RFAConv: Innovating Spatial Attention and Standard Convolutional Operation

Xin Zhang[1], Chen Liu[1], Tingting Song[1,*], Degang Yang[1,2,*], Yichen Ye[3], Ke Li[1], and Yingze Song[1]

1 College of Computer and Information Science, Chongqing Normal University

2 Chongqing Engineering Research Center of Educational Big Data Intelligent Perception and Application

3 College of Electronic and Information Engineering, Southwest University

**Corresponding author: E-mail:address:{ttsong,yangdg}@cqnu.edu.cn*

Presenter: Nguyen Duy Linh

ndlinh301@mail.ulsan.ac.kr

Jan. 20, 2024

- *Submitted on https://arxiv.org/*
- *Date: 12 Oct 2023*

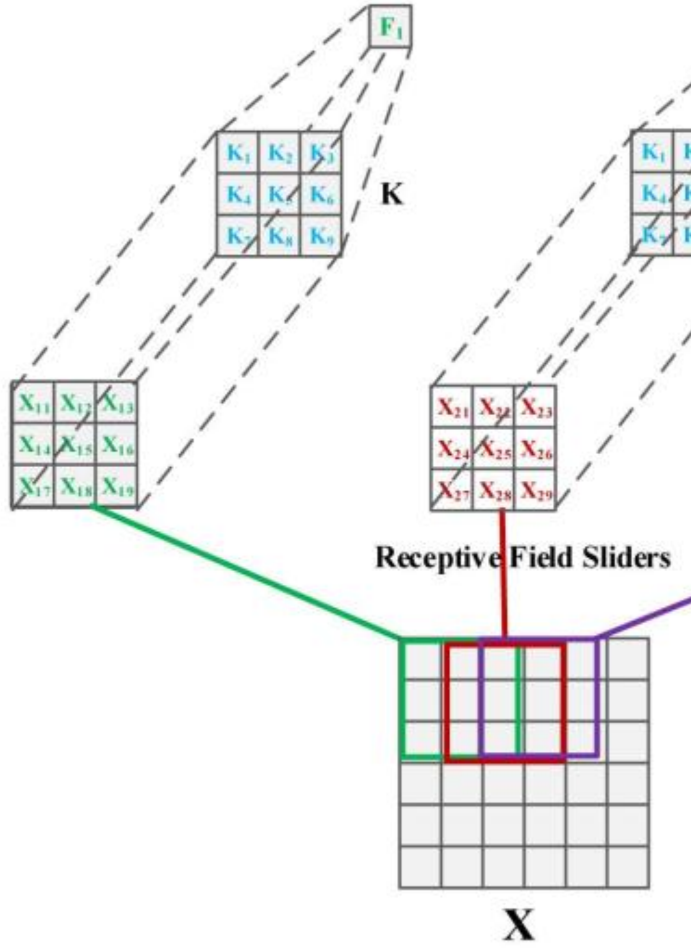
Motivation

- Convolutional neural networks have dramatically reduced the computational overhead and complexity of models by using the **convolutional operation** with **shared parameters**
- During the convolutional operation, the **kernel** uses the same parameters in each receptive field to extract information, which does not consider the differential information from different locations -> the performance of the network is limited
- The convolutional process does not take into account the significance of each feature -> further reduces the efficiency of the extraction features -> ultimately restricts the performance of the model
- The attention mechanism enables the model to concentrate on significant features
- The current spatial attention mechanism does not fully address the parameter sharing problem for larger convolutional kernels
- This paper proposes a novel receptive-field attention (RFA) that comprehensively addresses the above issues

Contributions

- **The main contributions in the paper:**
  1. Proposes a novel **receptive-field attention (RFA)** that comprehensively addresses the issue of parameter sharing for convolutional kernels and takes into account the significance of each feature in the receptive field
  2. Designs the new **convolutional operation (RFAConv)** that can replace standard convolutional operations in current neural networks
  3. Provides an **upgraded version of CBAM and CA** and conduct relevant experiments

Reviewing Standard Convolutional Operation



$$F_1 = X_{11} \times K_1 + X_{12} \times K_2 + X_{13} \times K_3 + \ldots + X_{1S} \times K_S$$
$$F_2 = X_{21} \times K_1 + X_{22} \times K_2 + X_{23} \times K_3 + \ldots + X_{2S} \times K_S$$
$$\ldots$$
$$F_N = X_{N1} \times K_1 + X_{N2} \times K_2 + X_{N3} \times K_3 + \ldots + X_{NS} \times K_S$$

Fig. 1. It simply represents a 3×3 convolution operation. The features are obtained by multiplying the convolution kernel with a receptive-field slider of the same size and then summing.

Reviewing Spatial Attention

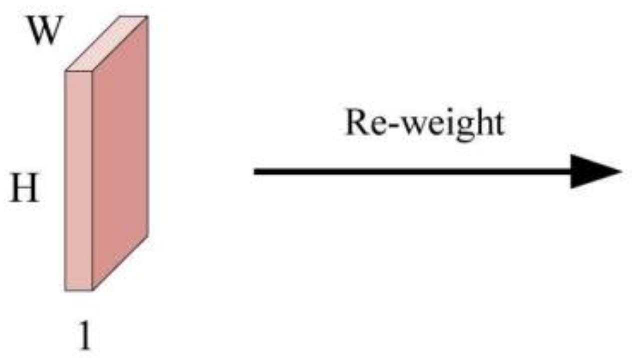


Fig. 2. The original feature map highlights the key features by learned attention map. This process of highlighting is the Re-weight ($\times$) operation.

$$F_1 = X_1 \times A_1$$
$$F_2 = X_2 \times A_2$$
$$\dots$$
$$F_N = X_N \times A_N$$

- The spatial attention mechanism uses the attention map obtained through learning to highlight the importance of each feature

Spatial Attention and Standard Convolutional Operation

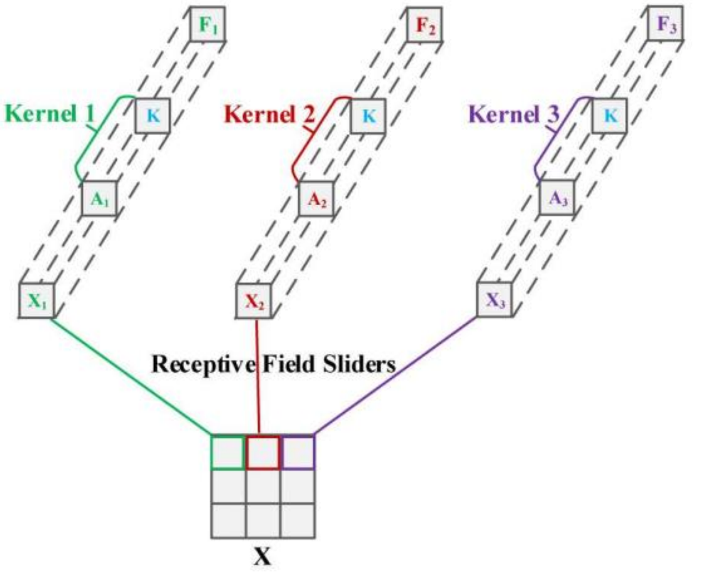


Fig. 3. The convolutional kernel parameter K_i obtained by multiplying the attentional weight A_i with the convolutional kernel parameter K is different in each receptive-field slider, i.e., $Kernel1 \neq Kernel2 \neq Kernel3 \neq ... \neq KernelN$.

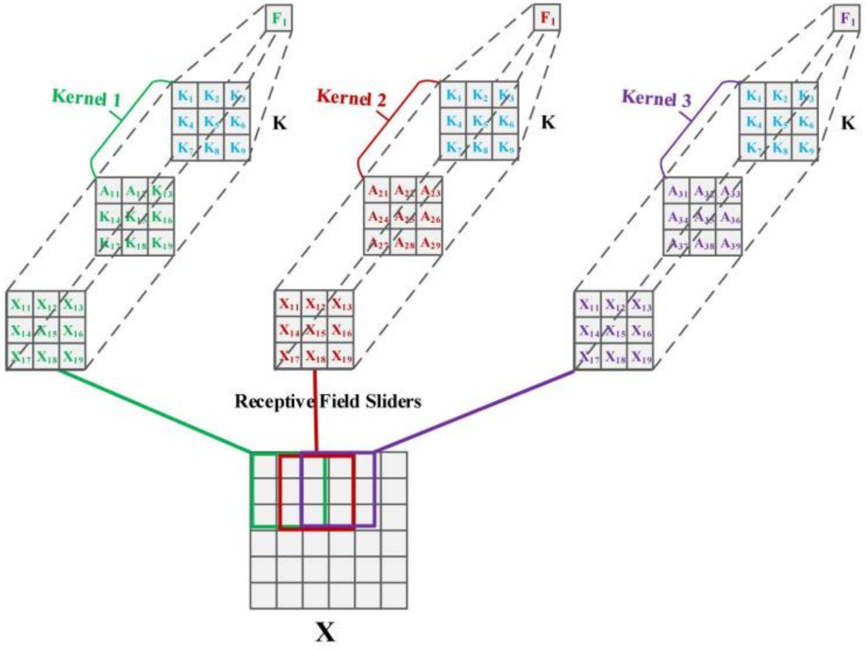


Fig. 4. It is obvious that there is an overlap of features in each receptive-field slider, which leads to the problem of sharing of attentional weights across sliders.

$$F_1 = X_1 \times A_1 \times K$$
$$F_2 = X_2 \times A_2 \times K$$
$$...$$
$$F_N = X_N \times A_N \times K$$

$$F_1 = X_{11} \times A_{11} \times K_1 + X_{12} \times A_{12} \times K_2 + X_{13} \times A_{13} \times K_3 + ... + X_{19} \times A_{19} \times K_9$$
$$F_2 = X_{21} \times A_{21} \times K_1 + X_{22} \times A_{22} \times K_2 + X_{23} \times A_{23} \times K_3 + ... + X_{29} \times A_{29} \times K_9$$
$$...$$
$$F_N = X_{N1} \times A_{N1} \times K_1 + X_{N2} \times A_{N2} \times K_2 + X_{N3} \times A_{N3} \times K_3 + ... + X_{N9} \times A_{N9} \times K_9$$

Receptive-Field Spatial Feature:

- The **receptive-field spatial feature** is specifically designed for convolutional kernels and is dynamically generated based on the kernel size
- The "**Spatial Feature"** refers to the original feature map.
- The "**Receptive-Field Spatial Feature"** is the feature map transformed by spatial features, which is composed of non-overlapping sliding windows.
- Each 3×3 size window in the receptive-field spatial feature represents a receptive-field slider

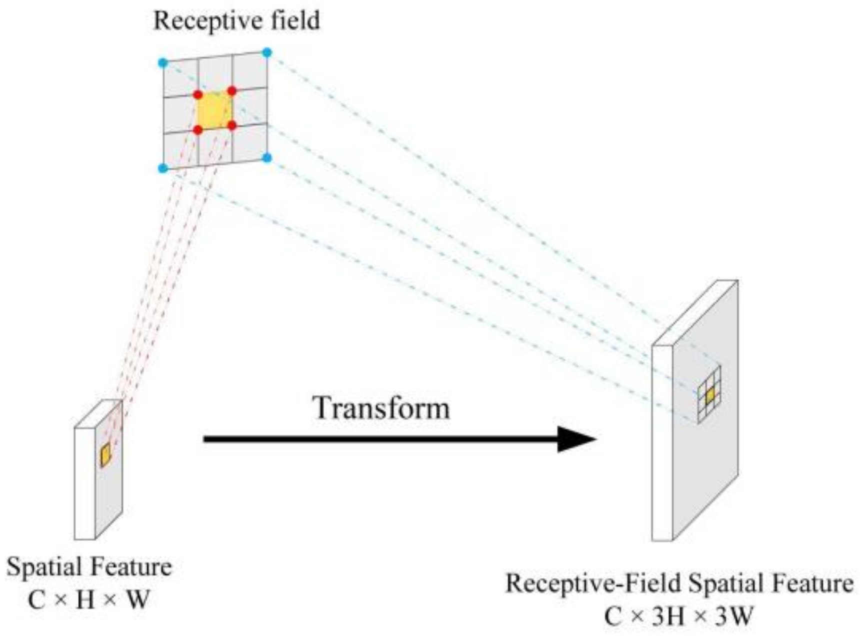
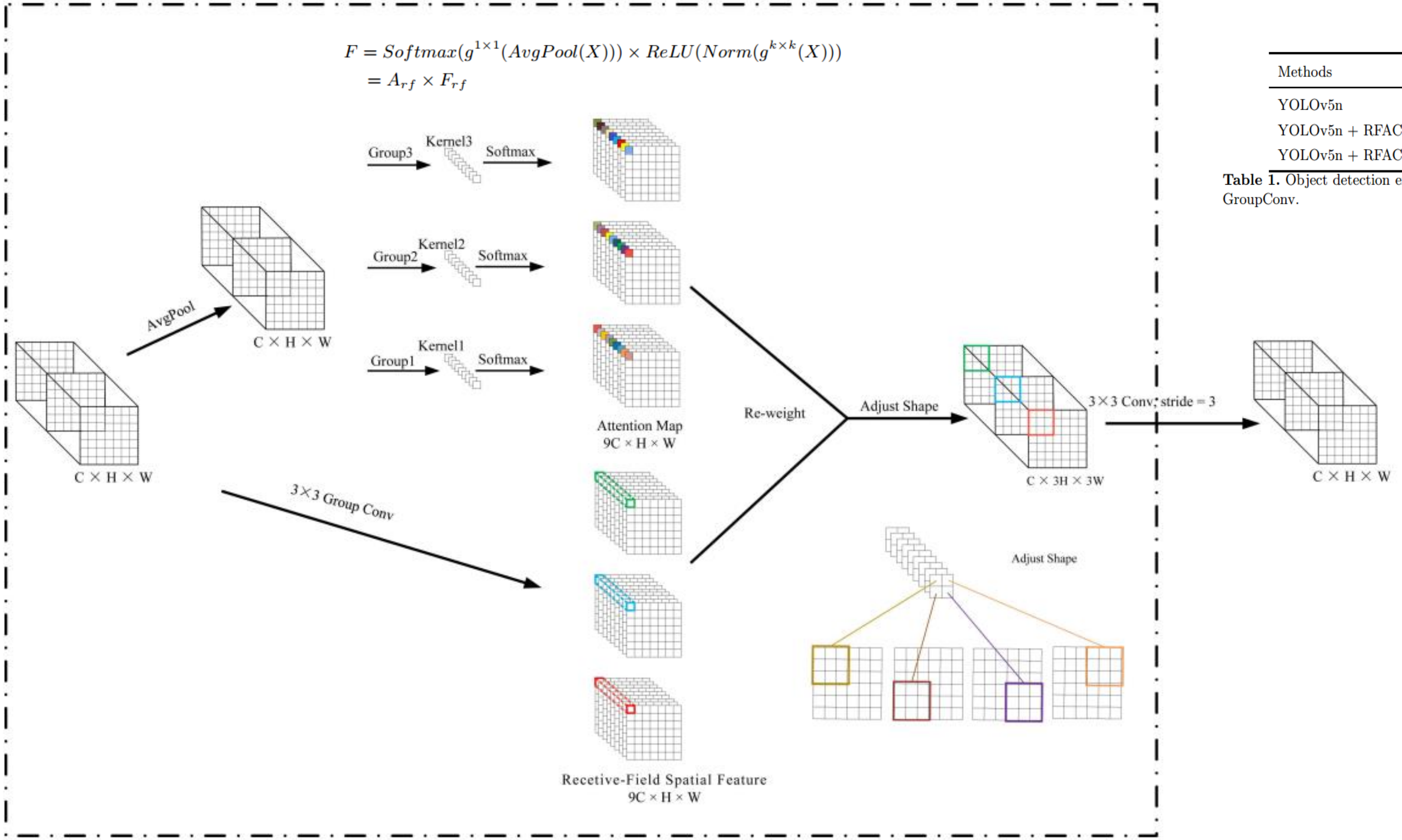


Fig. 5. The receptive-field spatial features are obtained by transforming the spatial features.

Receptive-Field Attention Convolution (RFAConv):



Methods	mAP50(%)	mAP(%)	FLOPS(G)	Param(M)	Training Time (Hours)
YOLOv5n	26.43	13.66	4.3	1.78	6.81
YOLOv5n + RFAConv (Unfold)	27.43	14.22	4.6	1.85	10.42
YOLOv5n + RFAConv (GroupConv)	27.58	14.36	4.7	1.85	7.37

Table 1. Object detection experiments based on YOLOv5n and VisDrone datasets to illustrate the advantages of RFAConv built on GroupConv.

Fig. 7. The detailed structure of RFAConv, which dynamically determines the importance of each feature in the receptive-field and solves the problem of parameters sharing.

RFCAConv and RFCBAMConv:

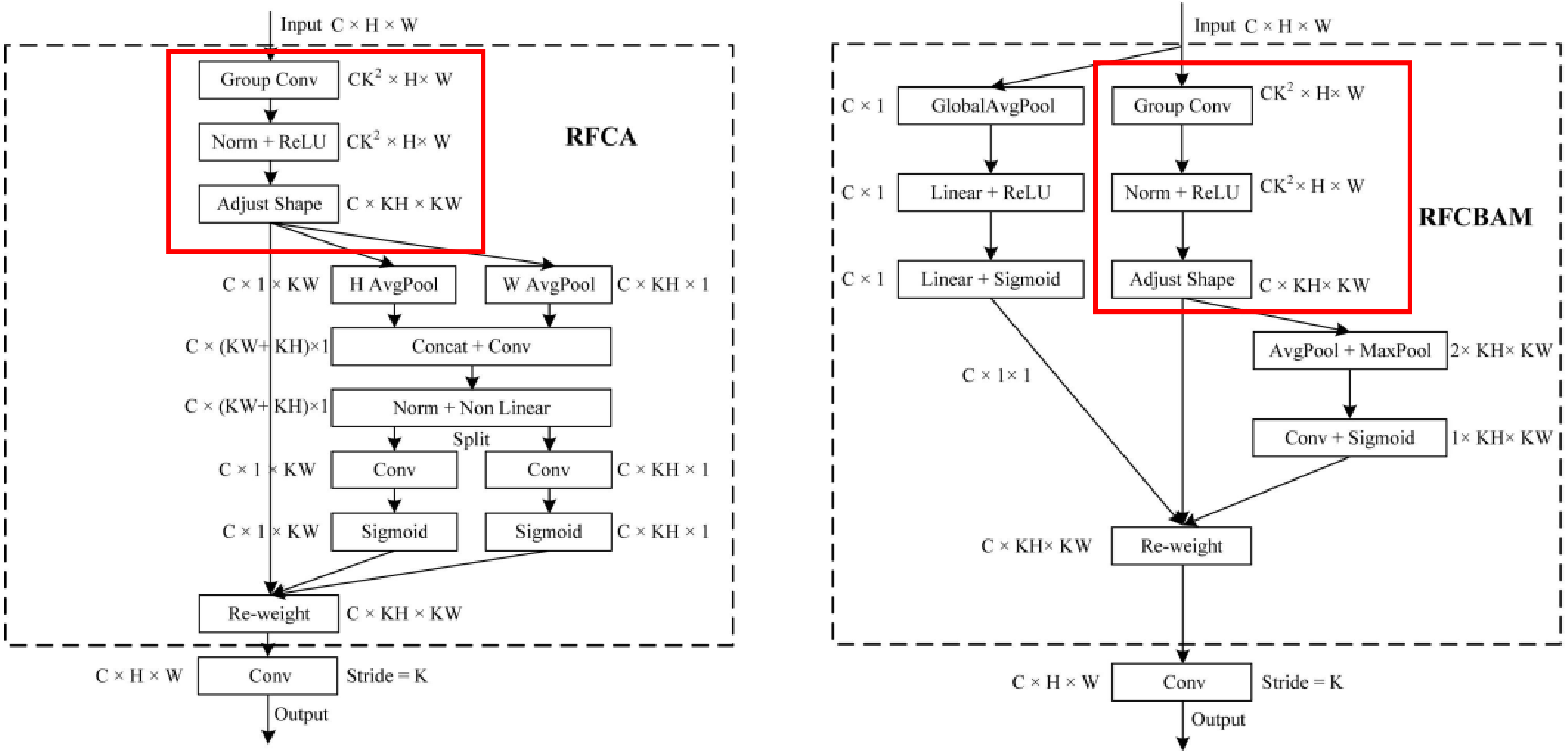


Fig. 8. Detailed structure of RFCAConv and RFCBAMConv, which focus on receptive-field spatial features. Comparing the original CBAM, We use SE attention to replace CAM in RFCBAM. Because, this can reduce computational overhead.

