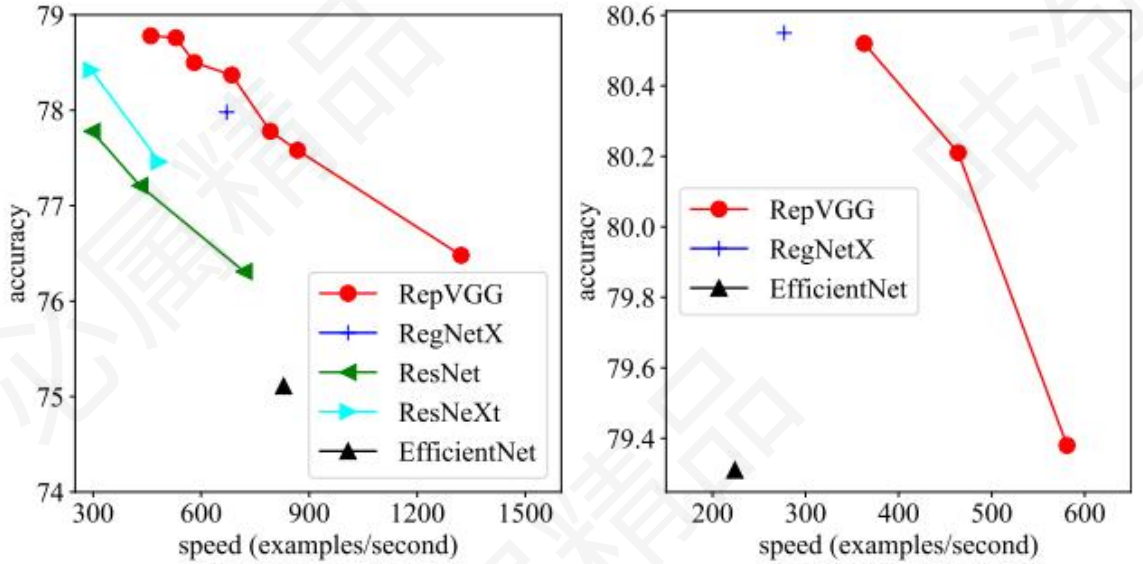
YOLOV7

- RepVGG：推理加速
  - 在backbone搞事情那得是大事了
  - 重参数化的作用就是加速省显存



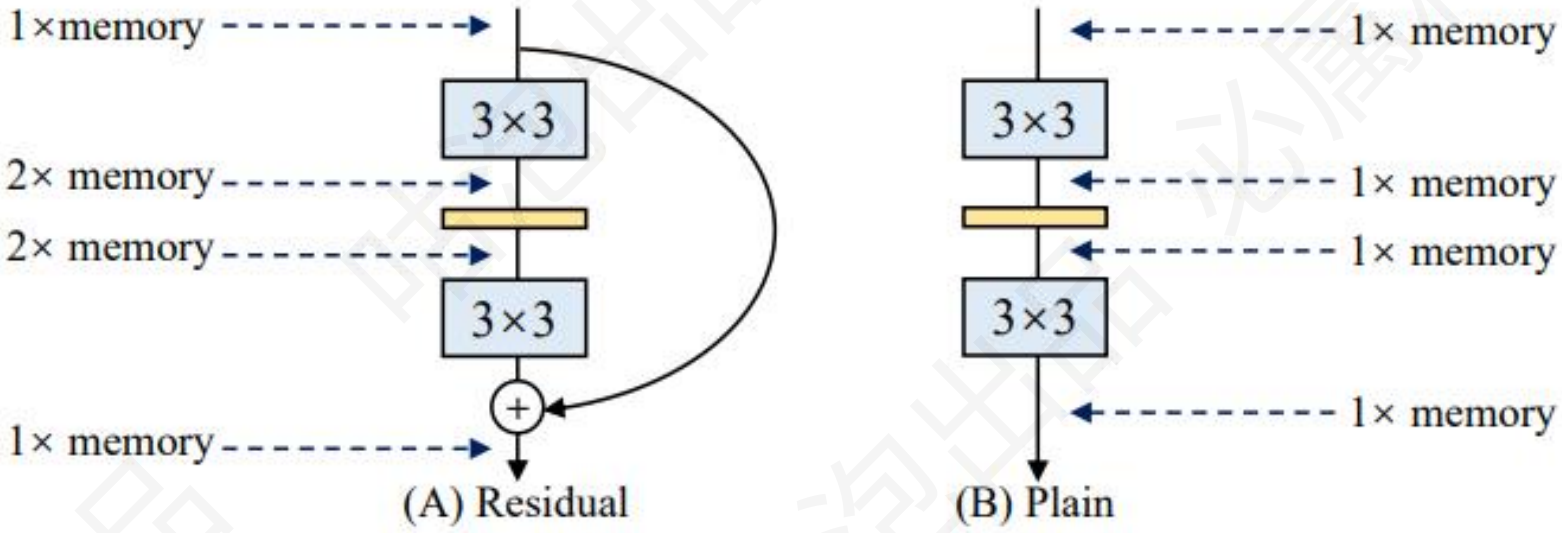
  - VGG拥护啥被比下去来着，是不是resnet出现了，它的核心就是多分支
  - 虽然说效果可以，但是resnet这种多分支结构会不会在速度和显存上有劣势呢

YOLOV7

- RepVGG：推理加速

  - 来算账了，VGG就一条路，走自己的路谁都不用管，就一倍显存

  - 只要带有分支了，首先就得互相等，然后一会要相加，中间那就得翻倍

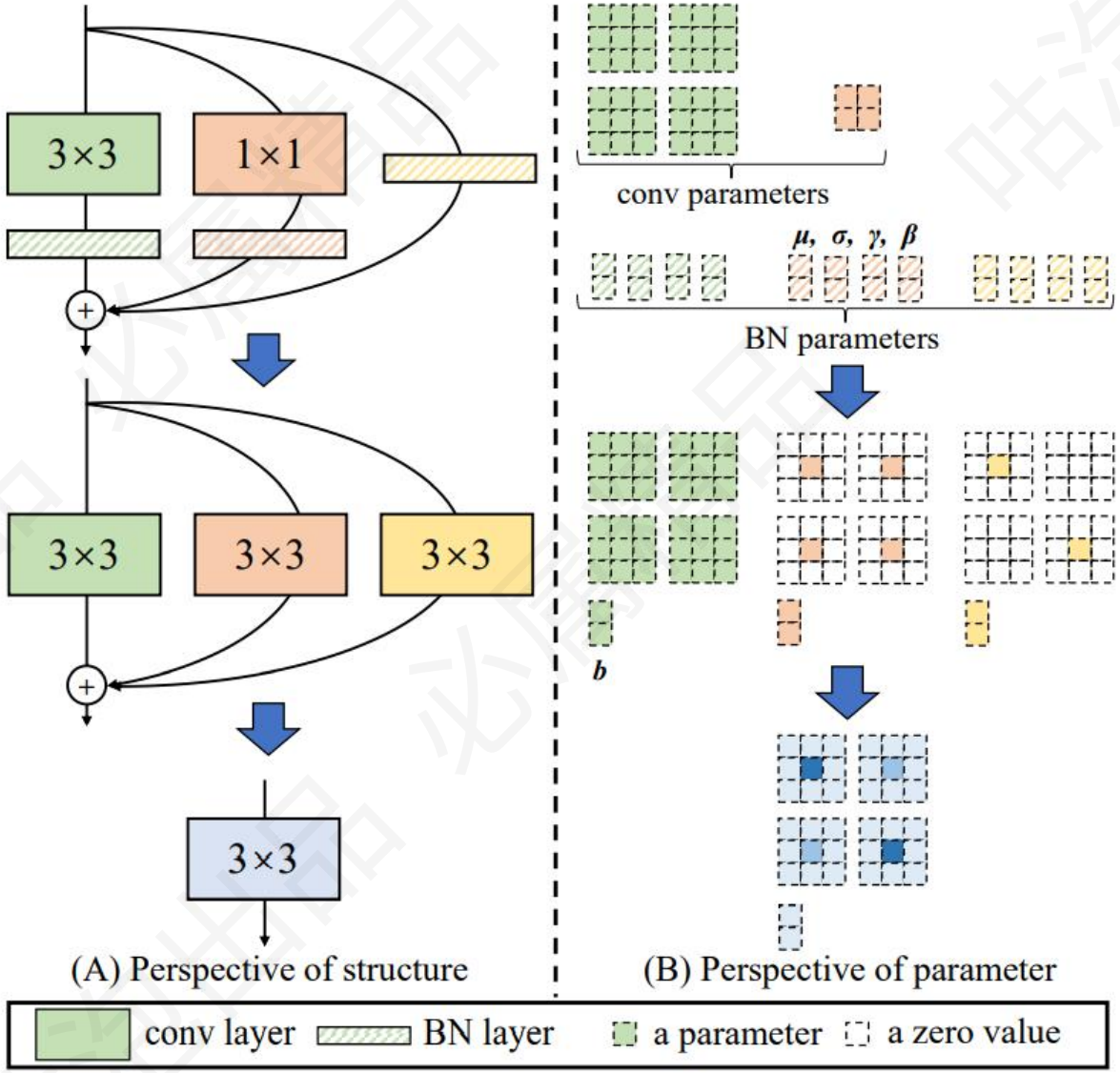


YOLOV7

- 当年的版本答案VGG的故事
  - VGG在2014年就告诉我们一件事，3*3的卷积是最好的，其他的谁也不好使
  - 应该与英伟达优化相关，并不是纯卷积核大小的问题，所以现在都3*3
  - 所以我们就得想想了能不能把多分支的，多种不同卷积核的，以及那些带BN的
  - 全部都转换成3*3的卷积然后叠加在一起呢，核心就是万物都是3*3，大一统了

