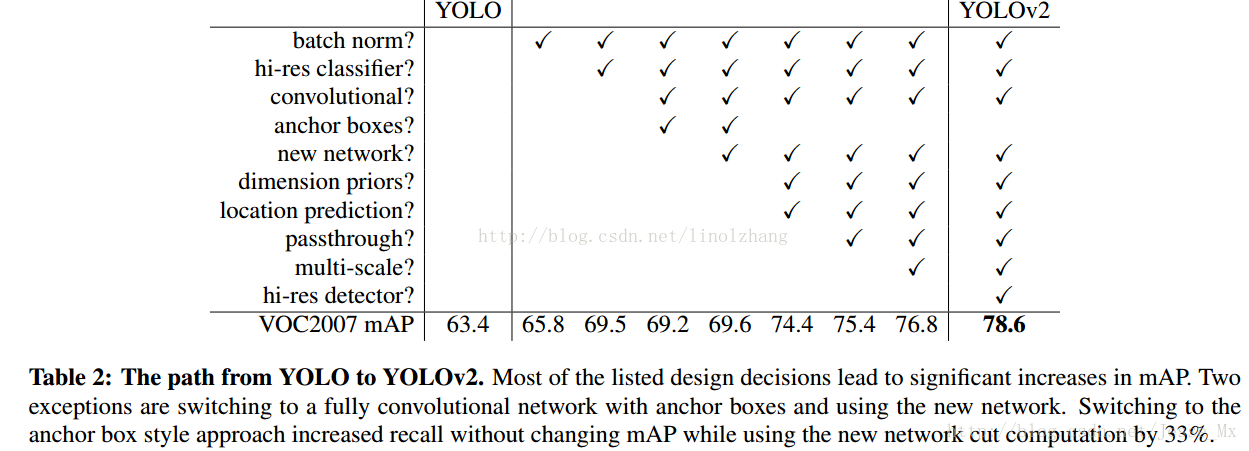
**YOLO-v2原文解读**

**一、久违的新版本**

       YOLO 问世已久，不过风头被SSD盖过不少，原作者自然不甘心，YOLO-v2 的提出给我们带来了什么呢？

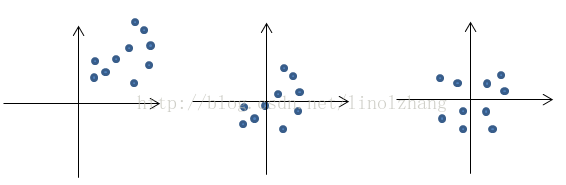
       先看一下其在 v1的基础上做了哪些改进，直接引用作者的实验结果了：



      条目不少，好多Trick，我们一个一个来看：

A、Batch Normalization（批量规范化）

先建立这样一个观点： 对数据进行预处理（统一格式、均衡化、去噪等）能够大大提高训练速度，提升训练效果。批量规范化，正是基于这个假设的实践，对每一层输入的数据进行加工。示意图：



Batch Normalization，简称 BN，由Google提出，是指对数据的归一化、规范化、正态化。BN 作为近几年最火爆的Trick之一，主流的CNN都已集成。

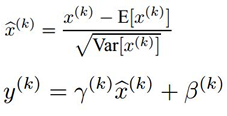
该方法的提出基于以下背景：

1）神经网络每层输入的分布总是发生变化，通过标准化上层输出，均衡输入数据分布，加快训练速度；可以设置较大的学习率和衰减，而不用去care初始参数，BN总能快速收敛，调参狗的福音。

2）通过规范化输入，降低激活函数在特定输入区间达到饱和状态的概率，避免 gradient vanishing 问题，举个例子：0.95^64≈0.0375，计算累积会产生数据偏离中心，导致误差的放大或缩小。

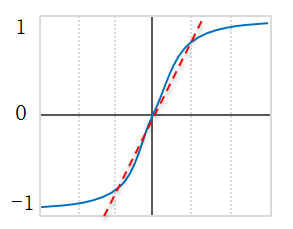
3）输入规范化对应样本正则化，在一定程度上可以替代 Drop Out；Drop Out的比例也可以被无视了，全自动的节奏。