Experiments and Discussions

- **Experiments:**
  - Verified the effectiveness on:
    - Classification,
    - Object detection,
    - Semantic segmentation
  - The equipment for all experiments are based on RTX3090

Experiments and Discussions

Classification experiments on ImageNet-1k

Layer Name	Output Size	Resnet18	Resnet34
Conv1	112×112		
Layer1	56×56	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 3$
Layer2	28×28	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 4$
Layer3	14×14	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 6$
Layer4	7×7	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} NewConv & 3 \times 3 \\ Conv & 3 \times 3 \end{bmatrix} \times 3$
	1×1	AvgPool 1000-d	

Table 2. The Resnet18 and Resnet34 are construct by the new convolution operation.

Classification experiments on ImageNet-1k

Models	FLOPS(G)	Param(M)	Top1(%)	Top5(%)
Resnet18	1.82	11.69	69.59	89.05
+ CAMConv(r)	1.83	11.75	70.76	89.74
+ CBAMConv(r)	1.83	11.75	69.38	89.12
+ CAConv(r)	1.83	11.74	70.58	89.59
+ RFAConv(r)	1.91$^{+0.09}$	11.85$^{+0.16}$	**71.23**$^{+1.64}$	**90.29**$^{+1.24}$
Resnet34	3.68	21.80	73.33	91.37
+ CAMConv(r)	3.68	21.93	74.03	91.69
+ CBAMConv(r)	3.68	21.93	72.95	91.26
+ CAConv(r)	3.68	21.91	73.76	91.68
+ RFAConv(r)	3.84$^{+0.16}$	22.16$^{+0.66}$	**74.25**$^{+0.92}$	**92.03**$^{+0.66}$

Table 3. Classification results on ImageNet-1K using the Resnet18 and Resnet34. The different convolutional operation constructed by the attention mechanism is compared.

Models	FLOPS(G)	Param(M)	Top1(%)	Top5(%)
Resnet18	1.82	11.69	69.59	89.05
+ RFCBAMConv(r)	1.90	11.88	**72.15**	**90.71**
+ RFCAConv(r)	1.92	11.89	72.01	90.64

Table 4. RFCBAMConv and RFCAConv improve the performance of CBAMConv and CAConv. The table shows that the classification accuracy is significantly improved on ImageNet-1k.

Classification experiments on ImageNet-1k

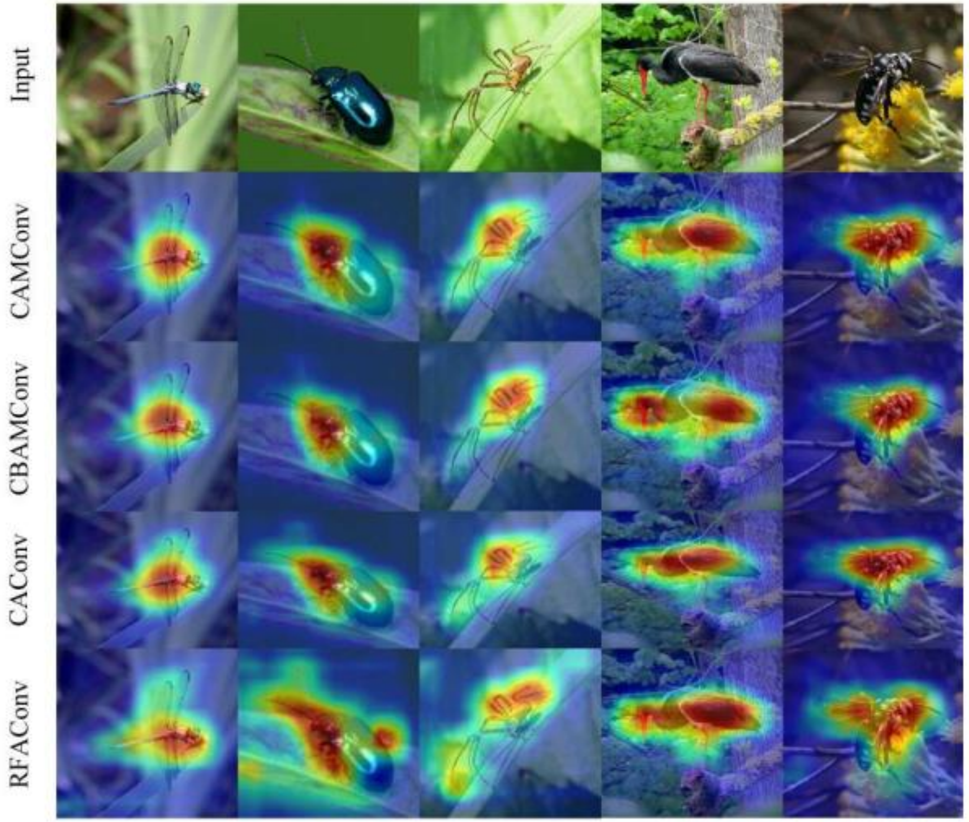


Fig. 9. Each network is built on ResNet18 based on attention convolution, and the construction process is shown in Table 2. We use Grad-CAM as our visualization tool to visualize networks without the last layer of classifiers. Compared to other attention convolution methods, our RFAConv can help the network to better recognize and highlight the key regions of objects

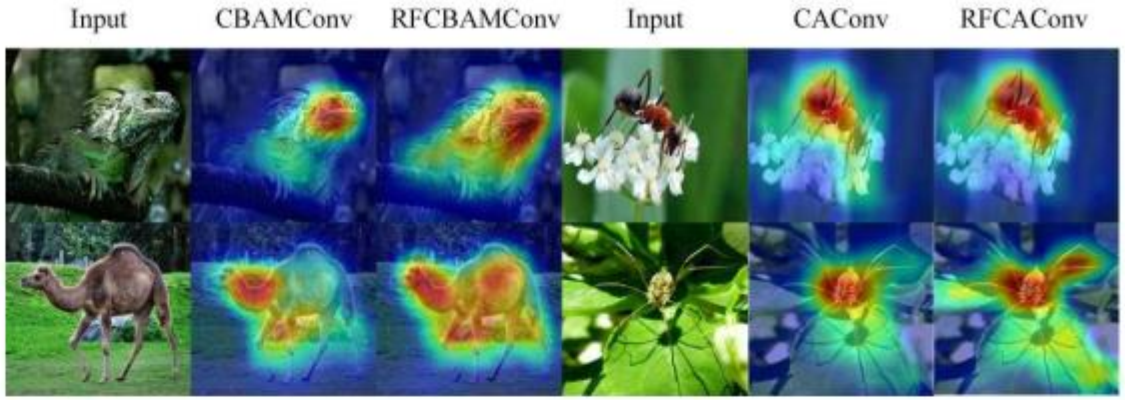


Fig. 10. We put the attention of CBAM and CA into the receptive-field spatial features and improve them to obtain RFCBAM and RFCA. Then RFCBAMConv and RFCAConv are constructed by the same method as RFA. As in Fig. 10, we visualize the different networks separately. Obviously, compared to CBAMConv and CAConv, the improved obtained RFCBAMConv and CAConv can help the network to better recognize and highlight the key regions of objects.

Experiments and Discussions

Object detection experiments on COCO2017

Models	FLOPS(G)	Param(M)	AP_{50}(%)	AP_{75}(%)	AP(%)	AP_S(%)	AP_M(%)	AP_L(%)	Time(ms)
YOLOv5n	4.5	1.8	45.6	28.9	27.5	13.5	31.5	35.9	4.4
+ CAMConv(r)	4.5	1.8	45.6	28.3	27.4	13.8	31.4	35.8	5.2
+ CBAMConv(r)	4.5	1.8	45.5	28.6	27.6	13.6	31.2	36.6	5.4
+ CAConv(r)	4.5	1.8	46.2	29.2	28.1	14.3	32	36.6	4.8
+ RFAConv(r)	4.7	1.9	**47.3**	**30.6**	**29**	**14.8**	**33.4**	**37.4**	5.3
YOLOv7-tiny	13.7	6.2	53.8	38.3	35.9	19.9	39.4	48.8	6.8
+ RFAConv(r)	14.1	6.3	**55.1**	**40.1**	**37.1**	**20.9**	**41.1**	**50**	8.4
YOLOv8n	8.7	3.1	51.9	39.7	36.4	18.4	40.1	52	4.2
+ CAMConv(r)	8.8	3.1	51.6	39	36.2	18	39.9	51.2	4.5
+ CBAMConv(r)	8.8	3.1	51.5	39.6	36.3	18.3	40.1	51.5	4.6
+ CAConv(r)	8.8	3.1	52.1	39.9	36.7	17.8	40.3	51.6	4.3
+ RFAConv(r)	9.0	3.2	53.4	41.1	37.7	18.9	41.8	52.7	4.5
+ RFCAConv(r)	9.1	3.2	**53.9**	**41.7**	**38.2**	**19.7**	**42.3**	**53.5**	4.7

Table 5. Object detection AP_{50}, AP_{75}, AP, AP_S, AP_M, and AP_L on the COCO2017 validation sets. We adopt the YOLOv5n, YOLOv7-tiny, and YOLOv8n detection framework and replace the original convolution with the novel convolutioanl operation constructed by attention mechanism.

COCO2017 contains 118,287 training set and 5,000 verification set

Object detection experiments on VOC7+12

Models	FLOPS(G)	Param(M)	mAP(%)	Time(ms)
YOLOv5n	4.2	1.7	41.5	2.7
+ CAMConv(r)	4.2	1.7	41.4	2.9
+ CBAMConv(r)	4.3	1.7	41.9	3
+ CAConv(r)	4.3	1.7	42.4	3
+ RFAConv(r)	4.5	1.8	**43.3**	3
YOLOv5s	15.9	7.1	48.9	3
+ CAMConv(r)	16	7.1	48.5	3.5
+ CBAMConv(r)	16	7.1	49	3.7
+ CAConv(r)	16.1	7.1	49.6	3.1
+ RFAConv(r)	16.4	7.2	50	5.1
+ RFCBAMConv(r)	16.4	7.2	50.1	3.9
+ RFCAConv(r)	16.6	7.2	**51**	4.4
YOLOv7-tiny	13.2	6.1	50.2	5
+ CAMConv(r)	13.2	6.1	50.3	5.4
+ CBAMConv(r)	13.2	6.1	50.1	5.4
+ CAConv(r)	13.2	6.1	50.5	5.4
+ RFAConv(r)	13.6	6.1	**50.6**	7.5
YOLOv8n	8.1	3	53.5	3
+ CAMConv(r)	8.1	3	52.8	3.1
+ CBAMConv(r)	8.2	3	53.3	3.1
+ CAConv(r)	8.2	3	53.8	2.9
+ RFAConv(r)	8.4	3.1	**54**	3.2

Table 6. Object detection mAP50 and mAP on the VOC7+12 validation set.

VOC consist of 16,551 training set and 4,952 validation set

Semantic segmentation

Backbone	Stride	MIOU(%)
Resnet18	8	58.9
+ CAMConv(r)	8	60.9
+ CBAMConv(r)	8	59.3
+ CAConv(r)	8	62.1
+ RFAConv(r)	8	60.8
+ RFCBAMConv(r)	8	62.1
+ RFCAConv(r)	8	**63.9**

Resnet18	16	64.6
+ CAMConv(r)	16	65.5
+ CBAMConv(r)	16	63.6
+ CAConv(r)	16	66.6
+ RFAConv(r)	16	65.4
+ RFCBAMConv(r)	16	67.7
+ RFCAConv(r)	16	**68.0**

Table 7. Results of experiments comparing different the novel convolutional operation based on DeepLabPlusV3.

VOC consist of 16,551 training set and 4,952 validation set

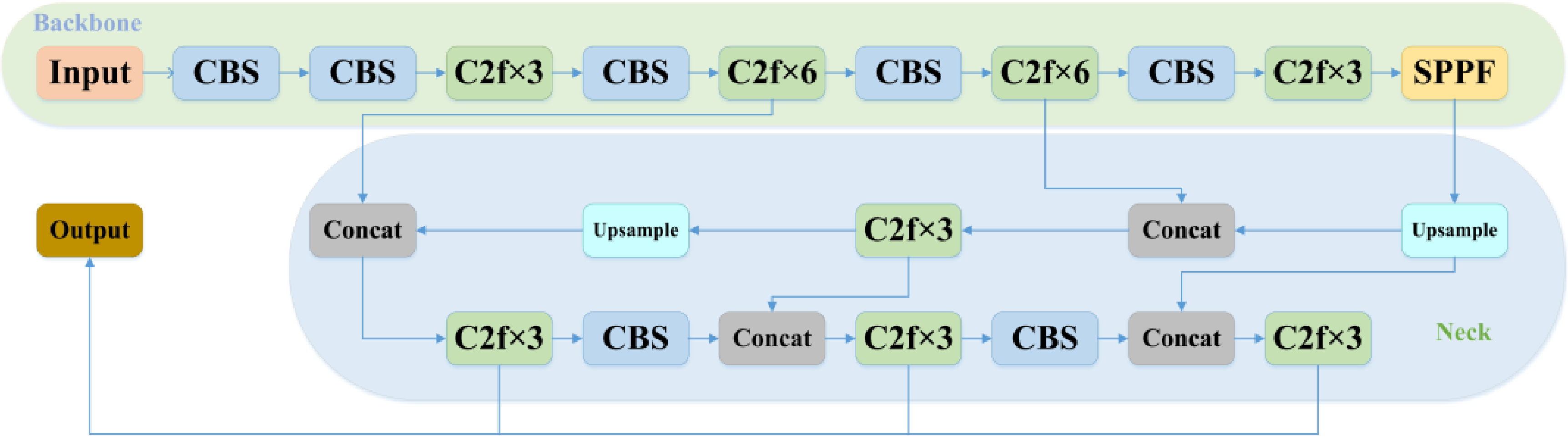
Conclusions

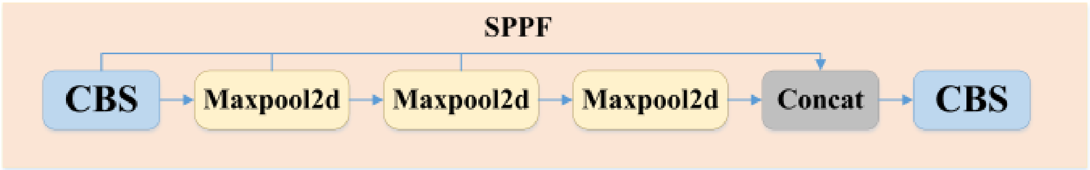
- This paper proposed a novel attention mechanism called RFA and devise a novel convolution operation, which improves CNN network performance.
- Emphasized the importance of directing attention to the receptive-field spatial feature to enhance network performance.
- **Opinions:**
  - Review the overview of Spatial Attention algorithms, Convolution Operation, their advantages, and disadvantages
  - Lacks the comparative results with SOTA in each dataset
  - Leverage this idea for the Tomatoes Detection topic based on YOLOv8
  - Try to implement and improve the RFA with light-weight convolutional techniques

Thank you for your attention!

Apendix

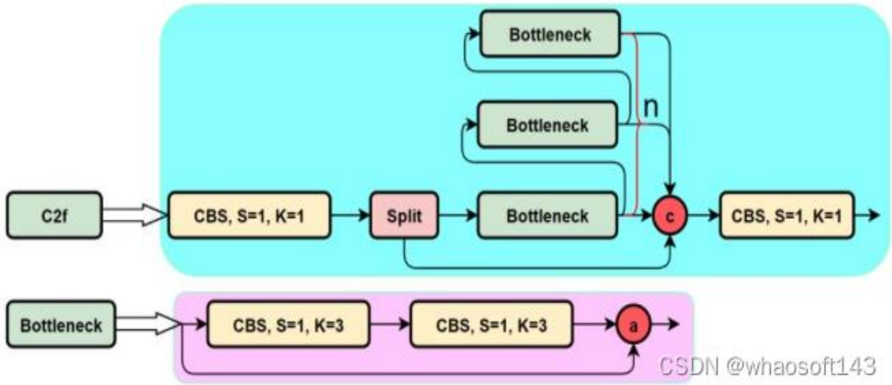
YOLOv8 architecture





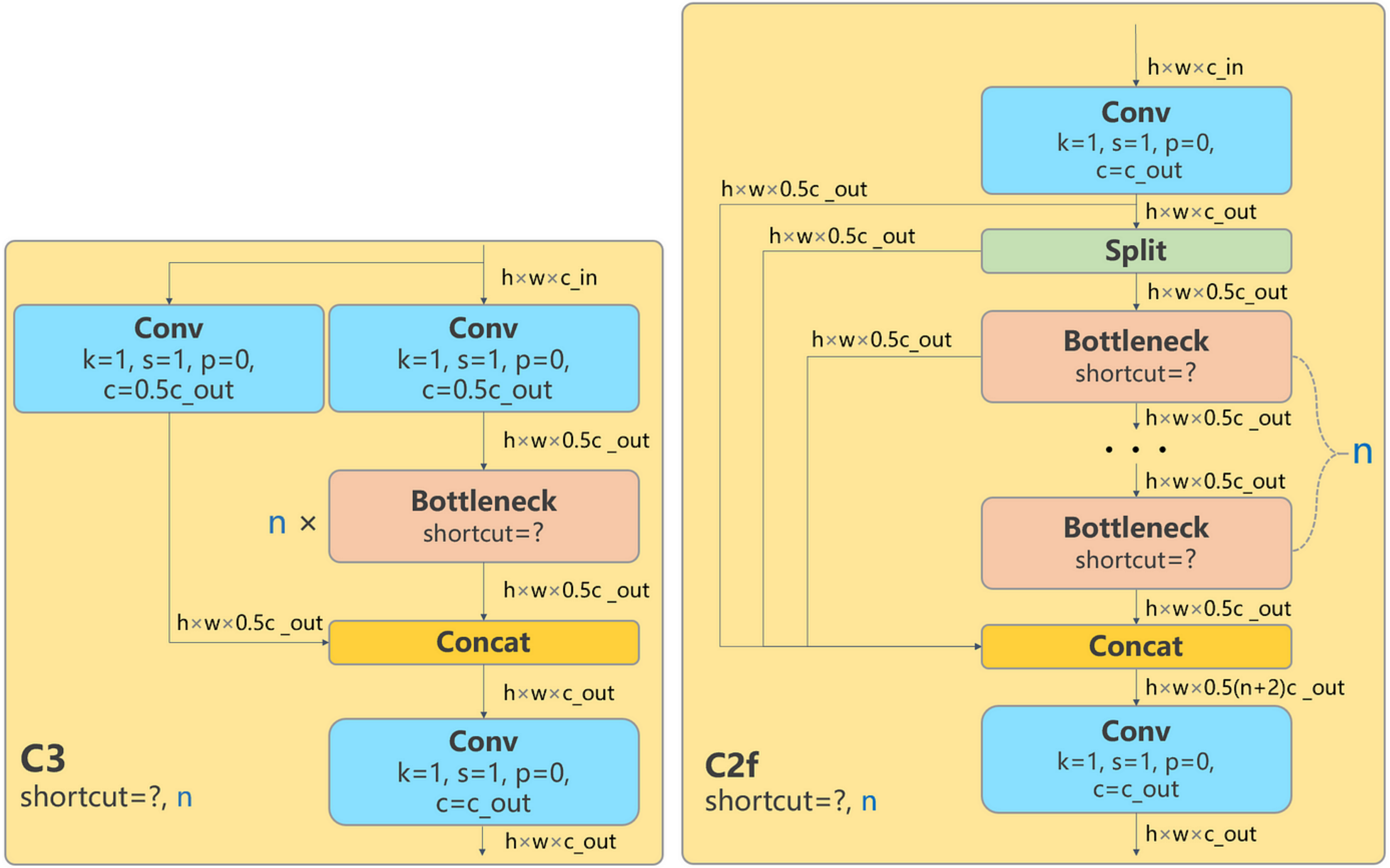
- **Backbone:**
  - CSP (Cross Stage Partial Network)
  - SPPF (Spatial Pyramid Pooling Fast)
  - C3 is replaced by C2f
- **Neck:** PAN (Path Aggregation Network)
  - Remove CONV structure
  - C3 is replaced by C2f
- **Detection head:** Free anchor
- **Loss function:**
  - Classification: BCE (Binary Cross Entropy) Loss
  - Regression: DFL (Distribution Focal Loss) + CIoU (Complete-IoU) Loss