YOLOV7

- 要完成的任务
  - 输入输出都是2个特征图
  - 1.卷积和BN的合并
  - 2.全部转成3*3的卷积
  - 3.多个卷积核再合并



YOLOV7

RepConv-回顾BN

- 一个batch数据，$x_1, x_2, \ldots, x_n$ 对他们来进行归一化操作：

$$\hat{x}_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \; \hat{x}_i = \frac{\gamma x_i}{\sqrt{\sigma^2 + \epsilon}} + \beta - \frac{\gamma \mu}{\sqrt{\sigma^2 + \epsilon}}$$

- 注意均值和方差都是每个channel单独计算，β是平移系数，γ是缩放系数
- 其中均值和方差都是可以求出来的，β和gamma是学出来的，ε是防止除0

YOLOV7

- RepConv-BN计算拆解

- 尽可能得往卷积的形式来靠，因为一会咱们要给他和卷积合并

$$\begin{pmatrix} \hat{F}_{1,i,j} \\ \hat{F}_{2,i,j} \\ \vdots \\ \hat{F}_{C-1,i,j} \\ \hat{F}_{C,i,j} \end{pmatrix} = \begin{pmatrix} \frac{\gamma_1}{\sqrt{\hat{\sigma}_1^2+\epsilon}} & 0 & \cdots & 0 \\ 0 & \frac{\gamma_2}{\sqrt{\hat{\sigma}_2^2+\epsilon}} & & \\ \vdots & & \ddots & \vdots \\ & & \frac{\gamma_{C-1}}{\sqrt{\hat{\sigma}_{C-1}^2+\epsilon}} & 0 \\ 0 & & 0 & \frac{\gamma_C}{\sqrt{\hat{\sigma}_C^2+\epsilon}} \end{pmatrix} \cdot \begin{pmatrix} F_{1,i,j} \\ F_{2,i,j} \\ \vdots \\ F_{C-1,i,j} \\ F_{C,i,j} \end{pmatrix} + \begin{pmatrix} \beta_1 - \gamma_1 \frac{\hat{\mu}_1}{\sqrt{\hat{\sigma}_1^2+\epsilon}} \\ \beta_2 - \gamma_2 \frac{\hat{\mu}_2}{\sqrt{\hat{\sigma}_2^2+\epsilon}} \\ \vdots \\ \beta_{C-1} - \gamma_{C-1} \frac{\hat{\mu}_{C-1}}{\sqrt{\hat{\sigma}_{C-1}^2+\epsilon}} \\ \beta_C - \gamma_C \frac{\hat{\mu}_C}{\sqrt{\hat{\sigma}_C^2+\epsilon}} \end{pmatrix}$$

- 特征图F，归一化后的结果也就相当于一个1*1*C的卷积（wx+b）

YOLOV7

- RepConv-BN计算拆解，更新后的卷积核：
  - filter weights: $\mathbf{W} = \mathbf{W}_{BN} \cdot \mathbf{W}_{conv}$
  - bias: $\mathbf{b} = \mathbf{W}_{BN} \cdot \mathbf{b}_{conv} + \mathbf{b}_{BN}$

- 合并方法（用一个卷积代替原来的卷积+BN）

Let, $\mathbf{W}_{BN} \in \mathbb{R}^{C \times C}$ and $\mathbf{b}_{BN} \in \mathbb{R}^{C}$ - are parameters of the BN

$\mathbf{W}_{conv} \in \mathbb{R}^{C \times (C_{prev} k^2)}$ and $\mathbf{b}_{conv} \in \mathbb{R}^{C}$ - are parameters of the Convolutional layer that precede BN

F_{prev} - input to the convolutional

C_{prev} - the number of channels of the input layer

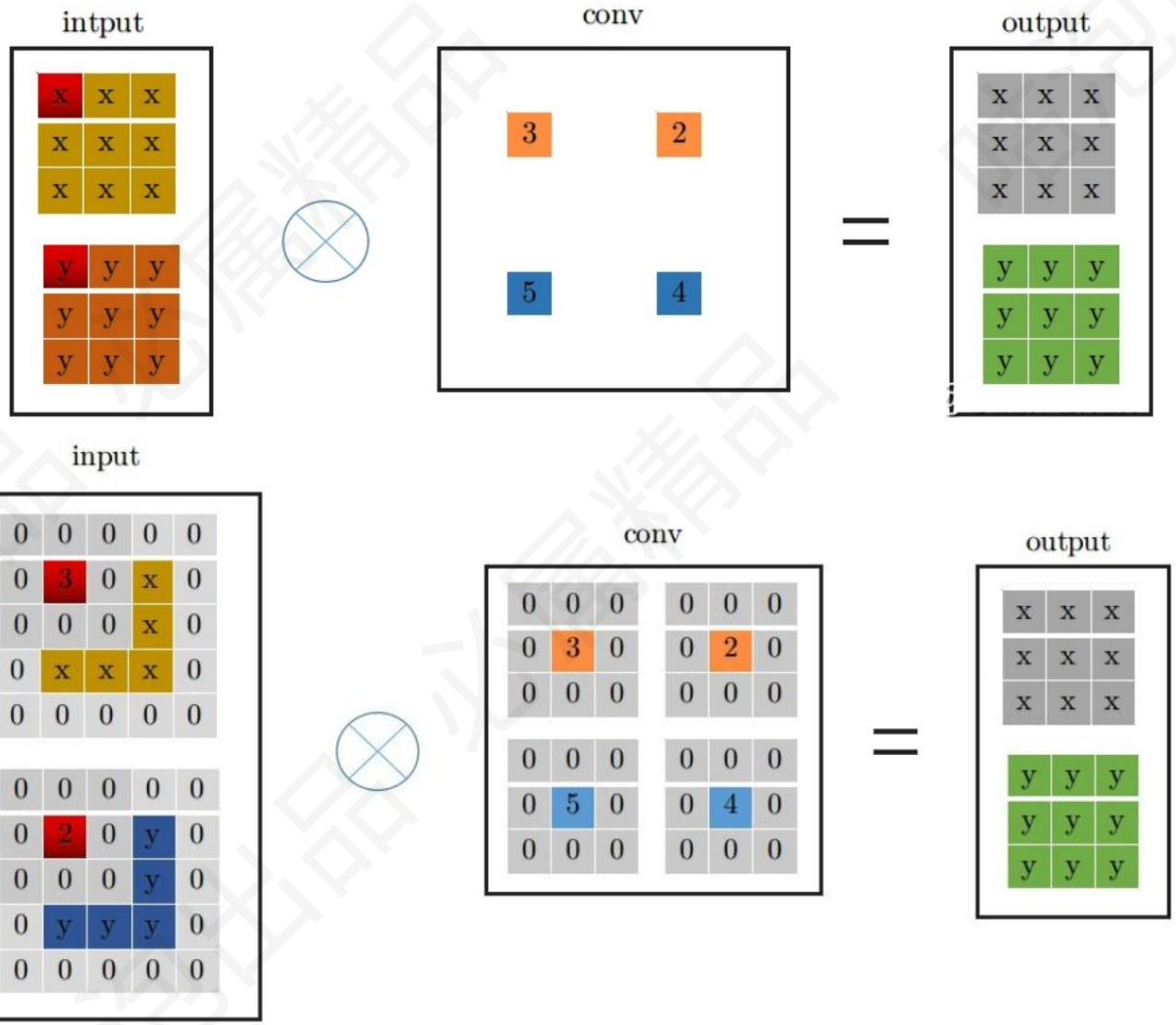
k - is the filter size.

