YOLO v1 算法

1. 目标检测算法分类

（1） 基于 Region Proposal 的 R-CNN 系算法（R-CNN，Fast R-CNN, Faster R-CNN），它们是 two-stage 的，需要先使用启发式方法（selective search）或者 CNN 网络（RPN）产生 Region Proposal，然后再在 Region Proposal 上做分类与回归。

（2） 是 Yolo，SSD 这类 one-stage 算法，其仅仅使用一个 CNN 网络直接预测不同目标的类别与位置

第一类方法准确度高一些，但是速度慢，但是第二类算法是速度快，但是准确性要低一些

2. 滑动窗口与 CNN

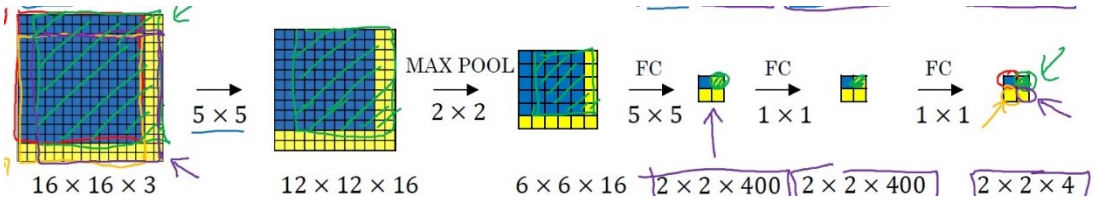
采用滑动窗口的目标检测算法思路非常简单，它将检测问题转化为了图像分类问题。基本原理是采用不同大小和比例（宽高比）的窗口在整张图片上以一定的步长进行滑动，然后对这些窗口对应的区域做图像分类，这样就可以实现对整张图片的检测。DPM 就是采用这种思路。

缺点是不知道要检测目标的大小是什么规模，所以你要设置不同大小和比例的窗口去滑动，而且还要选取合适的步长，这样会产生很多的子区域，并且都要经过分类器去做预测，这需要很大的计算量。

解决思路之一是减少要分类的子区域，这就是 R-CNN 的一个改进策略，其采用了 selective search 方法来找到最有可能包含目标的子区域（Region Proposal），其实可以看成采用启发式方法过滤掉很多子区域，这会提升效率

如果使用的是 CNN 分类器，那么滑动窗口是非常耗时的。但是结合卷积运算的特点，我们可以使用 CNN 实现更高效的滑动窗口方法。这里要介绍的是一种全卷积的方法，简单来说就是网络中用卷积层代替了全连接层，如图所示。输入图片大小是 16x16，经过一系列卷积操作，提取了 2x2 的特征图，但是这个 2x2 的图上每个元素都是和原图是一一对应的，如图上蓝色的格子对应蓝色的区域，这不就是相当于在原图上做大小为 14x14 的窗口滑动，且步长为 2，共产生 4 个字区域。最终输出的通道数为 4，可以看成 4 个类别的预测概率值，这样一次 CNN 计算就可以实现窗口滑动的所有子区域的分类预测。这其实是 overfeat 算法的思路。之所以 CNN 可以实现这样的效果是因为卷积操作的特性，就是图片的空间位置信息的不变性，尽管卷积过程中图片大小减少，但是位置对应关系还

是保存的。说点题外话，这个思路也被 R-CNN 借鉴，从而诞生了 Fast R-cNN 算法。



上面尽管可以减少滑动窗口的计算量，但是只是针对一个固定大小与步长的窗口，这是远远不够的。Yolo 算法很好的解决了这个问题，它不再是窗口滑动了，而是直接将原始图片分割成互不重合的小方块，然后通过卷积最后生产这样大小的特征图，基于上面的分析，可以认为特征图的每个元素也是对应原始图片的一个小方块，然后用每个元素来可以预测那些中心点在该小方格内的目标，这就是 Yolo 算法的朴素思想。

3. YOLO 设计理念

整体来看，Yolo 算法采用一个单独的 CNN 模型实现 end-to-end 的目标检测，整个系统如图所示：首先将输入图片 resize 到 448x448，然后送入 CNN 网络，最后处理网络预测结果得到检测的目标

