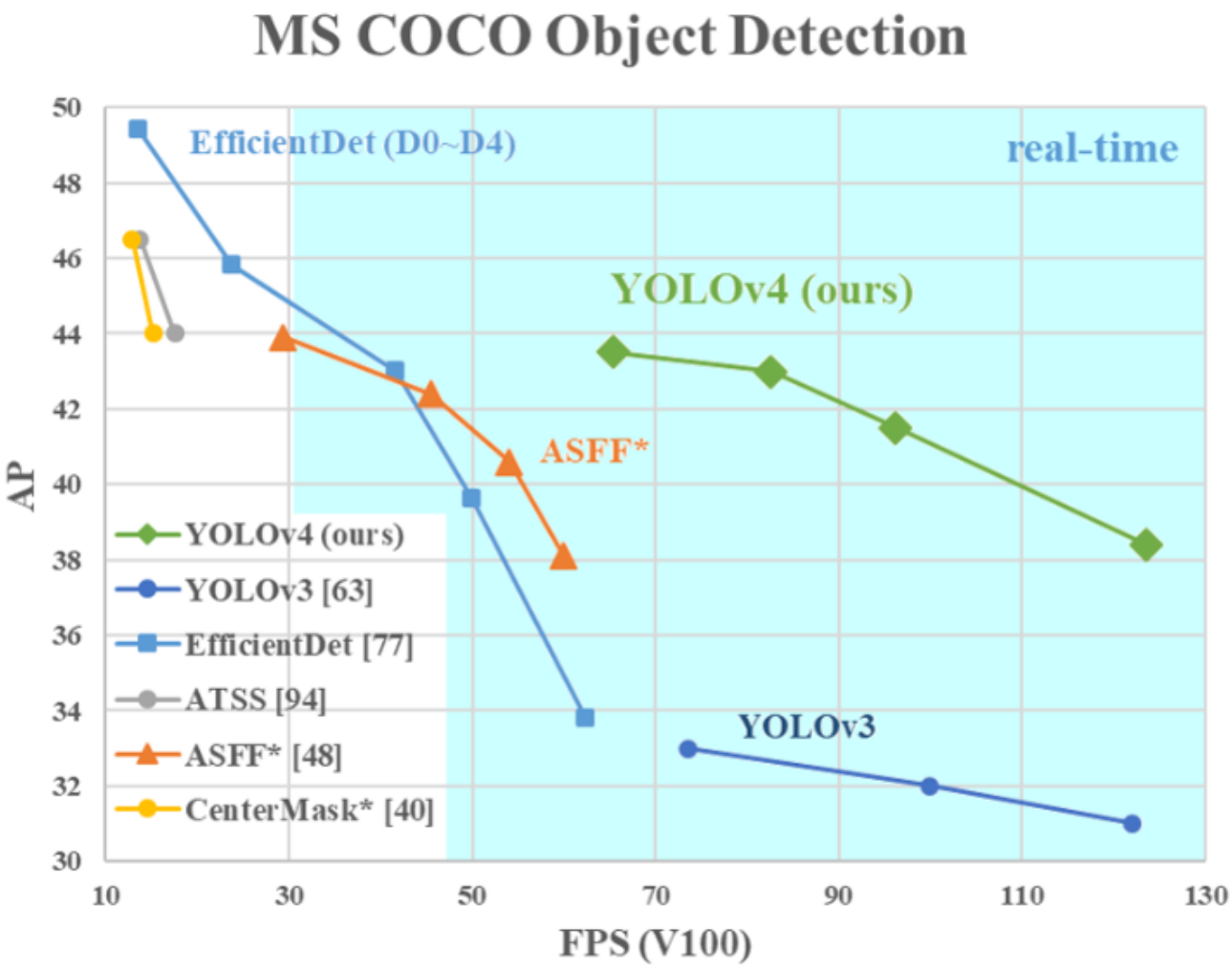
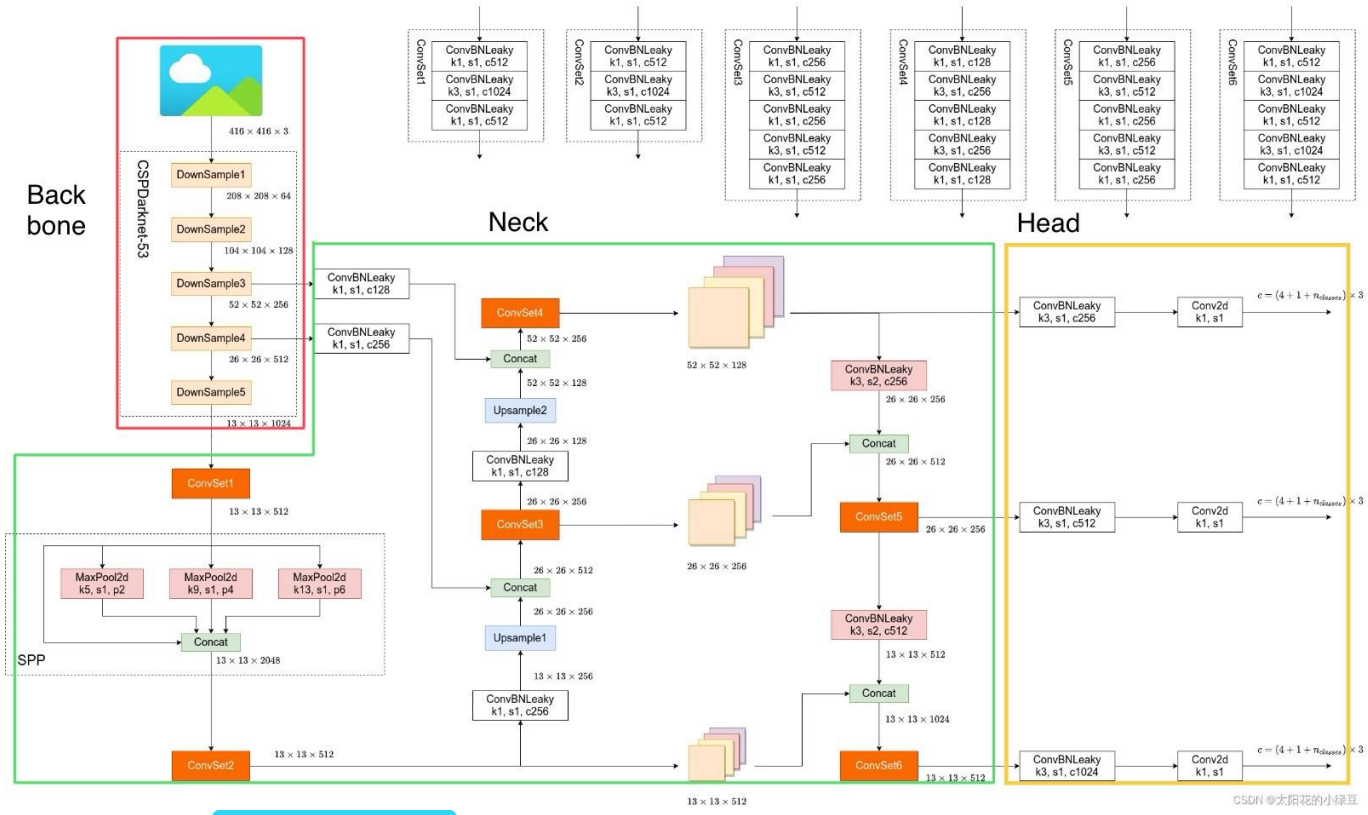
YOLO v4

