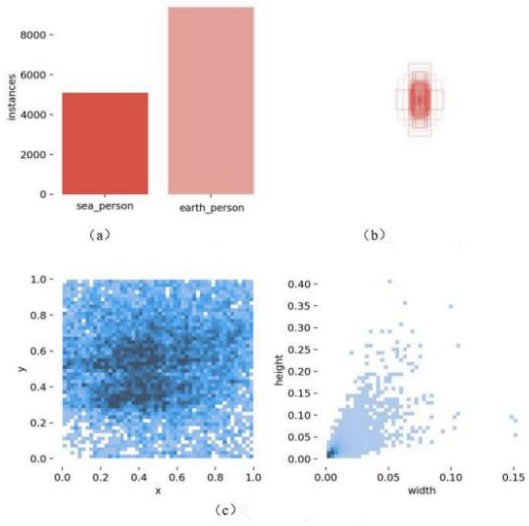


Figure 4.1 Overview of the TinyPerson dataset

In the TinyPerson dataset, there is a dense distribution of characters, diverse postures, and the presence of target occlusion and overlap. Detecting small targets in these distant and large background images is very challenging. So the model does not use images with overly dense character distributions during training, and when verifying network performance on the test set, it ignores dense crowds, uncertain areas, and reflections reflected in water.

(2) VisDrone2019 dataset

VisDrone2019 is a large image dataset collected by the AISKYEYE team at Tianjin University, collected by various models of drones. The VisDrone2019 dataset has the following characteristics: a wide range of shooting locations, covering multiple suburban areas in China; The complexity of the background varies, some are crowded, while others are sparse; The images were taken under different lighting intensities and weather conditions, with two lighting environments available: day and night. Additionally, some images were taken under cloudy and foggy conditions.

This dataset consists of 6471 training images, 1610 test images, and 548 validation images, with a total of 10 different categories of targets. As shown in Figure 4.2 (a), there is a significant difference in the number of samples of different categories on the training set, and there is a serious phenomenon of sample imbalance.

Among them, the number of cars is the highest, with a tag count far exceeding 140000; Next is Pedestrian, with a tag count of nearly 80000; However, the number of labels for tricycle and awning tricycle is less than 5000, accounting for a very small proportion. This class imbalance phenomenon increases the difficulty of classification, as the network is unable to fully learn the features of targets with small sample sizes, which can easily lead to target misdetection and affect the model's generalization ability. As shown in Figure 4.2 (b), there is a significant difference in the target scale in the image, which is due to the uncertainty of the drone's flight altitude. From the graph, it can be seen that the scale of pedestrian is relatively small, making detection more difficult. Car not only has a large number of labels, but also has a moderate target scale, making it easier to detect such targets. As shown in Figure 4.2 (c), the detection results are mostly based on small-scale detection boxes, but the overall scale of the target is larger than that of the TinyPerson dataset.

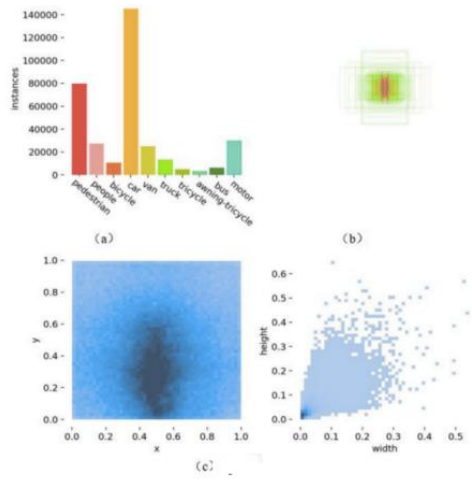


Figure 4.2 Overview of the VisDrone dataset

The VisDrone dataset has a dense distribution of targets and a large amount of overlap, with a single image containing dozens or even hundreds of targets. Moreover, when the drone is shooting downwards, the variable angle can easily cause the target to deform, making it difficult to detect small targets on this dataset. Due to the significant influence of factors such as the stability of the shooting equipment, weather conditions, and lighting changes on this dataset, only clearer images were selected for model training and testing, ignoring images with unclear targets caused

by factors such as drone oscillation, extreme weather interference, and environmental occlusion.

4.2.2 Data preprocessing

In many fields, there are problems such as limited training sets, single sample types, and high annotation costs. Due to the difficulty in obtaining sufficient training data, the trained model has poor generalization ability and performs poorly on the test set, resulting in overfitting. The best way to solve the problems that arise during the model training process mentioned above is to increase the number of training samples. It can be considered to expand the dataset from the perspective of increasing sample diversity, and through data augmentation, the computer can automatically generate a large amount of data to solve the problem of insufficient training samples and single sample types, which helps the network learn more image features and improve model performance. The basic data augmentation methods used by YOLOv5 include affine transformation and color jitter.

Affine transformation includes horizontal or vertical flipping, rotation, scaling, translation, and misalignment operations. Among them, flipping and rotating are the most common methods, both of which transform the original image in spatial position. The scaling operation includes two methods: external scaling and internal scaling. After external scaling, the image will be larger than the size of the original image. Cropping operations can be used to crop it to the original image size to prevent width and height disturbance of the original image. Shrinking processing will reduce the original image size, and it can be restored to the original image size by filling the boundaries. Translation operation refers to moving the original image along the X or Y axis without changing the image size and shape. And the wrong cutting operation not only changes the position of the image, but also changes the shape of the image. The above simple geometric transformations can change the problem of single image angles, which helps the model learn the features of images from different angles. This data augmentation method is effective for datasets with angle deviations.

Color jitter mainly enhances the color of an image, usually by adjusting the saturation, brightness, and color equality of the image to expand the dataset. The use

of color jitter operation can greatly eliminate the differences of targets in images with different backgrounds, effectively avoid interference from lighting and other factors, and improve the stability of the model.

In addition to using the basic data augmentation methods mentioned above to expand the dataset, YOLOv5 also uses the Mosaic data augmentation method, which is similar to CutMix [66]. The CutMix method randomly selects two images from a dataset, then crops one image and overlays it onto the other image, and then inputs the newly generated image into the network for training. The Mosaic method first selects four images, processes them through basic data augmentation methods such as cropping and scaling, and then randomly concatenates them into a new image. By utilizing the Mosaic data augmentation method, the network can process four image data at once during normalization, effectively improving the model speed.

The data preprocessing method adopted in the experiment is consistent with YOLOv5. Figures 4.3 and 4.4 respectively show the effect of Mosaic data augmentation on the TinyPerson and VisDrone datasets. In order to enhance the visual effect of the image, we simplified the labels. In the TinyPerson dataset, "0" and "1" are used to represent offshore personnel and beach personnel, respectively. In the VisDrone dataset, use the 10 numbers "0-9" to represent 10 different categories. From the figure, it can be seen that Mosaic data preprocessing operations can increase background complexity and help enhance the model's generalization ability.

