[YOLOV1算法原理 2](#_Toc1089577270)

[1、Abstract 2](#_Toc1769997436)

[2、Introduction 3](#_Toc1942317920)

[3、Unified Detection 5](#_Toc1367816536)

[4、Network Design 9](#_Toc587596536)

[5、Training 10](#_Toc1449187076)

[6、损失函数 11](#_Toc1324680356)

[7、测试 13](#_Toc1332047025)

[YOLOV2算法原理 14](#_Toc165185822)

[1 概述 14](#_Toc1255354279)

[2 回顾YOLOV1 14](#_Toc1109971780)

[3 YOLOv2精度的改进（Better） 15](#_Toc2014590037)

[3.1 Batch Normalization 16](#_Toc569302386)

[3.2 High Resolution Classifier 16](#_Toc1989849868)

[3.3 Convolutional With Anchor Boxes 17](#_Toc1250287182)

[3.4 Dimension Clusters（维度聚类） 18](#_Toc1955663841)

[3.5 Direct location prediction（直接位置预测） 20](#_Toc504393322)

[3.6 Fine-Grained Features（细粒度特征） 23](#_Toc1579209447)

[3.7 Multi-Scale Training 24](#_Toc1617469113)

[3.8 Further Experiments 26](#_Toc1524694555)

[4 YOLOv2速度的改进（Faster） 27](#_Toc1821394816)

[4.1 Darknet-19 27](#_Toc1306119927)

[4.2 Training for classification 29](#_Toc156304432)

[4.3 Training for detection 29](#_Toc1686876574)

[5 YOLOv2分类的改进（Stronger） 30](#_Toc114310299)

# YOLOV1算法原理

## 1、Abstract

作者提出了一种新的物体检测方法YOLO。YOLO之前的物体检测方法主要是通过region proposal产生大量的可能包含待检测物体的 potential bounding box，再用分类器去判断每个 bounding box里是否包含有物体，以及物体所属类别的 probability或者 confidence，如R-CNN,Fast-R-CNN,Faster-R-CNN等。

YOLO不同于这些物体检测方法，它将物体检测任务当做一个regression问题来处理，使用一个神经网络，直接从一整张图像来预测出bounding box 的坐标、box中包含物体的置信度。因为YOLO的物体检测流程是在一个神经网络里完成的，所以可以end to end来优化物体检测性能。

YOLO检测物体的速度很快，标准版本的YOLO在Titan X 的 GPU 上能达到45 FPS。网络较小的版本Fast YOLO在保持mAP是之前的其他实时物体检测器的两倍的同时，检测速度可以达到155 FPS。

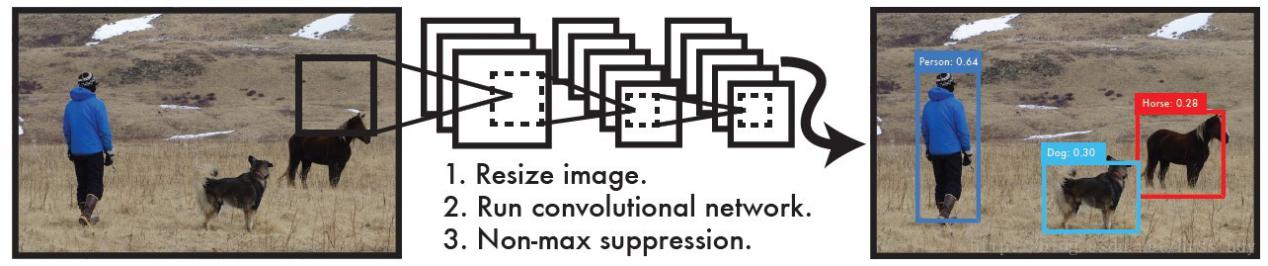
相较于其他的state-of-the-art 物体检测系统，YOLO在物体定位时更容易出错，但是在背景上预测出不存在的物体（false positives）的情况会少一些。而且，YOLO比DPM、R-CNN等物体检测系统能够学到更加抽象的物体的特征，这使得YOLO可以从真实图像领域迁移到其他领域，如艺术。

## 2、Introduction

YOLO之前的物体检测系统使用分类器来完成物体检测任务。为了检测一个物体，这些物体检测系统要在一张测试图的不同位置和不同尺寸的bounding box上使用该物体的分类器去评估是否有该物体。如DPM系统，要使用一个滑窗（sliding window）在整张图像上均匀滑动，用分类器评估是否有物体。

在DPM之后提出的其他方法，如R-CNN方法使用region proposal来生成整张图像中可能包含待检测物体的potential bounding boxes，然后用分类器来评估这些boxes，接着通过post-processing来改善bounding boxes，消除重复的检测目标，并基于整个场景中的其他物体重新对boxes进行打分。整个流程执行下来很慢，而且因为这些环节都是分开训练的，检测性能很难进行优化。

作者设计了YOLO（you only look once），将物体检测任务当做回归问题（regression problem）来处理，直接通过整张图片的所有像素得到bounding box的坐标、box中包含物体的置信度和class probabilities。通过YOLO，每张图像只需要看一眼就能得出图像中都有哪些物体和这些物体的位置。



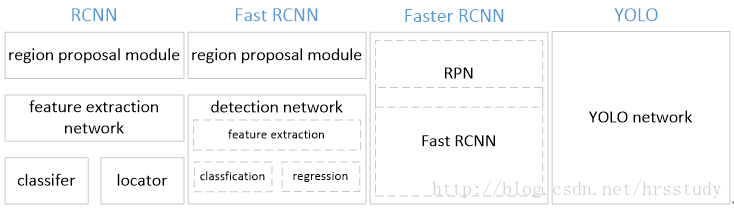
如图所示，使用YOLO来检测物体，其流程是非常简单明了的：   
如图所示，使用YOLO来检测物体，其流程是非常简单明了的：

A）将图像resize到448 \* 448作为神经网络的输入

B）运行神经网络，得到一些bounding box坐标、box中包含物体的置信度和 class probabilities

C）进行非极大值抑制，筛选Boxes

下图是各物体检测系统的检测流程对比：



YOLO模型相对于之前的物体检测方法有多个优点：

A）YOLO检测物体非常快。

因为没有复杂的检测流程，只需要将图像输入到神经网络就可以得到检测结果，YOLO可以非常快的完成物体检测任务。标准版本的YOLO在Titan X 的 GPU 上能达到45 FPS。更快的Fast YOLO检测速度可以达到155 FPS。而且，YOLO的mAP是之前其他实时物体检测系统的两倍以上。

