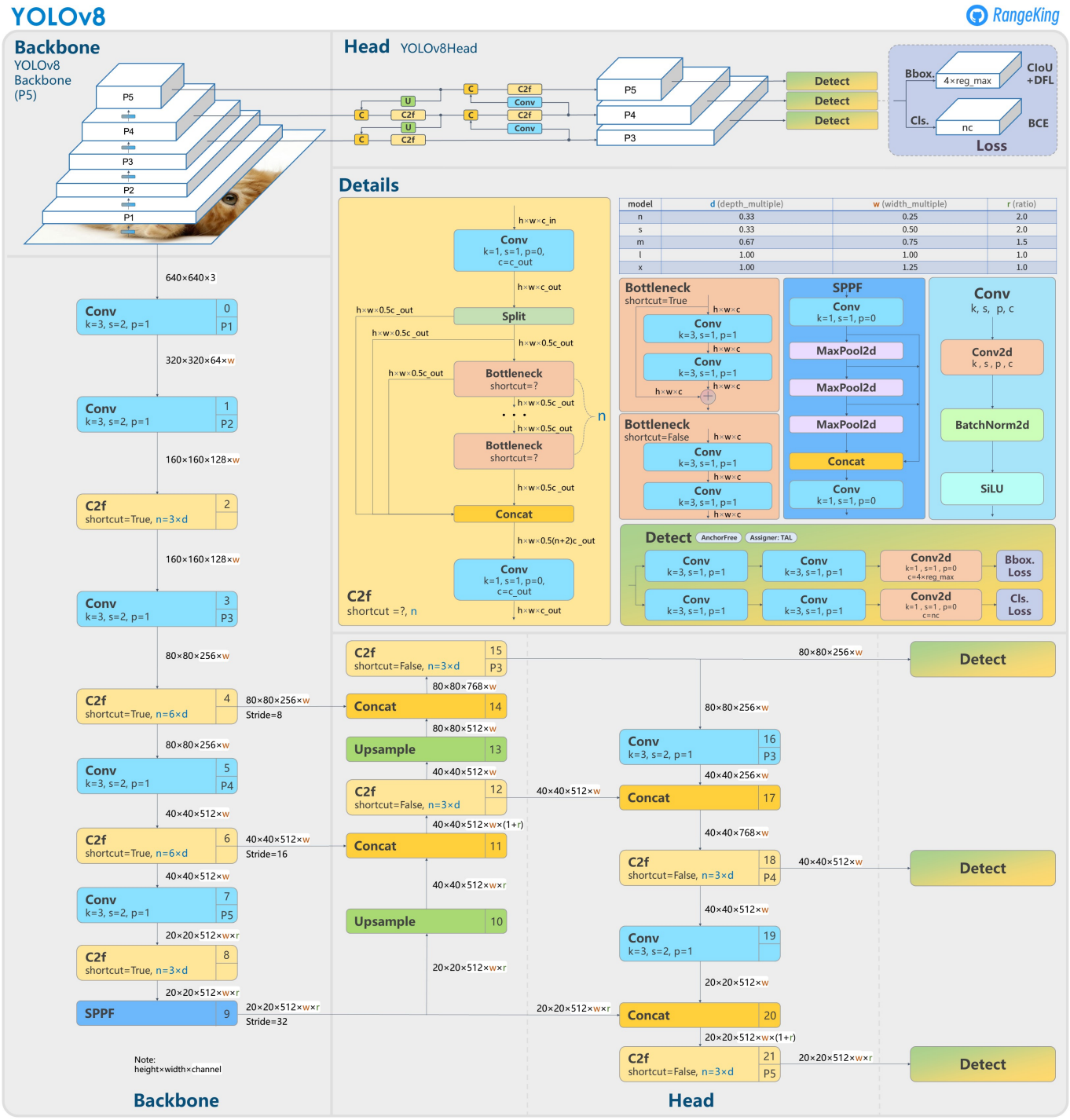
