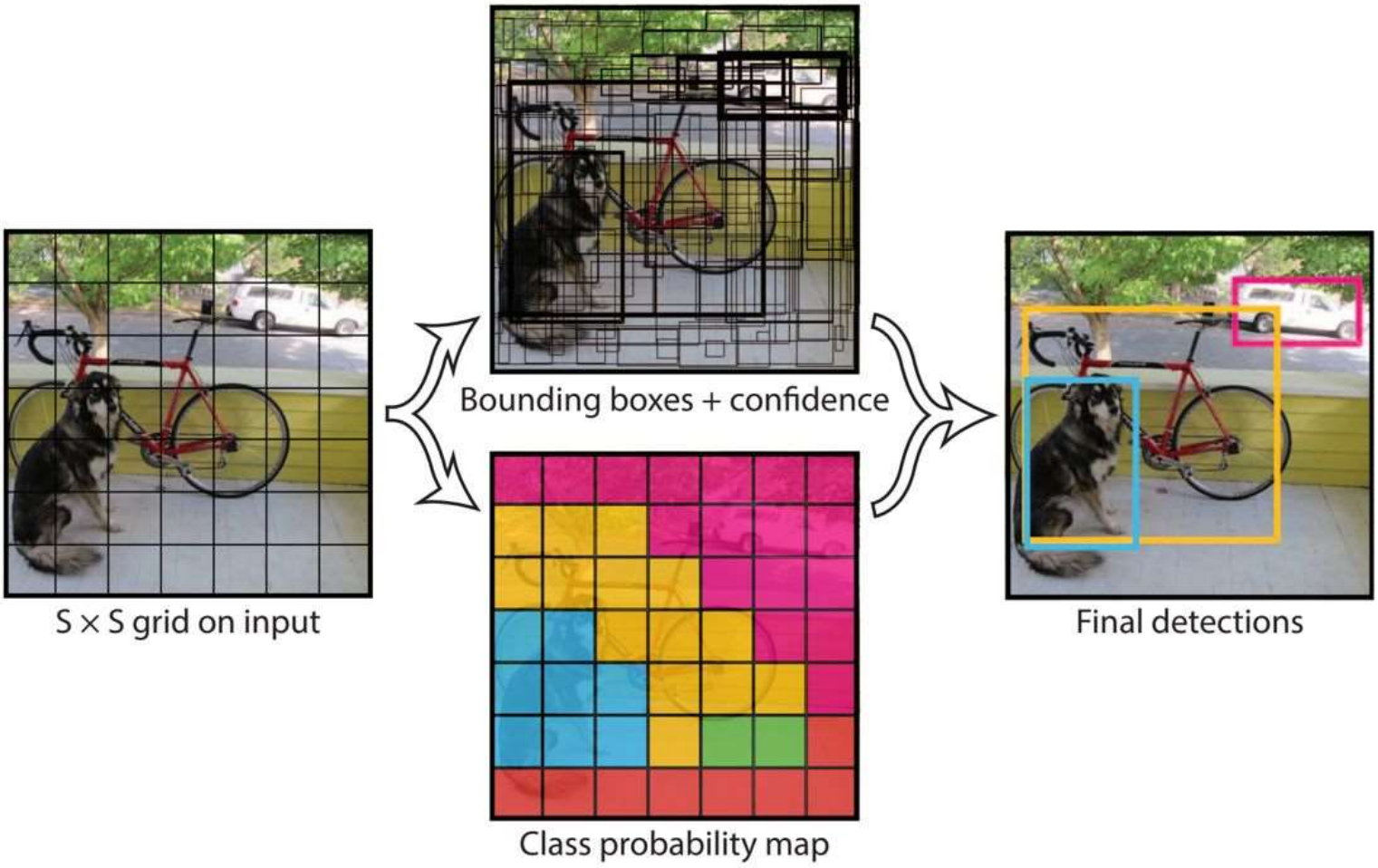


1.2 YOLO入门教程：YOLOv1(2)-浅析YOLOv1

Kissrabbit
业余写手/机器人/深度学习/计算机视觉

说明：

这一节的主题内容摘自于我的《目标检测手把手入门教程》专栏中介绍YOLOv1的一章。为了适应本教程型专栏，对内容进行了适当的调整。

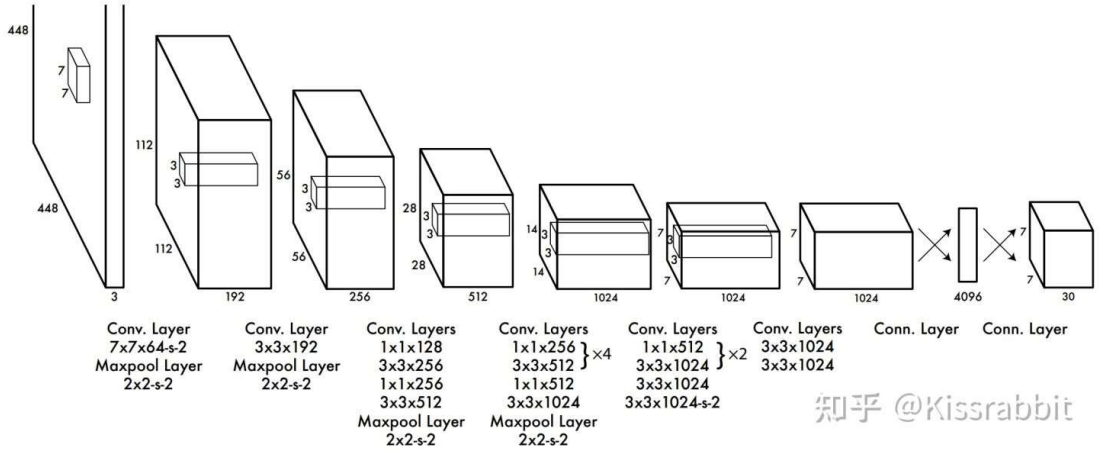
从本节开始，我们要正式进入YOLO从零开始的入门教程内容了，如果觉得前面的科普还不够充分，不妨搜索其他文章进行补充，笔者就不再花费过多的笔墨了。毕竟，前方的景色更美。

