**1、Batch Normalization**

BN（Batch Normalization）层简单讲就是对网络的每一层的输入都做了归一化，这样网络就不需要每层都去学数据的分布，收敛会快点。作者在YOLOv2种为每个卷积层都添加了BN层，由于BN可以规范模型，所以加入BN后就把dropout去掉了，实验证明添加了BN层可以提高2%的mAP。

**2、High Resolution Classifier**

YOLOv1在预训练的时候用的是224\*224的输入，一般预训练的分类模型都是在ImageNet数据集上进行的，然后在检测的时候采用448\*448的输入。这会导致从分类模型切换到检测模型的时候，模型还要适应图像分辨率的改变。

YOLOv2中将预训练分成两步：先用224\*224的输入从头开始训练网络，大概160个epoch，然后再将输入调整到448\*448，再训练10个epoch。\*\*注意这两步都是在ImageNet数据集上操作。\*\*最后再在检测的数据集上fine-tuning，也就是检测的时候用448\*448的图像作为输入就可以顺利过渡了。

**3、Convolutional With Anchor Boxes**

YOLOv1是利用全连接层直接预测bounding box的坐标。  
YOLOv2则借鉴了Faster R-CNN的思想，引入anchor。

YOLOv2做了以下改变：

（1）删掉全连接层和最后一个pooling层，使得最后的卷积层可以有更高分辨率的特征；

（2）缩减网络，用416\*416大小的输入代替原来448\*448。

这样做是希望希望得到的特征图都有奇数大小的宽和高，奇数大小的宽和高会使得每个特征图在划分cell的时候就只有一个中心cell。因为大的目标一般会占据图像的中心，所以希望用一个中心cell去预测，而不是4个中心cell。网络最终将416\*416的输入下采样32倍变为13\*13大小的feature map输出

YOLOv1中将输入图像分成7\*7的网格，每个网格预测2个bounding box，一共只有7\*7\*2=98个box。

YOLOv2中引入anchor boxes，输出feature map大小为13\*13，每个cell有5个anchor box预测得到5个bounding box，一共有13\*13\*5=845个box。增加box数量是为了提高目标的定位准确率。

**4、Dimension Clusters**

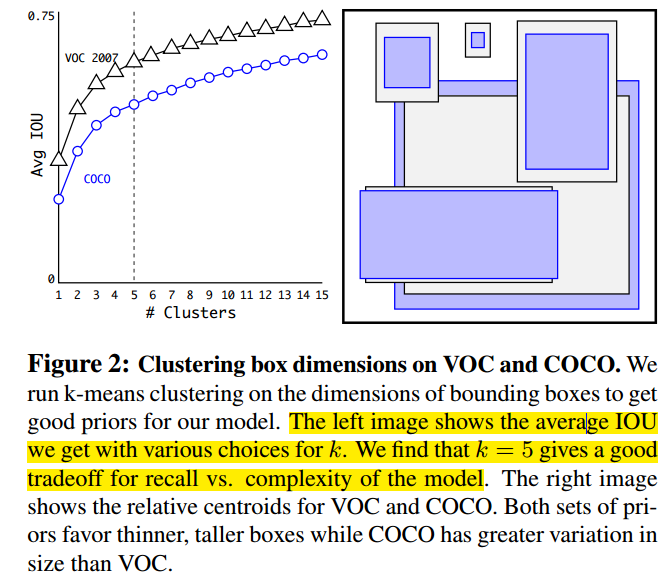
Faster R-CNN中anchor box的大小和比例是按经验设定的，然后网络会在训练过程中调整anchor box的尺寸。

如果一开始就能选择到合适尺寸的anchor box，那肯定可以帮助网络更好地预测。所以作者采用k-means的方式对训练集的bounding boxes做聚类，试图找到合适的anchor box。

作者发现采用标准的k-means（即用欧式距离来衡量差异），在box的尺寸比较大的时候其误差也更大，而我们希望的是误差和box的尺寸没有太大关系。所以通过IOU定义了距离函数，使得误差和box的大小无关：设置先验框的主要目的是为了使得预测框与ground truth的IOU更好，所以聚类分析师使用box与聚类中的box之间的IOU值作为距离指标。