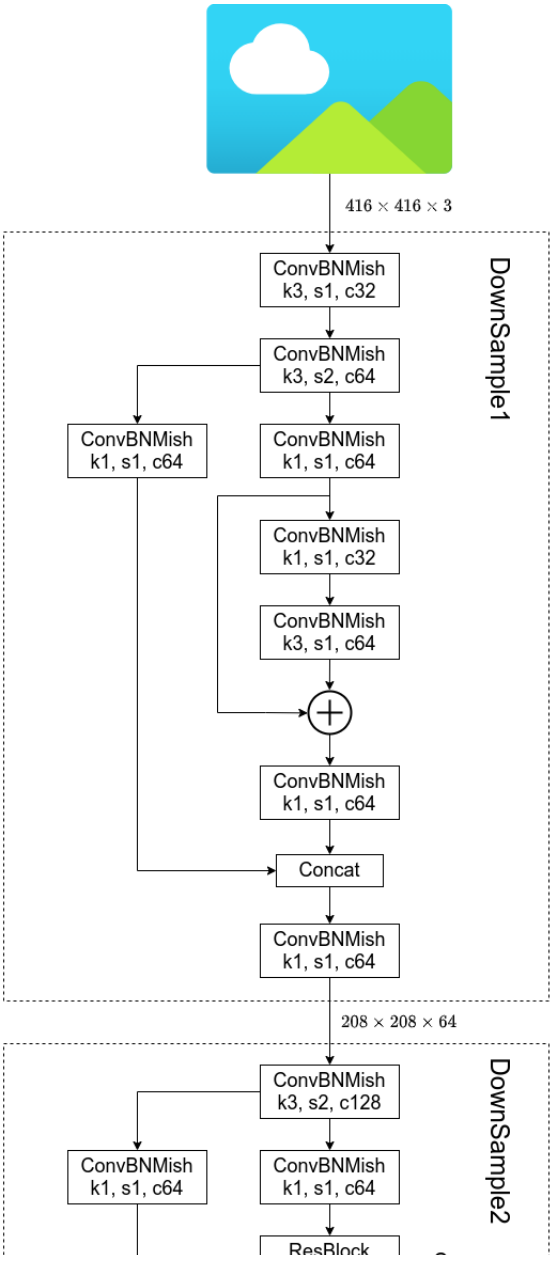


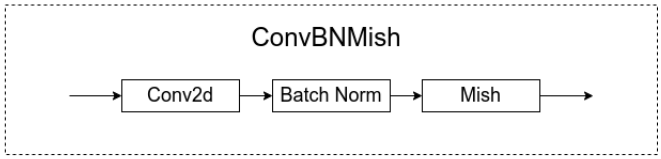
1.网络结构 (CSPDarknet-53)

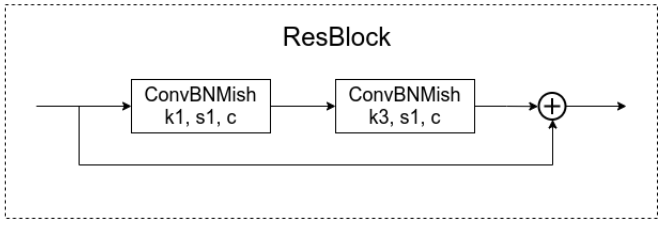
主要改动

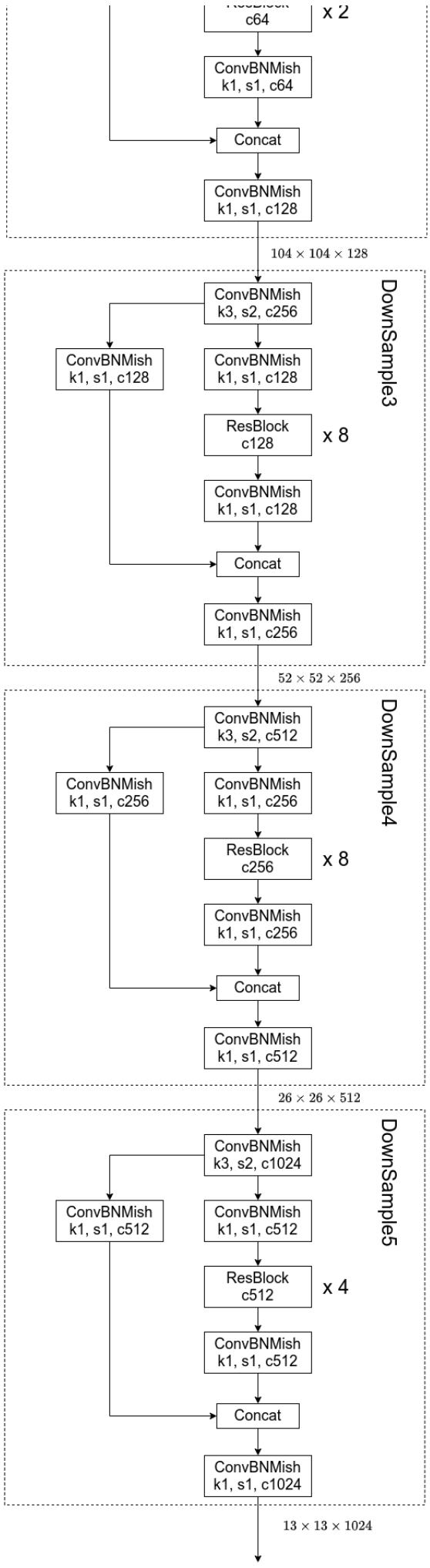
1. backbone引入CSP模块（激活函数改用Mish）
2. 颈部网络加入SPP模块
3. 颈部网络加入PAN模块
   (头部保持与yolov3相同)


Back bone
Neck
Head
CSPDarknet-53
DownSample1
DownSample2
DownSample3
DownSample4
DownSample5
ConvSet1
ConvSet2
ConvSet3
ConvSet4
ConvSet5
ConvSet6
SPP
MaxPool2d k5, s1, p2
MaxPool2d k9, s1, p4
MaxPool2d k13, s1, p6
Concat
Upsample1
Upsample2
ConvBNLeaky k1, s1, c128
ConvBNLeaky k1, s1, c256
ConvBNLeaky k1, s1, c512
ConvBNLeaky k3, s1, c1024
ConvBNLeaky k3, s2, c256
ConvBNLeaky k3, s2, c512
Conv2d k1, s1
416 × 416 × 3
208 × 208 × 64
104 × 104 × 128
52 × 52 × 256
26 × 26 × 512
13 × 13 × 1024
13 × 13 × 2048
CSDN @太阳花的小绿豆

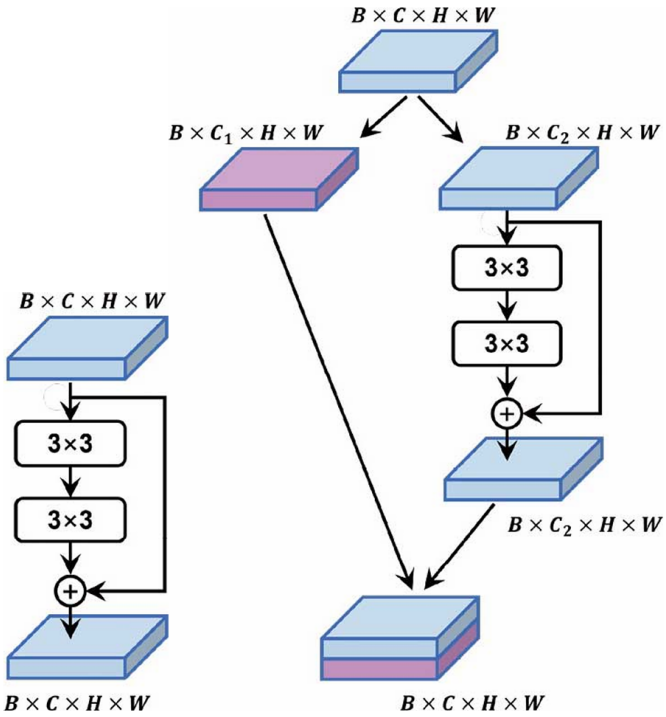

416 × 416 × 3
DownSample1
ConvBNMish k3, s1, c32
ConvBNMish k3, s2, c64
ConvBNMish k1, s1, c64
ConvBNMish k1, s1, c32
ConvBNMish k3, s1, c64
Concat
208 × 208 × 64
DownSample2
ConvBNMish k3, s2, c128
ResBlock