1、YOLOv1的网络架构

作为One-stage工作的开山之祖，YOLOv1以其简洁的网络结构和GPU实时检测速度而一鸣惊人，打破了R-CNN的“垄断”地位，为目标检测领域带来巨大的变革。

以现在的眼光来看待YOLOv1，会发现其中很多弊端，不过，在当年，YOLOv1可是炙手可热，为后来许多one-stage工作提供了框架基础。其里程碑意义不可磨灭。

YOLO-v1最大的特点就在于：**仅使用一个卷积神经网络端到端地实现检测物体的目的。**其网络整体的结构如下图所示：



YOLOv1网络结构

YOLO-v1的backbone网络是仿照GoogLeNet搭建的，但并没有采用Inception模块，而是使用 1×1 和 3×3 卷积层来堆砌的，所以网络的结构是非常简单的。基本上，照葫芦画瓢，我们就能搭建出整体的网络。

在那个年代，图像分类任务中，网络最后会将卷积输出的特征图拉平（**flatten**），得到一个一维向量，然后再接若干全连接层做预测。

YOLO-v1继承了这个思想，最后的 $7 \times 7 \times 1024$ （维度顺序： $H \times W \times C$ ）的特征图成 $7 \times 7 \times 1024$ 大小的向量，这个一维向量再接一层全连接层。但是，这里我们需要注意一点，如果只看图中的结构，是只有一个4096的全连接层，我们不妨算一下这里的参数量：

$$7 \times 7 \times 1024 \times 4096 + 4096 \approx 2 \times 10^8$$

仅仅这一层全连接，参数的量级就已经是8个0了，过于巨大。

因此，原文作者实际是先用了一个256的全连接先缓一缓，然后再连接4096，从而避免参数爆炸，这样我们再算一下：

$$7 \times 7 \times 1024 \times 256 + 256 + 256 \times 4096 + 4096 \approx 1.4 \times 10^7$$

少了一个数量级，似乎也没有好到哪去。当然，参数多并不意味着模型推理速度就一定会很慢，这些多的参数主要集中在全连接层，而全连接层的运算可以变换成普通的矩阵乘法，这种普通运算对GPU来说，连塞牙缝的资格都没有。但是，从资源占用的角度来看，过大的参数量还是会让研究者头疼的。

尽管如此，YOLOv1还是以其优越的检测性能惊艳全场。

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

截取自YOLOv1原文

事实上，YOLOv1本身最大的弊端就在于“**flatten**”这种方式本身。

基本上，对于“**flatten方式会破坏特征的空间结构信息**”的这一观点已经成为业界共识。现在几乎看不到还有哪个one-stage检测模型还会用全连接层来做检测了。

另外，我们注意到，YOLOv1中并没有使用**BatchNormalization（BN）**，这是因为在那个时候，BN还没有兴起。

2.YOLOv1的检测原理

一张图像输入给网络，网络最后输出一个$7 \times 7 \times 30$的特征图。其中，7×7是原图448×448经过64倍降采样（即网络最终的stride为64）得到的，而通道数**30**的含义是：

特征图的每个位置$(grid_x, grid_y)$预测**两个bounding box（bbox），而每个bbox包含五个输出参数：置信度C，矩形框参数(c_x, c_y, w, h)，共10个参数，再加上20个类别，一共就是30了。**置信度C的作用是判断此处是否有目标的中心点。

通常，置信度C也被记作**objectness预测**，表征此处是否有物体的中心点，即是否有物体。

PS：之所以是20个类别，是因为那时候的数据集只有PASCAL VOC，共20个类别，COCO还没提出来。

更一般的，我们可以用下面的公式来计算这个特征图的通道数：

$$5B + C$$

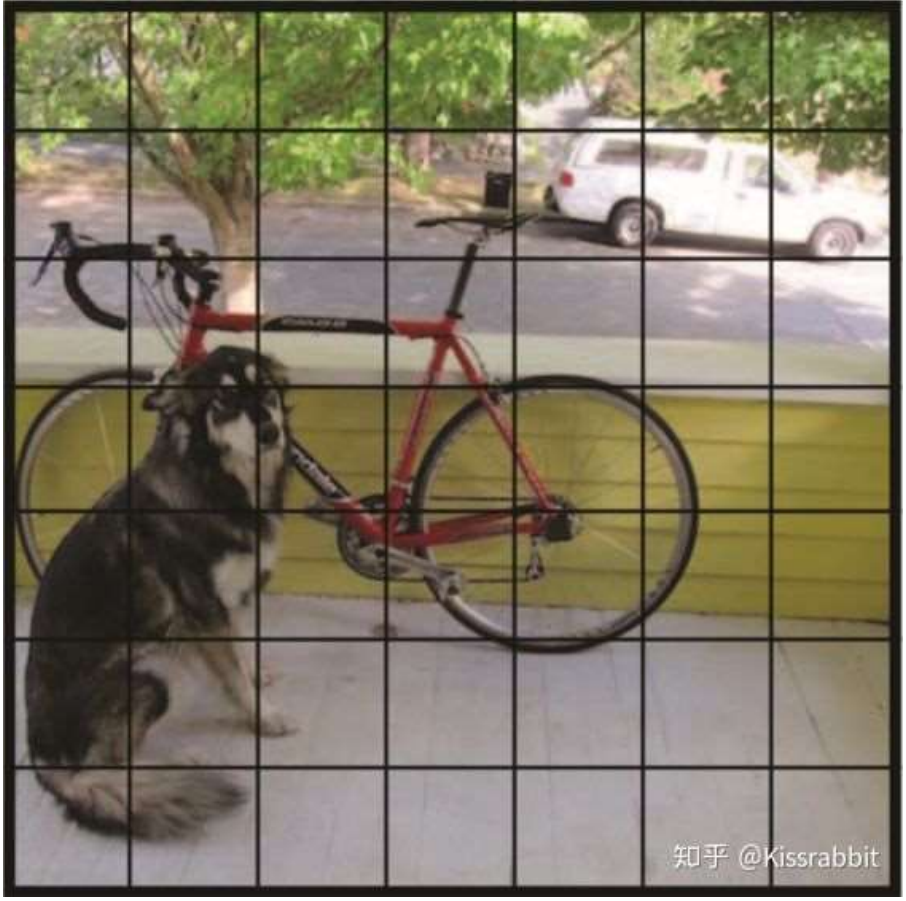
其中，5是指边界框的置信度和位置参数；B是每个位置预测的bbox数量，C是类别的数量（如PASCAL VOC中有20个类别，MSCOCO中常用的是80个类别）。