$k \times k$ part of F_{prev} reshaped into a $k^2 \cdot C_{prev}$ vector $\mathbf{f}_{i,j}$, so the resulting formula will be:

$$\hat{\mathbf{f}}_{i,j} = \mathbf{W}_{BN} \cdot (\mathbf{W}_{conv} \cdot \mathbf{f}_{i,j} + \mathbf{b}_{conv}) + \mathbf{b}_{BN}$$

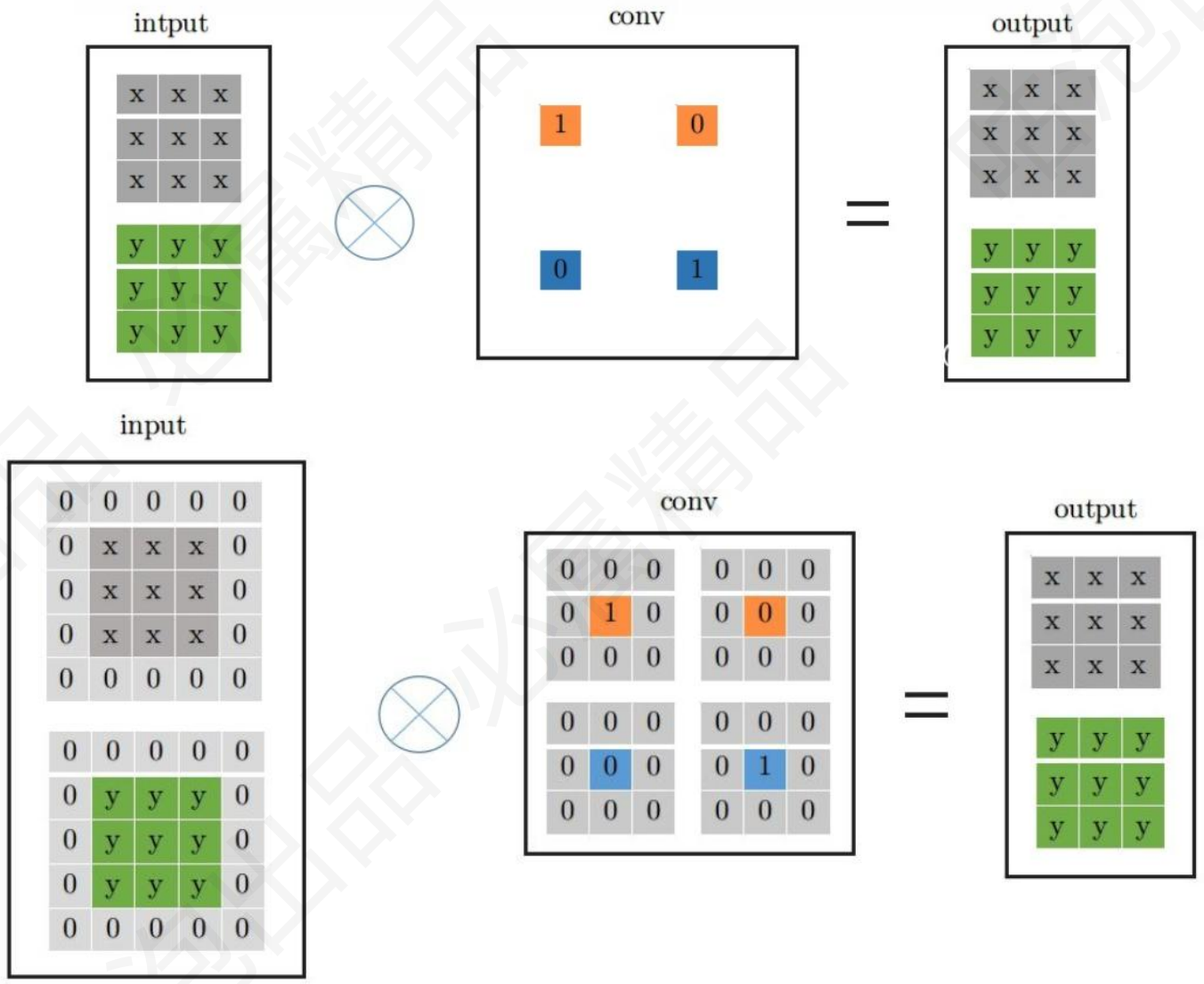
YOLOV7

- 1*1变换到3*3
- 其实计算方式没变
- 只不过需要pad卷积核
- 但是要注意原始输入也要pad
- 这样1*1的就可以用3*3替代



YOLOV7

- resnet残差链接转换
  - 不就直接拿过来就得了嘛
  - 跟我念：——得—
  - 但是要拼成卷积核的形式
  - 全剧终，都变成3*3的了

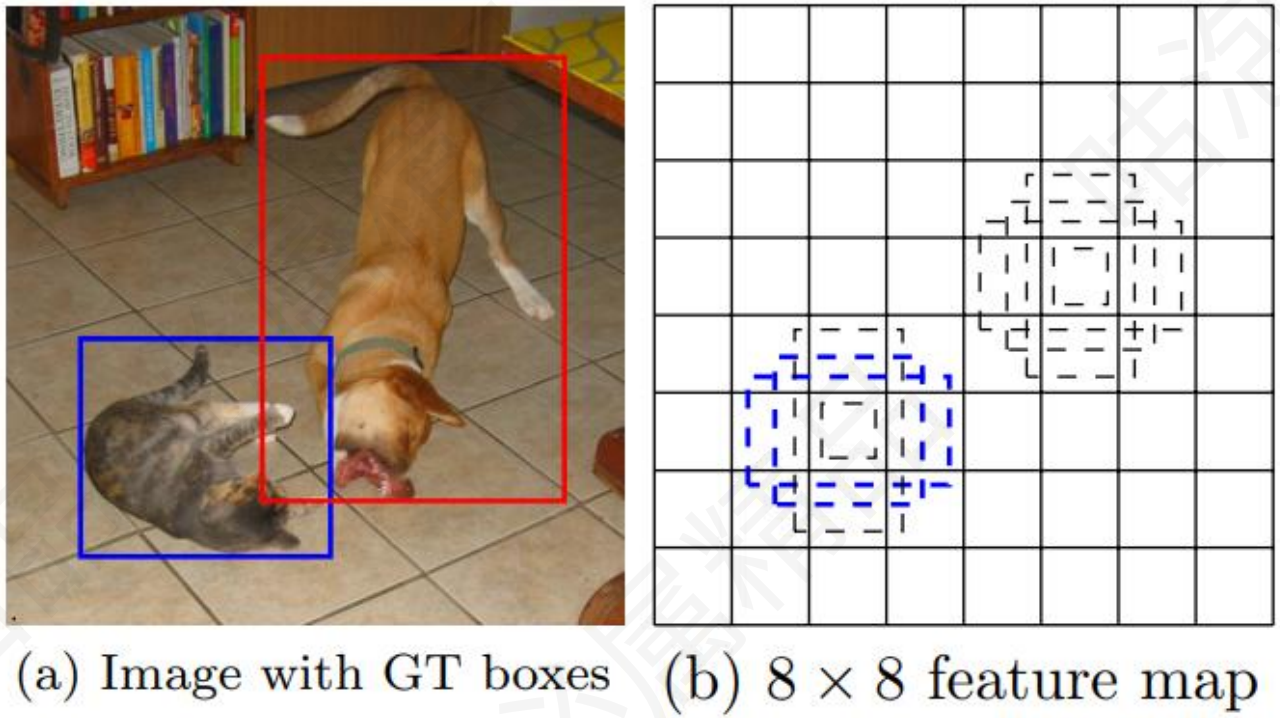


YOLOV7

- 正样本分配策略
  - 其实就是得看看特征图上每个点都会预测结果，哪些是正样本参与损失计算
  - 核心思想就是GT的中心点落在哪个点附近，其所对应的anchor就是正样本
  - 但是想一想，咱们任务是缺正样本还是负样本呢？
  - 为了正样本能更多，咱们直接threeble-kill

YOLOV7

- 正样本分配策略
  - 这里要根据差异来选择了
  - 肯定不能全部anchor都算
  - 1.长宽差异；2.类别预测差异



(a) Image with GT boxes (b) 8 × 8 feature map

- (1).初筛:0.25<gt与anchor长宽比例<4 (2).计算IOU (3).计算类别预测损失
  综上根据损失排名后再进行第二次筛选

YOLOV7

- 正样本分配之IOU损失计算：

  - 例如当前输入正样本3个，候选框13个则，可以得到[3,13]矩阵

0.9324	0.5090	0.9462	0.2005	0.5925	0.2649	0.6287	0.2600	0.8671	0.1942	0.0697	0.2821	0.4847
0.0783	0.5240	0.8098	0.8982	0.7448	0.3034	0.8672	0.9714	0.6882	0.2981	0.7148	0.4133	0.9950
0.4791	0.5816	0.2205	0.8987	0.4111	0.8098	0.3369	0.8633	0.2661	0.6773	0.3269	0.3906	0.3705

