

通用视觉框架OpenMMLab
第4讲 目标检测与MMDetection (下)

陈恺
2021年5月



本节内容：

- 单阶段目标检测算法
 - YOLO 系列模型
 - SSD 模型
 - RetinaNet 模型
- 无锚框目标检测算法
 - FCOS 模型
- 实例分割
 - Mask RCNN 模型
- 检测模型的评估方法
- 实践 MMDetection 2
 - 目标检测模型的训练与评估

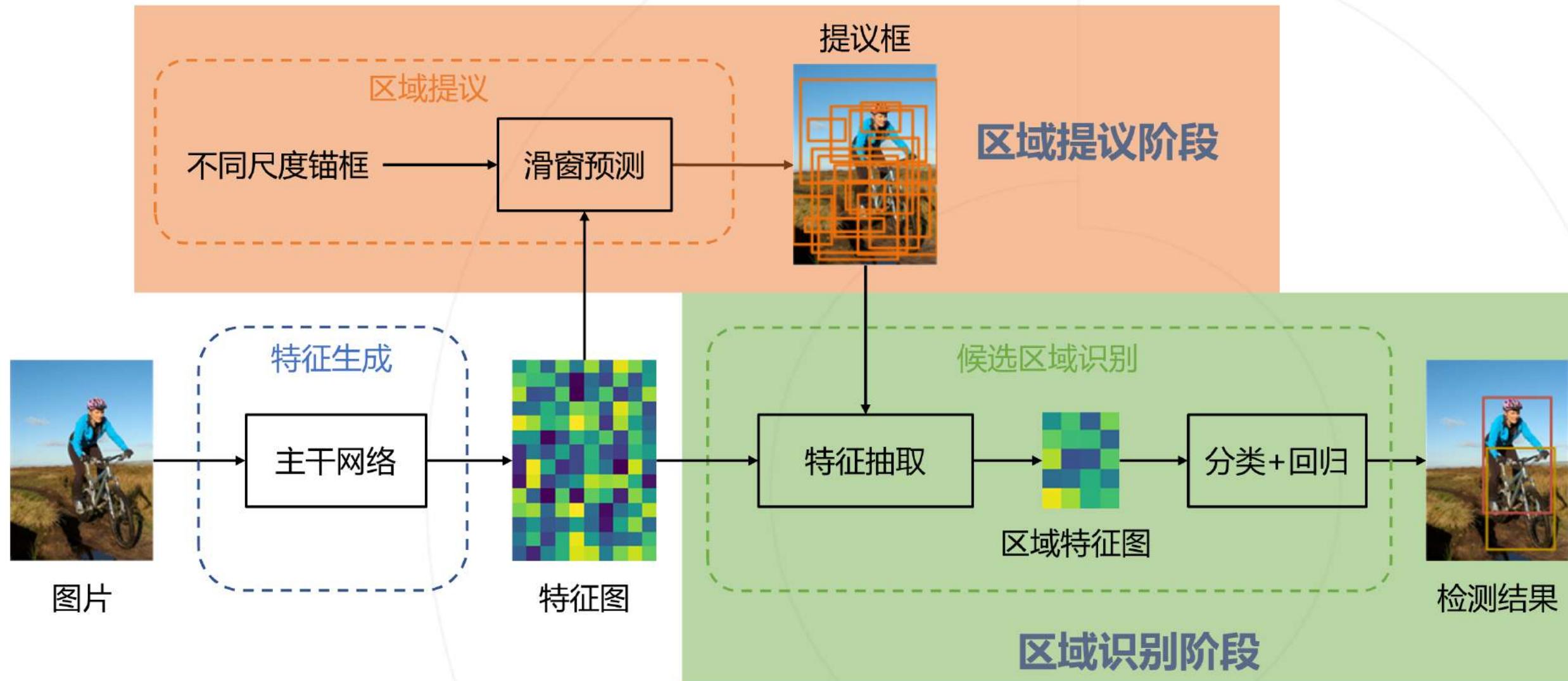


上节回顾：

- 目标检测的基本思想
- 两阶段算法 Two-staged Methods
 - RCNN 算法
 - Fast RCNN 算法
 - Faster RCNN 算法
 - FPN 算法
- 实践 MMDetection 1
 - 使用预训练模型进行推理

单阶段目标检测算法

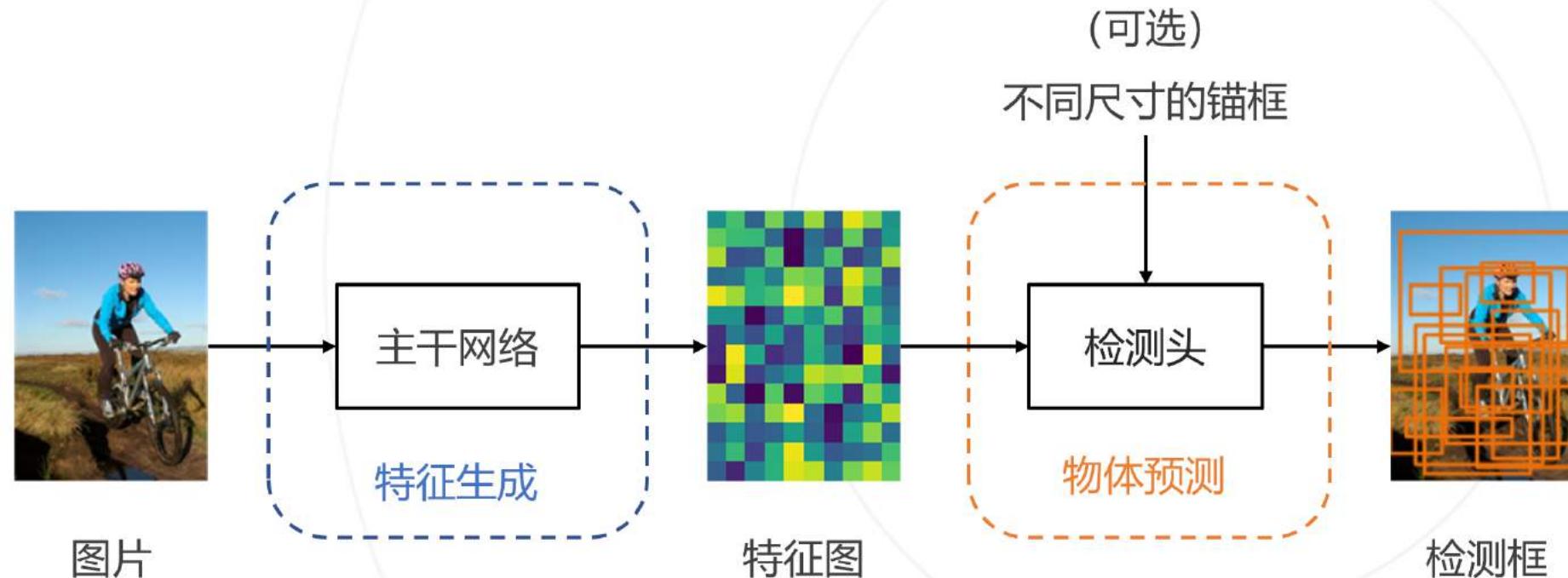
Single-staged Methods



特点：没有区域提议阶段，直接基于特征图产生类别和边界框预测

优点：推理速度快，结构简单，易于在不同设备上部署

缺点：多数情况下性能不如两阶段算法



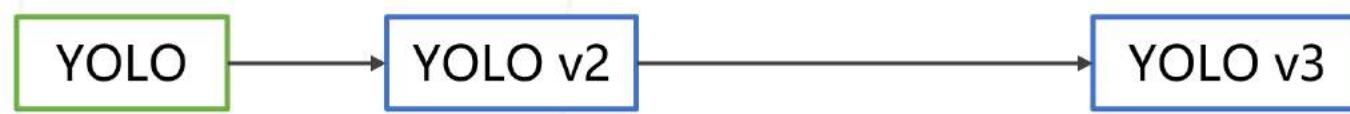


两阶段算法

单阶段算法

无锚框算法

基于特征图直接回归边界框



基于多尺度特征图与锚框

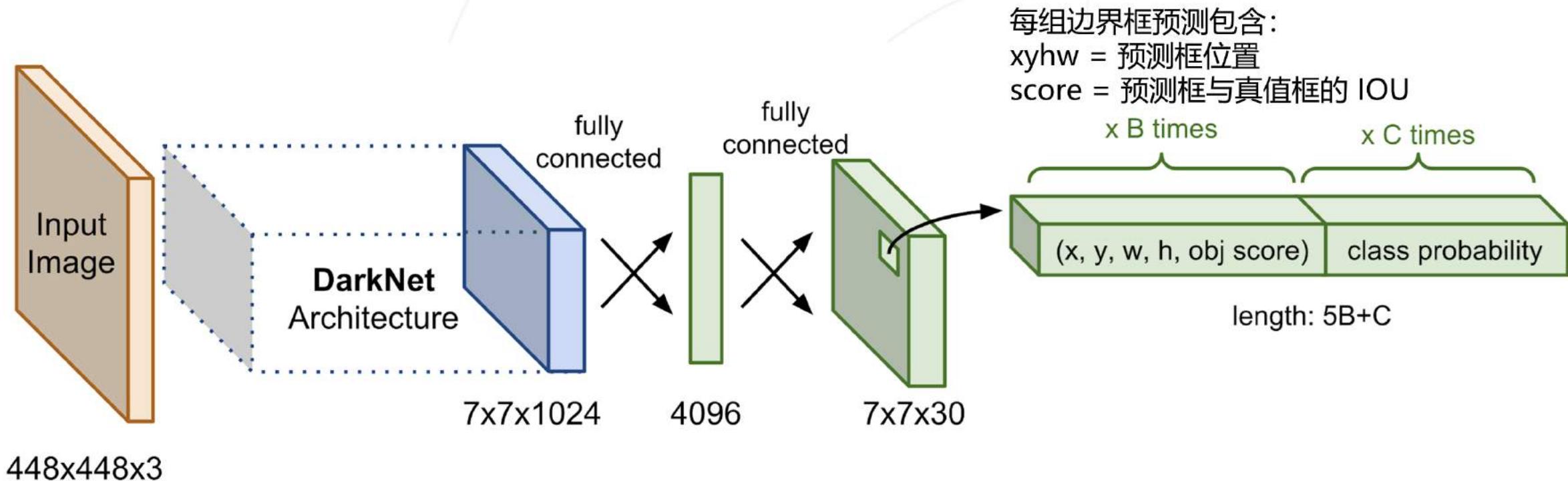


提出 Focal Loss

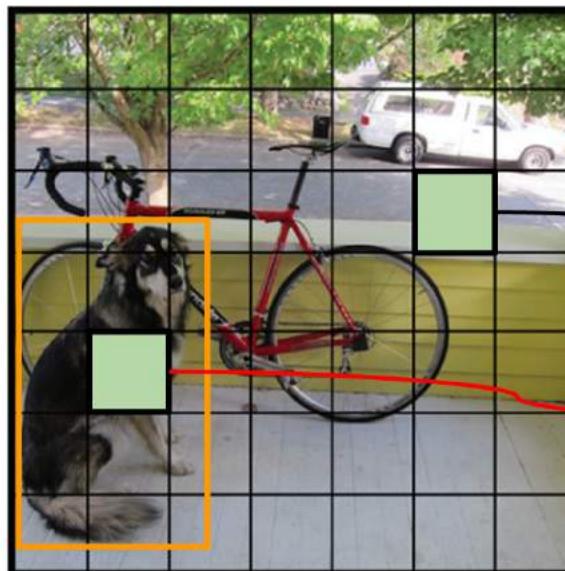
FCOS

多尺度特征图直接回归

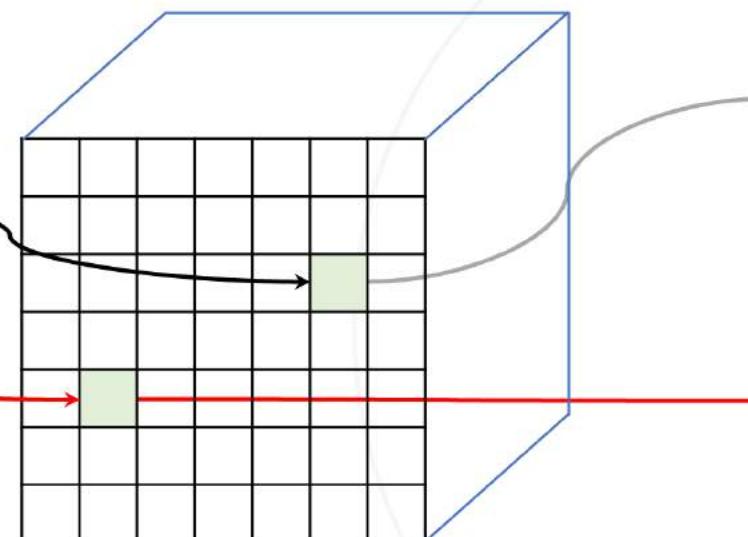
- 最早的单阶段算法之一
- DarkNet 结构的主干网络产生特征图
- 全连接层产生全部空间位置的预测结果，每个位置包含 C 维分类概率和 B 组边界框预测



