从YOLOv1到v3的进化之路

标签：gen bubuko AC 性能提升 hot 2.3 测试 区间 维度

引言：如今基于深度学习的目标检测已经逐渐成为自动驾驶，视频监控，机械加工，智能机器人等领域的核心技术，而现存的大多数精度高的目标检测算法，速度较慢，无法适应工业界对于目标检测实时性的需求，这时YOLO算法横空出世，以近乎极致的速度和出色的准确度赢得了大家的一致好评。基于此，我们选择YOLO算法来实现目标检测。YOLO算法目前已经经过了3个版本的迭代，在速度和精确度上获得了巨大的提升，我们将从YOLOV1开始讲起，直至目前最新的版本YOLOV3。

一、 YOLO V1 一步检测的开山之作

相对于传统的分类问题，目标检测显然更符合现实需求，因为往往现实中不可能在某一个场景只有一个物体，因此目标检测的需求变得更为复杂，不仅仅要求算法能够检验出是什么物体，还需要确定这个物体在图片哪里。

在这一过程中，目标检测经历了一个高度的符合人类的直觉的过程。既需要识别出目标的位置，将图片划分成小图片扔进算法中去，当算法认为某物体在这个小区域上之时，那么检测完成。那我们就认为这个物体在这个小图片上了。而这个思路，正是比较早期的目标检测思路，比如R-CNN。

后来的Fast R-CNN，Faster R-CNN[16]虽有改进，比如不再是将图片一块块的传进CNN提取特征，而是整体放进CNN提取特征图后，再做进一步处理，但依旧是整体流程分为 ‘区域提取’和‘目标分类’两部分（two-stage），这样做的一个特点是虽然确保了精度，但速度非常慢，于是以YOLO（You only look once）为主要代表的这种一步到位(one-stage)即端到端的目标检测算法应运而生了。

1.1 YOLO v1基本思想

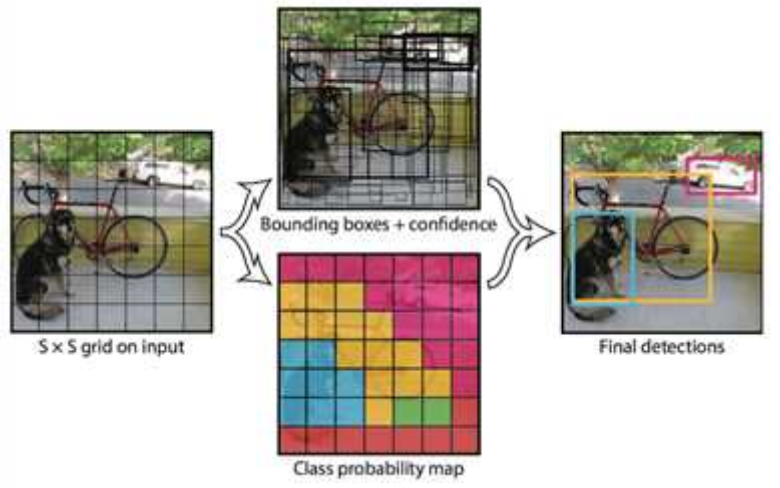
YOLO v1的核心思想在于将目标检测作为回归问题解决 ，YOLO v1首先会把原始图片放缩到448×448的尺寸，放缩到这个尺寸是为了后面整除来的方便。然后将图片划分成SxS个区域，注意这个区域的概念不同于上文提及将图片划分成N个区域扔进算法的区域不同。上文提及的区域是将图片进行剪裁，或者说把图片的某个局部的像素输入算法中，而这里的划分区域，只的是逻辑上的划分。

如果一个对象的中心落在某个单元格上，那么这个单元格负责预测这个物体。每个单元格需要预测B个边界框（bbox）值(bbox值包括坐标和宽高)，同时为每个bbox值预测一个置信度(confidence scores)。
此后以每个单元格为单位进行预测分析。

这个置信度并不只是该边界框是待检测目标的概率，而是该边界框是待检测目标的概率乘上该边界框和真实位置的IoU（框之间的交集除以并集）的积。通过乘上这个交并比，反映出该边界框预测位置的精度。如下式所示：

$$\text{confidence} = \text{P(Object)} \times IoU_{pred}^{truth}$$

每个边界框对应于5个输出，分别是x，y，w，h和置信度。其中x，y代表边界框的中心离开其所在网格单元格边界的偏移。w，h代表边界框真实宽高相对于整幅图像的比例。x，y，w，h这几个参数都已经被限制到了区间[0,1]上。除此以外，每个单元格还产生C个条件概率，。注意，我们不管B的大小，每个单元格只产生一组这样的概率。



