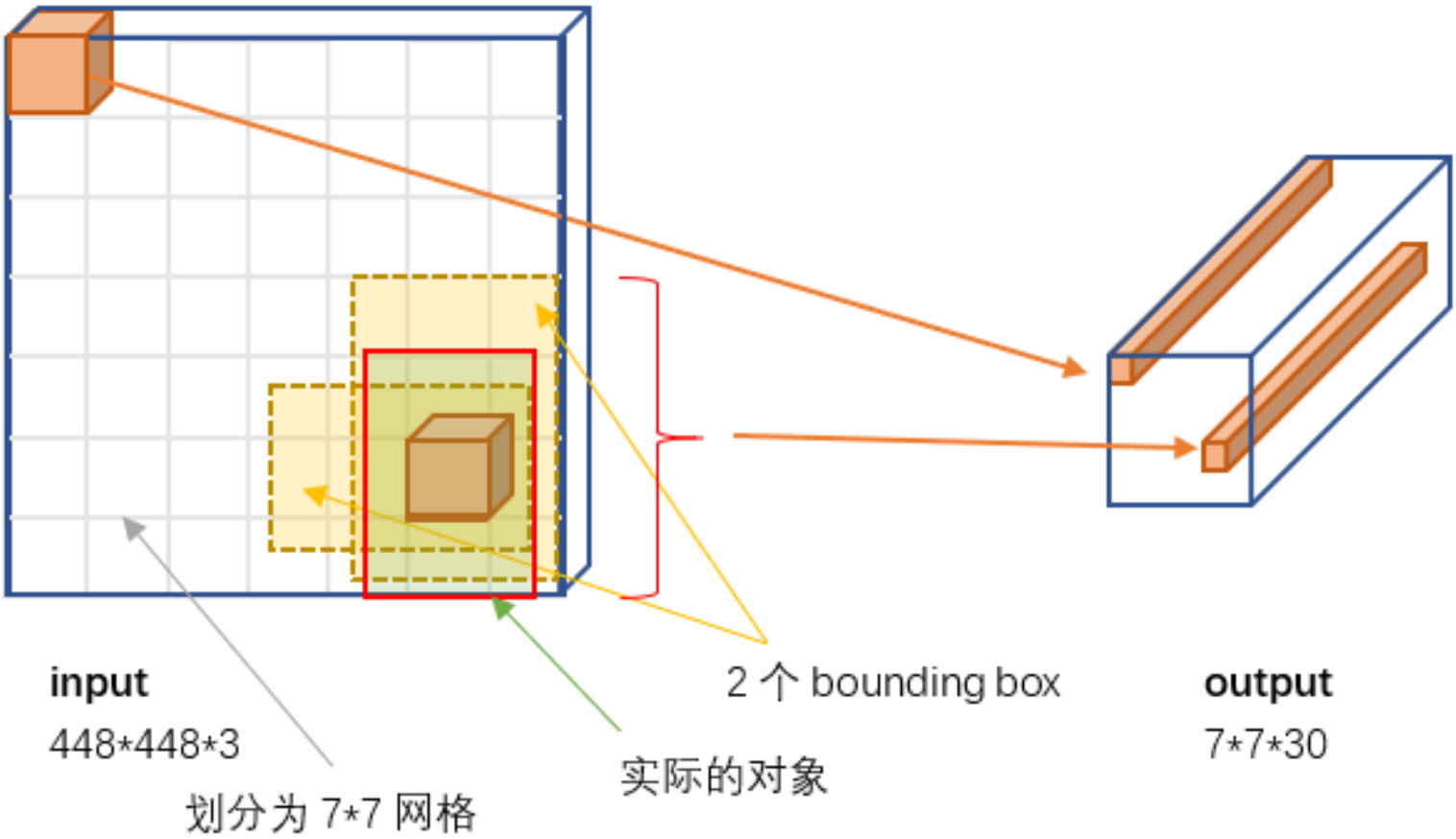


<机器爱学习>YOLO v1深入理解

X-猪
机器爱学习

954 人赞同了该文章

YOLO（You Only Look Once）是一种基于深度神经网络的对象识别和定位算法，其最大的特点是运行速度很快，可以用于实时系统。

现在YOLO已经发展到v3版本，不过新版本也是在原有版本

基础上不断改进演化的，所以本文先分析YOLO v1版本。

关于 YOLOv2/YOLO9000 和 YOLOv3 的分析理解请移步 YOLO v2 / YOLO 9000 和 YOLO v3深入理解。

对象识别和定位

输入一张图片，要求输出其中所包含的对象，以及每个对象的位置（包含该对象的矩形框）。

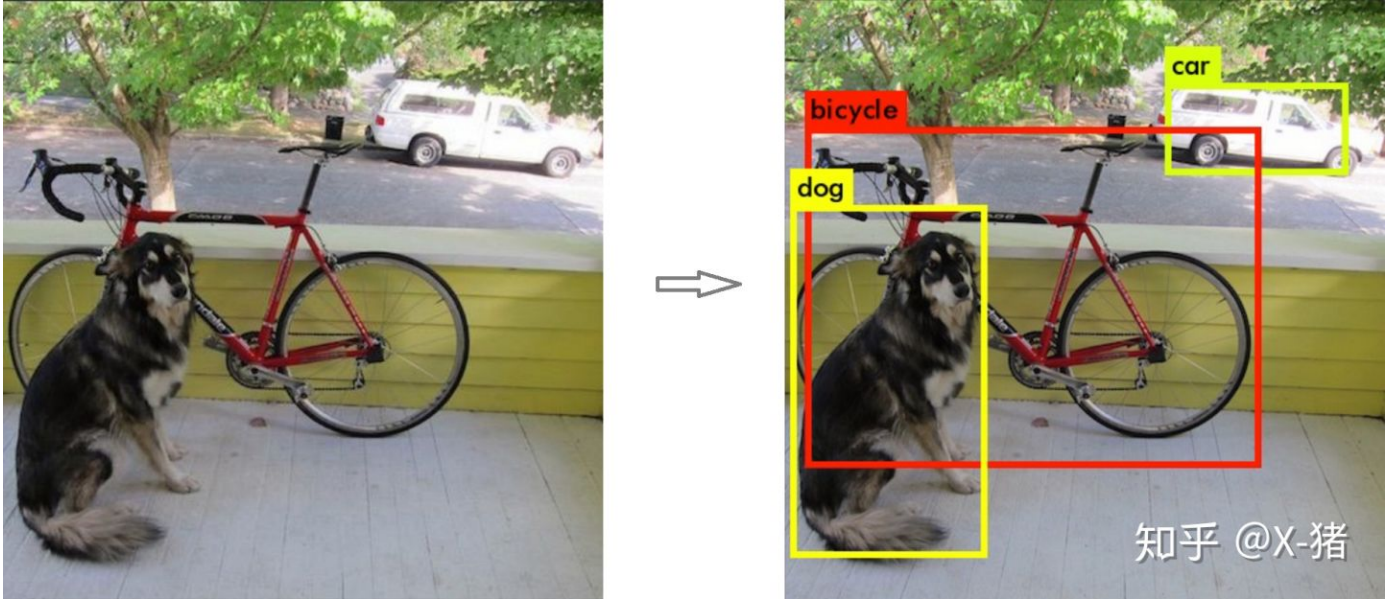


图1 对象识别和定位

对象识别和定位，可以看成两个任务：找到图片中某个存在对象的区域，然后识别出该区域中具体是哪个对象。

对象识别这件事（一张图片仅包含一个对象，且基本占据图片的整个范围），最近几年基于CNN卷积神经网络的各种方法已经能达到不错的效果了。所以主要需要解决的问题是，对象在哪里。

最简单的想法，就是遍历图片中所有可能的位置，地毯式搜索不同大小，不同宽高比，不同位置的每个区域，逐一检测其中是否存在某个对象，挑选其中概率最大的结果作为输出。显然这种方法效率太低。

RCNN/Fast RCNN/Faster RCNN

RCNN开创性的提出了候选区(Region Proposals)的方法，先从图片中搜索出一些可能存在对象的候选区（Selective Search），大概2000个左右，然后对每个候选区进行对象识别。大幅提升了对象识别和定位的效率。