- 将原图切分成 $S \times S$ 大小的格子，对应预测图上 $S \times S$ 个像素位置。
- 如果原图上某个物体的中心落于某个格子内，则对应位置的预测值应给出该物体的类别和边界框的位置（基于格子边界的偏移量）。
- 其余位置应预测为背景类别，不关心边界框预测结果。



将图像划分为 7×7 的格子



7 \times 7 分辨率的预测图
每个位置 $5B+C$ 个通道
 $= B$ 个框 + C 个类别的预测

类别预测 = 背景
边界框不计算 loss

类别预测 = 狗
边界框预测：
 X, Y = 相对于格子边界的偏移量
 W, H = 窗大小 / 图像大小
分数 = 预测框与真值框的交并比

多任务学习：回归、分类共同计入损失函数，通过 λ 控制权重

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

边界框需要产生**物体**预测时，
计算边界框坐标的回归损失

边界框需要产生**类别（物体/背景）**预测时，
计算边界框置信度的回归损失

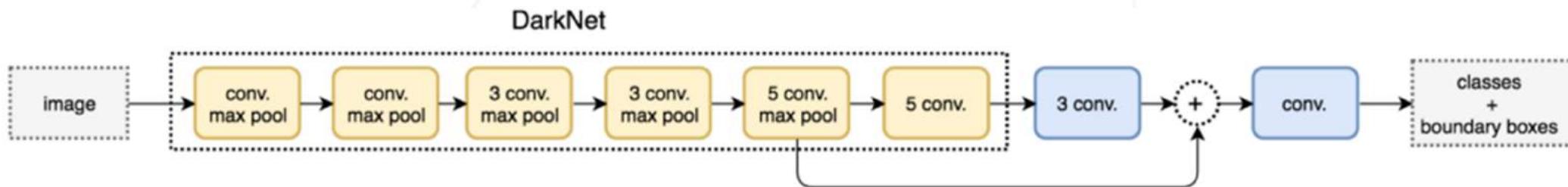
边界框需要产生**物体**预测时，
计算C个类别概率的回归损失

- 快！在Pascal VOC 数据集上，使用自己设计的 DarkNet 结构可以达到实时速度，使用相同的 VGG 可以达到 3 倍于 Faster RCNN 的速度
- 不依赖锚框，直接回归边界框

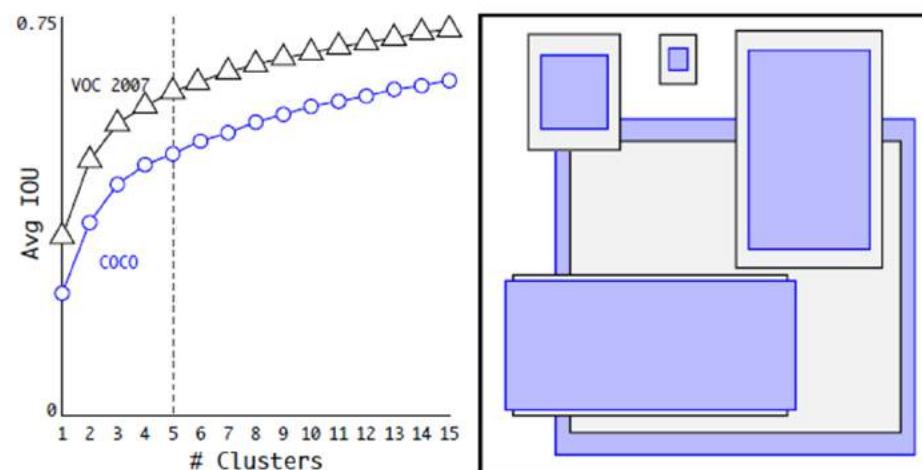
检测算法	主干网络	训练集	检测精度 (mAP)	检测速度 (FPS)
Fast YOLO	9层 DarkNet	VOC 2007 + 2012	52.7	155
YOLO	24层 DarkNet	VOC 2007 + 2012	63.4	45
Faster R-CNN VGG16	VGG 16	VOC 2007 + 2012	73.2	7
YOLO VGG 16	VGG 16	VOC 2007 + 2012	66.4	21

- 由于每个格子只能预测 1 个物体，因此对重叠物体、尤其是大量重叠的小物体容易产生漏检
- 直接回归边界框有难度，回归误差较大

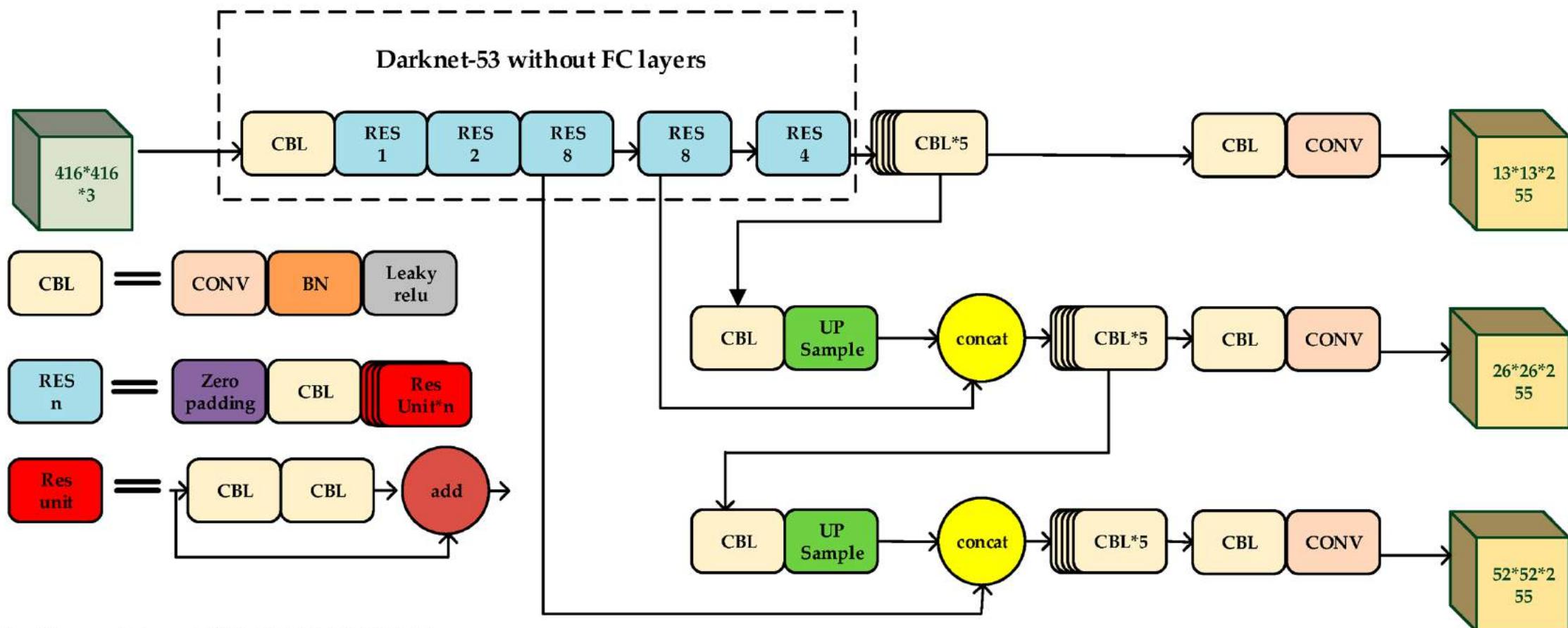
- 新的 DarkNet-19 结构的主干网络，加入 BN



- 加入锚框，并使用聚类方法设定锚框尺寸

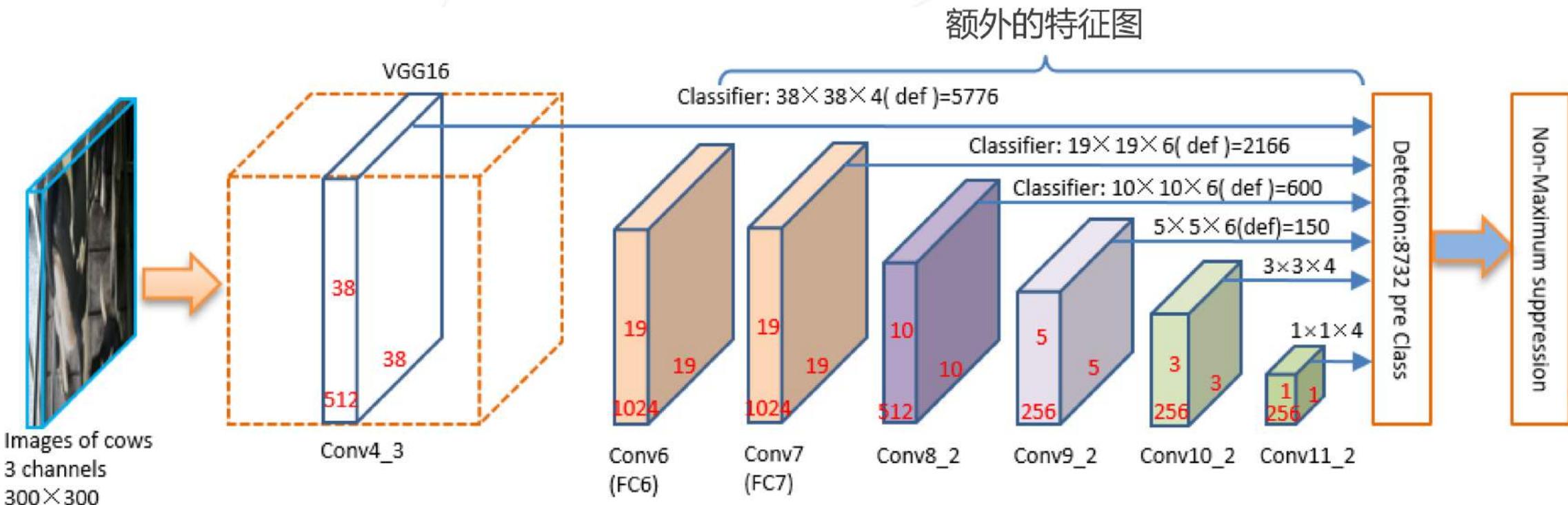


- 主干网络加入残差结构 → DarkNet 53
- 加入类 FPN 结构，基于多尺度特征图预测



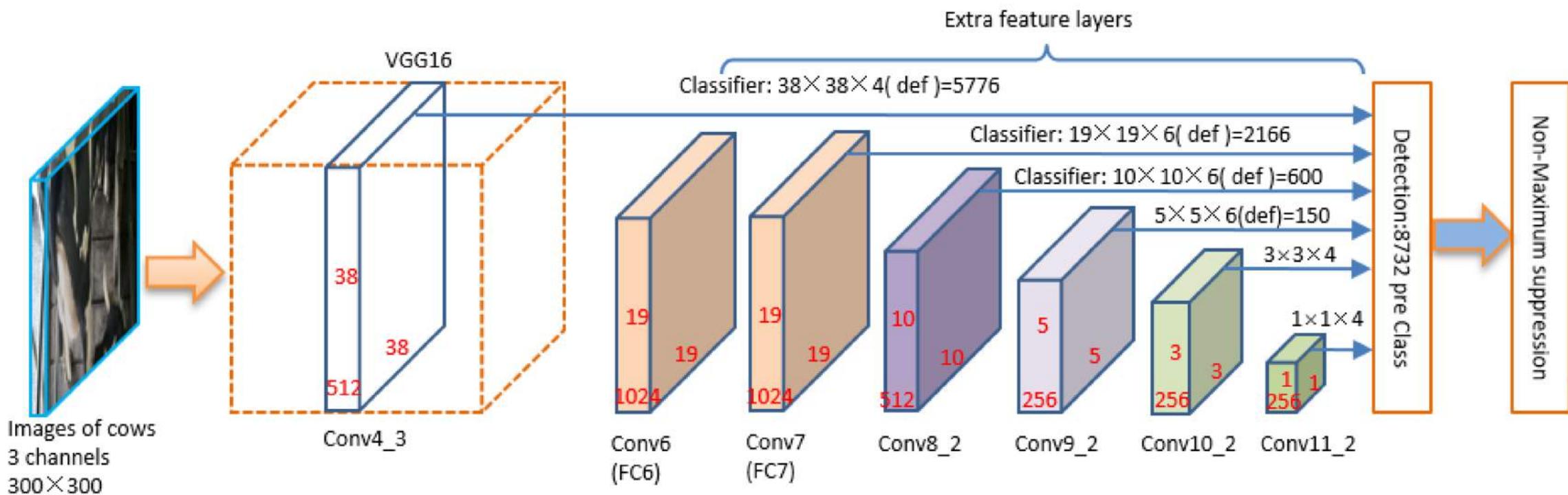
主干网络：

VGG 结构（只保留前4级卷积层） + 额外的卷积层，产生多尺度的特征图
每级特征图包含了不同尺度物体的信息



检测头：在每级特征图上，使用 3×3 卷积层产生 $(c+4)a$ 通道的预测图：共 a 个锚框（SSD原始论文中称为 default box），每个锚框预测 c 个分类概率，4个边界框偏移量。