那么，这个$7 \times 7 \times 30$的特征图到底是怎么来的呢？

首先，网络的输入是448×448的图片，经过网络64倍的降采样后，最后的卷积输出是7×7的（$448 \div 64 = 7$）——在这里停一下，因为这里就体现了YOLOv1的核心思想，也是自此之后绝大部分的one-stage检测的核心范式：

逐网格找东西。

具体来说就是，这个7×7相当于把原来的448×448的图片进行了7×7等分，如下图所示：



图三. YOLO-v1的网格等分思想

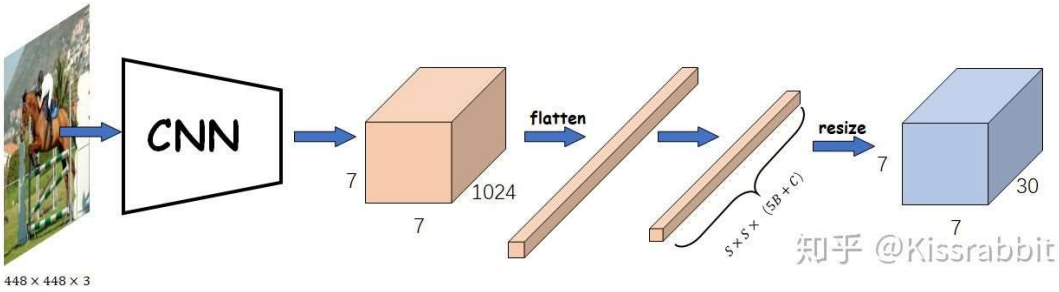
YOLOv1是想通过看这些网格来找到物体的**中心点坐标**，并确定其**类别。**具体来说，就是每一个网格都会输出 B 个bbox和 C 个类别的置信度，而每个bbox包含5个参数（框的置信度+框的坐标参数），因此，每个网格都会给出 $5B+C$ 个预测参数。因此，网络最终输出的预测参数总量就是：

$$S \times S \times (5B + C)$$

其中 $S = \dfrac{\text{输入图像尺寸}}{\text{网络的最大}\boldsymbol{stride}}$ ，

原文中，$\text{输入图像尺寸} = \mathbf{448}$ ，$\text{网络的最大}\boldsymbol{stride} = \mathbf{64}$ 。

因此，知道了最后输出的预测参数量后，在4096全连接层后面接多大的全连接层也就清楚了。于是，YOLOv1经过最后的全连接层来输出一个 $S \times S \times (5B + C)$ 大小的向量，随后再将其resize成一个 $S \times S \times (5B + C)$ 大小的特征图（这一步的resize没有任何特殊含义，仅仅是为了和上面的网格形式对应，处理起来方便，请读者不要多想。）



图四. YOLO-v1整体工作流程

因此，YOLOv1就是在每个网格上去做预测，理想情况下，包含了物体中心点的网格会有很高的置信度输出，而不包含中心点的网格的置信度输出应该十分接近0。

总的来说，YOLOv1一共有三部分输出，分别是objectness、class以及bbox：

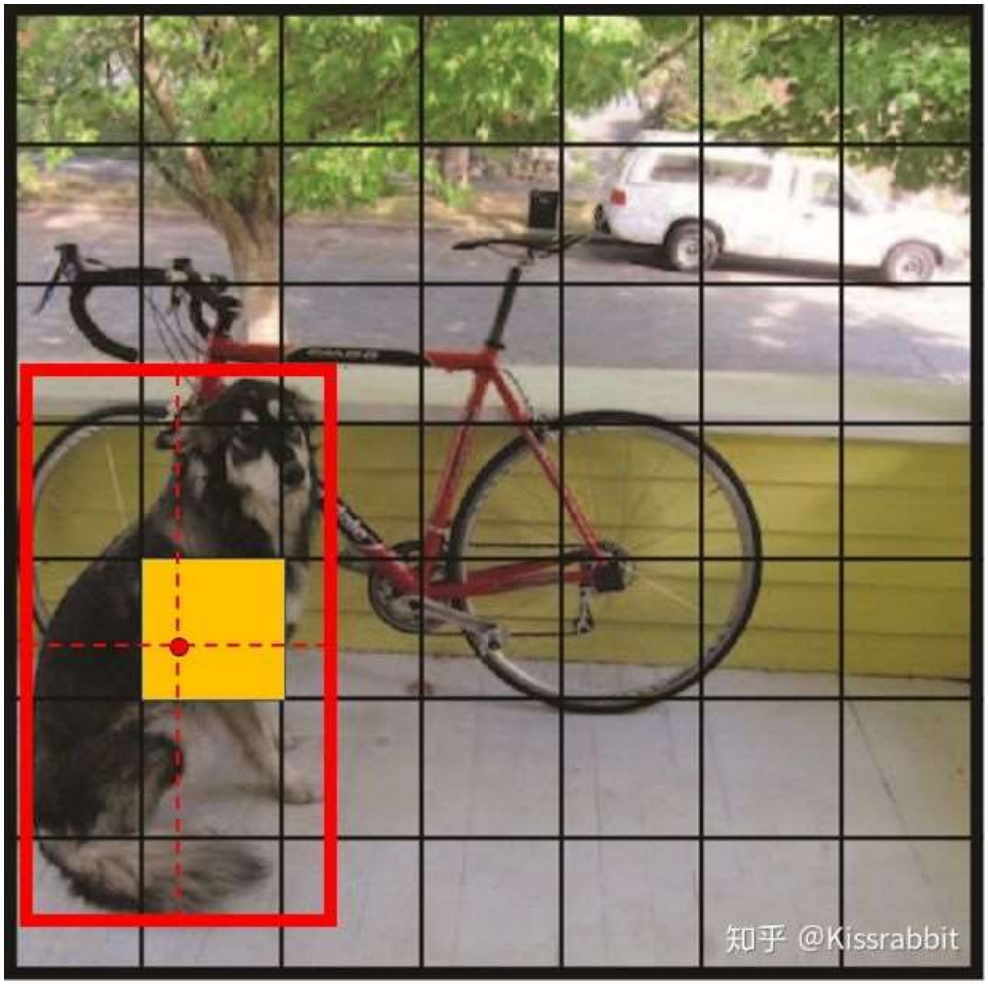
- objectness就是上面所说的**框的置信度**，用于表征该网格是否有物体；
- class就是**类别预测**；
- 而bbox就是**边界框(bounding box)**。

