Improved faster and more accurate detection network for pedestrian detection based on YOLOv5s

Student Name: Jun Gao (BSc Computer Science FT)

Student ID:2198815

Supervisor: Jinming duan

Worlds:6031

Contents

Abstract:

Pedestrian detection is a key component of many real-world applications. Pedestrian detection has a wide range of applications in autonomous driving, real-time surveillance systems and traffic systems. And in these applications, improving the recognition speed and accuracy of pedestrian detection plays a crucial role in ensuring pedestrian safety and avoiding traffic accidents. At the same time, pedestrian detection is a challenging task that has been extensively studied in the field of computer vision. Excellent pedestrian detection algorithms/frameworks need to overcome several challenges, such as dealing with variations in pedestrian appearance, dense crowd occlusion and cluttered scenes. These challenges make pedestrian detection an interesting and important research problem in computer vision. In these applications, accurate and efficient pedestrian detection is crucial to ensure pedestrian safety and accident avoidance. Traditional two-stage algorithms such as fast-RCNN RCNN have good accuracy but are not as fast as one-stage algorithms such as YOLO YOLOv5 is one of the most advanced target detection algorithms with good performance in terms of recognition speed and accuracy. In this paper, I propose four ways to improve YOLOv5s detection network for the key problem of pedestrian detection, namely to improve detection speed while minimising the impact of the improved method on detection accuracy, to improve detection accuracy while minimising the impact of the improved method on detection speed, and to maximise either detection accuracy or detection speed.

1.Introduction

There are two main types of algorithms used for pedestrian detection: one-stage pedestrian detection and two-stage pedestrian detection. The main differences lie in the structure of the algorithm itself, the neural network architecture on board and the speed at which the algorithm runs.

Two-stage pedestrian detection pedestrian detection divides the target detection into two steps: firstly, the extraction of areas in the image where pedestrians are likely to be present (i.e. candidate areas), and secondly, the classification and regression of these candidate areas. Commonly used algorithms for two-stage pedestrian detection include RCNN[1], Fast R-CNN[2], Faster R-CNN[3] and so on. Although these algorithms have higher detection accuracy, they are not as fast as one-stage pedestrian detection.

One-stage pedestrian detection (single-stage detection) refers to combining the classification and regression tasks of the target detection process into one, predicting the class (pedestrian) and location of the target directly from the input image or video. Typical one-stage pedestrian detection algorithms include YOLO[4], SSD[5], etc. Compared to two-stage pedestrian detection, one-stage pedestrian detection algorithms are simple in structure and fast in operation, and are more suitable for pedestrian detection in real-time scenarios.

The backbone of YOLOv3 and YOLOv4 was implemented by Joseph Redmon et al. in the Darknet framework with the Darknet-53 [6]convolutional neural network (CNN) architecture.In contrast, YOLOv5 uses a new backbone network, the CSPNet (Cross-Stage Partial Network).The core idea of the CSPNet is to divide the neural network into two stages: feature extraction and feature fusion. This architecture can effectively reduce the computational cost of the model and improve the accuracy and stability of the model.

Although the backbone network of YOLOv5 is different from Darknet-53, YOLOv5 still uses many of the same techniques and methods as the Darknet framework, such as: Residual block, route layer, shortcut layer, SPP layer, etc. Therefore, the relationship between YOLOv5 and Darknet-53 can be seen as being based on the same techniques and methods, but with different neural network architectures.

YOLOv5 uses a deeper neural network architecture than Darknet, and its backbone network is CSPNet rather than Darknet-53. In addition, YOLOv5 also uses other new techniques such as Swish activation function (SiLU[7]) and DropBlock regularisation. Thus, although YOLOv5 is based on Darknet, many improvements and optimisations have been made in the implementation to make it a more advanced and efficient target detection algorithm.

Nevertheless, YOLOv5 still has some room for improvement in the specific area of pedestrian detection. For the need to improve the speed of the algorithm, I have replaced part of the C3 module of YOLOv5s backbone with CondConv[7], a single convolutional module with strong feature extraction capabilities, to improve the FPS by 20.52% while maintaining the original mAP and to increase the For the direction of improving the accuracy of the model in pedestrian detection I improved the mAP from 87.6% to 88.4% by using Involution[8] and CondConv in the corresponding feature map to feature extraction layers of the neck, respectively, with little reduction in FP S. 88.4%.

2 Related works

There are two main different improvement schemes in the target detection project, improving the detection accuracy of the detection network as well as improving the detection speed of the detection network.

The main common solutions for improving the detection accuracy of deep neural networks are: data augmentation: by performing some random transformations on the input data, such as rotation, scaling, flipping, cropping, etc., the training dataset can be expanded, thus improving the generalization ability and robustness of the model. Model fusion: fusing the output of multiple models can improve the accuracy and robustness of the model. Common model fusion methods include voting, weighted averaging, stacking, etc. Attention mechanism[8]: Introducing an attention mechanism into a deep neural network can help the model to better focus on the important features, thus improving detection accuracy. Network structure optimisation: Improving the network structure can be effective in improving detection accuracy. For example, increasing the network depth, width, using more convolutional kernels and so on.

There are many papers that focus on the use of dynamic convolution in the convolutional layer as well as attention mechanisms to enhance the feature extraction capability of individual convolutional layers[9;10]. It is a mechanism for capturing local and global information in the input sequence. Dynamic convolutional attention is added before the convolutional layer in order to adjust the weights of the convolutional kernel according to the contextual information of the input sequence before each convolutional operation. This mechanism allows the network to assign different importance to different parts of the input sequence and to dynamically adapt to changes in the input sequence at different time steps.,This improvement enhances the detection network in three main ways. In terms of spatial adaptation, dynamic convolution is able to dynamically adjust the size and shape of the convolution kernel according to the different shapes of the input data, thus adapting to different sizes of input data and improving the adaptability and generalisation ability of the model; in terms of network depth reduction, dynamic convolution introduces a recursive structure in the network, which is able to achieve a function similar to pooling operations, and thus can reduce the network depth and reduce the computational In terms of effect enhancement, dynamic convolution can improve the effectiveness of the model for detection, recognition, segmentation and other tasks due to its spatial adaptability and its ability to handle local features in the input data in an adaptive manner.

Several papers on YOLOx[12] have also mentioned the improvement of detection networks by using the Decoupled Head mechanism [12;13] to obtain the following advantages: higher accuracy: the Decoupled Head mechanism helps the model to predict targets of different sizes and aspect ratios more accurately, as each head is focused on processing targets within a certain range. By using decoupled heads separately on feature maps of different resolutions, the model can better capture targets of different sizes and improve detection accuracy. Better versatility: The decoupled head mechanism can be used not only for target detection but also for other tasks such as face recognition, pose estimation, etc. This generality allows the decoupled head mechanism to be applied to a variety of different computer vision tasks, improving the usability and usefulness of the model.