SSD300 结构在所有特征图上产生共8732个检测框

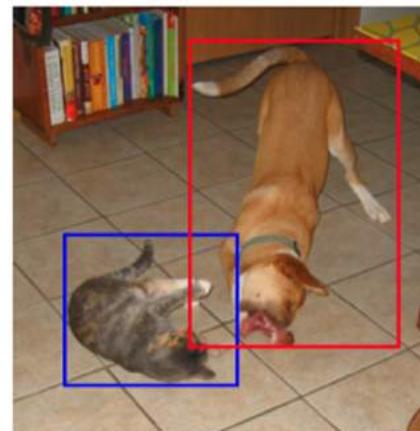


匹配：

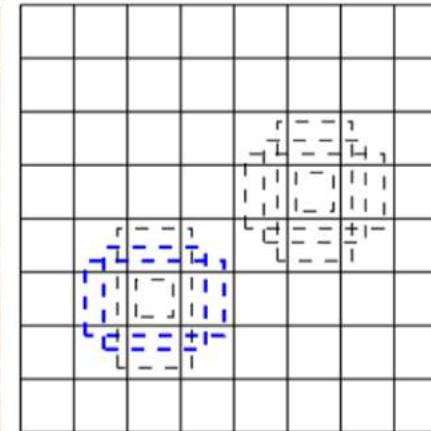
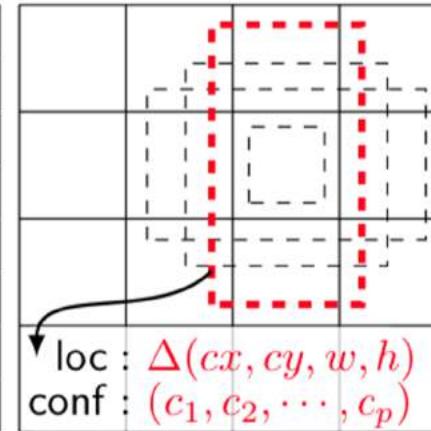
1. 将每个真值框与与其交并比最大的锚框进行匹配
2. 将剩余的每个锚框与与其交并比大于 0.5 的真值框进行匹配 → 一个真值框可匹配多个锚框

计算损失函数：

1. 匹配上的锚框为正样本，计算分类和边界框回归 loss
2. 其余框为负样本，仅计算分类 loss



(a) Image with GT boxes

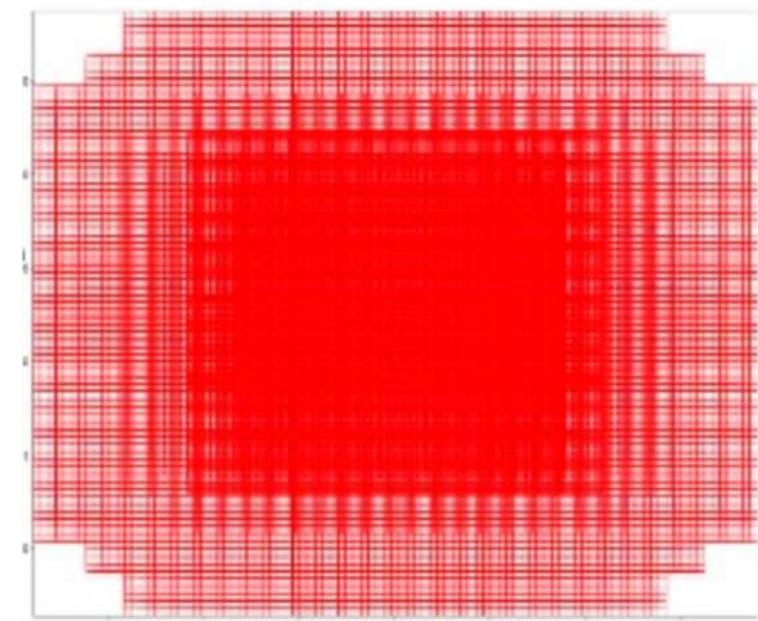
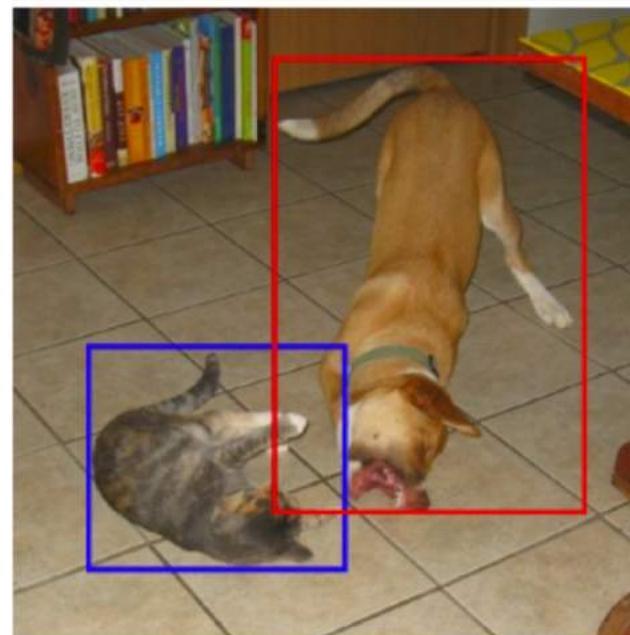
(b) 8×8 feature map(c) 4×4 feature map

锚框的数量远远大于真值框（数万 vs 数个），大量锚框是负样本。

检测头中的分类器面临正负样本不均衡问题，容易使模型对背景预测产生偏向性 (bias)。

$$L = \sum_{\text{pos boxes}} L_{\text{pos}} + \sum_{\text{neg boxes}} L_{\text{neg}}$$

数量少 数量多
↓
主导loss
↓
模型偏向背景预测



两阶段检测器通过区域提议阶段拒绝了大量负样本。检测头接受的正负样本比例并不悬殊。

单阶段检测器需要专门处理样本不均衡问题：

- YOLO 对正负样本的 loss 使用不同的权重
- SSD 在训练过程中使用**困难负样本挖掘** (Hard Negative Mining) 策略。
- RetinaNet 使用 Focal Loss (后面讲)。

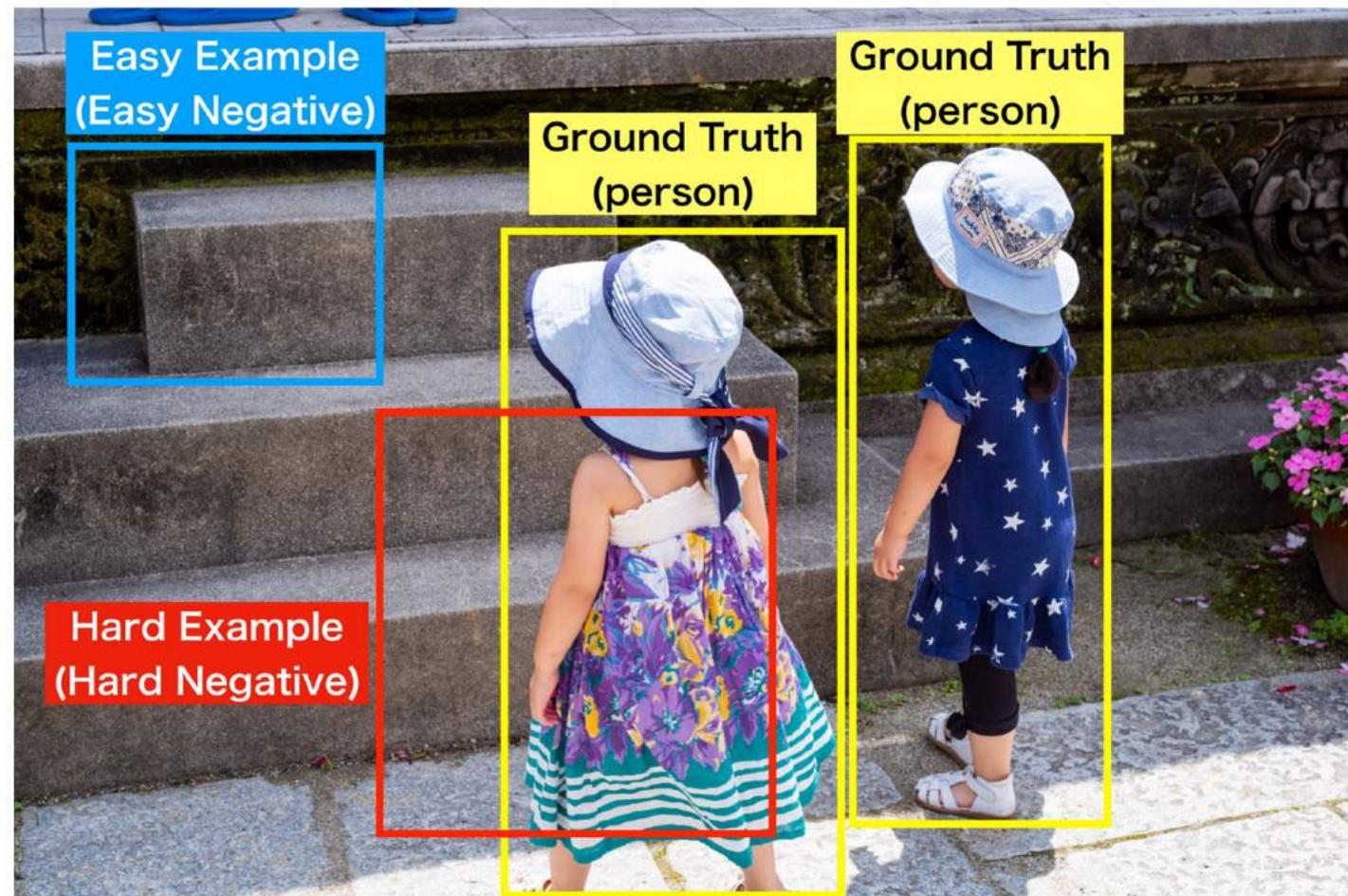
困难样本 = 分类器难以分类正确的样本

= loss 大的样本

负样本 = 真值为背景的样本

困难负样本 = 真值为背景，但分类器分类为前景，且置信度非常高的样本。

可能是真值框周围但与之 IoU 并不高的候选框。



- 为应对样本不均衡问题，SSD 在每次训练迭代中，仅选取分类 loss 最大的一部分负样本参与最终 loss 计算，正负样本比例保持在 1:3。
- 该策略称为困难负样本挖掘 (Hard Negative Mining)，可以让分类器“专注于”最困难的样本。
- 困难负样本挖掘是一个动态的过程，分类器参数更新后，困难负样本也随之变化。