ConvBNMish
Conv2d
Batch Norm
Mish


ResBlock
ConvBNMish k1, s1, c
ConvBNMish k3, s1, c



1.1 CSP结构

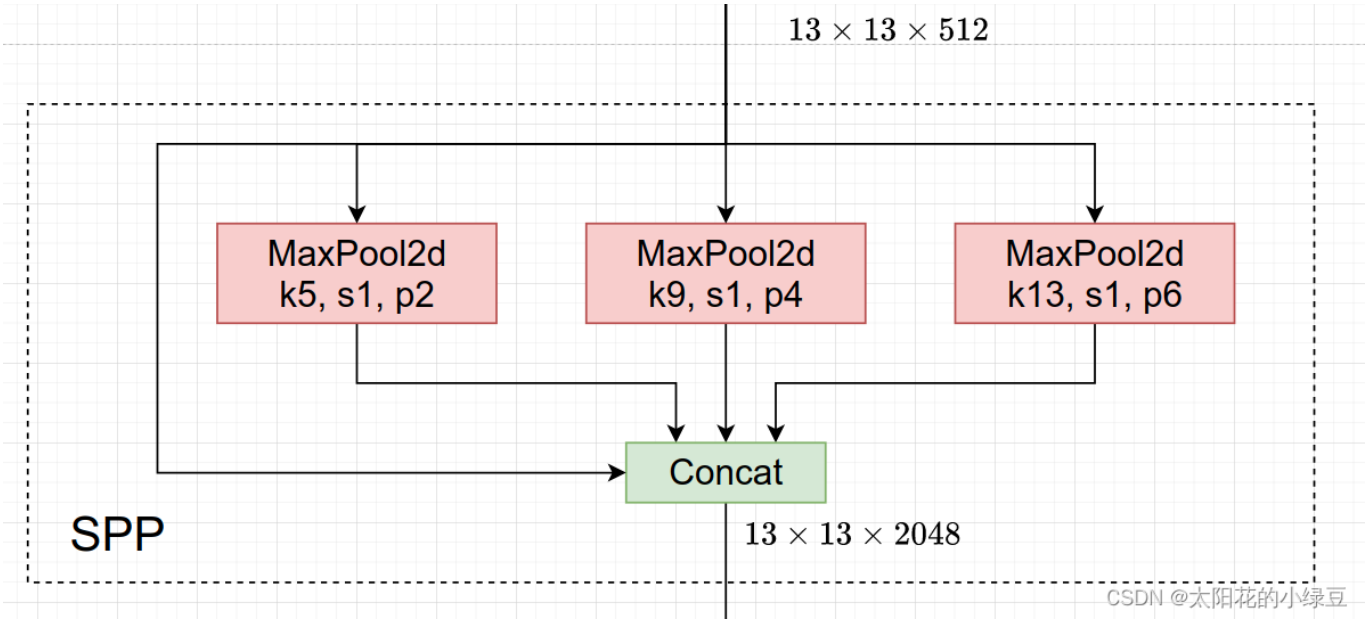


(a)标准的残差模块 (b)基于CSP结构改进的残差模块

CSPDarknet53就是将CSP结构融入了Darknet53中。CSP结构是在CSPNet（Cross Stage Partial Network）论文中提出的，CSP结构的合理性在于卷积神经网络中的特征往往具有很大的冗余，不同通道的特征图包含的信息可能是相似的，这一观点在GhostNet中也被充分说明了。因此，CSPNet的作者团队认为没有必要去处理全部的通道，而只需处理其中的一部分，另一部分保持不变即可。因此CSPNet作者说在目标检测任务中使用CSP结构有如下好处：

1. Strengthening learning ability of a CNN
2. Removing computational bottlenecks
3. Reducing memory costs 即减少网络的计算量以及对显存的占用，同时保证网络的能力不变或者略微提升。CSP结构的思想参考原论文中绘制的CSPDenseNet，进入每个stage（一般在下采样后）先将数据划分成两个部分，但具体怎么划分呢，在CSPNet中是直接按照通道均分，但在YOLOv4网络中是通过两个1x1的卷积层来实现的。在Part2后跟一堆Blocks然后在通过1x1的卷积层（图中的Transition），接着将两个分支的信息在通道方向进行Concat拼接

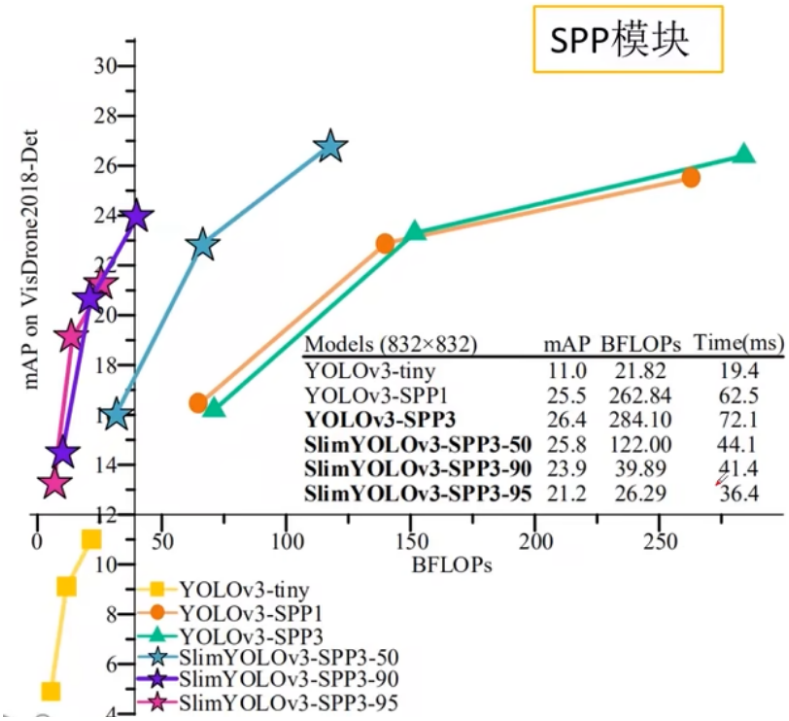
1.2 颈部网络加入SPP模块



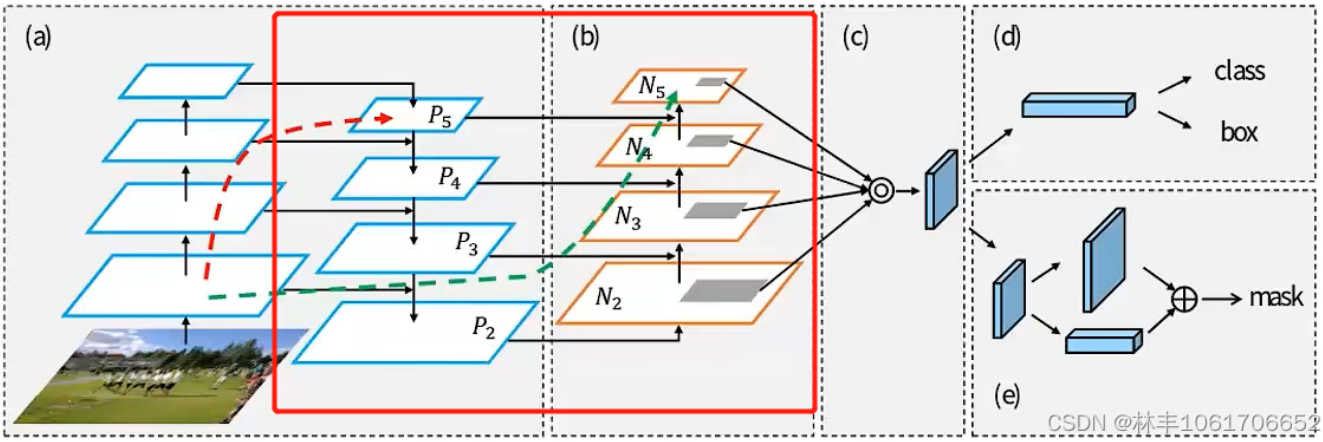
虽然最早的空间金字塔池化(spatial pyramid pooling， SPP)模块是由Kaiming He团队在2015年提出的，但在目标检测任务中常用的SPP模块则是由YOLOv3工作的作者团队所设计的SPP结构，包含4条并行的分支，且每条分支用了不同大小的池化核。SPP结构可以有效地聚合不同尺度的显著特征，同时扩大网络的感受野，进而提升模型的性能。