距离函数：

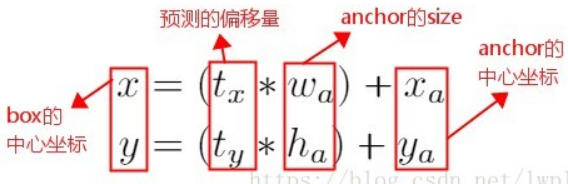




聚类的结果和手动设置的anchor box大小差别显著。聚类的结果中多是高瘦的box，而矮胖的box数量较少。

**5、Direct Location prediction**

在基于region proposal的目标检测算法中，是通过预测tx和ty来得到(x,y)值，也就是预测的是offsets。



这个公式是无约束的，预测的边界框很容易向任何方向偏移。

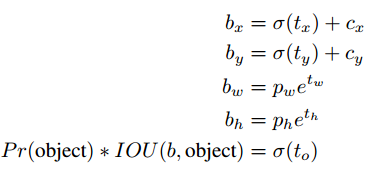
当tx=1时，box将向右偏移一个anchor box的宽度；

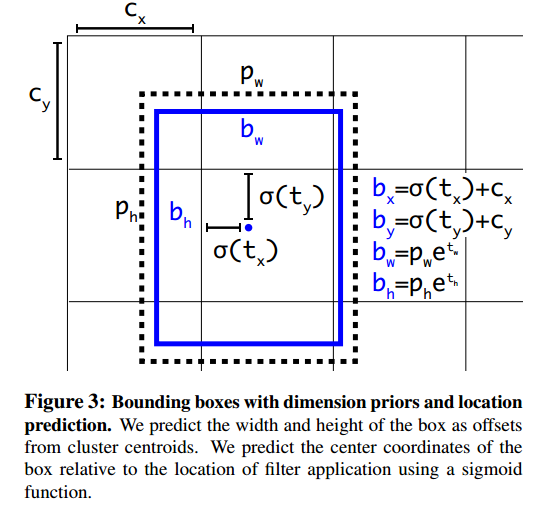
当tx=-1时，box将向左偏移一个anchor box的宽度；

因此，每个位置预测的边界框可以落在图片任何位置，这导致模型的不稳定性，在训练时需要很长时间来预测出正确的offsets

在这里作者并没有采用直接预测offset的方法，还是沿用了YOLO算法中直接预测相对于grid cell的坐标位置的方式。

前面提到网络在最后一个卷积层输出13\*13大小的feature map，然后每个cell预测5个bounding box，然后每个bounding box预测5个值：tx，ty，tw，th和to（这里的to类似YOLOv1中的confidence）。看下图，tx和ty经过sigmoid函数处理后范围在0到1之间，这样的归一化处理也使得模型训练更加稳定；cx和cy表示一个cell和图像左上角的横纵距离；pw和ph表示bounding box的宽高，这样bx和by就是cx和cy这个cell附近的anchor来预测tx和ty得到的结果。





**6、Fine-Grained Features**

细粒度特征：这里添加了一个直通层（passthrough layer），即就是源码中的reorg layer，将前面一层的26\*26的特征图和本层13\*13的特征图进行连接，与ResNet网络的shortcut类似，以前面更高分辨率的特征图为输入，然后将其连接到后面的低分辨率特征图上。

在13\*13的特征图上做预测，虽然对于大目标已经足够了，但对小目标不一定足够好，这里合并前面大一点的特征图可以有效的检测小目标。

**具体操作：**对于26\*26\*512的特征图，经passthrough层处理之后就变成了13\*13\*2048的新特征图（特征图大小变为1/4，而通道数变为以前的4倍），然后与后面的13\*13\*1024特征图连接在一起形成13\*13\*3072的特征图，最后在该特征图上卷积做预测。