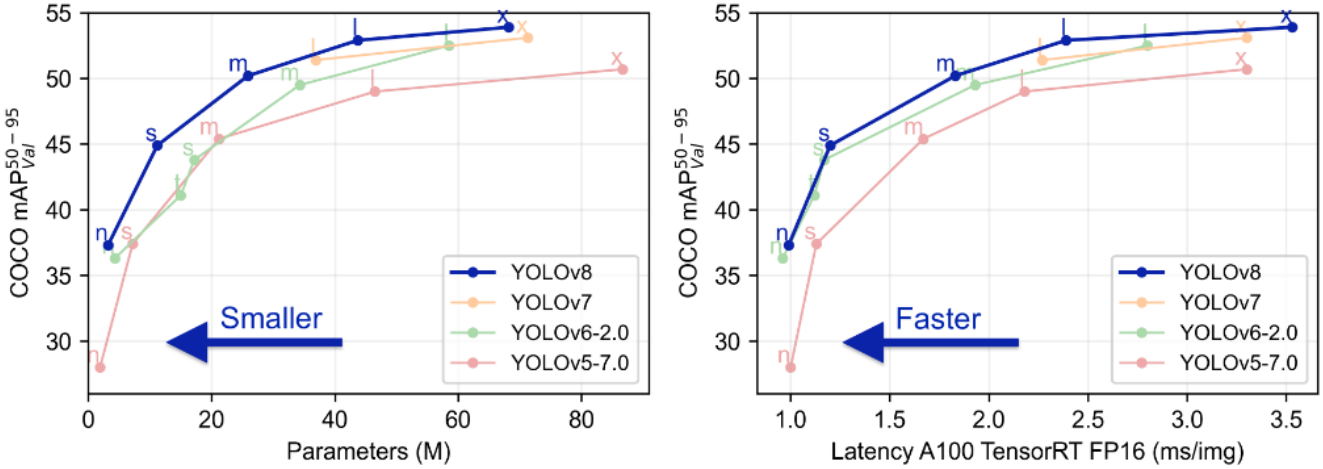
[RangeKing绘制YOLOv8结构图]