Q SPP操作之后得到的特征图尺寸是多少？深度是多少？

Q 既然SPP这么好用，可以用在其他尺度的特征图上吗？



1.3 PAN结构



红色框左半部分是 FPN，就是将高层特征图与低层特征图进行融合（这里的融合一般是将高层特征图进行上采样，然后与低层特征图直接相加或者在通道维度 concat 的方式融合，称为top-down），就是让浅层特征图层也能有深层特征图的语义信息；

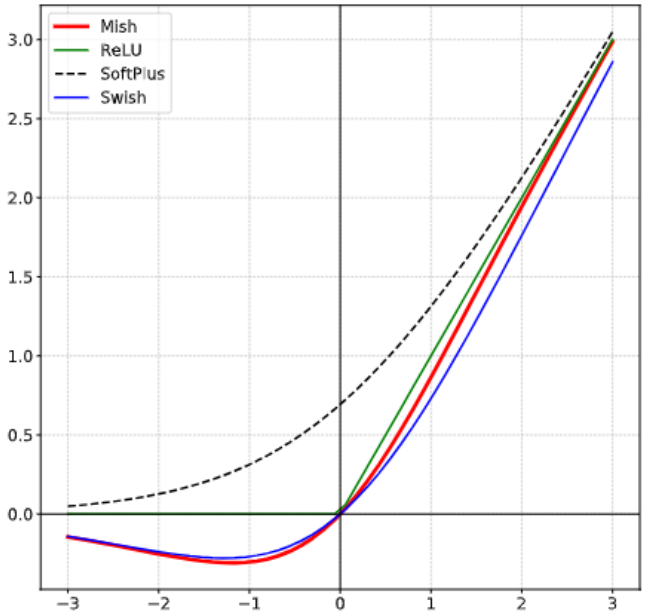
红色框右半部分是相反的思想，将低层的特征图与高层特征图进行融合（这里就是将低层特征图进行下采样，然后和高层的融合，称为bottom-up，使得高层特征图也能有浅层的语义信息，如边缘形状等）。其中激活函数不是Mish而是LeakyReLU。

YOLOv4的PAN结构和原始论文的融合方式略有差异，原始论文中的融合方式，在特征层之间融合时是直接通过相加的方式进行融合的，但在YOLOv4中是通过在通道方向Concat拼接的方式进行融合的

Q SPP和FPN/PAN都进行了特征的融合，二者的区别是什么？

1.4 Mish激活函数

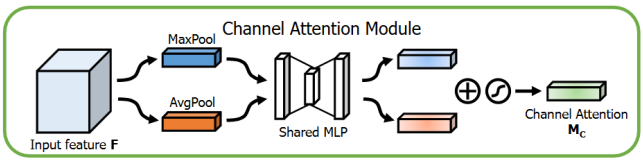
$$f(x) = xtanh(ln(1+e^x))$$



对比ReLU,Mish函数在负半轴靠近原点的位置并非直接截断，保留了部分负值信息

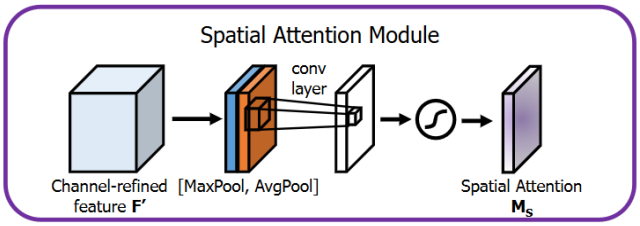
1.5 SAM注意力机制

- channel-wise attention



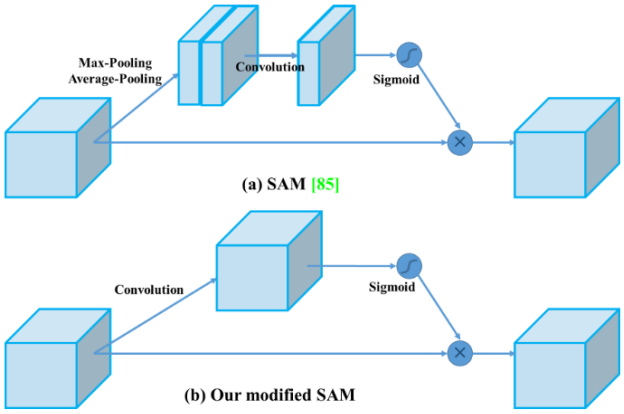
特征图的每一个通道都代表着一个不同的特征，因此，通道注意力是关注什么样的特征是有意义的。 输入是一个 H×W×C 的特征图 F，先分别进行一个空间的全局最大池化和平均池化得到两个 1×1×C 的压缩层，经过卷积和激活函数为每个通道分配不同的权重，得到权重系数 Mc。最后，Mc * 输入的特征F 即可得到权值化的新特征。

- spatial-wise attention



在通道注意力模块之后，再引入空间注意力模块来关注哪里的特征是有意义的。与通道注意力相似，给定一个 H×W×C 的特征 F'，先分别进行一个通道维度的最大池化和平均池化得到两个 H×W×1 的压缩层，并将它们按照通道拼接在一起。然后，经过卷积和激活函数，得到权重系数 Ms。最后，Ms * 输入的特征F' 相乘即可得到权值化后的新特征。

- yolov4使用的SAM



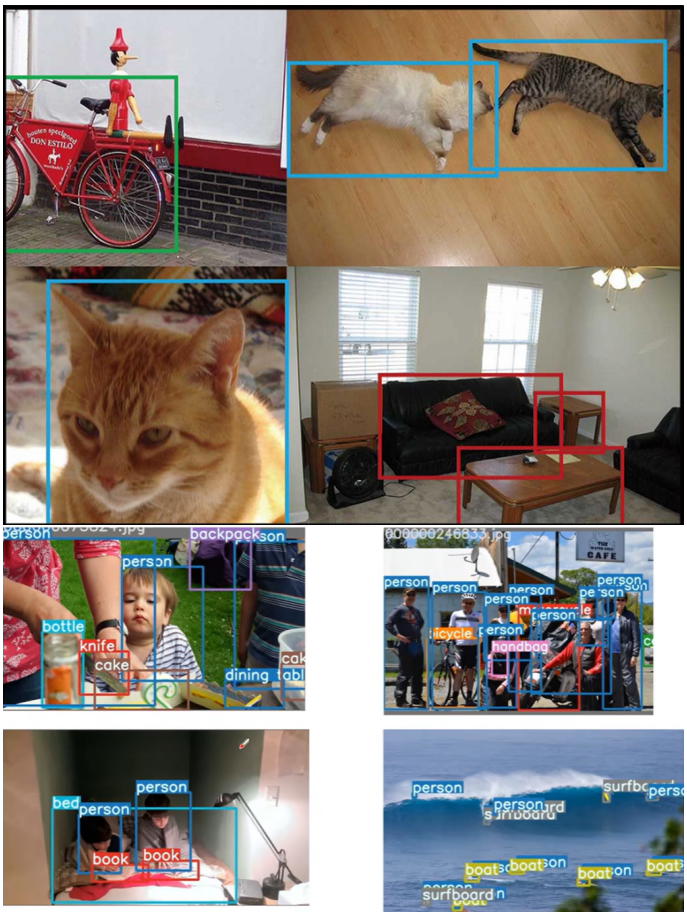
yolov4在spatial-wise attention的基础上去掉了最大池化和平均池化的操作