明确了网络的检测基本原理后，接下来就应该考虑如何使得网络学习到这一能力，换言之，或者说，为了训练这个YOLO-v1，到底应该如何去设计groundtruth呢？

3、YOLOv1的正样本制作方法

我们已经知道，YOLOv1最后输出一个 7×7 的网格，每个网格会给出30个参数，包括两个bbox的5个参数（框的置信度+框的坐标参数）和20个类别置信度。首先，先说一下每个bbox的5个参数的含义。

由于YOLOv1是去预测物体的中心点，并给出矩形框，因此，包含中心点的网格，我们认为这里是有物体的，即这一网格的objectness的概率为1： $Pr(objectness) = 1$ ，如下图所示：



包含中心点的网格会被视为训练过程中的正样本，其他为负样本。

黄颜色代表这个网格有物体， $Pr(objectness) = 1$ 也就意味着，物体的中心点落在了这个网格中，如图中的红点所示，也就是一个**正样本**。

*======== **细节说明分割线** ========*

在这里暂停一下，根据一些读着的反馈，大家都会问测试阶段怎么知道Pr(objectness)是0还是1。这里需要做个解释：

*论文中所说的Pr(objectness)=1，以现在的技术角度来看，就是指“**正样本**”。*

*首先要清楚一点，YOLO一共有三个预测：objectness、class、bbox。其中，o**bjectness是一个二分类，即有物体还是无物体，也就是“边界框的置信度”**，对应loss函数中的那个“C”，**没物体的标签显然就是0，而有物体的标签可以直接给1，也可以计算当前预测的bbox与gt之间的IoU***

*作为有物体的标签，注意，这个IoU是objectness预测学习的标签。class就是类别预测，只有**正样本**处的grid cell才会被训练，也就是Pr(objectness)=1的地方，注意，这个Pr(objectness)=1就是指正样本的地方，和IoU没关，和YOLO没关，只和label有关，因为gt box的中心点落在哪个grid，哪个grid就是正样本，也就是Pr(objectness)=1。bbox也是同理。*

*在测试阶段，YOLO一共会输出三个预测，是否有物体的objectness预测、class预测和bbox预测。首先，我们计算score=objectness*class作为每个边界框的得分score，这个score也就是边界框的置信度confidence，论文中给出的计算公式如下：*

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

YOLOv1中给出的测试阶段bbox的置信度计算公式

*对于这个公式最大的问题就在于那个 IOU_{pred}^{truth} ，因为**测试阶段根本就没有真实框能够让我们去计算IoU**。但是，在上面已经说到了，YOLO的objectness的学习标签就是预测框和真实框之间的IoU，所以，在测试阶段，这个 IOU_{pred}^{truth} 其实就是指YOLO预测的objectness，它的物理意义就是：**这个grid cell是否有物体**。所以可以认为objectness隐含了IoU的概念，但本质就是有无物体的预测。因此，上面的那种写法（个人认为）具有误导性。*

基于YOLO的三个预测，我们稍微改写一下测试阶段的confidence计算公式：

$$score = Pr\left(objectness\right) \times Pr\left(Class_i\right)$$

这个IoU就是objectness预测，因为objectness在训练过程中的正样本标签就是IoU，

*========= **细节说明分割线** =========*

OK，我们继续——

注意看上面的图，我们会发现这个中心点相对于它所在的网格的四边是有偏距的，这其实就是由于降采样带来的量化误差，因此，我们只要获得了这个量化误差，就能获得中心点的准确坐标了，那么YOLOv1中是怎么计算这个量化误差的呢？