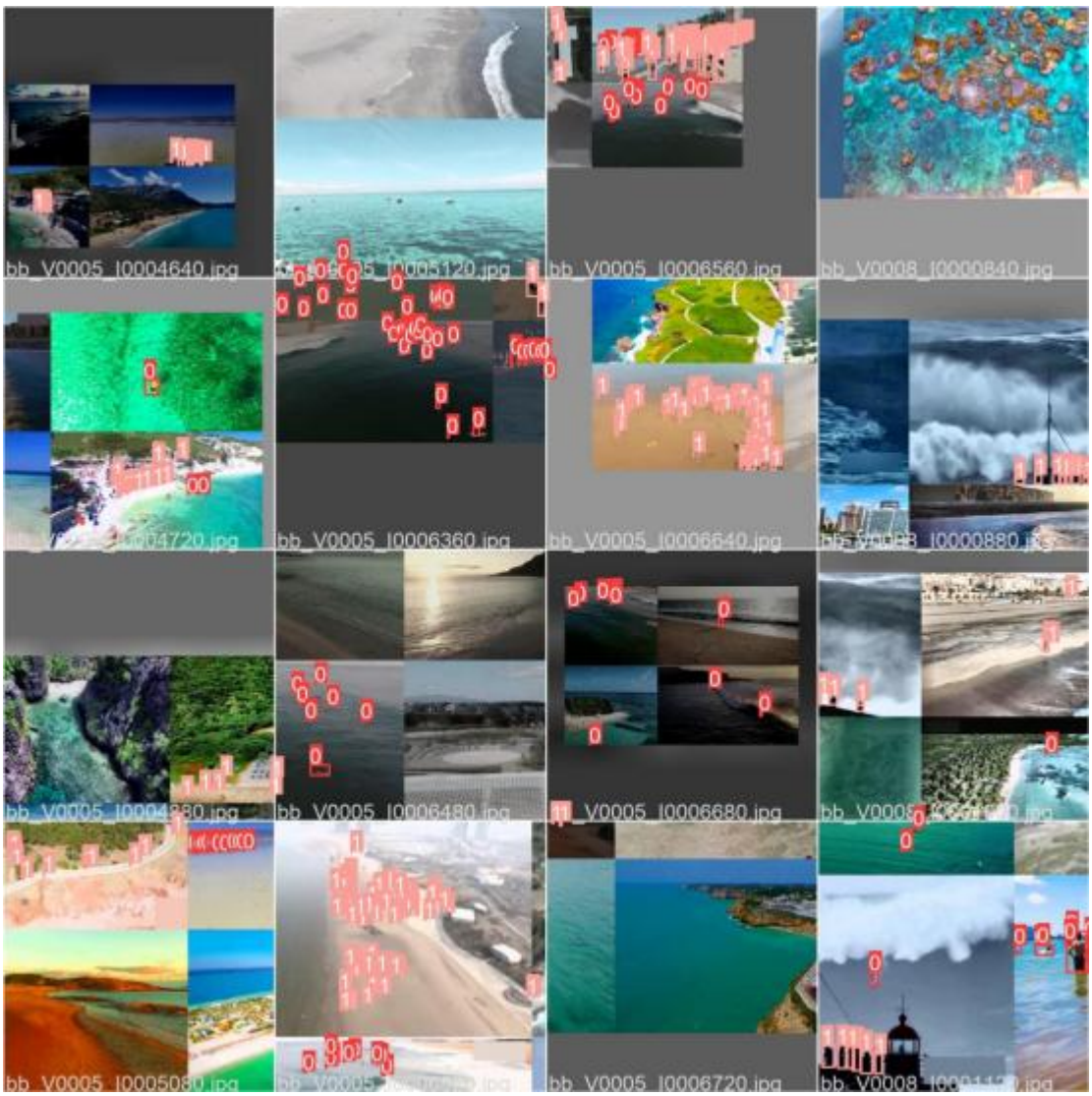


Figure 4.3 TinyPerson data augmentation effect

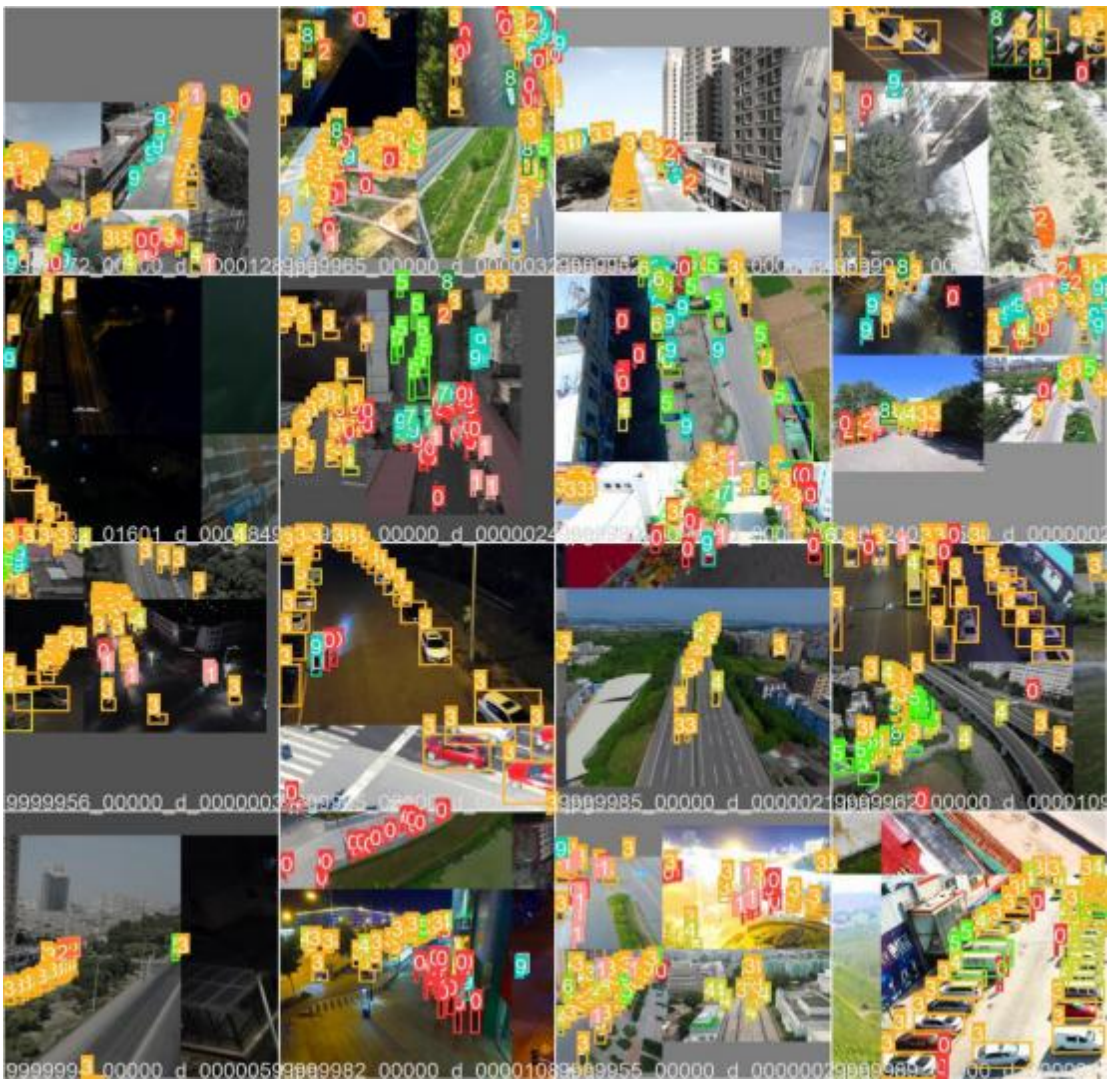


Figure 4.4 VisDrone Data Enhancement Effect

4.3 Experimental evaluation indicators

In object detection tasks, classification and bounding box regression are the standards used to measure the performance of algorithms. The correctness of the category is determined based on the confidence score of the category, and the position is determined by whether the IoU value of the predicted box and the true box is greater than the set threshold. IoU represents the degree of overlap between the predicted box and the annotated box, which is calculated using formula 4 As shown in Figure 1. Where aTea (D) represents the predicted box area, and aTea (G) represents the annotated box area.

$$IoU = \frac{area(D) \cap area(G)}{area(D) \cup area(G)} \tag{4.1}$$

The Confusion Matrix (CM) is used to analyze the classification performance of a model, from which we can intuitively see whether the samples of each category are confused and the proportion of confusion. Each row of the matrix represents the true category of the sample, and each column represents the predicted category. The larger the value on the diagonal, the higher the probability of correct classification and the better the classification performance of the model.

Taking the binary confusion matrix as an example, as shown in Table 4.3, Positive and Negative represent sample labels, with the label of the target to be detected being Positive and the background being Negative. True and False represent the predicted results, which are true when the prediction is correct and false otherwise. From this, it can be seen that TP indicates correct classification; FP represents predicting negative samples as positive examples; FN means predicting positive samples as negative samples will miss the target detection; TN represents predicting negative samples as negative examples, which is not considered in object detection tasks.

Table 4.3 Binary Confusion Matrix

predictive value / true value	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

The indicators commonly used to measure algorithm performance in object detection include accuracy, precision, recall, average precision (AP), and mean average precision (mAP). The specific meanings of the above indicators are as follows:

(1) Accuracy: Accuracy represents the proportion of correctly predicted targets to the total number of samples, as shown in formula 4.2.

$$A = \frac{TP + TN}{TP + FP + TN + FN} \qquad (4.2)$$

(2) Precision: Precision, also known as precision, represents the proportion of predicted true positive cases to all predicted positive cases. Used to measure the probability of a positive example being correctly predicted in the prediction results. The calculation method is shown in formula 4.3.

$$P = \frac{TP}{TP + FP} \qquad (4.3)$$

(3) Recall rate: Recall, also known as recall rate, represents the proportion of predicted true positive samples to the total number of actual positive samples, used to reflect missed detections. The calculation method is shown in formula 4.4.

$$R = \frac{TP}{TP + FN} \qquad (4.4)$$

(4) Mean Precision (mAP): Precision and Recall are a pair of mutually restrictive performance indicators, which have single point value limitations and cannot fully evaluate model performance. Therefore, the mAP indicator is introduced to balance the calculation results of both indicators. Using Precision as the vertical axis and Recall as the horizontal axis, the P-R curve can be obtained. The area enclosed by the P-R curve and the coordinate axis is the value of AP, while mAP represents the mean of AP for all categories in the entire dataset. Its calculation method is shown in formula 4.5. Generally, the threshold is set to 0.5, which means that prediction boxes with IoU greater than 0.5 are valid map@.5 Represent. map@.5:. 95 refers to taking the IoU from 0.5 to 0.95, calculating the mAP value every 0.05, and finally calculating the mean of all mAPs.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad (4.5)$$

4.4 TinyPerson Dataset performance evaluation

In order to improve the performance of YOLOv5 on small target datasets, we added P2 detection head and CBAM attention mechanism to the YOLOv5s model, and optimized its feature fusion structure based on BiFPN network. To verify the performance of the improved model, multiple ablation experiments and comparative

experiments were conducted on the TinyPerson dataset, and the convergence and classification performance of the model were verified. Below is a detailed introduction to the experimental content on this dataset.