--"We modify SAM from spatial-wise attention to point-wise attention"

Q 为什么作者说yolov4的SAM是point-wise attention？

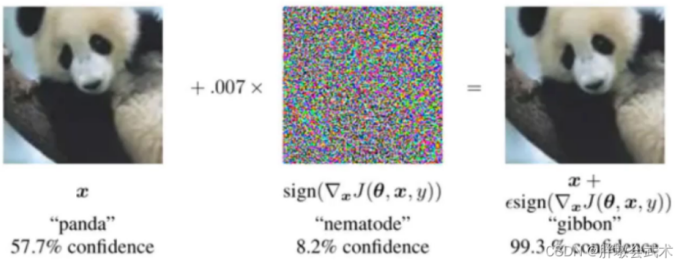
2.训练策略

2.1 Mosaic data augmentation



增加数据的多样性
增加正样本的数量
BN操作能一次性统计多张图片的参数，降低了算力需求

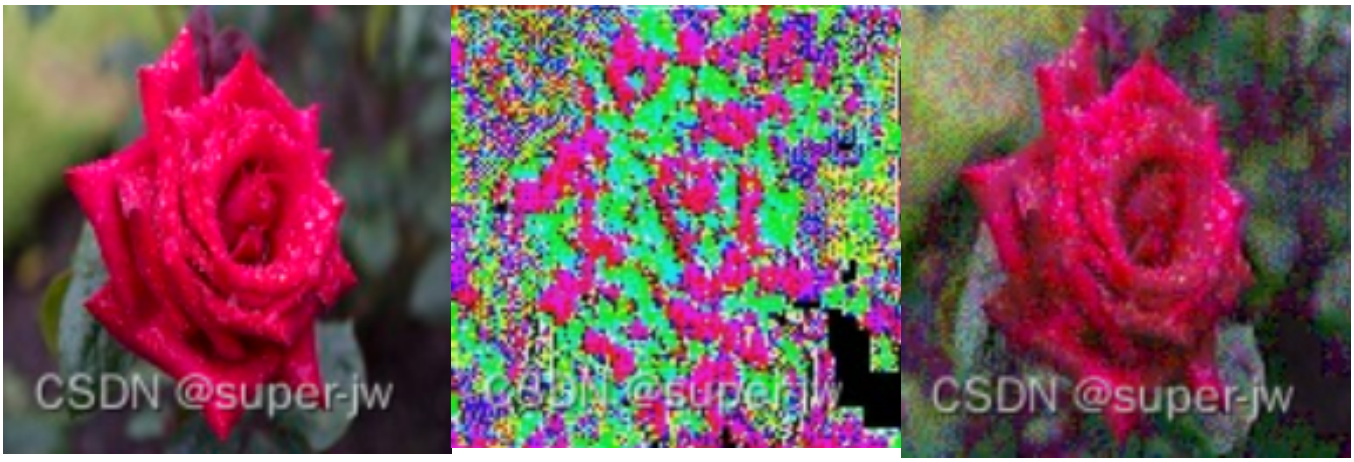
2.2 Self-Adversarial Training， SAT



一种方法是先采用FGSM为原始图像引入噪声，即自我攻击，然后用修改后的图片进行训练，目的是让网络不要将这些噪声误认为特征

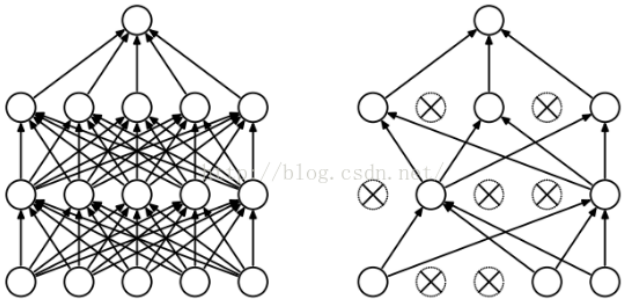
```
def FGSM_ATTACK(img, epslion, grad):
    grad_sign = grad.sign()
    noise_fgsm = epslion * grad_sign
    adv_fgsm = noise_fgsm + img
    adv_fgsm = torch.clamp(adv_fgsm, 0, 1)
    return adv_fgsm, noise_fgsm
```

左：原图（玫瑰） 中：噪声 右：结果图（识别为郁金香）



2.3 Dropblock

- DropOut


(a) Standard Neural Net (b) After applying dropout.

DropOut在全连接层使用，进行随机放弃连接，即将神经元激活输出随机置0。

DropOut之前的计算：

$$
\begin{aligned}
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\mathbf{y}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}),
\end{aligned}
$$

使用DropOut后的计算：

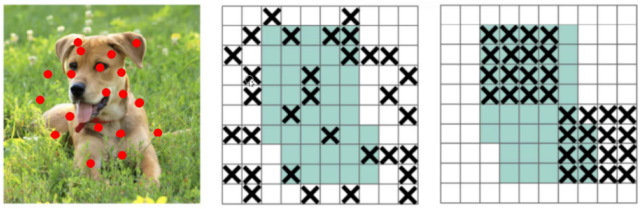
$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

其中p是概率大小，公式中Bernoulli函数，是为了以概率p，随机生成一个0、1的向量

- DropBlock

DropOut的主要问题就是随机drop特征，这一点对于FC层是有效的，但在卷积层是无效的，因为卷积层的特征是空间强相关的。当特征相关时，即使有DropOut，信息仍能传送到下一层，导致过拟合。

DropBlock是适用于卷积层的正则化方法，它作用的对象的特征图。在DropBlock中，特征在一个个block中，当应用DropBlock时，一个feature map中的连续区域会一起被drop掉。那么模型为了拟合数据网络就不得不往别出看以寻找新的证据。



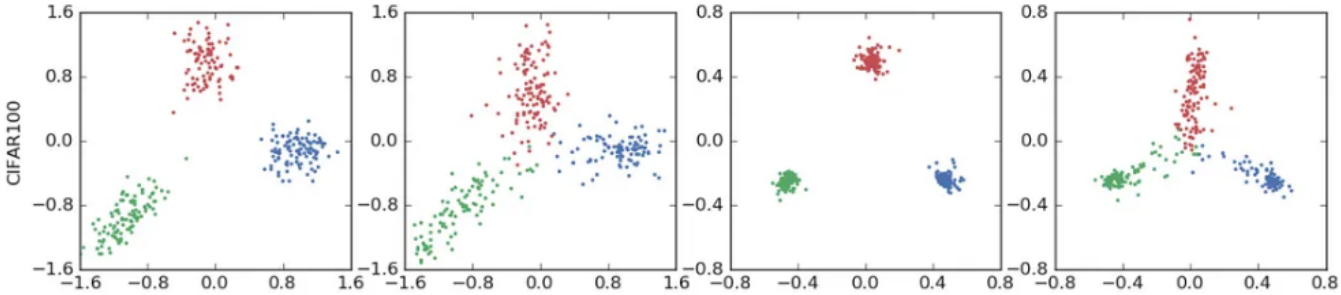
其中(a)是输入到卷积网络的原始图像，(b)和©中的绿色区域包括激活单元，这些激活单元在输入图像中包含语义信息。随机丢弃激活对删除语义信息无效，因为附近的激活包含紧密相关的信息。 相反，删除连续区域可以删除某些语义信息（例如，头或脚），从而强制其余单元学习用于分类输入图像的其它特征，这样就增加了模型的泛化能力。

2.4 Label Smoothing

将绝对化标签值进行平滑,例如将[0,1]的概率标签经过变换成[0.05,0.95] one-hot编码会使模型在训练时会被强烈引导去预测某一类的概率为1

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}}$$

以3分类，softmax层为例，要想要输出[0,0,1]的结果，输入则必须为[-∞,-∞,+∞]，一方面会使得网络自觉良好导致过拟合，一方面会降低模型对错误标签的鲁棒性