不过RCNN的速度依然很慢，其处理一张图片大概需要49秒。因此又有了后续的Fast RCNN 和Faster RCNN，针对 RCNN的神经网络结构和候选区的算法不断改进，Faster RCNN已经可以达到一张图片约0.2秒的处理速度。下图来自 R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms

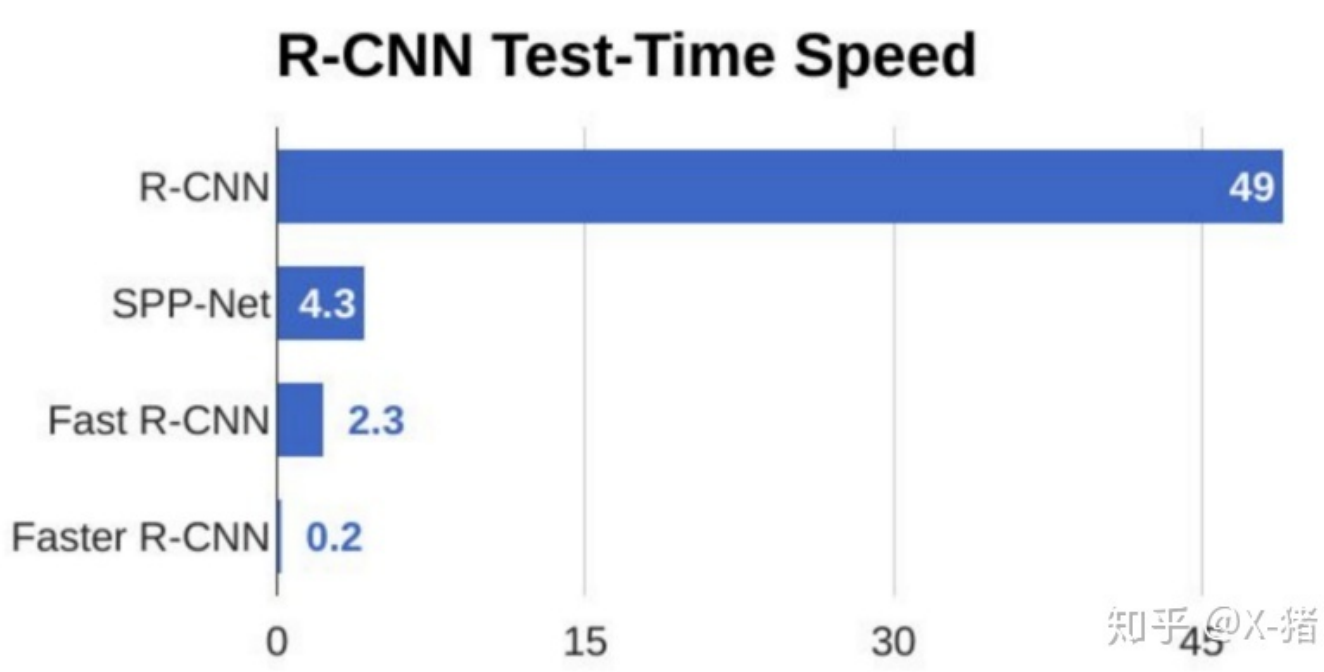


图2 RCNN系列处理速度

但总体来说，RCNN系列依然是两阶段处理模式：先提出候选区，再识别候选区中的对象。

YOLO

YOLO意思是You Only Look Once，创造性的将候选区和对象识别这两个阶段合二为一，看一眼图片（不用看两眼哦）就能知道有哪些对象以及它们的位置。

实际上，YOLO并没有真正去掉候选区，而是采用了预定义的候选区（准确点说应该是预测区，因为并不是Faster RCNN所采用的Anchor）。也就是将图片划分为 7*7=49 个网格（grid），每个网格允许预测出2个边框（bounding box，包含某个对象的矩形框），总共 49*2=98 个 bounding box。可以理解为98个候选区，它们很粗略的覆盖了图片的整个区域。

```
RCNN：我们先来研究一下图片，嗯，这些位置很可能存在一些对象，你们对这些位置再检测一下看到底是哪些对
YOLO：我们把图片大致分成98个区域，每个区域看下有没有对象存在，以及具体位置在哪里。
RCNN：你这么简单粗暴真的没问题吗？
YOLO：当然没有.....咳，其实是有一点点问题的，准确率要低一点，但是我非常快！快！快！
RCNN：为什么你用那么粗略的候选区，最后也能得到还不错的bounding box呢？
YOLO：你不是用过边框回归吗？我拿来用用怎么不行了。
```

RCNN虽然会找到一些候选区，但毕竟只是候选，等真正识别出其中的对象以后，还要对候选区进行微调，使之更接近真实的bounding box。这个过程就是**边框回归**：将候选区bounding box调整到更接近真实的bounding box。