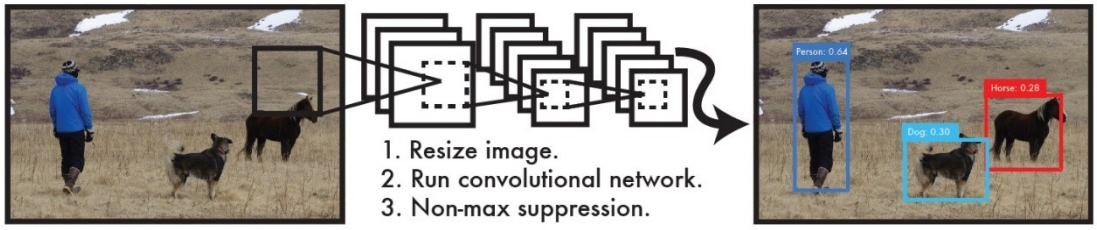


fig 3.0

具体来说，Yolo 的 CNN 网络将输入的图片分割成$S \times S$网格，然后每个单元格负责去检测那些中心点落在该格子内的目标，如图 6 所示，可以看到狗这个目标的中心落在左下角一个单元格内，那么该单元格负责预测这个狗。每个单元格会预测 B 个边界框（bounding box）以及边界框的置信度（confidence score）。所谓置信度其实包含两个方面：

（1）是这个边界框含有目标的可能性大小

（2）是这个边界框的准确度。

前者记为$Pr(\text{object})$，当该边界框是背景时（即不包含目标），此时$Pr(\text{object}) = 0$。而当该边界框包含目标时$Pr(\text{object}) = 1$。边界框的准确度可以用预测框与实际框（ground truth）的 IOU（intersection over union，交并比）来表征，记为IOU_{pred}^{ruth}。因此置信度可以定义为$Pr(\text{ object })*\text{IOU}_{\text{pred}}^{\text{truth}}$；很多人可能

将 Yolo 的置信度看成边界框是否含有目标的概率，但是其实它是两个因子的乘积，预测框的准确度也反映在里面。

边界框的大小与位置可以用 4 个值来表征：(x,y,w,h)，其中(x,y)是边界框的中心坐标，而w和 h 是边界框的宽与高。还有一点要注意，中心坐标的预测值（x,y）是相对于**每个单元格左上角坐标点的偏移值**，并且单位是相对于单元格大小的，单元格的坐标定义如图所示。而边界框的 w 和 h 预测值是相对于整个图片的宽与高的比例，这样理论上 4 个元素的大小应该在 [0，1] 范围。这样，每个边界框的预测值实际上包含 5 个元素：(x,y,w,h,c)，其中前 4 个表征边界框的大小与位置，而最后一个值是置信度。

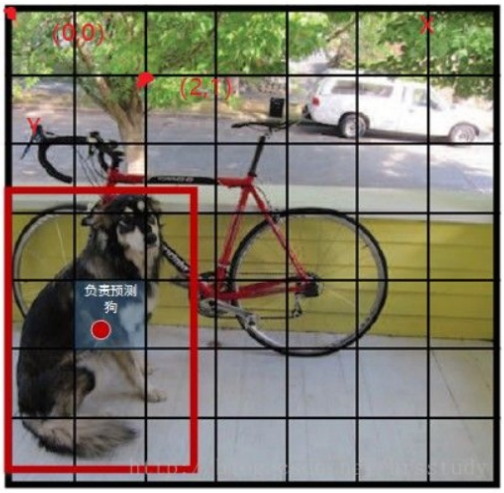


图 3.1 网格划分

对于每一个单元格其还要给出预测出 C 个类别概率值，其表征的是由该单元格负责预测的边界框其目标属于各个类别的概率。但是这些概率值其实是在各个边界框置信度下的条件概率，即 $Pr(\text{class}_i|\text{ object })$ 。

值得注意的是，不管一个单元格预测多少个边界框，其只预测一组类别概率值，这是 Yolo 算法的一个缺点，在后来的改进版本中，Yolo9000 是把类别概率预测值与边界框是绑定在一起的。同时，我们可以计算出各个边界框类别置信度（class-specific confidence scores）：

$$Pr(\text{class}_i|\text{ object })*Pr(\text{object})*IOU_{\text{pred}}^{\text{truth}} = Pr(\text{class}_i)*\text{IOU}_{\text{pred}}^{\text{truth}}$$