C2f



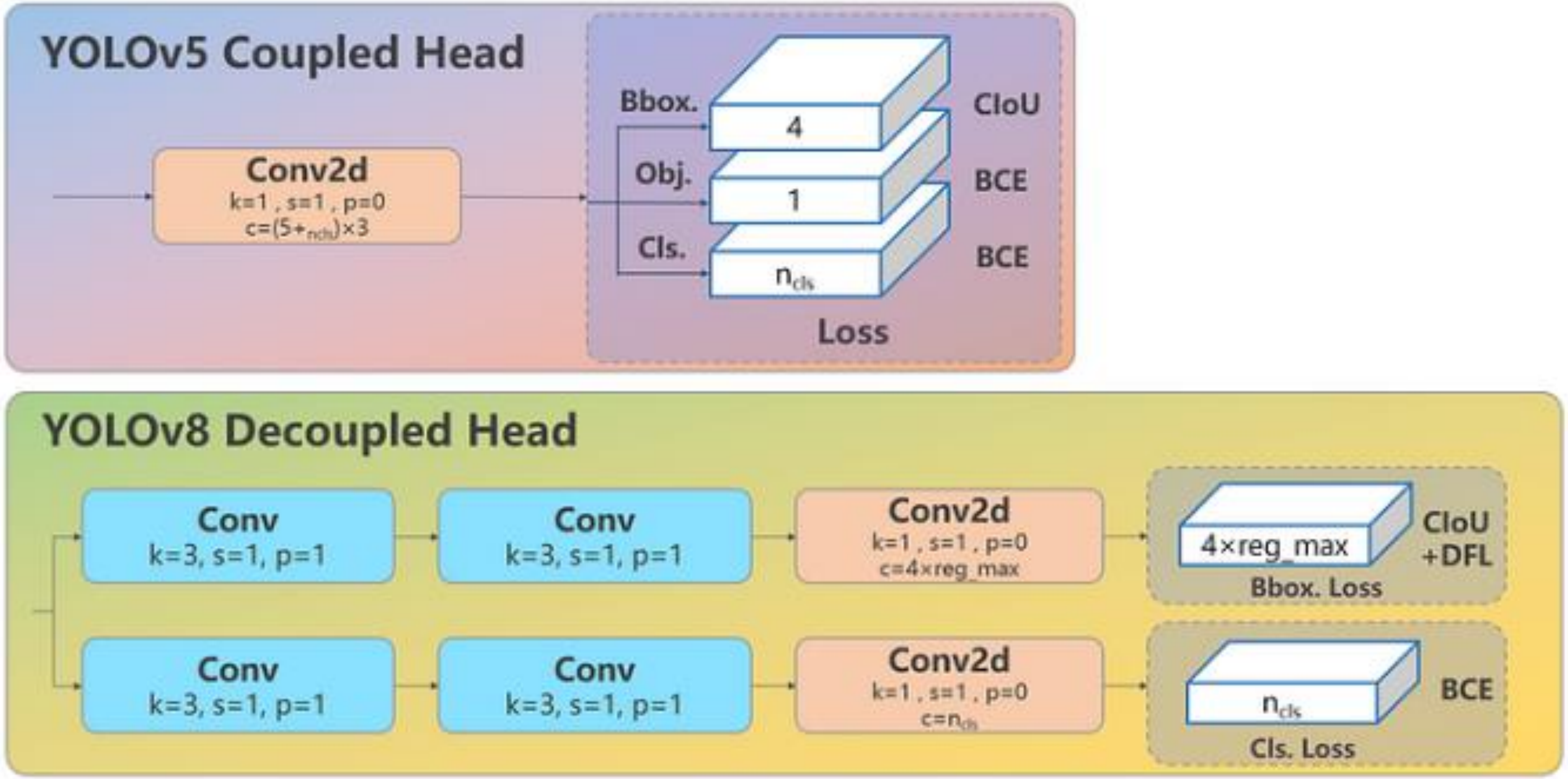
https://blog.csdn.net/qq_29788741/article/details/128626422

C3 vs C2f



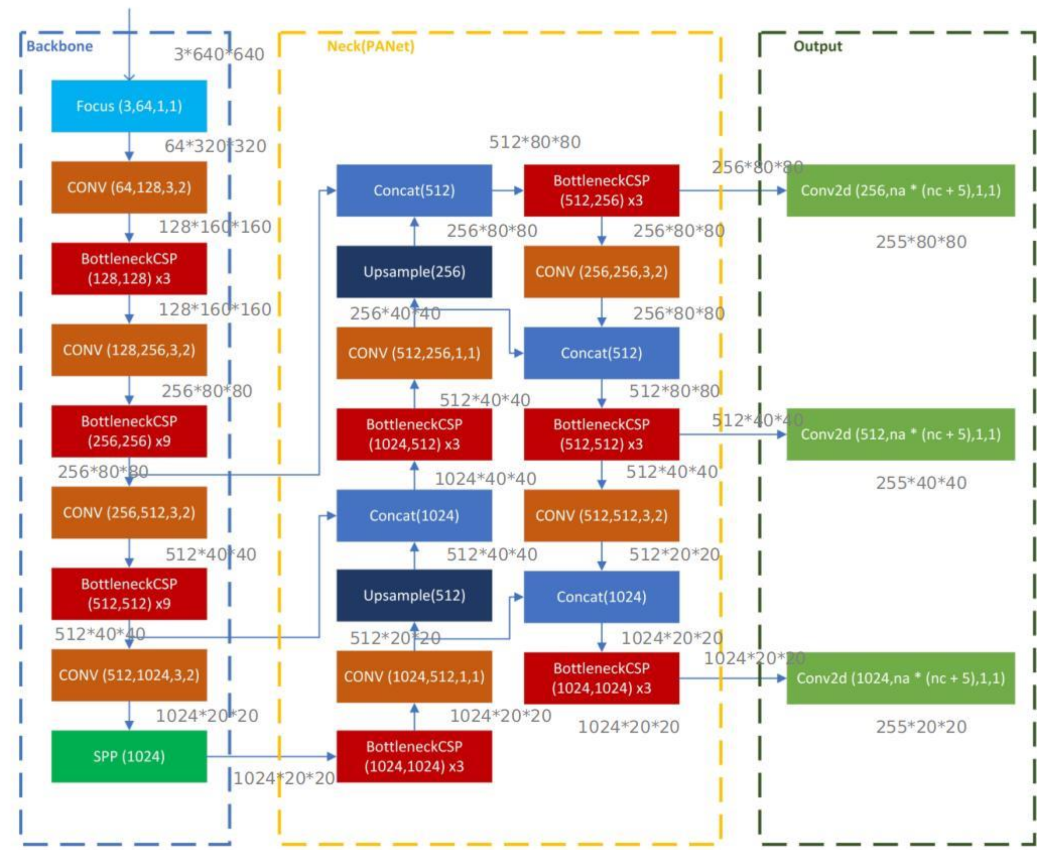
https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

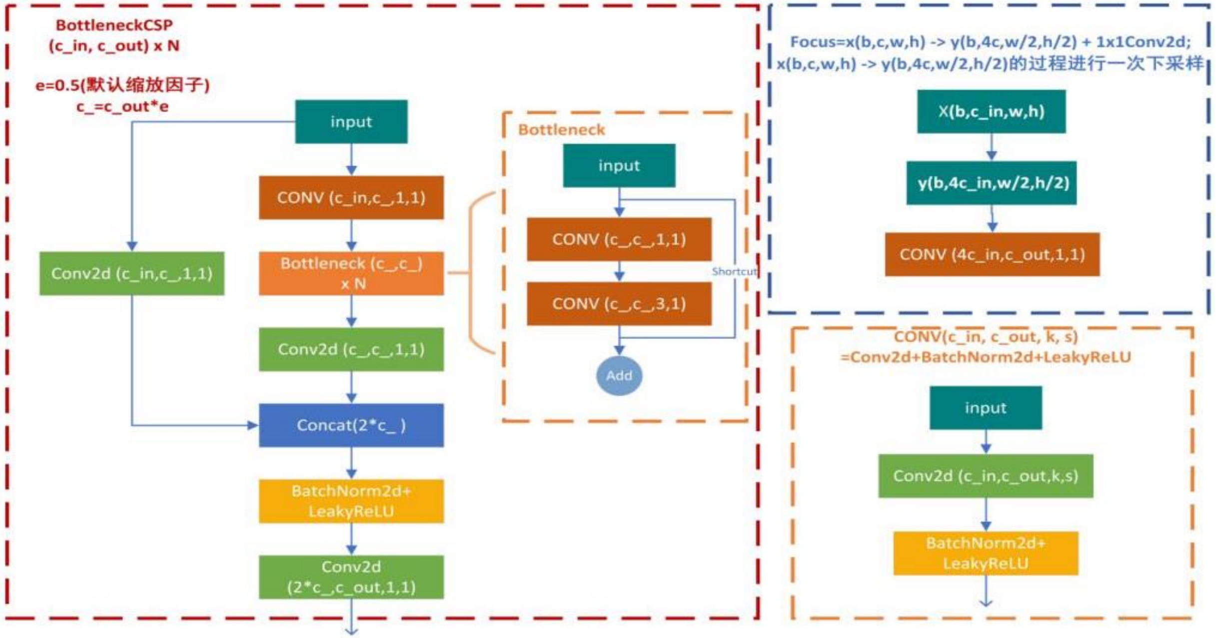
Detection heads

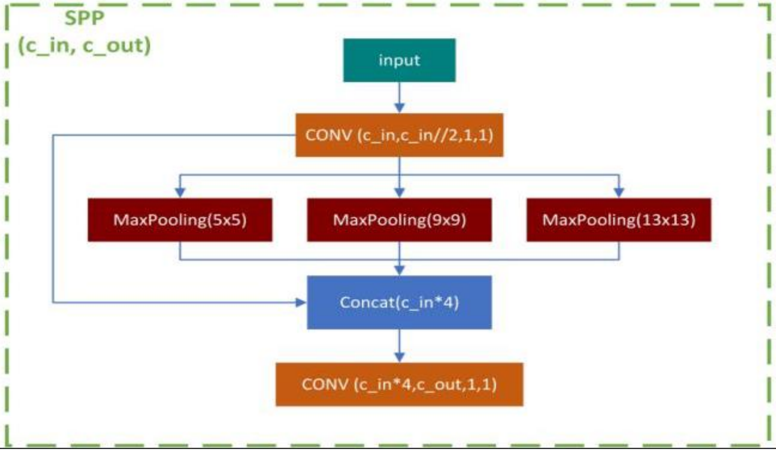


https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

Review YOLOv5 architecture



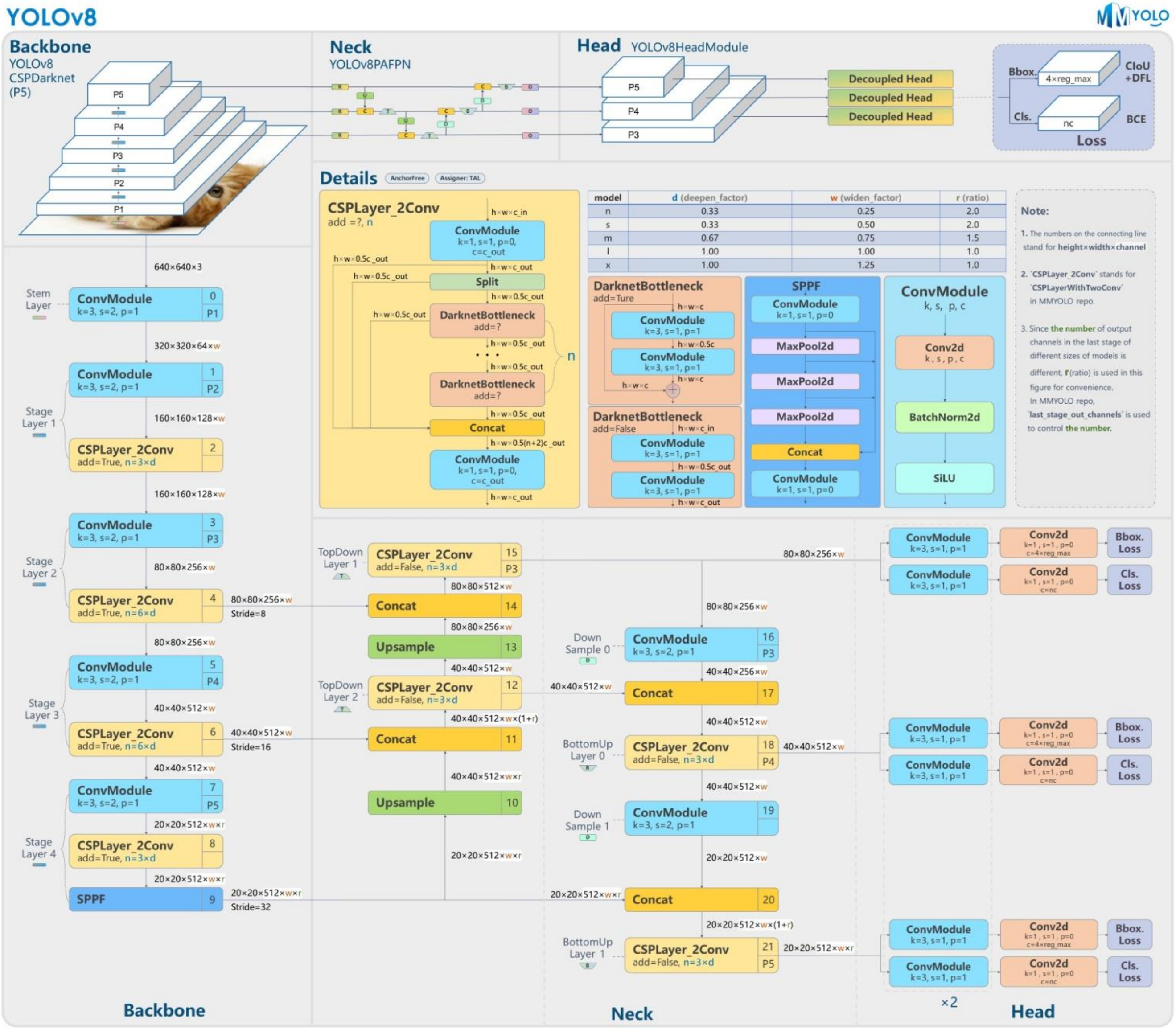




- **Three parts:**
  - Backbone: CSPDarknet53+ SPP layer
    - CSP (Cross Stage Partial Network)
    - SPP (Spatial Pyramid Pooling)
  - Neck: PANet (Path Aggregation Network)
  - YOLO detection head

https://github.com/ultralytics/yolov5/issues/6998#41

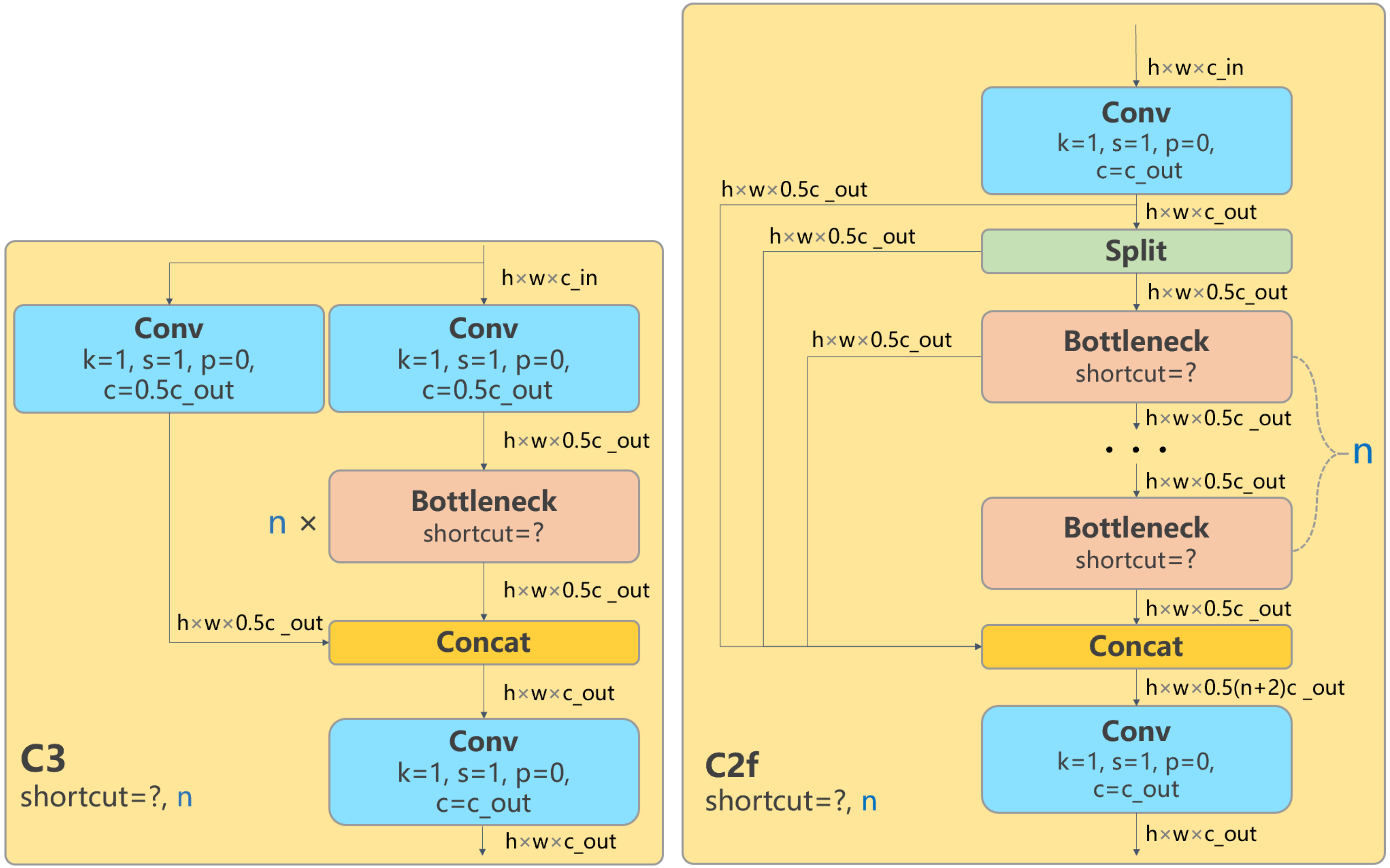
Review YOLOv8 architecture



model	d (deepen_factor)	w (widen_factor)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