图一：YOLO预测图示

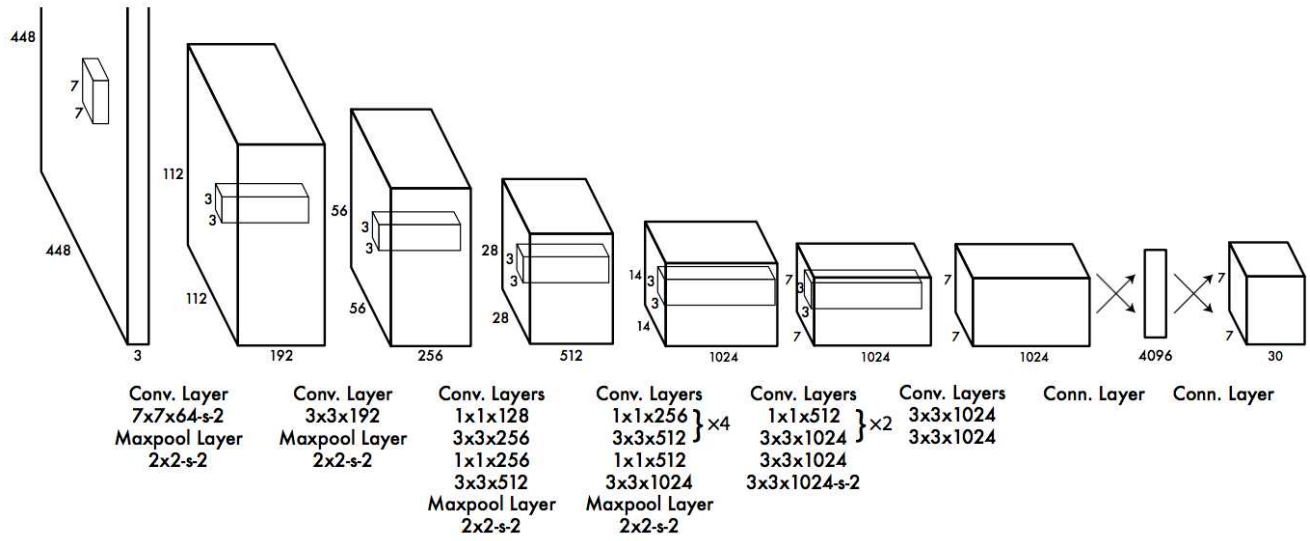
在test的非极大值抑制阶段，对于每个边界框，按照下式衡量该框是否应该予以保留。

$$\text{confidence} \times \text{P}(\text{Class}_i|\text{Object}) = \text{P}(\text{Class}_i) \times IoU_{pred}^{truth}$$

这就是每个单元格的个体分类置信度得分（class-specific confidence scores），这即包含了预测的类别信息，也包含了对bbox值的准确度。 我们可以设置一个阈值，把低分的class-specific confidence scores滤掉，剩下的留给给非极大值抑制，得到最终的标定框。

在PASCAL VOC进行测试时，使用S=7, B=2。由于共有20类，故C=20。所以，网络输出大小为7×7×30

1.2网络模型结构



图二：网络框架

该网络结构包括 24 个卷积层，最后接 2 个全连接层。Draknet[13]网络借鉴 GoogleNet 的思想，在每个1x1的卷积层之后再接一个3?3的卷积层的结构替代 GoogleNet 的Inception结构。论文中还提到了更快版本的 Yolo，只有 9 个卷积层，其他则保持一致。

1.3损失函数

YOLO v1全部使用了均方差（mean squared error）作为损失（loss）函数。由三部分组成：坐标误差、IOU误差和分类误差。

考虑到每种loss的贡献率，YOLO v1给坐标误差（coordErr）设置权重λcoord=5。在计算IoU误差时，包含物体的格子与不包含物体的格子，二者的IOU误差对网络loss的贡献值是不同的。若采用相同的权值，那么不包含物体的格子的置信度值近似为0，变相放大了包含物体的格子的置信度误差，在计算网络参数梯度时的影响。为解决这个问题，YOLO 使用λnoobj=0.5修正（置信度误差）iouErr。（此处的‘包含’是指存在一个物体，它的中心坐标落入到格子内）。