Common solutions for improving the detection speed of deep neural networks include: network pruning: by removing some redundant weights and neurons from the network, the number of parameters of the model can be reduced, improving the running speed and robustness of the model, while also improving the detection accuracy; network structure optimisation: improving the network structure can reduce the computational load of the model, thus improving the detection speed. For example, using lightweight network structures, reducing the number of convolutional kernels, reducing the number of pooling layers, etc.; hardware acceleration: using hardware accelerators such as GPUs and ASICs can greatly accelerate the computation speed of the model, thus improving detection speed; parallel computing: using multiple GPUs or distributed computing, the computation task can be spread across multiple computing devices, thus accelerating the computation speed of the model.
Data pre-processing: some pre-processing of the input data, such as downsampling, resizing the image, removing redundant information, etc., can reduce the amount of computation and thus improve the detection speed.
For network structure optimisation, a fast neural network using a convolutional structure based on partial convolution (PConv) is proposed in FasterNet [14]. PConv extracts spatial features more efficiently by reducing redundant memory accesses and computations on reduced floating point operations (FLOPs), based on which the further proposed FasterNet achieves much higher calculations §than other networks without affecting the accuracy of each visual task; the MobileNeXt proposed in [15] by proposing a new bottleneck layer structure and based on this improves model accuracy while reducing the number of parameters in the classification problem.

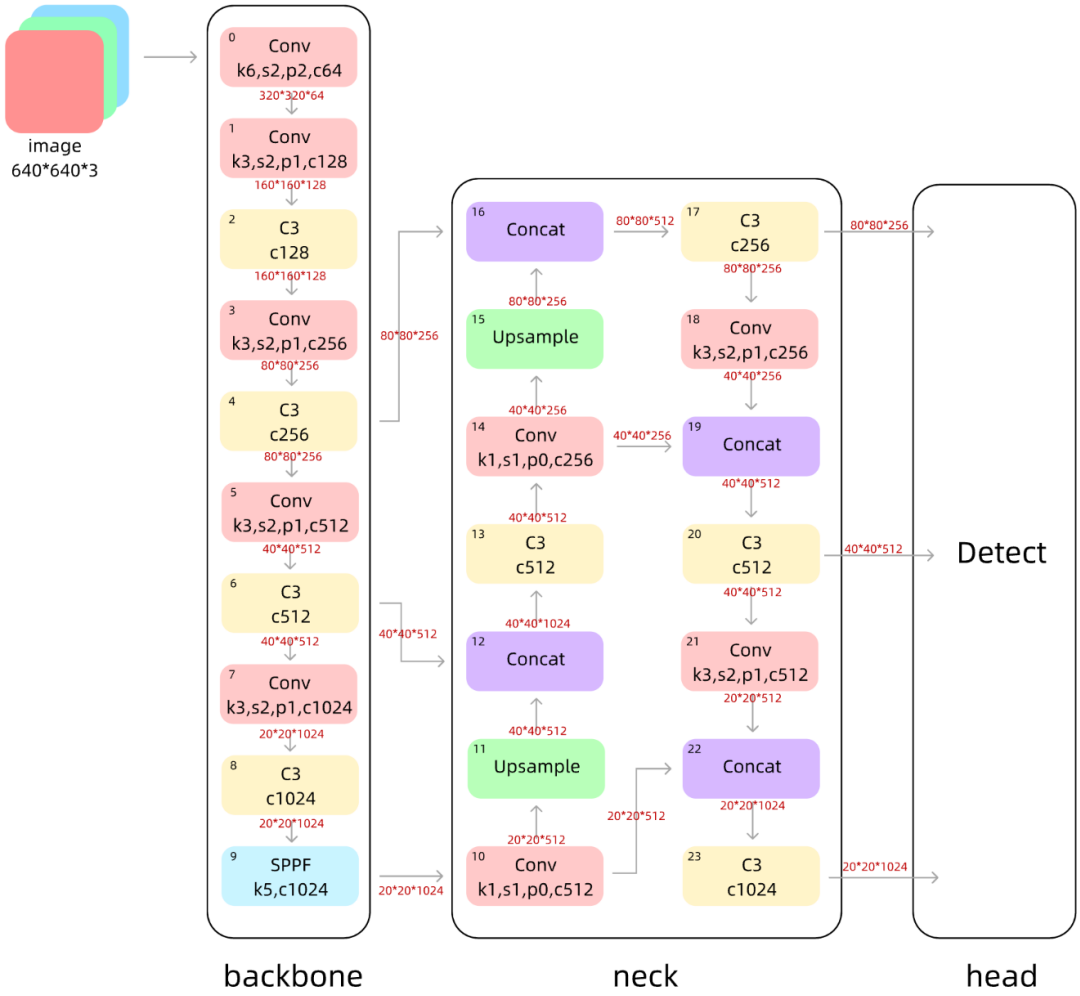
3 Detecting network optimisation schemes

In this section I first review the general framework of the yolov5 algorithm through some diagrams and general formulas. Then, by giving a brief introduction to the principles of CondConv [16] and Involution [17], I go on to explain specifically how I added them to the network framework, and also how these improvements caused an improvement in detection network accuracy and detection speed, as well as the application scenarios for these improvements.

3.1 YOLOv5 detection network

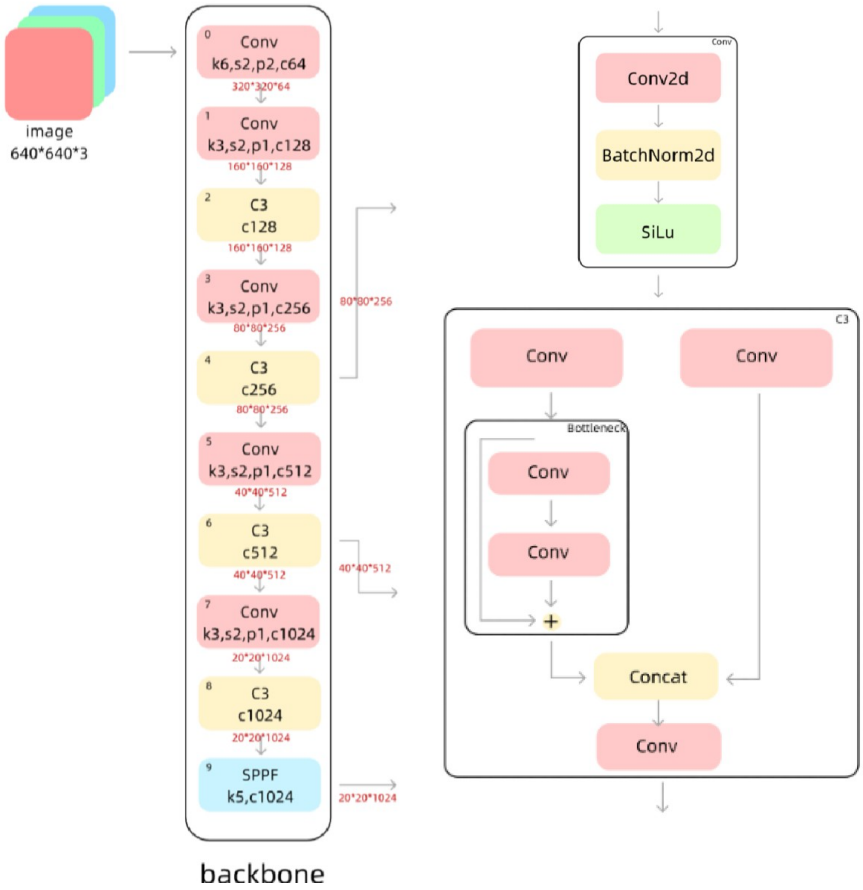
Pedestrian detection refers to the identification and tracking of pedestrians by using computer vision technology to analyse the image information in an image or video from which the location and posture of a traveller can be automatically detected. The technology of pedestrian detection has a wide range of applications in areas such as intelligent video surveillance, autonomous driving and human-computer

interaction. The main objective is to provide basic data for subsequent applications such as pedestrian tracking, behavioural analysis and event prediction through the detection of pedestrians. the YOLO algorithm is the dominant position in the target detection task, with the network combining operational speed and recognition accuracy. For the YOLOv5 detection network used in this report the network can be divided into 3 main parts, backbone for feature extraction; neck for feature fusion; and head for detection.