边界框类别置信度表征的是该边界框中目标属于各个类别的可能性大小以及边界框匹配目标的好坏。等式左边第一项就是每个网格预测的类别信息，第二三项就是每个 bounding box 预测的 confidence。这个乘积即 encode 了预测的 box 属于某一类的概率，也有该 box 准确度的信息。得到每个 box 的 class-specific confidence

score 以后，设置阈值，滤掉得分低的 boxes，对保留的 boxes 进行 NMS 处理，就得到最终的检测结果

小结：每个单元格需要预测（B*5+C）个值。如果将输入图片划分为 S*S 网格，那么最终预测值为 S*S*(B*5+C) 大小的张量。对于 PASCAL VOC 数据，其共有 20 个类别，如果使用 S=7,B=2 ，那么最终的预测结果就是 7*7*30 大小的张量。

4. YOLO 网络设计

Yolo 采用卷积网络来提取特征，然后使用全连接层来得到预测值。网络结构参考 GooLeNet 模型，包含 24 个卷积层和 2 个全连接层。对于卷积层，主要使用 1x1 卷积来做 channle reduction，然后紧跟 3x3 卷积。对于卷积层和全连接层，采用 Leaky ReLU 激活函数：$\max(x, 0.1x)$。最后一层采用线性激活函数。

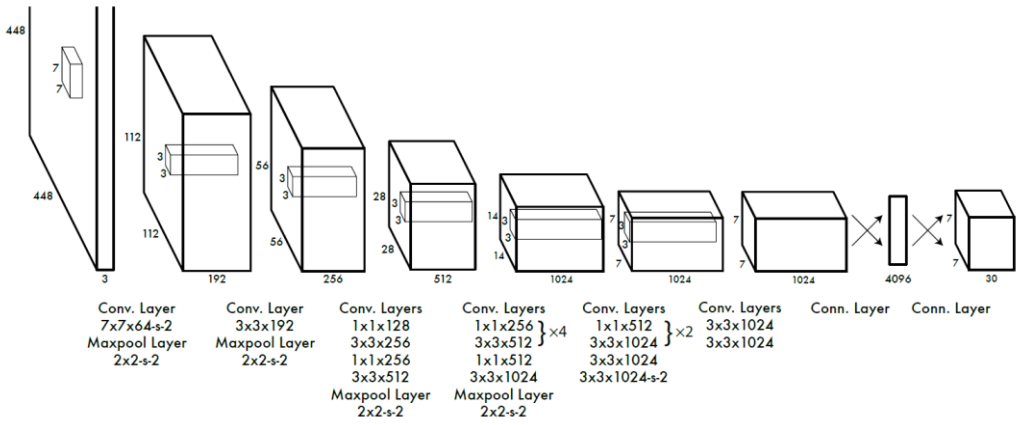


图 4.1 YOLO 网络结构

网络的最后输出为 7*7*30 大小的张量。这个张量所代表的具体含义如图 4.2 所示。对于每一个单元格，前 20 个元素是类别概率值，然后 2 个元素是边界框置信度，两者相乘可以得到类别置信度，最后 8 个元素是边界框的(x, y, w, h)。对于边界框为什么把置信度 c 和 (x,y,w,h) 都分开排列，而不是按照(x,y,w,h,c)这样排列是为了计算方便，因为实际上这 30 个元素都是对应一个单元格，其排列是可以任意的。这里来解释一下，首先网络的预测值是一个二维张量 P，其 shape 为 [batch,7*7*30] 。采用切片，那么$P_{[:,0:7*7*20]}$就是类别概率部分，而$P[:, 7*7*20:7*7*(20+2)]$是置信度部分，最后剩余部分$P_{[:,7*7*(20+2):]}$是边界框的预测结果。这样，提取每个部分是非常方便的，这会方面后面的训练及预测时的计算。