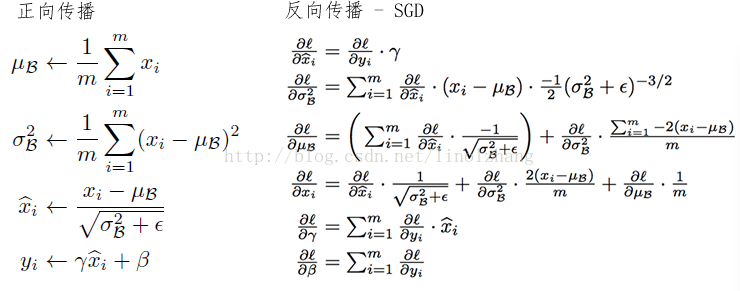
BN 的做法是在卷积池化之后，激活函数之前，对每个数据输出进行规范化（均值为 0，方差为 1）。



公式很简单，第一部分是Batch内数据归一化（其中 E为Batch均值，Var为方差），Batch数据近似代表了整体训练数据。

第二部分是亮点，即引入附加参数γ和β（Scale & Shift），Why？因为简单的归一化相当于只使用了激活函数中近似线性的部分（正则化是将所有数据集中到非线性部分，如下图红色虚线），破坏了原始数据的特征分布，这会降低模型表达能力。这两个参数需要通过训练得到：





关于这一层的函数定义、反向求导 等具体推理本章不再做进一步介绍，大家肯定可以搜到很多专门讲 BN的文献。

B、High Resolution Classifier

YOLO对应训练过程分为两步，第一步是通过 ImageNet 训练集 进行高分辨率的预训练，这一步训练的是分类网络；第二步是训练检测网络，是在分类网络的基础上进行fine-tune。

之前的 YOLO -v1以分辨率224\*224训练分类网络，YOLO -v2 将分类网络的分辨率提高到 448\*448，高分辨率样本对于效果有一定的提升（文中mAp提高了约4%）。高分辨率对于精度的提高是显而易见的，这点我们不去论证。

C、New Network（新网络）

为保证后续 Anchor Boxes 讲解的连续性，这里将New Network提前。

作者对网络进行了改进：

1）不同于SSD的VGG-16网络，作者采用的基础网络是Googlenet，并且加入了自己的订制，来看数据对比：

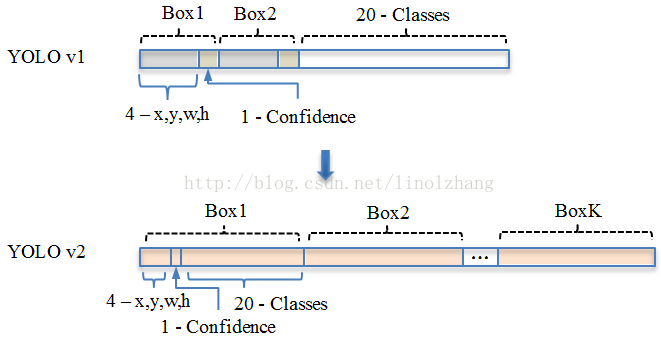
|  |  |  |
| --- | --- | --- |
|  | Googlenet | VGG-16 |
| 前向传播运算量（一次） | 85.2亿次 | 306.9亿次 |
| 精度（224 \* 224）  single-crop，top-5 accuracy | 88% | 90% |

整体来看，VGG-16整体精确度较高，但计算量过于复杂，性价比不高。

2）YOLO-v2采用了常用的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）做预测，把1\*1的卷积核置于3\*3的卷积核之间，用来压缩特征。YOLO-v2包含19个卷积层、5个最大值池化层（max pooling layers ）。

D、Convolutional With Anchor Boxes

Faster的 Anchor 机制又一次得到印证，与SSD一样，Anchor建立了和原始坐标的对应关系：定义了不同的Scale和宽高比，一个中心对应*K*个不同尺度和宽高比的Boxes。