左二为未使用Label Smoothing的训练和测试标签，右二为使用Label Smoothing的训练和测试标签

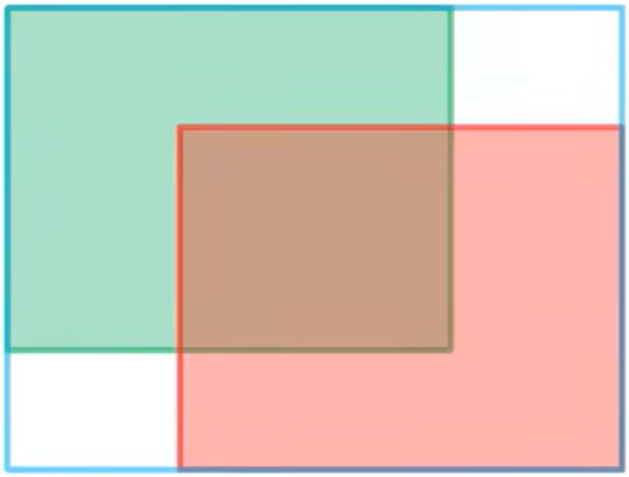
从可视化的角度，label smoothing使簇内更紧密，簇间更分离

2.5 CIOU

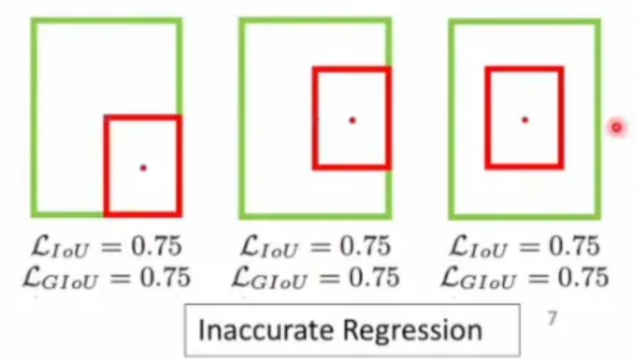
- IOU 问题：当ground truth(GT)和prediction这两个bbox不相交时，loss为0
- GIOU 引入最小封闭矩形A_c

$$GIoU = IoU - \frac{A_c - U}{A_c}$$

$$L_{GIoU} = 1 - GIoU = 1 + \frac{A_c - U}{A_c} - IoU$$



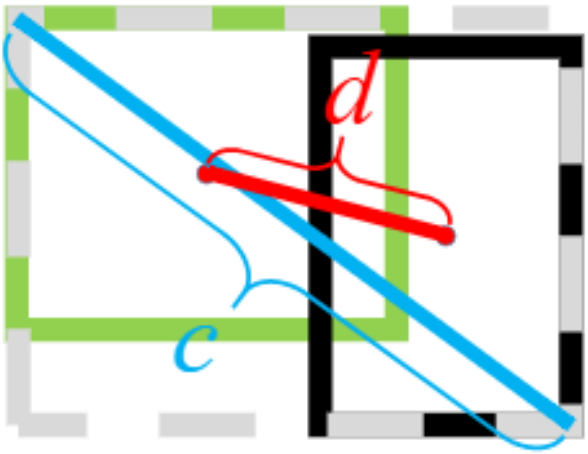
问题：当ground truth(GT)和prediction这两个bbox重合时无法更新loss



- DIOU

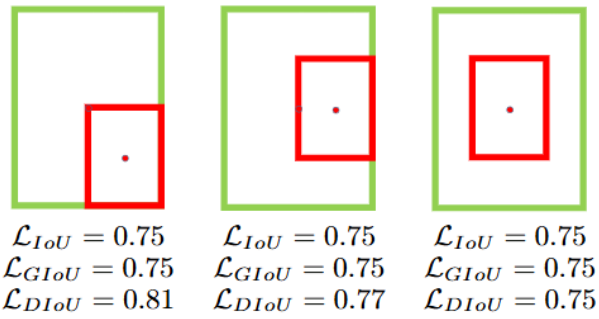
$$DIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2}$$

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

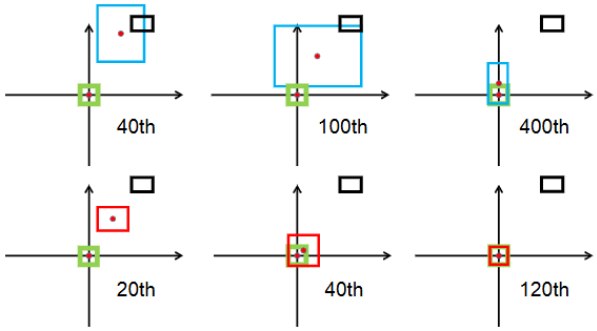


(图中的d是中心点的距离，即为公式中的$\rho^2(b, b^{gt})$)

DIoU的使用引入了中心点的距离，解决了GIou的不足



同时加快了收敛速度



- CIOU
  "a good loss for bounding box regression should consider three important geometric factors, i.e., overlap area, central point distance and aspect ratio"

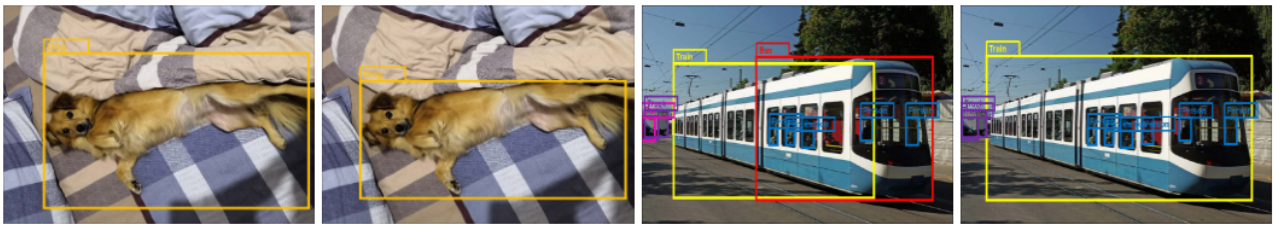
$$CIoU = IoU - (\frac{\rho^2(b, b^{gt})}{c^2} + \alpha v)$$

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

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

$$L_{CIoU} = 1 - IoU + \rho 2(b, bgt)c2 + \alpha v$$

Loss / Evaluation	AP		AP75	
	IoU	GIoU	IoU	GIoU
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
$\mathcal{L}_{CIoU}$	**49.21**	**48.42**	**54.28**	**52.87**
Relative improv. %	**5.67%**	**5.67%**	**8.95%**	**8.43%**
$\mathcal{L}_{CIoU}(D)$	**49.32**	**48.54**	**54.74**	**53.30**
Relative improv. %	**5.91%**	**5.94%**	**9.88%**	**9.31%**



$\mathcal{L}_{GIoU}$ $\mathcal{L}_{CIoU}$ $\mathcal{L}_{GIoU}$ $\mathcal{L}_{CIoU}$

- 小结
  - IoU_Loss：考虑检测框和目标框重叠面积。
  - GIoU_Loss：在IOU的基础上，解决边界框不重合时的问题。
  - DIoU_Loss：在IOU和GIOU的基础上，考虑边界框中心点距离的信息。
  - CIoU_Loss：在DIOU的基础上，考虑边界框长宽比的尺度信息

2.6 Eliminate grid sensitivity

Direct location prediction

- For example, we can create 5 anchor boxes with the following shapes.
- Rather than predicting 5 arbitrary boundary boxes, we predict offsets relative to each anchor box.

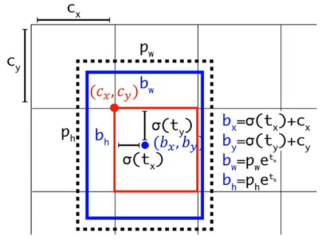


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

- The network predicts 5 coordinates for each predicted box

t_x, t_y, t_w, t_h, and t_o.