Backbone

The backbone section of YOLOv5 uses the CSPDarknet53 architecture, which extracts features from the input image and gradually abstracts them for object recognition and bounding box prediction in the subsequent neck and head sections. Specifically, the convolutional layer in backbone and the C3 module are used to extract features at different levels, with the convolutional layer being used to extract low-level features and the C3 module to extract high-level features. The interleaving of these two modules is used to extract features from the input image.

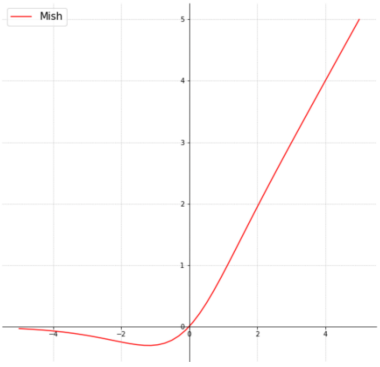


One of the Conv layers (also known as the CBS module) consists of Conv2d, BatchNorm2d and SiLU, specifically, each of these three parts plays the following roles:

Conv2d is the convolution layer, which is mainly used to extract features from the input image. It contains internal parameters such as convolution kernel, step size and padding, and can be used to perform feature extraction on the image by sliding the convolution kernel to obtain the output feature map.

BatchNorm2d is a batch normalisation layer that speeds up the network training process and improves the stability and accuracy of the model. batchNorm2d performs normalisation and scaling operations on the output feature map of Conv2d, making it easier for the feature map to be processed and learned by subsequent layers.

SiLU (Sigmoid Linear Unit) is an activation function of the form f(x) = x * sigmoid(x) that transforms the input data non-linearly to enhance the expressiveness of the model. siLU performs activation operations on the output feature map of BatchNorm2d, transforming it into features with greater expressiveness.



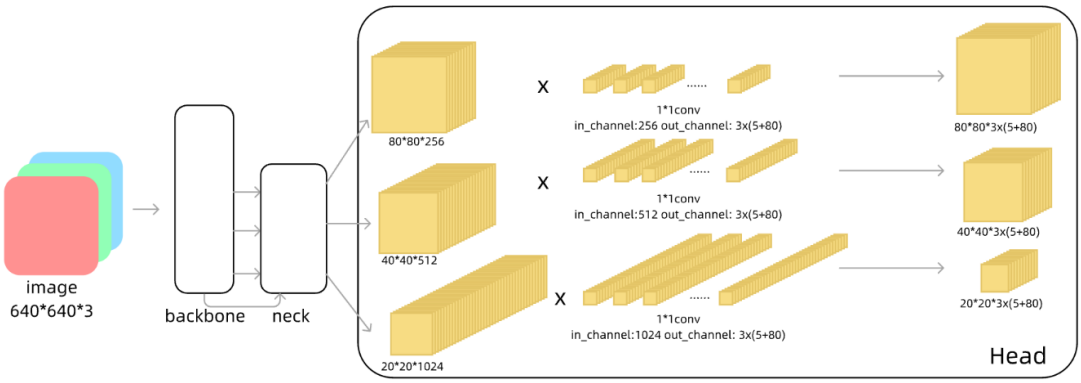
Specifically, the convolutional layer is used to extract low-level features of the image, such as edges and textures, while the C3 module is used to extract higher-level features, such as shapes and object parts, etc. The C3 module consists of three parts: a 1x1 convolutional layer, a 3x3 convolutional layer and a cross-stage partially connected (CSP) layer, where the 1x1 convolutional layer is used to reduce the number of channels in the feature map, the 3x3 convolutional layer is used to extract higher-level features, and the CSP layer is used to split the input feature map into two parts, one of which is computed in the next step, and the other is stitched with the output of the residual block and then computed again.
Overall, the convolutional layer and the C3 module play different roles in YOLOv5's backbone, with the convolutional layer used to extract low-level features and the C3 module used to extract high-level features, and the combination of the two gives YOLOv5 better performance in target detection tasks.

Neck

The neck part of YOLOv5 extracts feature maps at different scales through a combination of FPN (Feature Pyramid Network) and PAN (Path Aggregation Network) parts, and fuses them for the detection of targets at different scales. The smaller scale targets are mainly detected by the higher scale feature maps, while the larger scale targets need to be detected using the lower scale feature maps. The feature maps generated in the neck section cover the spatial features of small, medium and large targets, and these written features are passed to the head section for detection and classification work.

Head

The head part of YOLOv5 uses the YOLOv5-Detection module, which is used to perform object recognition and bounding box prediction on the features from neck. Specifically, the YOLOv5-Detection module uses multiple convolutional and pooling layers to extract features, and then uses multiple detection heads to predict different classes of objects and their bounding boxes.



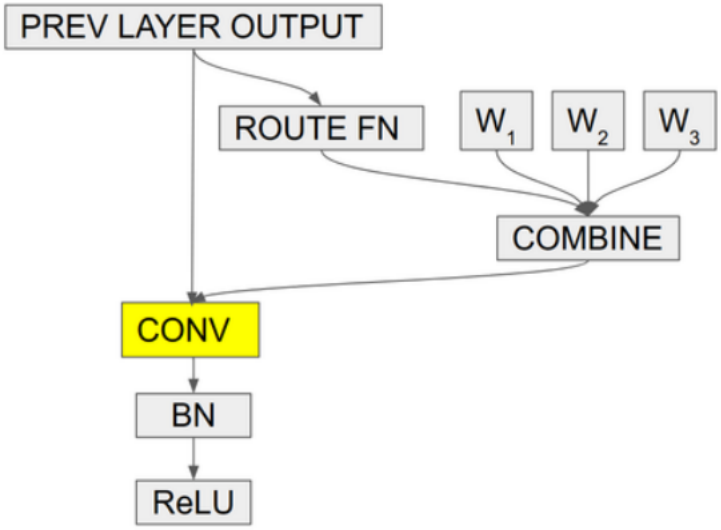
3.2 Specific improvement options

As the main objective of this project is to improve the accuracy and recognition speed of pedestrian detection, subsequent experiments were conducted to improve the NECK and backbnone of the YOLOv5s network structure in two directions, mainly to improve the mAP and mainly to improve the FPS, respectively, in order to obtain performance improvements.