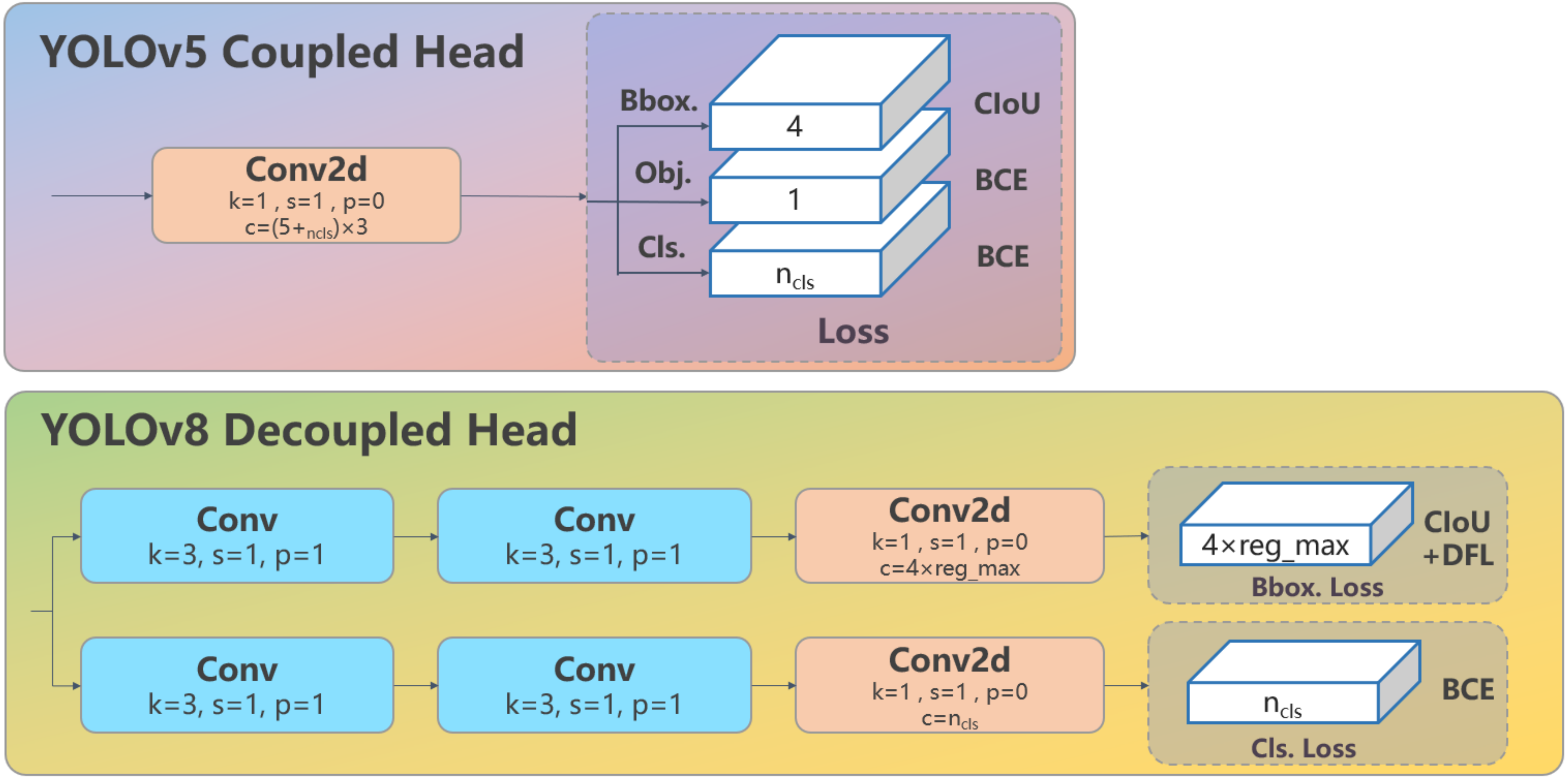
https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

C3 vs C2f



https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

Detection head



https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

YOLOv8 innovation and improvement points:

1. **Backbone:** The idea of CSP is still used , but the C3 module in YOLOv5 is replaced by the C2f module to achieve further lightweight, and YOLOv8 still uses the SPPF module used in YOLOv5 and other architectures
2. **PAN-FPN:** There is no doubt that YOLOv8 still uses the idea of PAN, but by comparing the structure diagrams of YOLOv5 and YOLOv8, we can see that YOLOv8 deletes the convolution structure in the PAN-FPN upsampling stage in YOLOv5, and also replaces the C3 module with C2f module
3. **Decoupled-Head:** Did you smell something different? Yes, YOLOv8 went to Decoupled-Head;
4. **Anchor-Free:** YOLOv8 abandoned the previous Anchor-Base and used the idea of Anchor-Free;
5. **Loss function:** YOLOv8 uses BCE Loss as classification loss and DFL Loss+CIOU Loss as regressiontion loss

https://github.com/akashAD98/yolov8_in_depth

- **Classification Loss:** BCE (Binary Cross Entropy) Loss

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

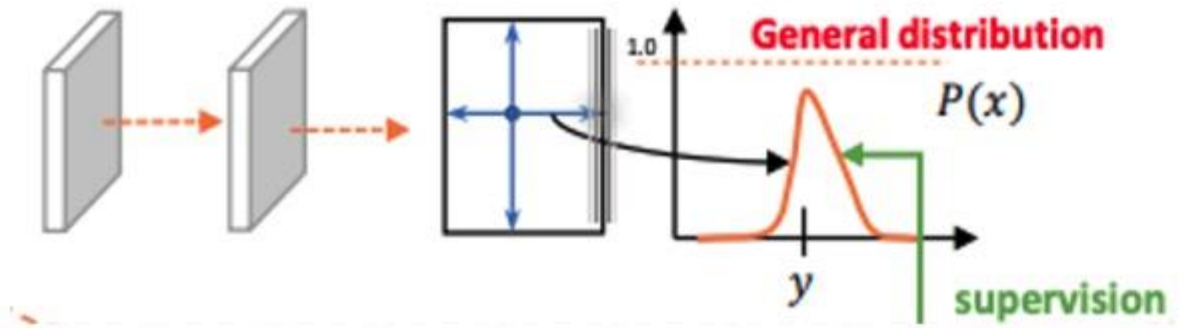
  - y_i is GT label
  - y_i_hat is predicted label

https://paperswithcode.com/method/varifocal-loss

- **Regression Loss**: DFL (Distribution Focal Loss)

$$DFL_{(s_i,s_{i+1})} = -((y_{i+1} - y)\log(s_i) + (y - y_i)\log(s_{i+1}))$$

- s is probability
- y is a label
- y_i and y_{i+1} are interval orders ($y_i \le y \le y_{i+1}$)



https://blog.csdn.net/qq_29788741/article/details/128626422

Regression Loss: CIoU

The CIoU loss function:

$$L_{CIoU} = 1 - IoU + \frac{d^2}{C^2} + \alpha v$$

where,

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

α is a trade-off parameter and is defined as:

$$\alpha = \frac{v}{(1-IoU)+v}$$

α is a function of **IoU**. The above equation states that the **aspect ratio factor is less important in the case of no overlap and more important in the case of more overlap**.

https://learnopencv.com/iou-loss-functions-object-detection/

RFAConv code

```python
class RFAConv(nn.Module):
    def __init__(self,in_channel,out_channel,kernel_size,stride=1):
        super().__init__()
        self.kernel_size = kernel_size

        self.get_weight = nn.Sequential(nn.AvgPool2d(kernel_size=kernel_size, padding=kernel_size // 2, stride=stride),
                                        nn.Conv2d(in_channel, in_channel * (kernel_size ** 2), kernel_size=1, groups=in_channel,bias=False))
        self.generate_feature = nn.Sequential(
            nn.Conv2d(in_channel, in_channel * (kernel_size ** 2), kernel_size=kernel_size,padding=kernel_size//2,stride=stride, groups=in_channel, bias=False),
            nn.BatchNorm2d(in_channel * (kernel_size ** 2)),
            nn.ReLU())

        self.conv = nn.Sequential(nn.Conv2d(in_channel, out_channel, kernel_size=kernel_size, stride=kernel_size))

    def forward(self,x):
        b,c = x.shape[0:2]
        weight =  self.get_weight(x)
        h,w = weight.shape[2:]
        weighted = weight.view(b, c, self.kernel_size ** 2, h, w).softmax(2)
        feature = self.generate_feature(x).view(b, c, self.kernel_size ** 2, h, w)
        weighted_data = feature * weighted
        conv_data = rearrange(weighted_data, 'b c (n1 n2) h w -> b c (h n1) (w n2)', n1=self.kernel_size,
                              n2=self.kernel_size)
        return self.conv(conv_data)
```

Unfold method

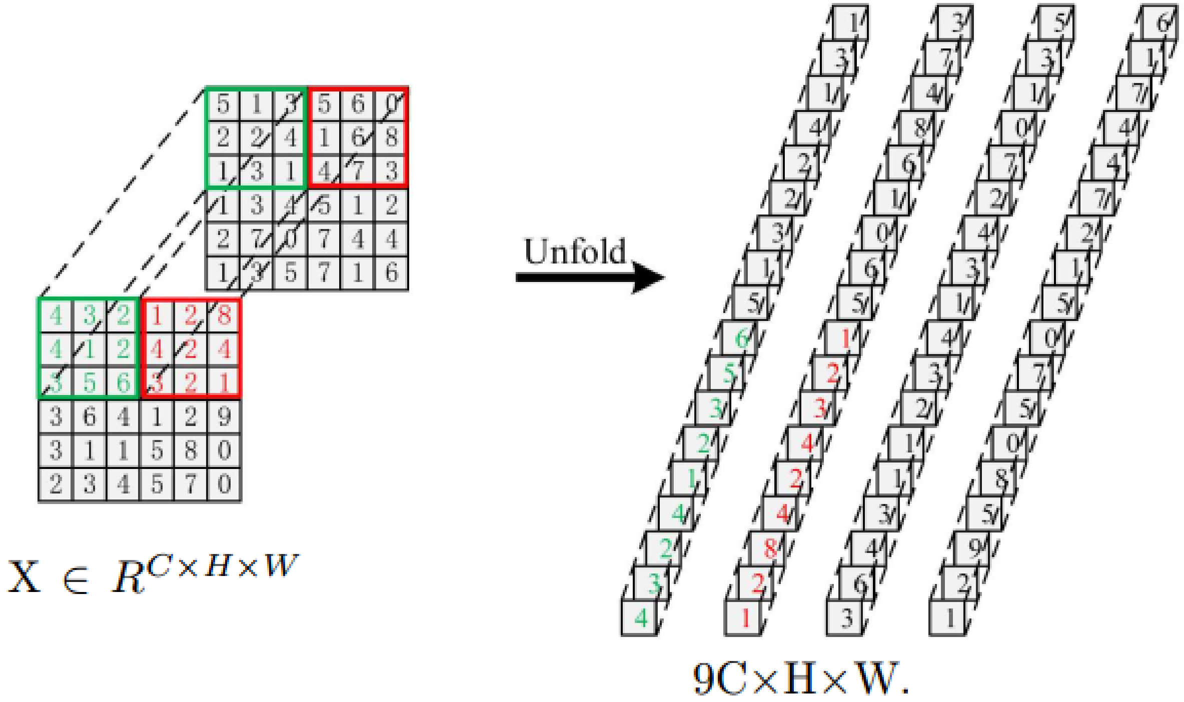
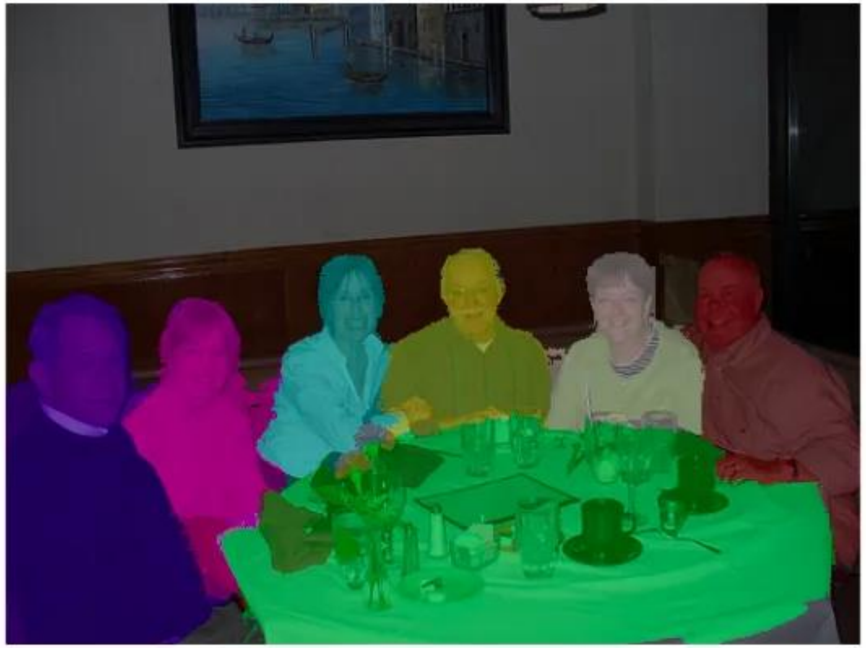


Fig. 6. In the figure, it shows in detail an example of extracting 3×3 receptive-field spatial features by the Unfold method. For the convenience of display, we set the step size to 3.

Semantic segmentation vs Instance segmentation

There are different types of segmentation techniques, including semantic segmentation which assigns a class label to each pixel in the image, and instance segmentation which not only assigns class labels but also distinguishes between different instances of the same class (e.g. different cars in an image).

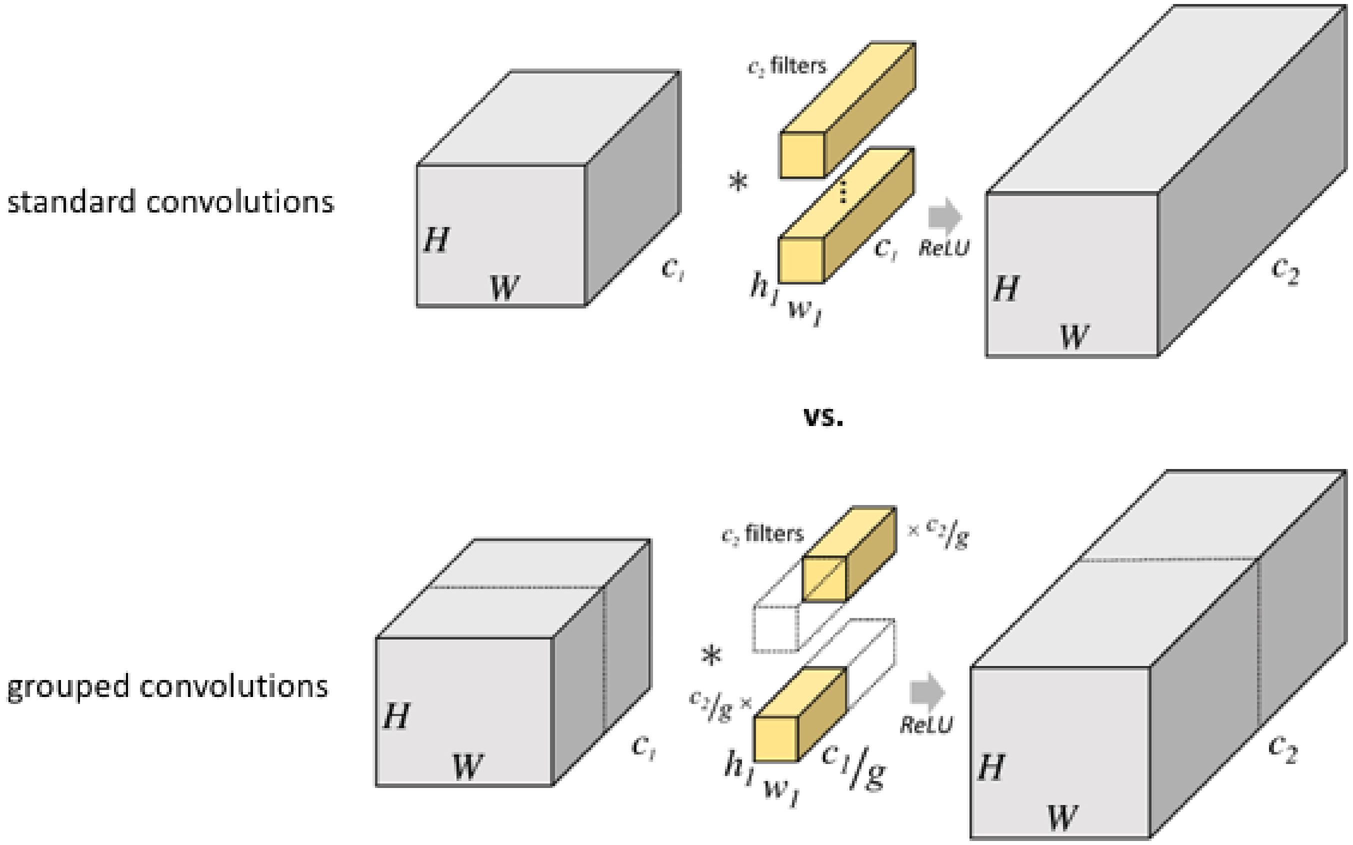
Semantic Segmentation



Instance Segmentation

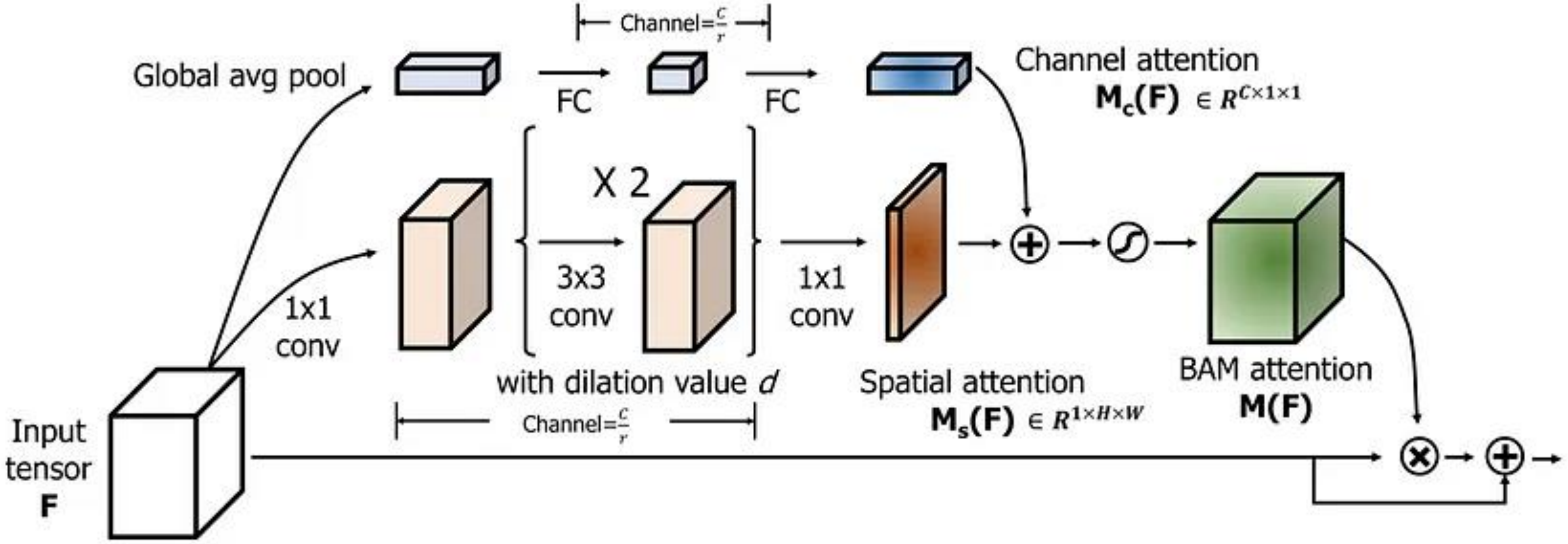
https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/