4.4.1 Convergence analysis

To verify the convergence performance of the model on the TinyPerson dataset, the convergence of YOLOv5s (orange curve) and YOLOv5s P2 CBAM BiFPN (blue curve) on different metrics was compared in Figure 4.5. The loss curves including bounding box loss, confidence loss, and category loss on both the training and validation sets were compared. Additionally, precision, recall map@.5 And map@.5 Convergence curves for these four performance metrics:. 95.

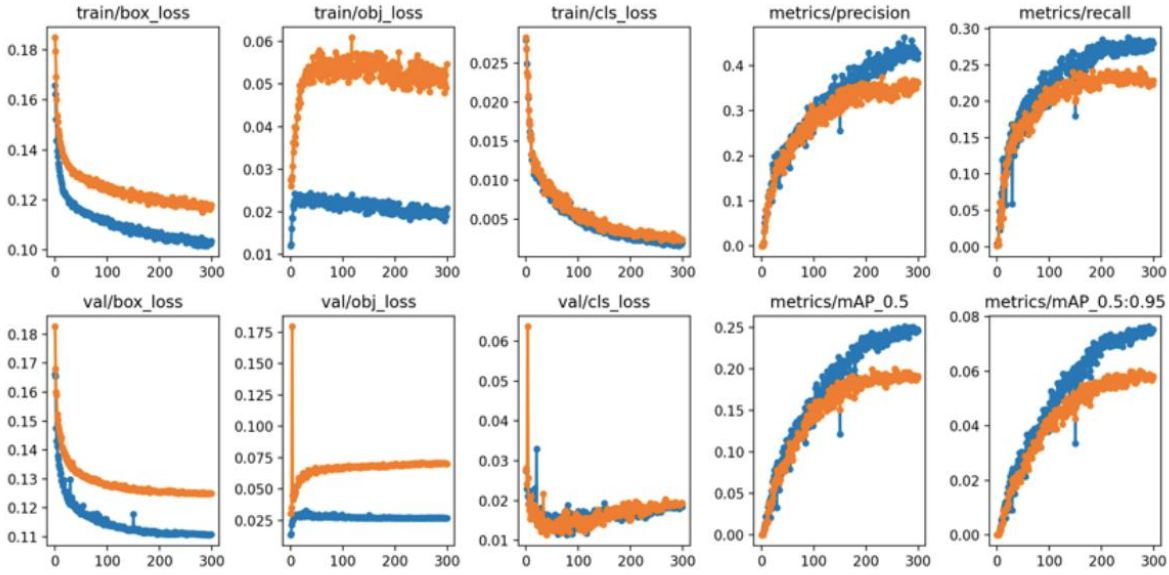


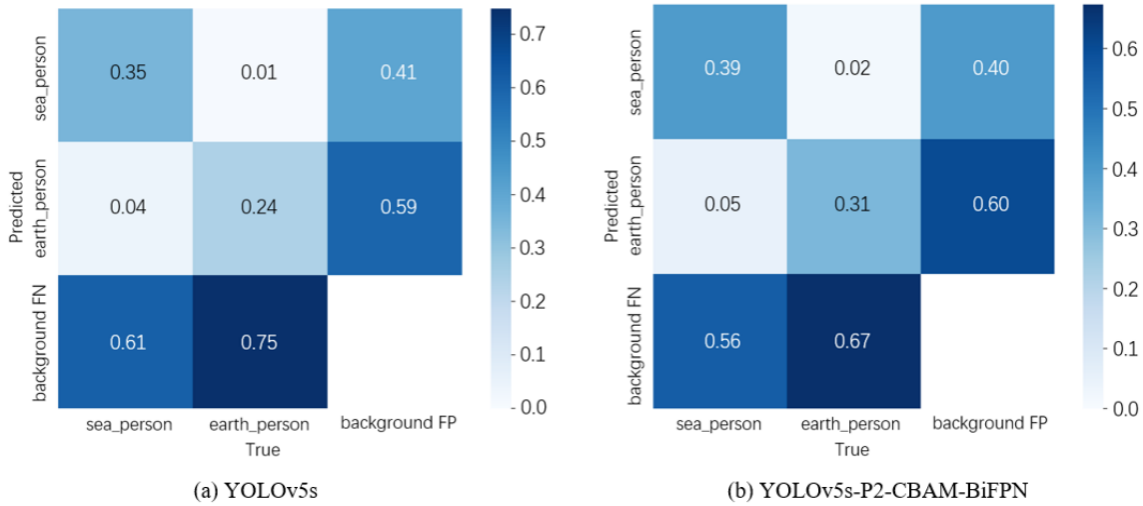
Figure 4.5 Convergence curves of different models on TinyPerson

Observing the loss curves of the two models on TinyPerson's training and validation sets, it can be seen that the loss values of the YOLOv5s-P2-CBAM-BiFPN model remain relatively low. The values of bounding box loss and confidence loss are significantly smaller than YOLOv5s; The loss curve of classification loss is basically the same as YOLOv5s, and the loss value is close to 0. According to the four

performance metric curves, YOLOv5s-P2-CBAM BiFPN has better performance, which can significantly improve the converged values of each indicator while ensuring convergence speed. Overall, YOLOv5s-P2-CBAM-BiFPN has better convergence.

4.4.2 Classification accuracy evaluation

The confusion matrices generated by YOLOv5s and YOLOv5s-P2-CBAM BiFPN models on the TinyPerson dataset are shown in Figure 4.6. As shown in Figure 4.6 (a), in the original YOLOv5s, sea_ Person and Earth_ The classification accuracy of person is 35% and 24%, respectively. Among them, sea_ Person has a 4% chance of being predicted as Earth_ The probability of being predicted as a background for a person is 61%. And Earth_ Person has a 1% chance of being classified as sea_ The probability of a person being classified as a background is as high as 75%. From this, it can be seen that the original YOLOv5s had serious missed detections, resulting in poor model performance.



(a) YOLOv5s (b) YOLOv5s-P2-CBAM-BiFPN

Figure 4.6 Confusion matrices generated by different models on TinyPerson

As shown in Figure 4.6 (b), in the YOLOv5s P2 CBAM BiFPN model, sea_ Person and Earth_ The classification accuracy of person is 39% and 31%, respectively, which is 4% and 7% higher than YOLOv5s; The probability of these

two types of targets being predicted as background is 56% and 67%, respectively, which is 5% and 8% lower than YOLOv5s; The probability of classification errors is 1% higher than YOLOv5s, but in the face of significant performance gains, this 1% error rate has little impact on model performance.

In summary, YOLOv5s-P2-CBAM-BiFPN can effectively improve classification accuracy, improve the problem of small target missed detection, and significantly improve model performance.

4.4.3 Ablation experiment

To verify the performance gains brought by the three improvement strategies of P2 detection head, CBAM module, and BiFPN network to the model, we designed six sets of ablation experiments. The experimental results on the TinyPerson test set are shown in Table 4.4. From the first three rows of the table, it can be seen that when adding CBAM module or BiFPN separately to the YOLOv5s model, its map@.5 Both have improved by less than 1%, with little improvement in network performance. After adding P2 detection head to YOLOv5s, although the accuracy was reduced, the recall rate increased by 2.7%, map@.5 Improved by 1.1%, effectively improving the target missed detection situation. From the last three rows of the table, it can be seen that adding CBAM module and BiFPN network to YOLOv5s-P2 can improve the accuracy and recall of the model, respectively. Compared to YOLOv5s P2, YOLOv5s P2 CBAM BiFPN map@.5 Improved by 1.5%, indicating that the CBAM module and BiFPN network also contribute to improving model performance. Overall, adding P2 detection head, CBAM module, and BiFPN sequentially to YOLOv5s yields the best results. Compared with the original YOLOv5s, YOLOv5s-P2-CBAM-BiFPN can increase the recall rate by 3.4% while ensuring accuracy, map@.5 Increase by 2.6%, map@.5 . 95 increased by 1.13%.

Table 4.4 Results of ablation experiments on the TinyPerson test set

Method				P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
YOLOv5s	P2	CBAM	BiFPN				
√				41.5	25.0	22.8	6.63
√		√		41.6	25.8	23.5	6.85
√			√	41.3	25.8	23.6	7.03
√	√			40.1	27.7	23.9	7.37
√	√	√		41.6	27.5	24.6	7.52
√	√	√	√	**41.7**	**28.4**	**25.4**	**7.76**

The experimental results on the TinyPerson validation set are shown in Table 4.5. Overall, all three improvement strategies can bring significant performance gains to the model, and the performance improvement on the validation set is significantly higher than that on the test set. From the table, it can be seen that compared with the original YOLOv5s, the accuracy, recall, and map@.5 And map@.5 . 95 increased by 8.2%, 4.6%, 5.1%, and 1.78%, respectively.