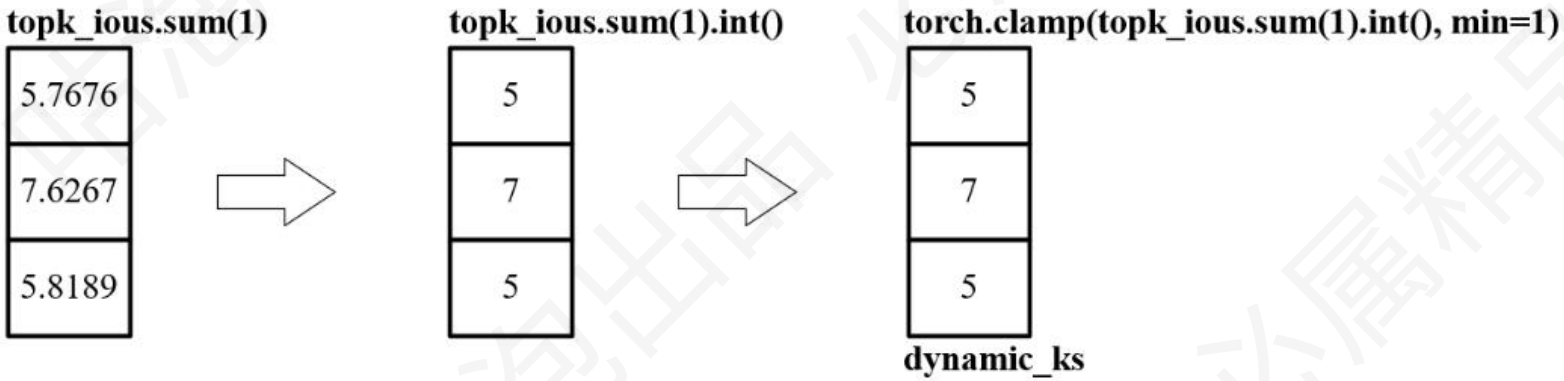
  - 接下来TOPK，多了是10个，要是不够就会更少，得到[3,10]矩阵

0.9462	0.9324	0.8671	0.6287	0.5925	0.5090	0.4847	0.2821	0.2649	0.2600
0.9950	0.9714	0.8982	0.8672	0.8098	0.7448	0.7148	0.6882	0.5240	0.4133
0.8987	0.8633	0.8098	0.6773	0.5816	0.4791	0.4111	0.3906	0.3705	0.3369

YOLOV7

- 正样本分配之IOU损失计算：

  - 得到每一个GT所对应的候选框数量，最少也得一个



  - sum我觉得可以当作有一些IOU都比较小，虽然排在前10，但是可能也没啥用

  - 求和就相当于，这些里面我根据IOU大小来看到底取几个合适，不是固定的

YOLOV7

- 正样本分配之IOU损失计算：

  - 但是在计算的时候有些候选框可能会对应多个GT，这就得筛选了，只能对应1个

matching_matrix.sum(0)

1	2	1	2	2	2	2	1	0	1	0	1	2

cost

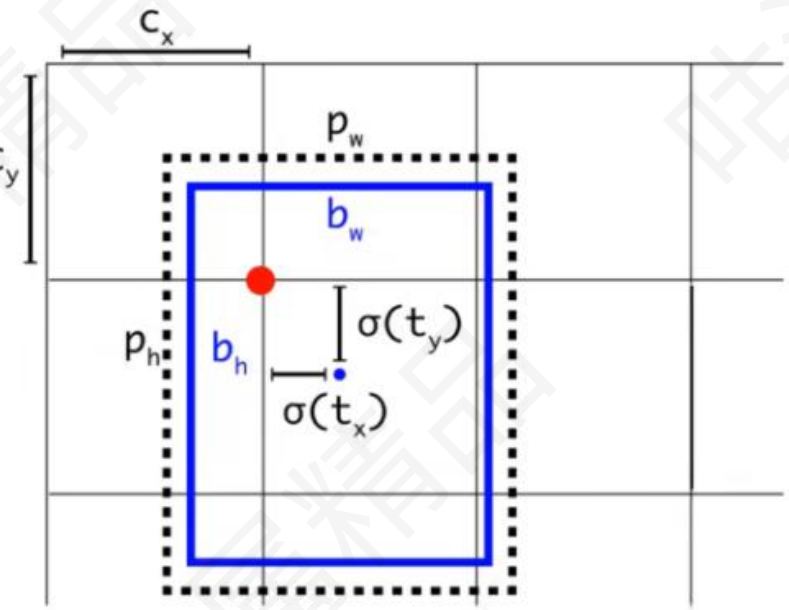
100001	8.263	68.259	100008	41.235	2.568	9.457	100006	98.288	100.235	100005	100009	11.248
1.258	100000	5.897	0.8982	3.257	5.126	100000	100000	100000	8.298	100002	0.4133	100003
100000	10.581	100000	41.898	100000	100000	8.336	80.863	90.256	100000	100.56	105.36	0.3705

  - 2就表示当前候选框对应2个GT，就得选其中损失最小的那一个

YOLOV7

- 要预测哪些家伙：

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$



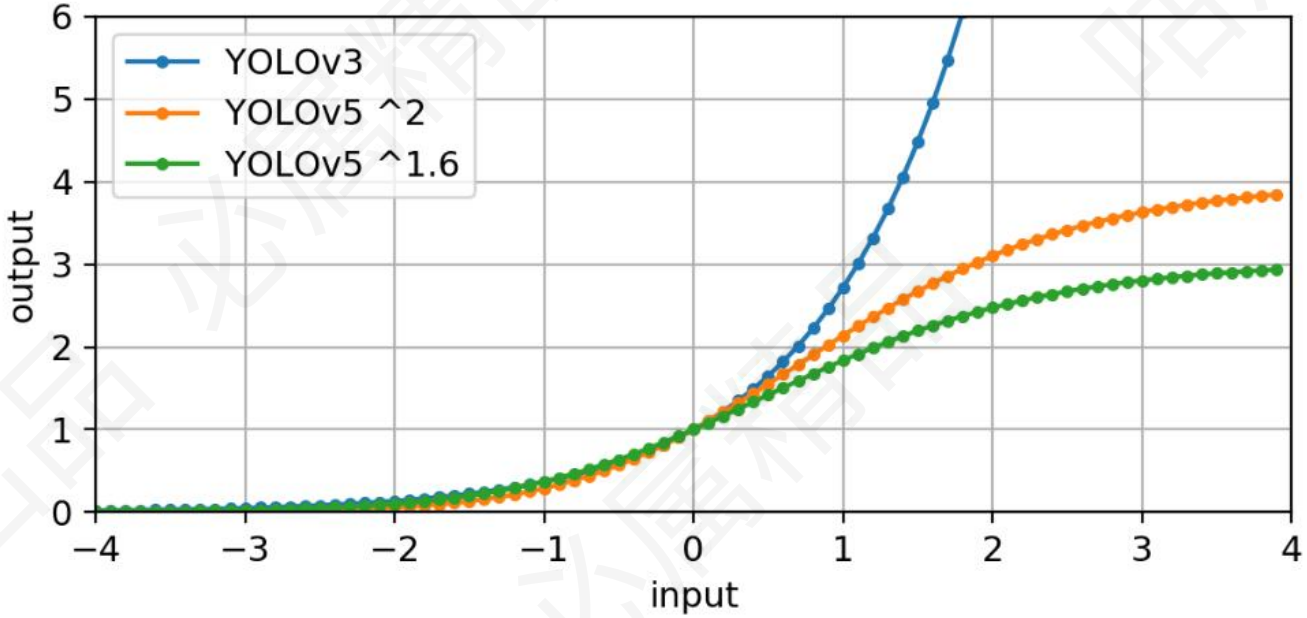
- V3V4要预测的结果：
- (黑色:anchor，蓝色:gt)

$$b_x = 2\sigma(t_x) - 0.5 + c_x$$
$$b_y = 2\sigma(t_y) - 0.5 + c_y$$
$$b_w = p_w(2\sigma(t_w))^2$$
$$b_h = p_h(2\sigma(t_h))^2$$

- V5V7要预测的结果：
- 这些数字都是有说法的，想想之前咱们算正样本的时候是不是有一个4倍的数字
  参考链接：https://github.com/ultralytics/yolov5/issues/471（建议阅读）

YOLOV7

- 预测值套sigmoid的原因
  - V3梯度直接炸了
  - V5平方后取值范围0-4
  - 也就是倍率不能差异太多



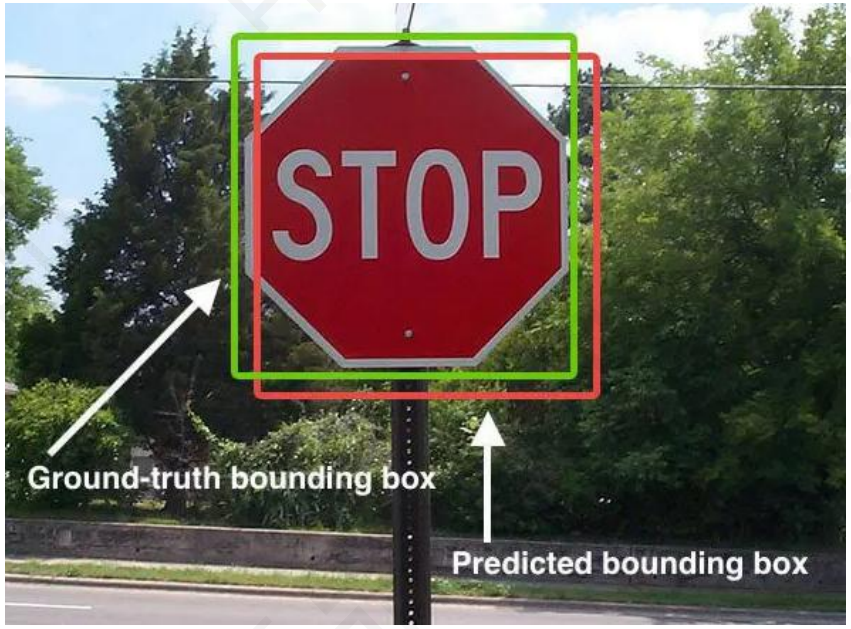
- 咱们选正样本的时候也是要选符合一定范围倍率的（0.25-4）
  这里用我语文老师的话就是上下文遥相呼应