既然反正最后都是要调整的，干嘛还要先费劲去寻找候选区呢，大致有个区域范围就行了，所以YOLO就这么干了。

不过话说回来，边框回归为啥能起作用，我觉得本质上是因为 分类信息 中已经包含了 位置信息。就像你看到主子的脸和身体，就能推测出耳朵和屁股的位置。

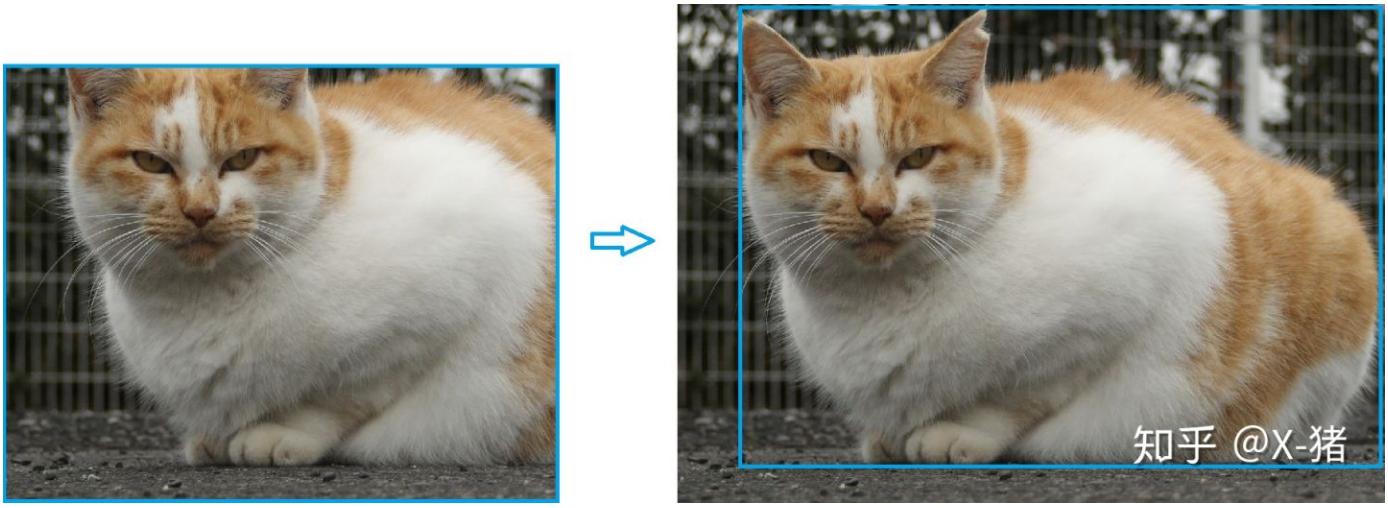


图3 边框调整

格局：吴军博士新书 格局越大 成就越大

京东

¥ 44.30 去购买 ›

下面具体看下YOLO的实现方案。

1）结构

去掉候选区这个步骤以后，YOLO的结构非常简单，就是单纯的卷积、池化最后加了两层全连接。单看网络结构的话，和普通的CNN对象分类网络几乎没有本质的区别，最大的差异是最后输出层用线性函数做激活函数，因为需要预测bounding box的位置（数值型），而不仅仅是对象的概率。所以粗略来说，YOLO的整个结构就是输入图片经过神经网络的变换得到一个输出的张量，如下图所示。

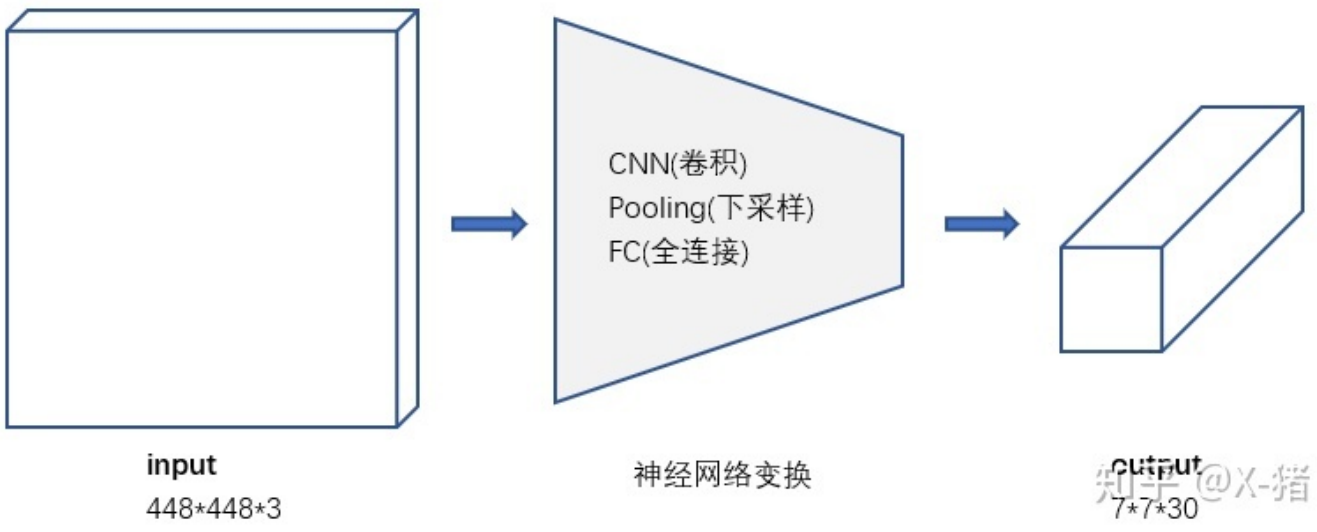


图4 输入 - 输出

因为只是一些常规的神经网络结构，所以，理解YOLO的设计的时候，重要的是理解输入和输出的映射关系.