B）YOLO可以很好的避免背景错误，产生false positives。

不像其他物体检测系统使用了滑窗或region proposal，分类器只能得到图像的局部信息。YOLO在训练和测试时都能够看到一整张图像的信息，因此YOLO在检测物体时能很好的利用上下文信息，从而不容易在背景上预测出错误的物体信息。和Fast-R-CNN相比，YOLO的背景错误不到Fast-R-CNN的一半。

C）YOLO可以学到物体的泛化特征。

当YOLO在自然图像上做训练，在艺术作品上做测试时，YOLO表现的性能比DPM、R-CNN等之前的物体检测系统要好很多。因为YOLO可以学习到高度泛化的特征，从而迁移到其他领域。

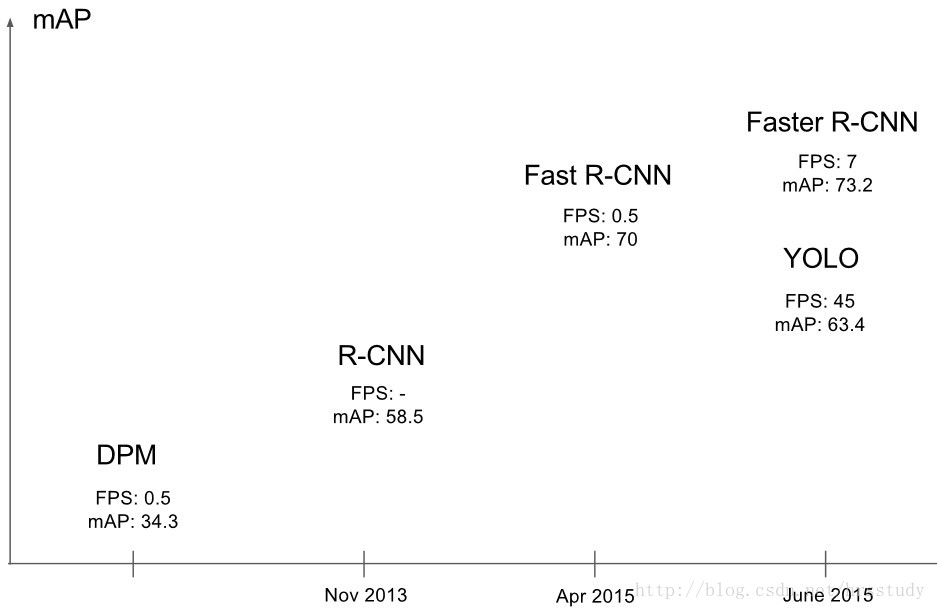
尽管YOLO有这些优点，它也有一些缺点：

A）YOLO的物体检测精度低于其他state-of-the-art的物体检测系统。

B）YOLO容易产生物体的定位错误。

C）YOLO对小物体的检测效果不好（尤其是密集的小物体，因为一个栅格只能 预测2个物体）。

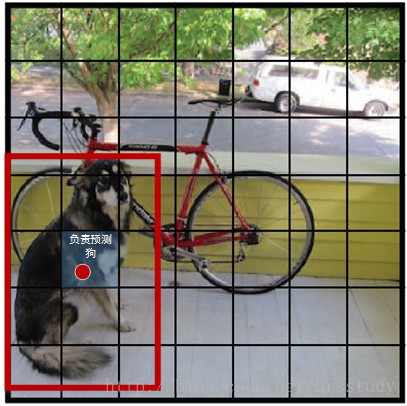
下图是各物体检测系统的检测性能对比：



## 3、Unified Detection

YOLO将输入图像划分为S\*S的栅格，每个栅格负责检测中心落在该栅格中的物体，如下图：

（怎样知道是物体的中心？？？）



（1）每一个栅格预测B个bounding boxes，以及这些bounding boxes的confidence scores。

这个 confidence scores反映了模型对于这个栅格的预测：该栅格是否含有物体，以及这个box的坐标预测的有多准。

公式定义如下：

gdfg

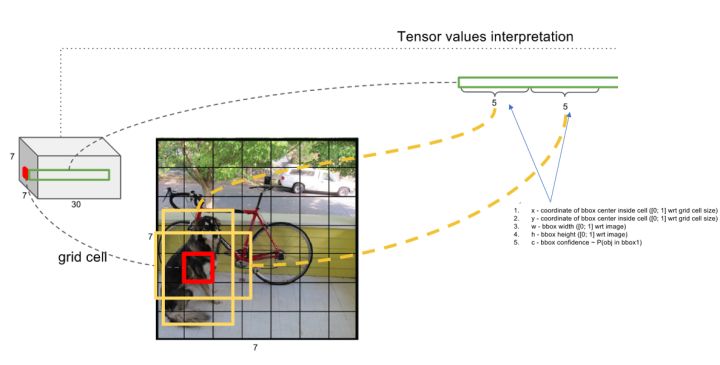
如果这个栅格中不存在一个 object，则confidence score应该为0；否则的话，confidence score则为 predicted bounding box与 ground truth box之间的 IOU（intersection over union）。

YOLO对每个bounding box有5个predictions：x, y, w, h,confidence。

坐标x,y代表了预测的bounding box的中心与栅格边界的相对值。

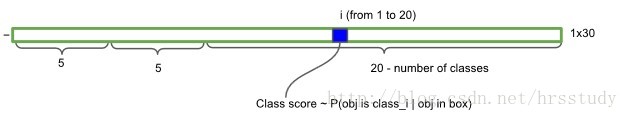
坐标w,h代表了预测的bounding box的width、height相对于整幅图像width,height的比例。

confidence就是预测的bounding box和ground truth box的IOU值。



（2）每一个栅格还要预测C个 conditional class probability（条件类别概率）：Pr(Classi|Object)。即在一个栅格包含一个Object的前提下，它属于某个类的概率。

我们只为每个栅格预测一组（C个）类概率，而不考虑框B的数量。



注意：

conditional class probability信息是针对每个网格的。

confidence信息是针对每个bounding box的。

（3）在测试阶段，将每个栅格的conditional class probabilities与每个 bounding box的 confidence相乘：

rthr

这样既可得到每个bounding box的具体类别的confidence score。

这乘积既包含了bounding box中预测的class的 probability信息，也反映了bounding box是否含有Object和bounding box坐标的准确度。