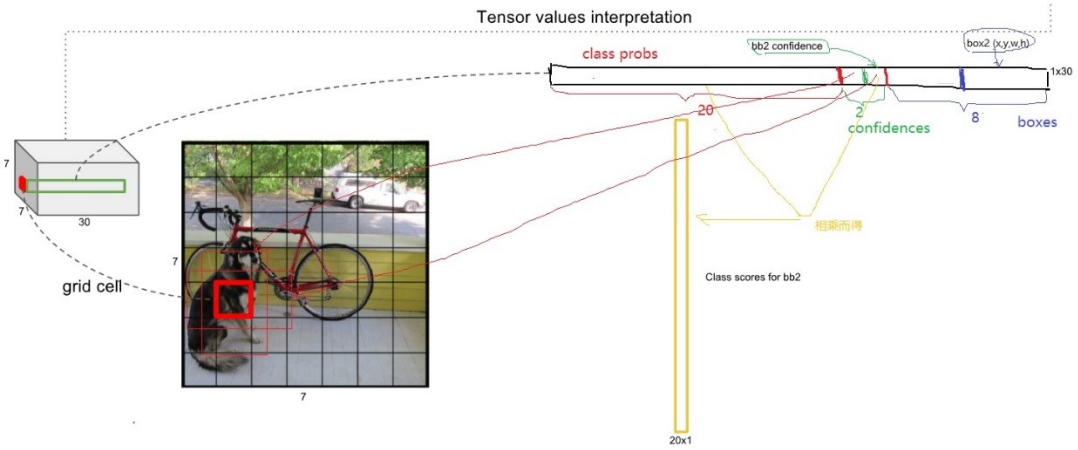


图 4.2 张量含义

在训练之前，先在 ImageNet 上进行了预训练，其预训练的分类模型采用图 4.1 中前 20 个卷积层，然后添加一个 average-pool 层和全连接层。预训练之后，在预训练得到的 20 层卷积层之上加上随机初始化的 4 个卷积层和 2 个全连接层。由于检测任务一般需要更高清的图片，所以将网络的输入从 224x224 增加到了 448x448。整个网络的流程如下图所示：

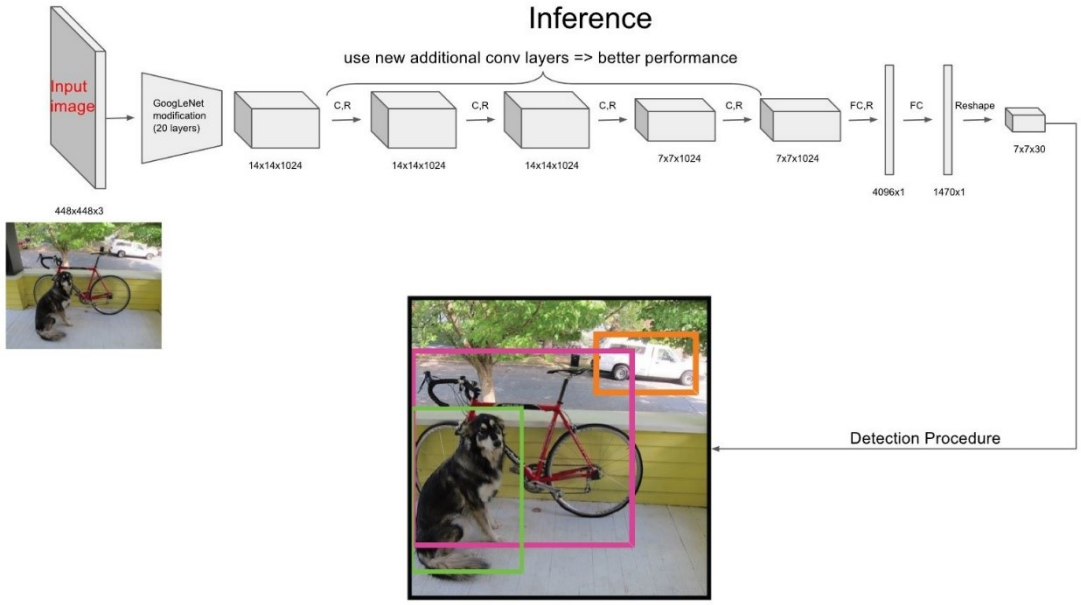


图 4.3 YOLO 网络流程

5. 损失函数分析

YOLO 算法将目标检测看成回归问题，所以采用的是均方差损失函数

每个 grid 有 30 维，这 30 维中，8 维是回归 box 的坐标，2 维是 box 的 confidence，还有 20 维是类别。其中坐标的 x,y 用对应网格的 offset 归一化到 0-1 之间，w,h 用图像的 width 和 height 归一化到 0-1 之间。

损失函数的设计目标就是让坐标（x,y,w,h），confidence，classification 这个三个方面达到很好的平衡。简单的全部采用了 sum-squared error loss 来做这件事会有以下不足：