2）输入和输出的映射关系

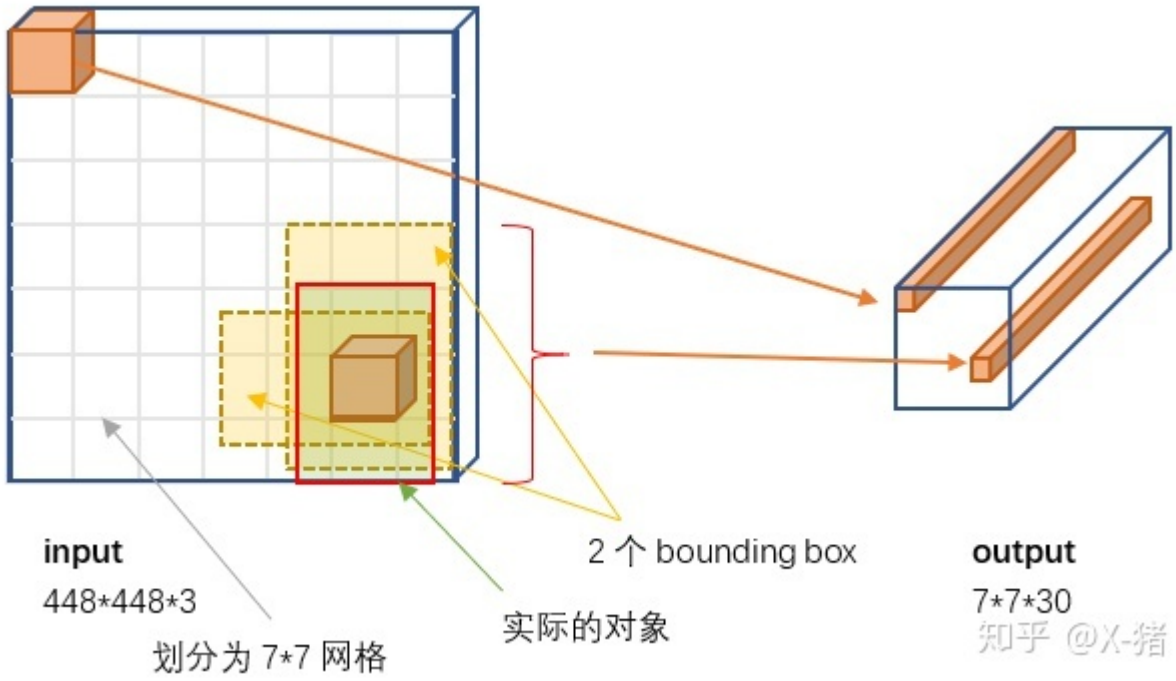


图5 输入 - 输出

3）输入

参考图5，输入就是原始图像，唯一的要求是缩放到448*448的大小。主要是因为YOLO的网络中，卷积层最后接了两个全连接层，全连接层是要求固定大小的向量作为输入，所以倒推回去也就要求原始图像有固定的尺寸。那么YOLO设计的尺寸就是448*448。

4）输出

输出是一个 7*7*30 的张量（tensor）。

4.1）7*7网格

根据YOLO的设计，输入图像被划分为 7*7 的网格（grid），输出张量中的 7*7 就对应着输入图像的 7*7 网格。或者我们把 7*7*30 的张量看作 7*7=49个30维的向量，也就是输入图像中的每个网格对应输出一个30维的向量。参考上面图5，比如输入图像左上角的网格对应到输出张量中左上角的向量。

要注意的是，并不是说仅仅网格内的信息被映射到一个30维向量。经过神经网络对输入图像信息的提取和变换，网格周边的信息也会被识别和整理，最后编码到那个30维向量中。

4.2）30维向量

具体来看每个网格对应的30维向量中包含了哪些信息。

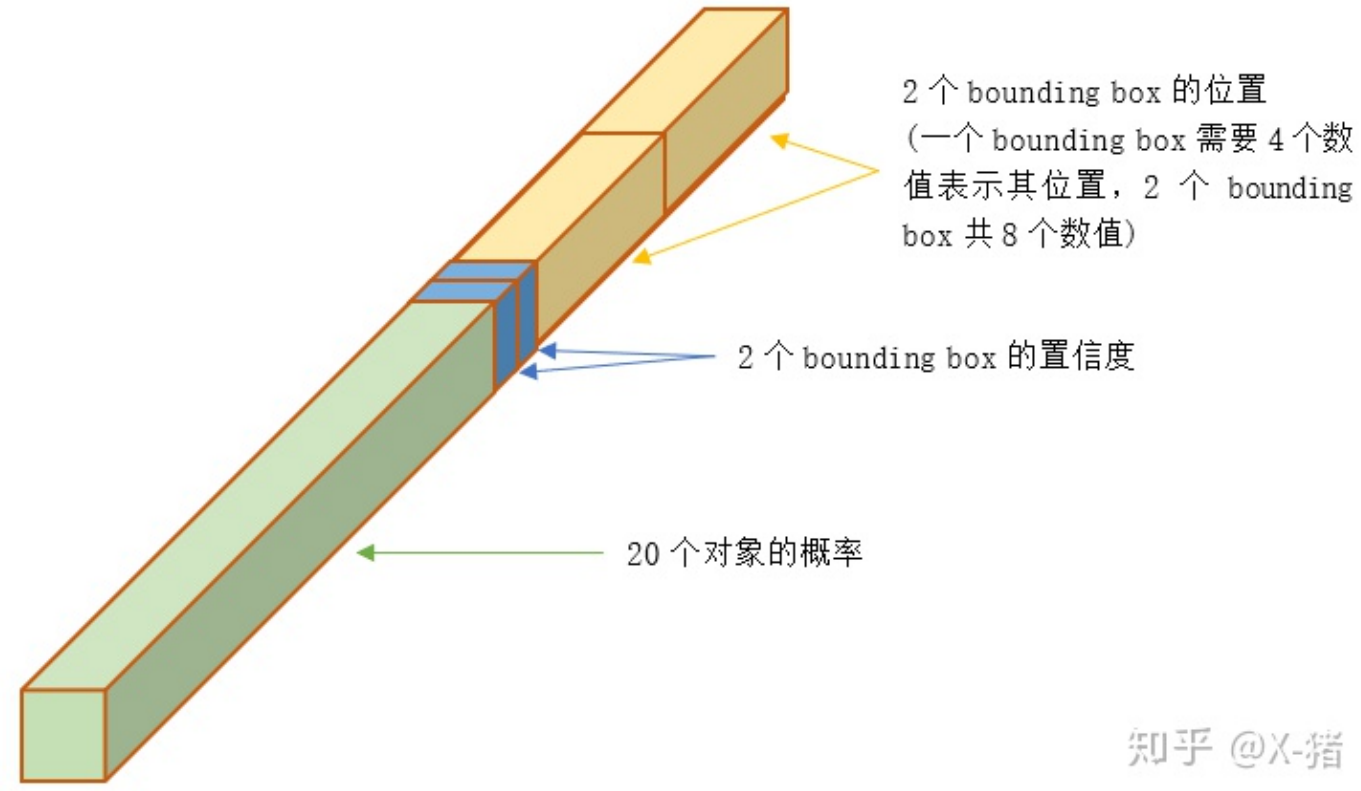


图6 30维输出向量

① 20个对象分类的概率

因为YOLO支持识别20种不同的对象（人、鸟、猫、汽车、椅子等），所以这里有20个值表示该网格位置存在任一种对象的概率。可以记为 $P(C_1|Object),.....,P(C_i|Object),.....P(C_{20}|Object)$ ，之所以写成条件概率，意思是如果该网格存在一个对象Object，那么它是 C_i 的概率是 $P(C_i|Object)$ 。（记不清条件概率的同学可以参考一下 理解贝叶斯定理）