Standard convolution vs Group convolution



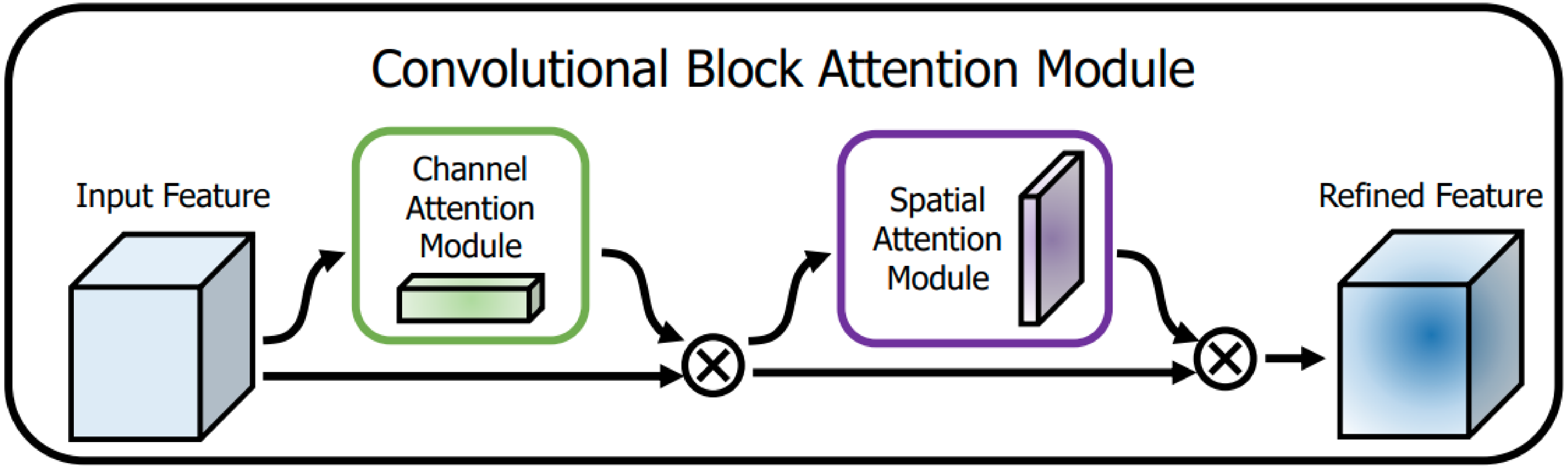
https://www.jeremyjordan.me/convnet-architectures/

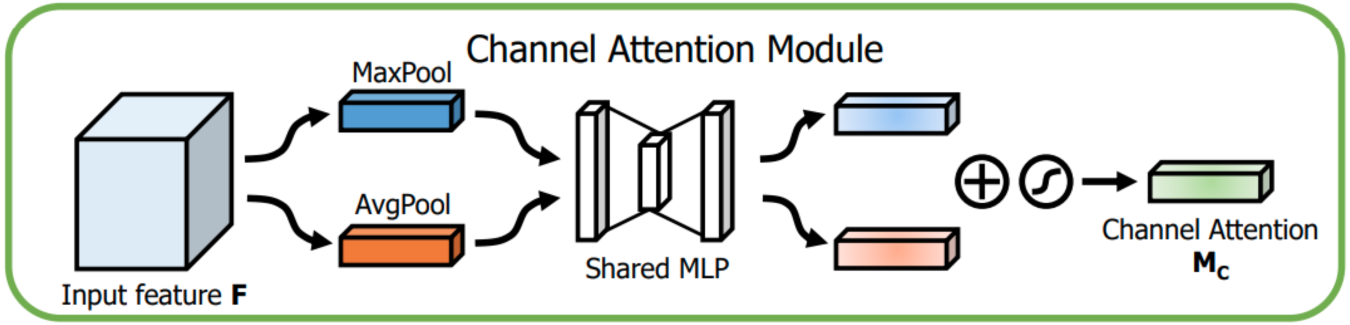
BAM: Bottleneck Attention Module

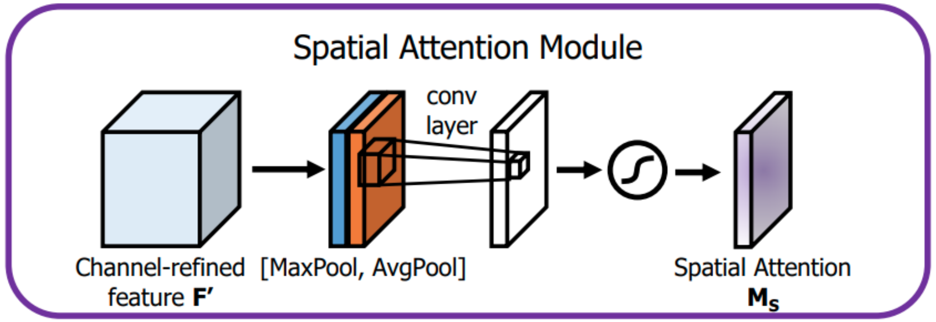


https://medium.com/lunit/bam-and-cbam-self-attention-modules-for-cnn-585e042c607f

CBAM: Convolutional Block Attention Module

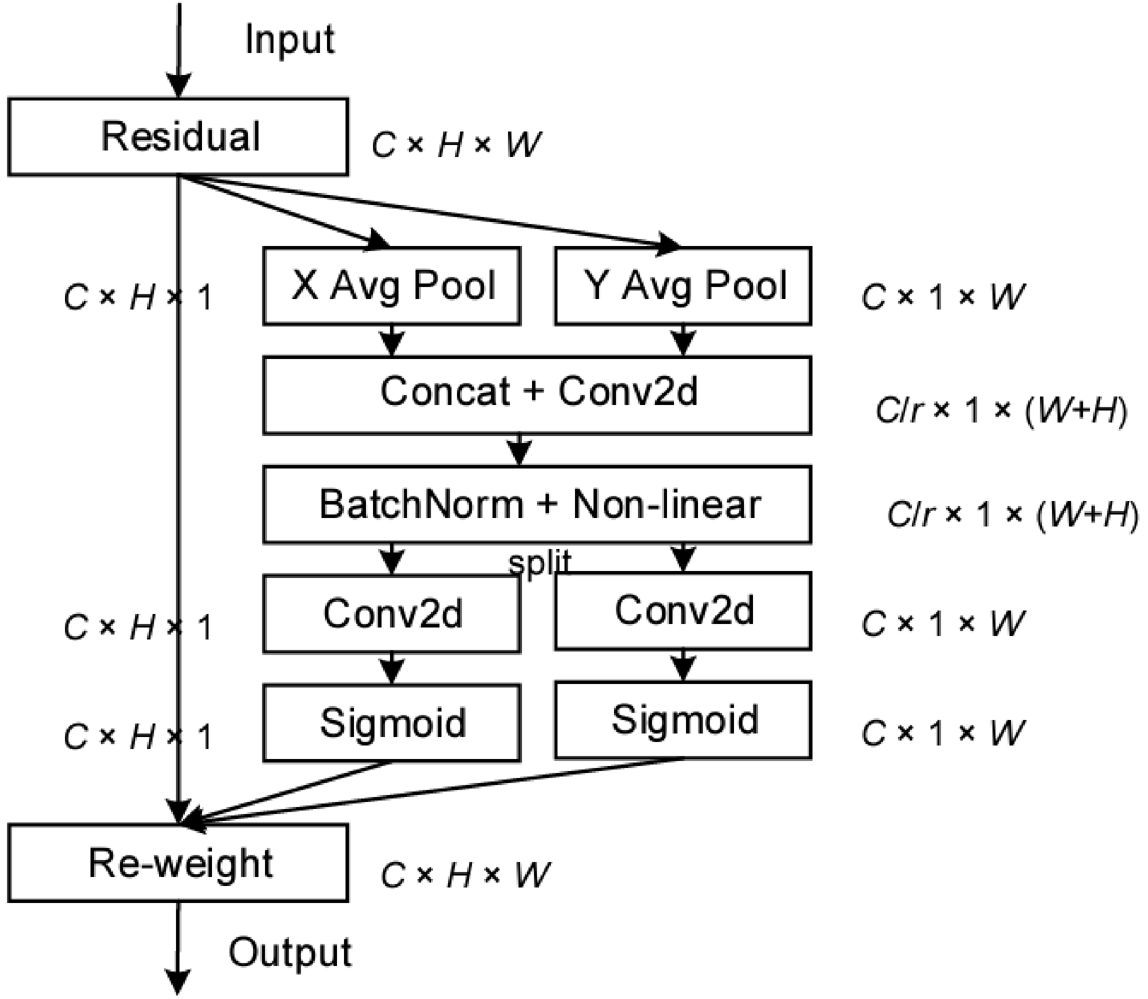






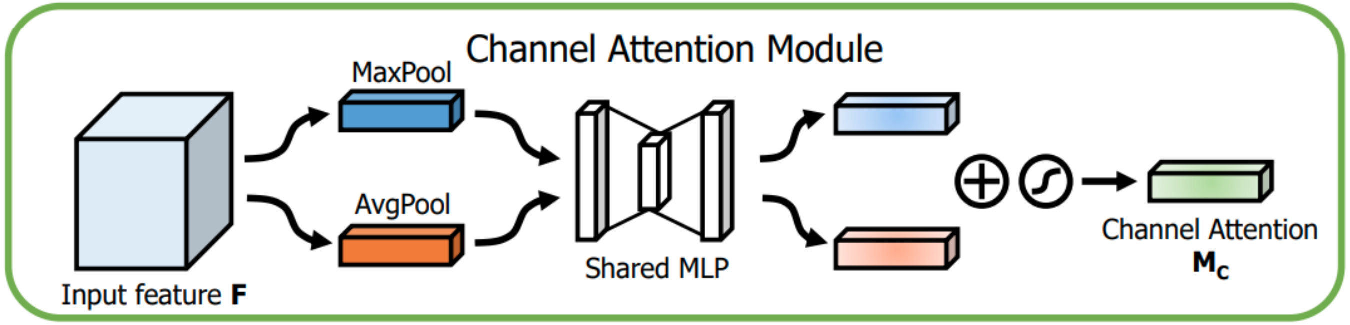
https://doi.org/10.48550/arXiv.1807.06521

CA: Coordinate Attention Module



https://medium.com/@hichengkang/paper-review-coordinate-attention-for-efficient-mobile-network-design-52ff23074e04

CAM: Channel Attention Module



https://doi.org/10.48550/arXiv.1807.06521

Task Aligned Assigner for matching strategy

Task Aligned Assigner

$$t = s^{\alpha} \times u^{\beta}$$

- s is the classification score
- u is the IOU value
- α and β are the weight hyperparameters

https://learnopencv.com/iou-loss-functions-object-detection/

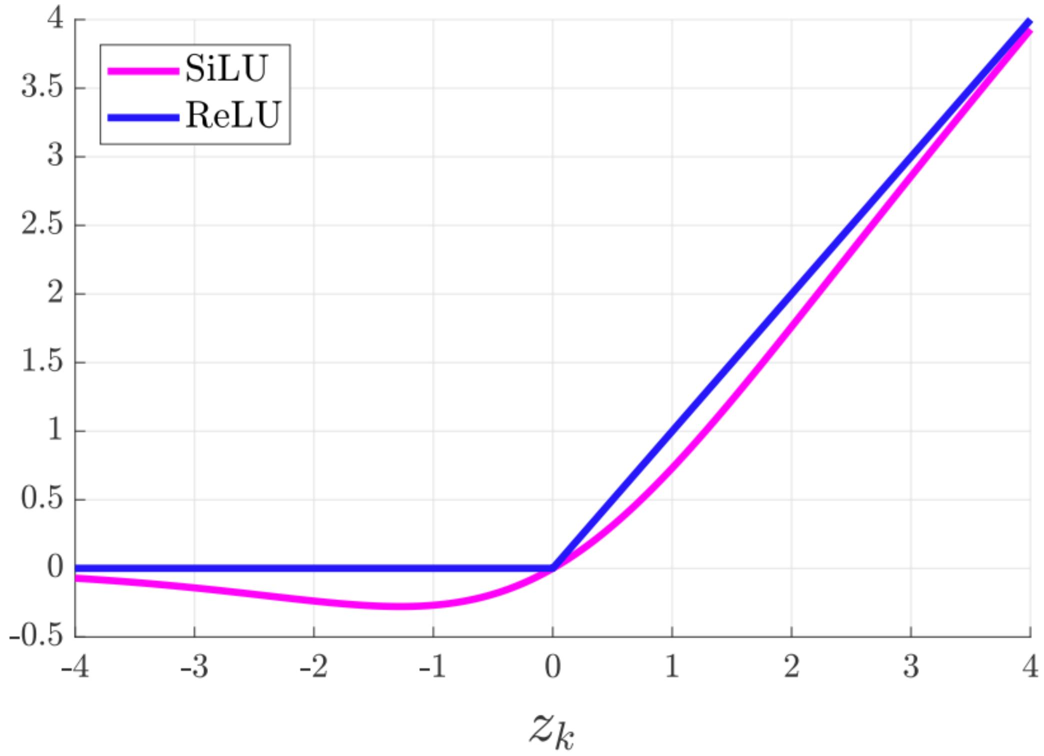
Classification Loss: VFL (VariFacal Loss)

$$VFL(p,q) = -q(q\log(p) + (1-q)\log(1-p)) \text{ if } q > 0$$

$$VFL(p,q) = -\alpha p^{\gamma}\log(1-p)$$

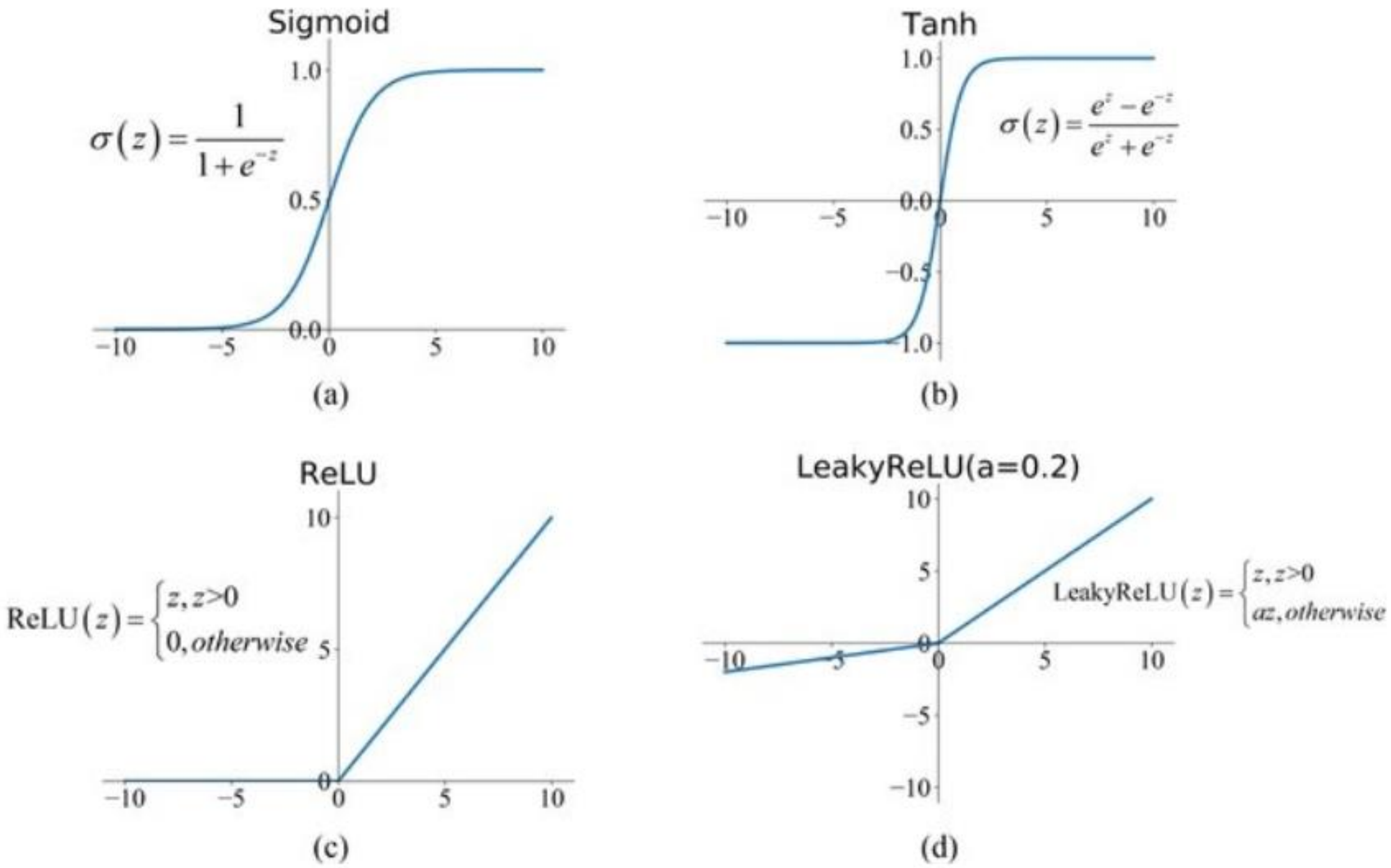
https://paperswithcode.com/method/varifocal-loss