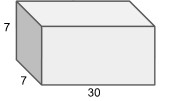
（4）将YOLO用于PASCAL VOC数据集时：

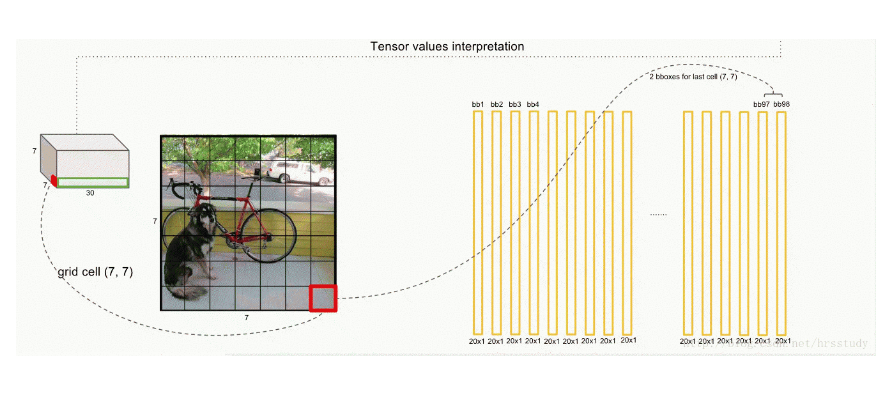
论文使用的 S=7，即将一张图像分为7×7=49个栅格每一个栅格预测B=2个boxes（每个box有 x,y,w,h,confidence，5个预测值），同时C=20（PASCAL数据集中有20个类别）。

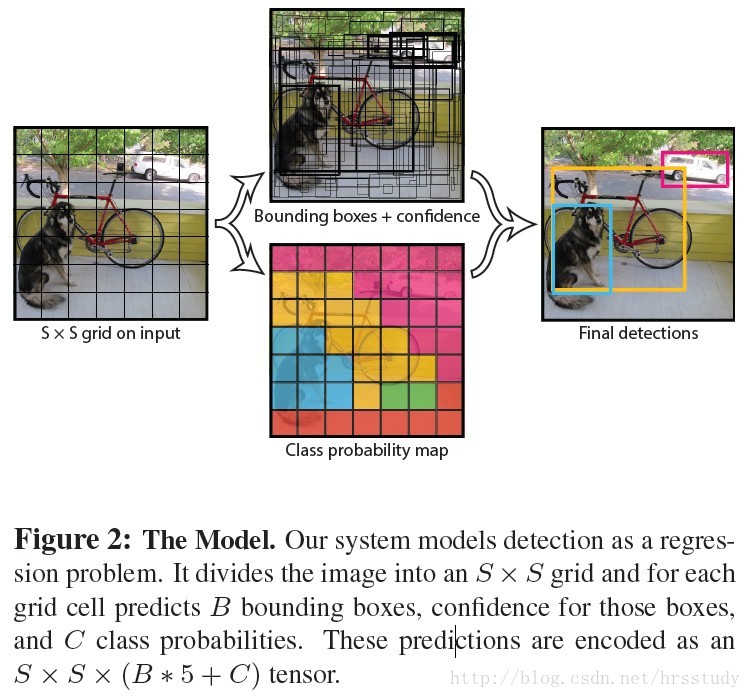
因此，最后的prediction是7×7×30 { 即S \* S \* ( B \* 5 + C) }的Tensor。