首先，对于给定的**真实的bbox坐标** $(x_{min}, y_{min}, x_{max}, y_{max})$ ，计算它的宽 w 和高 h ：

$$w = x_{max} - x_{min}$$

$$h = y_{max} - y_{min}$$

中心点坐标：

$$center_x = \frac{x_{min} + x_{max}}{2}$$

$$center_y = \frac{y_{min} + y_{max}}{2}$$

获得了中心点坐标后，我们就可以直接用下式就能确定出它落在了网格的哪个位置：

$$grid_x = \lfloor \frac{center_x}{stride} \rfloor$$

$$grid_y = \lfloor \frac{center_y}{stride} \rfloor$$

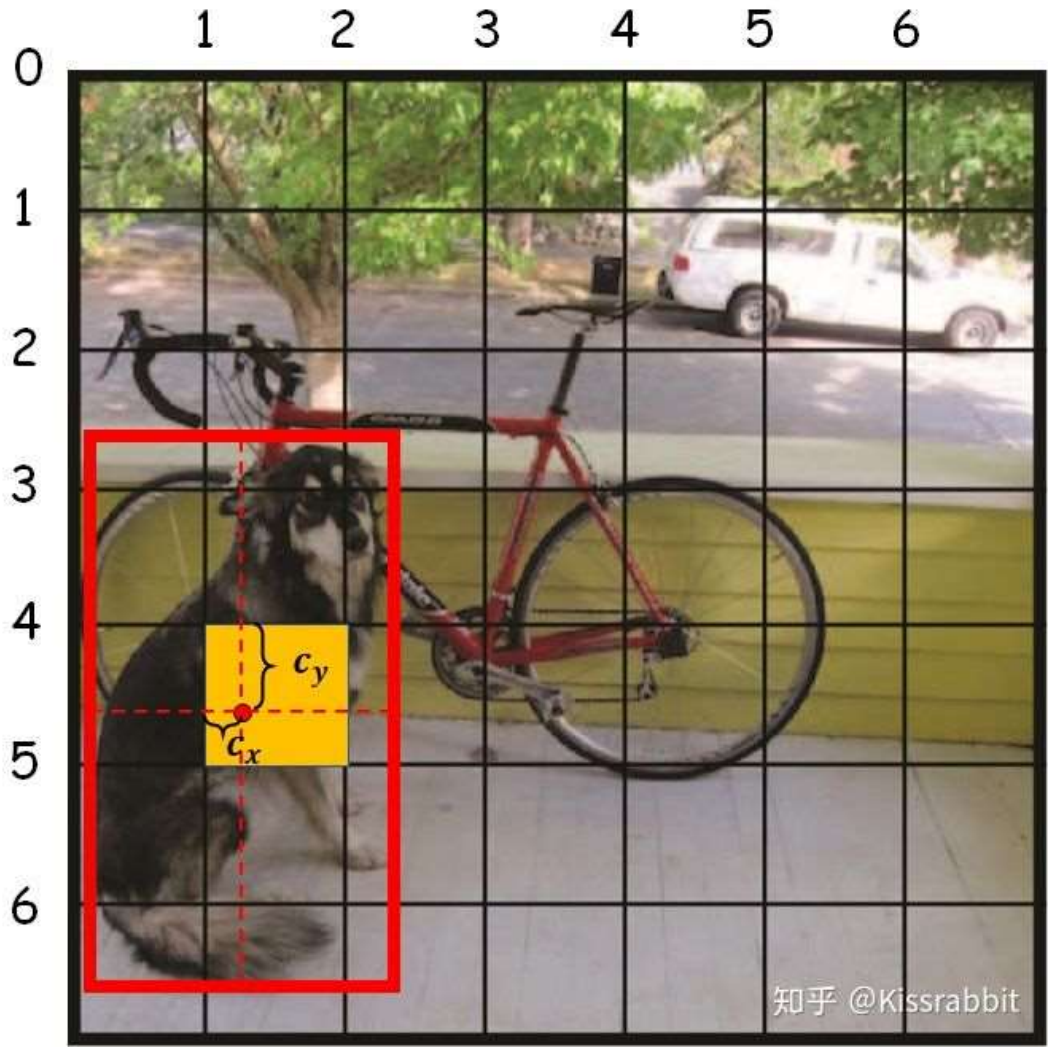
因此，这一处就是**正样本**。其中的 $stride$ 就是降采样的倍数， $\lfloor \cdot \rfloor$ 表示向下取整。

很明显，量化误差就出来了：

$$c_x = \frac{center_x}{stride} - \lfloor \frac{center_x}{stride} \rfloor$$

$$c_y = \frac{center_y}{stride} - \lfloor \frac{center_y}{stride} \rfloor$$

这里的 c_x， c_y 便是YOLOv1关于中心点坐标所要学习的目标，显然， $0 \leq c_x, c_y \leq 1$ 。可以参见下图：



那么，我们如何从网络输出的 c_x， c_y 反解出 $center_x$， $center_y$ 呢？很简单，在 7×7 的网格中，对于每个位置 $\left(grid_x, grid_y\right)$ ，用下式即可计算出相应的中心点：

$$center_x = (grid_x + c_x) \times stride$$

$$center_y = (grid_y + c_y) \times stride$$

对于一个矩形框，中心点确定了它的位置，那么对于它的大小，则由宽和高来决定，那么关于bbox剩下的两个坐标参数就直接设定为 w， h 。不过，由于 $0 \leq c_x, c_y \leq 1$ ，而 w， h 通常远大于1，所以，这会很容易造成两部分的loss差距过大。因此，实际上，YOLOv1先将将 w， h 都除以图像的大小，也就是简单的归一化操作：

$$w = \frac{w}{w_{image}}$$

$$h = \frac{h}{h_{image}}$$

到此，我们已经清楚了YOLOv1是如何确定正样本，以及正样本的位置信息是如何得到的。其实，读者若是已经对anchor-free方法有所了解了，是不是会觉得这里头很熟悉呢？

是的，没错，YOLOv1就是最早的anchor-free通用检测器！

瞧瞧，是不是觉得有的时候科研就是在开“历史倒车”呢？不过，从螺旋发展的角度来看，这并不奇怪，毕竟那时候连anchor box都没有呢，说YOLOv1是anchor-free模型反而有点本末倒置了。

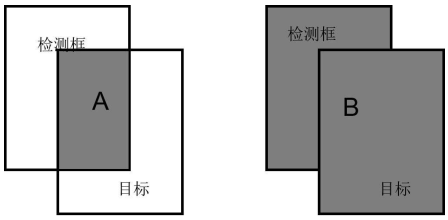
不过，这些都不重要，我们继续看点实在的东西。

我们已经确定了bbox的位置参数，接下来，再看**框的置信度** C ，或者说是**objectness。**它的作用其实就是用来表征此处是否有物体，和RPN 的概念是一样的，因此，学习标签可以用0和1，分别表示有物体额没有物体。但是，YOLO认为框的置信度也应该具备表征对bbox预测的定量评价的能力，因此，尽管对于没有物体处，其学习标签是0，但对于有物体的地方，则采用另一种做法：