Focal loss 是一种特殊设计的损失函数，可以让正、负、简单、困难样本产生产生相对均衡的损失函数，从而解决样本不均衡问题。

回顾二分类交叉熵 CE Loss:

$$\text{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1 - p) & \text{otherwise.} \end{cases}$$

p 是模型输出的正类概率

为了简化公式，设 p_t 为：

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

此时损失函数可以改写成：

$$\text{CE}(p_t) = \log p_t$$

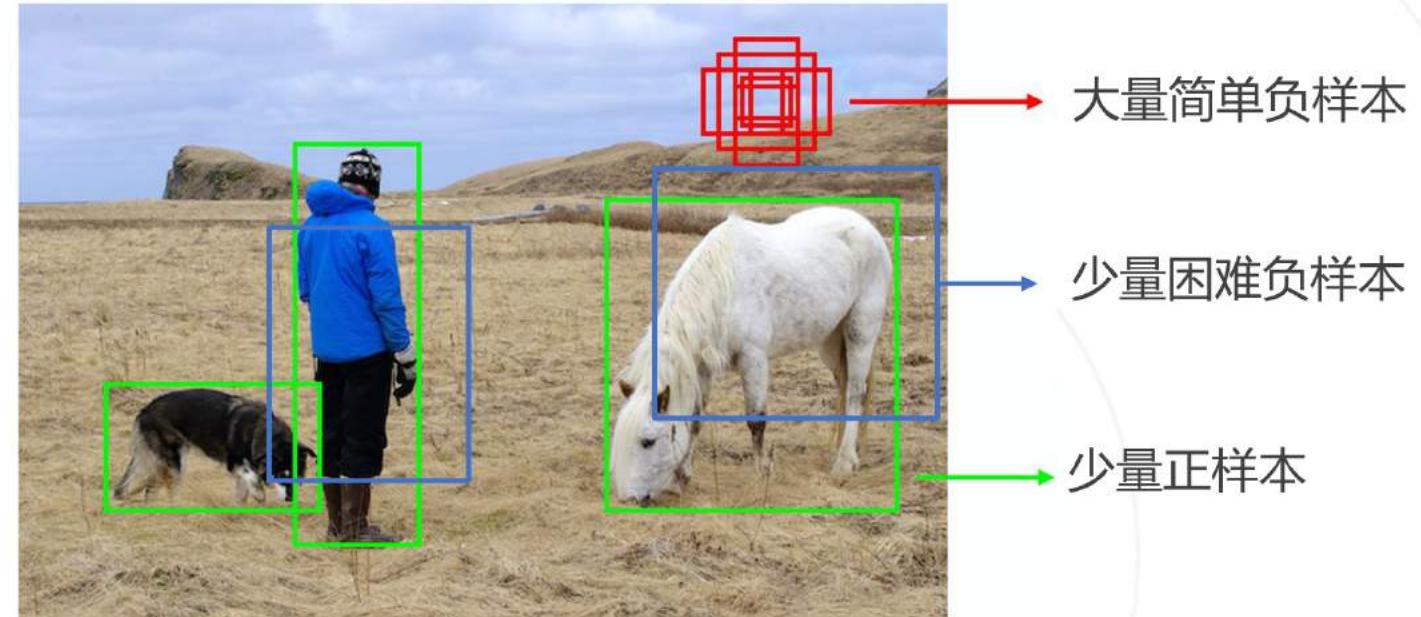
Focal Loss 定义为：

$$\text{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

经验统计：

负样本：正样本 $\approx 1000 : 1$

简单样本：困难样本 $\approx 1000 : 1$



$$L = \sum_{\text{pos box}} L_{\text{pos}} + \sum_{\text{neg box}} L_{\text{neg}}$$

$$= \sum_{\text{pos box}} L_{\text{pos}} + \underbrace{\sum_{\text{hard neg}} L_{\text{neg}}}_{\text{少量正样本和困难负样本产生少量 loss}} + \sum_{\text{easy neg}} L_{\text{neg}}$$

经验统计：

负样本：正样本 $\approx 1000 : 1$

简单样本：困难样本 $\approx 1000 : 1$

困难样本 loss 接近 1

简单样本 loss 很小但不是 0

少量正样本和困难
负样本产生少量
loss

大量简单负样本
产生大量 loss

困难样本 loss：简单样本 loss $\approx 1 : 40$

学习重点与损失函数失配

对于正负样本不均衡：在 loss 中增加正样本的权重，减少负样本的权重

$$L = \sum_{\text{pos box}} L_{\text{pos}} + \sum_{\text{neg box}} L_{\text{neg}}$$

引入加权因子 $\alpha \in [0, 1]$ 对正负样本的 loss 进行加权：

定义 $\alpha_t = \begin{cases} \alpha, & \text{针对正样本} \\ 1 - \alpha, & \text{针对负样本} \end{cases}$

重新加权后的 CE Loss，负样本权重降低

$$\text{CE}(p_t) = -\alpha_t \log(p_t)$$

实验结果： α 对模型精度的影响

α	AP	AP ₅₀	AP ₇₅
.10	0.0	0.0	0.0
.25	10.8	16.0	11.7
.50	30.2	46.7	32.8
.75	31.1	49.4	33.0
.90	30.8	49.7	32.3
.99	28.7	47.4	29.9
.999	25.1	41.7	26.1

$$L = \sum_{\text{pos box}} L_{\text{pos}} + \sum_{\text{hard neg}} L_{\text{neg}} + \sum_{\text{easy neg}} L_{\text{neg}}$$

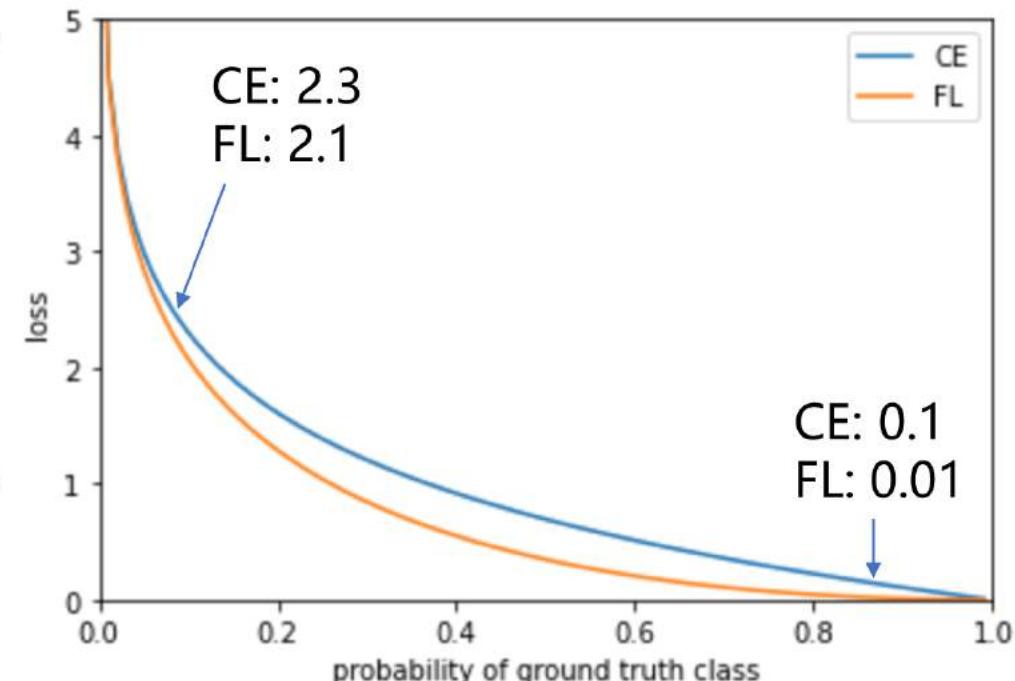
简单样本的 loss 并不为 0
需要设法降低简单样本产生的 loss

Focal Loss 的方案：在交叉熵的基础上增加调节因子 $(1 - p_t)^\gamma$ ，其中 γ 是可调节的超参：

$$\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

工作原理：

- 困难样本对应的 p_t 接近 0，调节因子对 loss 几乎不产生影响
- 简单样本对应的 p_t 大多落在 $[0.6, 1.0]$ 区间，调节因子大幅降低 loss
- 二者共同作用，简单样本的 loss 比重降低



综合加权因子和调节因子得到完整的 Focal Loss:

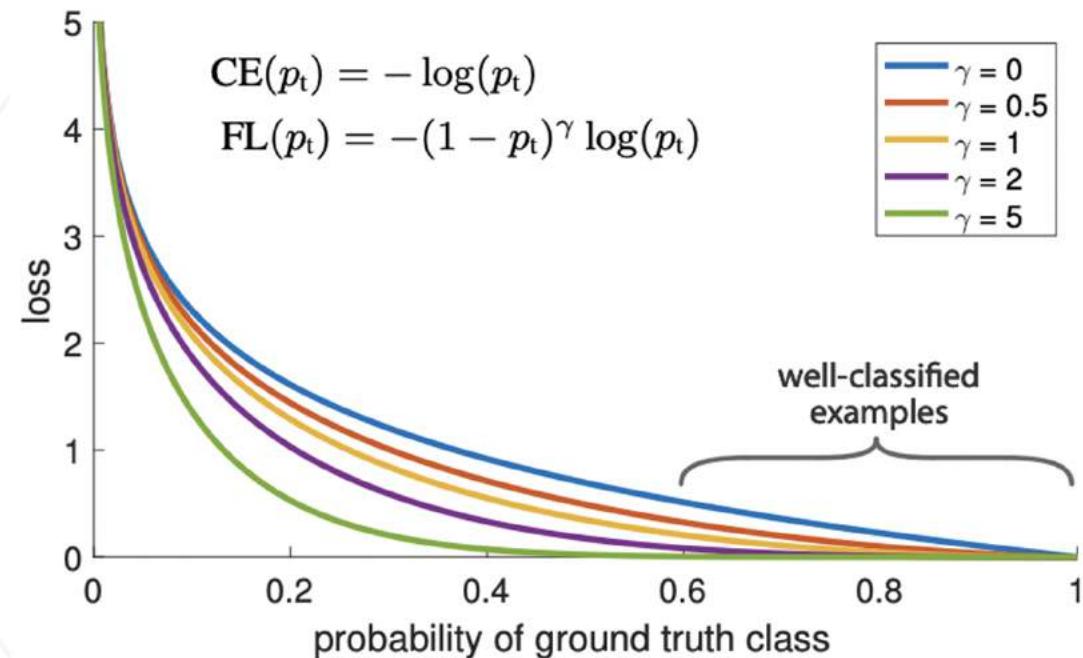
$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

实验表明 $\gamma = 2, \alpha = 0.25$ 效果最好

γ	α	AP	AP ₅₀	AP ₇₅
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	34.0	52.5	36.5
5.0	.25	32.2	49.6	34.8

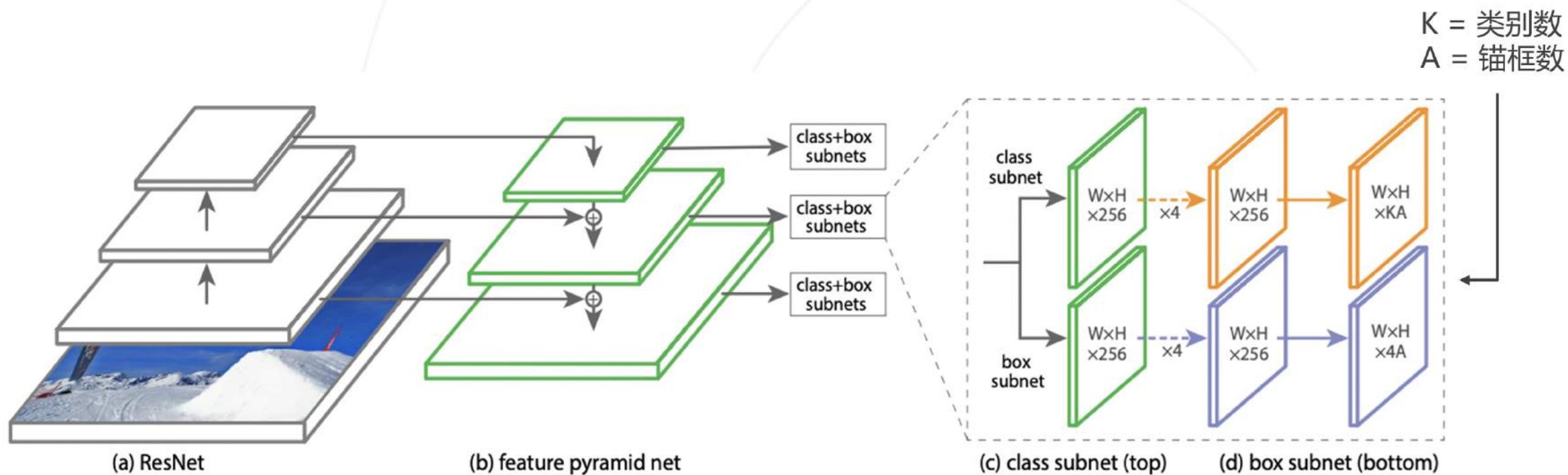
由于两个超参是相互作用的，所以 γ 增大，通常要对应减少 α

α 用于调节正负样本的权重
 γ 用于调节简单样本权重降低的速率



基于 Focal Loss，论文作者设计了单阶段检测模型 RetinaNet，包含：

- ResNet 主干网络与 FPN 特征金字塔，产生多尺度特征图
- 在每级特征图上设置 Anchor
- 两分支、4 层卷积构成的检测头，针对每个锚框产生 K 类类别预测 以及 4 个边界框偏移量