Table 4.5 Results of ablation experiments on TinyPerson validation set

Method				P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
YOLOv5s	P2	CBAM	BiFPN				
√				36.4	23.4	21.2	6.42
√		√		39.9	24.0	22.2	6.76
√			√	38.7	23.8	22.7	7.10
√	√			40.7	26.3	24.5	7.65
√	√	√		41.7	**28.2**	25.9	8.10
√	√	√	√	**44.6**	28.0	**26.3**	**8.20**

The results of ablation experiments on the TinyPerson dataset show that our proposed methods are effective and beneficial for improving model performance. Below is an analysis of the reasons why each improvement strategy causes changes in network performance. Adding a P2 detection head can effectively improve model performance. This is because the average absolute size of the target in the TinyPerson dataset is only 18 pixels, while the original YOLOv5 can only detect resolutions greater than 8 × The target of 8 is that the model has serious missed detections. After adding the P2 detection head, the network can detect a resolution of no less than 4 × The target of 4 has enabled many previously missed small targets to be re detected. But the feature map fused by the P2 detection head only undergoes 4 times downsampling, containing a lot of interference information. So CBAM attention mechanism was added to suppress the interference of irrelevant information and improve the network's attention to important features. The experimental results show that the CBAM module can improve model performance, indicating that the use of CBAM filters out a lot of interference information from shallow features. Subsequently, inspired by BiFPN, the

features extracted from the backbone network were fully fused with those processed by the Neck end. This approach enriches the feature information of small targets, which is beneficial for improving the network's classification and localization capabilities.

Figure 4.7 compares the detection performance of YOLOv5s, YOLOv5s P2, and YOLOv5s P2 CBAM BiFPN models on the TinyPerson dataset. It can be seen that the target scale of the TinyPerson dataset is very small, making detection difficult. However, the overall performance of the above three models on this dataset is good. Compared to others, the original YOLOv5s is better at detecting sparse targets, but its detection performance for overlapping targets is not good. None of the characters in the dense area in the middle were detected. YOLOv5s-P2 has improved the detection performance in dense areas to a certain extent, but missed some targets. The model with the best detection performance is YOLOv5s-P2 CBAM BiFPN, which can detect more smaller scale targets, effectively alleviate missed detections, and can basically correctly classify and locate targets in dense areas, outputting prediction boxes that are more suitable for the target to be detected.

Figure 4.7 Detection performance of different models on TinyPerson

4.4.4 Comparative experiment

We replicated four YOLO series algorithms on the TinyPerson dataset, namely TPH YOLOv5, YOLOv6s, YOLOv7 tiny, and YOLOX-S. The P-R curves of these four algorithms and YOLOv5s-P2 CBAM BiFPN on the TinyPerson test set are shown in Figure 4.8.

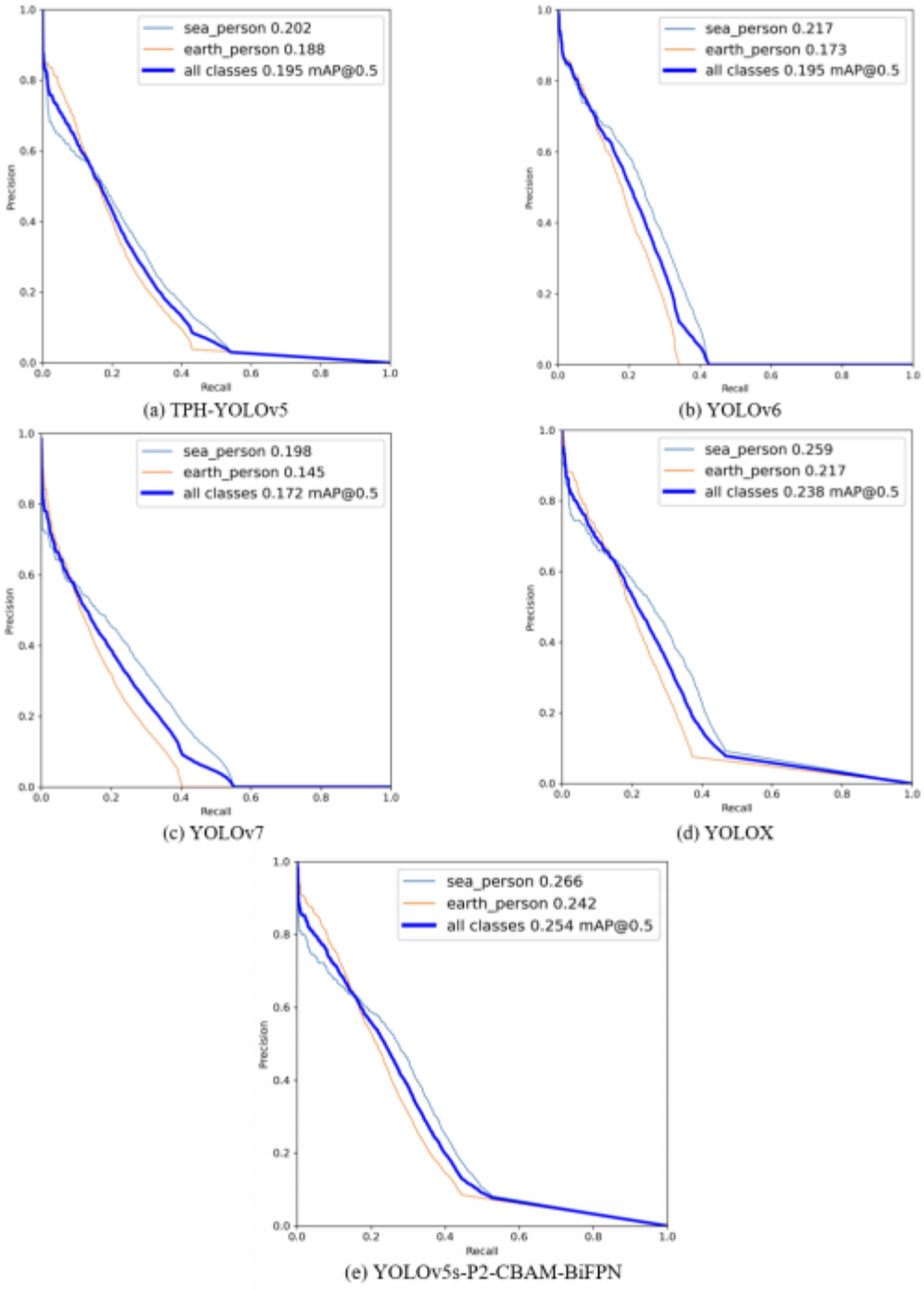


Figure 4.8 P-R curves of different models on the TinyPerson test set

It should be noted that the larger the area enclosed by the P-R curve and the coordinate axis, the higher the AP value, and the better the model performance. From Figure 4.8, it can be seen that after the recall rates of YOLOv6 and YOLOv7 reach a certain level, the accuracy of the model is 0, so the mAP value is low. Although TPH-YOLOv5 did not show an accuracy of 0, the area enclosed by the P-R curve and the

coordinate axis is relatively small. The two models with better performance are YOLOX and YOLOv5s-P2-CBAM-BiFPN. Among them, YOLOv5s-P2-CBAM-BiFPN has the largest area enclosed by the P-R curve and coordinate axis, the highest mAP value, and the best model performance.

Table 4.6 compares the performance evaluation metrics of TPH-YOLOv5, YOLOv6, YOLOv7, YOLOX, and YOLOv5s P2-CBAM BiFPN on the TinyPerson test set. Comprehensive sea_ Person and Earth_ The detection results of these two types of targets can be seen from YOLOv5s P2 CBAM BiFPN, except for sea_ The accuracy of person is lower than that of YOLOX model, and all other indicators are higher than other algorithms map@.5 Improved by 5.9% compared to TPH-YOLOv5 and YOLOv6, 8.2% compared to YOLOv7, and 1.6% compared to YOLOX; map@.5 Compared to TPH-YOLOv5, YOLOv6, YOLOv7, and YOLOX,. 95 has increased by 2.58%, 1.91%, 3.55%, and 0.88%, respectively.

Table 4.6 Performance Comparison of Different Algorithms on the TinyPerson Test Set

Method	Class	P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
TPH-YOLOv5[67]	sea_person	33.1	27.9	20.2	5.03
	earth_person	32.0	23.6	18.8	5.34
	all	32.6	25.8	19.5	5.18
YOLOv6[68]	sea_person	38.4	28.7	21.7	6.27
	earth_person	35.3	23.2	17.3	5.43
	all	36.8	26.0	19.5	5.85
YOLOv7[69]	sea_person	31.5	30.7	19.8	4.65
	earth_person	31.6	20.3	14.5	3.78
	all	31.5	25.5	17.2	4.21
YOLOX[70]	sea_person	**42.6**	30.7	25.9	7.32
	earth_person	37.5	24.6	21.7	6.44
	all	40.0	27.7	23.8	6.88
YOLOv5s-P2-CBAM-BiFPN (ours)	sea_person	40.7	**32.0**	**26.6**	**7.81**
	earth_person	**42.6**	**24.7**	**24.2**	**7.70**
	all	**41.7**	**28.4**	**25.4**	**7.76**

The comparison results on the TinyPerson validation set are shown in Table 4.7. It can be found that YOLOv5s-P2 CBAM BiFPN has the best performance, and its performance on the validation set is better than that on the test set, while the other four comparison algorithms have better performance map@.5 All are lower than the test set. The improved model map@.5 Compared to TPH YOLOv5, YOLOv6, YOLOv7, and YOLOX, it has increased by 8.8%, 8.1%, 12.1%, and 2.8% respectively; map@.5 Compared to the four YOLO series algorithms mentioned above,. 95 has improved by 3.41%, 2.72%, 4.48%, and 1.04%, respectively.