② 2个bounding box的位置

每个bounding box需要4个数值来表示其位置，(Center_x,Center_y,width,height)，即(bounding box的中心点的x坐标，y坐标，bounding box的宽度，高度)，2个bounding box共需要8个数值来表示其位置。

③ 2个bounding box的置信度

bounding box的置信度 = 该bounding box内存在对象的概率 * 该bounding box与该对象实际bounding box的IOU

用公式来表示就是

$$Confidence = Pr(Object) * IOU_{pred}^{truth}$$

$Pr(Object)$ 是bounding box内存在对象的概率，区别于上面第①点的 $P(C_i|Object)$ 。Pr(Object)并不管是哪个对象，它体现的是 有或没有 对象的概率。第①点中的 $P(C_i|Object)$ 意思是假设已经有一个对象在网格中了，这个对象具体是哪一个。

IOU_{pred}^{truth} 是 bounding box 与 对象真实bounding box 的IOU（Intersection over Union，交并比）。要注意的是，现在讨论的30维向量中的bounding box是YOLO网络的输出，也就是预测的bounding box。所以 IOU_{pred}^{truth} 体现了预测的bounding box与真实bounding box的接近程度。

还要说明的是，虽然有时说"预测"的bounding box，但这个IOU是在训练阶段计算的。等到了测试阶段（Inference），这时并不知道真实对象在哪里，只能完全依赖于网络的输出，这时已经不需要（也无法）计算IOU了。

综合来说，一个bounding box的置信度Confidence意味着它 是否包含对象且位置准确的程度。置信度高表示这里存在一个对象且位置比较准确，置信度低表示可能没有对象 或者 即便有对象也存在较大的位置偏差。

简单解释一下IOU。下图来自Andrew Ng的深度学习课程，IOU=交集部分面积/并集部分面积，2个box完全重合时IOU=1，不相交时IOU=0。

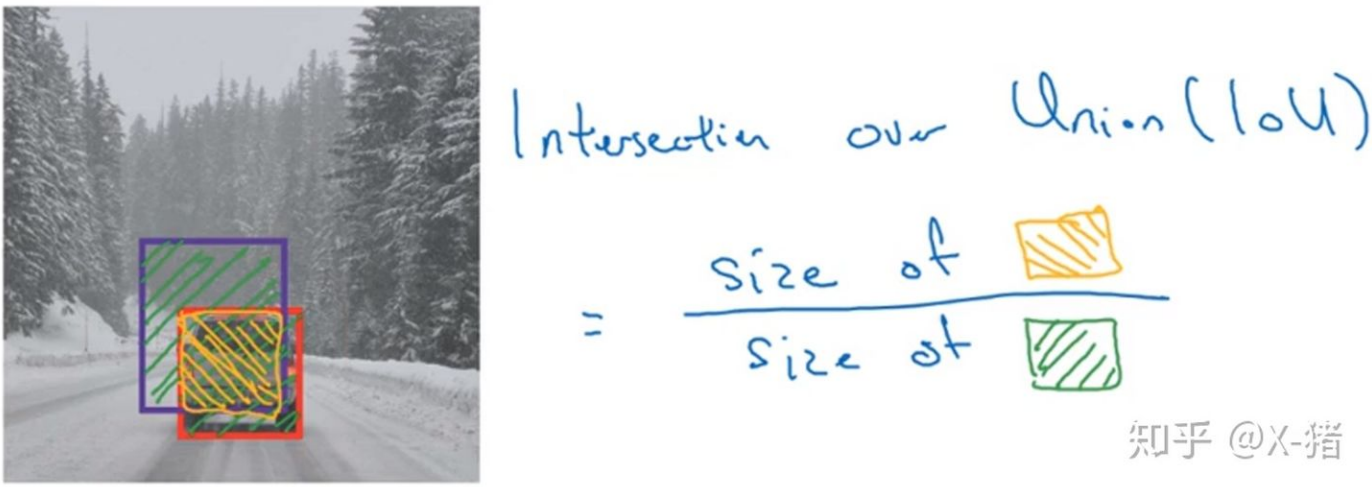


图7 IOU

总的来说，30维向量 = 20个对象的概率 + 2个bounding box * 4个坐标 + 2个bounding box的置信度

4.3）讨论

① 一张图片最多可以检测出49个对象

每个30维向量中只有一组（20个）对象分类的概率，也就只能预测出一个对象。所以输出的7*7=49个 30维向量，最多表示出49个对象。

② 总共有 49*2=98 个候选区（bounding box）

每个30维向量中有2组bounding box，所以总共是98个候选区。

③ YOLO的bounding box并不是Faster RCNN的Anchor

Faster RCNN等一些算法采用每个grid中手工设置n个Anchor（先验框，预先设置好位置的bounding box）的设计，每个Anchor有不同的大小和宽高比。YOLO的bounding box看起来很像一个grid中2个Anchor，但它们不是。YOLO并没有预先设置2个bounding box的大小和形状，也没有对每个bounding box分别输出一个对象的预测。它的意思仅仅是对一个对象预测出2个bounding box，选择预测得相对比较准的那个。

这里采用2个bounding box，有点不完全算监督算法，而是像进化算法。如果是监督算法，我们需要**事先**根据样本就能给出一个正确的bounding box作为回归的目标。但YOLO的2个bounding box事先并不知道会在什么位置，只有经过前向计算，网络会输出2个bounding box，这两个bounding box与样本中对象实际的bounding box计算IOU。这时才能确定，IOU值大的那个bounding box，作为负责预测该对象的bounding box。

训练开始阶段，网络预测的bounding box可能都是乱来的，但总是选择IOU相对好一些的那个，随着训练的进行，每个bounding box会逐渐擅长对某些情况的预测（可能是对象大小、宽高比、不同类型的对象等）。所以，这是一种进化或者非监督学习的思想。

