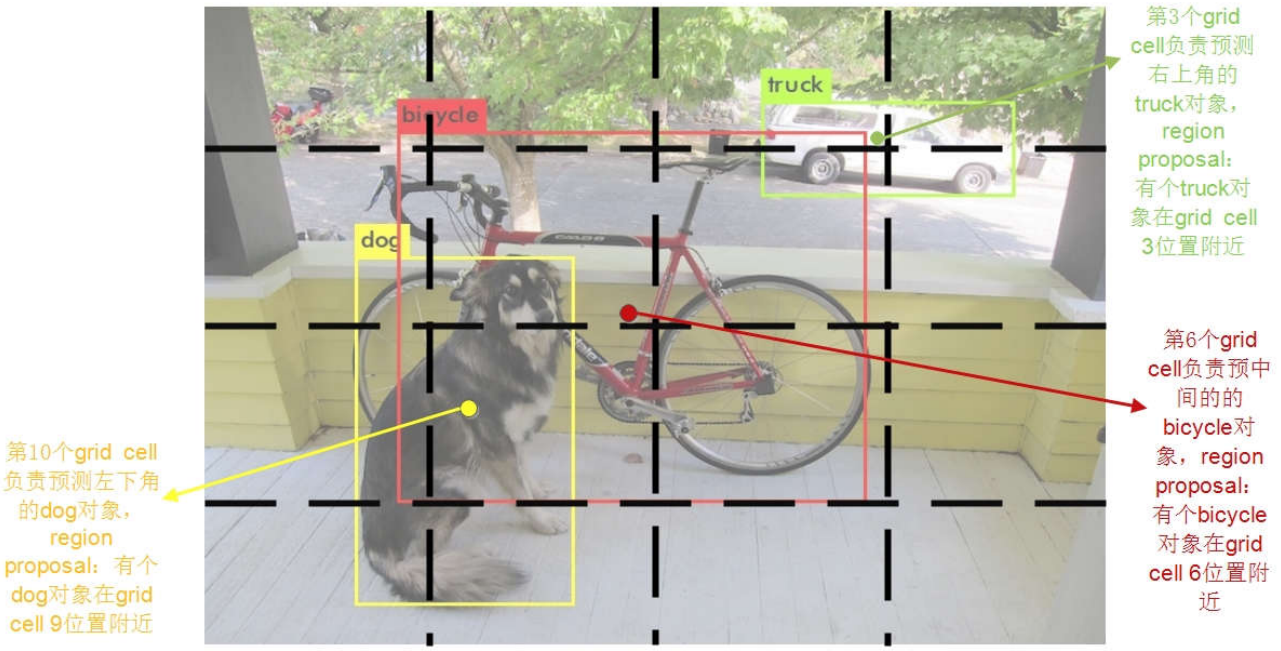
关于YOLO(v3)算法，你想知道的全在这里了

本文是我学习YOLO算法的理解总结，文中不会谈及YOLO的发展过程，不会与其他对象检测算法进行对比，也不会介绍YOLO9000相关的内容(会在文中阐述原因)，只单单总结目前YOLOv3算法的具体流程和实现细节。所以，下文中所有提到的YOLO，如非特别说明，均指YOLOv3。
如果需要了解更多对象检测算法，可以参考以下部分相关论文：
R-CNN
Fast R-CNN
Faster R-CNN
SSD
YOLOv1
YOLOv2
YOLOv3
RetinaNet

1.概述

在YOLO算法发表之前，大部分表现比较好的对象检测（Object Detection）算法都是以R-CNN为代表两阶段算法，这样的算法存在一个很明显的问题，那就是速度太慢，对于实时性要求很高的应用场景是不适用的。YOLO算法的作者没有走优化算法第一阶段或者第二阶段的老路子，而是直接提出一步完成预测，而且是只在一个CNN网络模型中完成图片中所有位置对象的box和类别预测，其预测速度大大提升，完全可以满足实时对象检测。