无锚框目标检测算法

Anchor-free Methods

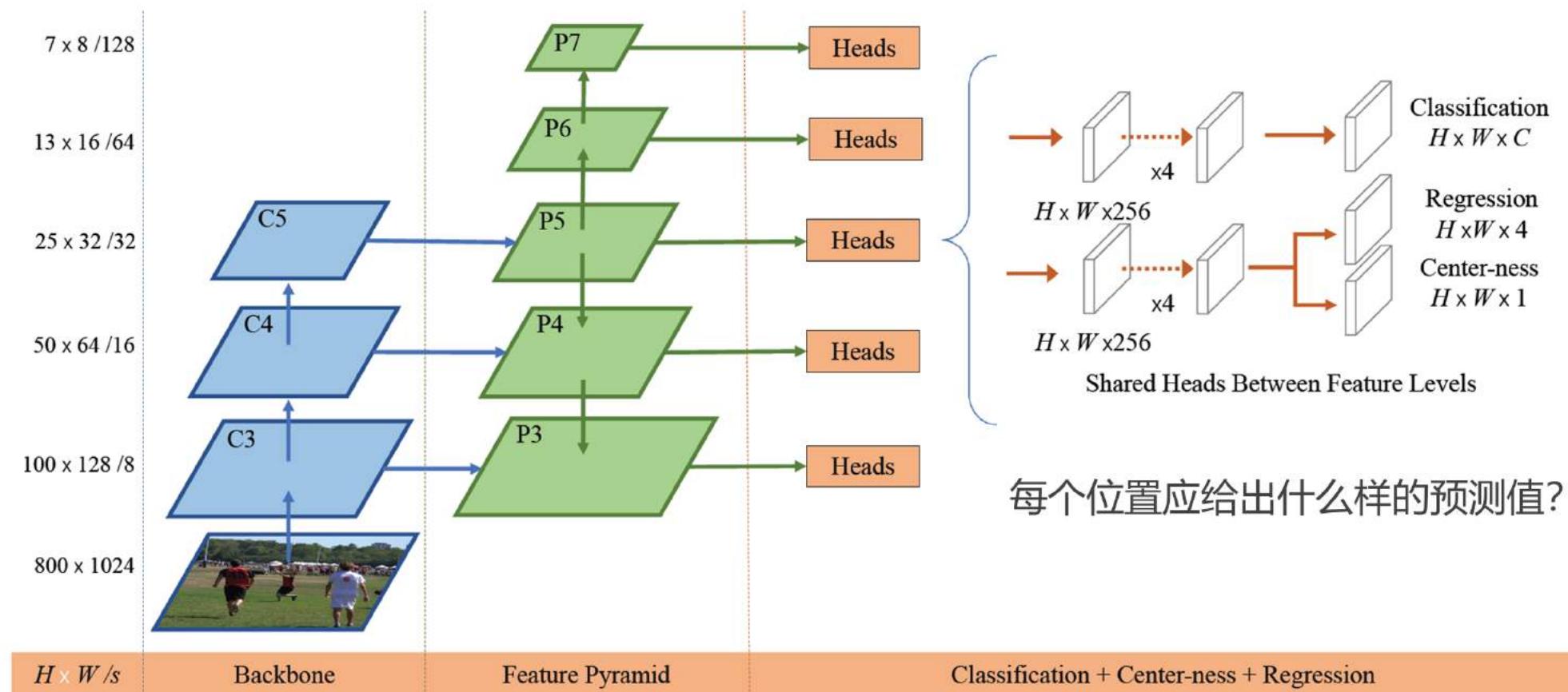
锚框

- Faster RCNN、YOLO v2 / v3、RetinaNet 都是基于锚框的检测算法
- 锚框引入了大量超参数（如大小、长宽比、数量、IoU 阈值等），需要人工调整
- 锚框涉及大量复杂计算，如锚框与真值框之间的 IoU 以及锚框与真值框的匹配等

无锚框

- 无锚框算法尝试直接基于特征学习边界框的位置，不依赖锚框，从而减降低模型复杂度
- YOLO v1 是无锚框算法，但由于提出时间较早，相关技术并不完善，性能不如基于锚框的算法

- 主干网络与特征金字塔产生多尺度特征图
- 检测头在多级特征图上完成预测，对于每个位置，预测**类别、边界框位置和中心度**三组数值



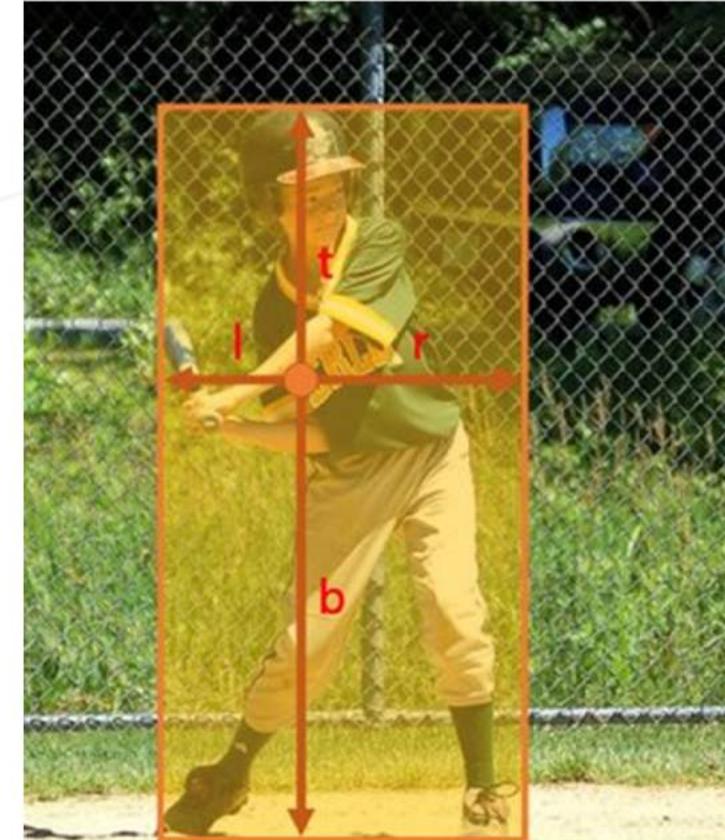
基本规则

- 如果特征图上某个位置（在原图上对应的位置）位于某个物体的**边界框内部**，则该位置的预测值应给出该物体的：
 - 类别
 - 边界框相对于该**位置**的偏移量
 - 中心度，用于衡量预测框的优劣
- 如果某个位置不在任何物体的边界框内部，分类为背景。

对比：锚框算法基于 IoU 判定正负样本，FCOS 有更多正样本

对比：锚框算法给出基于锚框的偏移量

类似 YOLO 的 score



回顾

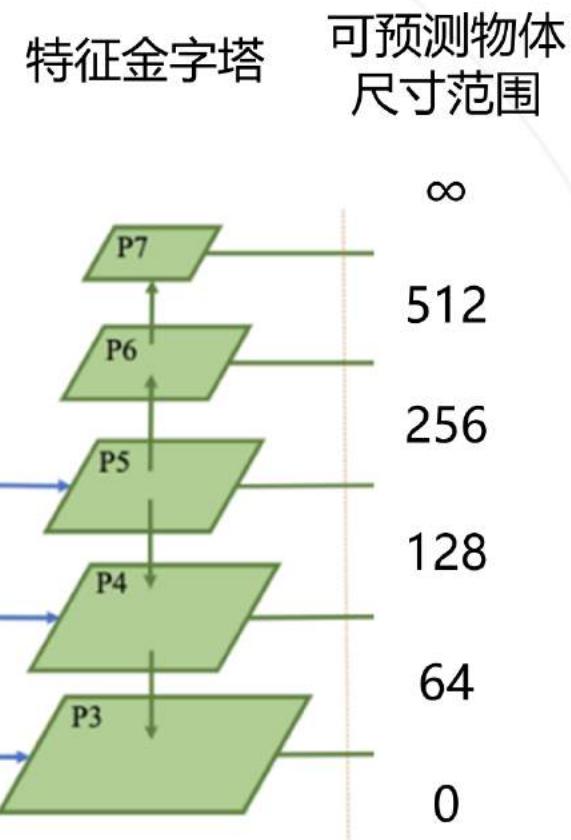
- YOLO 直接回归边界框，但不能处理重叠物体
- 基于锚框的算法可以基于不同的锚框，产生不同目标物体的预测

FCOS 如何处理重叠问题？

重叠的物体尺度通常不同，由不同尺度的特征图给出对应目标物体的预测：

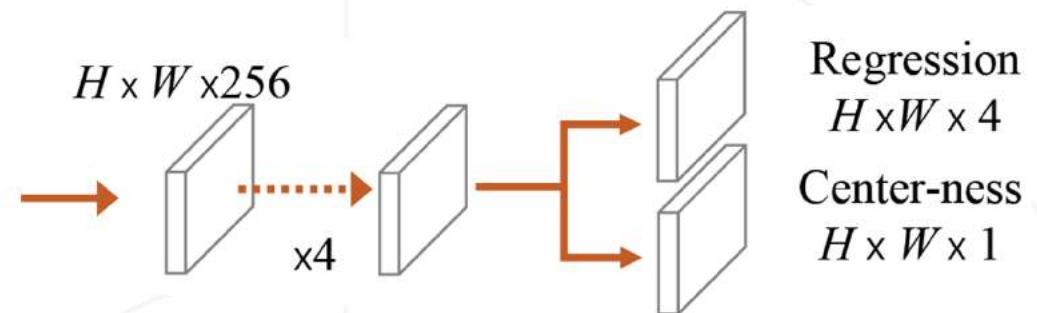
- 小物体的预测基于低层次特征图给出
- 大物体的预测基于高层次特征图给出

FCOS 的多尺度匹配策略可以很大程度解决重叠问题。如果同层特征图还有重叠，预测重叠物体中较小的那个



FCOS 将所有框内的位置归类为正样本：

- 有更多正样本用于训练
- 但也会让分类器更容易在物体周围产生低质量预测框



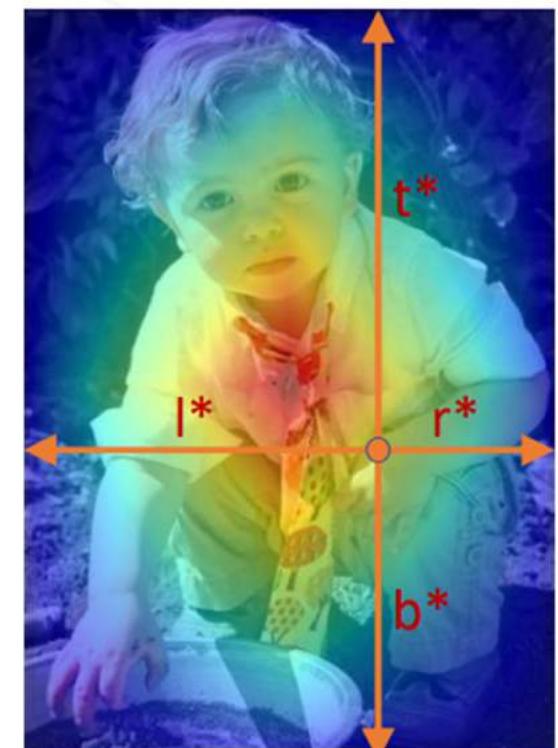
解决方案：

- 每个位置额外预测一个**中心度值**，用于该预测框的优劣
- 推理阶段：置信度 = $\sqrt{\text{中心度} \times \text{分类概率}}$