另外论文中经常提到**responsible**。比如：Our system divides the input image into an S*S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. 这个 responsible 有点让人疑惑，对预测"负责"是啥意思。其实没啥特别意思，就是一个Object只由一个grid来进行预测，不要多个grid都抢着预测同一个Object。更具体一点说，就是在设置训练样本的时候，样本中的每个Object归属到且仅归属到一个grid，即便有时Object跨越了几个grid，也仅指定其中一个。具体就是计算出该Object的bounding box的中心位置，这个中心位置落在哪个grid，该grid对应的输出向量中该对象的类别概率是1（该gird负责预测该对象），所有其它grid对该Object的预测概率设为0（不负责预测该对象）。

还有：YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. 同样，虽然一个grid中会产生2个bounding box，但我们会选择其中一个作为预测结果，另一个会被忽略。下面构造训练样本的部分会看的更清楚。

④ 可以调整网格数量、bounding box数量

7*7网格，每个网格2个bounding box，对448*448输入图像来说覆盖粒度有点粗。我们也可以设置更多的网格以及更多的bounding box。设网格数量为 S*S，每个网格产生B个边框，网络支持识别C个不同的对象。这时，输出的向量长度为：

$$C + B * (4 + 1)$$

整个输出的tensor就是：

$$S * S * (C + B * (4 + 1))$$

YOLO选择的参数是 7*7网格，2个bounding box，20种对象，因此 输出向量长度 = 20 + 2 * (4+1) = 30。整个输出的tensor就是 7*7*30。

因为网格和bounding box设置的比较稀疏，所以这个版本的YOLO训练出来后预测的准确率和召回率都不是很理想，后续的v2、v3版本还会改进。当然，因为其速度能够满足实时处理的要求，所以对工业界还是挺有吸引力的。

5）训练样本构造

作为监督学习，我们需要先构造好训练样本，才能让模型从中学习。

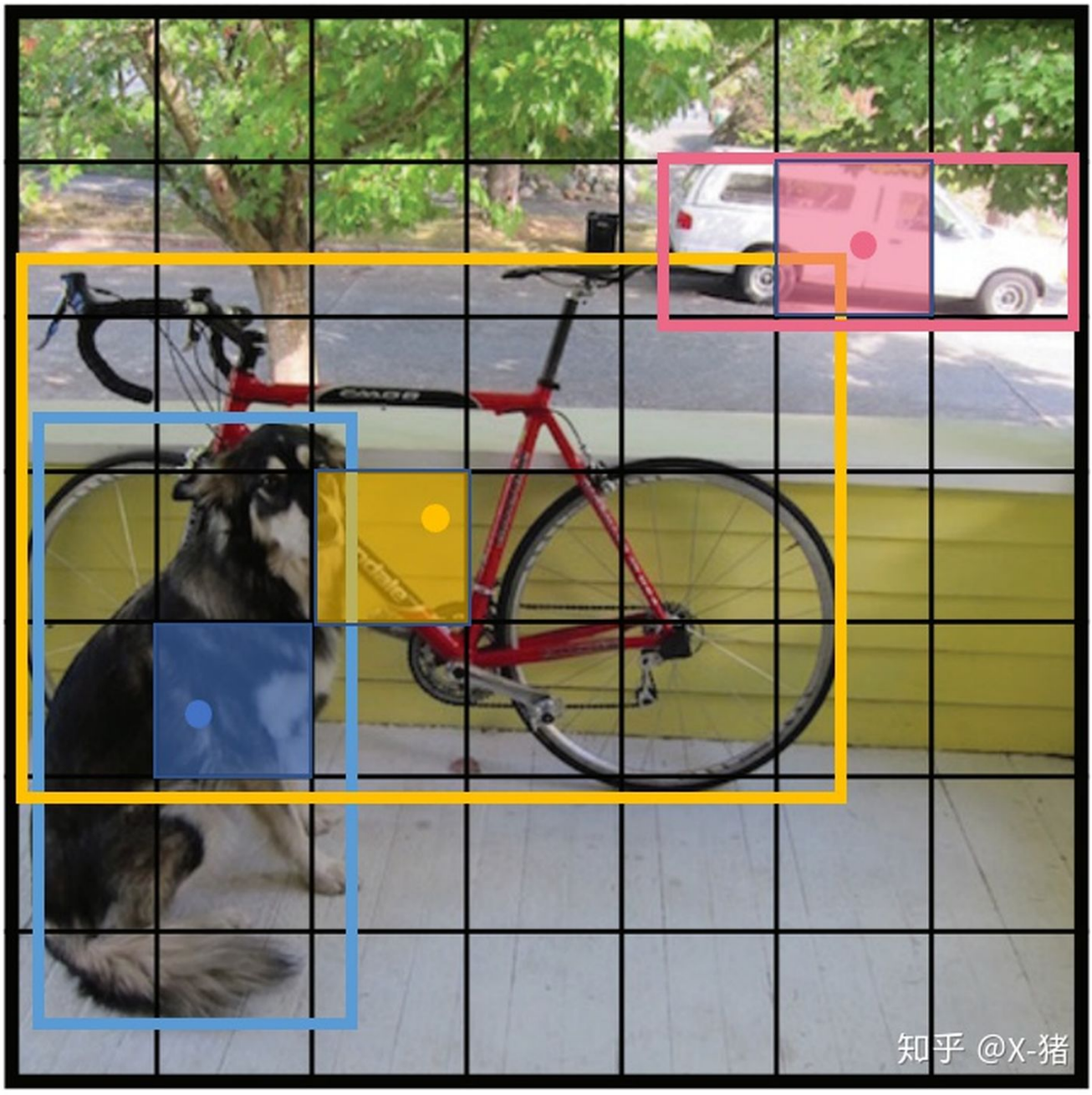


图8 输入样本图片

对于一张输入图片，其对应输出的7*7*30张量（也就是通常监督学习所说的标签y或者label）应该填写什么数据呢。

首先，输出的 7*7维度 对应于输入的 7*7 网格。 然后具体看下30维向量的填写（请对照上面图6）。

① 20个对象分类的概率

对于输入图像中的每个对象，先找到其中心点。比如图8中的自行车，其中心点在黄色圆点位置，中心点落在黄色网格内，所以这个黄色网格对应的30维向量中，自行车的概率是1，其它对象的概率是0。所有其它48个网格的30维向量中，该自行车的概率都是0。这就是所谓的"中心点所在的网格对预测该对象负责"。狗和汽车的分类概率也是同样的方法填写。

② 2个bounding box的位置

训练样本的bounding box位置应该填写对象实际的bounding box，但一个对象对应了2个bounding box，该填哪一个呢？上面讨论过，需要根据网络输出的bounding box与对象实际bounding box的IOU来选择，所以要在训练过程中动态决定到底填哪一个bounding box。参考下面第③点。