3.2.1 Model Lightweighting and FPS Enhancements

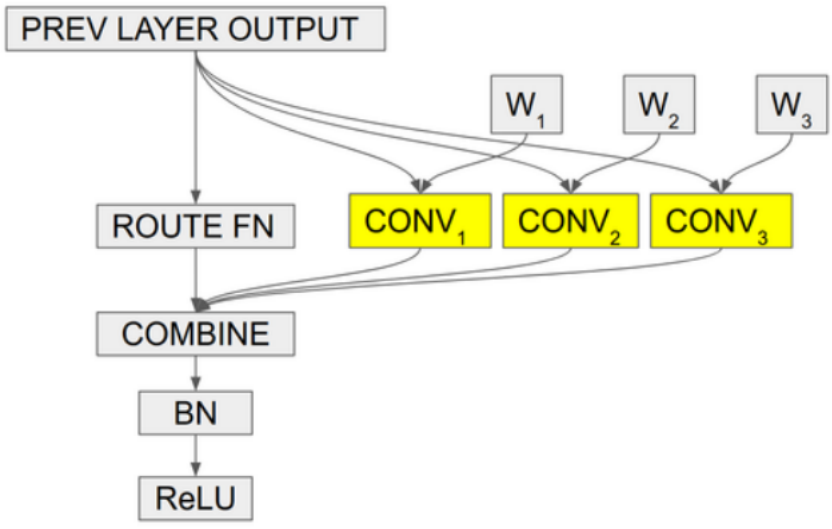
Since increasing the number of layers in the network leads to an increase in model complexity while enhancing the representational power of the model, what if there was a structure that had the feature extraction power of a multi-layer convolutional kernel while taking less time for convolutional computation? layer in YOLOv5s backbone.

CondConv



$$\mathbf{Output(x) = \sigma((\alpha 1 \cdot W1 + \ldots + \alpha n \cdot Wn) * x)} \quad \mathbf{(1)}$$

Unlike regular convolutional kernels (Form.1)where different inputs share the same convolutional kernel, in the CondConv layer, the convolutional kernel changes depending on the different inputs. Specifically, it parameterises the convolution kernel in CondConv by:



$$\mathbf{\sigma((\alpha 1 \cdot W1 + \ldots + \alpha n \cdot Wn) * x) = \sigma(\alpha 1 \cdot (W1 * x) + \ldots + \alpha n \cdot (Wn * x))} \quad \mathbf{(2)}$$

In the formula αi = ri(x) the example-dependent weights are calculated through a routing function with learning parameters. （In (form.2) n represents the number of experts, σ the activation function. Since the convolutional layers need to be adjusted to use CondConv, the original convolutional kernel is aligned

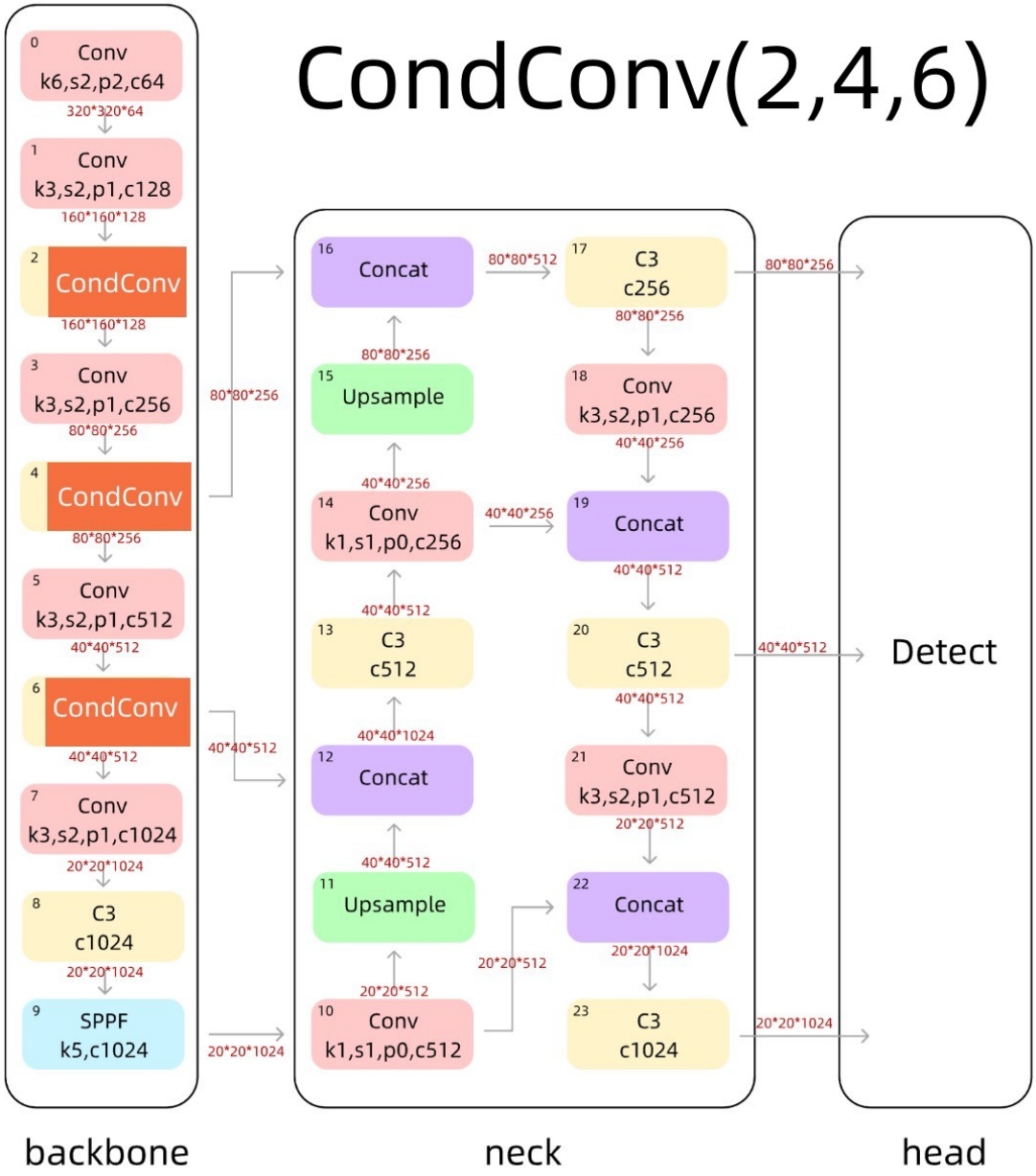
with each kernel Wi in terms of the number of dimensions. In the face of better image feature extraction, this is usually achieved by adding the length and width or dimensionality of the regular convolution kernel. However, as each additional parameter in the convolution needs to be convolved with the parameters in the input feature map, the apparent increase (in aspect or number of channels) often add the computational effort exponentially. In contrast, in the CondConv layer, a convolution kernel is computed as a linear combination of n experts by computing a convolution kernel for every input feature map before using convolution. The neat thing is that each convolution kernel in CondConv only needs to be computed once, but is applied to many different positions of the input feature map. This means that by adding just n in terms of enhanced feature extraction capability, we can add the capacity of the network while adding its feature extraction capability, without the large increase in inference cost that would be associated with conventional convolution; every additional parameter corresponds to only 1 additional multiplication. The CondConv kernel is therefore mathematically equivalent to the more expensive expert linear mixture formulation, where every expert corresponds to a ordinary convolution. In(form.3)We go through the three steps of global average pooling, full connection layer, and Sigmoid activation to Mapping relationships to the exampledependent routing weights αi = ri(x). are obtained from different feature maps based on input from the previous layer

r(x) = Sigmoid(GlobalAveragePool(x) R) 3 (3)

Consequently, CondConv has the same feature extraction capability and at the same time performs better in terms of computational efficiency than conventional convolution, because it only needs to be computed once in the face of expensive convolution operations. This formulation is interlinked with previous work on conditional computation and expert mixing. The results obtained by the routing function according to the different feature maps of the input are crucial for CondConv performance: if the parameters learned by the routing function are the same for all feature map of the input, the CondConv layer has the same feature extraction power as the regular convolution. CondConv, which consists of the above formula, is characterised by computational efficiency and greater feature extraction power.