Table 4.7 Performance comparison of different algorithms on the TinyPerson validation set

Method	Class	P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
TPH-YOLOv5[67]	sea_person	31.3	23.2	17.5	4.81
	earth_person	34.4	22.0	17.4	4.77
	all	32.9	22.6	17.5	4.79
YOLOv6[68]	sea_person	38.3	28.4	20.8	6.47
	earth_person	32.1	21.7	15.5	4.48
	all	35.2	25.0	18.2	5.48
YOLOv7[69]	sea_person	28.4	23.5	15.5	4.30
	earth_person	30.5	18.2	12.9	3.14
	all	29.5	20.9	14.2	3.72
YOLOX[70]	sea_person	39.7	29.7	25.4	7.83
	earth_person	39.8	25.1	21.6	6.50
	all	39.7	27.4	23.5	7.16
YOLOv5s-P2-CBAM-BiFPN (ours)	sea_person	**42.3**	**30.1**	**27.0**	**8.56**
	earth_person	**46.8**	**25.9**	**25.6**	**7.85**
	all	**44.6**	**28.0**	**26.3**	**8.20**

The partial detection results of different algorithms on the TinyPerson dataset are shown in Figure 4.9. From the deep-sea image on the left, it can be seen that YOLOv5s-P2-CBAM-BiFPN has the best detection performance for small targets with long distances and low contrast. Almost all characters in the image are detected, which can meet the needs of real-time monitoring. From the land image on the right, it can be seen that the improved model can better classify and locate dense crowds. It can detect more small-scale characters, indicating that our proposed optimization strategy can effectively improve the small object detection performance of YOLOv5. However, other models have more serious missed detections, among which YOLOv7 has the worst performance, detecting the least number of targets.

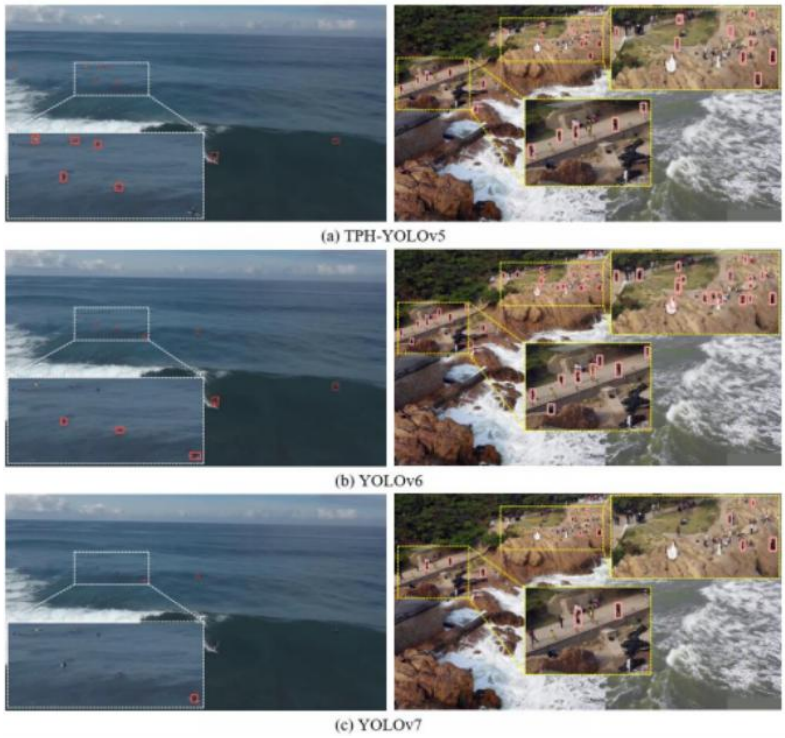


Figure 4.9 Comparison of detection results on the TinyPerson dataset

4.5 Performance evaluation of the VisDrone2019 dataset

The effectiveness of the P2 detection head, CBAM attention mechanism, and BiFPN based feature fusion network has been verified in Section 4.4. However, due to the small target scale of the TinyPerson dataset, it is difficult to significantly improve the detection performance on this dataset. In order to better validate the performance of the model, multiple sets of experiments were conducted on the unmanned aerial vehicle dataset VisDrone2019.

4.5.1 Convergence analysis

Figure 4 10 compared the convergence performance of YOLOv5s (orange curve) and YOLOv5s-P2-CBAM BiFPN (blue curve) on the VisDrone dataset. Observing the loss curves of the two models on the training and validation sets, it can be seen that the converged values of bounding box loss, confidence loss, and classification loss of YOLOv5s-P2-CBAM-BiFPN are much smaller than YOLOv5s, and are basically close to 0. The model training effect is very ideal.

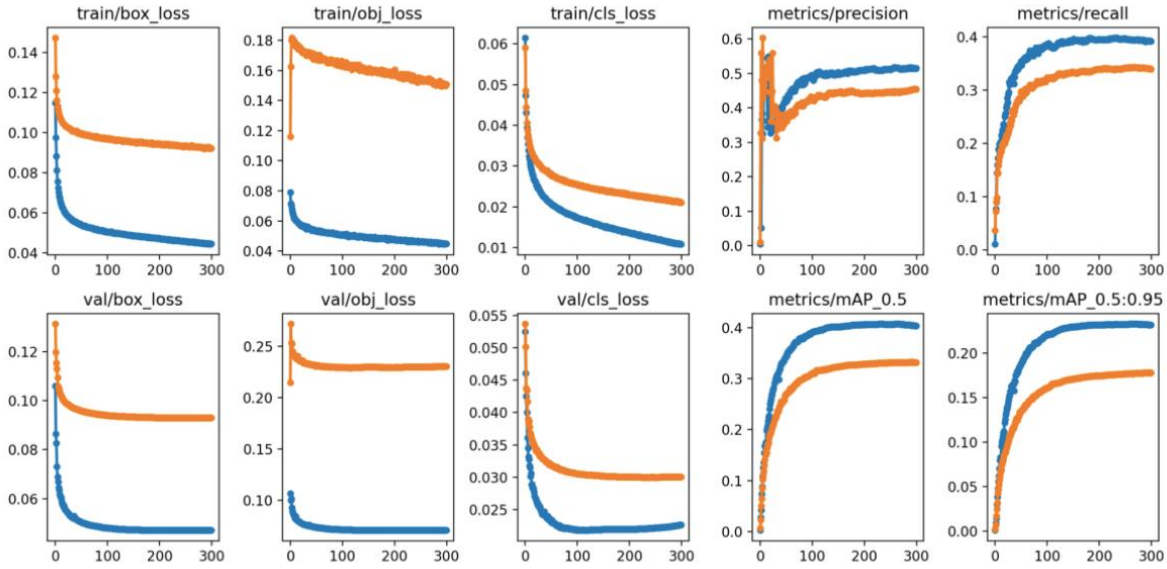


Figure 4.10 Convergence curves of different models on VisDrone

Observing the convergence curves of the performance metrics of the two models, it can be seen that the accuracy of YOLOv5s is only about 45%, while YOLOv5s P2 CBAM BiFPN has reached over 50%. At the same time, the recall rate of the improved model has increased from approximately 35% to 40%, map@.5 Approximately increased from 32% to 40%, map@.5 . 95 increased by about 6%. From this, it can be seen that the improved model converges with higher performance indicators, more stable model, and stronger predictive ability.

In summary, YOLOv5s P2 CBAM BiFPN has a faster convergence speed and better training performance than the original YOLOv5s.

4.5.2 Classification accuracy evaluation

Due to the presence of multiple categories in the VisDrone dataset, in order to better analyze the classification performance of the model, a confusion matrix was generated for the original YOLOv5s and YOLOv5s-P2-CBAM-BiFPN on this dataset, as shown in Figure 4 11 and Figure 4 As shown in Figure 12. Observing these two images can lead to the following three conclusions:

(1) Compared with YOLOv5s, the confusion rate of YOLOv5s-P2-CBAM-BiFPN has decreased, and the classification accuracy of each category has been significantly improved. Among them, bus has the largest growth rate, increasing by

11%, truck, tricycle, and motor have all increased by 9%, and earning cycle has increased by 8%; The bicycle growth rate was the smallest, only increasing by 3%; The increase in other categories is between 4% and 6%.

(2) The main reason that affects the performance of the model is the omission of many targets. For YOLOv5s-P2-CBAM-BiFPN, the probability of each category's target being predicted as the background has significantly decreased, with tricycle, warning cycle, truck, and motor experiencing the largest reductions, with reductions of 12%, 11%, 9%, and 9%, respectively; The decline in other categories is between 3% and 6%. It can be seen that the overall performance of the improved model has been greatly improved.