[YOLOv8 vs YOLOv5]



上表所示是基于COCO Val 2017数据集测试并对比Yolov8和Yolov5的mAP、参数量和FLOPs结果。由此可以看出，Yolov8相比Yolov5精度提升比较多，n/s/m模型参数量和flops增加不少，相比Yolov5大部分模型推理速度变慢了。

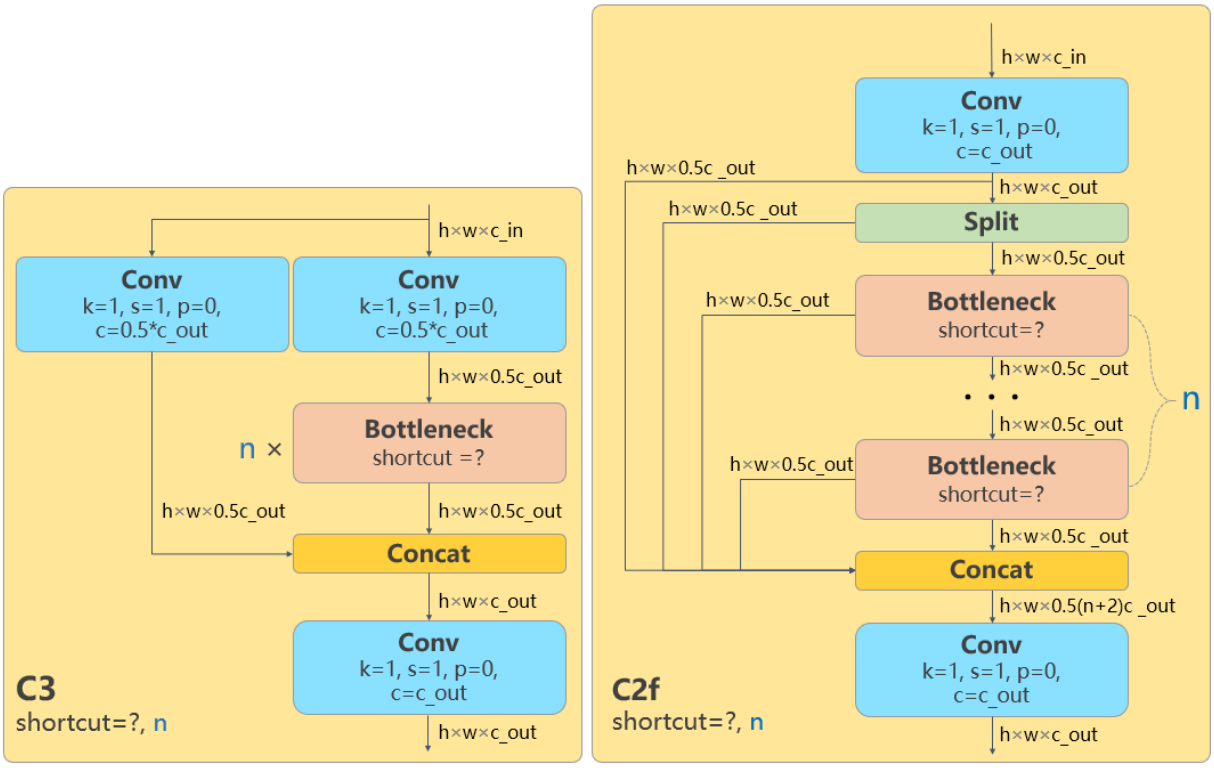
YOLOv8的主要改动

- backbone
  - 核心block由 **C3** 变为 **C2f**
- head
  - 由 **Anchor-based** 转为 **Anchor-free**，输出特征图改变
- 损失函数计算
  - 引入了 **Distribution Focal Loss(DFL)**

1. Backbone (CSPDarknet)

C3->C2f

[C3和C2f对比图]（有两处错误）



Q. 还记得C3吗？

"The C2 module in YOLOv8 stands for the CSP (Cross Stage Partial) Bottleneck with 2 convolutions.

The C2f module is a faster implementation of the C2 module. It improves the execution speed of the model while maintaining similar performance. This optimization is achieved by making certain modifications to the original C2 module." --Ultralytics