In conclusion, CondConv is a conditional computation-based convolutional operation that can dynamically adjust the computation of the convolutional operation at runtime based on the features of the input data by introducing a learnable conditional weight in the convolutional layer, thus better adapting to different input data distributions.

YOLOv5s backbone is a neural network structure consisting of multiple convolutional and pooling layers to extract feature information from the input image and provide feature map input for subsequent target detection tasks. It has the highest number of parameters compared to neck and head, and takes on the role of extracting high level feature representations from the input image. Therefore, by using CondConv to replace C3 layers of different sizes in the backbone of the original network structure, and by using CondConv, a special convolution with strong feature extraction capability, instead of C3 layers composed of three different convolutional layers, the overall depth of the network is actually reduced while maintaining its feature extraction capability as much as possible, thereby reducing the operational complexity of the deep convolutional network. The aim is to actually reduce the overall depth of the network while maintaining its feature extraction capability as much as possible, thus increasing the overall inference speed (FPS).



Through experiments (Section 4.3), I found that replacing the C3 modules of YOLOv5s backbone 2, 4 and 6 with CondConv, a single convolutional module with strong feature extraction capability, improved the FPS of the whole network without losing mAP compared to the performance of the original YOLOv5s network framework. This improvement also has minimal impact on mAP while improving FPS compared to the other improvements in (Section 4.3). The specific network structure for this improvement is shown above. the improvement in comparison with the baseline is shown in the table below

Backbone	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
Yolov5s_CondConv(2,4,6)	**0.876**	**11.4**	**178486038**	**153**	**111.423**

3.2.2 Improving the accuracy of the model

This project focuses on the detection of pedestrians targeting the 'person' category, and most of the images in the dataset are taken from a surveillance perspective, so the labelled bounding box is mainly small and medium targets. Therefore, enhancing the detection accuracy of this part is crucial to improving the overall network recognition accuracy (mAP).

Small target detection has always been a challenging problem in the field of computer vision, as the detection and localisation of these targets is often a difficult task due to their small size in the image, and to solve this problem I use Involution, a convolutional kernel with strong feature extraction capabilities.

Involution

In order to cope with the characteristics of transformations with inclusive space and inverse features in the channel domain, the inverse convolution kernel H ∈ RH ×W ×K ×K ×G was created on demand.

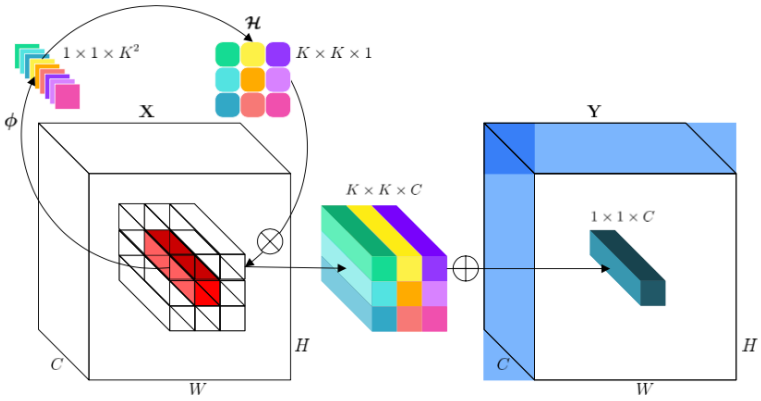
$$Y_{i,j,k} = \sum H_{i,j,u+\lfloor K/2 \rfloor, v+\lfloor K/2 \rfloor, \lceil kG/C \rceil} X_{i+u,j+v,k} \cdot \quad (u,v) \in \Delta K \tag{4}$$

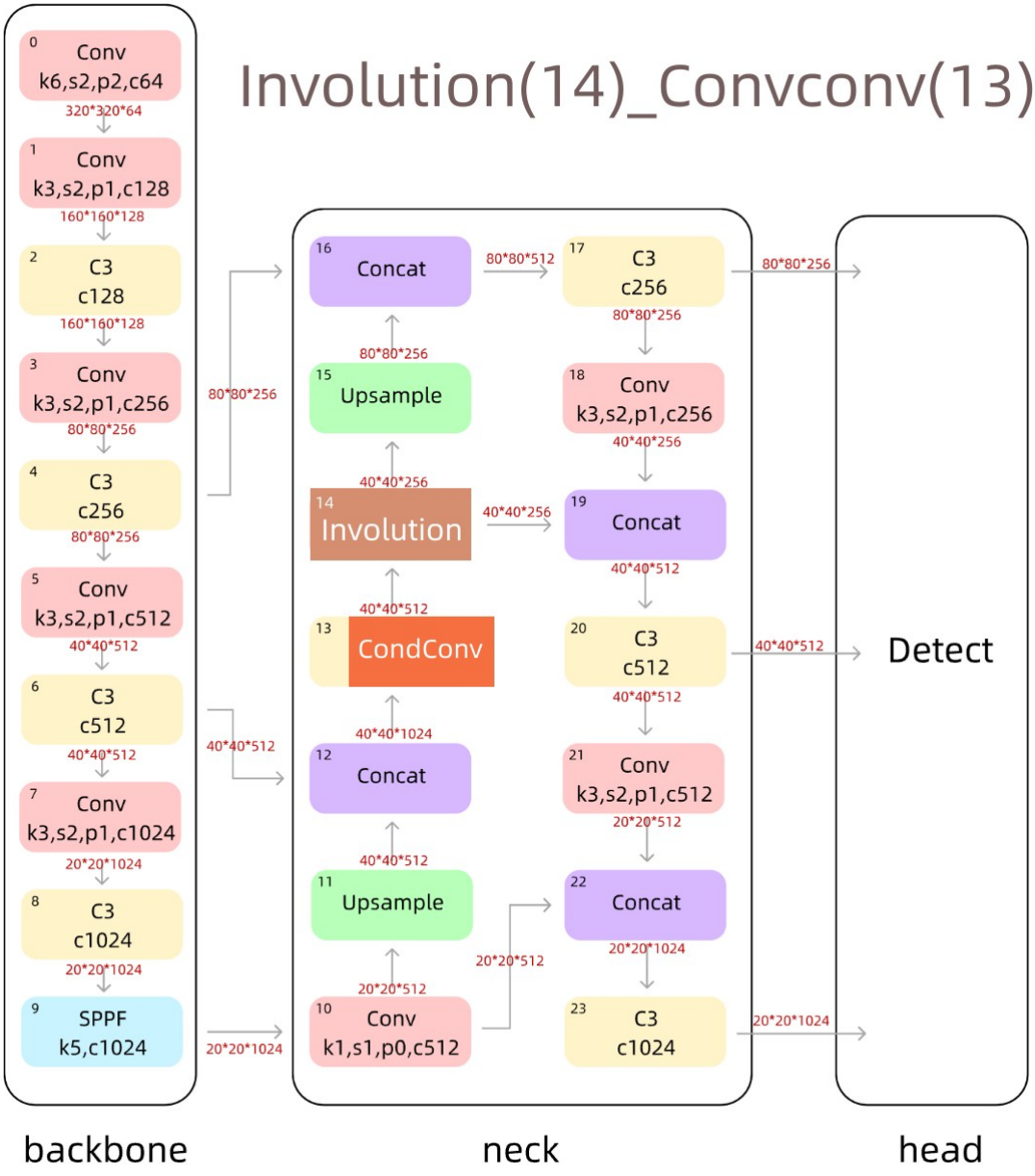
When we have an involution kernel Hi,j,-,-,g ∈ RK ×K, g = 1, 2, - -, G, it shares in dimension. The G mentioned above records the number of convolution kernels that share the involution. the feature map of the involution is obtained from (form.4)

$$\mathcal{H}_{i,j} = \phi(\mathbf{X}_{i,j}) = \mathbf{W}_1 \sigma(\mathbf{W}_0 \mathbf{X}_{i,j}). \tag{5}$$