简单的全部采用了sum-squared error loss来做这件事会有以下不足：

a) 8维的localization e

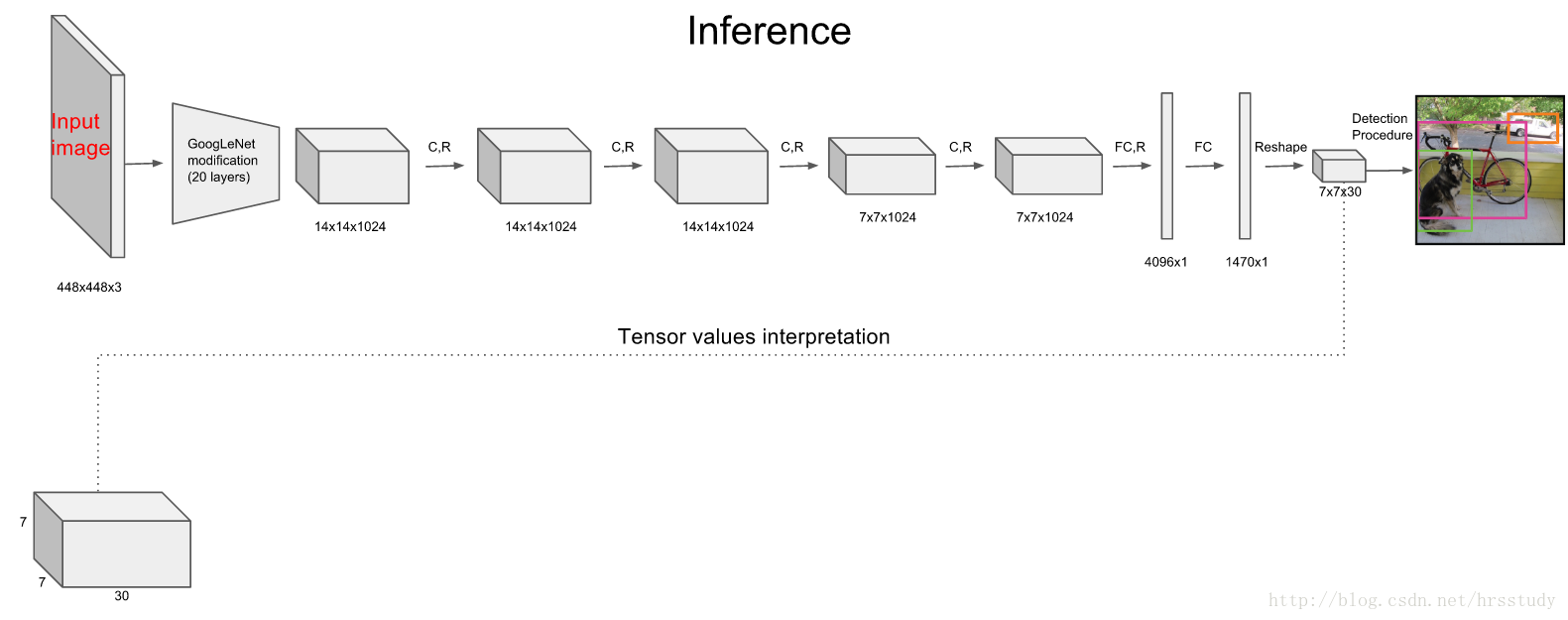


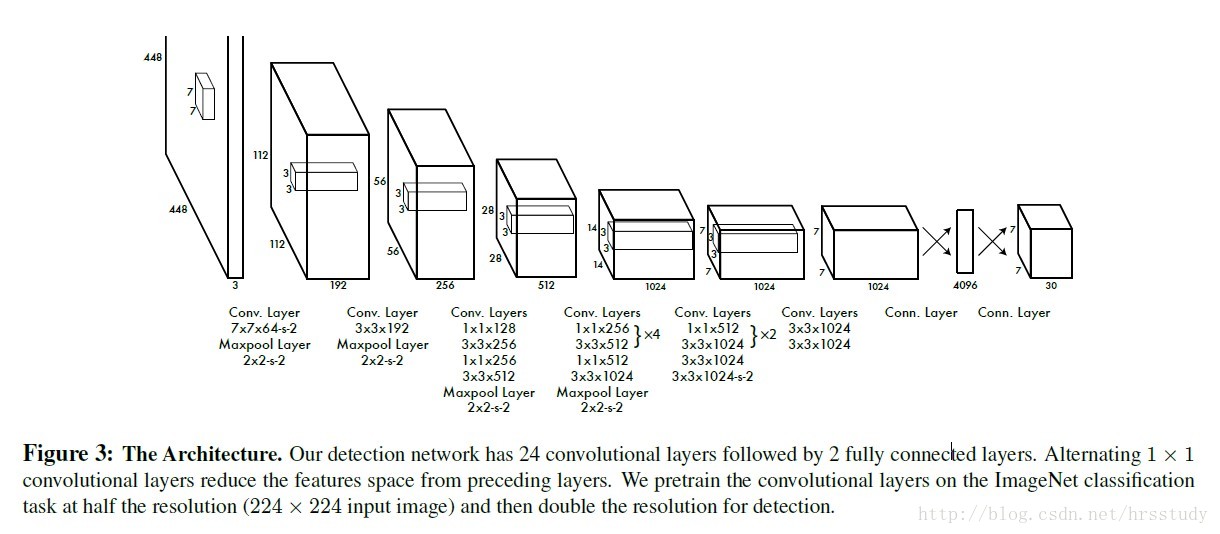




## 4、Network Design

YOLO检测网络包括24个卷积层和2个全连接层,如图所示：





其中，卷积层用来提取图像特征，全连接层用来预测图像位置和类别概率值。

YOLO网络借鉴了GoogLeNet分类网络结构。不同的是，YOLO未使用inception module，而是使用1x1卷积层（此处1x1卷积层的存在是为了跨通道信息整合）+3x3卷积层简单替代。

## 5、Training

首先利用ImageNet 1000-class的分类任务数据集Pretrain卷积层。使用上述网络中的前20 个卷积层，加上一个 average-pooling layer，最后加一个全连接层，作为 Pretrain 的网络。训练大约一周的时间，使得在ImageNet 2012的验证数据集Top-5的精度达到 88%，这个结果跟 GoogleNet 的效果相当。

将Pretrain的结果的前20层卷积层应用到Detection中，并加入剩下的4个卷积层及2个全连接。同时为了获取更精细化的结果，将输入图像的分辨率由 224\* 224 提升到 448\* 448。将所有的预测结果都归一化到 0~1, 使用 Leaky RELU 作为激活函数。

为了防止过拟合，在第一个全连接层后面接了一个 ratio=0.5 的 Dropout 层。为了提高精度，对原始图像做数据提升。

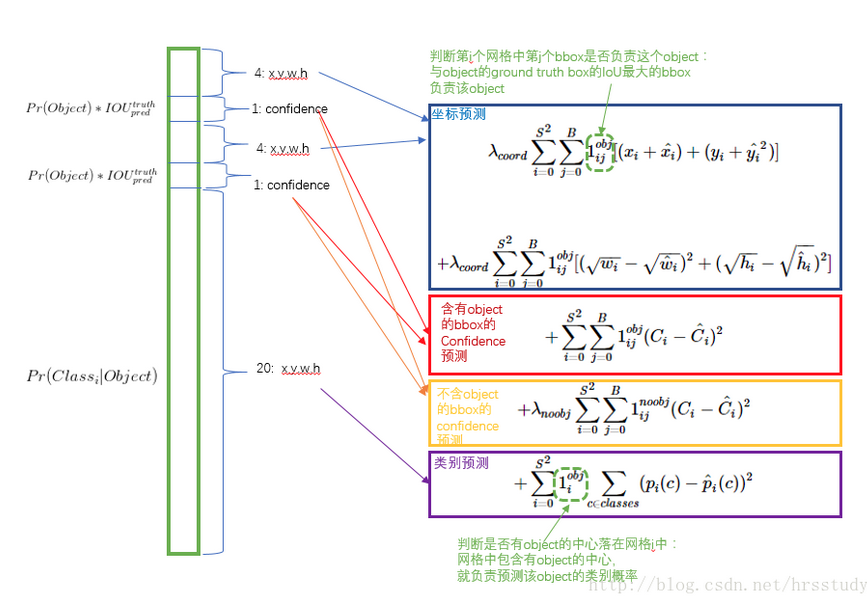
## 6、损失函数

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：

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

b) 如果一些栅格中没有object（一幅图中这种栅格很多），那么就会将这些栅 格中的bounding box的confidence 置为0，相比于较少的有object的 栅格，这些不包含物体的栅格对梯度更新的贡献会远大于包含物体的栅格对 梯度更新的贡献，这会导致网络不稳定甚至发散。



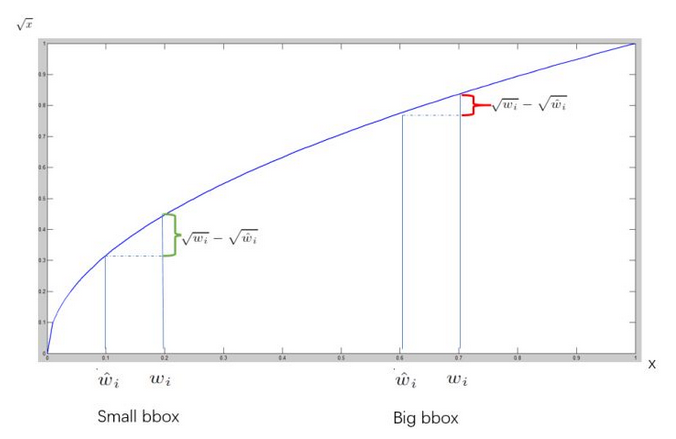
解决方案如下：

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 λcoord ,在pascal VOC训练中取5。（上图蓝色框）

B）对没有object的bbox的confidence loss，赋予小的loss weight，记为 λnoobj ，在pascal VOC训练中取0.5。（上图橙色框）