YOLO-v1：S\*S\* (B\*5 + C) => 7\*7（2\*5+20）。其中B对应Box数量，5对应 Rect 定位+置信度。每个Grid只能预测对应两个Box，这两个Box共用一个分类结果（20 classes），这是很不合理的临时方案，看来作者为第二篇论文预留了改进，没想被 SSD 抢了风头。

YOLO-v2：S\*S\*K\* (5 + C) => 13\*13\*9（5+20）。分辨率改成了13\*13，更细的格子划分对小目标适应更好，再加上与Faster一样的K=9，计算量增加了不少。通过Anchor Box改进，mAP由69.5下降到69.2，Recall由81%提升到了88%。

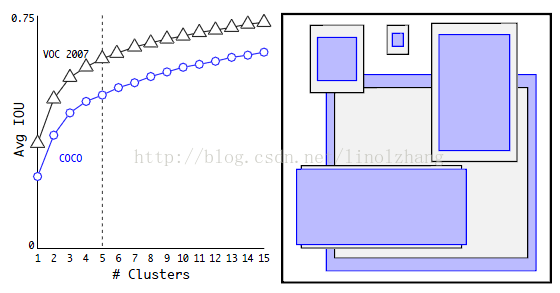
SSD：S\*S\*K\*(4 + C) => 7\*7\*6\*( 4+21 )。对应C=21，代表20种分类类别和一种背景类。

E、Dimension Clusters（维度聚类）

还是针对Anchors，Faster的Anchor对应 K=9，那么为什么等于9呢？宽高比为什么定位成这样（1:1,1:2,2:1）？对于SSD选择了K=6，那么K到底等于几合适？宽高比又该怎么设计？ 作者给出了解决方案，这个解决方案就是聚类。作者在 VOC和COCO上通过Ground Truth进行聚类统计（采用K-means算法），得到如下两个有用信息：

1）从K=1到K=5，IOU曲线上升较快（对应匹配度高），因此从效果和复杂度进行Trade Off， 选定了 Anchor Box个数为5；对应左图

2）统计发现，瘦高的框比扁平的框要多一些（人比车多？），选定了5种不同 宽高比+Scale 的 Anchor Box；对应右图。



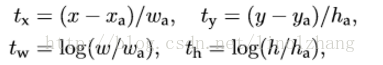
注：k-mans 采用的距离函数（度量标准） 描述为：

*d*(box, centroid) = 1 - IOU(box, centroid)

作者实验发现，5种boxes的Avg IOU(61.0)就和Faster R-CNN的9种Avg IOU(60.9)相当。说明K-means方法的生成的boxes更具有代表性。

F、Direct location prediction（直接位置预测）

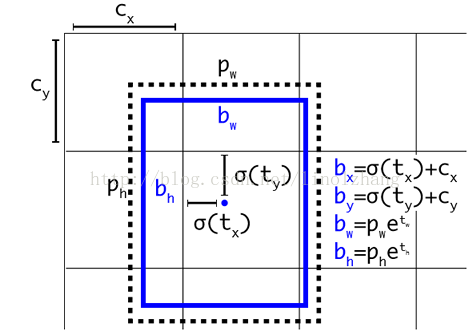
直接Anchor Box回归导致模型不稳定，对应公式也可以参考 Faster-RCNN论文，该公式没有任何约束，中心点可能会出现在图像任何位置，这就有可能导致回归过程震荡，甚至无法收敛：



针对这个问题，作者在预测位置参数时采用了强约束方法：

1）对应Cell距离左上角的边距为（*Cx*，*Cy*），*σ*定义为sigmoid激活函数，将函数值约束到［0，1］，用来预测相对于该Cell 中心的偏移（不会偏离cell）；

2）预定Anchor（文中描述为bounding box prior）对应的宽高为（*Pw*，*Ph*），预测 Location 是相对于Anchor的宽高乘以系数得到，如下图所示：