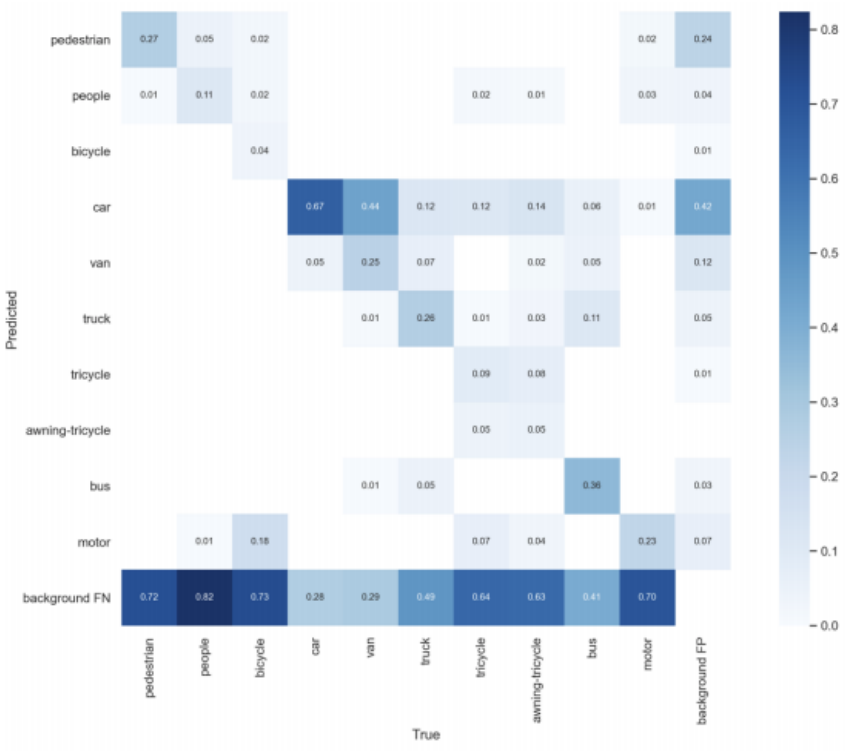


Figure 4 11 YOLOv5s generated confusion matrix on VisDrone

(2) Due to the imbalanced distribution of categories in the VisDrone dataset, the classification accuracy of each class of samples varies greatly. In the confusion matrix generated by YOLOv5s-P2-CBAM-BiFPN, the highest accuracy is car, which can reach 73%; Next is bus, with an accuracy of 47%; The lowest is cycle, only 7%.

4.5.3 Ablation experiment

The results of ablation experiments on the VisDrone test set are shown in Table 4.8. From the first three rows of the table, it can be seen that adding CBAM module or BiFPN network separately on the basis of YOLOv5s can improve recall, but the accuracy has decreased. This approach does not significantly improve the overall performance of the model, which is consistent with the TinyPerson dataset.

From the fourth row of Table 4.8, it can be seen that after adding the P2 detection head, the accuracy of the model has been improved by 3.3% compared to the original YOLOv5s, and the recall rate is map@.5 Both increased by 4.6%, map@.5 . 95 increased by 3%. From this, it can be seen that adding a P2 detection head can significantly improve model performance, and the improvement effect is even more significant than on the TinyPerson dataset. This is because there are many resolutions below 4 in the TinyPerson dataset × The small target of 4 was missed by the network due to its low resolution, which affected the overall performance of the algorithm. And VisDrone has a slightly larger target scale than the TinyPerson dataset, so the number of targets that the network can detect increases, and the performance indicators also improve accordingly.

According to the fourth and fifth rows of Table 4.8, adding the CBAM module to the P2 detection head can effectively improve the accuracy of the model. From the fourth and sixth lines, it can be seen that YOLOv5s P2 CBAM BiFPN is more efficient than YOLOv5s P2 map@.5 Improved by 1.1%, map@.5 . 95 increased by 0.8%. It can be seen that the combination of CBAM module and BiFPN network has a better effect.

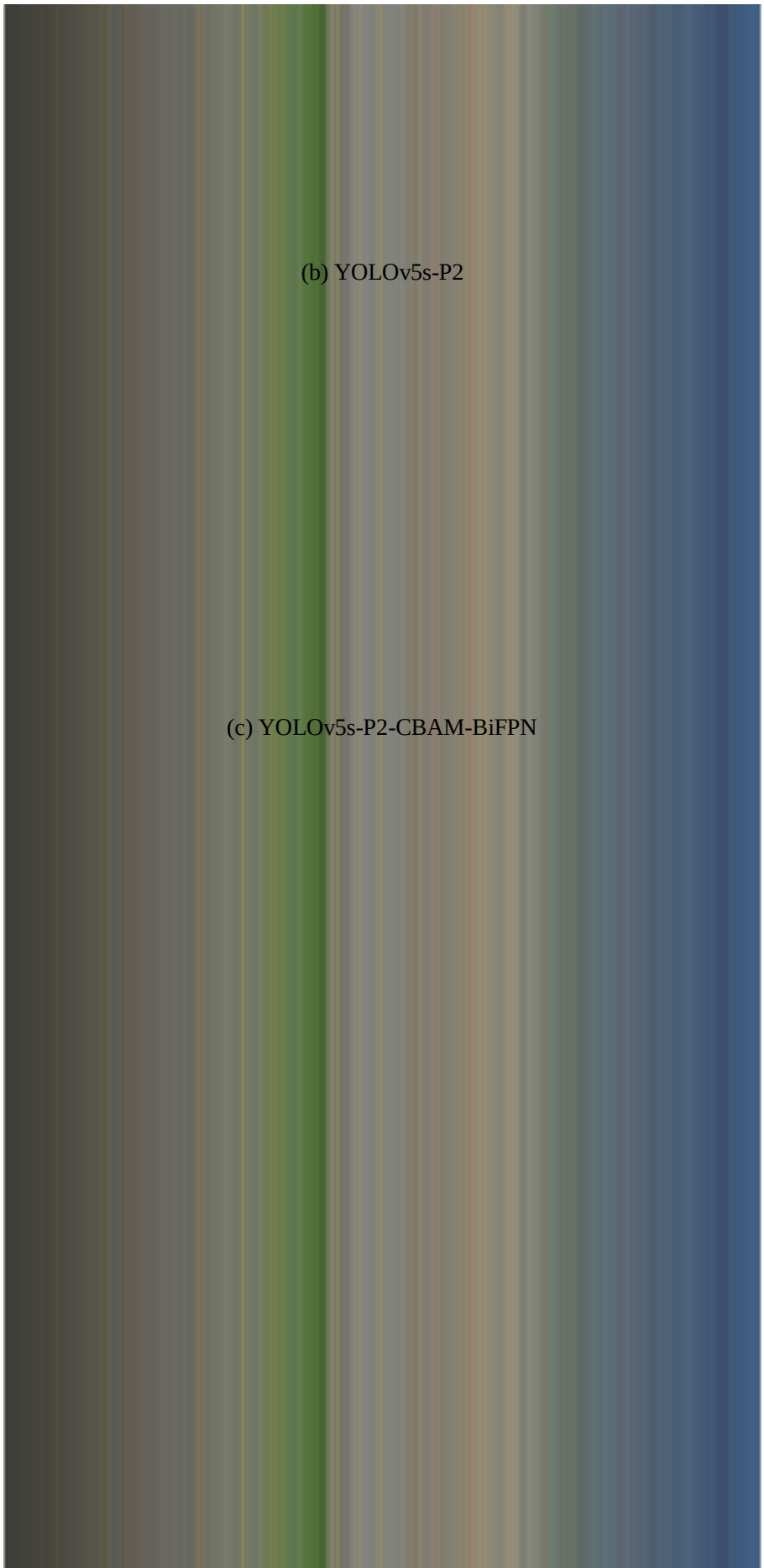
Table 4.8 Results of VisDrone Test Set Ablation Experiment

Method				P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
YOLOv5s	P2	CBAM	BiFPN				
√				39.1	30.0	27.7	14.7
√		√		38.6	31.0	28.0	14.9
√			√	38.3	31.3	28.3	14.9
√	√			42.4	34.6	32.3	17.7
√	√	√		**44.3**	34.7	32.9	18.1
√	√	√	√	44.2	**35.0**	**33.4**	**18.5**

Based on the results of ablation experiments on the TinyPerson and VisDrone datasets, it can be concluded that although there are significant differences between the two datasets, the overall performance of the proposed improvement measures tends to be consistent. The basic rule is as follows: (1) P2 detection head can effectively improve model performance. (2) The combination of CBAM and BiFPN has a more significant effect. (3) The YOLOv5s-P2-CBAM-BiFPN model has the best performance, as it performs well on the VisDrone test set map@.5 Improved by 5.7% compared to YOLOv5s, map@.5 . 95 is 3.8% higher than YOLOv5s.

The partial detection results of YOLOv5s, YOLOv5s P2, and YOLOv5s P2 CBAM BiFPN on the VisDrone dataset are shown in Figure 4.13. By comparison, it can be seen that YOLOv5s performs poorly in detecting targets such as smaller pedestrians and distant cars, and misses a large number of small targets. YOLOv5s-P2 improves this phenomenon, and the network can detect some small targets that were previously missed. The best performance is the YOLOv5s P2-CBAM BiFPN model, which has more accurate border positioning, detects more small targets, and can basically correctly classify and recognize overlapping and low contrast targets. This model has stronger detection ability for small targets.