YOLO算法创新性地提出了将输入图片进行N*N的栅格化（每个小单元叫grid cell），然后将图片中某个对象的位置的预测任务交与该对象中心位置所在的grid cell的bouding box。简单理解的话，可以认为这也是一种很粗糙的区域推荐（region proposal），在训练的时候，我们通过grid cell的方式告诉模型，图片中对象A应该是由中心落在特定grid cell 的某个范围内的某些像素组成，模型接收到这些信息后就在grid cell周围以一定大小范围去寻找所有满足对象A特征的像素，经过很多次带惩罚的尝试训练后，它就能找到这个准确的范围了（说明不是瞎找），当然这个方位不仅是指长宽的大小范围，也包括小幅度的中心位置坐标变化，但是不管怎么变，中心位置不能越过该grid cell的范围。这样，大大限制了模型在图片中瞎找时做的无用功。同时，一个grid cell的所有bounding box（用来预测对象位置和类别的最小单元）只预测一个对象，在训练中这个对象的类别是固定的（就是ground truth box中心落入到该grid cell中的对象的类别），那么需要提取的特征也是特定的，这样就不会造成类别上的冲突，从而将位置检测和类别识别结合到一个CNN网络中预测，即只需要扫面一遍（you only look once）图片就能图片中所有对象的类别和位置信息。举例如下图。



其实，从上面的描述中我们可以察觉出YOLO算法一个很大的缺陷，那就是对于挨得很近的两个对象，他们的中心可能位于同一个grid cell中，但是这个grid cell只负责预测一个对象，这时候就一定会舍弃掉一个对象，造成检测准确率下降。当然，这也是有解决方法的，那就是尽量减小图片栅格话的栅格单元尺寸，从而将这两个对象分隔到不同的grid cell中。

以上是我个人理解的YOLO算法的核心思想，不管是YOLOv1还是v2、v3，其主要的核心还是以上所述，只是在bounding box的拟合方式、骨干网络的设计、模型训练的稳定性、精度方面有所提升罢了。下面对整个模型的网络结构、训练实现细节进行阐述。

2.训练

既然已经有了you only look once的想法，那接下来就要将这个想法数学化，这样才能用数学的方法训练模型学习拟合坐标和类别的特征，用于后期的预测。YOLO算法几乎是输入原图就直接预测出每个grid cell“附近”是否有某个对象和具体的 box位置，那最终这个想法数学化后便体现在loss函数上，这里我先不给出loss函数的具体公式，因为在提出loss函数之前要先了解三个概念：anchor box、置信度(confidence)和对象条件类别概率(conditional class probabilities)。作者提出，在网络最后的输出中，对于每个grid cell对应bounding box的输出有三类参数：一个是对象的box参数，一共是四个值，即box的中心点坐标（x,y）和box的宽和高（w,h）;一个是置信度，这是个区间在[0,1]之间的值；最后一个是一组条件类别概率，都是区间在[0,1]之间的值，代表概率。下面分别具体介绍这三个参数的意义。

2.1 anchor box（bounding box prior）

anchor box(论文中也称为bounding box prior，后面均使用anchor box)其实就是从训练集的所有ground truth box中统计出来的在训练集中最经常出现的几个box形状。比如，在某个训练集中最常出现的box形状有扁长的、瘦高的和宽高比例差不多的正方形这三种形状。我们可以预先将这些统计上的先验（或来自人类的）经验加入到模型中，这样模型在学习的时候，瞎找的可能性就更小了些，当然就有助于模型快速收敛了，从而加快训练速度。以前面提到的训练数据集中的ground truth box最常出现的三个形状为例，当模型在训练的时候我们可以告诉它，你要在grid cell 1附件找出的对象的形状要么是扁长的、要么是瘦高的、要么是长高比例差不多的正方形，你就不要再瞎试其他的形状了。

要在模型中使用这些形状，总不能告诉模型有个形状是瘦高的，还有一个是矮胖的，我们需要量化这些形状。YOLO的做法是想办法找出分别代表这些形状的宽和高，有了宽和高，尺寸比例即形状不就有了。YOLO作者的办法是使用k-means算法在训练集中所有样本的ground truth box中聚类出具有代表性形状的宽和高，作者将这种方法称作维度聚类（dimension cluster）。细心的读者可能会提出这个问题：到底找出几个anchor box算是最佳的具有代表性的形状。YOLO作者方法是做实验，聚类出多个数量不同anchor box组，分别应用到模型中，最终找出最优的在模型的复杂度和高召回率(high recall)之间折中的那组anchor box。作者在COCO数据集中使用了9个anchor box，我们前面提到的例子则有3个anchor box。