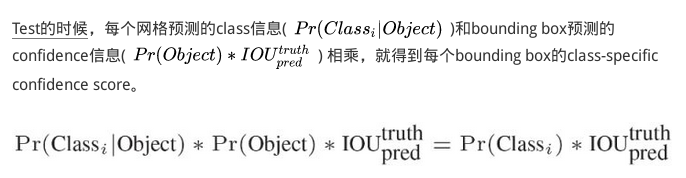
C）有object的bbox的confidence loss (上图红色框) 和类别的loss （上 图紫色框）的loss weight正常取1。

D）对不同大小的bbox预测中，相比于大bbox预测偏一点，小box预测偏一 点更不能忍受。而sum-square error loss中对同样的偏移loss是一样。 为 了缓和这个问题，作者用了一个比较取巧的办法，就是将box的width和 height取平方根代替原本的height和width。 如下图：small bbox的横 轴值较小，发生偏移时，反应到y轴上的loss（下图绿色）比big box(下 图红色)要大。



E）一个网格预测多个bounding box，在训练时我们希望每个object（ground true box）只有一个bounding box专门负责（一个object 一个bbox）。 具体做法是与ground true box（ object）的IOU最大的bounding box 负责该ground true box(object)的预测。这种做法称作bounding box predictor的specialization(专职化)。每个预测器会对特定（sizes,aspect ratio or classed of object）的ground true box预测的越来越好。（个 人理解：IOU最大者偏移会更少一些，可以更快速的学习到正确位置）

## 7、测试



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。

# YOLOV2算法原理

## 1 概述

时隔一年，YOLO（You Only Look Once: Unified, Real-Time Object Detection）从v1版本进化到了v2版本，作者在darknet主页先行一步放出源代码，论文在我们等候之下终于在12月25日发布出来，本文对论文重要部分进行了翻译理解工作，不一定完全对，如有疑问，欢迎讨论。博主如果有新的理解，也会更新文章。

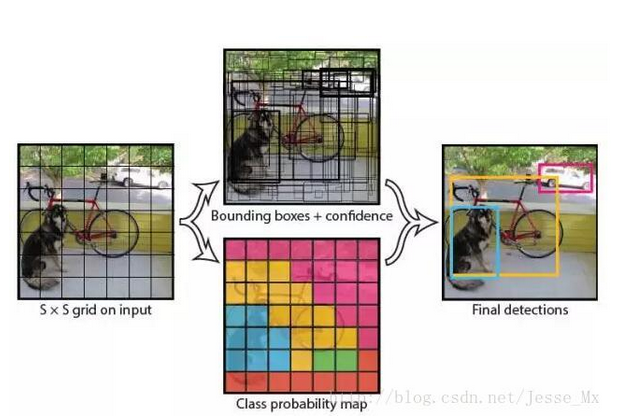
新的YOLO版本论文全名叫“YOLO9000: Better, Faster, Stronger”，主要有两个大方面的改进：

第一，作者使用了一系列的方法对原来的YOLO多目标检测框架进行了改进，在保持原有速度的优势之下，精度上得以提升。VOC 2007数据集测试，67FPS下mAP达到76.8%，40FPS下mAP达到78.6%，基本上可以与Faster R-CNN和SSD一战。这一部分是本文主要关心的地方。

第二，作者提出了一种目标分类与检测的联合训练方法，通过这种方法，YOLO9000可以同时在COCO和ImageNet数据集中进行训练，训练后的模型可以实现多达9000种物体的实时检测。这一方面本文暂时不涉及，待后面有时间再补充。

## 2 回顾YOLOV1

YOLOv2始终是在v1版本上作出的改进，我们先简单回顾YOLOv1的检测步骤：



A）给个一个输入图像，首先将图像划分成7 \* 7的网格。

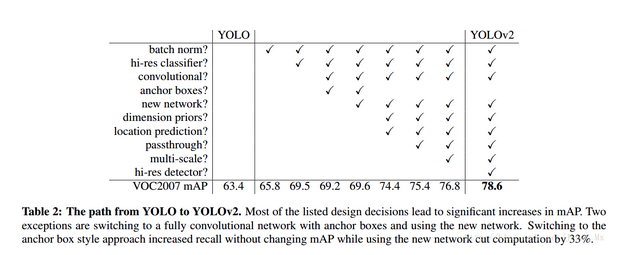
B）对于每个网格，每个网格预测2个bouding box（每个box包含5个预测 量）以及20个类别概率，总共输出7×7×（2\*5+20）=1470个tensor

C）根据上一步可以预测出7 \* 7 \* 2 = 98个目标窗口，然后根据阈值去除可 能性比较低的目标窗口，再由NMS去除冗余窗口即可。

YOLOv1使用了end-to-end的回归方法，没有region proposal步骤，直接回归便完成了位置和类别的判定。种种原因使得YOLOv1在目标定位上不那么精准，直接导致YOLO的检测精度并不是很高。

## 3 YOLOv2精度的改进（Better）

先来一个总览图，看看它到底用了多少技巧，以及这些技巧起了多少作用：



### 3.1 Batch Normalization

CNN在训练过程中网络每层输入的分布一直在改变, 会使训练过程难度加大，但可以通过normalize每层的输入解决这个问题。新的YOLO网络在每一个卷积层后添加batch normalization，通过这一方法，mAP获得了2%的提升。batch normalization 也有助于规范化模型，可以在舍弃dropout优化后依然不会过拟合。

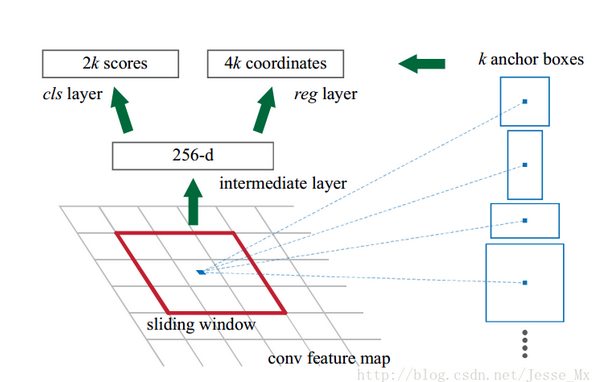
### 3.2 High Resolution Classifier

目前的目标检测方法中，基本上都会使用ImageNet预训练过的模型（classifier）来提取特征，如果用的是AlexNet网络，那么输入图片会被resize到不足256 \* 256，导致分辨率不够高，给检测带来困难。为此，新的YOLO网络把分辨率直接提升到了448 \* 448，这也意味之原有的网络模型必须进行某种调整以适应新的分辨率输入。

对于YOLOv2，作者首先对分类网络（自定义的darknet）进行了fine tune，分辨率改成448 \* 448，在ImageNet数据集上训练10轮（10 epochs），训练后的网络就可以适应高分辨率的输入了。然后，作者对检测网络部分（也就是后半部分）也进行fine tune（微调）。这样通过提升输入的分辨率，mAP获得了4%的提升。