If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to

$b_x = \sigma(t_x) + c_x$
$b_y = \sigma(t_y) + c_y$
$b_w = p_w e^{t_w}$
$b_h = p_h e^{t_h}$
$\Pr(\text{object}) * \text{IoU}(b, \text{object}) = \sigma(t_o) = \hat{C}$

5% mAP Improvement

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

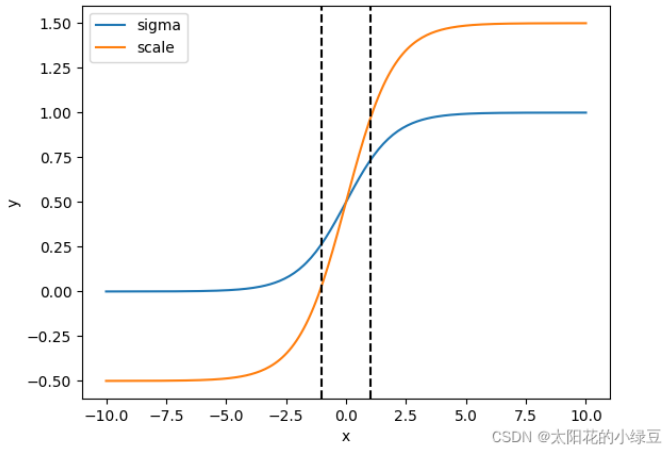
其中：t_x 是网络预测的目标中心x坐标偏移量（相对于网格的左上角） t_y是网络预测的目标中心y坐标偏移量（相对于网格的左上角） c_x 是对应网格左上角的x坐标 c_y是对应网格左上角的y坐标 但在YOLOv4的论文中作者认为这样做不太合理，比如当真实目标中心点非常靠近网格的边界时，网络的预测值需要负无穷或者正无穷时

才能取到，而这种很极端的值网络一般无法达到。为了解决这个问题，作者引入了一个大于1的缩放系数$scale_{xy}$

$$b_x = (\sigma(t_x) * scale_{xy} - \frac{scale_{xy} - 1}{2}) + c_x$$

$$b_y = (\sigma(t_y) * scale_{xy} - \frac{scale_{xy} - 1}{2}) + c_y$$

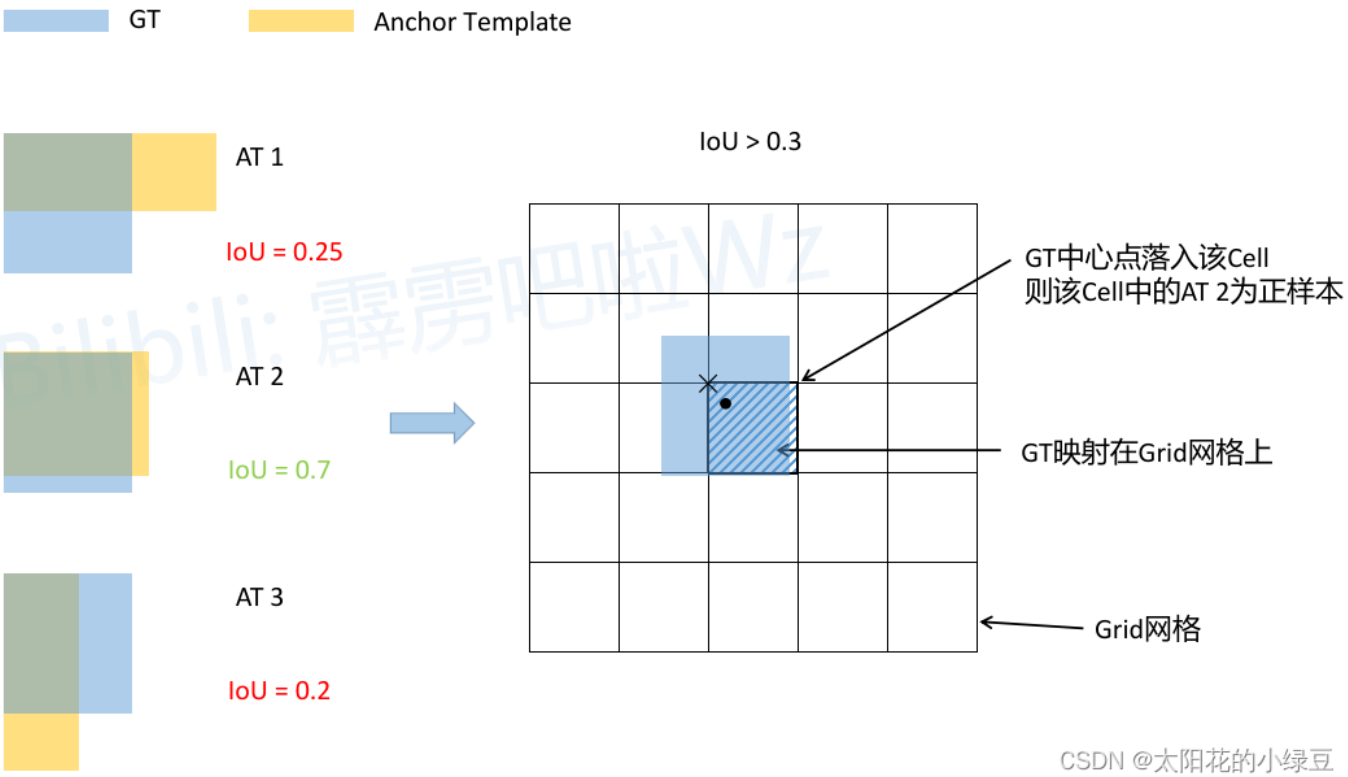
yolov5中将$scale_{xy}$设置为2，sigma函数图像缩放为如下图，通过引入缩放系数scale以后，x在同样的区间内，y的取值范围更大，由原来的(0,1)调整到了(−0.5,1.5)，且y更容易取到0和1。