作者通过使用 维度聚类和直接位置预测这两项Anchor Boxes改进方法，将 mAP 提高了5%。

G、Fine-Grained Features（细粒度特征）

SSD通过不同Scale的Feature Map来预测Box来实现多尺度，而YOLO v2则采用了另一种思路：添加一个pass-through layer，来获取上一层26x26的特征，并将该特征同最后输出特征（13\*13）相结合，以此来提高对小目标的检测能力。通过Pass-through 把26 \* 26 \* 512的特征图叠加成13 \* 13 \* 2048的特征图，与原生的深层特征图相连接。YOLO-v2 使用扩展后的的特征图（add pass-through），将mAP提高了1%。

PS：这里实际上是有个Trick，网络最后一层是13\*13，相对原来7\*7的网络来讲，细粒度的处理目标已经double了，再加上一层26\*26的Feature共同决策，这两层的贡献等价于SSD的4层以上，但计算量其实并没有增加多少。

H、Multi-Scale Training（多尺度训练）

为了让 YOLO-v2 适应不同Scale下的检测任务，作者尝试通过不同分辨率图片的训练来提高网络的适应性。

PS：网络只用到了卷积层和池化层，可以进行动态调整（检测任意大小图片）

具体做法是：每经过10批训练（10 batches）就会随机选择新的图片尺寸，尺度定义为32的倍数，（ 320,352，…，608 ），为了最后一层得到特征图尺度为13\*13（416=13\*32），YOLO v2 输入图片尺寸为416 \* 416，降采样参数为32。

**二、训练过程**

作者采用的深度学习框架是Darknet，该框架作者使用很少，具体不作描述。

A、预训练 - 训练分类网络（Training for classiﬁcation）

采用随机梯度下降法SGD，在 ImageNet-1000分类数据集上训练了160个epochs，参数设定：

初始学习率 - starting learning rate：0.1

多项式速率衰减 - polynomial rate decay：4的幂次

权值衰减 - weight decay：0.0005

动量 - momentum：0.9

B、数据增广方法（Data augmentation）

采用了常见的data augmentation，包括：

随机裁剪、旋转 - random crops、rotations

色调、饱和度、曝光偏移 - hue、saturation、exposure shifts

C、多分辨率训练

通过初始的224 \* 224训练后，把分辨率上调到了448 \* 448，同样的参数又训练了10个epochs，然后将学习率调整到了10^{-3}。

D、训练检测网络 - Training for detection

把分类网络改成检测网络，去掉原网络最后一个卷积层，增加了三个 3 \* 3 （1024 ﬁlters）的卷积层，并且在每一个卷积层后面跟一个1 \* 1的卷积层，输出个数是检测所需要的数量。初始学习率为10^{-3}，训练了160个epochs（划分为60 | 10 | 90），权值衰减 与 momentum参数与前面一样。

**三、交叉数据集训练**

大家都知道，不同的数据集有不同的作用，通常我们采用一个数据集进行训练，而作者提出了新的思路：

通过ImageNet训练分类，COCO和VOC数据集来训练检测，这是一个很有价值的思路，可以让我们在公网上达到比较优的效果。 通过将两个数据集混合训练，如果遇到来自分类集的图片则只计算分类的Loss，遇到来自检测集的图片则计算完整的Loss。

这里面是有问题的，ImageNet对应分类有9000种，而COCO则只提供80种目标检测，这中间如何Match？答案就是multi-label模型，即假定一张图片可以有多个label，并且不要求label间独立。