**7、Multi-Scale Training**

YOLOv2中只有卷积层和池化层，因此不需要固定的输入图片的大小。

为了让模型更有鲁棒性，作者引入了多尺度训练。就是在训练过程中，每迭代一定的次数，改变模型的输入图片大小。

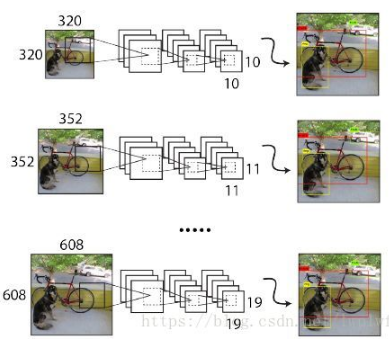
注意：这一步是在检测数据集上fine-tuning时候采用的，不要跟前面在Imagenet数据集上的两步预训练分类模型混淆。

具体操作：在训练时，每10个batch？（应该是10个epoch）网络就会随机选择另一种size的输入。

网络输入是416\*416，经过5次max pooling之后会输出13\*13的feature map，也就是下采样32倍，因此作者采用32的倍数作为输入的size，具体采用320、352、384、416、448、480、512、544、576、608共10种size。

输入图片大小为320\*320时，特征图大小为10\*10，输入图片大小为608\*608时，特征图大小为19\*19。

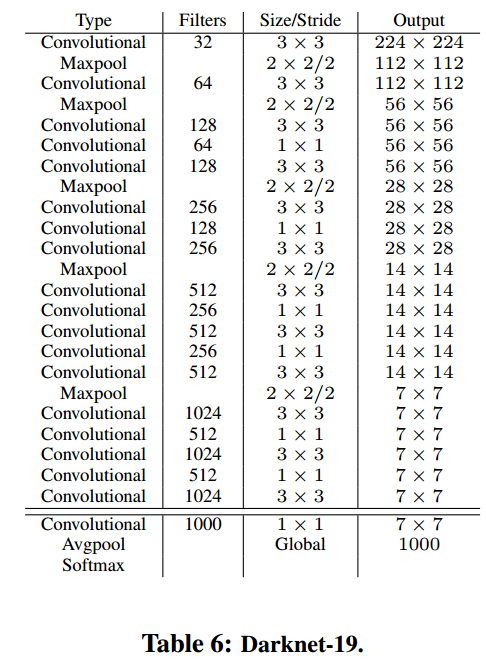
每次改变输入图片大小还需要对最后检测层进行处理，然后开始训练。



**Faster**

**1、Darknet-19**

网络包含19个卷积层和5个max pooling层，而在YOLOv1中采用的GooleNet，包含24个卷积层和2个全连接层，因此Darknet-19整体上卷积卷积操作比YOLOv1中用的GoogleNet要少，这是计算量减少的关键。最后用average pooling层代替全连接层进行预测。



**2、Training for Classification**

这里的Training for Classification都是在ImageNet上进行预训练。

YOLOv2的训练主要包括三个阶段：

**第一阶段：**在ImageNet分类数据集上从头开始预训练Darknet-19，训练160个epoch。输入图像的大小是224\*224，初始学习率为0.1。另外在训练的时候采用了标准的数据增加方式比如随机裁剪，旋转以及色度，亮度的调整等。

**第二阶段：**将网络的输入调整为448\*448，继续在ImageNet数据集上fine-tuning分类模型，训练10个epoch。参数的除了epoch和learning rate改变外，其他都没变，这里learning rate改为0.001。

**3、Training for Detection**

**第三阶段：**修改Darknet-19分类模型为检测模型，并在检测数据集上继续fine-tuning网络。

网络修改包括：移除最后一个卷积层、global avgpooling层以及softmax层，新增了三个3\*3\*2014卷积层，同时增加了一个passthrough层，最后使用1\*1卷积层输出预测结果。输出通道数计算如下。

对于VOC数据，每个cell预测num=5个bounding box，每个bounding box有5个坐标值和20个类别值，所以每个cell有125个filter。即：filter\_num = num \* (classes + 5) = 5 \* (20 + 5) = 125

**YOLOv2的结构示意图如下**：