YOLOV7

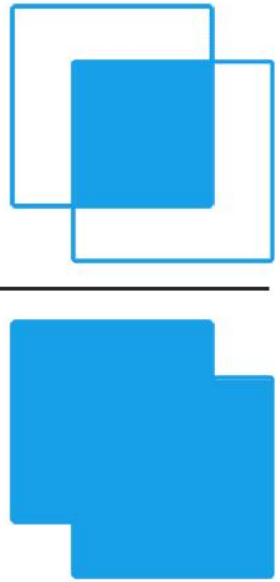
- AUX辅助输出
  - 输入更大，这回1280，但是上来就得下采样也是得到640实际输入
  - 4个输出层，每层还是3个anchor，这回一共得4*2=8个输出层了（带辅助了）
  - 但是辅助头在选择正样本时仍使用主头的预测结果（主的才是重要的）
  - 为了增加召回，选择了周围5个区域，也就是偏移量由0.5->1

YOLOV7

- 物体检测评估指标
  - IOU一会要用到，GT与预测之间的差异
  - 后续所有评估指标都建立在此基础上



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

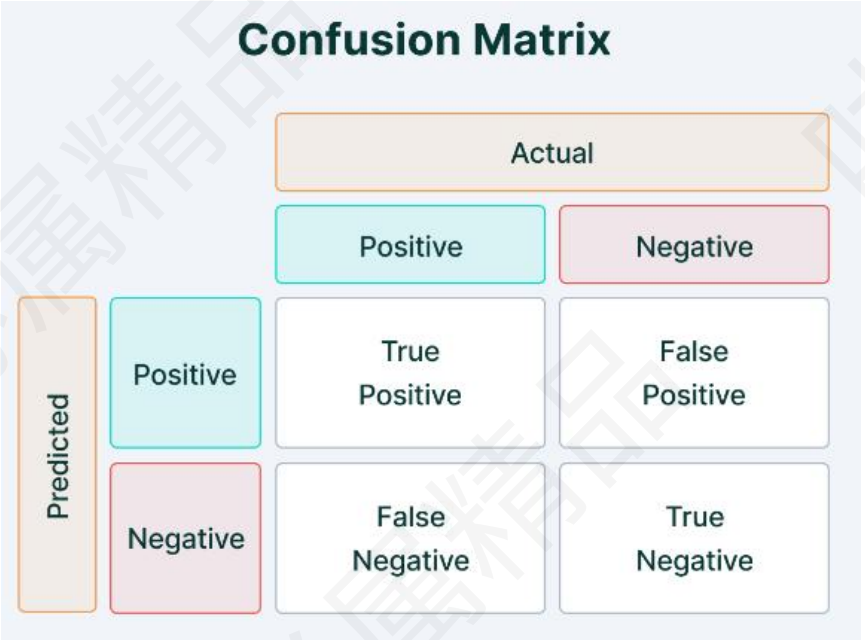


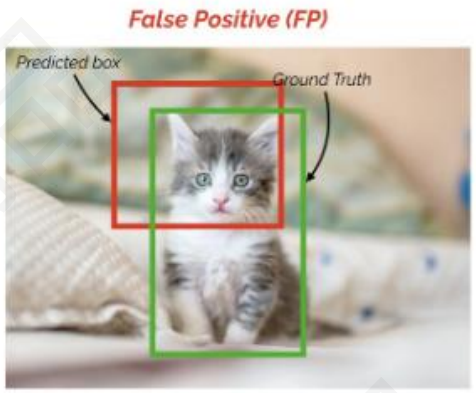
YOLOV7

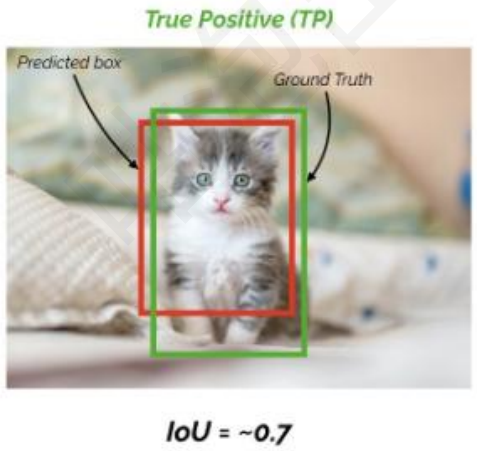
- 评估指标：

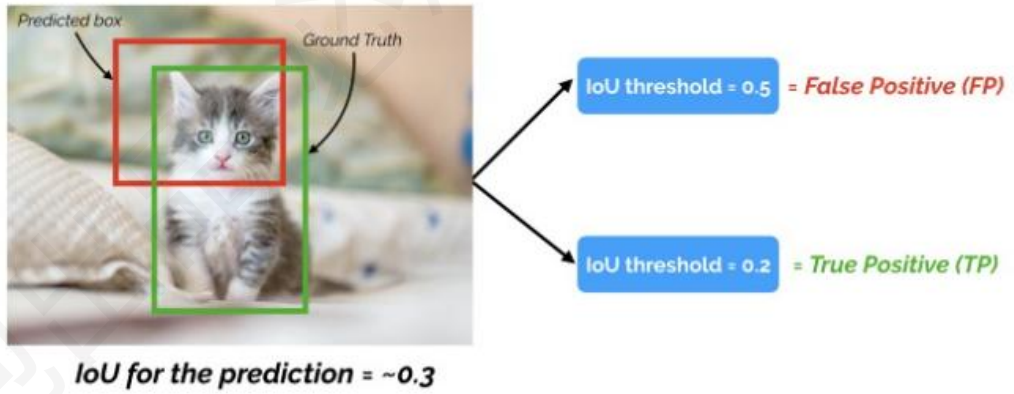
  - MAP这东西咋来的呢

  - 先通过阈值来选这几个数



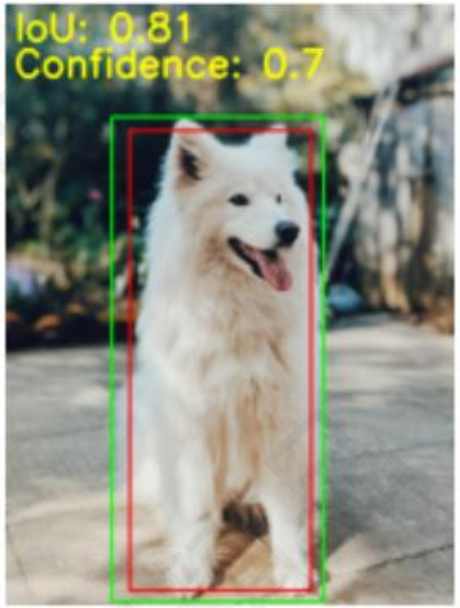


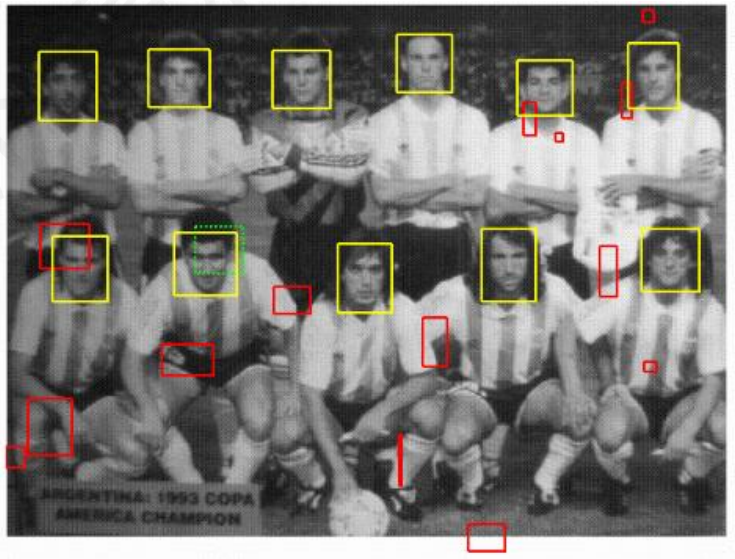




YOLOV7

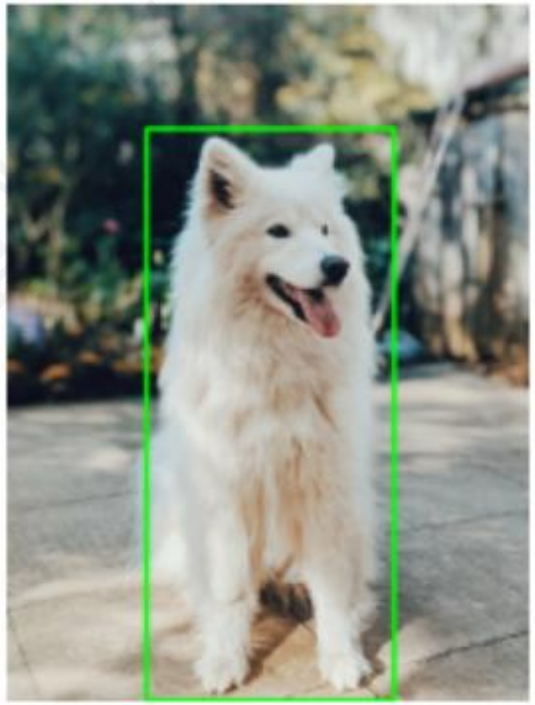
- TP, FP, TN,FN都是啥
  - TP的条件有两项：置信度和IOU都满足阈值
  - 就是既跟GT很相似然后预测是物体的可能性也高
  - FP的条件也挺多：1.置信度满足了但是IOU不够
  - 2.前两项都满足了，但是最终预测的类别不同