1）8 维的 localization error 和 20 维的 classification error 同等重要显然是不合理的

2）如果一个网格中没有 object（一幅图中这种网格很多），那么就会将这些网格中的 box 的 confidence push 到 0，相比于较少的有 object 的网格，这种做法是 overpowering 的，这会导致网络不稳定甚至发散,这个解决方案为：

a. 更重视 8 维的坐标预测，给这些损失前面赋予更大的 loss weight，记为 λ_{coord} 在 pascal VOC 训练中取 5
b. 对没有 object 的 box 的 confidence loss，赋予小的 loss weight，记为 λ_{noobj} 在 pascal VOC 训练中取 0.5
c. 有 object 的 box 的 confidence loss 和类别的 loss 的 loss weight 正常取 1

3）对不同大小的 box 预测中，相比于大 box 预测偏一点，小 box 预测偏一点肯定更不能被忍受的。而 sum-square error loss 中对同样的偏移 loss 是一样，解决方法是：**就是将 box 的 width 和 height 取平方根代替原本的 height 和 width**。这个参考图 5.1 很容易理解，小 box 的横轴值较小，发生偏移时，反应到 y 轴上相比大 box 要大

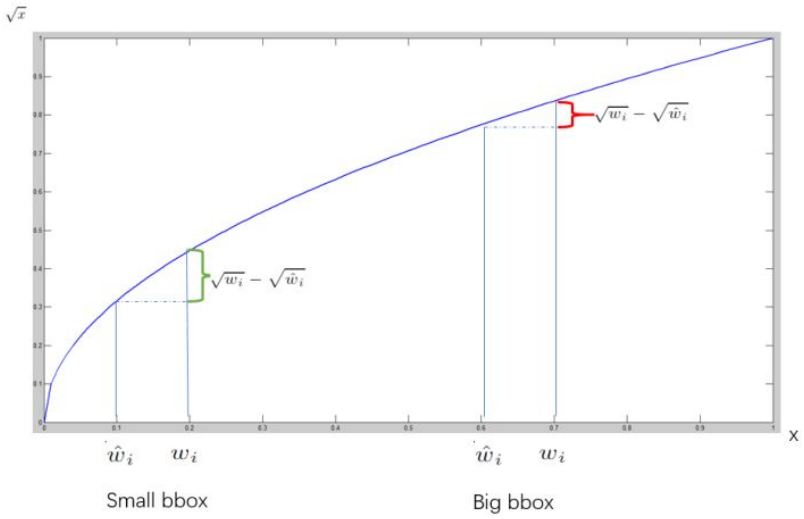
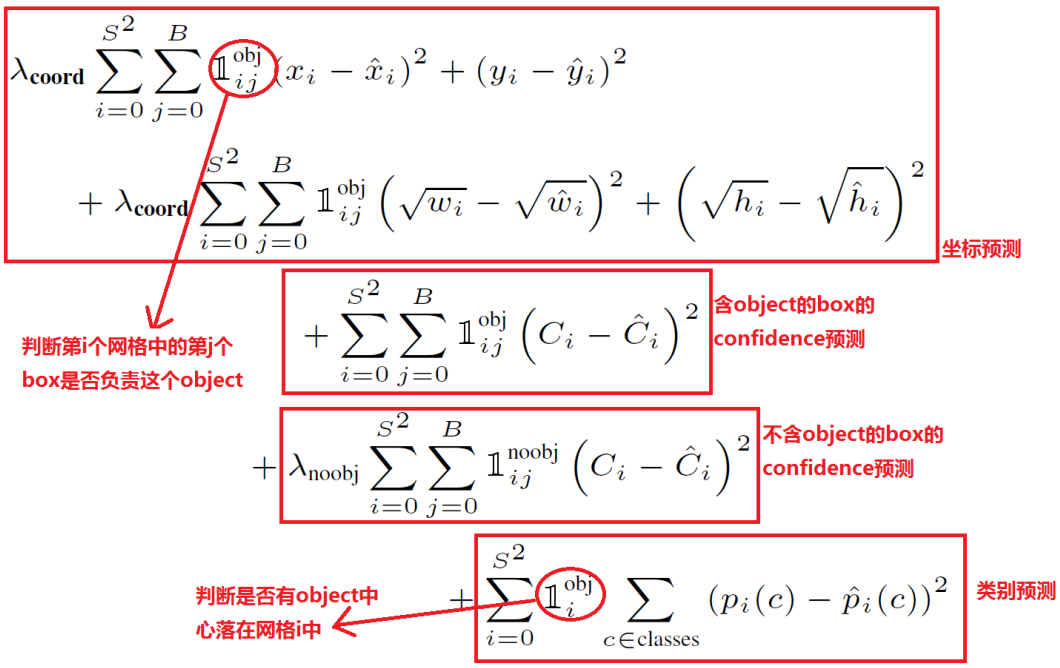