具体来说，在训练过程中，对于被标记为**正样本**的 $(grid_x, grid_y)$ 处：

第1步：YOLOv1网络输出B个bbox；

第2步：计算这B个bbox与此处的真实bbox之间的**交并比（Intersection over Union，IoU），**得到B个IoU值；



$$IoU = \frac{A}{B}$$

交并比（Intersection over Union，IoU）

第3步：选择其中最大的IoU值来作为置信度 C 的学习目标；

第4步：同时，对于bbox的位置信息参数（即中心点和框的宽高），也只让这个IoU最大的那个bbox去反向传播，其他B-1个bbox就忽略了。可见，B个bbox之间是有“竞争”关系的。但这种竞争关系不是显式的，因为彼此之间没有进行建模约束。

举例来说，假设当前预测的B个矩形框和真实的矩形框的最大 $IoU = 0.28$ ，对应的预测bbox为 B_i ，那么网络就会将这个 0.28 作为置信度 C 的学习目标，同时只有 B_i 会去计算regression部分的loss，然后反向传播，其他的bbox就都被忽略了。

可见，倘若bbox都学得不好，那么算出的IoU也就都不高，网络预测的置信度学出来的也就偏低。但换个角度来看，这个置信度直接就能衡量定位的好坏。

让网络学习IoU，即所谓的IoU-Aware，这一思想还是在后来的ECCV的IoU-Net提出的呢~

另外，关于YOLOv1中只让IoU最高的那个bbox去回归，其他都忽略掉这一点，笔者认为是多余的，因为B个bbox是完全平权的，没有差异性，没有约束性，倒不如就令B=1，简单省事。

总结一下，网络的最后输出中，每个正样本 $(grid_x, \ grid_y)$ 位置处的每个bbox都包含如下参数：

$$(C_{box}, c_x, c_y, w, h)$$

最后再加上类别：

$$(C_1, C_2, \ldots, C_{20})$$

由这些参数，计算出相应矩形框的位置：

$$center_x = (grid_x + c_x) \times stride$$

$$center_y = (grid_y + c_y) \times stride$$

$$w_{box} = w \times w_{image}$$

$$h_{box} = h \times h_{image}$$

明确了学习目标见，制作训练标签也就清楚了。在制作标签时，遵循如下步骤：

1. 首先计算，中心点落入的网格位置：

$$grid_x = \lfloor \frac{center_x}{stride} \rfloor$$

$$grid_y = \lfloor \frac{center_y}{stride} \rfloor$$

2. 对于 $(gride_x,\ \ gride_y)$ 位置，我们认为此处有物体，因此 $Pr(objectness) = 1$，此处即为正样本，并计算量化误差与矩形框的宽和高：

$$c_x = \frac{center_x}{stride} - \lfloor \frac{center_x}{stride} \rfloor$$

$$c_y = \frac{center_y}{stride} - \lfloor \frac{center_y}{stride} \rfloor$$

$$w = \frac{w}{w_{image}}$$

$$h = \frac{h}{h_{image}}$$

3. 确定类别标签即可。

最后，我们再说一下损失函数的设计，直接上图：

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

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

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

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

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

图六. YOLO-v1中的损失函数

图中的第一行关于 c_x, c_y 的损失计算，原文中用的是 x, y 。第二行是关于 w, h 的损失计算。注意，这两行中都用一个 $\mathbb{1}_{ij}^{obj}$ 符号，这个符号表示正样本，即只有**正样本才会计算bbox 的损失和class损失**，而其他地方的损失则不考虑。这就会带来一个问题：

有物体的地方，我们让网络去学习，逼近它，而没有物体的地方，理论上应该让 c_x, c_y, w, h 都是零，但是这些没有物体的地方的并没有考虑进来，因此不会被学习到。那怎么办?

这就是**第三行**和**第四行**的作用。

在第三行中， $\hat{C}$ 是预测框与真实框计算的 IoU ，预测的越准， IoU 越趋近于 1 ，那么网络预测的 C 也就越趋近于 1 ，而没有物体的地方，也就是第四行，其中的 $\hat{C}$ 都是0（因为这些地方是不应该有框的，所以直接设为 0 即可），那么网络在这些地方预测的 C 也就会趋近于0，在测试的时候，置信度很低的地方我们也就不会考虑了，它输出怎样的bbox也就都无所谓了。

因此，框的置信度的本质就是一个有无物体的二分类，其作用就是判断此处是前景还是背景。

另外，很明显，一张图中，有物体的网格数目是小于没有物体的网格数目的，因此，为了平衡他们两个的损失，需要赋予不同的权重，正样本的置信度损失的权重给 1 ，没有物体的，即负样本的损失权重给 $\lambda_{noobj} = 0.5$ 。

最后的第五行，就是类别概率的损失计算，同样，我们也只考虑正样本的类别损失，其他地方不管。

从上面的损失，我们发现，都是用MSE来计算的，这是因为YOLO-v1关于框的五个参数和类别都是用的是线性函数（全连接层不加激活函数）来做的预测。至于为什么YOLOv1用线性函数来预测，而不是用 $softmax$ ，咱也不知道，也许是某种历史问题。

以上，便是有关于训练YOLO-v1的全部了。

那么，在测试的时候我们应该怎么做呢?

1.计算bbox和类别：

$$center_x = (grid_x + c_x) \times stride$$

$$center_y = (grid_y + c_y) \times stride$$

$$w_{box} = w \times w_{image}$$

$$h_{box} = h \times h_{image}$$

$$class = max(C_1, C_2, \ldots, C_{20})$$

2.计算每个边界框的得分：

在YOLOv1中，边界框的得分score=**该处的框的置信度即objectness**与**类别的置信度class**的乘积：

$$score = C_{box} \times Pr(class)$$

因为在训练阶段，只有正样本出的class预测才会被学习到，而负样本，也就是背景给出的class预测不会被学习，这就会导致在推理的时候，有物体的地方会有可靠的class预测输出，而没有物体的地方的class预测输出约等于瞎预测，毕竟没有被训练。但这个时候，由于objectness的作用就是判断是否有物体，因此，对于前景，objectness的值会很接近1，反之很接近0。那么，即使class瞎预测，但objectness只要给出接近0的值，那么这个地方的score也会很低，从而滤除了背景。