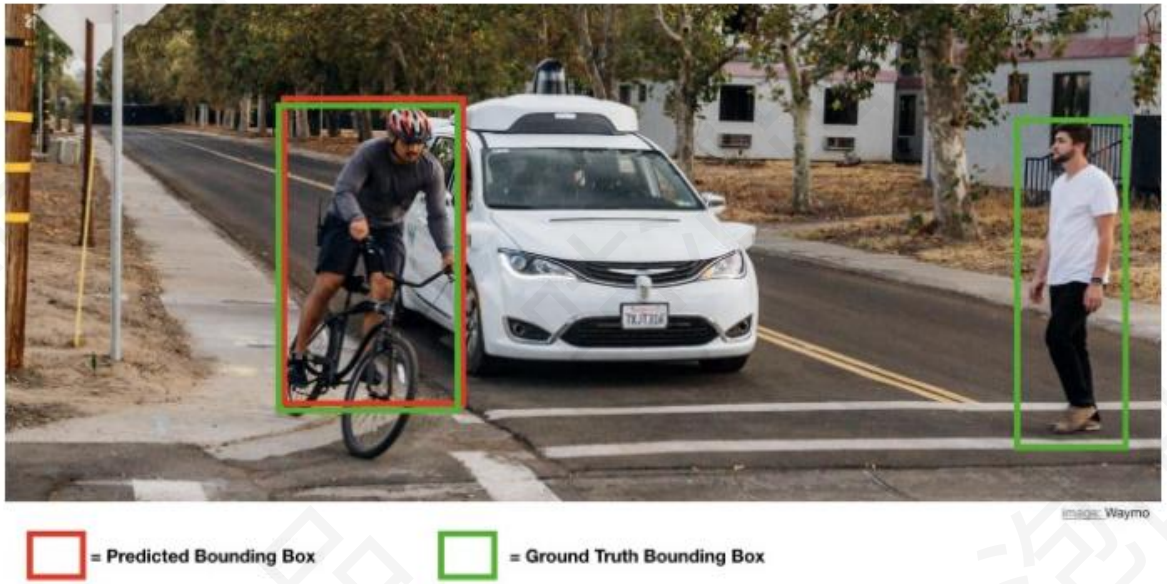
YOLOV7

- TP, FP, TN,FN都是啥
  - TN就别让我画了，不需要这个指标也没法画，背景当背景了
  - FN就是有GT，但是没有满足条件的预测框，当背景了
  - FN就相当于有GT没有被检测到，没人管它
  - 基本上一会要算的所有指标只需要这三项即可



YOLOV7

- 一般需要先计算P和R
- 这两项是咱们后续算AP的基础
- 两个绿色GT框，一个红色预测框



$$True\ Positives\ (TP) = 1$$

$$Fasle\ Positives\ (FP) = 0$$

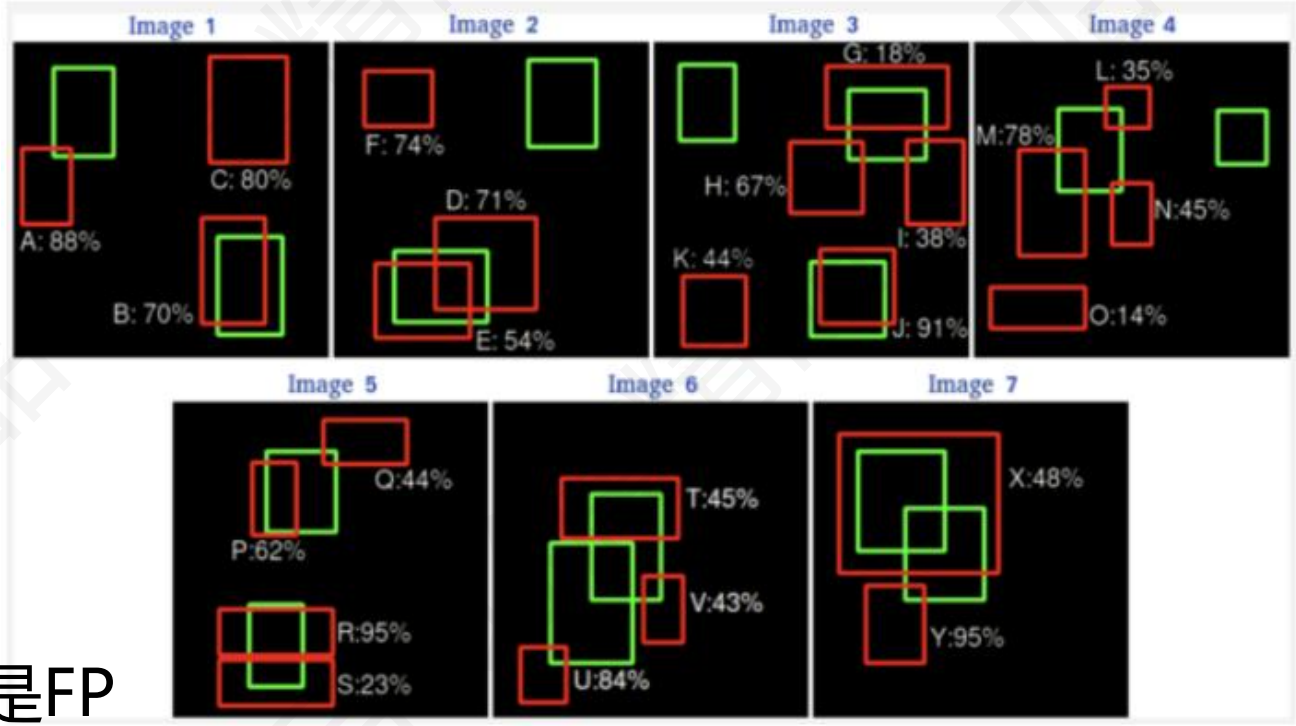
$$False\ Negatives\ (FN) = 1$$

$$Recall = \frac{TP}{TP + FN} = \frac{1}{1 + 1} = 0.5$$

$$Precision = \frac{TP}{TP + FP} = \frac{1}{1 + 0} = 1$$

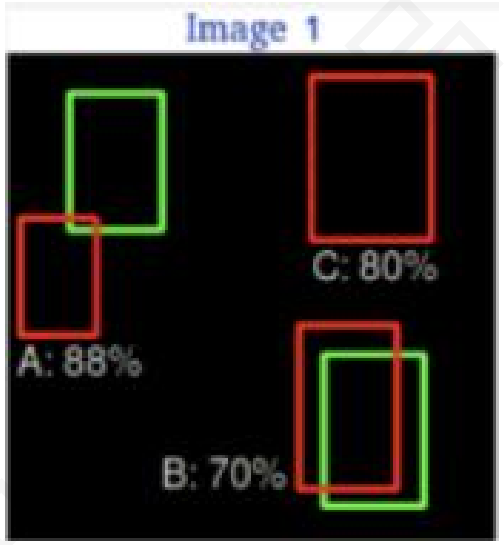
YOLOV7

- PR曲线
  - 7张测试图像数据
  - 绿色的是GT红色是预测
  - IOU需要满足条件默认0.5
  - 第一步就是先找到哪些是TP哪些是FP



YOLOV7

- PR曲线
  - 图1中只有B是TP
  - A不够，C离了个大谱
  - （7个图中的预测值每一个都跟GT算了一下）
  - 得到右边大长图



Images	Detections	Confidences	TP or FP
Image 1	A	88%	FP
Image 1	B	70%	TP
Image 1	C	80%	FP

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
Image 5	P	62%	TP
Image 5	Q	44%	FP
Image 5	R	95%	TP
Image 5	S	23%	FP
Image 6	T	45%	FP
Image 6	U	84%	FP
Image 6	V	43%	FP
Image 7	X	48%	TP
Image 7	Y	95%	FP