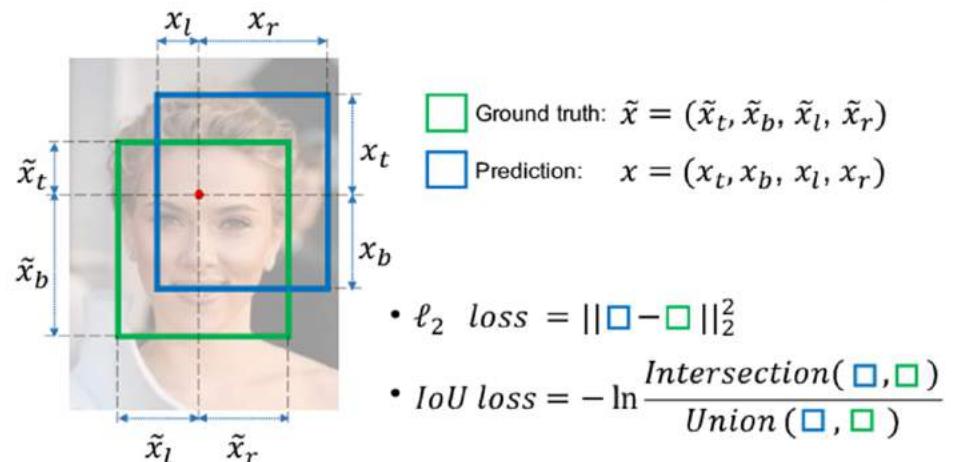
中心度定义：

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$

几何意义：位置越靠近边界框中心越接近 1，越靠近边界越接近 0



- 每个位置的分类损失 $L_{cls}(\mathbf{p}_{x,y}, c_{x,y}^*)$ ，使用 Focal loss
- 对于前景样本的边界框回归损失 $L_{reg}(\mathbf{t}_{x,y}, \mathbf{t}_{x,y}^*)$ ，使用 IoU loss



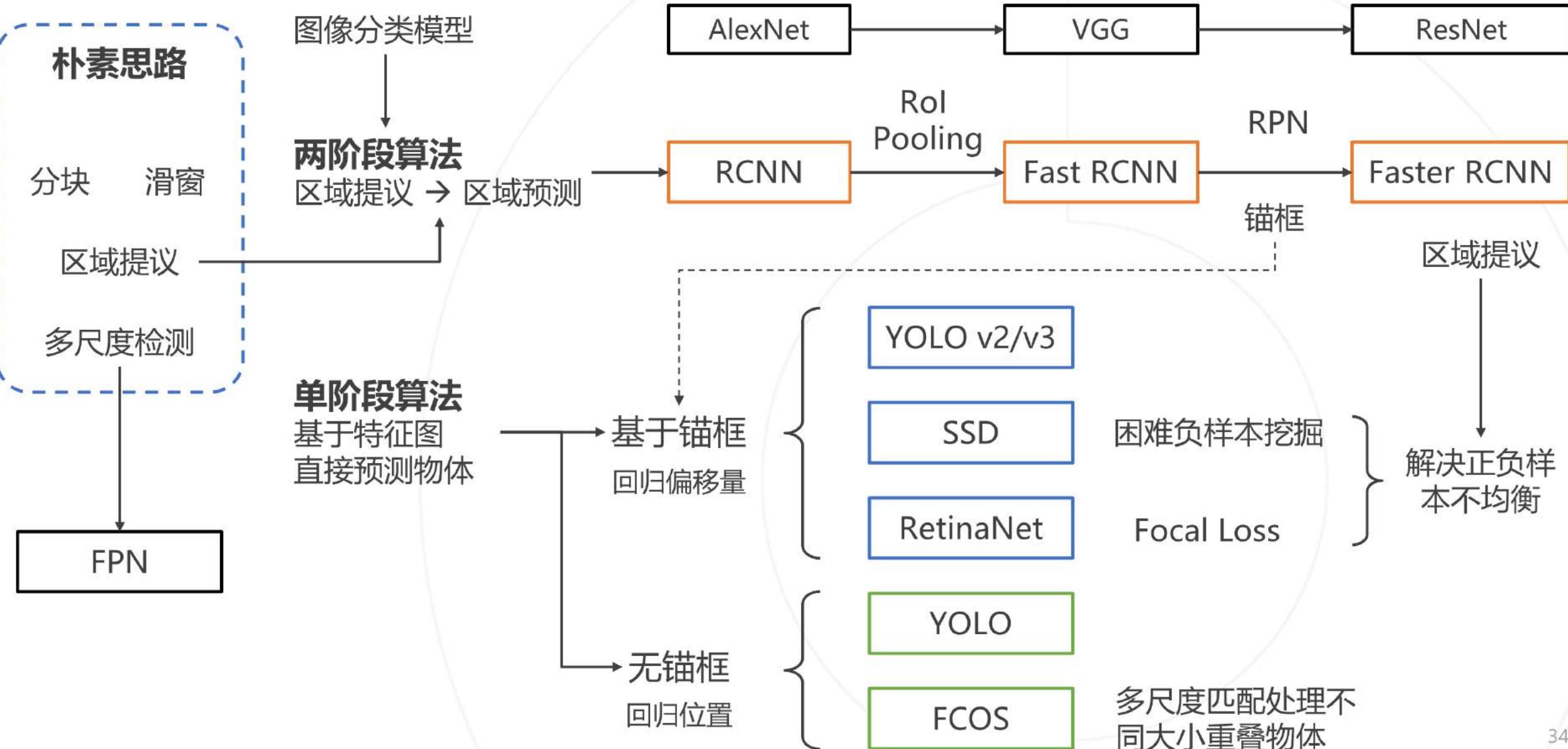
- 对于前景样本的 Center-ness 值，因为介于 0-1 之间，使用 BCE loss
- 三者求和共同优化

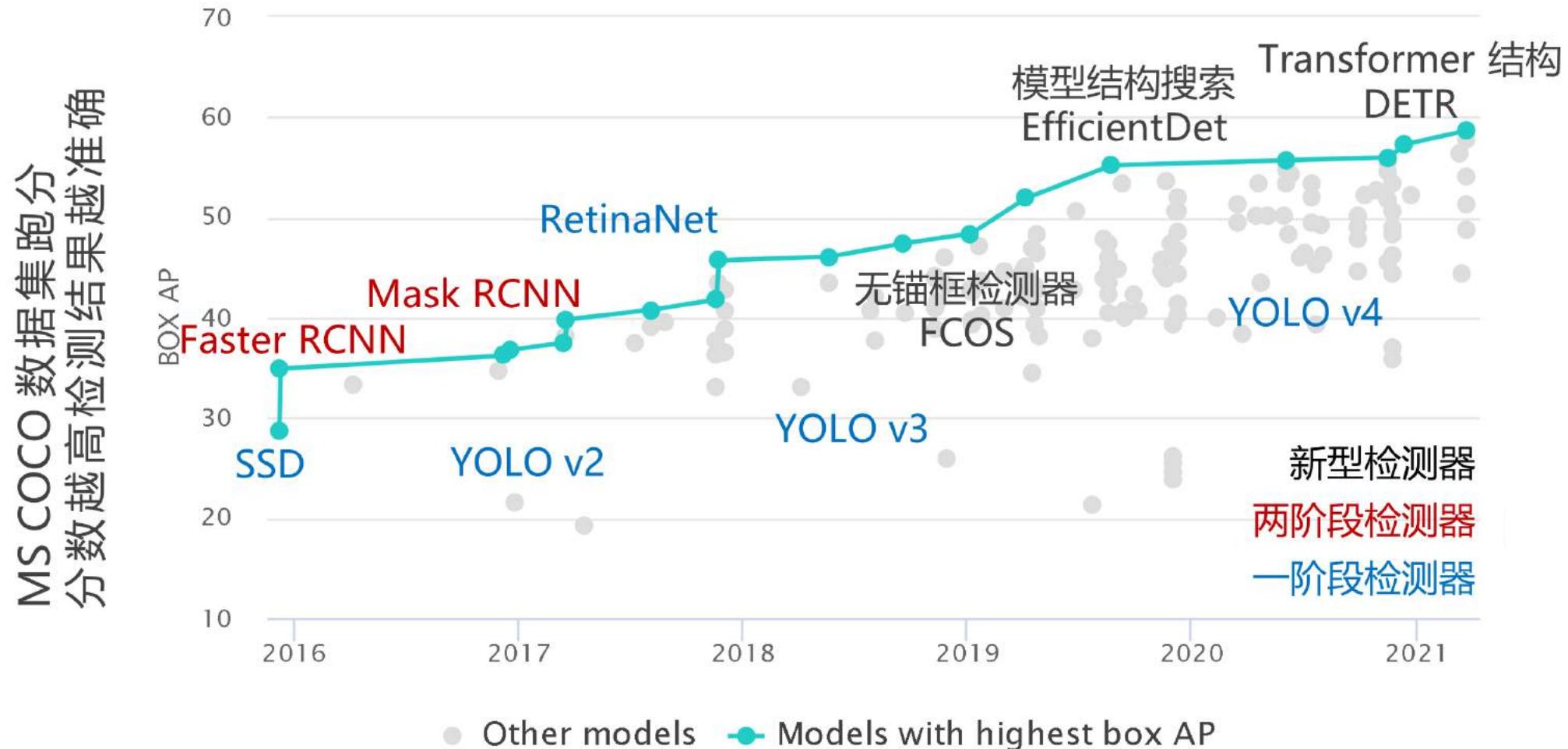
x, y	位置	$\mathbf{p}_{x,y}$	类别概率预测
		$c_{x,y}^*$	类别真值

$\mathbf{t}_{x,y} = (l, t, r, b)$	边界框预测值
$\mathbf{t}_{x,y}^* = (l^*, t^*, r^*, b^*)$	边界框真值