那么有了量化的anchor box后，怎么在实际的模型中加入anchor box的先验经验呢？我们在前面第1章中简单提到过最终负责预测grid cell中对象box的单元是bounding box,那我们可以让一个grid cell输出（预测）多个bounding box，然后每个bounding box负责预测不同的形状不就行了？比如前面例子中的3个不同形状的anchor box，我们的一个grid cell会输出3个参数相同的bounding box，第一个bounding box负责预测的形状与anchor box 1类似的box，其他两个bounding box依次类推。你可能觉得这里和我们第1章中提到的一个grid cell只预测一个对象位置是矛盾的，但是，一个grid cell输出多个bounding box，不一定每个都要用上，我们只要最终预测最好的一个bounding box的输出，其他的舍弃掉就可以了。那么如何分别在训练中找到这个最佳的bounding box呢？方法是求出每个bounding box的输出box与ground truth box的IOU(交并比)，IOU最大的bounding box就是最好的。

到此，还有最后一个问题需要解决，我们才能真正在训练中使用anchor box，那就是我们怎么告诉模型第一个bounding box负责预测的形状与anchor box 1类似，第二个bounding box负责预测的形状与anchor box 2类似？YOLO的做法是不让bounding box直接预测实际box的宽和高(w,h)，而是将预测的宽和高分别与anchor box的宽和高绑定，这样不管一开始bounding box输出的(w,h)是怎样的，经过转化后都是与anchor box的宽和高相关，这样经过很多次挑拣(IOU最大的bounding box)训练后，每个bounding box就知道自己该负责怎样形状的box预测了。这个绑定的关系是什么？那就是下面这个公式：

$$b_w = a_w e^{t_w}$$
$$b_h = a_h e^{t_h}$$

其中，a_w和a_h为anchor box的宽和高，t_w和t_h为bounding box直接预测出的宽和高，b_w和b_h为转换后预测的实际宽和高，这也就是最终预测中输出的宽和高。你可能会想，这个公式这么麻烦，为什么不能用$b_w = a_w * t_w, b_h = a_h * t_h$ 这样的公式，我的理解是上面的公式虽然计算起来比较麻烦，但是在误差函数求导后还带有t_w和t_h参数，而且也好求导*(此观点只是个人推测，需要进一步查证)*。

既然提到了最终预测的宽和高公式，那我们也就直接带出最终预测输出的box中心坐标(b_x, b_y)的计算公式，我们前面提到过box中心坐标总是落在相应的grid cell中的，所以bounding box直接预测出的t_x和t_y也是相对grid cell来说的，要想转换成最终输出的绝对坐标，需要下面的转换公式：

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$

其中，$\sigma(t_x)$为sigmoid函数，c_x和c_y分别为grid cell方格左上角点相对整张图片的坐标。作者使用这样的转换公式主要是因为在训练时如果没有将t_x和t_y压缩到(0,1)区间内的话，模型在训练前期很难收敛。 最终可以得出实际输出的box参数公式：

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= a_w e^{t_w} \\
b_h &= a_h e^{t_h}
\end{aligned}
$$

关于box参数的转换还有一点值得一提，作者在训练中并不是将t_x、t_y、t_w和t_h转换为b_x、b_y、b_w和b_h后与ground truth box的对应参数求误差，而是使用上述公式的逆运算将ground truth box的参数转换为与t_x、t_y、t_w和t_h对应的g_x、g_y、g_w和g_h，然后再计算误差。

关于anchor box训练相关的问题除了与loss函数相关的基本上都解释清楚了，但是预测的问题还没有解释清楚，还存在一个很关键的问题：在训练中我们挑选哪个bounding box的准则是选择预测的box与ground truth box的IOU最大的bounding box做为最优的box，但是在预测中并没有ground truth box，怎么才能挑选最优的bounding box呢？这就需要另外的参数了，那就是下面要说到的置信度。