不过针对[为什么C2f比C2快的issue]，官方的回应也并不能使人信服，让我们还是关注对比C2f和C3吧

优点：C2f借鉴ELAN的多分支思想，使得**梯度传播的路径**更加丰富
不足：C2f模块中存在 Split 等操作对特定硬件部署没有之前那么友好

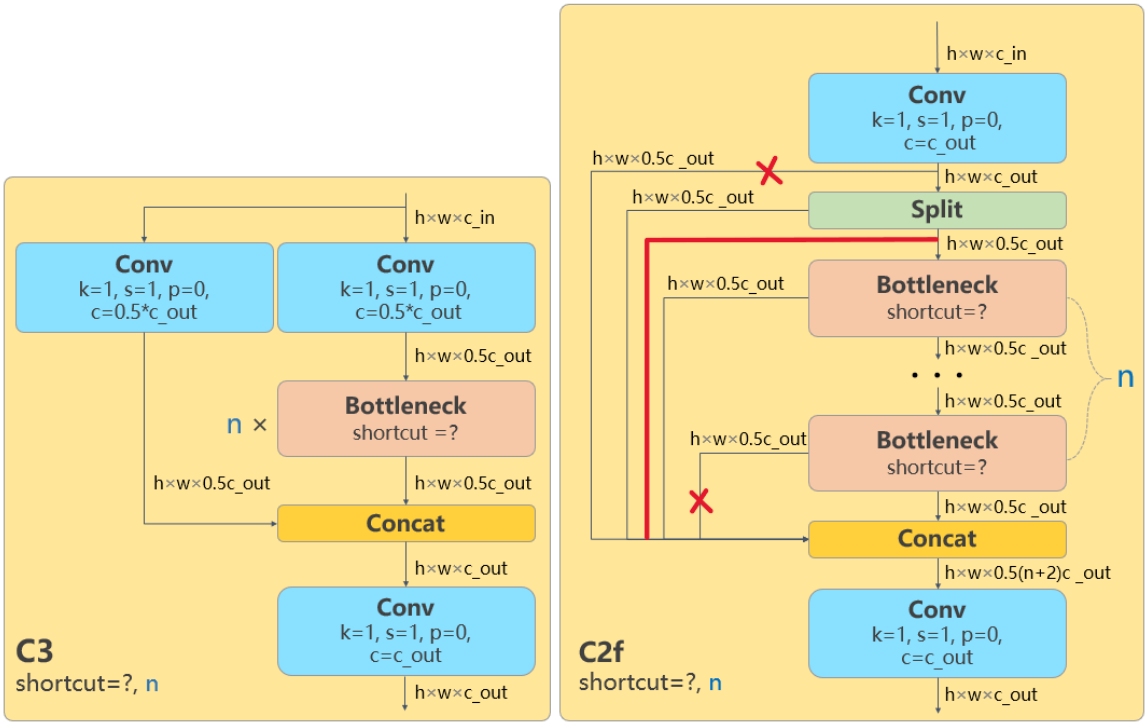
其他改动

- 第一个卷积层的Kernel size从6×6改为3x3
- Block数由C3模块3-6-9-3改为C2f模块的3-6-6-3
  （可能是为了在一定程度上减少参数量）

再次回到C2f的结构，让我们来看看代码，解决前面遗留的问题

```
class C2f(nn.Module):
    """Faster Implementation of CSP Bottleneck with 2 convolutions."""
    def __init__(self, c1: int, c2: int, n: int = 1, shortcut: bool = False, g:
int = 1, e: float = 0.5):
        """
        Initialize a CSP bottleneck with 2 convolutions.
        Args:
            c1 (int): Input channels.
            c2 (int): Output channels.
            n (int): Number of Bottleneck blocks.
            shortcut (bool): Whether to use shortcut connections.
            g (int): Groups for convolutions.
            e (float): Expansion ratio.
        """
        super().__init__()
        self.c = int(c2 * e)  # hidden channels
        self.cv1 = Conv(c1, 2 * self.c, 1, 1)
        self.cv2 = Conv((2 + n) * self.c, c2, 1)  # optional act=FReLU(c2)
        self.m = nn.ModuleList(Bottleneck(self.c, self.c, shortcut, g, k=((3, 3),
(3, 3)), e=1.0) for _ in range(n))
```