YOLO的精髓在一在于把背景和前景的各个类别的学习给解耦了。objectness分支就负责学习前景和背景，本质是个二分类，等价于Faster R-CNN中的RPN，而类别学习只学正样本的信息，标签里也没加进去背景标签，这就等于Faster R-CNN的第二阶段。而像SSD和RetinaNet，都是把背景作为一类标签加到了class里，把背景和前景的各个类别的学习耦合到了一块去，那自然要比YOLO学得麻烦一些。

*个人感觉，这才是YOLO为什么对Focal loss不敏感的原因，因为它用objectness分支把背景和前景的各个类别的学习解耦了。class分支只需要学正样本就好了，不需要像RetinaNet那样，单独的背景要和其他一大堆前景标签一块battle，把置信度拉到自己头上。**Objectness预测分支才是YOLO的灵魂！**如果没有这个分支，那和SSD、RetinaNet也就没差别了。*

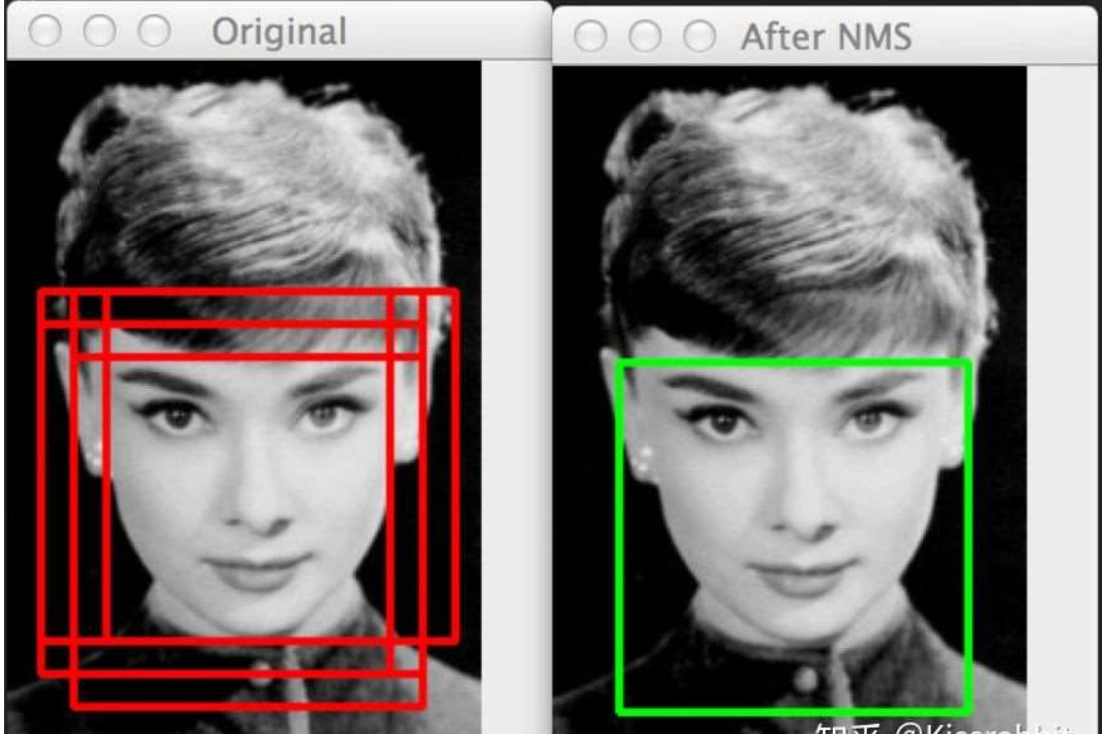
用这个得分去做后续的**非极大值抑制处理（NMS）**。最后保留下来的结果，就是网络的最终预测输出。

到此，关于YOLOv1，我们就全部讲完了~整体来看，YOLOv1还是非常简单的，无论是groundtruth制作，网络预测，都非常简单，这也是我为什么会把它作为本教程的重点示例来讲解。在下一章，我们将对YOLOv1做一次优化改进，来实现一个简单而更好的目标检测网络。

补充：

最后简单说一下什么是**非极大值抑制**吧。

事实上，如果我们把所有的预测结果都可视化出来，会发现有很多冗余，即多个box检测到了同一个物体，而我们对于每一个物体只需要一个框就够了。因此，我们有必要去剔除掉多余的结果。以下面的人间检测为例：



左图中，是未经过处理的，我们可以看到有多个红框检测到了同一张脸，为了提出掉多余的框，采用如下步骤：

1. 首先挑选出得分score最高的框；

2. 依次计算其他框与这个得分最高的框的 IoU ，超过给定 IoU 阈值的框舍掉。

3. 对每一类别都进行以上的操作，直到无框可剔除为止。

编辑于 2021-10-21 18:05

目标检测　计算机视觉　深度学习（Deep Learning）

文章被以下专栏收录

第二卷-基于YOLO的目标检测入门教程
本专栏将从零开始完整实现YOLOv1至v3模型

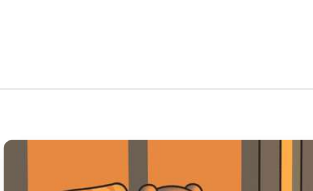
推荐阅读

1.3 YOLO入门教程：YOLOv1(3)-改进YOLOv1

Kissr...　发表于第二卷-基...

1.4 YOLO入门教程：YOLOv1(4)-搭建YOLOv1(上)

Kissr...　发表于第二卷-基...



1.1 YOLO入门教程：YOLOv1(1)-目标检测结构

Kissr...　发表

38 条评论 切换为时间排序

写下你的评论...