The shape of the involution convolution convolution kernel(H) changes according to the shape of the input feature map(X).(form.5)

Involution transforms the convolution operation into two parts, mainly by rearranging the dimensions to the space, sharing the result of the convolution computation in the dimension C; and performing a downscaling operation in the part K*K*C, transforming it into the result 1*1*C.


$1 \times 1 \times K^2$
$\mathcal{H}$
$K \times K \times 1$
ϕ
X
$K \times K \times C$
Y
$1 \times 1 \times C$
H
C
W
H
C
W


Involution(14)_Convconv(13)
0 Conv k6,s2,p2,c64
320*320*64
1 Conv k3,s2,p1,c128
160*160*128
2 C3 c128
160*160*128
3 Conv k3,s2,p1,c256
80*80*256
4 C3 c256
80*80*256
5 Conv k3,s2,p1,c512
40*40*512
6 C3 c512
40*40*512
7 Conv k3,s2,p1,c1024
20*20*1024
8 C3 c1024
20*20*1024
9 SPPF k5,c1024
20*20*1024
80*80*256
40*40*512
16 Concat
80*80*512
17 C3 c256
80*80*256
80*80*256
15 Upsample
40*40*256
18 Conv k3,s2,p1,c256
40*40*256
14 Involution
40*40*256
19 Concat
40*40*512
40*40*512
13 CondConv
20 C3 c512
40*40*512
40*40*1024
12 Concat
21 Conv k3,s2,p1,c512
20*20*512
40*40*512
11 Upsample
22 Concat
20*20*1024
20*20*512
20*20*512
10 Conv k1,s1,p0,c512
23 C3 c1024
80*80*256
40*40*512
Detect
20*20*1024
backbone
neck
head

The neck part of YOLOv5 extracts feature maps at different scales through FPN and PAN and fuses them in order to use them for the detection of targets at different scales. For this part of the structure, I replace layer 14 of the original detection network by using Involution and replace layer 13 of the original detection network by using CondConv to improve the information flow transfer capability of the detection network by using involution's unique decoupling of channel information from spatial information. This improves the information flow transfer capability of the detection network by using the unique decoupling of channel information and spatial information, while at the same time improving the expression capability of the network and to some extent facilitating the miniaturisation of the network structure. Using this improved solution, the detection network reduces the computational complexity by 0.57% to 14.9 GFLOPs with little impact on the detection speed, and improves the detection accuracy by 0.91%, resulting in a detection accuracy of 88.4% mAP

Neck	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
Involution(14)_Convconv(13)	**0.884**	**14.9**	**11496167**	**204**	**92.048**

3.2.3 Improvements to improve the detection speed of the detection network at the expense of accuracy

There is also an improvement that reduces the mAP by a certain amount in exchange for a large increase in FPS, in which the same CondConv is used as in 3.2.1 to replace layer C3 of the yolov5s backbone. The difference is that layers 2, 4, 6, and 8 of the original network are replaced in this improvement, which significantly reduces the complexity of the network operations and significantly reduces the number of layers in the network, but with a reduction in feature extraction. Nonetheless, this improved scheme is still an improved method for achieving the highest operational complexity of lightweight detection networks during experiments and is particularly suitable for deployment in edge-side pedestrian detection device

Backbone	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
Yolov5s_CondConv(2,4,6,8)	**0.869**	**10.4**	**17615382**	**140**	**116.614**

3.2.4 Improvements to the detection network to improve accuracy at the expense of detection speed

There are also improvements that can be made to increase the mAP by a large amount while affecting the FPS to a greater extent. In this improvement I reduce the detection loss of the detection network for small targets in the dataset by adding a new detection head to the neck part of yolov5s for detecting very small targets. Specifically, when the input image size is 640*640, the original yolov5 has three detection layers for small, medium and large targets, corresponding to feature maps of 80*80, 8*8, 40*40, 16*16,

20*20 and 32*32. This improved solution adds a small target detection layer to the original one, corresponding to feature maps of 160*160 and 4*4. map with 4*4 targets.

```
anchors:
  - [5,6, 8,14, 15,11]  #4
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32


   [-1, 1, Conv, [256, 1, 1]], #18  80*80
   [-1, 1, nn.Upsample, [None, 2, 'nearest']], #19  160*160
   [[-1, 2], 1, Concat, [1]], #20 cat backbone p2  160*160
   [-1, 3, C3, [256, False]], #21 160*160
```

Neck	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
Yolov5s_4neck	**0.890**	**26.8**	**7672042**	**259**	**71.487**

4. The experiment

4.1 Experimental objectives and selection of evaluation metrics

The main objective of this experiment is to improve the accuracy and recognition speed of pedestrian detection by using the following MAPs, GFLOPs and FPS as reference quantities to evaluate and analyse the improvement points.

In order to effectively evaluate the detection effectiveness of the model, the performance of the model is measured by MAP (mean average precision), while the complexity of the model is expressed in terms of the number of parameters, the complexity of the operation is expressed in terms of GFLOPs, and the detection speed of the algorithm is expressed in terms of FPS. The specific expression for mAP is shown in equation (form.6):

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k} \quad (6)$$

$$\text{Precision (P)} = TP / (TP + FP) \quad (7)$$

$$\text{Recall (R)} = TP / (TP + FN) \quad (8)$$

$$AP = \int_0^1 P(R)\, dR \quad (9)$$

In (form.7;8), True Positives (TP) is the number of positive samples successfully classified by the model; False Positives (FP) is the number of negative samples incorrectly classified as positive by the model; and False Negatives (FN) is the number of positive samples missed by the model. In equation (form.6), k denotes the number of categories, while APi is the AP value of the ith category, for this project a higher value means that the model has a high TP along with high accuracy, representing a higher accuracy rate. GFLOP is a measure of the complexity of an algorithm or model and can be used to determine the amount of computation in a model. 1GFlops = 1,000MFlops.

FPS=1000/(inference time+pre-process+NMS) (10)

The FPS (Frames Per Second) is obtained from equation (form.10) and provides the most intuitive representation of the model inference speed. The minimum requirement for real time target detection is an FPS of 60 or more, below 60 FPS does not satisfy the need for real time target detection, so subsequent experiments will need to improve this parameter.

pre-process: image pre-processing time, including image keeping aspect ratio scaling and padding padding, channel transformation (HWC->CHW) and upscaling, etc.

inference: inference speed, the time from the image input model to the model output result after pre-processing;

NMS: post-processing time, where the output of the model is transformed, etc.