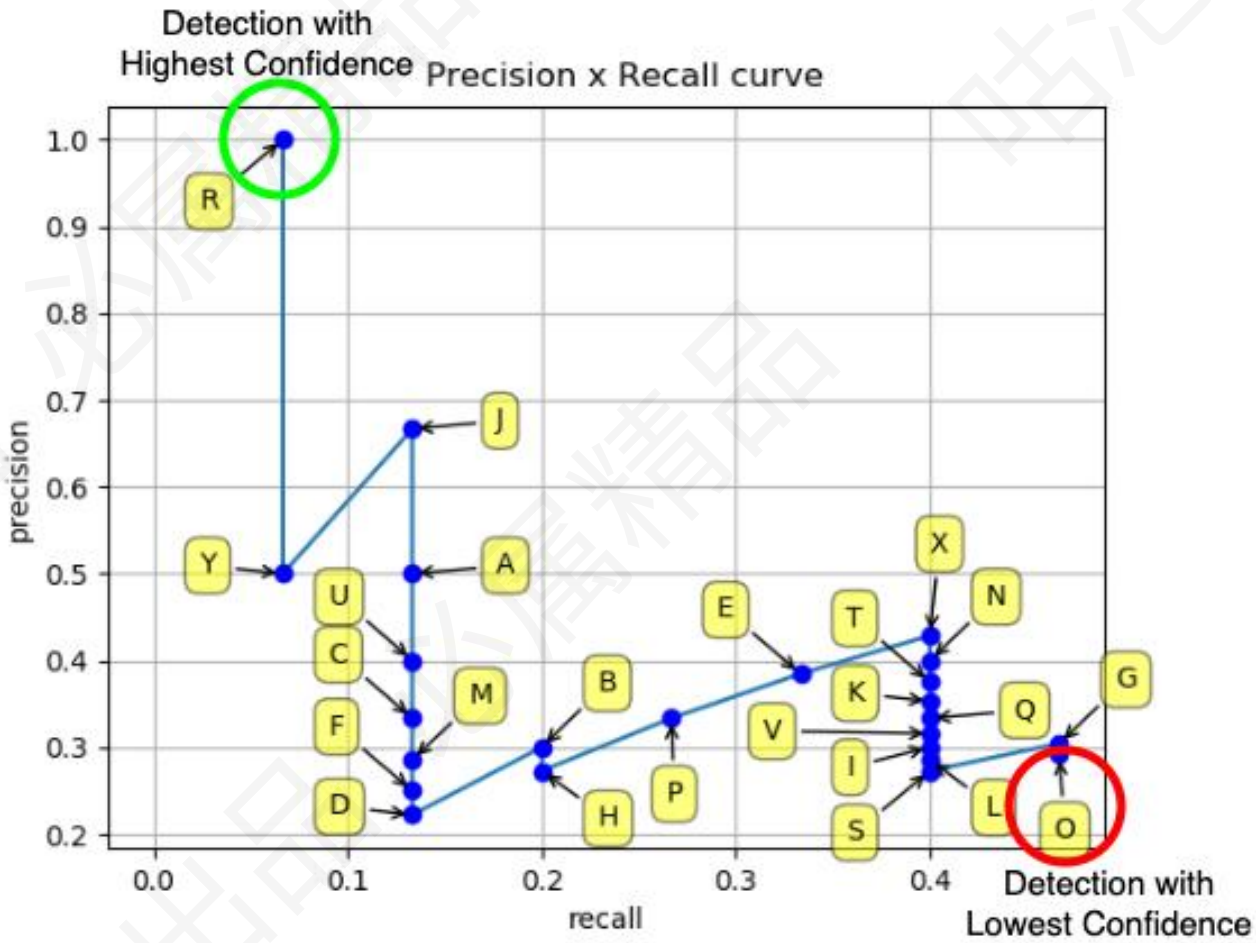
YOLOV7

- PR曲线
  - 先按照置信度排序
  - 同一GT只能匹配一个预测框
  - 选择置信度大的那个当TP
  - ACC TP/FP表示累加之后的结果

Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666
Image 2	E	54%	1	0	5	8	0.3846	0.3333
Image 7	X	48%	1	0	6	8	0.4285	0.4
Image 4	N	45%	0	1	6	9	0.4	0.4
Image 6	T	45%	0	1	6	10	0.375	0.4
Image 3	K	44%	0	1	6	11	0.3529	0.4
Image 5	Q	44%	0	1	6	12	0.3333	0.4
Image 6	V	43%	0	1	6	13	0.3157	0.4
Image 3	I	38%	0	1	6	14	0.3	0.4
Image 4	L	35%	0	1	6	15	0.2857	0.4
Image 5	S	23%	0	1	6	16	0.2727	0.4
Image 3	G	18%	1	0	7	16	0.3043	0.4666
Image 4	O	14%	0	1	7	17	0.2916	0.4666

YOLOV7

- PR曲线
  - 就是把排序好的P和R值画出来
  - 这俩基本是矛盾的，很难又准又全
  - AP就是要求PR曲线的面积了
  - （不同数据集求法可能不同的）



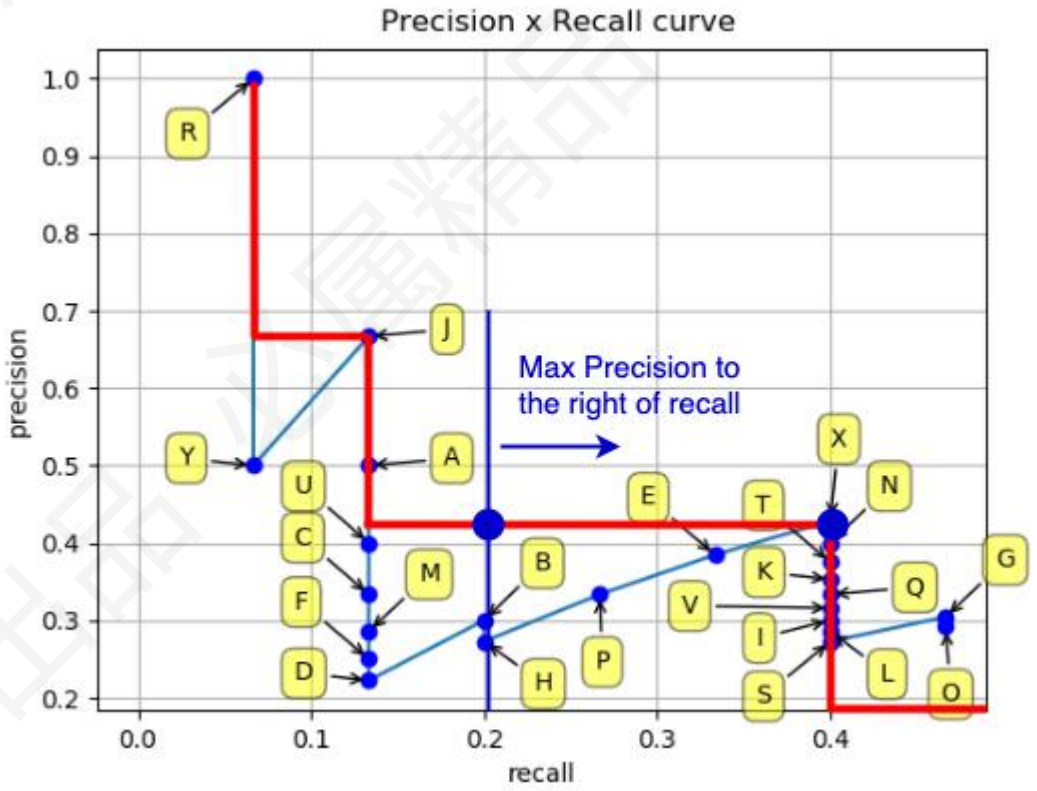
YOLOV7

- AP计算（11点插值方法）
  - VOC中的计算方法：$AP = \frac{1}{11} \sum_{r \in (0, 0.1, \ldots, 1)} p_{interp(r)}$ （0-1，间隔0.1一共11个位置）
  - 也就是计算其与坐标轴围成的面积大小
  - 单个类别的这么算，MAP就是各类平均(10年之前计算方法)