(a) YOLOv5s



(b) YOLOv5s-P2

(c) YOLOv5s-P2-CBAM-BiFPN

Figure 4.13 Detection performance of different models on VisDrone

4.5.4 Comparative experiment

The lightweight versions of TPH-YOLOv5, YOLOv6, YOLOv7, and YOLOX models were also replicated on the VisDrone dataset. The comparison results between these four models and YOLOv5s-P2-CBAM-BiFPN on the VisDrone 2019 validation set are shown in Table 4.9. According to the data in the table, it can be seen that YOLOv5s P2-CBAM BiFPN has higher accuracy than other YOLO series algorithms, and it has more advantages in detecting small targets than other models.

Table 4.9 Performance comparison of different algorithms on the VisDrone validation set

Method	P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)
TPH-YOLOv5[67]	46.0	36.0	35.7	19.1
YOLOv6[68]	47.0	**40.1**	37.4	21.9
YOLOv7[69]	44.1	37.7	34.6	18.0
YOLOX[70]	49.8	40.0	39.7	22.1
YOLOv5s-P2-CBAM-BiFPN (ours)	**51.8**	39.4	**41.2**	**23.7**

The performance comparison of the above five models on the VisDrone2019 test set is shown in Table 4 As shown in 10. From the table, it can be seen that the overall performance of the improved model is superior to other algorithms. Compared with TPH-YOLOv5, YOLOv6, YOLOv7, and YOLOX, YOLOv5s P2 CBAM BiFPN has better performance map@.5 Increased by 4.3%, 1.7%, 3.8%, and 1.9% respectively; map@.5 . 95 is 3.4%, 0.1%, 3.6%, and 1.6% higher than the other four YOLO series models, respectively.

Table 4.10 Performance Comparison of Different Algorithms on the VisDrone Test Set

Method	P(%)	R(%)	mAP@.5(%)	mAP@.5:.95(%)

TPH-YOLOv5[67]	41.9	30.6	29.1	15.1
YOLOv6[68]	**45.1**	**35.1**	31.7	18.4
YOLOv7[69]	41.3	33.6	29.6	14.9
YOLOX[70]	44.2	34.0	31.5	16.9
YOLOv5s-P2-CBAM-BiFPN (ours)	44.2	35.0	**33.4**	**18.5**

Figure 4.14 shows the P-R curves of TPH-YOLOv5, YOLOv6, YOLOv7, YOLOX, and the improved model on the VisDrone test set, respectively. The overall performance of each model and its detection performance for each type of target can be intuitively seen from the figure. By comparison, it can be seen that the model with the best performance is based on the improved YOLOv5s, which has the largest area enclosed by the P-R curve and coordinate axis. From Figures 4.14 (b) and 4.14 (c), it can be seen that when the recall rate reaches approximately 80%, if its value is further increased, the accuracy of YOLOv6 and YOLOv7 will both decrease to 0. A similar situation has also occurred on the TinyPerson dataset, which has a negative impact on the overall performance of the model. So in Table 4 In the comparison experiment results of 10, YOLOv6 had the highest accuracy and recall, but its mAP value was lower than YOLOv5s-P2-CBAM BiFPN.

4.6 Summary of this chapter

This chapter first explains the experimental environment, dataset used, and performance evaluation indicators, and then focuses on experimental verification of the three improvement measures based on the YOLOv5s model proposed in Chapter 3. The results of the ablation experiment indicate that all improvement measures can effectively improve the performance of the model. On the TinyPerson test set, YOLOv5s P2 CBAM BiFPN's map@.5 And map@.5 . 95 is 2.6% and 1.13% higher than YOLOv5s, respectively. On the VisDrone2019 test set, its map@.5 And map@.5 . 95 is 5.7% and 3.8% higher than YOLOv5s.

In the horizontal comparison experiment, YOLOv5s-P2-CBAM-BiFPN was compared with TPH-YOLOv5, YOLOv6s, YOLOv7 tiny, and YOLOX-S. The experimental results indicate that YOLOv5s-P2 CBAM BiFPN has better detection performance for small targets. On the TinyPerson test set, its map@.5 Compared to

the four YOLO series algorithms mentioned above, it is 5.9%, 5.9%, 8.2%, and 1.6% higher; map@.5 . 95 increased by 2.58%, 1.91%, 3.55%, and 0.88% respectively. On the VisDrone2019 test set, YOLOv5s - P2-CBAM BiFPN outperforms the four comparison algorithms mentioned above map@.5 Higher by 4.3%, 1.7%, 3.8%, and 1.9% respectively; map@.5 . 95 increased by 3.4%, 0.1%, 3.6%, and 1.6% respectively.

5 SUMMARY AND OUTLOOK

5.1 Work Summary

There are many difficulties in the topic of small object detection, which requires high algorithm performance. On the one hand, it is because the target resolution is generally very small at long distances and in large backgrounds, and the network can extract fewer effective features of the target to be detected, leading to serious missed detections. On the other hand, due to the presence of a large number of similar features in images, false positives are prone to occur, and the interference of noise and complex backgrounds further increases the difficulty of detection. In order to improve the performance of YOLOv5 algorithm on small target datasets, we optimized the network structure of YOLOv5s from three aspects. The main innovative points are as follows:

(1) Transform three-scale detection into four-scale detection. The feature map of the original YOLOv5 model used for small object detection has a large receptive field, which is prone to losing the position information of small objects, resulting in a large number of small object missed detections. To improve this issue, a 4x downsampling detection head was added, reducing the local receptive field of the feature map used for small object detection, allowing the network to detect smaller targets. The YOLOv5s-P2 model integrates larger scale feature maps, which can transmit more low-level geometric information from bottom to top, improving the expression ability of feature maps and making it more conducive to predicting the position of small targets. Moreover, working together with the four detection heads can improve the multi-scale detection performance of the model. The experimental results indicate that on the TinyPerson test set, YOLOv5s-P2 exhibits map@.5 1.1% higher than YOLOv5s; On the VisDrone2019 test set, it is 4.6% higher than YOLOv5s.

(2) Increase CBAM attention mechanism. In order to suppress the interference of irrelevant information and enhance the model's attention to small targets, the CBAM attention mechanism was introduced into the Neck network of YOLOv5s. Adding multiple CBAM modules between two feature fusions allows the network to

focus more attention on important channels and key regions of the feature map when processing a large amount of feature information, enhancing the expression ability of key features in the feature map and improving the accuracy of small target localization. The experimental results indicate that on the TinyPerson test set, YOLOv5s P2 CBAM exhibits map@.5 1.8% higher than YOLOv5s; On the VisDrone2019 test set, it was 5.2% higher than YOLOv5s.

(3) A feature fusion structure based on BiFPN. Due to the important role played by the Neck network in feature transmission and processing, in order to further improve the feature processing ability of the Neck network, the idea of bi-directional cross scale connections in the BiFPN model was borrowed to optimize the feature fusion structure of YOLOv5s. The improved network adds two feature fusion paths on the basis of the original FPN+PAN structure, which are used to fully integrate the features extracted by the backbone network with the features on the Neck end. This feature processing method enhances the information transfer between feature maps at different scales, which is beneficial for improving the classification and localization capabilities of the model. The experimental results indicate that on the TinyPerson test set, YOLOv5s P2 CBAM BiFPN exhibits map@.5 2.6% higher than YOLOv5s; On the VisDrone2019 test set, it is 5.7% higher than YOLOv5s.

The improved model achieved good detection performance on both the TinyPerson and VisDrone2019 datasets, proving that our proposed improvement scheme is beneficial for enhancing the small object detection capability of YOLOv5.

5.2 Prospect

By optimizing the network structure of YOLOv5s, the model's small object detection capability has been effectively improved.

In order to further improve algorithm performance, future research directions can be divided into the following three aspects:

(1) The main reason for the serious phenomenon of missed and false detection of targets is that the resolution of small targets is too small and the effective features are insufficient. Therefore, it is possible to consider using high-resolution samples to train the model and fundamentally solve small targets

The problem of insufficient information.

(2) There is a significant amount of occlusion in natural scene images, especially in small target datasets where objects are densely distributed and overlap is more severe. Improving the detection performance of algorithms for occluded targets has practical significance and is the future direction

Resolved problems.

(3) Due to YOLOv5's extremely fast speed and small model size, it is easy to deploy. So the improved model can be further lightweight processed and deployed on hardware devices for real-time object detection.

REFERENCES

[1] Neubeck A, Van Gool L. Efficient non-maximum suppression[C]. 18th International Conference on Pattern Recognition (ICPR'06). IEEE, 2006, 3: 850-855.

[2] JaegerPF, Kohl SAA, BickelhauptS, et al. Retina U-Net: Embarrassingly simple exploitation of segmentation supervision for medical object detection[C]. Machine Learning for Health Workshop. PMLR, 2020: 171- 183.

[3] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Ieee, 2005, 1: 886-893.

[4] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91- 110.