在倒数第三行可以看到concat数是0.5(n+2)没问题，说明是原图中多画了一条线，另外第一条分支应该调整到Split之后再引出，错误的两处在下图标出

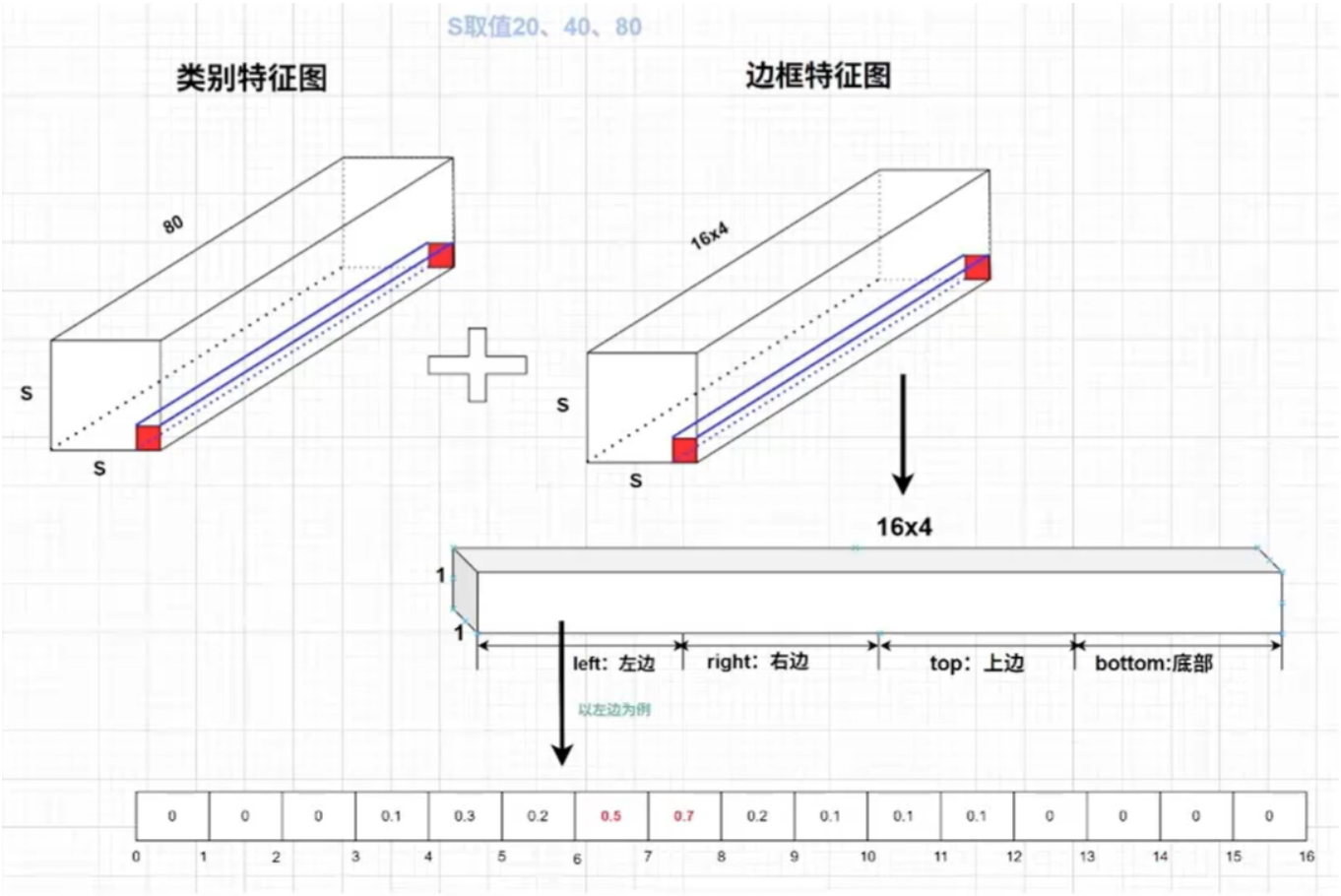


2. 训练策略

2.1 Anchor-free

[Anchor-free图解]

[输出特征图]



解耦头中负责预测坐标的特征图大小为$s*s*64(s=20,40,80)$

- 64:=4x16
- 16:以grid cell中心点为基准，输出预测框向左，右，上，下四个方向的偏移量l,r,t,b，以left方向为例，并不会直接输出具体的偏移量，而是会输出一个list,记录了从偏移量为{0,1,2,...,7,8,...,14,15}的概率分布，计算偏移量期望得到最终的偏移

16在代码中是一个参数，叫做reg_max，可以根据下采样的倍数，输入的尺寸进行调整。yolov8中最小的输出特征图20 * 20，1个像素对应原图中32个像素，即感受野是32，reg_max设置为16，则单一方向的偏移量最大为15，预测框的大小最大为30 * 30，对应原图中的大小为30 * 32=960，大于输入尺寸的宽度640，说明16足以满足大目标的检测。

Q. reg_max=16是怎么得到的？

640大小的输入经过1/32倍的降采样得到20 * 20大小的特征图，为了能够覆盖原始输入大小，预测框的大小最大至少也应为20 * 20，即单一方向的偏移量至少应为10，而16是大于10的最小的2的幂次方，这样可以使参数调整更加灵活，比如输入变为320 * 320时，reg_max直接调成为8即可，假如输入为1280 * 1280，调成32即可。

- yolov3 anchor-based: 一个grid对应三个anchor,即三个预测框
- yolov8 anchor-free: 一个grid对应一个预测框

yolov8预测框的总数：80x80+40x40+20x20=8400

注意： yolov8已无anchor概念，网络上沿用的anchor描述实际上是想表达预测框，anchor point实际指的是grid cell中心点，不要被误导