完全重合，IoU loss = 0
 完全不重合，IoU loss = ∞

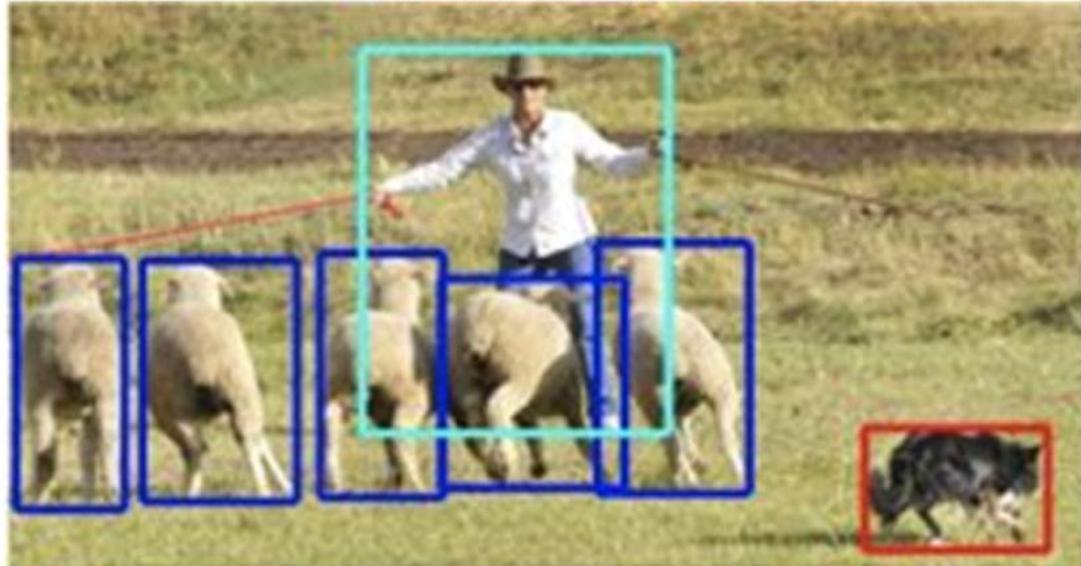
与L2相比，大小物体产生同等大小的loss





实例分割

Instance Segmentation



目标检测

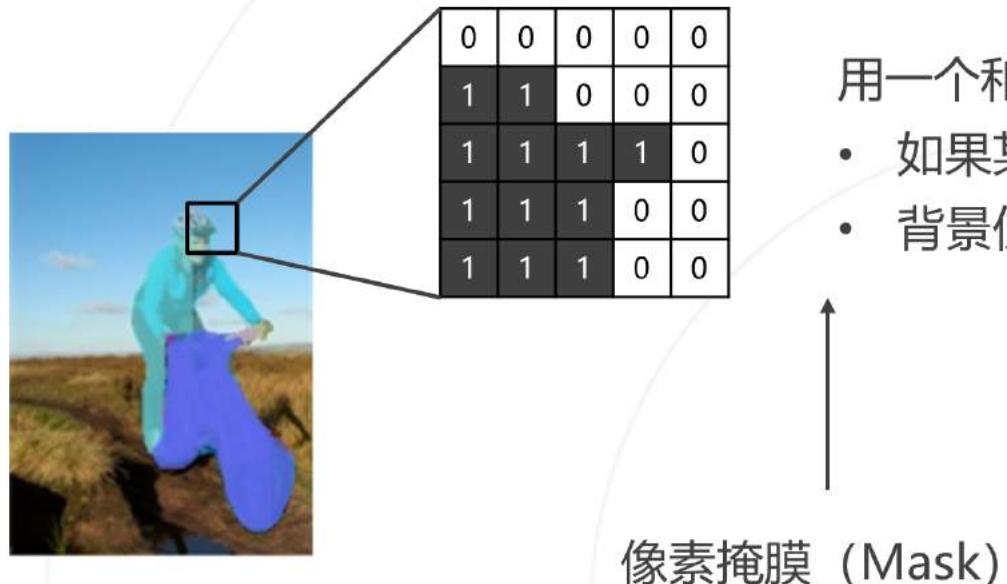
定位物体的边界
给出边界的坐标



实例分割

判定物体占据的像素
生成像素掩膜

如何表示物体占据的像素？

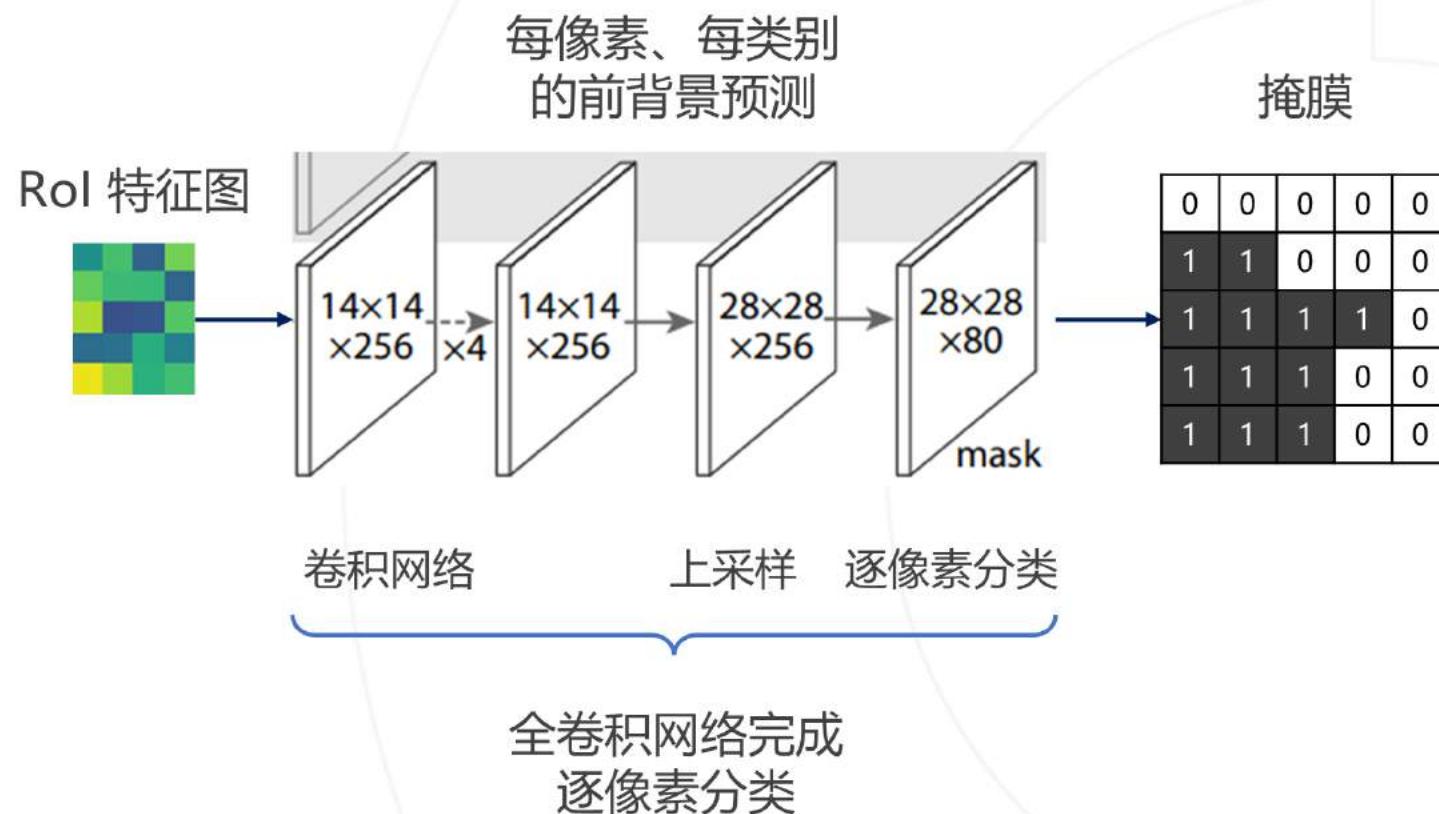


用一个和图像等大的**二进制**图像

- 如果某个像素被物体占据，对应值为1，
- 背景位置值为0

实例分割 → 对于图像中的每个物体产生像素掩膜
产生像素掩膜 → 对每个像素点进行二分类

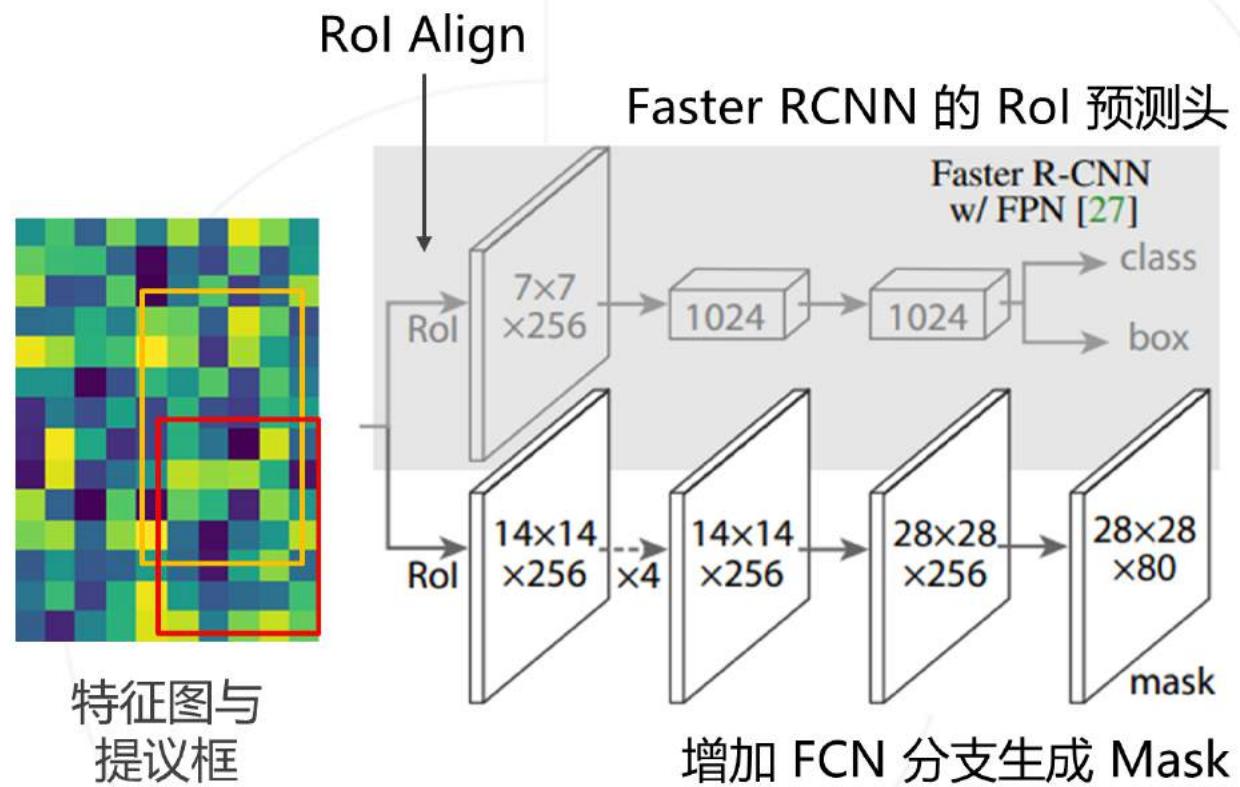
全卷积网络 (Fully Convolutional Network) 是面向语义分割任务提出的模型
可以在实例分割模型中产生掩膜



Mask RCNN = Faster RCNN + FCN 分支

1. ROI Align 提取 14×14 的 ROI 特征图，
比 ROI Pooling 更精细
2. FCN 产生该区域的掩膜
3. 根据提议框将掩膜 Warp 回原图尺寸

训练时在 mask 分支使用逐像素的二类交叉熵

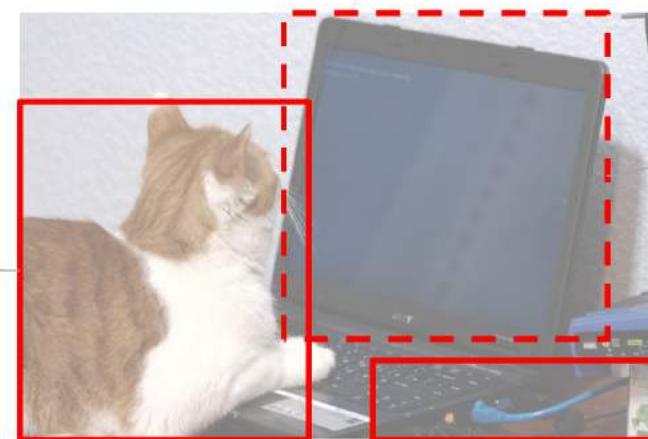


目标检测模型的评估方法

Evaluation

- 正确结果 (True Positive): 算法检测到了某类物体 (Positive), 图中也确实有这个物体, 检测结果正确 (True)
 - 假阳性 (False Positive): 算法检测到了某类物体 (Positive), 但图中其实没有这个物体, 检测结果错误 (False)
 - 假阴性 (False Negative): 算法没有检测到物体 (Negative), 但图中其实有某类物体, 检测结果错误 (False)
- **检测到的衡量标准:** 对于某个检测框, 图中存在同类型的真值框且与之交并比大于阈值 (通常取0.5)