SiLU Activation



$$SiLU(x) = x * \sigma(x) = x * \frac{1}{1 + e^{-x}}$$

Activation function



Filter out unsuitable labels during training

glenn-jocher commented on Jan 9, 2022 · Member

@alicera label candidate criteria are applied to filter out unsuitable labels during training here:

yolov5/utils/augmentations.py
Lines 272 to 277 in 6865d19

```
272  def box_candidates(box1, box2, wh_thr=2, ar_thr=100, area_thr=0.1, eps=1e-16):  # box1(4,n), box2(4,n
273      # Compute candidate boxes: box1 before augment, box2 after augment, wh_thr (pixels), aspect_ratio
274      w1, h1 = box1[2] - box1[0], box1[3] - box1[1]
275      w2, h2 = box2[2] - box2[0], box2[3] - box2[1]
276      ar = np.maximum(w2 / (h2 + eps), h2 / (w2 + eps))  # aspect ratio
277      return (w2 > wh_thr) & (h2 > wh_thr) & (w2 * h2 / (w1 * h1 + eps) > area_thr) & (ar < ar_thr)  #
```

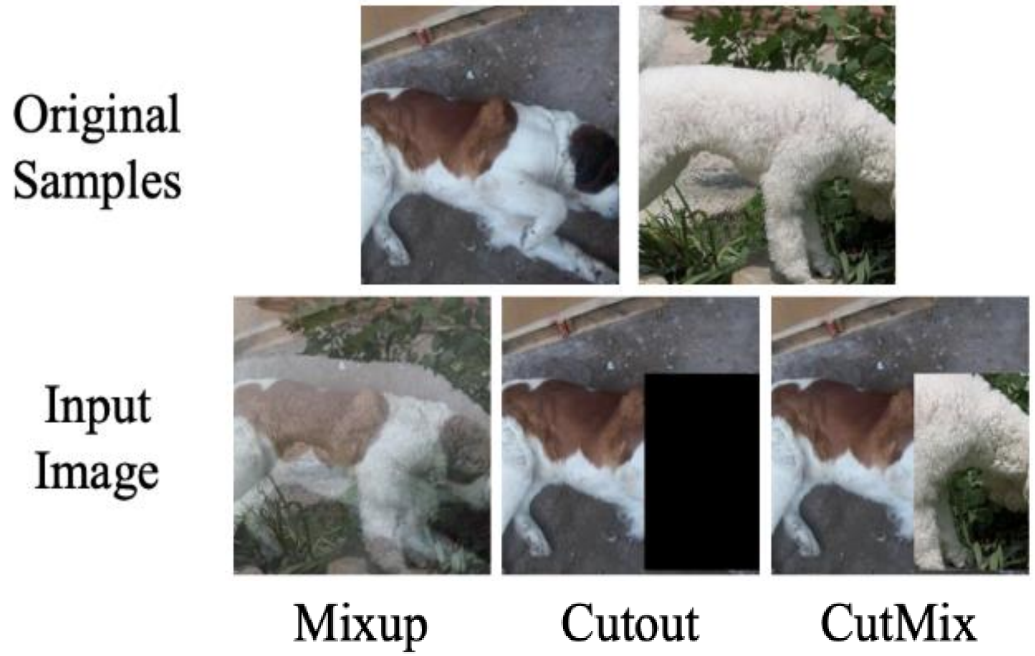
The current settings reject boxes with widths or heights < 2 pixels.

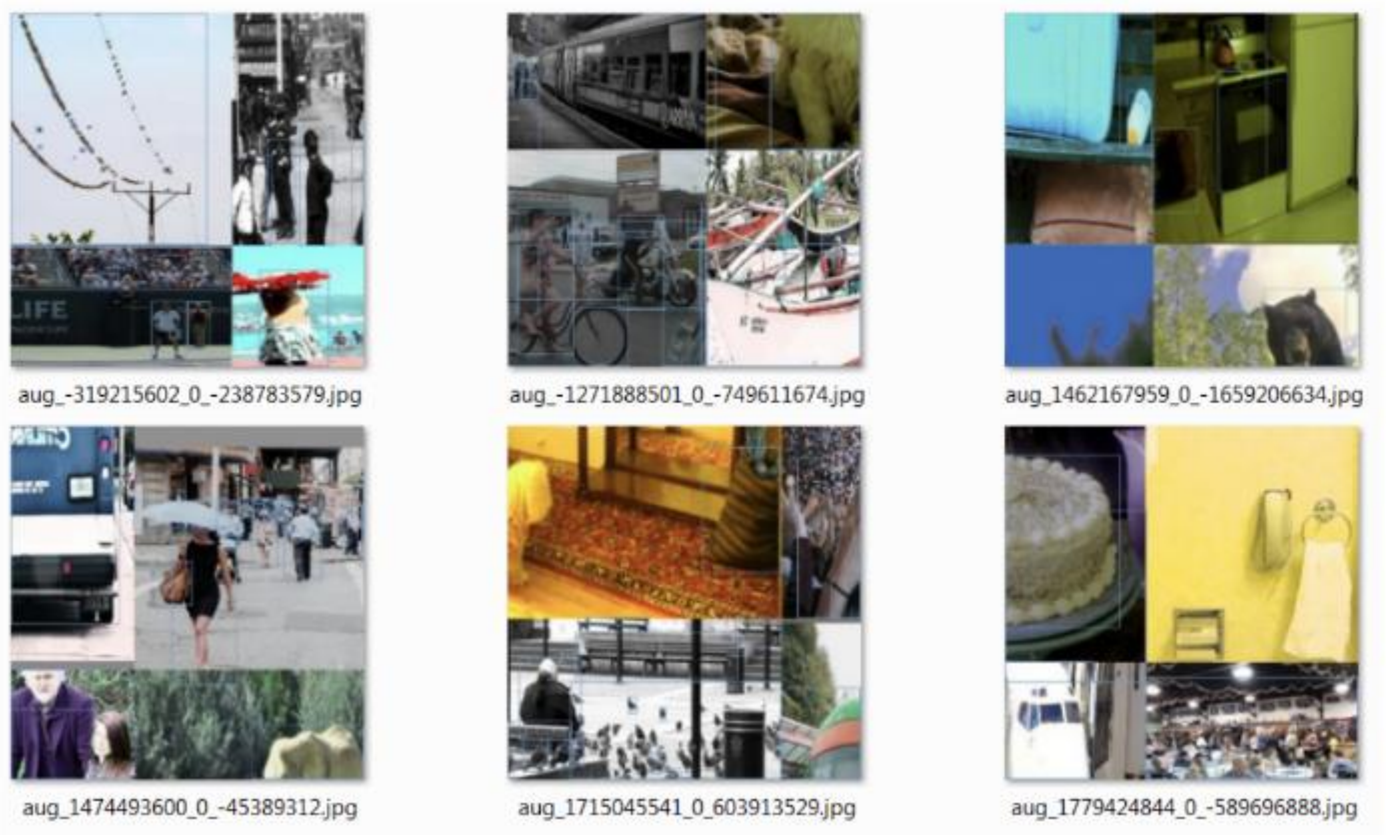
👍 1

Parameters of the four structures

	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
depth_multiple	0.33	0.67	1.0	1.33
width_multiple	0.50	0.75	1.0	1.25
BottleneckCSP数 BCSPn(True)	1,3,3	2,6,6	3,9,9	4,12,12
BottleneckCSP数 BCSPn(Flase)	1	2	3	4
Conv卷积核数量	32,64,128,256,512	48,96,192,384,768	64,128,256,512,1024	80,160,320,640,1280

CV Data Augmentation





Mosaic

https://blog.roboflow.com/yolov4-data-augmentation/

Non-Max Suppression (NMS)

The following is the process of selecting the best bounding box using NMS-

Step 1: Select the box with highest objectiveness score

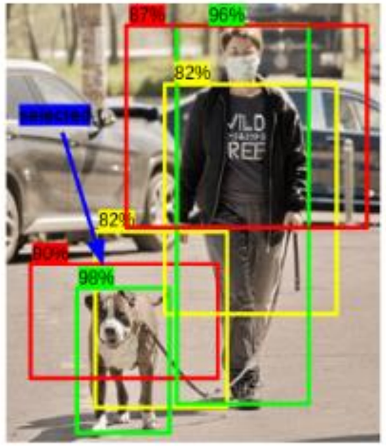
Step 2: Then, compare the overlap (intersection over union) of this box with other boxes

Step 3: Remove the bounding boxes with overlap (intersection over union) >50%

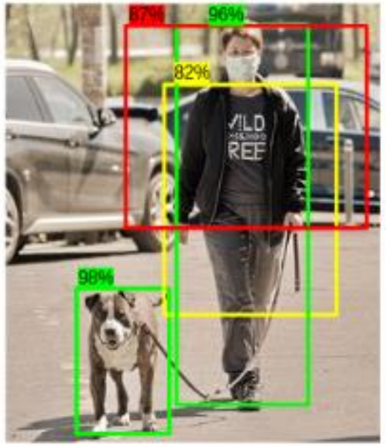
Step 4: Then, move to the next highest objectiveness score

Step 5: Finally, repeat steps 2-4

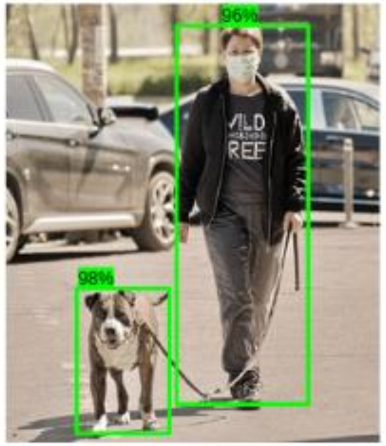
For our example, this loop will run twice. The below images show the output after different steps.



Step 1: Selecting Bounding box with highest score



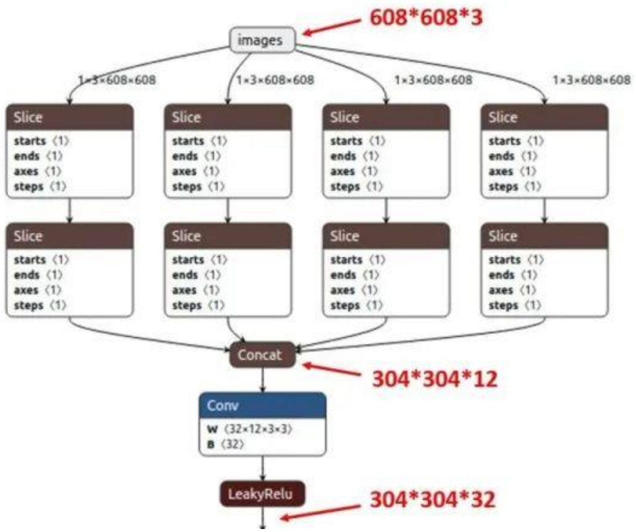
Step 3: Delete Bounding box with high overlap

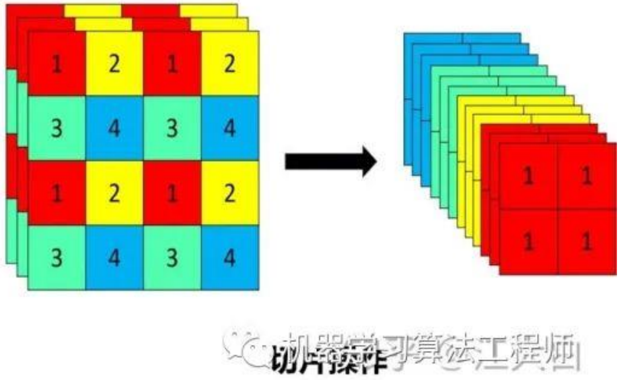


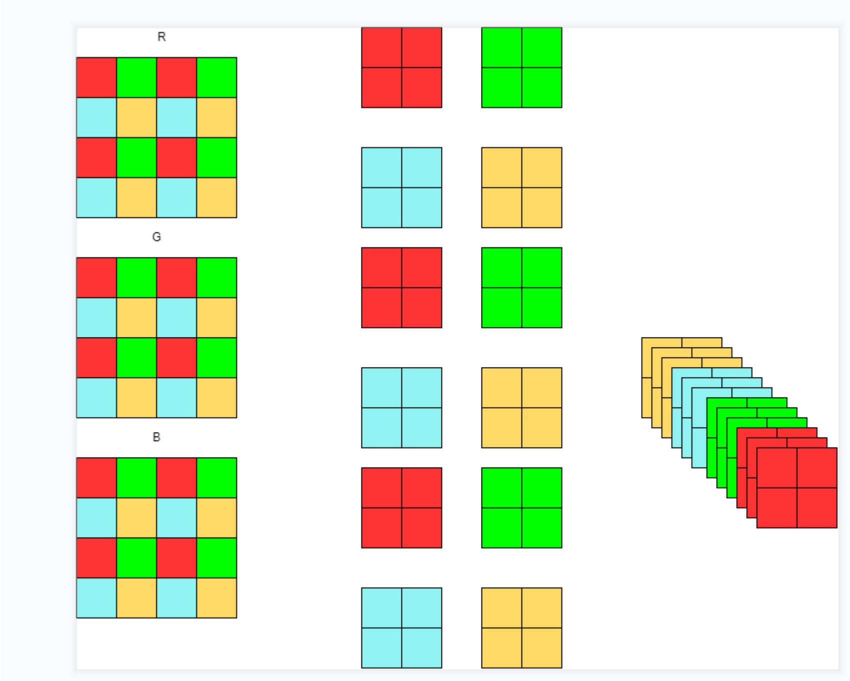
Step 5: Final Output

https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

Focus module







https://programmer.ink/think/analysis-of-yolov5-network-module.html

Cosine lr schedule

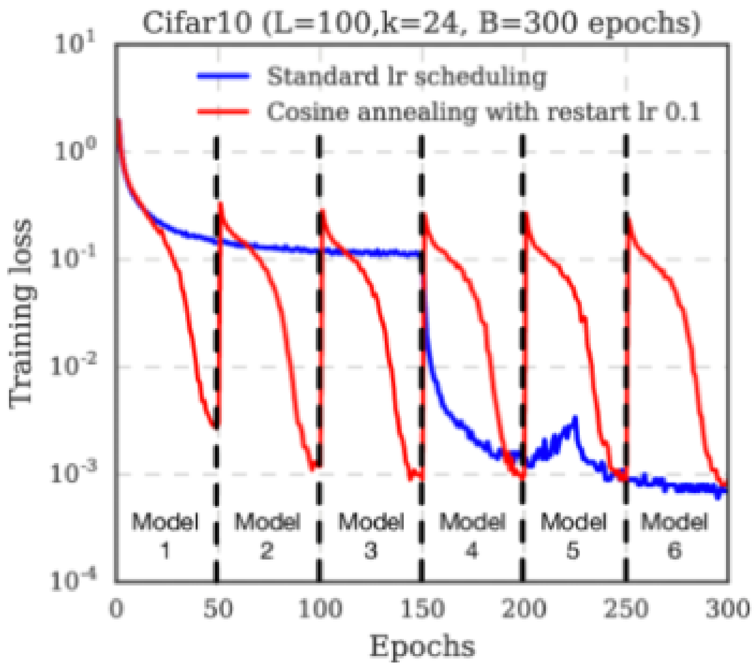
Cosine Annealing is a type of learning rate schedule that has the effect of starting with a large learning rate that is relatively rapidly decreased to a minimum value before being increased rapidly again. The resetting of the learning rate acts like a simulated restart of the learning process and the re-use of good weights as the starting point of the restart is referred to as a "warm restart" in contrast to a "cold restart" where a new set of small random numbers may be used as a starting point.

$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)\right)$$

Where where η^i_{min} and η^i_{max} are ranges for the learning rate, and T_{cur} account for how many epochs have been performed since the last restart.