2.2 样本匹配

①预处理： 边框：还原到网络输入尺度 中心：还原到网络输入尺度

②初筛：对所有grid cell中心点在GT框中的预测框作为初步正样本

③精筛：计算初步正样本的对齐分数alignment_metric，选择分数最高的top-N(默认取10)作为正样本。

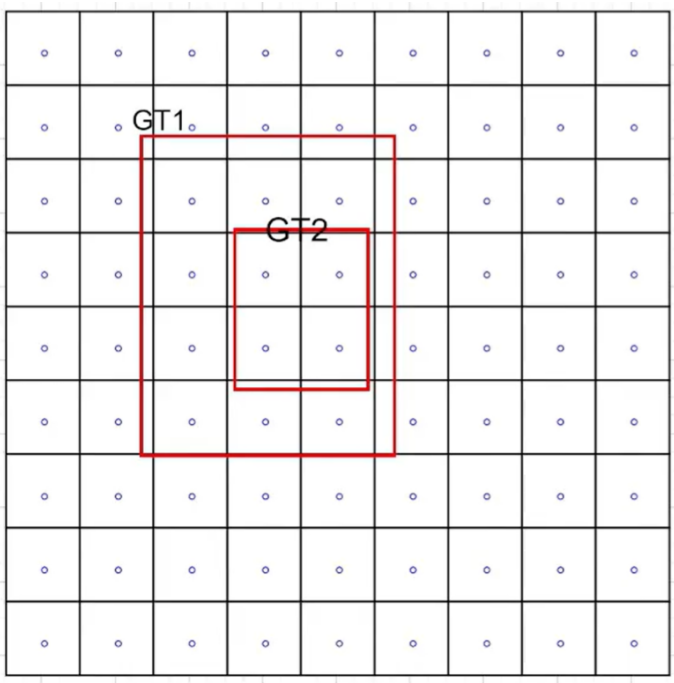
④去除重复分配：通过mask矩阵与anchor求和，找出分给多个GT框的anchor，只留下CIOU值大的预测框

Task alignment learning(TAL)任务对齐学习

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

s 和 u 分别为分类得分和 IoU 值，α 和 β 为权重超参(v8默认值α=0.5、β=6.0)

$$t=class_confi_score^{0.5}+CIOU^{6}$$



为什么anchor-free了?

Anchor-Free通过不依赖数据集中的先验知识，使网络对“物体形状”有更好的表达能力，更具泛化潜能。在运动物体、尺寸不一物体检测上有所提升，同时检测被遮挡物体时也能更加灵活。Anchor-Based依赖于训练数据集的先验知识，天然的会诱导模型过拟合以往见过的情况。