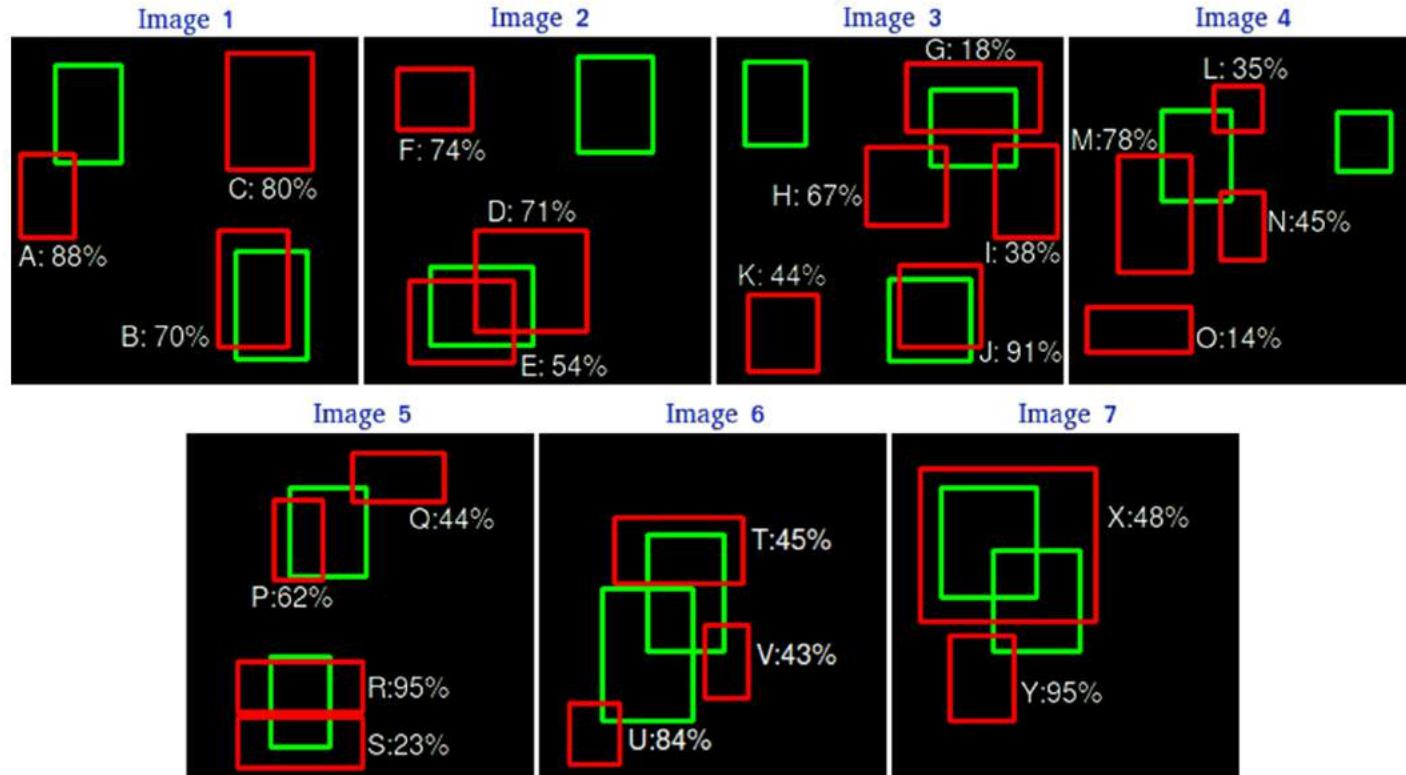
正确结果: 图中这个位置有一只猫,且检测算法检测到了



假阴性: 图中这个位置有一台笔记本电脑, 但检测器没有检测到。

假阳性: 图中这个位置没有物体,但检测算法输出了一个检测框。

例子



检测框重叠：置信度最高的TP，其余FP

真值框重叠：一个TP，其余FN

Images	Detections	Confidences	TP or FP
Image 1	A	88%	FP
Image 1	B	70%	TP
Image 1	C	80%	FP
Image 2	D	71%	FP
Image 2	E	54%	TP
Image 2	F	74%	FP
Image 3	G	18%	TP
Image 3	H	67%	FP
Image 3	I	38%	FP
Image 3	J	91%	TP
Image 3	K	44%	FP
Image 4	L	35%	FP
Image 4	M	78%	FP
Image 4	N	45%	FP
Image 4	O	14%	FP

$$\text{召回率 (recall)} = \frac{\text{正确结果总数}}{\text{真值框总数}} = \frac{\#TP}{\#TP + \#FN}$$

$$\text{准确率 (precision)} = \frac{\text{正确结果总数}}{\text{检测框总数}} = \frac{\#TP}{\#TP + \#FP}$$

真值框总数与检测算法无关，因此只需将检测结果区分为 TP 和 FP 即可计算 recall 和 precision

两种极端情况：

1. 检测器将所有锚框都判断为物体：召回率 $\approx 100\%$ ，但大量背景框预测为物体，FP很高，准确率很低；
2. 检测器只输出置信度最高的1个检测框：以很大概率检测正确，准确率=100%，但因为大量物体被预测为背景，FN很高，召回率很低。

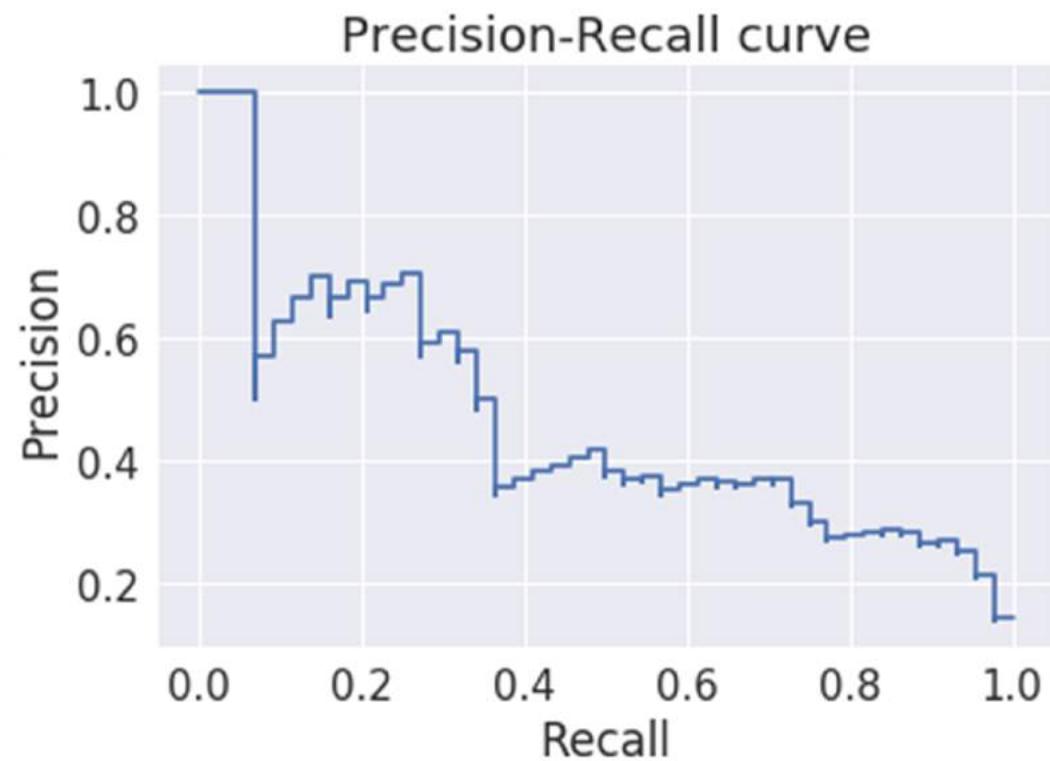
一个完美的检测器应该有100%召回率和100%的精度；在算法能力有限的情况下，应该平衡二者。

通常做法：将检测框按置信度排序，仅输出置信度最高的若干个框（置信度通常取类别概率的最大值）

为得到阈值无关的评分，可以遍历阈值，并对 Precision 和 Recall 求平均

具体做法：

- 检测框按置信度排序，取前 K 个框计算 Precision 和 Recall。
- 遍历 K 从 1 至全部检测框，将得到的 Precision 和 Recall 值绘制在坐标系上，得到 PR 曲线。
- 定义 $\text{Average Precision} = \text{Precision}$ 对 Recall 的平均值，即 PR 曲线下的面积，作为检测器的性能衡量指标。



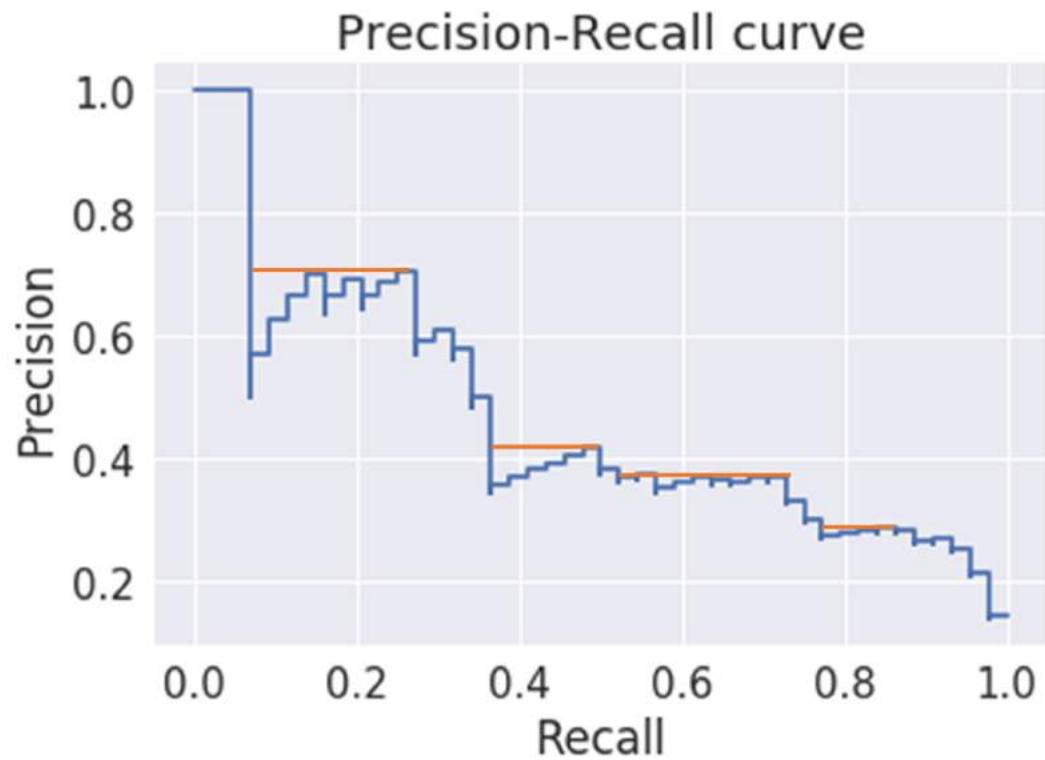
$$\text{召回率 (recall)} = \frac{\#TP}{\text{真值框总数}}$$

真值框总数与检测算法无关，TP 个数随检测框个数增多而增多或不变，因此 recall 随检测框个数增多而增加或不变。

$$\text{准确率 (precision)} = \frac{\#TP}{\#TP + \#FP}$$

每增加一个检测框，如果检测框是正确的，准确率上升，否则下降，因此 precision 会有上下波动。

为避免起伏的影响，在计算面积前对PR曲线进行插值。

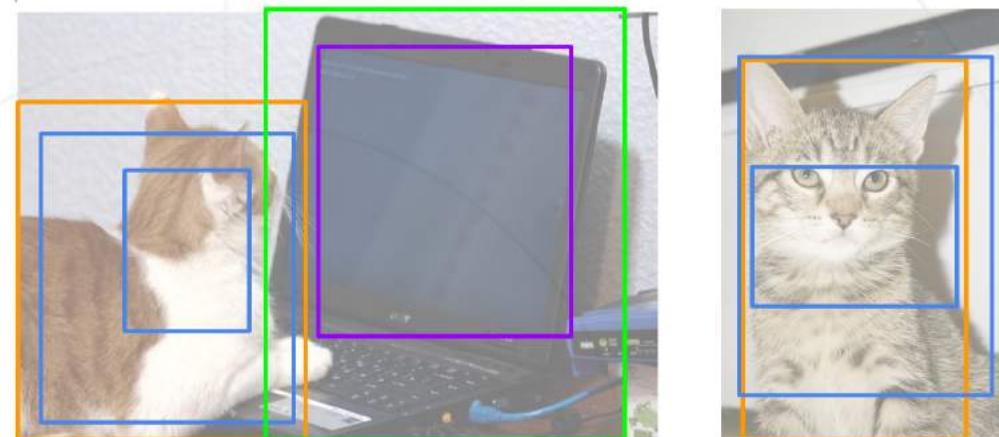


分类别统计AP，并按类别平均即得到 Mean AP。

Mean AP 的完整计算流程：

- 将数据集中全部图像上的检测框按预测类别分类。
- 对于某一类别的所有检测框，计算 AP：
 - 按置信度将该类别的所有检测框排序
 - 逐一与真值框比较，判定 TP 或 FP，并绘制 PR 曲线
 - 对 PR 曲线插值，计算 AP
- 求所有类别的 AP 的平均，得到 Mean AP

测试集包含两张图像



猫类别：2个真值框 + 4个预测框
电脑类别：1个真值框 + 1个预测框

部分数据集（如 COCO）还要求在不同的 IoU 阈值下计算 Mean AP 并平均，作为最终评分。
可衡量检测器在不同定位精度要求下的性能。

谢谢大家

更多内容可参见 GitHub 上的代码文档以及教程；
如有问题欢迎加入我们的微信群进行提问。