4.2 Dataset preparation

The images in the dataset used in this project were mainly taken from a surveillance perspective. The quality of the annotation and the number of bounding boxes in each image is high. There is only one annotation category, 'person' {0}, in which the annotated pedestrians are mainly annotated with three features: pedestrians, riders, and partially visible pedestrians.

The preparation of the dataset has a significant impact on the subsequent work. A low quality dataset will have an incalculable impact on the subsequent work, while a well-labelled dataset will provide a solid foundation for subsequent training. It is also important to analyse the image characteristics of the dataset for subsequent improvement.

The VOC (Visual Object Class) dataset is also a common dataset for target detection and image segmentation tasks. It contains 20 classes. By extracting the 'person' class images from the VOC2012 and VOC2007 data, it was transformed into a yolo format, i.e. the original .xml label class file was transformed (into a .xml label class file suitable for the yolo algorithm) with label, anchor frame centroid x, y, anchor frame length and width per line txt file with label, anchor frame centroid x, y, anchor frame length and width w, h as annotated data) and perform a data cleaning operation to remove the images that cause the most mAP loss and filter out 1000 of them. Finally, the data was split into three parts: train, val, and test, in the ratio of 7:2:1. The YOLOv5s network model was trained on this dataset (VOC_person).

To enhance image diversity the WiderPerson dataset containing data from the three classes 'pedestrian, rider, partially visible pedestrian' was used and merged with the previous VOC_person dataset and some additional self-labelled images to form a dataset of 10,000 images with only one class [person], also divided into train, val, test in the same 7:2:1 ratio. This dataset (WiderPerson) obtained 87.3% MAP in

the YOLOv5 algorithm using the YOLOv5s network model, and also improved in terms of the number of labels per image and overall quality compared to VOC_person.

Datasets	mAP50	Images	Labels/Images
VOC_person	**0.823**	**1000**	**13.175**
WilderPerson_person	**0.873**	**10000**	**17.833**

Since this dataset WilderPerson_person itself has 10,000 images and 10-25 labeled frames per image, the training was slowed down by renting a NAVIDIA TESLA V100 GPU-32GB from the cloud platform. cuda11.3, torch1.12.1+cu113, and tensorboard for the analysis of the experimental results. In order to ensure the accuracy of the experiments, epochs = 100, batch-size == 128, single-cls = 'store_false' were used as the base parameters for the subsequent evaluation.

baseline	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s(baseline)	**0.876**	**15.8**	**7012822**	**213**	**92.452**

4.3 Backbone section experiments

In the process of improving only the Backbone of YOLOv5s, I have obtained the following results.

Backbone	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s(baseline)	**0.876**	**15.8**	**7012822**	**213**	**92.452**
1.Yolov5s_CondConv(2,4,6)	**0.876**	**11.4**	**178486038**	**153**	**111.423**
2.Yolov5s_CondConv(2,4,6,8)	**0.869**	**10.4**	**17615382**	**140**	**116.614**
3.Yolov5s_PConv(2,4,6,8)	**0.866**	**11.7**	**5548566**	**146**	**108.82**
4.Yolov5s_DyConv(2,4,6,8)	**0.867**	**10.5**	**17699542**	**148**	**104.652**
5.Yolov5s_CondConv(2,4)	**0.870**	**12.3**	**7289622**	**166**	**102.645**
6.Yolov5s_CondConv(4,6)	**0.879**	**12.3**	**99225942**	**166**	**101.616**
7.Yolov5s_CondConv(6)	**0.880**	**15.2**	**8865110**	**179**	**99.219**
8.Yolov5s_FasterNet	**0.807**	**9.0**	**5703990**	**182**	**97.921**
9.Yolov5s_Involution(0,1,3,5,7)	**0.878**	**14.8**	**6296808**	**221**	**86.646**

1.Yolov5s_CondConv(2,4,6) represents the use of CondConv to replace the C3 layer in the backbone of YOLOv5s at layers 2, 4 and 6. This improvement minimizes the impact on mAP in the improvement idea with the main objective of improving FPS, i.e. reasoning at 111FPS(18) with mAP consistent with the baseline (YOLOv5s) and reducing the computational complexity to 11.4GFLOPs.Also, the largest FPS improvement pursued with a small reduction in mAP is **2.Yolov5s_CondConv(2 ,4,6,8)**, in which all C3 layers in the baseline, i.e. layers 2, 4, 6 and 8 in the backbone, are replaced by CondConv, which sacrifices 0.7% of mAP but increases the inference speed to 116.614 FPS and reduces the computational complexity to 10.4GFLOPs. These two improvements are suitable for some pedestrian detection projects where lightweight and high real-time performance is required.

2.Yolov5s_PConv(2,4,6,8) replaces the original C3 layer by using a lightweight convolution based on a partial convolution, but this convolution is not good enough for feature extraction because it is more like a "T" shape than a normal convolution kernel. This is because it is more like a "T" shape than a normal convolution kernel, as the space used to store the convolution parameters inside it is more prone to losing some features than a normal convolution. Since using only a single layer of PConv would result in a significant drop in feature extraction, I replaced the C3 layer above with a suitably deeper PConv to improve its FPS with less impact on mAP, and the experimental results showed that the improvement in FPS (16) due to the weaker feature extraction ability of PConv itself did not compensate for the drop in

mAP (1%). However, this does not mean that this convolution does not improve the overall performance, so I tried to replace the original backbone completely with a FasterNet built on it.8.Yolov5s_FasterNet is a lightweight network structure based on a partial convolution, using this backbone Although there is a significant reduction in GFLPOPs, there is a decrease in mAP (7%) and an increase in FPS (5). The Yolov5 is a very small computational complexity (9FLOPs), but due to its small computational complexity, it is more suitable for deployment in edge-side devices such as JETSON NANO or Raspberry Pi, where the computational power is limited. In **4.Yolov5s_DyConv(2,4,6,8)**, DyConv is used to replace the C3 layer of the YOLOv5s backbone at layers 2, 4, 6 and 8. The attention weighting of these convolutions depends on the input features. This means that different convolutions are used for different inputs, improving the ability of the convolutional layer to extract features without actually increasing the depth of the network.

4.4 Neck section experiments

Neck	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
1.Involution(14)_Convconv(13)	**0.884**	**14.9**	**11496167**	**204**	**92.048**
2.Yolov5s_4neck	**0.890**	**26.8**	**7672042**	**259**	**71.487**
3.Yolov5s_DyConv	**0.870**	**12.8**	**18524502**	**180**	**94.905**
4.Yolov5s_BifusionNeck	**0.878**	**16.7**	**7174230**	**237**	**83.072**
5.Yolov5s_CondConv(13, 17, 20, 23)	**0.869**	**11.6**	**22789718**	**161**	**96.817**

The analysis of the dataset revealed that since the dataset images were taken from a surveillance perspective, most of the pedestrian annotations were dominated by small and medium targets, which means that in order to improve the model recognition capability, the use of convolution with better feature extraction capability in this part can exploit the model recognition potential

Using **1.Involution(14)_Convconv(13)** as a guideline, this improvement is the best solution for the whole Neck section, by replacing the C3 layer with CondConv in layer 13 and the Conv layer with Involution in layer 14 of YOLOv5s Neck. In **2.Yolov5s_4neck**, a new detection head for very small targets was added to the Neck part of the baseline to improve the recognition capability of the model. (1.4%), but this has a significant impact on the computational complexity and has the undesirable effect of slowing down the computational speed. However, both of these improvements are still well suited to edge-end devices that require a certain level of accuracy and have sufficient computing power.