你杠就是你对 2021-10-20

您好，想请问训练的时候是只有含目标中心的cell会训练吗（就是不断CNN返回损失函数来调参）？还是所有cell都会一起训练，通过pr(objectness)来分是否有目标中心的。 还有就是预测的时候每个格子的bbox是放在一起筛选最终的那个吗？

1

知乎用户 (作者) 回复 你杠就是你对 2021-10-20

Objectness预测是全部cell都要参与计算损失，因为它要学会判断每个cell是否有中心点，也就是是否有物体。class box这俩只有包含了物体中心的cell（正样本）会去计算损失，

1

你杠就是你对 回复 知乎用户 (作者) 2021-10-21

那训练时，不含目标中心的cell预测的多个bbox是选哪一个来计算损失呢？也通过IOU吗？

赞

查看全部 7 条回复

小木头呀 2021-11-24

博主，没太搞懂这20个class具体概率大小在训练阶段和测试阶段是怎么得到的？

赞

知乎用户 2021-11-21

这么一种只训练一个proposal的方法是不是也可以理解为降低了一点计算量啊🤔

赞

zlzzy 2021-11-11

只能送博主俩字，牛皮啊

赞

北海 2021-10-28

博主，你的微信群我该怎么加啊，还有你的书啥时候出来啊，已经迫不及待地想买了！

赞

知乎用户 (作者) 回复 北海 2021-10-28

入群方式请私聊我~书遥遥无期

赞

aaabc 2021-10-24

您好 想请问一下预测之后 每个格子是只会留一个bbox吗？因为我看吴恩达老师的课里说 每个格子只能预测一个对象 所以才引入了anchor box

赞

知乎用户 (作者) 回复 aaabc 2021-10-26

对，虽然每个格子有B个框，但最后只会保留置信度最高的那个，也就是每个格子只有一个输出

赞

aaabc 回复 知乎用户 (作者) 2021-10-26

所以说最后筛选的过程应该是 每个单元格先根据score保留最高的一个 然后筛选掉单元格输出中低于阈值的 最后将剩下的做NMS吗?

赞

查看全部 6 条回复

量积质变 2021-10-21

感谢，写的很清楚，但是我有一个小问题。
文中说的Pr(objectness)=1 ，就是B个Bbox的confidence 都等于1，这个是要看那个的iou最高 就用哪个的bbox计算损失吗

赞

知乎用户 (作者) 回复 量积质变 2021-10-21

YOLOv1论文中给出了Pr(objectness)=1 是正样本的意思，和网路预测的objectness不是一个概念。实际上，YOLOv1会学习objectness，也就是每个bbox的confidence，这俩是一回事，我在文中的斜体字部分给出了修改后的score计算公式，里面的Pr(objectness)就是YOLOv1预测的bbox的confidence。

赞

未来可期 2021-10-16

置信度即objectness在训练和测试时值是不一样吗?
训练时：Pr(object)*IOU=IOU,有物体时
Pr(object)*IOU=0,没有物体时
而测试时：Pr(object)=1,有物体时
Pr(object)=0,没有物体时

赞

知乎用户 (作者) 回复 未来可期 2021-10-16

你的问题我明白。
其实，论文的表述是存在一定的歧义性的，甚至误导性。我尽可能地详细地解释一下，后续我会及时在文章中更新。

论文中所说的Pr(objectness)=1，以现在的技术角度来看，就是指“正样本”。
首先要清楚一点，YOLO一共有三个预测：objectness、class、bbox。其中，objectness是一个二分类，即有物体还是无物体，也就是“边界框的置信度”，对应loss函数中的那个“C”，没物体的标签显然就是0，而有物体的标签可以直接给1，也可以计算当前预测的bbox与gt之间的IoU作为有物体的标签，注意，这个IoU是objectness预测学习的标签，和YOLO没关系。class就是类别预测，只有正样本处的grid cell才会被训练（也就是Pr(objectness)=1的地方，注意，这个Pr(objectness)=1就是指正样本的地方，和IoU没关，和YOLO没关，只和label有关，因为gt box的中心点落在哪个grid，哪个grid就是正样本，也就是Pr(objectness)=1）。bbox也是同理。

在测试阶段，YOLO一共会输出三个预测，是否有物体的objectness预测、class预测和bbox预测。首先，我们计算score=objectness*class作为每个边界框的得分score，论文中写的Pr(class)*IoU其实就是这个，这个IoU就是objectness预测，因为objectness在训练过程中的正样本标签就是IoU，所以可以认为objectness隐含了IoU的概念，但本质就是有无物体的预测。

不要忘记，objectness在训练过程中已经学会了判断每个grid cell是否有物体，那么，显然，对于有物体的地方，objectness会很接近1，class的准确预测也应该会很接近1，没有物体的地方，objectness会很接近0，class则瞎预测，毕竟它在训练阶段只计算有物体部分的class损失，不过很明显objectness会起到主导作用，即使class瞎预测了一个很接近1的值，objectness知道这里没有物体，因此会给个接近0的值，则score就很接近0了。

希望能解决你的问题~

2

未来可期 回复 知乎用户 (作者) 2021-10-16

博主太好了吧，感觉看到了希望，谢谢博主倾心答复

赞

展开其他 2 条回复

未来可期 2021-10-16

我的意思是在测试时应该不考虑IOU吗，是因为预测的时候没有GT box,所以无法算IOU吗
麻烦看一下，小白入门

赞

臭咸鱼 2021-08-19

写得很赞！

赞

是小戴啊 2021-07-23

保姆级教程，请收下我的膝盖。不愧是哈工大的男人，强者如斯

赞

所剩无几 2021-05-10

赞，讲的很清晰

赞

小蜗牛 2021-05-08

很多细节都有说到，必须赞，给力

赞

知乎用户 (作者) 回复 小蜗牛 2021-05-08

感谢

赞

知乎用户 2021-04-21

牛！！！！

赞

Moretime 04-05

这个非极大值抑制不怎么好，只有一个物体一种类别，体现不出非极大值抑制的能力，一个物体一个类别，直接选最大的得分框不就完事了

赞

知乎用户 (作者) 回复 Moretime 04-05

那如果图中有多个物体，多个类别的时候呢?

赞

dcdc3 03-29

博主，请问最后两层全连接的方式，会不会存在某个cell的bbox参数完全没被训练过，然后测试的时候这个cell又恰好有目标，这时候预测能准么

赞

理心炼丹 02-10

感觉有问题 边框回归时候从公式看 并不是只有IOU最大的进行回归 而是两个都要进行回归 因为是求和符号

赞