③ 2个bounding box的置信度

上面讨论过置信度公式

$$Confidence = Pr(Object) * IOU_{pred}^{truth}$$

IOU_{pred}^{truth} 可以直接计算出来，就是用网络输出的2个bounding box与对象真实bounding box一起计算出IOU。

然后看2个bounding box的IOU，哪个比较大（更接近对象实际的bounding box），就由哪个bounding box来负责预测该对象是否存在，即该bounding box的 $Pr(Object) = 1$ ，同时对象真实bounding box的位置也就填入该bounding box。另一个不负责预测的bounding box的 $Pr(Object) = 0$ 。

总的来说就是，与对象实际bounding box最接近的那个bounding box，其 $Confidence = IOU_{pred}^{truth}$ ，该网格的其它bounding box的 $Confidence = 0$ 。

举个例子，比如上图中自行车的中心点位于4行3列网格中，所以输出tensor中4行3列位置的30维向量如下图所示。

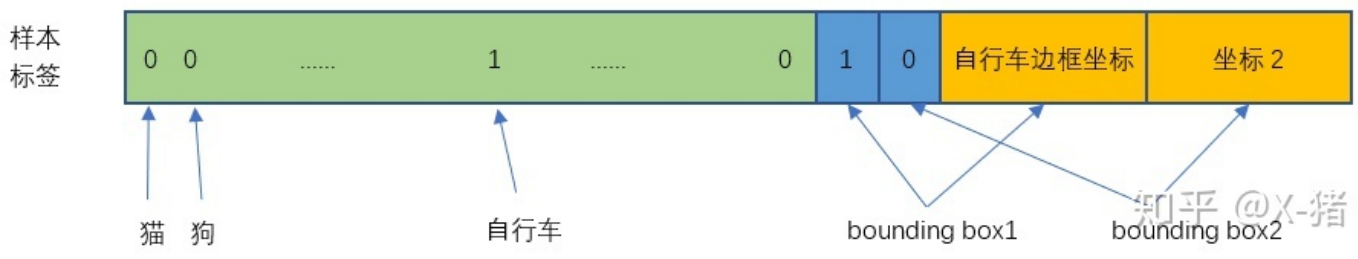


图9 训练样本的一个30维向量

翻译成人话就是：4行3列网格位置有一辆自行车，它的中心点在这个网格内，它的位置边框是bounding box1所填写的自行车实际边框。

注意，图中将自行车的位置放在bounding box1，但实际上是在训练过程中等网络输出以后，比较两个bounding box与自行车实际位置的IOU，自行车的位置（实际bounding box）放置在IOU比较大的那个bounding box（图中假设是bounding box1），且该bounding box的置信度设为1。

6）损失函数

损失就是网络实际输出值与样本标签值之间的偏差。

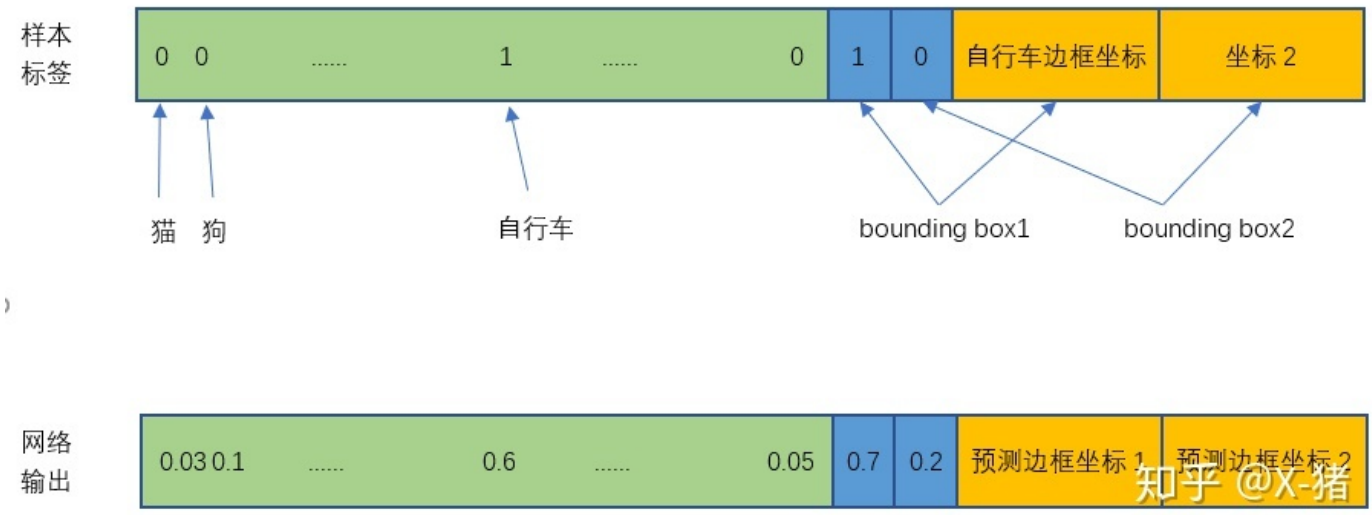


图10 样本标签与网络实际输出

YOLO给出的损失函数如下

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

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

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$ 置信度误差(边框内有对象)

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$ 置信度误差(边框内无对象)

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

知乎 @X-猪

图11 损失函数

公式中

1_i^{obj} 意思是网格i中存在对象。

1_{ij}^{obj} 意思是网格i的第j个bounding box中存在对象。

1_{ij}^{noobj} 意思是网格i的第j个bounding box中不存在对象。

总的来说，就是用网络输出与样本标签的各项内容的误差平方和作为一个样本的整体误差。 损失函数中的几个项是与输出的30维向量中的内容相对应的。

① 对象分类的误差

公式第5行，注意 1_i^{obj} 意味着存在对象的网格才计入误差。

② bounding box的位置误差

公式第1行和第2行。

a）都带有 1_{ij}^{obj} 意味着只有"负责"（IOU比较大）预测的那个bounding box的数据才会计入误差。

b）第2行宽度和高度先取了平方根，因为如果直接取差值的话，大的对象对差值的敏感度较低，小的对象对差值的敏感度较高，所以取平方根可以降低这种敏感度的差异，使得较大的对象和较小的对象在尺寸误差上有相似的权重。

c）乘以 λ_{coord} 调节bounding box位置误差的权重（相对分类误差和置信度误差）。YOLO设置 $\lambda_{coord}=5$ ，即调高位置误差的权重。