### 3.3 Convolutional With Anchor Boxes

之前的YOLO利用全连接层的数据完成边框的预测，导致丢失较多的空间信息，定位不准。作者在这一版本中借鉴了Faster R-CNN中的anchor思想，回顾一下，anchor是RNP网络中的一个关键步骤，说的是在卷积特征图上进行滑窗操作，每一个中心可以预测9种不同大小的建议框。看到YOLOv2的这一借鉴，我只能说SSD的作者是有先见之明的。



为了引入anchor boxes来预测bounding boxes，作者在网络中果断去掉了全连接层。剩下的具体怎么操作呢？首先，作者去掉了后面的一个池化层以确保输出的卷积特征图有更高的分辨率。然后，通过缩减网络，让图片输入分辨率为416 \* 416，这一步的目的是为了让后面产生的卷积特征图宽高都为奇数，这样就可以产生一个center cell。作者观察到，大物体通常占据了图像的中间位置， 就可以只用中心的一个cell来预测这些物体的位置，否则就要用中间的4个cell来进行预测，这个技巧可稍稍提升效率。最后，YOLOv2使用了卷积层降采样（factor为32），使得输入卷积网络的416 \* 416图片最终得到13 \* 13的卷积特征图（416/32=13）。

加入了anchor boxes后，可以预料到的结果是召回率上升，准确率下降。我们来计算一下，假设每个cell预测9个建议框，那么总共会预测13 \* 13 \* 9 = 1521个boxes，而之前的网络仅仅预测7 \* 7 \* 2 = 98个boxes。具体数据为：没有anchor boxes，模型recall为81%，mAP为69.5%；加入anchor boxes，模型recall为88%，mAP为69.2%。这样看来，准确率只有小幅度的下降，而召回率则提升了7%，说明可以通过进一步的工作来加强准确率，的确有改进空间。

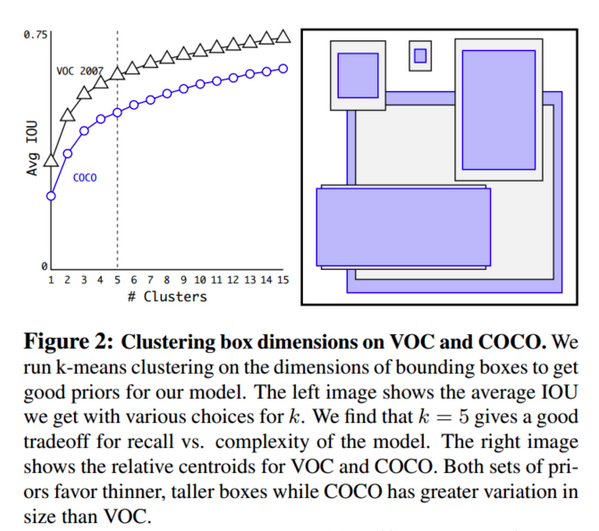
### 3.4 Dimension Clusters（维度聚类）

作者在使用anchor的时候遇到了两个问题，第一个是anchor boxes的宽高维度往往是精选的先验框（hand-picked priors），虽说在训练过程中网络也会学习调整boxes的宽高维度，最终得到准确的bounding boxes。但是，如果一开始就选择了更好的、更有代表性的先验boxes维度，那么网络就更容易学到准确的预测位置。

和以前的精选boxes维度不同，作者使用了K-means聚类方法类训练bounding boxes，可以自动找到更好的boxes宽高维度。传统的K-means聚类方法使用的是欧氏距离函数，也就意味着较大的boxes会比较小的boxes产生更多的error，聚类结果可能会偏离。为此，作者采用的评判标准是IOU得分（也就是boxes之间的交集除以并集），这样的话，error就和box的尺度无关了，最终的距离函数为：

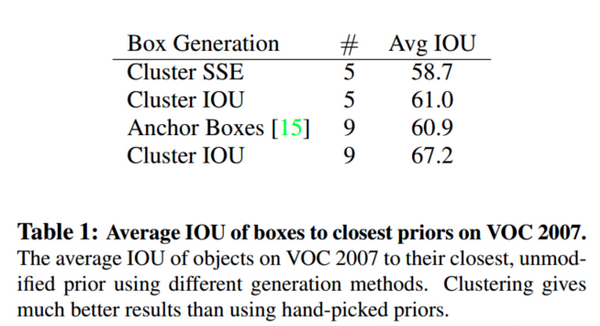
2018-02-27 14-29-11 的屏幕截图

作者通过改进的K-means对训练集中的boxes进行了聚类，判别标准是平均IOU得分，聚类结果如下图：



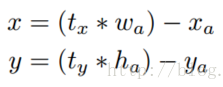
可以看到，平衡复杂度和IOU之后，最终得到k值为5，意味着作者选择了5种大小的box维度来进行定位预测，这与手动精选的box维度不同。结果中扁长的框较少，而瘦高的框更多（这符合行人的特征），这种结论如不通过聚类实验恐怕是发现不了的。

当然，作者也做了实验来对比两种策略的优劣，如下图，使用聚类方法，仅仅5种boxes的召回率就和Faster R-CNN的9种相当。说明K-means方法的引入使得生成的boxes更具有代表性，为后面的检测任务提供了便利。

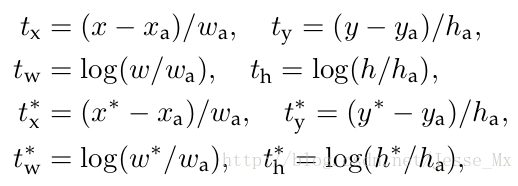


### 3.5 Direct location prediction（直接位置预测）

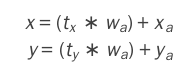
那么，作者在使用anchor boxes时发现的第二个问题就是：模型不稳定，尤其是在早期迭代的时候。大部分的不稳定现象出现在预测box的 (x,y) 坐标上了。在区域建议网络中，预测 (x,y) 以及 tx，ty 使用的是如下公式：



后来修改博文时，发现这个公式有误，作者应该是把加号写成了减号。理由如下，anchor的预测公式来自于Faster-RCNN，我们来看看人家是怎么写的：



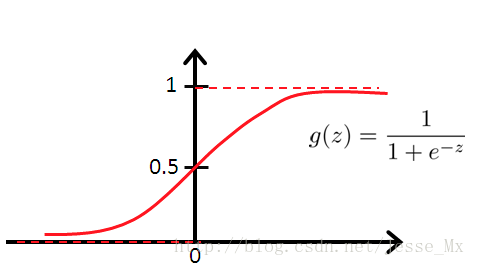
公式中，符号的含义解释一下：x 是坐标预测值，xa 是anchor坐标（预设固定值），x∗ 是坐标真实值（标注信息），其他变量 y，w，h 以此类推，t 变量是偏移量。然后把前两个公式变形，就可以得到正确的公式：