3. 损失函数计算

$$Loss = \gamma_1 L_{cls} + \gamma_2 L_{CIOU} + \gamma_3 L_{DFL}$$

yolov8对$\gamma 1$,$\gamma 2$,$\gamma 3$默认分别取值0.5, 7.5, 1.5

观察上式不难发现，回归损失在CIOU_Loss的基础上新增了**Distribution Focal Loss(DFL)**

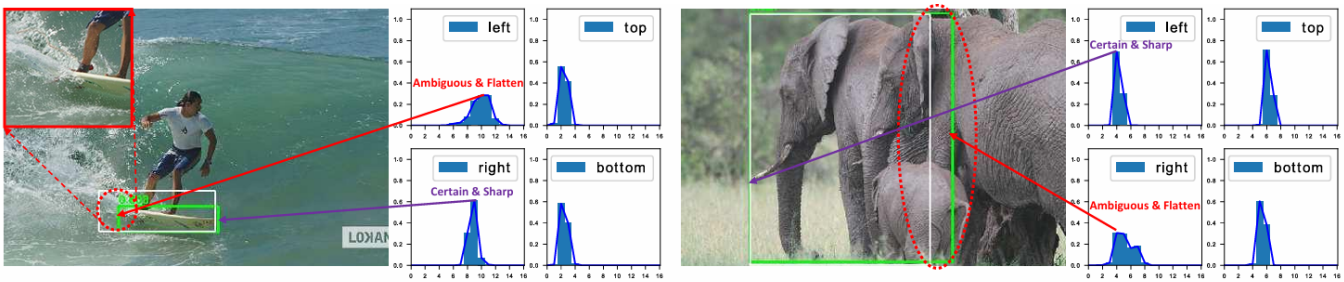
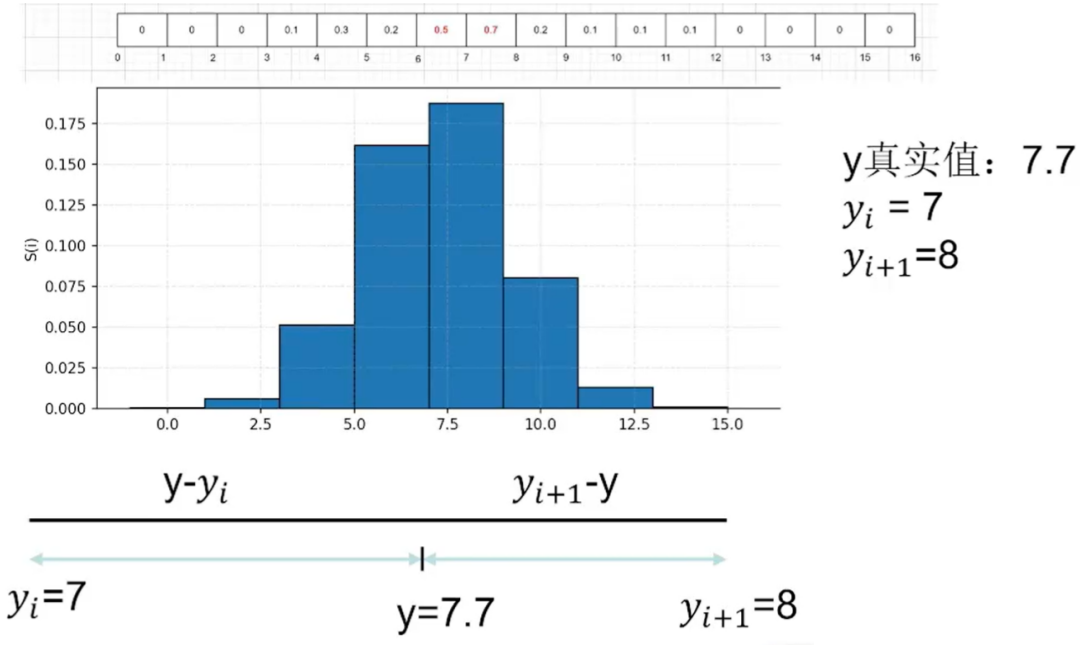


Figure 3: Due to occlusion, shadow, blur, etc., the boundaries of many objects are not clear enough, so that the ground-truth labels (white boxes) are sometimes not credible and Dirac delta distribution is limited to indicate such issues. Instead, the proposed learned representation of General distribution for bounding boxes can reflect the underlying information by its shape, where a flatten distribution denotes the unclear and ambiguous boundaries (see red circles) and a sharp one stands for the clear cases. The predicted boxes by our model are marked green.

数据集中有一类难样本，难以用准确的边界框来框定它们，比如左图中的冲浪板，它的左边有一部分被海浪遮挡，再比如右图中，左边大象右边的部分被第二只大象的头遮挡，这种情况下在真实值附近的真实值都会被分配很大的概率

对left, right, top, bottom在不同偏移量(y)对应的概率(x)绘制直方图可视化，在左图对应的left直方图中，呈现出一块"Flatten"区域，同样在右图对应的right直方图中也存在"Flatten"区域，而其他的直方图都呈现"Sharp"的形状。这与上面对难样本的分析是一致的。然而我们希望直方图能呈"Sharp"的形状而不要存在"Flatten & Ambiguous"的情况，因此要在损失函数中进行奖惩，和真实值近的预测具有更高的概率，而削弱离真实值较远的预测值概率。



DFL 只考虑真实值左边和右边的一个预测值，根据距离为其赋予不同的权重

$$DFL(S_i, S_{i+1}) = -((y_{i+1} - y)log(S_i) + (y - yi)log(S_{i+1}))$$

$$-log(S_i) = -log(\frac{e^{x_i}}{\Sigma_j^{e^{x_j}}})$$

S_i是对概率x_i的Softmax处理

通过降低Loss,y=8的概率会增大，y=7的概率会减小，使直方图呈现"Sharp"的形状

分类损失实操时仍然采用BCE（虽然也提供了Varifocal Loss VFL的用法）