4.5Head section experiments

Head	mAP50	GFLOPs	Parameters	Layers	FPS
Yolov5s	**0.876**	**15.8**	**7012822**	**213**	**92.452**
1.Yolov5s_DecoupledHead	**0.890**	**56.2**	**142321622**	**196**	**-**
2.Yolov5s_DecoupledHeadxs	**0.883**	**21.3**	**8764200**	**264**	**65.14**

Yolov5s head uses two network structures, SPP (Spatial Pyramid Pooling) and PAN (Path Aggregation Network), which are mainly used to extract multi-scale information from feature maps and enhance the network's perception of target detection. The PAN is mainly used to fuse feature maps from different scales to improve the accuracy and robustness of target detection. The final output layer consists mainly

of multiple convolutional and detection layers, which are used to convert the feature maps into detection results and output information such as the position, class and confidence of the detected frames.

In 1.Yolov5s_DecoupledHead, because YOLOv5s target detection models typically use a single unified head (head) to process all outputs in the target detection task, such as bounding box coordinates, category confidence levels, etc.. This design makes it difficult to fully explore the relationships and characteristics between different outputs as the head needs to process many different pieces of information at the same time, and therefore may affect the performance of the model. The decoupled head in my port of YOLOX separates the different outputs of the target detection task into separate decoupled heads, each of which is responsible for processing only one specific output, such as bounding box, category confidence and semantic segmentation. This design allows each head to specialise in optimising the processing of a single output, thus fully exploiting the features and relationships of each output and improving the performance and accuracy of the model.

Also, I experimented to decoupled heads in YOLOX also using a lightweight design 2.Yolov5s_DecoupledHeadxs to reduce the complexity and inference time of the model by reducing the number of parameters and the amount of computation.

However, these two modules, although they have a bright performance in terms of mAP improvement, are only used as a control group for reference due to their massive increase in computational complexity and great reduction in computational speed.

5. Conclusions:

The following four improvement schemes performed relatively well in the experiments, the first two reducing computational complexity or improving computational accuracy with little negative impact on the other evaluation parameters, while the latter two pursued optimal performance in reducing computational complexity or improving computational accuracy with low best results while causing a small negative impact on the other evaluation parameters, but still within tolerable limits.

The most comprehensive improvement solution with the main objective of increasing detection speed is **Yolov5s_CondConv(2,4,6)**, which achieves an inference speed of 111 FPS(18) with the same mAP as the baseline (YOLOv5s) (87,6mAP), while reducing the computational complexity to 11.4 GFLOPs. This was one of the best-performing improvement solutions throughout the experiment in terms of improving detection speed without any negative impact on detection accuracy.

Involution(14)_Convconv(13) was the most comprehensive solution with the main objective of improving detection accuracy. This solution improved mAP to 88.4% with almost no reduction in FPS, a 0.8% improvement compared to baseline and a 5.7% reduction in complexity to 14.9 GFLOPs. GFLOPs. was one of the best-performing solutions throughout the experiment in terms of improving detection accuracy with little or no negative impact on detection speed.

The **Yolov5s_CondConv(2,4,6,8)** solution is the most suitable solution for cases where the computational power is limited, such as the JETSON NANO or Raspberry Pi, and it reduces the computational complexity by 33.5% in the experiments. The computational complexity of this improved scheme is only 10.5 GFLOPs and the inference speed is increased to 116.614 FPS compared to baseline, a 26.1% increase, with only a 0.7% decrease in recognition accuracy due to the negative impact of these improvements.

In the case of a higher demand for detection accuracy, I improved the detection capability of the original detection network for small and very small targets by adding a detection head for very small targets to the neck part of YOLOv5s(**Yolov5s_4neck**), which improved the detection accuracy from 1.4% to 89%,

but due to the large increase in computational complexity of the detection head pair, the However, due to the significant increase in computational complexity associated with the addition of the detection head pair, the FPS is reduced from 92.452 FPS to 71.487 FPS, although this improvement still meets the minimum requirement of 60 FPS for real-time target detection.

reference

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524**.Nov 2013. https://arxiv.org/abs/1311.2524**

[2] Ross Girshick.Fast R-CNN. arXiv:1504.08083 **Apr 2015. https://arxiv.org/abs/1504.08083**

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497**.Jun 2015. https://arxiv.org/abs/1506.01497**

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640.Jun 2015. https://arxiv.org/abs/1506.02640

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector.arXiv:1512.02325.Dec 2015. https://arxiv.org/abs/1512.02325

[6] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv:1804.02767**.Apr 2018. https://arxiv.org/abs/1804.02767v1**

[7] Stefan Elfwing, Eiji Uchibe, Kenji Doya. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. arXiv:1702.03118**.Feb 2017. https://arxiv.org/abs/1702.03118**

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin.Attention Is All You Need. arXiv:1706.03762**.Jun 2017. https://arxiv.org/abs/1706.03762**

[9] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, Zicheng Liu. Dynamic Convolution: Attention over Convolution Kernels. arXiv:1912.03458 **.Dec 2019. https://arxiv.org/abs/1912.03458**

[10] Chao Li, Aojun Zhou, Anbang Yao.OMNI-DIMENSIONAL DYNAMIC CONVOLUTION. arXiv:2209.07947**.** Sep 2022.**https://arxiv.org/abs/2209.07947**

[11] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, Xiangxiang Chu. YOLOv6 v3.0: A Full-Scale Reloading. arXiv:2301.05586.Jan 2023. https://arxiv.org/abs/2301.05586

[12] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun. YOLOX: Exceeding YOLO Series in 2021. arXiv:2107.08430 .Jul 2021.https://arxiv.org/abs/2107.08430

[13] Songtao Liu, Di Huang, Yunhong Wang. Learning Spatial Fusion for Single-Shot Object Detection. arXiv:1911.09516**.Nov 2019. https://arxiv.org/abs/1911.09516**

[14] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, S.-H. Gary Chan. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. arXiv:2303.03667 , Mar 2023. https://arxiv.org/abs/2303.03667

[15] Zhou Daquan, Qibin Hou, Yunpeng Chen, Jiashi Feng, Shuicheng Yan. Rethinking Bottleneck Structure for Efficient Mobile Network Design. arXiv:2007.02269 ,Nov 2020. https://arxiv.org/abs/2007.02269

[16] Brandon Yang, Gabriel Bender, Quoc V. Le, Jiquan Ngiam.CondConv: Conditionally Parameterized Convolutions for Efficient Inference. arXiv:1904.04971**.Apr 2019. https://arxiv.org/abs/1904.04971**

[17] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, Qifeng Chen.Involution: Inverting the Inherence of Convolution for Visual Recognition. arXiv:2103.06255.Mar 2021. https://arxiv.org/abs/2103.06255

[18] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn & Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. International Journal of Computer Vision.sep 2009. https://link.springer.com/article/10.1007/s11263-009-0275-4

[19] Shifeng Zhang, Yiliang Xie, Jun Wan, Hansheng Xia, Stan Z. Li, Guodong Guo. WiderPerson: A Diverse Dataset for Dense Pedestrian Detection in the Wild. arXiv:1909.12118 .Sep 2019. https://arxiv.org/abs/1909.12118