2.2 置信度(confidence)

置信度是每个bounding box输出的其中一个重要参数，作者对他的作用定义有两重：一重是代表当前box是否有对象的概率$P_r(Object)$，注意，是对象，不是某个类别的对象，也就是说它用来说明当前box内只是个背景（backgroud）还是有某个物体（对象）；另一重表示当前的box有对象时，它自己预测的box与物体真实的box可能的IOU_{pred}^{truth}的值，注意，这里所说的物体真实的box实际是不存在的，这只是模型表达自己框出了物体的自信程度。以上所述，也就不难理解作者为什么将其称之为置信度了，因为不管哪重含义，都表示一种自信程度：框出的box内确实有物体的自信程度和框出的box将整个物体的所有特征都包括进来的自信程度。经过以上的解释，其实我们也就可以用数学形式表示置信度的定义了：

$$
C_i^j = P_r(Object) * IOU_{pred}^{truth}
$$

其中，C_i^j表示第i个grid cell的第j个bounding box的置信度。对于如何训练C_i^j的方法，在loss函数小结中说明。

2.3 对象条件类别概率(conditional class probabilities)

对象条件类别概率是一组概率的数组，数组的长度为当前模型检测的类别种类数量，它的意义是当bounding box认为当前box中有对象时，要检测的所有类别中每种类别的概率，其实这个和分类模型最后输出的一组类别概率是类似的，只是二者存在两点不同：1.YOLO的对象类别概率中没有background一项，也不需要，因为对background的预测已经交给置信度了，所以它的输出是有条件的，那就是在置信度表示当前box有对象的前提下，所以条件概率的数学形式为$P_r(class_i|Object)$;2.分类模型中最后输出之前使用softmax求出每个类别的概率，也就是说各个类别之间是互斥的，而YOLO算法的每个类别概率是单独用逻辑回归函数(sigmoid函数)计算得出了，所以每个类别不必是互斥的，也就是说一个对象可以被预测出多个类别。这个想法其实是有一些YOLO9000的意思的，这也是我为什么不介绍YOLO9000的原因，因为YOLOv3已经有9000类似的功能，不同只是不能像9000一样，同时使用分类数据集和对象检测数据集，且类别之间的词性是有从属关系的。

介绍完所有的输出参数后，我们总结下模型最终输出层的输出维数是多少。假如一个图片被分割成S*S个grid cell，我们有B个anchor box，也就是说每个grid cell有B个bounding box, 每个bounding box内有4个位置参数，1个置信度，classes个类别概率，那么最终的输出维数是：$S*S*[B*(4+1+classes)]$。

2.4 loss函数

介绍完模型最终输出中有哪些参数后，我们应该可以定义loss函数了，作者使用了最简单的差平方和误差（sum-squared error），使用的原因很简单，因为好优化。那我们试着给出loss函数的公式：

$$
\begin{aligned}
loss = &\sum_{i=0}^{S^2}\sum_{j=0}^{B}[(x_i^j - \widehat{x}_i^j)^2 + (y_i^j - \hat{y}_i^j)^2] \\
&+ \sum_{i=0}^{S^2}\sum_{j=0}^{B}[(w_i^j - \widehat{w}_i^j)^2 + (h_i^j - \widehat{h}_i^j)^2] \\
&+ \sum_{i=0}^{S^2}\sum_{j=0}^{B}(C_i^j - \widehat{C}_i^j)^2 \\
&+ \sum_{i=0}^{S^2}\sum_{j=0}^{B}\sum_{c\in classes}(p_i^j(c) - \hat{p}_i^j(c))^2
\end{aligned}
$$

如果看过YOLO的论文你会发现，这里的公式和论文中的公式虽然相似，但是差别还是很大的。其实，作者是在上面这个公式的基础上加了很多限制和优化参数，上面的公式只是我为了更好说明YOLO的loss公式而给出的对比公式，这样有助于更好的理解