③ bounding box的置信度误差

公式第3行和第4行。

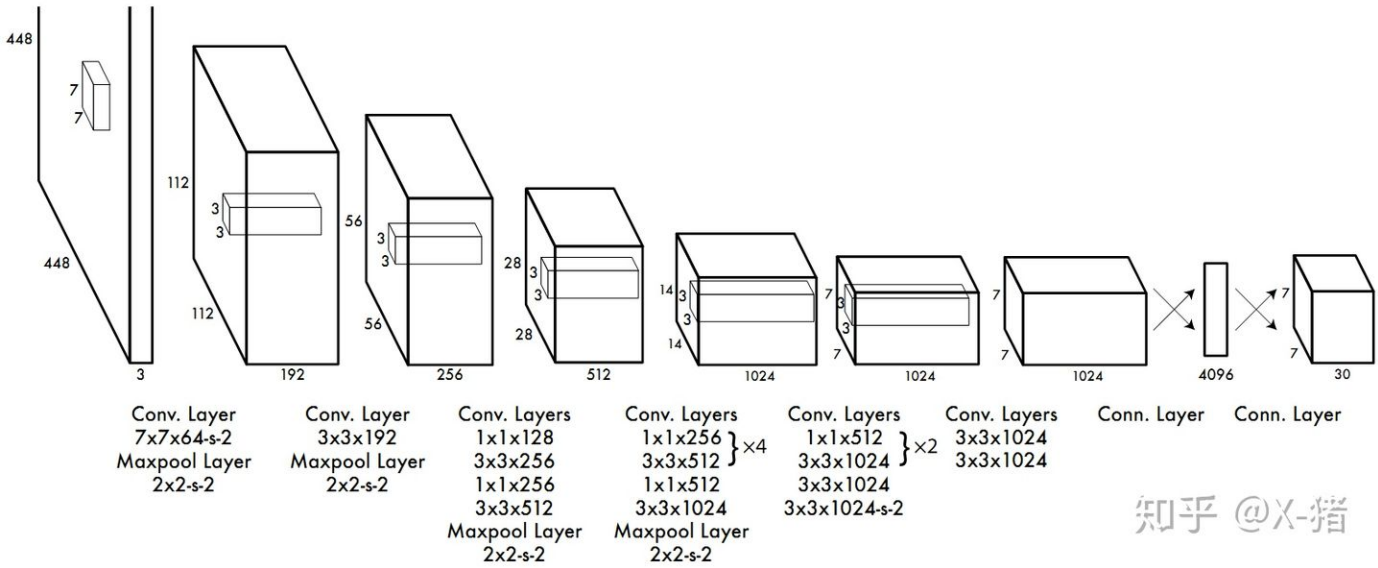
a）第3行是存在对象的bounding box的置信度误差。带有 1_{ij}^{obj} 意味着只有"负责"（IOU比较大）预测的那个bounding box的置信度才会计入误差。

b）第4行是不存在对象的bounding box的置信度误差。因为不存在对象的bounding box应该老老实实的说"我这里没有对象"，也就是输出尽量低的置信度。如果它不恰当的输出较高的置信度，会与真正"负责"该对象预测的那个bounding box产生混淆。其实就像对象分类一样，正确的对象概率最好是1，所有其它对象的概率最好是0。

c）第4行会乘以 λ_{noobj} 调节不存在对象的bounding box的置信度的权重（相对其它误差）。YOLO设置 $\lambda_{noobj}=0.5$ ，即调低不存在对象的bounding box的置信度误差的权重。

7）训练

YOLO先使用ImageNet数据集对前20层卷积网络进行预训练，然后使用完整的网络，在PASCAL VOC数据集上进行对象识别和定位的训练和预测。YOLO的网络结构如下图所示：



YOLO的最后一层采用线性激活函数，其它层都是Leaky ReLU。训练中采用了drop out和数据增强（data augmentation）来防止过拟合。更多细节请参考原论文。

8）预测（inference）

训练好的YOLO网络，输入一张图片，将输出一个 7*7*30 的张量（tensor）来表示图片中所有网格包含的对象（概率）以及该对象可能的2个位置（bounding box）和可信程度（置信度）。 为

了从中提取出最有可能的那些对象和位置，YOLO采用NMS（Non-maximal suppression，非极大值抑制）算法。

9）NMS（非极大值抑制）

NMS方法并不复杂，其核心思想是：选择得分最高的作为输出，与该输出重叠的去掉，不断重复这一过程直到所有备选处理完。

YOLO的NMS计算方法如下。

网络输出的7*7*30的张量，在每一个网格中，对象 C_i 位于第j个bounding box的得分：

$$Score_{ij} = P(C_i|Object) * Confidence_j$$

它代表着某个对象 C_i 存在于第j个bounding box的可能性。

每个网格有：20个对象的概率*2个bounding box的置信度，共40个得分（候选对象）。49个网格共1960个得分。Andrew Ng建议每种对象分别进行NMS，那么每种对象有 1960/20=98 个得分。

NMS步骤如下：

1）设置一个Score的阈值，低于该阈值的候选对象排除掉（将该Score设为0）

2）遍历每一个对象类别

2.1）遍历该对象的98个得分

2.1.1）找到Score最大的那个对象及其bounding box，添加到输出列表

2.1.2）对每个Score不为0的候选对象，计算其与上面2.1.1输出对象的bounding box的IOU

2.1.3）根据预先设置的IOU阈值，所有高于该阈值（重叠度较高）的候选对象排除掉（将Score设为0）

2.1.4）如果所有bounding box要么在输出列表中，要么Score=0，则该对象类别的NMS完成，返回步骤2处理下一种对象

3）输出列表即为预测的对象

10）小结

YOLO以速度见长，处理速度可以达到45fps，其快速版本（网络较小）甚至可以达到155fps。这得益于其识别和定位合二为一的网络设计，而且这种统一的设计也使得训练和预测可以端到端的进行，非常简便。

不足之处是小对象检测效果不太好（尤其是一些聚集在一起的小对象），对边框的预测准确度不是很高，总体预测精度略低于Fast RCNN。主要是因为网格设置比较稀疏，而且每个网格只预测两

个边框，另外Pooling层会丢失一些细节信息，对定位存在影响。 更多细节请参考原论文。

最后，如果你竟然坚持看到这里，觉得还有所帮助的话，请点个赞：）๑๑

参考

You Only Look Once: Unified, Real-Time Object Detection

Andrew Ng的深度学习工程师 - 04卷积神经网络

图解YOLO

你真的读懂yolo了吗?

目标检测|YOLO原理与实现

编辑于 2019-12-19

深度学习（Deep Learning） 目标检测 计算机视觉