图 5.1

4）一个网格预测多个 box，希望的是每个 box predictor 专门负责预测某个 object，具体做法就是看当前预测的 box 中与 ground truth box 哪个 IoU 大，就哪个负责。这种做法称作 box predictor 的 specialization

整个的损失函数如下：



6. 非极大值抑制算法

对于 Bounding Box 的列表 B 及其对应的置信度 S,采用下面的计算方式

选择具有最大 score 的检测框 M,将其从 B 集合中移除并加入到最终的检测结果 D 中.通常将 B 中剩余检测框中与 M 的 IoU 大于阈值 Nt 的框从 B 中移除.重复这个过程,直到 B 为空.

非极大值抑制的方法是：假设有 6 个矩形框，根据分类器的类别分类概率做排序，假设从小到大属于检测目标的概率分别为 A、B、C、D、E、F。

(1)从最大概率矩形框 F 开始，分别判断 A~E 与 F 的重叠度 IOU 是否大于某个设定的阈值（常用的阈值是 0.3～0.5）

(2)假设 B、D 与 F 的重叠度超过阈值，那么就扔掉 B、D；并标记第一个矩形框 F，是我们保留下来的。

(3)从剩下的矩形框 A、C、E 中，选择概率最大的 E，然后判断 E 与 A、C 的重叠度，重叠度大于一定的阈值，那么就扔掉；并标记 E 是我们保留下来的第二个矩形框。

就这样一直重复，找到所有被保留下来的矩形框。

算法的输入：算法对一幅图产生的所有的候选框，以及每个框对应的 score

算法的输出：正确的候选框组（输入候选框组的一个子集）

7. 网络预测

不考虑 batch，只预测一张输入图片。根据前面的分析，最终的网络输出是 [7，7，30] ，但是我们可以将其分割成三个部分：类别概率部分为 [7，7，20] ，

置信度部分为 [7，7，2] ，而边界框部分为 [7，7，2，4] （对于这部分不要忘记根据原始图片计算出其真实值）。然后将前两项相乘（矩阵 [7，7，20] 乘以 [7，7，2] 可以各补一个维度来完成 [7，7，1，20] x[7,7,2,1]）可以得到类别置信度值为 [7,7,2,20] [注 1]，这里总共预测了 7*7*2=98 个边界框。

两种策略来得到检测框的结果：

1）对于每个预测框根据类别置信度选取置信度最大的那个类别作为其预测标签，经过这层处理得到各个预测框的预测类别及对应的置信度值，其大小都是 [7*7*2] 。一般情况下，会设置置信度阈值，就是将置信度小于该阈值的 box 过滤掉，所以经过这层处理，剩余的是置信度比较高的预测框。再对这些预测框使用 NMS 算法，最后留下来的就是检测结果。一个值得注意的点是 NMS 是对所有预测框一视同仁，还是区分每个类别，分别使用 NMS。Ng 在 deeplearning.ai 中讲应该区分每个类别分别使用 NMS，但是看了很多实现，其实还是同等对待所有的框，我觉得可能是不同类别的目标出现在相同位置这种概率很低吧。

2）先使用 NMS，然后再确定各个 box 的类别。对于 98 个 boxes，首先将小于置信度阈值的值归 0，然后分类别地对置信度值采用 NMS，这里 NMS 处理结果不是剔除，而是将其置信度值归为 0。最后才是确定各个 box 的类别，当其置信度值不为 0 时才做出检测结果输出。这个策略不是很直接，但是貌似 Yolo 源码就是这样做的。Yolo 论文里面说 NMS 算法对 Yolo 的性能是影响很大的，所以可能这种策略对 Yolo 更好。

名词解释&注释：

IOU：两个边界框的交集部分除以它们的并集,当两个边界框完全重合时，其 IOU=1，完全不重合时，IOU=0

NMS:非极大值抑制

注 1：此处乘法为点乘，点乘要求两矩阵维度相同，python 中有广播机制，故此处矩阵维度不同

参考博文

https://zhuanlan.zhihu.com/p/32525231
https://blog.csdn.net/c20081052/article/details/80236015

orangerfun@foxmail.com