[5] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[C]. Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. Ieee, 2001, 1: I-I.

[6] Ojala T, Pietikainen M, Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns[J]. IEEE Transactions on pattern analysis and machine intelligence, 2002, 24(7): 971-987.

[7] Cortes C, Vapnik V. Support-vector networks[J]. Machine learning, 1995, 20(3): 273-297.

[8] Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an

application to boosting[J]. Journal of computer and system sciences, 1997, 55(1): 119-139.

[9] Felzenszwalb P, McAllester D, Ramanan D. A discriminatively trained, multiscale,

deformable part model[C]. 2008 IEEE conference on computer vision and pattern recognition. Ieee, 2008: 1-8.

[10] Felzenszwalb PF, Girshick RB, McAllester D. Cascade object detection with deformable part models[C]. 2010 IEEE Computer society conference on computer vision and pattern recognition. Ieee, 2010: 2241-2248.

[11] Felzenszwalb P F, Girshick R B, McAllester D, et al. Object detection with discriminatively trained part-based models[J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 32(9): 1627- 1645.

[12] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[13] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[14] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[15] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]. International conference on machine learning. PMLR, 2015: 448- 456.

[16] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.

[17] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]. Thirty-first AAAI conference on artificial intelligence. 2017.

[18] Chollet F. Xception: Deep learning with depthwise separable convolutions[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1251- 1258.

[19] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[20] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700- 4708.

[21] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.

[22] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[23] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510- 4520.

[24] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3[C]. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314- 1324.

[25] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.

[26] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]. Proceedings of the European conference on computer vision (ECCV). 2018: 116- 131.

[27] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

[28] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

[29] He K, Zhang X, RenS, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9): 1904- 1916.

[30] Girshick R. Fast r-cnn[C]. Proceedings of the IEEE international conference on computer vision. 2015: 1440- 1448.

[31] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28.

[32] Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[J]. Advances in neural information processing systems, 2016, 29.

[33] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431- 3440.

[34] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]. Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.

[35] Cai Z, Vasconcelos N. Cascade r-cnn: Delving into high quality object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6154- 6162.

[36] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[37] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.

[38] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[39] Bochkovskiy A, Wang C Y, Liao H Y M. Yolov4: Optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.

[40] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117- 2125.

[41] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.

[42] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]. European conference on computer vision. Springer, Cham, 2016: 21-37.

[43] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]. Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.

[44] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]. International conference on machine learning. PMLR, 2019: 6105-6114.

[45] Tan M, Pang R, Le Q V. Efficientdet: Scalable and efficient object detection[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 10781- 10790.

[46] Kong T, Yao A, Chen Y, et al. Hypernet: Towards accurate region proposal generation and joint object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 845-853.

[47] Law H, Deng J. Cornernet: Detecting objects as paired keypoints[C]. Proceedings of the European conference on computer vision (ECCV). 2018: 734-750.

[48] Duan K, Bai S, XieL, et al. Centernet: Keypoint triplets for object detection[C]. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 6569-6578.

[49] Zhou X, Wang D, KrähenbühlP. Objects as points[J]. arXiv preprint arXiv:1904.07850, 2019. [50] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Networks[J].

Advances in Neural Information Processing Systems, 2014, 3:2672-2680.

[51] Li J, Liang X, Wei Y, et al. Perceptual generative adversarial networks for small object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1222- 1230.

[52] Bai Y, Zhang Y, Ding M, et al. Sod-mtgan: Small object detection via multi-task generative adversarial network[C]. Proceedings of the European Conference on Computer Vision (ECCV). 2018: 206-221.

[53] BottouL. Stochastic gradient descent tricks[M]. Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012: 421-436.

[54] Qian N. On the momentum term in gradient descent learning algorithms[J]. Neural networks, 1999, 12(1): 145- 151.

[55] DuchiJ, HazanE, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of machine learning research, 2011, 12(7).

[56] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[57] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks[C]. Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2011: 315-323.

[58] Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve neural network acoustic models[C]. Proc. icml. 2013, 30(1): 3.

[59] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]. European conference on computer vision. Springer, Cham, 2014: 818-833.

[60] Wang C Y, Liao H Y M, Wu Y H, et al. CSPNet: A new backbone that can enhance learning capability of CNN[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 390-391.

[61] Yu X, Gong Y, Jiang N, et al. Scale match for tiny person detection[C]. Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2020: 1257- 1265.

[62] Laskar Z, Kannala J. Context aware query image representation for particular object retrieval[C]. Scandinavian Conference on Image Analysis. Springer, Cham, 2017: 88-99.

[63] Park J, Woo S, Lee J Y, et al. Bam: Bottleneck attention module[J]. arXiv preprint arXiv:1807.06514, 2018.

[64] Woo S, Park J, Lee J Y, et al. Cbam: Convolutional block attention module[C]. Proceedings of the European conference on computer vision (ECCV). 2018: 3- 19.

[65] Zhu P, Wen L, Du D, et al. Vision meets drones: Past, present and future[J]. arXiv preprintarXiv:2001.06303, 2020.

[66] Yun S, Han D, Oh S J, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features[C]. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 6023-6032.

[67] Zhu X, Lyu S, Wang X, et al. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 2778-2788.

[68] Li C, Li L, Jiang H, et al. YOLOv6: a single-stage object detection framework for industrial applications[J]. arXiv preprint arXiv:2209.02976, 2022.

[69] Wang C Y, BochkovskiyA, Liao H Y M. YOLOv7: Trainable bag-of-freebies sets new state- of-the-art for real-time object detectors[J]. arXiv preprint arXiv:2207.02696, 2022.

[70] Ge Z, Liu S, Wang F, et al. Yolox: Exceeding yolo series in 2021[J]. arXiv preprint arXiv:2107.08430, 2021.

APPENDIX A

KEY FRAGMENTS OF THE SOURCE CODE OF THE PROJECT

Data models:

```
import torch
import torch.nn as nn

class YOLOv5s_P2(nn.Module):
    def __init__(self, num_classes=80):
        super().__init__()


        self.backbone = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),

            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Conv2d(128, 256, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 512, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(512),
            nn.ReLU(),
        )


        self.neck = nn.Sequential(
            nn.Conv2d(512, 256, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 128, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(128, 256, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(256),
            nn.ReLU(),
```

```
            nn.Conv2d(256, 512, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(512),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, 256, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 128, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(128, 256, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 512, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(512),
            nn.ReLU(),
        )

          self.detection_head = nn.Sequential(
            nn.Conv2d(512, 256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 128, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 512, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(512),
            nn.ReLU(),

            nn.Conv2d(512, num_classes, kernel_size=1,
stride=1),#!/usr/bin/env python

            nn.ReLU(),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
```

```
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),
        )

    def forward(self, x):
        x = self.backbone(x)

        x = self.neck(x)

         x = self.detection_head(x)

        return x


import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms

class YOLOv5s_P2_CBAM_BiFPN(nn.Module):
    def __init__(self, num_classes=80):
        super().__init__()

        self.backbone = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
```

```
            nn.BatchNorm2d(64),
            nn.ReLU(),

            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Conv2d(128, 256, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 512, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(512),
            nn.ReLU(),
        )
        self.neck = BiFPN([512, 256, 128, 64], num_repeats=3)

        self.detection_head = nn.Sequential(
            nn.Conv2d(512, 256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 128, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),

            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),

            nn.Conv2d(256, 512, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(512),
            nn.ReLU(),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
```

```
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),

            nn.Upsample(scale_factor=2, mode='nearest'),

            nn.Conv2d(512, num_classes, kernel_size=1, stride=1,
padding=0),
            nn.BatchNorm2d(num_classes),
            nn.ReLU(),
        )

        self.cbam = CBAM(512)

    def forward(self, x):
        x = self.backbone(x)
        x = self.neck(x)

        x = self.cbam(x)
        x = self.detection_head(x)

        return x


import torch
from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, non_max_suppression,
scale_coords
from utils.torch_utils import select_device, time_synchronized


weights = r'D:\yolov5\train\exp\weights\best.pt'
source = '0'
imgsz = 640
conf_thres = 0.25
iou_thres = 0.45
device = ''

device = select_device(device)

model = attempt_load(weights, map_location=device)
imgsz = check_img_size(imgsz, s=model.stride.max())
if torch.cuda.is_available():
    model.half()


dataset = LoadStreams(source, img_size=imgsz,
stride=model.stride.max())
```

```
names = model.module.names if hasattr(model, 'module') else
model.names


model.eval()
for path, img, im0s, vid_cap in dataset:
    img = torch.from_numpy(img).to(device)
    img = img.half() if model.dtype is torch.half else img.float()
img /= 255.0
    if img.ndimension() == 3:
        img = img.unsqueeze(0)

    pred = model(img, augment=False, visualize=False)[0]

    pred = non_max_suppression(pred, conf_thres, iou_thres)


    for i, det in enumerate(pred):
        if len(det):

            det[:, :4] = scale_coords(img.shape[2:], det[:, :4],
im0s.shape).round()

            for *xyxy, conf, cls in reversed(det):
                label = f'{names[int(cls)]} {conf:.2f}'
                print(label, xyxy)
```