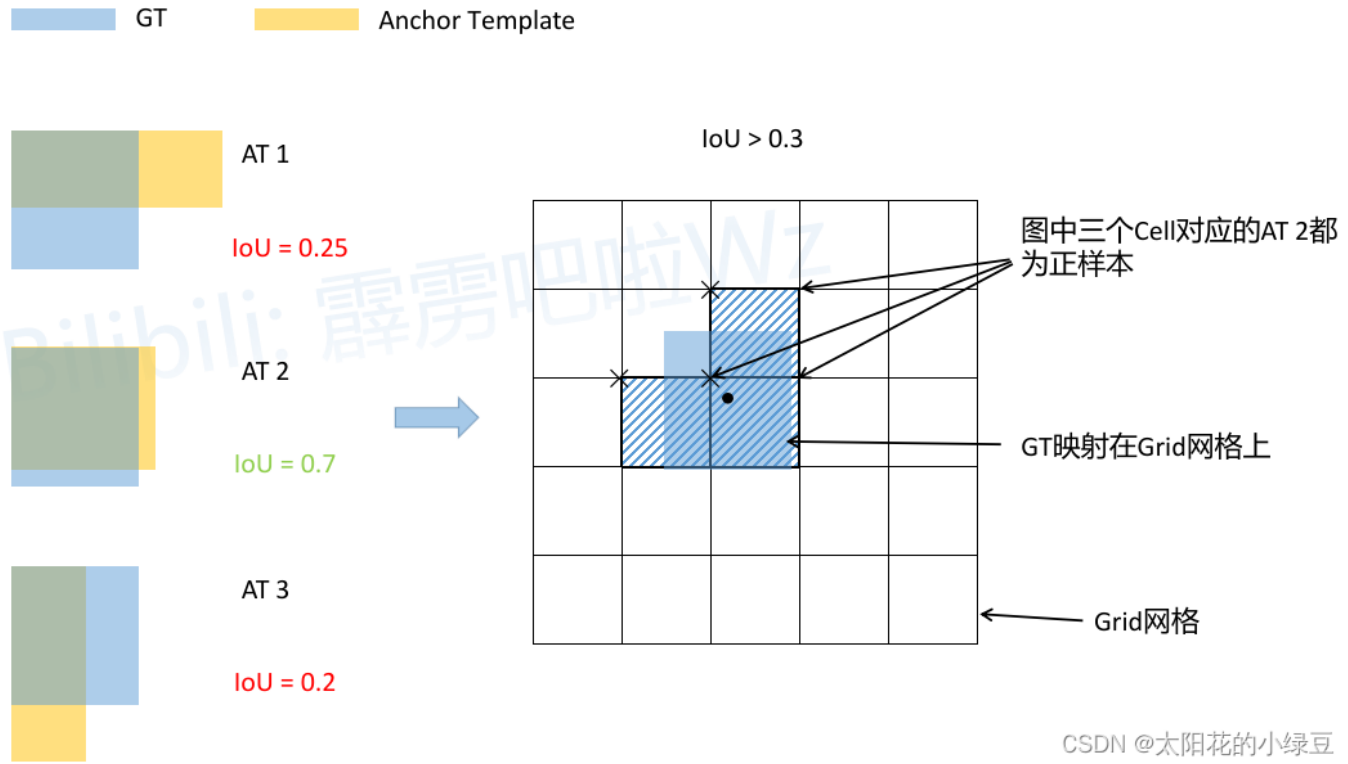
2.7 Match Positive Samples

在YOLOv3中针对每一个GT都只匹配了一个Anchor,但在YOLOv4中一个GT可以同时匹配多个Anchor模板

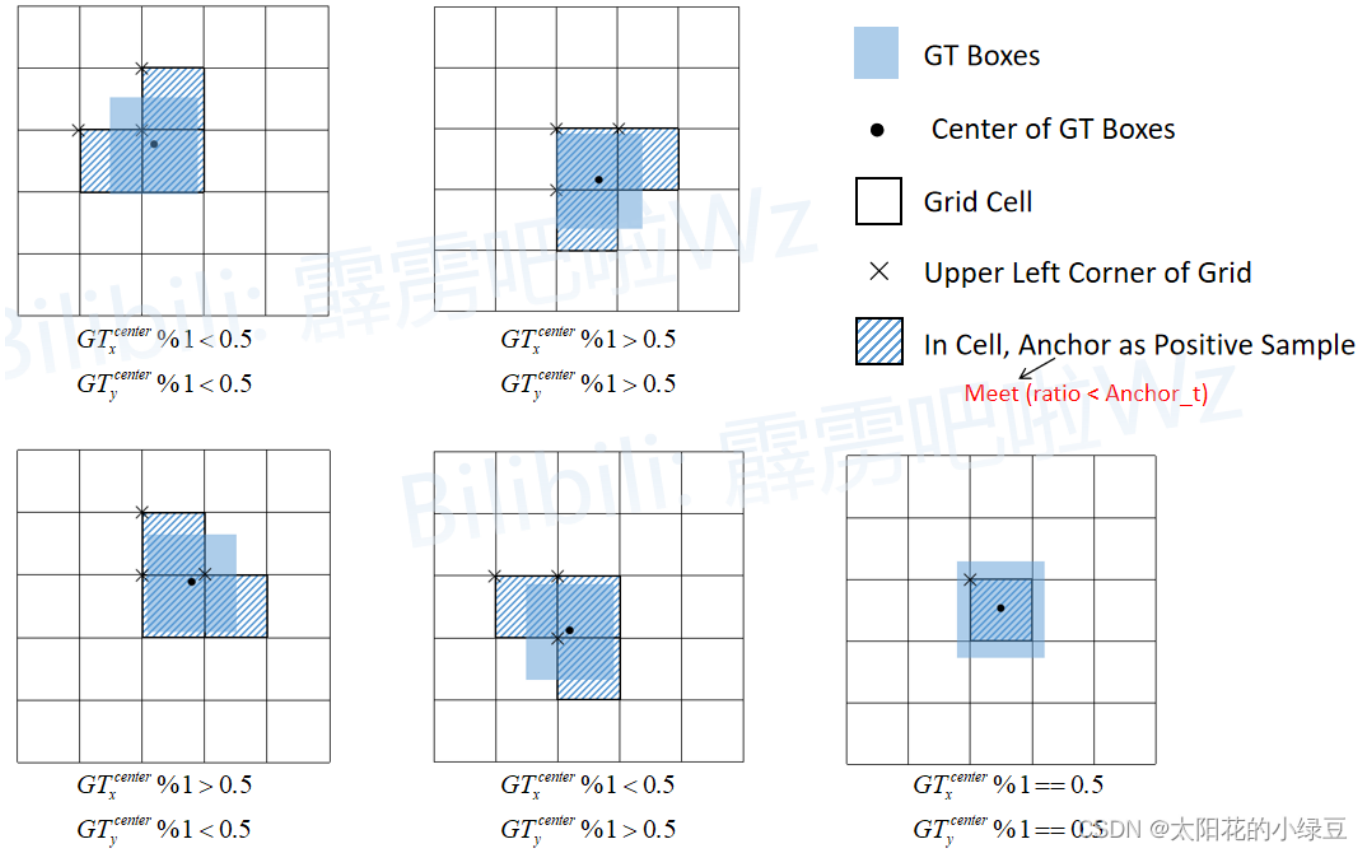
1. 将每个GT Boxes与每个Anchor模板进行匹配（这里直接将GT和Anchor模板左上角对齐，然后计算IoU）
2. 如果GT与某个Anchor模板的IoU大于给定的阈值，则将GT分配给该Anchor模板，如图中的AT 2
3. 将GT投影到对应预测特征层上，根据GT的中心点定位到对应grid cell（图中黑色的×表示cell的左上角）
4. 则该grid cell对应的AT2为正样本



但在YOLOv4中关于匹配正样本的方法又有些许不同。主要原因在于2.6 Eliminate grid sensitivity中提到的缩放因子$scale_{xy}$,通过缩放后网络预测中心点的偏移范围已经从原来的(0,1)调整到了(−0.5,1.5)。所以对于同一个GT Boxes可以分配给更多的Anchor，即正样本的数量更多了。



1. 将每个GT Boxes与每个Anchor模板进行匹配（这里直接将GT和Anchor模板左上角对齐，然后计算IoU）
2. 如果GT与某个Anchor模板的IoU大于给定的阈值，则将GT分配给该Anchor模板，如图中的AT 2（如果AT 1也满足，则同时分配给AT 1和 AT 2）
3. 将GT投影到对应预测特征层上，根据GT的中心点定位到对应grid cell,并根据GT中心点在cell中的位置选择是否扩展正样本，以及扩展的grid cell位置
4. 如扩展，则这三个grid cell对应的AT2均为正样本



刚刚说了网络预测中心点的偏移范围已经调整到了(−0.5,1.5)，所以按理说只要Grid Cell左上角点距离GT中心点在(−0.5,1.5)范围内它们对应的Anchor都能漂移到GT的位置处。在回过头看看刚刚上面的例子，GT_x^{center}，GT_y^{center} 它们距离落入的Grid Cell左上角距离都小于0.5，所以该Grid Cell上方的Cell以及左侧的Cell都满足条件，即Cell左上角点距离GT中心在(−0.5,1.5)范围内。这样会让正样本的数量得到大量的扩充。但是，YOLOv5源码中扩展Cell时只会往上、下、左、右四个方向扩展，不会往左上、右上、左下、右下方向扩展。

Reference:

1. https://blog.csdn.net/qq_37541097/article/details/123229946
2. https://arxiv.org/pdf/2004.10934.pdf
3. https://zhuanlan.zhihu.com/p/358710763
4. https://blog.csdn.net/c2250645962/article/details/106210730