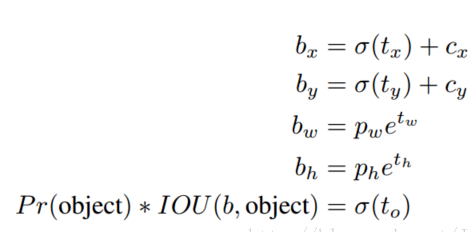
这个公式的理解为：当预测 tx=1，就会把box向右边移动一定距离（具体为anchor box的宽度），预测 tx=−1，就会把box向左边移动相同的距离。

这个公式没有任何限制，使得无论在什么位置进行预测，任何anchor boxes可以在图像中任意一点结束（我的理解是，tx 没有数值限定，可能会出现anchor检测很远的目标box的情况，效率比较低。正确做法应该是每一个anchor只负责检测周围正负一个单位以内的目标box）。模型随机初始化后，需要花很长一段时间才能稳定预测敏感的物体位置。

在此，作者就没有采用预测直接的offset的方法，而使用了预测相对于grid cell的坐标位置的办法，作者又把ground truth限制在了0到1之间，利用logistic回归函数来进行这一限制。



现在，神经网络在特征图（13 \*13 ）的每个cell上预测5个bounding boxes（聚类得出的值），同时每一个bounding box预测5个值，分别为 tx,ty,tw,th,to ，其中前四个是坐标，to是置信度。如果这个cell距离图像左上角的边距为 (cx,cy) 以及该cell对应box（bounding box prior）的长和宽分别为 (pw,ph)，那么预测值可以表示为：

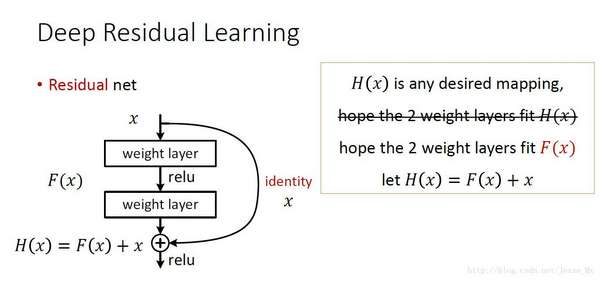


这几个公式参考上面Faster-RCNN和YOLOv1的公式以及下图就比较容易理解。tx,ty 经sigmod函数处理过，取值限定在了0~1，实际意义就是使anchor只负责周围的box，有利于提升效率和网络收敛。σ 函数的意义没有给，但估计是把归一化值转化为图中真实值，使用 e 的幂函数是因为前面做了 ln 计算，因此，σ(tx)是bounding box的中心相对栅格左上角的横坐标，σ(ty)是纵坐标，σ(to)是bounding box的confidence score。

定位预测值被归一化后，参数就更容易得到学习，模型就更稳定。作者使用Dimension Clusters和Direct location prediction这两项anchor boxes改进方法，mAP获得了5%的提升。

### 3.6 Fine-Grained Features（细粒度特征）

上述网络上的修改使YOLO最终在13 \* 13的特征图上进行预测，虽然这足以胜任大尺度物体的检测，但是用上细粒度特征的话，这可能对小尺度的物体检测有帮助。Faser R-CNN和SSD都在不同层次的特征图上产生区域建议（SSD直接就可看得出来这一点），获得了多尺度的适应性。这里使用了一种不同的方法，简单添加了一个转移层（ passthrough layer），这一层要把浅层特征图（分辨率为26 \* 26，是底层分辨率4倍）连接到深层特征图。



这个转移层也就是把高低两种分辨率的特征图做了一次连结，连接方式是叠加特征到不同的通道而不是空间位置，类似于Resnet中的identity mappings。这个方法把26 \* 26 \* 512的特征图连接到了13 \* 13 \* 2048的特征图，这个特征图与原来的特征相连接。YOLO的检测器使用的就是经过扩张的特征图，它可以拥有更好的细粒度特征，使得模型的性能获得了1%的提升。（这段理解的也不是很好，要看到网络结构图才能清楚）

补充：关于passthrough layer，具体来说就是特征重排（不涉及到参数学习），前面26 \* 26 \* 512的特征图使用按行和按列隔行采样的方法，就可以得到4个新的特征图，维度都是13 \* 13 \* 512，然后做concat操作，得到13 \* 13 \* 2048的特征图，将其拼接到后面的层，相当于做了一次特征融合，有利于检测小目标。

### 3.7 Multi-Scale Training

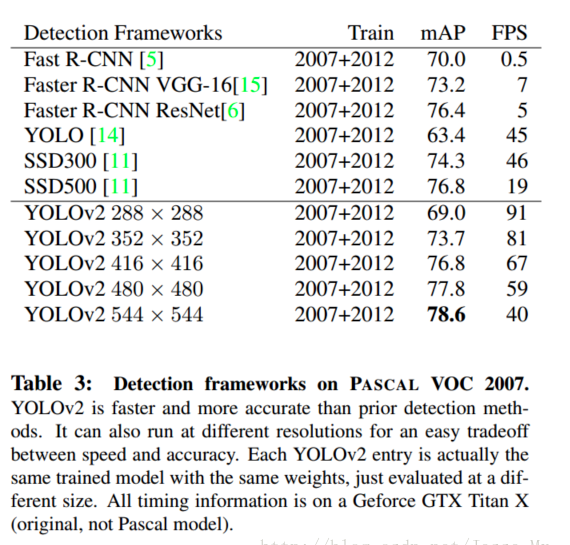
原来的YOLO网络使用固定的448 \* 448的图片作为输入，现在加入anchor boxes后，输入变成了416 \* 416。目前的网络只用到了卷积层和池化层，那么就可以进行动态调整（意思是可检测任意大小图片）。作者希望YOLOv2具有不同尺寸图片的鲁棒性，因此在训练的时候也考虑了这一点。