YOLO的loss函数公式中加入的每个参数的意义，下面给出真正的YOLO loss函数公式（这个公式也是我根据自己的理解总结出来的，三篇论文中都未给出此类似的公式）：

$$
\begin{aligned}
loss = & \sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}[(x_i^j-\widehat{x}_i^j)^2+(y_i^j-\hat{y}_i^j)^2] \\
& +\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}[(w_i^j-\widehat{w}_i^j)^2+(h_i^j-\widehat{h}_i^j)^2] \\
& +\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{G}_{ij}(C_i^j-\widehat{C}_i^j)^2 \\
& +\sum_{i=0}^{S^2}\sum_{j=0}^{B}\sum_{c\in classes}\mathbb{I}_{ij}^{obj}(p_i^j(c)-\hat{p}_i^j(c))^2
\end{aligned}
$$

细心的你一定也注意到了，这个公式和YOLOv1论文中的公式还是不一样。那是因为在YOLOv3中，作者将置信度和条件类别概率放到了每个bounding box中，即每个bounding box都有一对置信度和条件类别概率，而v1中所有的bounding box共用一对置信度和条件类别概率，上文中在说明输出的各个参数时，默认解释的是v3的输出格式，关于v1的细节不再赘述。下面几点是loss函数的几点细节:

- v1和v2 loss中的λ_{coord}和λ_{noobj} 参数去哪了
  v3去掉了v1和v2 loss中的λ_{coord}和λ_{noobj} 参数
  这点在论文中未提及，但是在作者的实现源码中未使用，可能模型已经变得更加稳定，不需要这样的微调。当然，在darknet的yolov3.cfg配置文件中也没有这两个参数的配置。

- 参数$\mathbb{I}_{ij}^{obj}$
  在训练中，如果某个grid cell的bounding box没有负责预测某个对象，那我们就不应该训练该bounding box的条件类别概率和坐标参数，因为使用这些参数的前提是在明确清楚该bounding box负责预测某个ground truth box(后面说明怎么决定是否负责)，即不应该根据条件类别概率和中心坐标输出误差调整相应的weights，那如何不进行这部分训练呢，当然是不让他们对loss做出贡献，也就没有它们什么事情了，这个时候就需要 参数$\mathbb{I}_{ij}^{obj}$了，当该bounding box负责预测某个ground truth box时，$\mathbb{I}_{ij}^{obj}=1$，否则，$\mathbb{I}_{ij}^{obj}=0$。