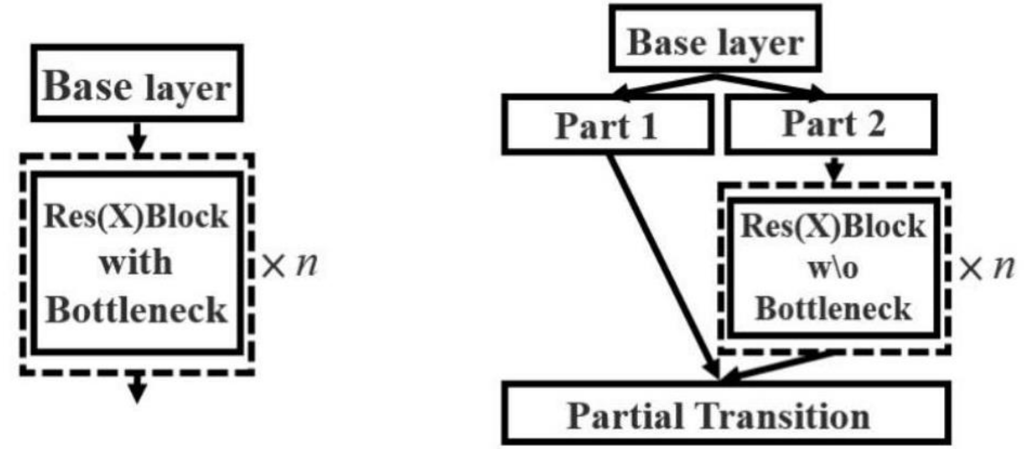
Text Source: Jason Brownlee

Image Source: Gao Huang

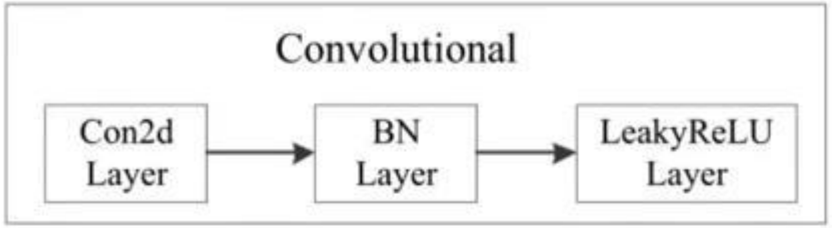


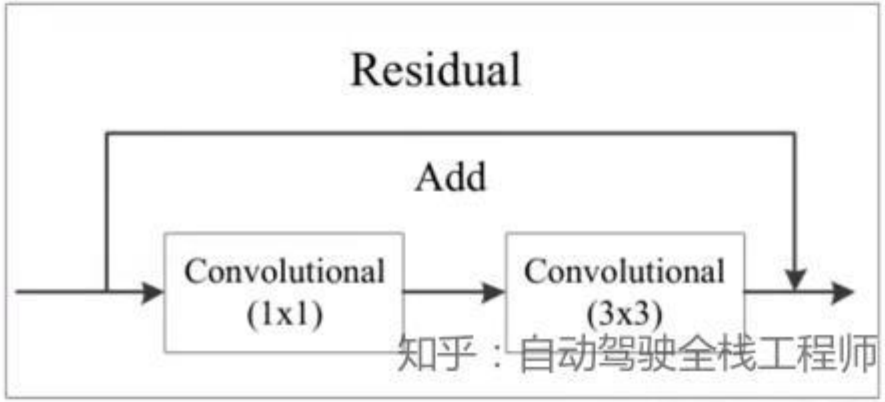
CSP module - Bottleneck

CSP （Cross Stage Partial Network） 跨阶段局部网络

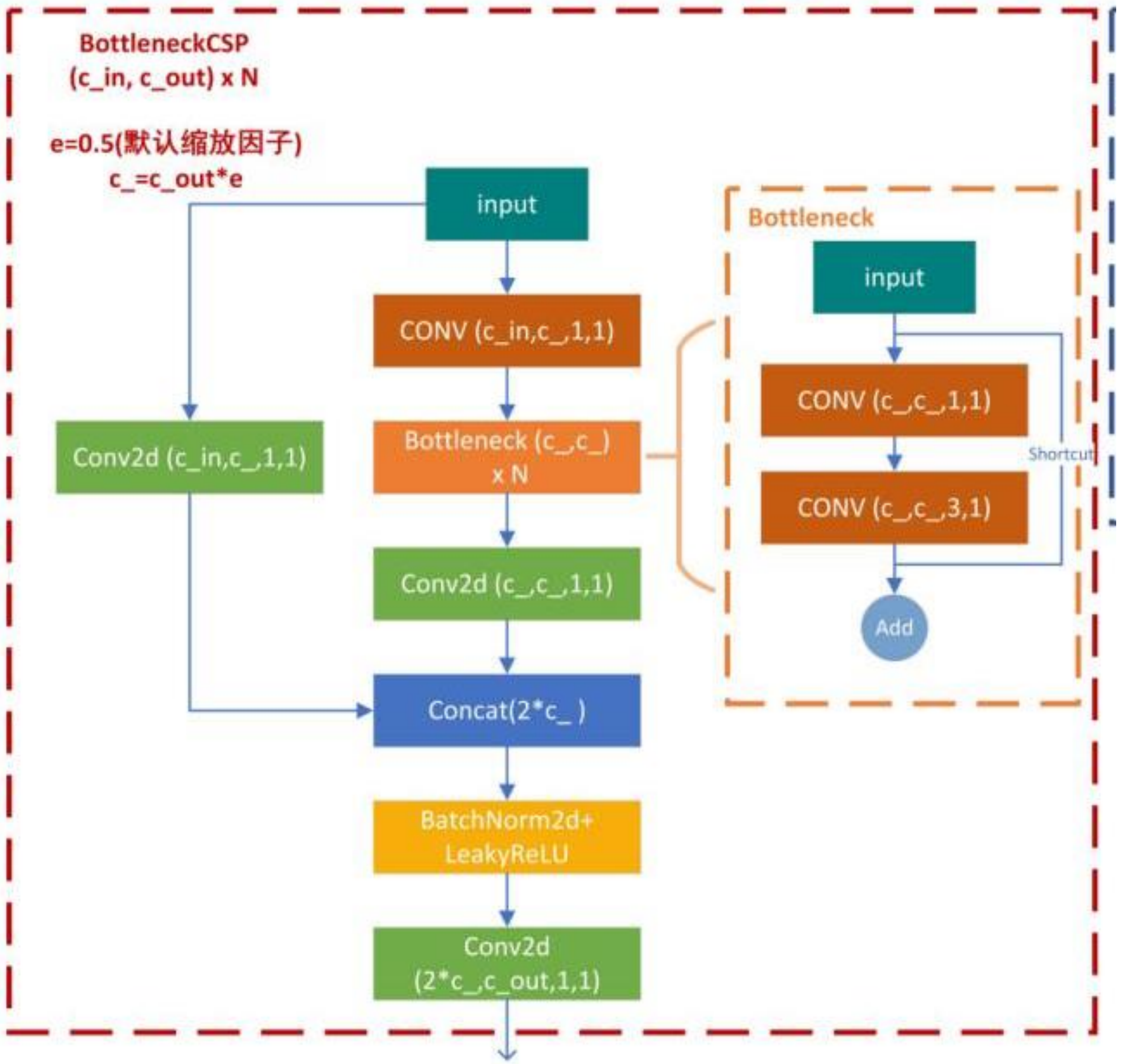


(a) ResNe(X)t (b) CSPResNe(X)t

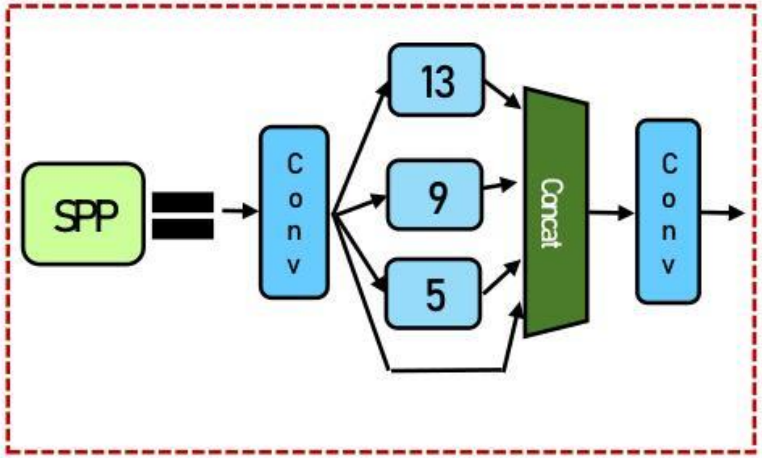


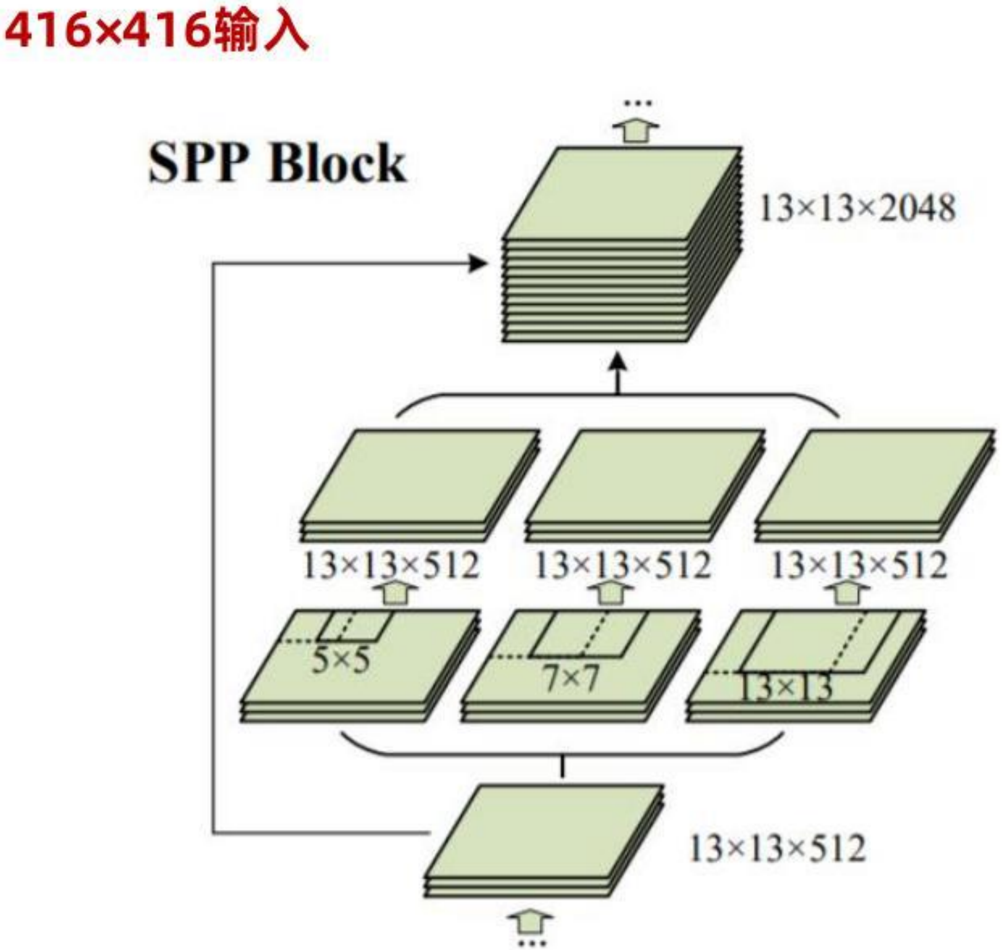


CSP module - Bottleneck



SPP – Spatial Pyramid Pooling





Add the SPP block over the CSP, since it significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed.

Neck – Path Aggregation Network

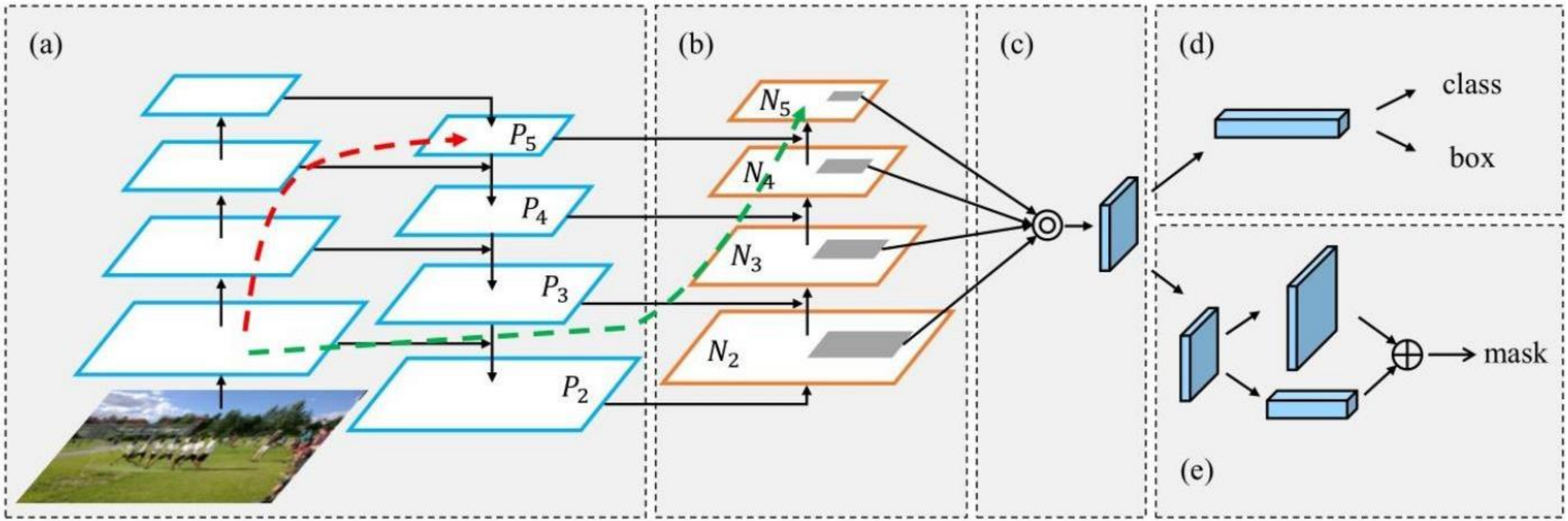
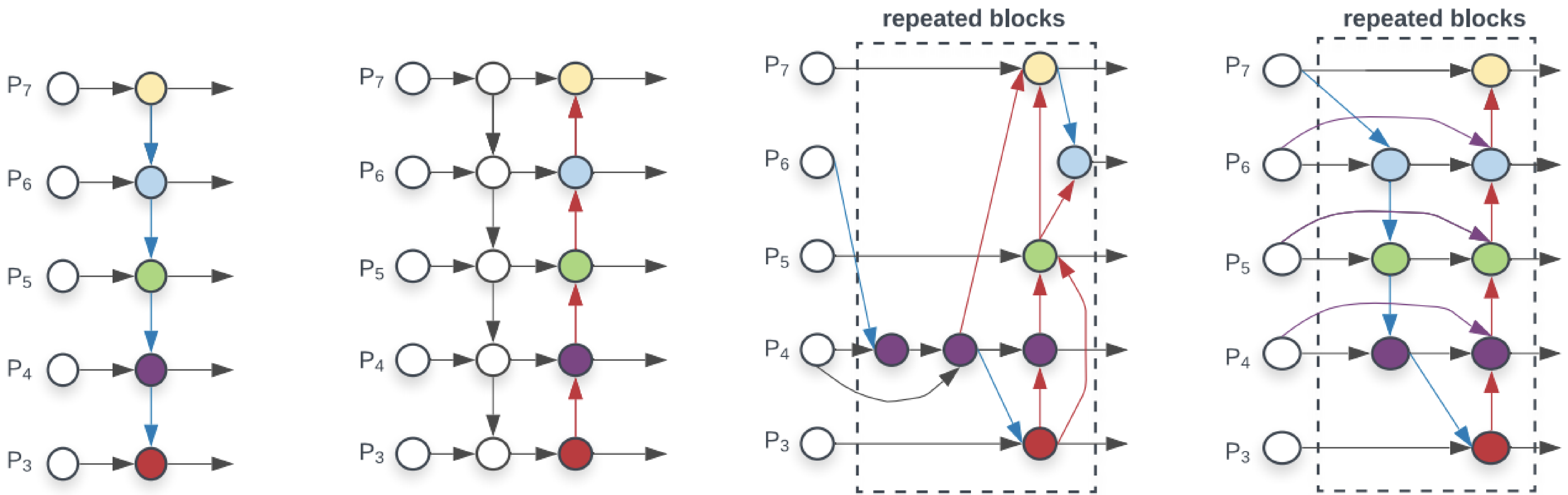


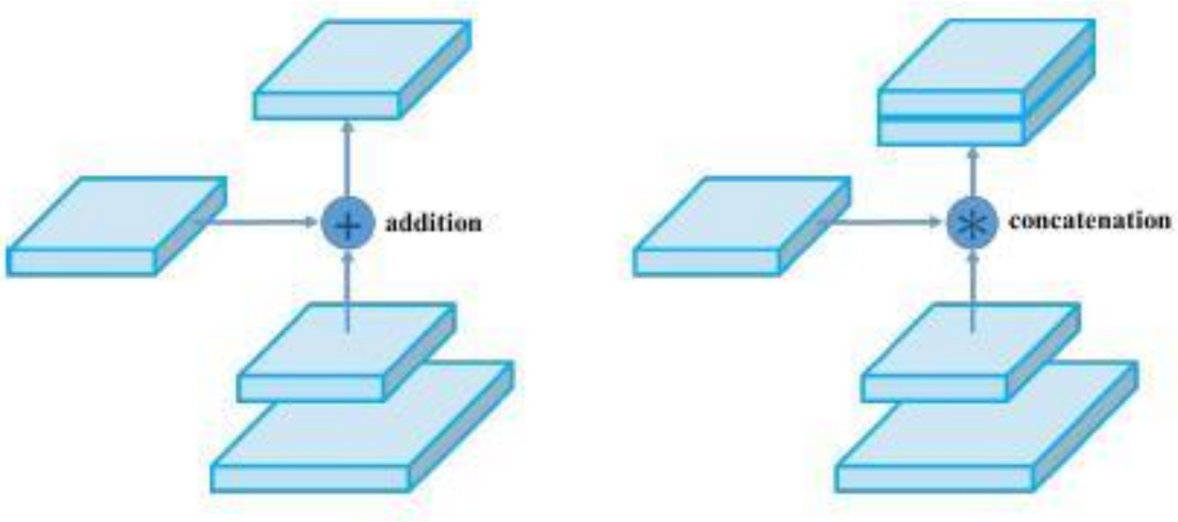
Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Neck – Path Aggregation Network

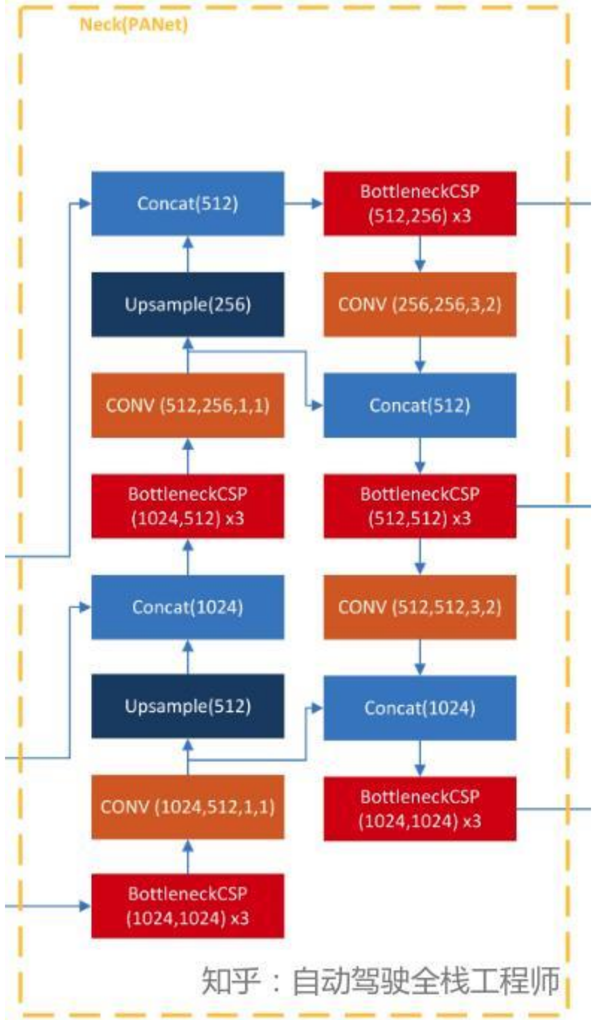


(a) FPN (b) PANet (c) NAS-FPN (d) BiFPN

Neck – Path Aggregation Network



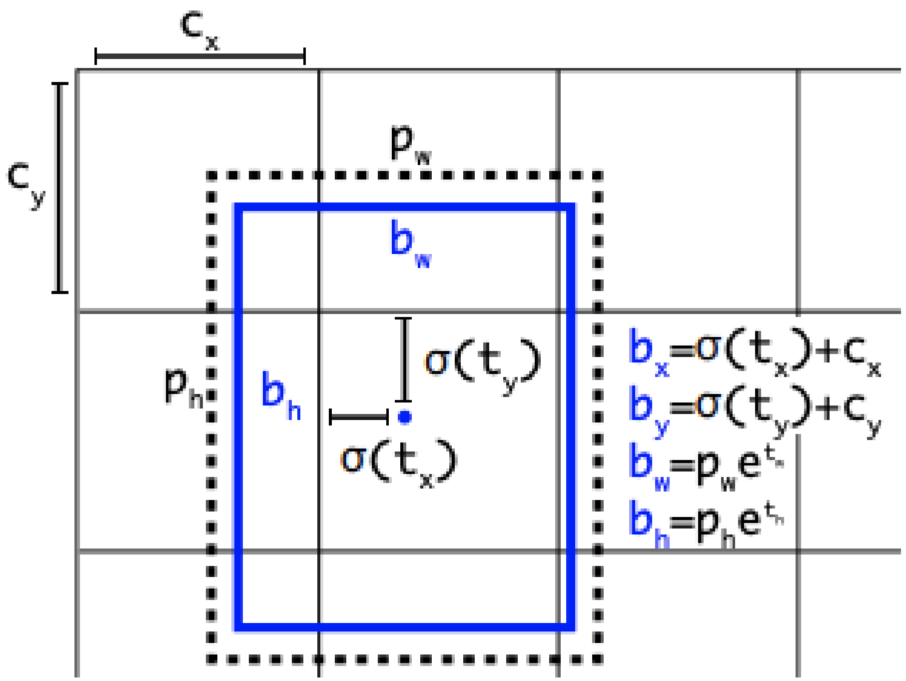
(a) PAN [49]   (a) Our modified PAN



Anchor

- There will be Anchor frame with initial length and width
- the anchor frame initially set by Yolov5 on the Coco data set:

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

Total Loss

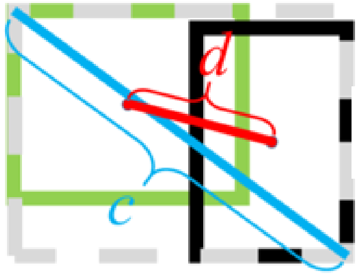
The YOLOv5 loss consists of three parts:

- Classes loss(BCE loss)
- Objectness loss(BCE loss)
- Location loss(CIoU loss)

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

Total Loss

$$Total\ Loss = \lambda_{box}L_{box} + \lambda_{obj}L_{obj} + \lambda_{cls}L_{cls}$$



$\lambda_{box} = balancing\ parameter\ loss\ regression$
$\lambda_{obj} = balancing\ parameter\ loss\ objectness$
$\lambda_{cls} = balancing\ parameter\ loss\ classification$

$$L_{box} = \sum_{i=0}^{s^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} L_{CIoU}$$

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

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

$\rho = distance\ of\ center\ point$

$c = distance\ of\ border\ points\ (top - left, right - bottom)$

$w = width\ prediction\ box$

$h = height\ prediction\ box$

$w^{gt} = width\ ground\ truth\ box$

$h^{gt} = height\ ground\ truth\ box$

$s^2 = grid\ size$

$B = number\ of\ anchors$

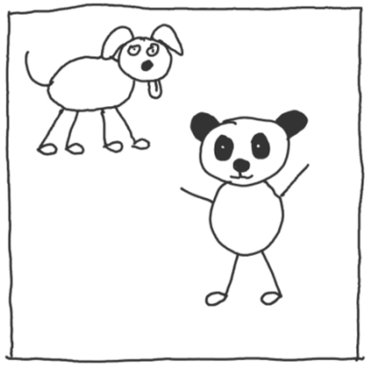
$i = grid$

$j = anchor$

*$\mathbb{1}$ **obj** is equal to one when there is an object in the cell, and 0 otherwise.*

Total Loss

$$\text{Binary Cross-entropy} = -\Big(\underbrace{p(x)\cdot\log q(x)}_{\text{This cancels out if the target is 0}} + \underbrace{(1-p(x))\cdot\log(1-q(x))}_{\text{This cancels out if the target is 1}}\Big)$$

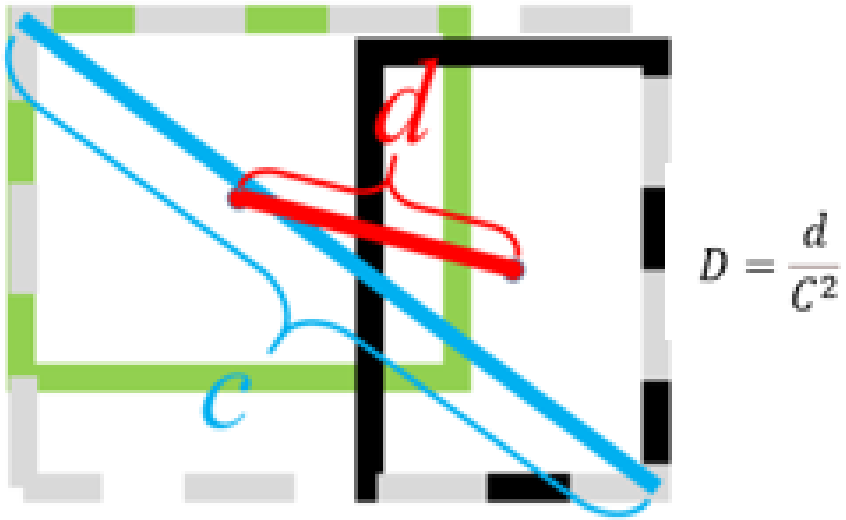


TARGET	PREDICTION
$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.6 \\ 0.7 \\ 0.4 \end{bmatrix}$

https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451

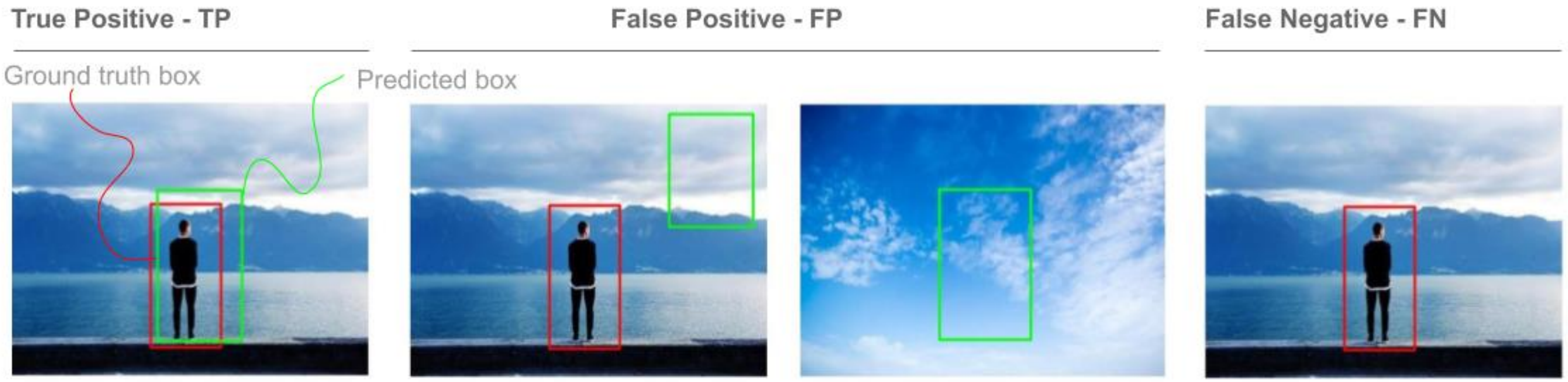
Distance IoU Loss

$$DIoU = 1 - \text{IoU} + \frac{d}{c^2}$$



DIoU loss is invariant to the scale of regression problem, and like GIoU loss, DIoU loss also provides the moving directions for predicted bounding boxes for non-overlapping cases.

TP, FP and FN



The object **is there**, and the model **detects** it, with an IoU against ground truth box **above** the **threshold**.

Left: The object **is there**, but the predicted box has an IoU against ground truth box **less** than **threshold**.

Right: The object is **not there**, and the model **detects** one.

The object **is there**, and the model **doesn't** detect it. The ground truth object has **no** prediction.

Image source: https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html

Metric

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

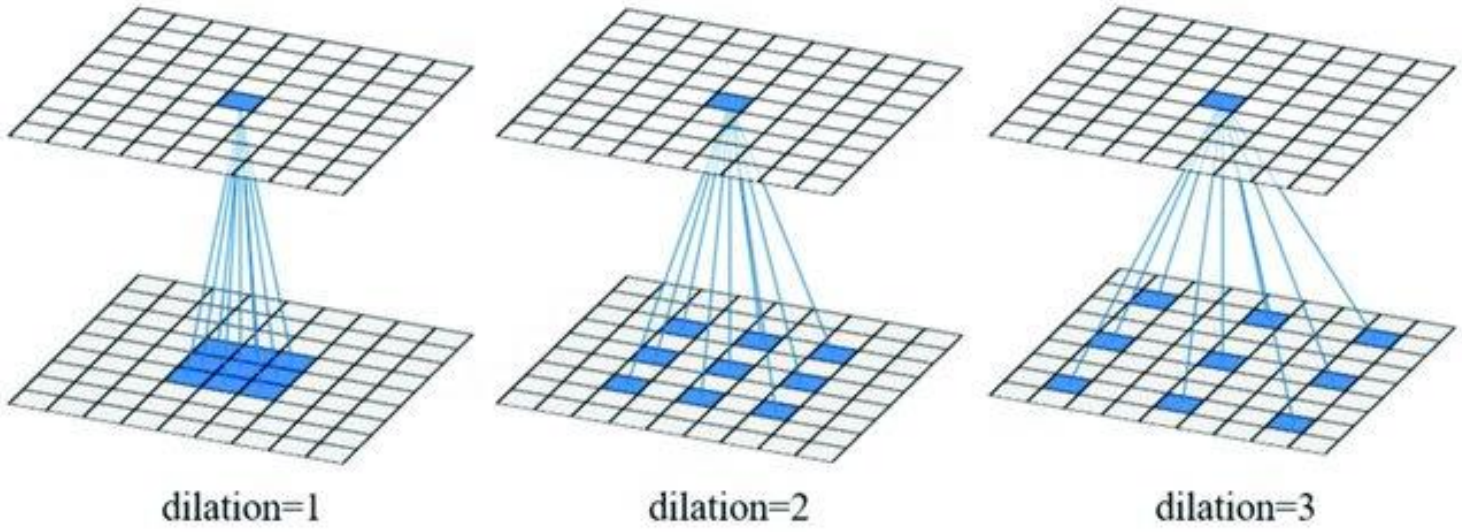
$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

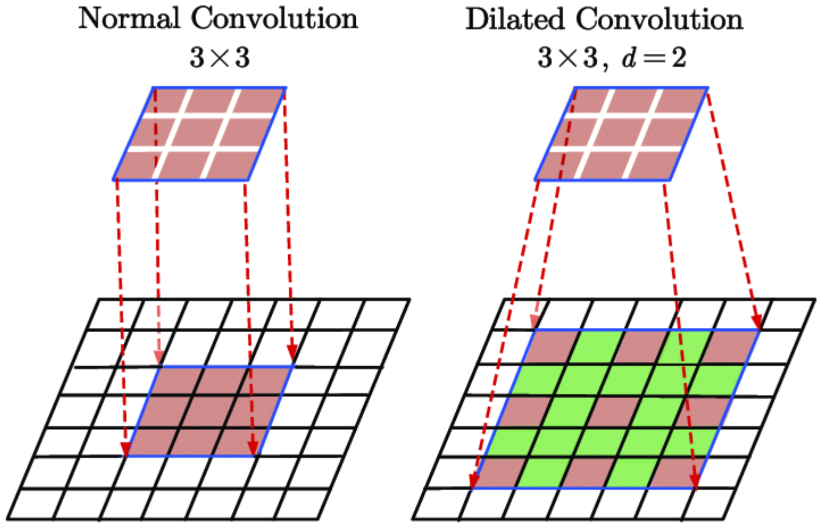
$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

Source: https://www.researchgate.net/post/What_is_the_best_metric_precision_recall_f1_and_accuracy_to_evaluate_the_machine_learning_model_for_imbalanced_data

Dilated convolution





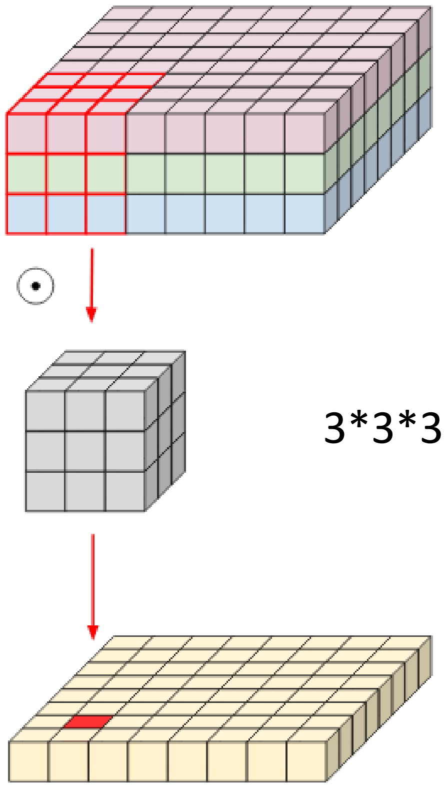
Dilated Convolution

Introduced by Yu et al. in Multi-Scale Context Aggregation by Dilated Convolutions

Dilated Convolutions are a type of convolution that "inflate" the kernel by inserting holes between the kernel elements. An additional parameter l (dilation rate) indicates how much the kernel is widened. There are usually $l-1$ spaces inserted between kernel elements.

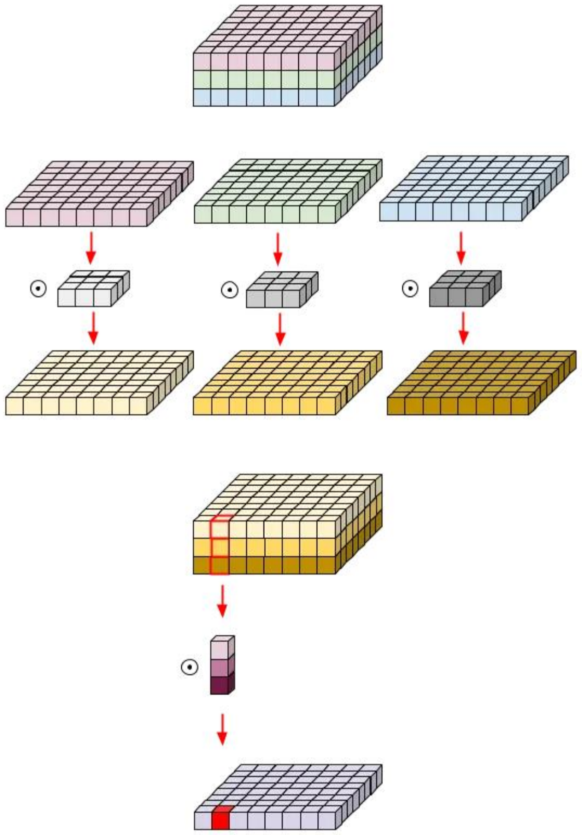
https://paperswithcode.com/method/dilated-convolution

Normal convolution vs Depth-wise convolution



3*3*3*3 = 81

Normal convolution



3*(3*3*1) = 27

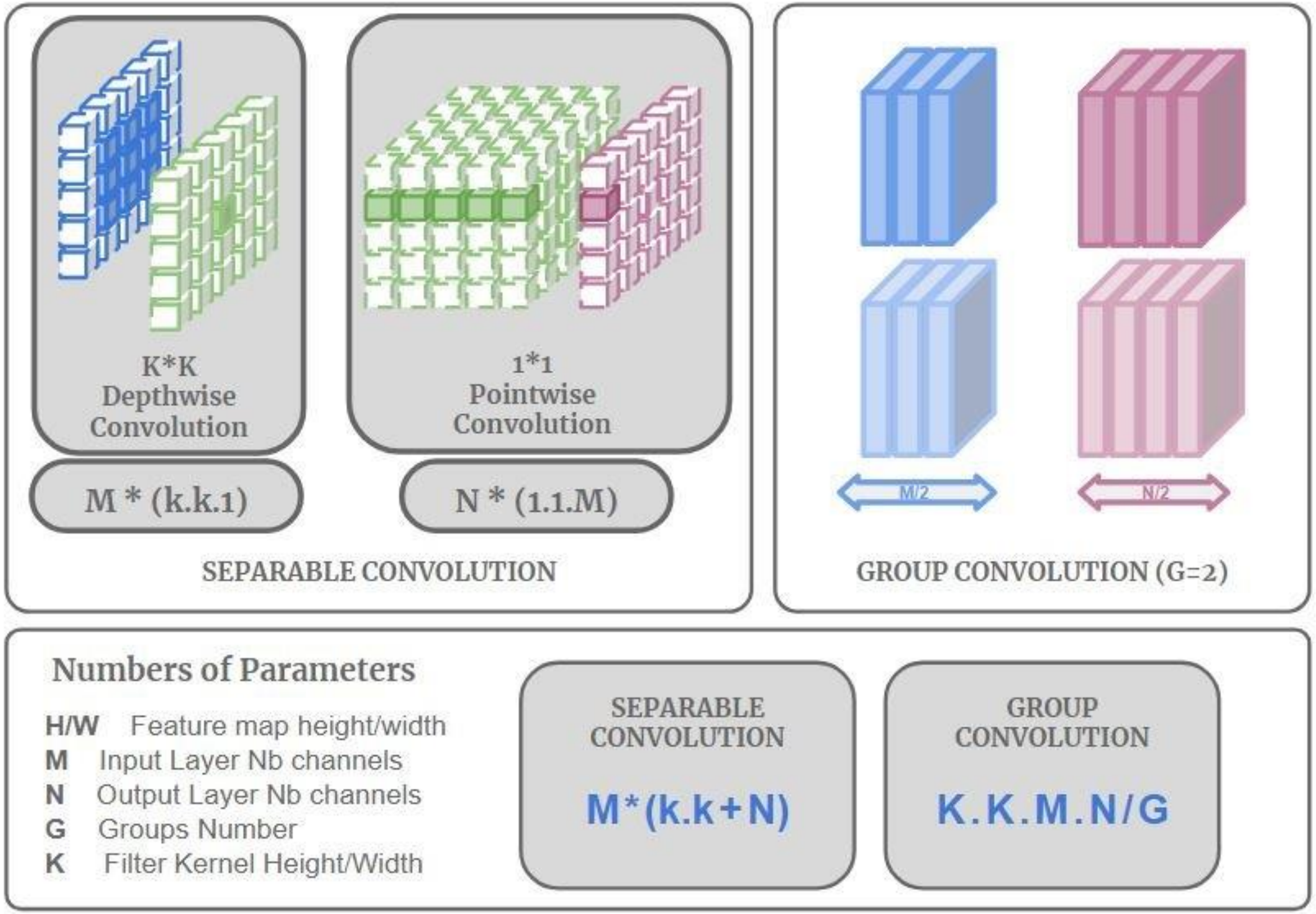
3*(1*1*3) = 9

27 + 9 = 36

Depthwise Separable Convolution

https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Normal convolution vs Depth-wise convolution



https://www.researchgate.net/publication/329740351_Analysis_of_Efficient_CNN_Design_Techniques_for_Semantic_Segmentation/figures?lo=1

YOLOv8 Loss Function

YOLOv8 losses included classification loss (VFL Loss) and regression loss (CIOU Loss + Distribution Focal loss (DFL)), and the three losses were weighted by a certain weight ratio. The three formulas are as follows:

$$VFL(p,q) = \begin{cases} -q(q(\log(p) + (1-q)\log(1-p)) & q > 0 \\ -\alpha p^{\gamma}\log(1-p) & q = 0 \end{cases} \tag{14}$$

$$\mathcal{L}_{\mathrm{CIoU}} = 1 - \mathrm{IoU} + \frac{\rho^2(\mathrm{b}, \mathrm{b}^{\mathrm{gt}})}{\mathrm{c}^2} + \alpha v \tag{15}$$

$$DFL(S_i, S_{i+1}) = -((y_{i+1} - y)\log(S_i) + (y - y_i)\log(S_{i+1})) \tag{16}$$

AG: Attention Gate