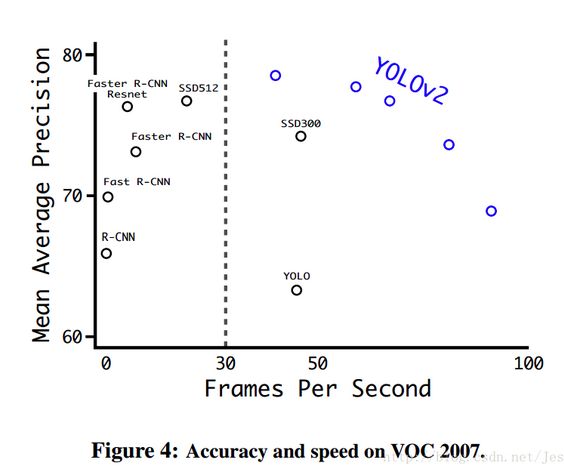
不同于固定输入网络的图片尺寸的方法，作者在几次迭代后就会微调网络。每经过10次训练（10 epoch），就会随机选择新的图片尺寸。YOLO网络使用的降采样参数为32，那么就使用32的倍数进行尺度池化{320,352，…，608}。最终最小的尺寸为320 \* 320，最大的尺寸为608 \* 608。接着按照输入尺寸调整网络进行训练。

这种机制使得网络可以更好地预测不同尺寸的图片，意味着同一个网络可以进行不同分辨率的检测任务，在小尺寸图片上YOLOv2运行更快，在速度和精度上达到了平衡。

在小尺寸图片检测中，YOLOv2成绩很好，输入为228 \* 228的时候，帧率达到90FPS，mAP几乎和Faster R-CNN的水准相同。使得其在低性能GPU、高帧率视频、多路视频场景中更加适用。

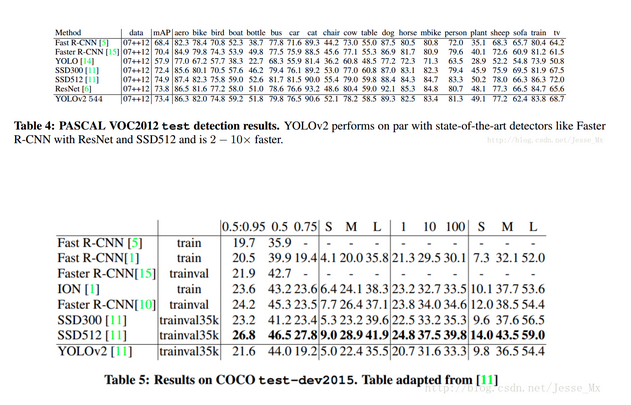
在大尺寸图片检测中，YOLOv2达到了先进水平，VOC2007 上mAP为78.6%，仍然高于平均水准，下图是YOLOv2和其他网络的成绩对比：





### 3.8 Further Experiments

作者做了进一步的实验，在VOC2012上对YOLOv2进行训练，下图是和其他方法的对比。YOLOv2精度达到了73.4%，并且速度更快。同时YOLOV2也在COCO上做了测试（IOU=0.5），也和Faster R-CNN、SSD作了成绩对比。总的来说，比上不足，比下有余。



## 4 YOLOv2速度的改进（Faster）

YOLO一向是速度和精度并重，作者为了改善检测速度，也作了一些相关工作。

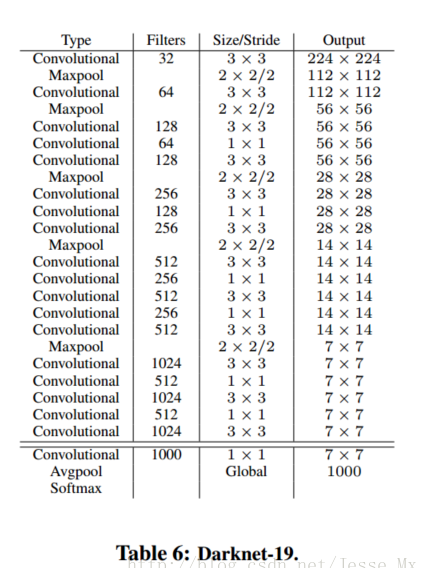
大多数检测网络有赖于VGG-16作为特征提取部分，VGG-16的确是一个强大而准确的分类网络，但是复杂度有些冗余。224 \* 224的图片进行一次前向传播，其卷积层就需要多达306.9亿次浮点数运算。

YOLOv2使用的是基于Googlenet的定制网络，比VGG-16更快，一次前向传播仅需85.2亿次运算。可是它的精度要略低于VGG-16，单张224 \* 224取前五个预测概率的对比成绩为88%和90%（低一点点也是可以接受的）。

### 4.1 Darknet-19

YOLOv2使用了一个新的分类网络作为特征提取部分，参考了前人的先进经验，比如类似于VGG，作者使用了较多的3 \* 3卷积核，在每一次池化操作后把通道数翻倍。借鉴了network in network的思想，网络使用了全局平均池化（global average pooling），把1 \* 1的卷积核置于3 \* 3的卷积核之间，用来压缩特征。也用了batch normalization（前面介绍过）稳定模型训练。

最终得出的基础模型就是Darknet-19，如下图，其包含19个卷积层、5个最大值池化层（maxpooling layers ），下图展示网络具体结构。Darknet-19运算次数为55.8亿次，imagenet图片分类top-1准确率72.9%，top-5准确率91.2%。



### 4.2 Training for classification

作者使用Darknet-19在标准1000类的ImageNet上训练了160次，用的随机梯度下降法，starting learning rate 为0.1，polynomial rate decay 为4，weight decay为0.0005 ，momentum 为0.9。训练的时候仍然使用了很多常见的数据扩充方法（data augmentation），包括random crops, rotations, and hue, saturation, and exposure shifts。 （这些训练参数是基于darknet框架，和caffe不尽相同）

初始的224 \* 224训练后，作者把分辨率上调到了448 \* 448，然后又训练了10次，学习率调整到了0.001。高分辨率下训练的分类网络在top-1准确率76.5%，top-5准确率93.3%。

### 4.3 Training for detection

分类网络训练完后，就该训练检测网络了，作者去掉了原网络最后一个卷积层，转而增加了三个3 \* 3 \* 1024的卷积层（可参考darknet中cfg文件），并且在每一个上述卷积层后面跟一个1 \* 1的卷积层，输出维度是检测所需的数量。对于VOC数据集，预测5种boxes大小，每个box包含5个坐标值和20个类别，所以总共是5 \* （5+20）= 125个输出维度。同时也添加了转移层（passthrough layer ），从最后那个3 \* 3 \* 512的卷积层连到倒数第二层，使模型有了细粒度特征。

作者的检测模型以0.001的初始学习率训练了160次，在60次和90次的时候，学习率减为原来的十分之一。其他的方面，weight decay为0.0005，momentum为0.9，依然使用了类似于Faster-RCNN和SSD的数据扩充（data augmentation）策略。

## 5 YOLOv2分类的改进（Stronger）

这一部分，作者使用联合训练方法，结合wordtree等方法，使YOLOv2的检测种类扩充到了上千种，具体内容待续