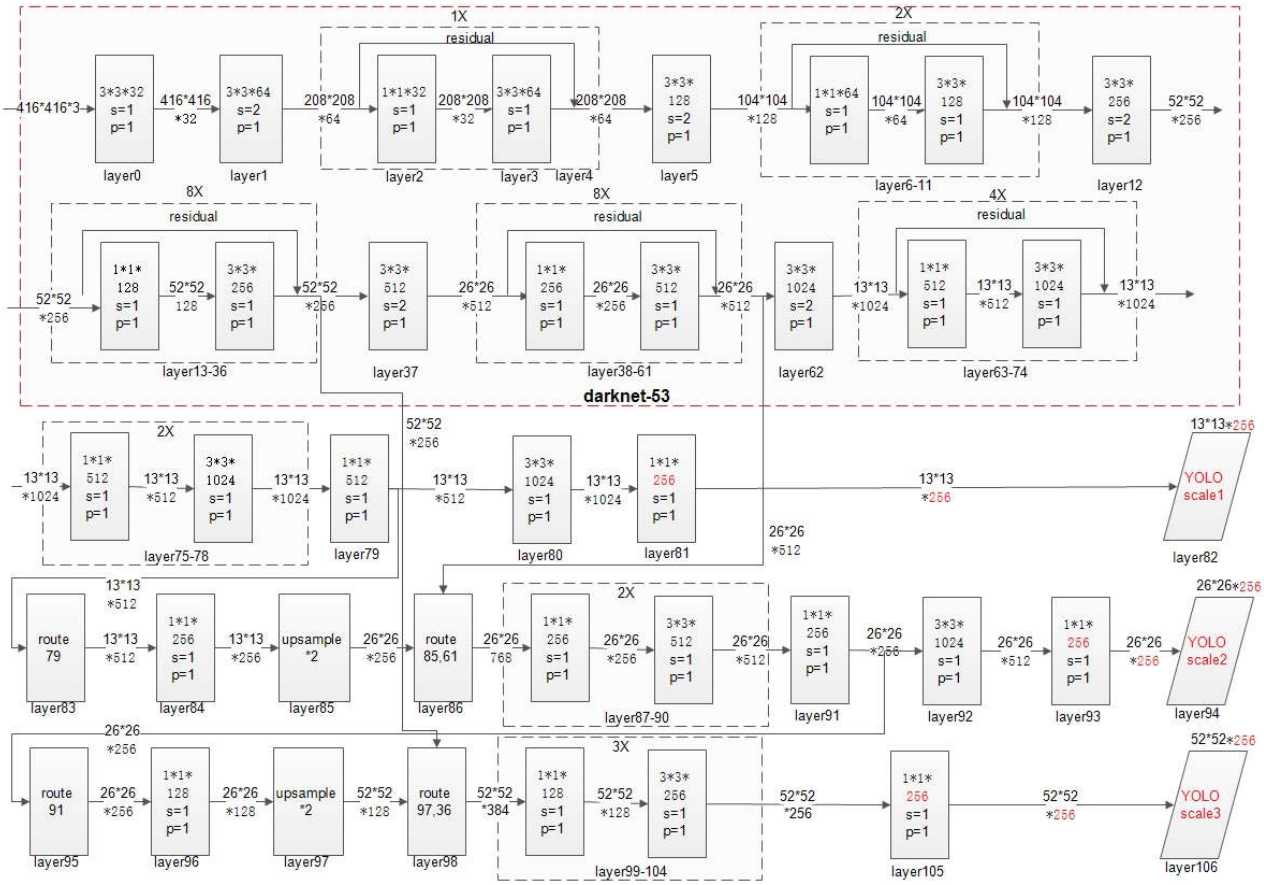
- C_i^j和$\mathbb{G}_{ij}$的值如何确定
  训练中，C_i^j的取值是由grid cell的bounding box有没有负责预测某个对象决定的。如果负责，那么$C_i^j=1$，否则，$C_i^j=0$。下面我们来说明如何确定某个grid cell的bounding box是否负责预测该grid cell中的对象：前面在说明anchor box的时候提到每个bounding box负责预测的形状是依据与其对应的anchor box（bounding box prior）相关的，那这个anchor box与该对象的ground truth box的IOU在所有的anchor box（与一个grid cell中所有bounding box对应，COCO数据集中是9个）与ground truth box的IOU中最大，那它就负责预测这个对象，因为这个形状、尺寸最符合当前这个对象，这时$C_i^j=1$，其他情况下$C_i^j=0$。注意，你没有看错，就是所有anchor box与某个ground truth box的IOU最大的那个anchor box对应的bounding box负责预测该对象，与该bounding box预测的box没有关系。另外，这里有个例外，当预测的某个bounding box（没错这个又和bounding box预测的box有关）与某个对象的ground truth box的IOU大于设定的阈值时（论文中是0.5，darknet中针对COCO数据集使用的是0.7），忽略该bounding box所有输出的对loss的误差贡献，包括置信度误差，这时$\mathbb{G}_{ij}=0$，其他情况$\mathbb{G}_{ij}=1$。结合之前的说明可以看出，参数$\mathbb{I}_{ij}^{obj}$ 和C_i^j的值其实是保持一致的。 如果你不理解为何作者这样做，建议阅读faster-rcnn的论文，作者的做法其实是借鉴了faster-rcnn的anchor box思想。

- grid cell的个数S如何确定
  自v2后，YOLO算法网络结构中只使用卷积和池化操作进行特征提取和推理运算，去掉了传统的全连接层，这样的做法有一个好处，就是理论上整个网络不再限制输入图片的尺寸，因为卷积层本来就对输入的尺寸没有限制。作者在这样的基础上，每隔一段训练周期随意改变输入层的尺寸后再进行下一阶段训练，这样就让模型在不同尺寸的图片上都表现良好，作者将之称为Multi-Scale training。当然，输入的尺寸并不是随心所欲地改变，而是在（10*32, 11*32, 12*32, ..., 19*32）这几个尺寸中随机选择。当确定了出入层的大小后，模型通过卷积层输入输出尺寸公式计算后，便能预先知道在某个输入尺寸前提下，最后的输出层尺寸是多少了，也就是grid cell的个数，即loss函数中S的值。其实，S的取值只可能在集合（10*10,11*11, 12*12,...,19*19）中，因为YOLO模型的降采样（down-sample）因子是32。