$$\text{AP} = \frac{1}{11}(1 + 0.66 + 3 \times 0.4285 + 6 \times 0)$$

$$\text{AP} = \frac{1}{11}(2.9521)$$

$$\text{AP} = 0.2683$$



YOLOV7

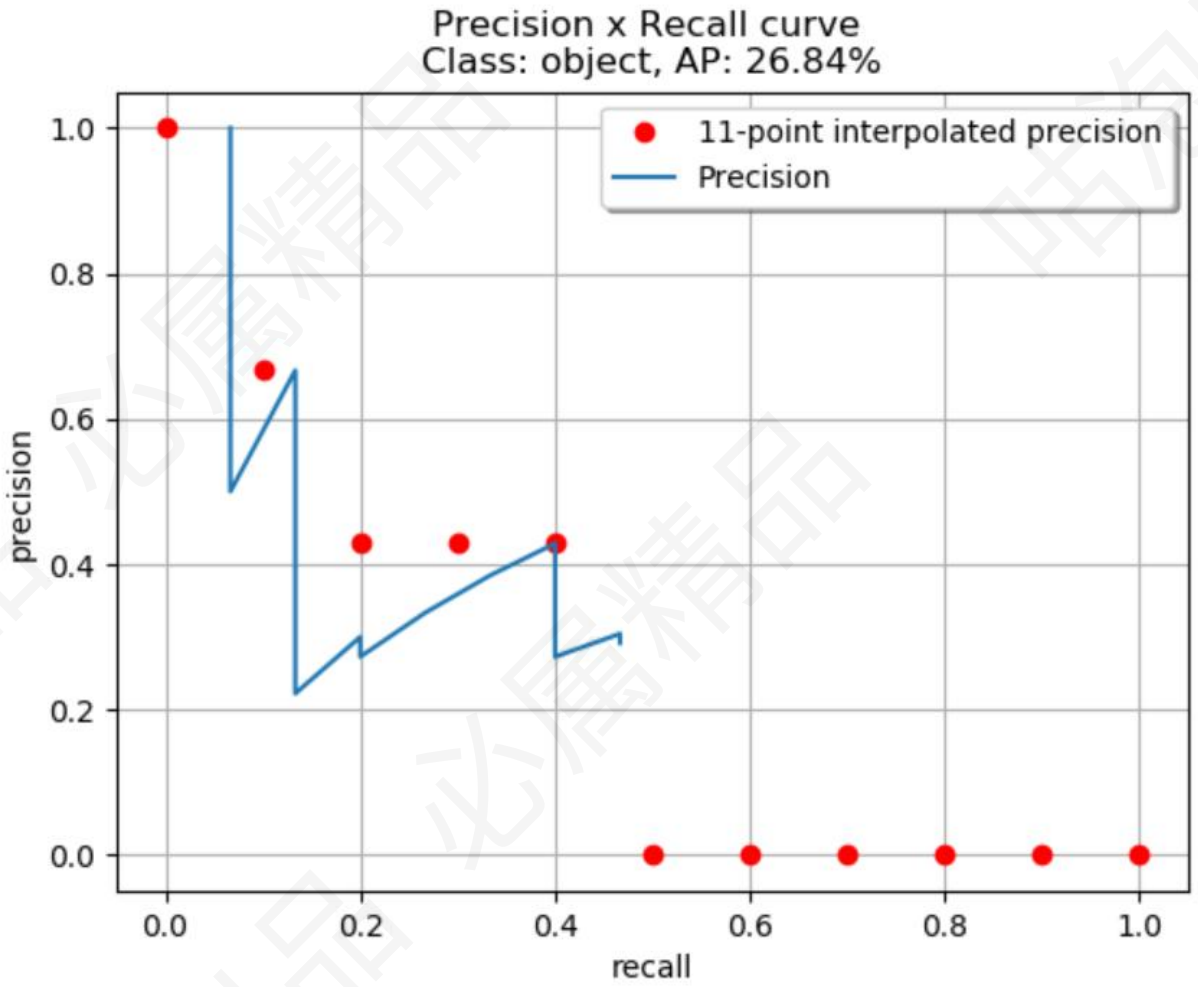
- 插值方法（梦回2008）
  - 每一个点考虑右边最大的就行了
  - 画出来11个点，带入公式

$$AP = \frac{1}{11}(1 + 0.66 + 3 \times 0.4285 + 6 \times 0)$$

$$AP = \frac{1}{11}(2.9521)$$

$$AP = 0.2683$$

  - 参考链接：

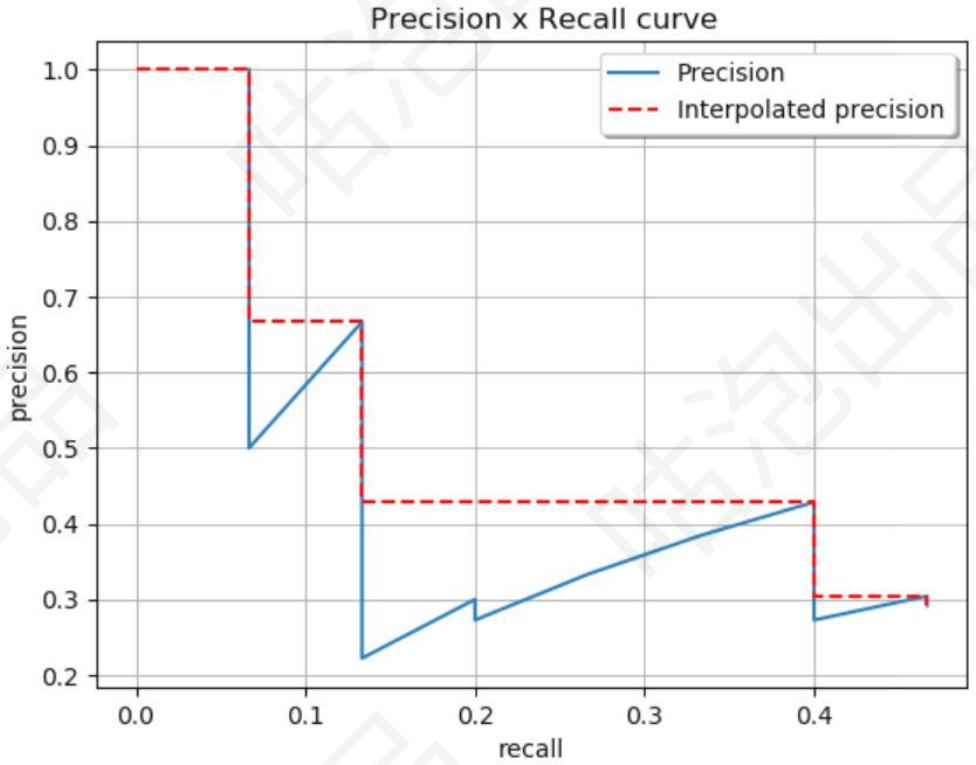


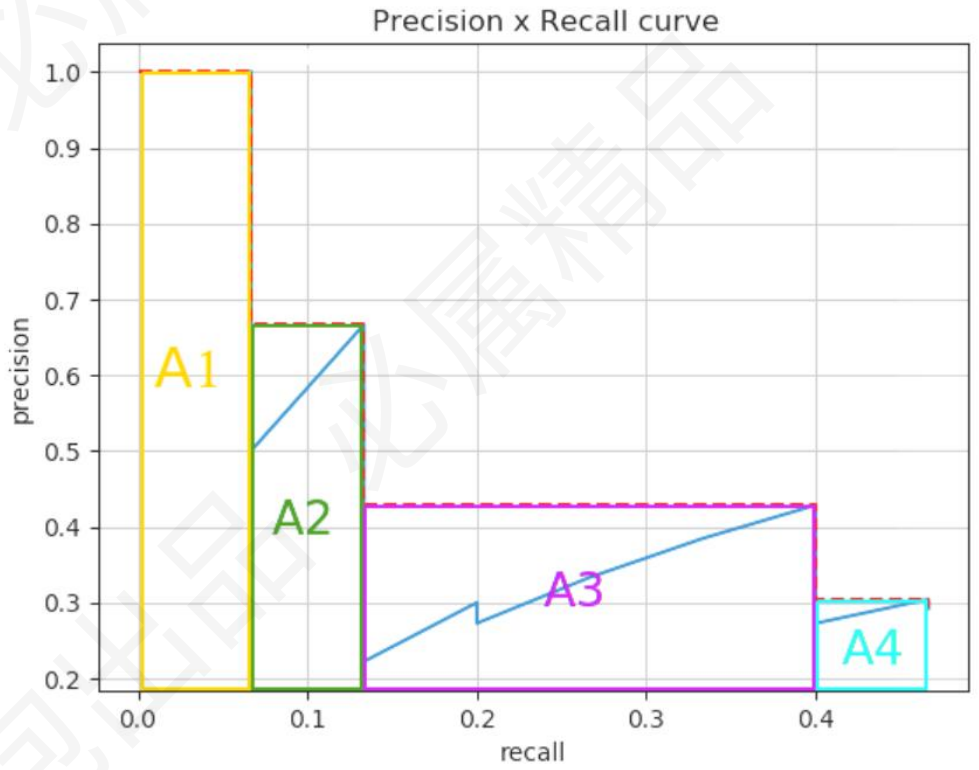
https://github.com/rafaelpadilla/Object-Detection-Metrics

YOLOV7

- AP计算：

  - 10年之后就开始这种了，考虑全部的点，$AP = A1 + A2 + A3 + A4$





YOLOV7

- COCO咋做的呢

  - COCO中的计算方法其实更科学，VOC一视同仁，COCO多劳多得

  - 阈值不再仅仅是0.5，从0.5到0.95间隔0.05，这样就有10个阈值了

  - 然后计算10个的平均AP也就是(AP@[0.5:0.05:0.95])

```
Average Precision (AP):
   AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small            % AP for small objects: area < 32^2
   AP^medium           % AP for medium objects: 32^2 < area < 96^2
   AP^large            % AP for large objects: area > 96^2
```