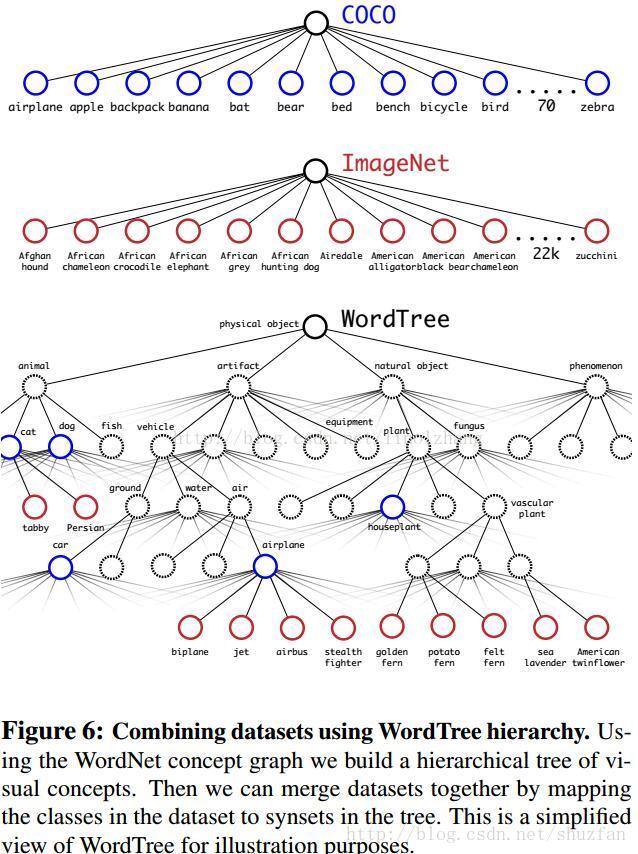
还是通过作者Paper里的图来说明，由于ImageNet的类别是从WordNet选取的，作者采用以下策略重建了一个树形结构（称为分层树）：

1）遍历Image-net的label，然后在WordNet中寻找该label到根节点(指向一个物理对象)的路径；

2）如果路径只有一条，那么就将该路径直接加入到分层树结构中；

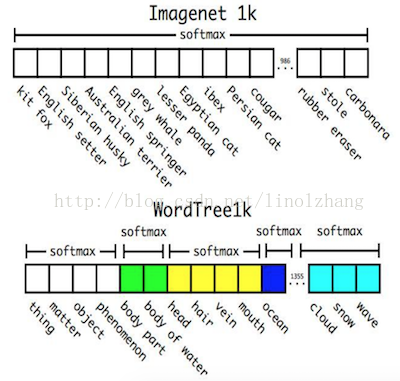
3）否则，从剩余的路径中选择一条最短路径，加入到分层树。

     这个分层树我们称之为 Word Tree，作用就在于将两种数据集按照层级进行结合。



      分类时的概率计算借用了决策树思想，某个节点的概率值等于 该节点到根节点的所有条件概率之积。

      另外，softmax操作也同时应该采用分组操作，下图上半部分为ImageNet对应的原生Softmax，下半部分对应基于Word Tree的Softmax：



       通过上述方案构造Word Tree，得到对应9418个分类，通过重采样保证Imagenet和COCO的样本数据比例为4:1（这个没有太明显的意义，你也可以改成6:1试试效果）。

四. 效果如何？

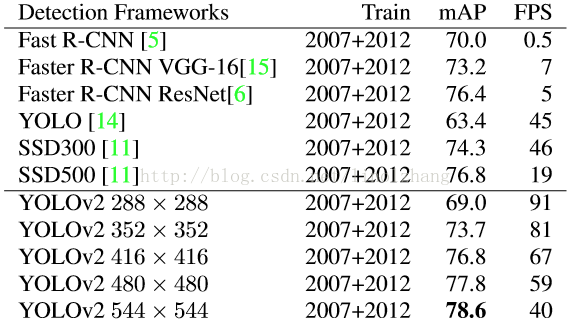
      YOLO-v2 在大尺寸图片上能够实现高精度，在小尺寸图片上运行更快，可以说在速度和精度上达到了平衡。

      总结下不同分辨率下的震撼的效果：

1）低分辨率 － 228 \* 228，帧率达到90FPS，mAP几乎与Faster媲美；

2）高分辨率，在VOC2007 上mAP达到78.6%，同时FPS＝40；

       看图说话：



       相比SSD，YOLO-v2添加了诸多工程Trick，虽然在算法理论上并没有明确的突破，但效果着实提升不少，相信实用性仍是我们的第一出发点，为作者点赞！