2.5 跨尺寸预测（Predictions across scales）

YOLO算法从三个不同的尺寸预测对象box，这三个不同的尺寸来自不同层级的卷积层的输出。该方法借鉴了feature pyramid network的思想: 由于卷积层每隔几层，特征映射(feature mapping)的宽和高就会减少，而通道数会增加，随着网络层次的加深，特征映射组成的形状类似于金字塔，如果将不同层级的特征映射转换为最终的输出，那么将有助于提升模型在对象不同尺度大小上的表现，即有助于提高模型从小目标到大目标的综合检测（box的精度）能力，关于feature pyramid network的具体内容，此处不详

细展开，可参考论文。YOLO并没有完全使用feature pyramid network的思想，在说明YOLO的做法前，我们先看下YOLO模型的网络结构，我们以检测COCO数据集输入尺寸为416*416的网络结构为例(COCO数据集类别数为80，anchor box总数为9):



从上面的模型的网络结构图我们可以明显看出基于darknet-53的最新的模型结构有以下几个特点:
1. 从网络的不同层次映射不同尺寸的输出，如图中从79层（外加两个卷积层）得到13*13的的输出;从91层（外加两个卷积层）得到26*26的输出; 最后再得到52*52的输出。
2. 后面的高层结合使用低层特征（图中的86、98层，分别使用了61层和36层的特征映射），使高层能使用细粒度（fine grained）特征和更多的语义信息。
3. 最后一个尺寸输出使用了前两个尺寸计算的特征映射，使得最后的尺寸输出也能使用细粒度。 4. 每个YOLO输出层中，每个grid cell的bounding box数量为3，而不是9，这样不同的YOLO输出层便能负责不同尺寸大小的对象预测了。例如，COCO数据集中，作者让YOLO scale1负责预测的尺寸有(10,13)、 (16,30)和 (33,23), YOLO scale2负责预测的尺寸有(30,61)、(62,45)和(59,119)，YOLO scale3负责预测的尺寸有(116,90)、(156,198)和(373,326)。

2.6 batch normalization和data augmentation

在训练中使用batch normalization和data augmentation是标准组件，此处不再展开。

以上，就是YOLO算法训练部分的细节，训练好模型后，下面就是如何使用模型进行预测了，虽然YOLO是号称训练和预测使用一套模型，但是在具体使用中，还是有些细微的差别。

3. 预测

使用非极大值抑制(NMS non-max suppression)

虽然说在训练时只要求一个对象只被一个grid cell负责预测，这在训练时是可以通过label限制的，但是在预测时，尤其是那种尺寸比较大的对象，可能跨越多个grid cell，且每个grid cell预测它都挺合理的，那么模型就会对一个对象预测出多个box，但我们最终只需要一个box就可以了，这种情况就需要使用非极大值抑制算法选择出预测最好的box，并且剔除掉其他较次的box，最后只采用这个最好的box作为最终的预测box。该算法基本上在所有的对象检测算法中都会使用，所以是个通用算法，此处不再详述。

骨干网络（backbone network，论文中成为特征提取器，feature extractor)

YOLO模型的骨干网络是作者自己设计的darknet-53，顾名思义，它有53个卷积层，网络结构的具体细节如2.5小结YOLO模型网络结构图中红色虚线框出的部分（去掉了作为分类器时的全连接层）。从图中可以看出，作者在网络中使用了大量的残差模块。这里darknet-53有个细节，作者的残差模块(residual unit)使用的是resnet v2里面提到的pre-activation残差模块。

最后

以上就是我个人对YOLOv3算法的所有理解，由于个人理解可能存在偏差，文中不免会有谬误，欢迎通过issue或者邮件批评指正或者交流。