对于相等的误差值，大物体误差对检测的影响应小于小物体误差对检测的影响。这是因为，相同的位置偏差占大物体的比例远小于同等偏差占小物体的比例。YOLO将物体大小的信息项（w和h）进行求平方根来改进这个问题，但并不能完全解决这个问题。

综上，YOLO v1在训练过程中Loss计算如下式所示：

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

在激活函数上：

$$
\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}
$$

在最后一层使用的是标准的线性激活函数，其他的层都使用leaky rectified 线性激活函数。

1.4总结

YOLO v1作为一步检测的开山之作，最大的特点就是速度快。其将物体检测作为回归问题进行求解，使用单个网络完成整个检测的方法，大大提升了同类目标检测算法的速度，并且实现了召回率低，表现为背景误检率低

的有点。YOLO v1可以获取到图像的整体信息，相比于region proposal等方法，有着更广阔的“视野”。对其种类的物体，训练后识别效果也十分优异，具有很强的泛化能力。

但是YOLO v1的精准性和召回率相对于fast rcnn比较差。其对背景的误判率比Fast RCNN的误判率低很多。这说明了YOLO v1中把物体检测的思路转成回归问题的思路有较好的准确率，但是对于bounding box的定位不是很好。

二、 YOLOv2/YOLO9000 更准、更快、更强

YOLOv1对于bounding box的定位不是很好，在精度上比同类网络还有一定的差距，所以YOLOv2对于速度和精度做了很大的优化，并且吸收了同类网络的优点，一步步做出尝试。

YOLO V2在V1基础上做出改进后提出。其受到Faster RCNN方法的启发，引入了anchor。同时使用了K-Means方法，对anchor数量进行了讨论，在精度和速度之间做出折中。并且修改了网络结构，去掉了全连接层，改成了全卷积结构。在训练时引入了世界树（WordTree）结构，将检测和分类问题做成了一个统一的框架，并且提出了一种层次性联合训练方法，将ImageNet分类数据集和COCO检测数据集同时对模型训练。

2.1更准

YOLOv2对每批数据都做了一个归一化预处理。通过在每一个卷积层后添加batch normalization，极大的改善了收敛速度同时减少了对其它正则方法的依赖（舍弃了dropout优化后依然没有过拟合），使得mAP获得了2%的提升。（mAP：平均精度均值（mean Average Precision））

YOLOv1在分辨率为224×224的图片上进行预训练，在正式训练时将分辨率提升到448×448，这需要模型去适应新的分辨率。但是YOLOv2是直接使用448×448的输入，随着输入分辨率的增加，模型提高了4%的mAP.

在预测框的数量上，由于YOLOv2将网络的输入分辨率调整到416×416，保证为多次卷积后，下采样率为32，得到13×13的特征图（feature map）。在这上面使用9种anchor boxes[7]，得到13×13×9=1521个，这比YOLOv1大多了。

YOLOv1利用全连接层的数据完成边框的预测，会导致丢失较多的空间信息，使定位不准。在YOLOv2中作者借鉴了Faster R-CNN中的anchor思想，来改善全连接层带来的影响。

Anchor是RPN（region proposal network）网络在Faster R-CNN中的一个关键步骤，是在卷积特征图上进行滑窗操作，每一个中心可以预测9种不同大小的候选框。

为了引入anchor boxes来预测候选框，作者在网络中去掉了全连接层。并去掉了最后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，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中心框（center cell）。作者观察到，大物体通常占据了图像的中间位置，可以只用中心的一个框来预测这些物体的位置，否则就要用中间的4个格子来进行预测，这个技巧

可稍稍提升效率。最后，YOLOv2使用了卷积层降采样（采样因子为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）。

没有anchor boxes的情况下，模型召回率（recall）为81%，mAP为69.5%；加入anchor boxes，模型召回率为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%。

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

为了使网络更易学到准确的预测位置，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框宽高维度。传统的K-means聚类方法使用的是欧氏距离函数，也就意味着较大的框会比较小的框产生更多的误差，聚类结果可能会偏离。为此，作者采用IOU得分作为评价标准，这样的话，误差就和框的尺度无关了，最终的距离函数为：

对数据集的聚类结果如下:

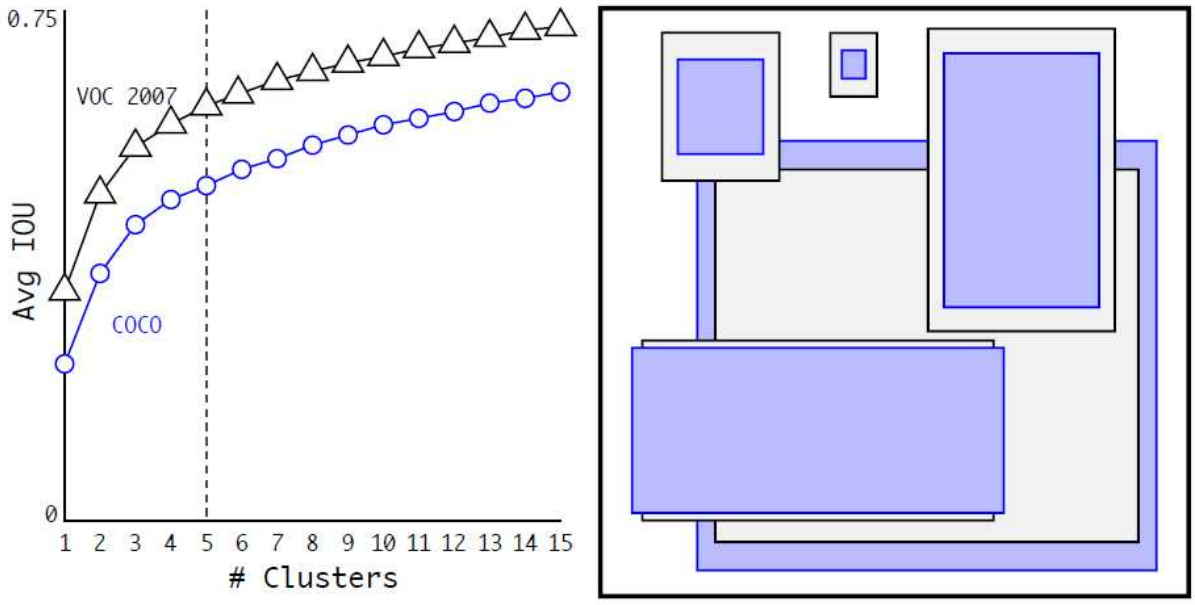


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

图三：聚类数目与Avg IoU的关系（使用VOC2007和COCO数据集）

可以看出k=5在模型复杂度与召回率之间取一个折中值。

在使用anchor的时候，遇到的第二个问题是加入anchor box的模型不稳定。作者认为模型不稳定的原因来自于预测bbox的(x，y)。如下：

$$x = (t_x * w_a) - x_a$$

$$y = (t_y * h_a) - y_a$$

在Faster R-CNN的预测中，偏移因子，是没有限制的，因此收敛会比较慢。故我们想让每个模型预测目标附近的一个部分，论文对采用了和YOLOv1一样的方法，直接预测中心点，并使用Sigmoid函数将偏移量限制在0到1之间(这里的尺度是针对网格框)。

计算公式如下:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$\Pr(\text{object}) * \text{IOU}(b, \text{object}) = \sigma(t_o)$$

bx,by,bw,bh，是预测的bbox的中心点坐标和宽高，中心点坐标的尺度是相对于网格。

如图四:

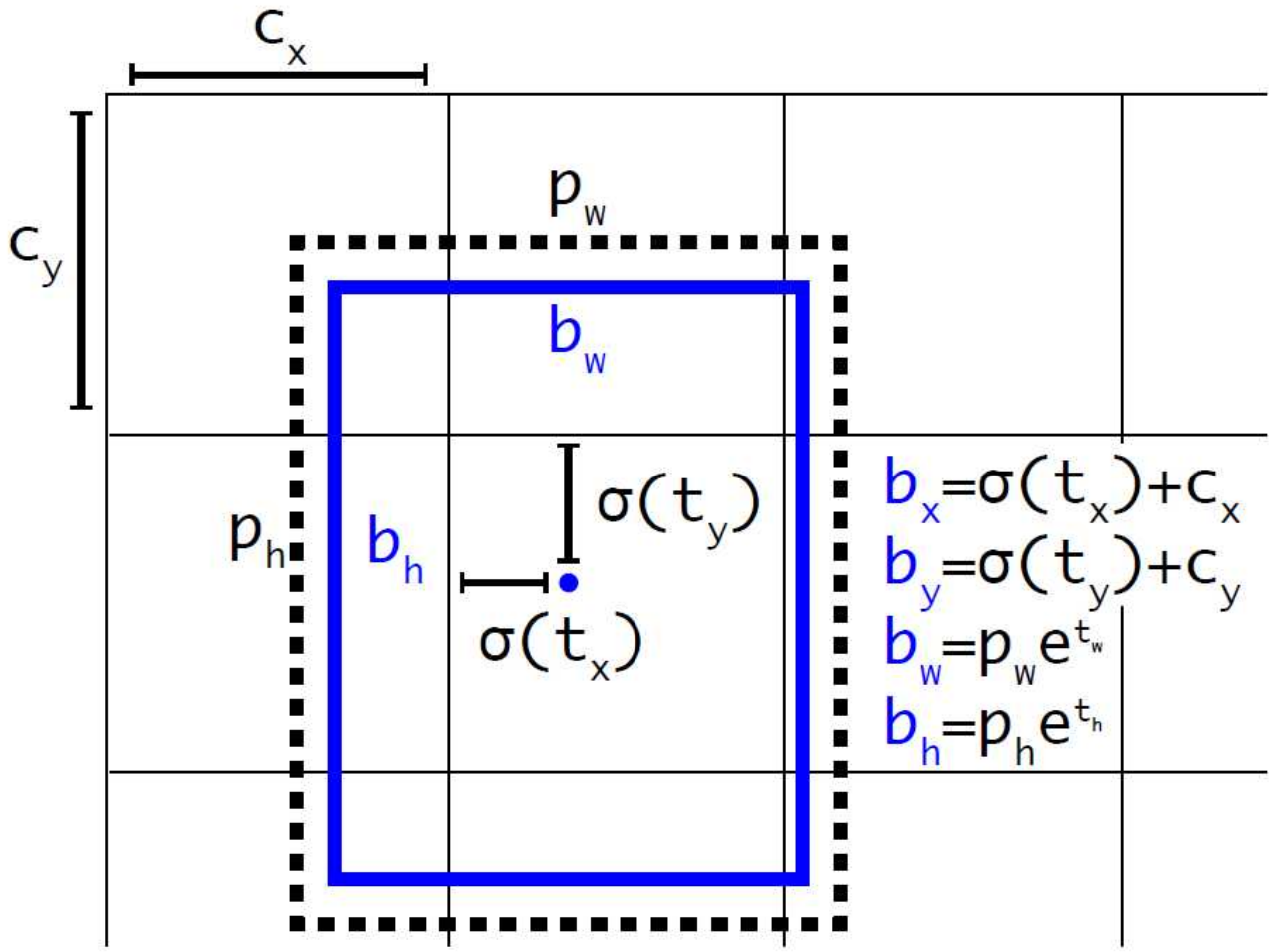


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

图四：各个参数的位置图示

经过维度聚类和直接位置预测的操作，在原有的anchor boxes版本上又提升了5%的mAP。

YOLOv1在对于大目标检测有很好的效果，但是对小目标检测上，效果欠佳。为了改善这一问题，作者参考了Faster R-CNN和SSD的想法，在不同层次的特征图上获取不同分辨率的特征。作者将上层的(前面26×26)高分辨率的特征图（feature map）直接连到13×13的feature map上。把26×26×512转换为13×13×2048，并拼接住在一起使整体性能提升1%。

Multi-Scale Training

和GoogleNet训练时一样，为了提高模型的鲁棒性（robust），在训练的时候使用多尺度[6]的输入进行训练。因为网络的卷积层下采样因子为32，故输入尺寸选择32的倍数288,352,…,544。

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

图五：不同尺度训练的精度与其他网络的精度对比

2.2更快

大多数目标检测的框架是建立在VGG-16上的，VGG-16在ImageNet上能达到90%的top-5（最后概率向量最大的前五名中出现了正确概率即为预测正确），但是单张图片需要30.69 billion 浮点运算，YOLO2是依赖于DarkNet-19的结构，该模型在ImageNet上能达到91%的top-5，并且单张图片只需要5.58 billion 浮点运算，大大的加快了运算速度。DarkNet的结构图如下：

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.

图六：YOLOv2网络结构

YOLOv2去掉YOLOv1的全连接层，同时去掉YOLO v1的最后一个池化层，增加特征的分辨率，修改网络的输入，保证特征图有一个中心点，这样可提高效率。并且是以每个anchor box来预测物体种类的。

作者将分类和检测分开训练，在训练分类时，以Darknet-19为模型在ImageNet上用随机梯度下降法（Stochastic gradient descent）跑了160epochs，跑完了160 epochs后，把输入尺寸从224×224上调为

448×448，这时候学习率调到0.001，再跑了10 epochs， DarkNet达到了top-1准确率76.5%，top-5准确率93.3%。

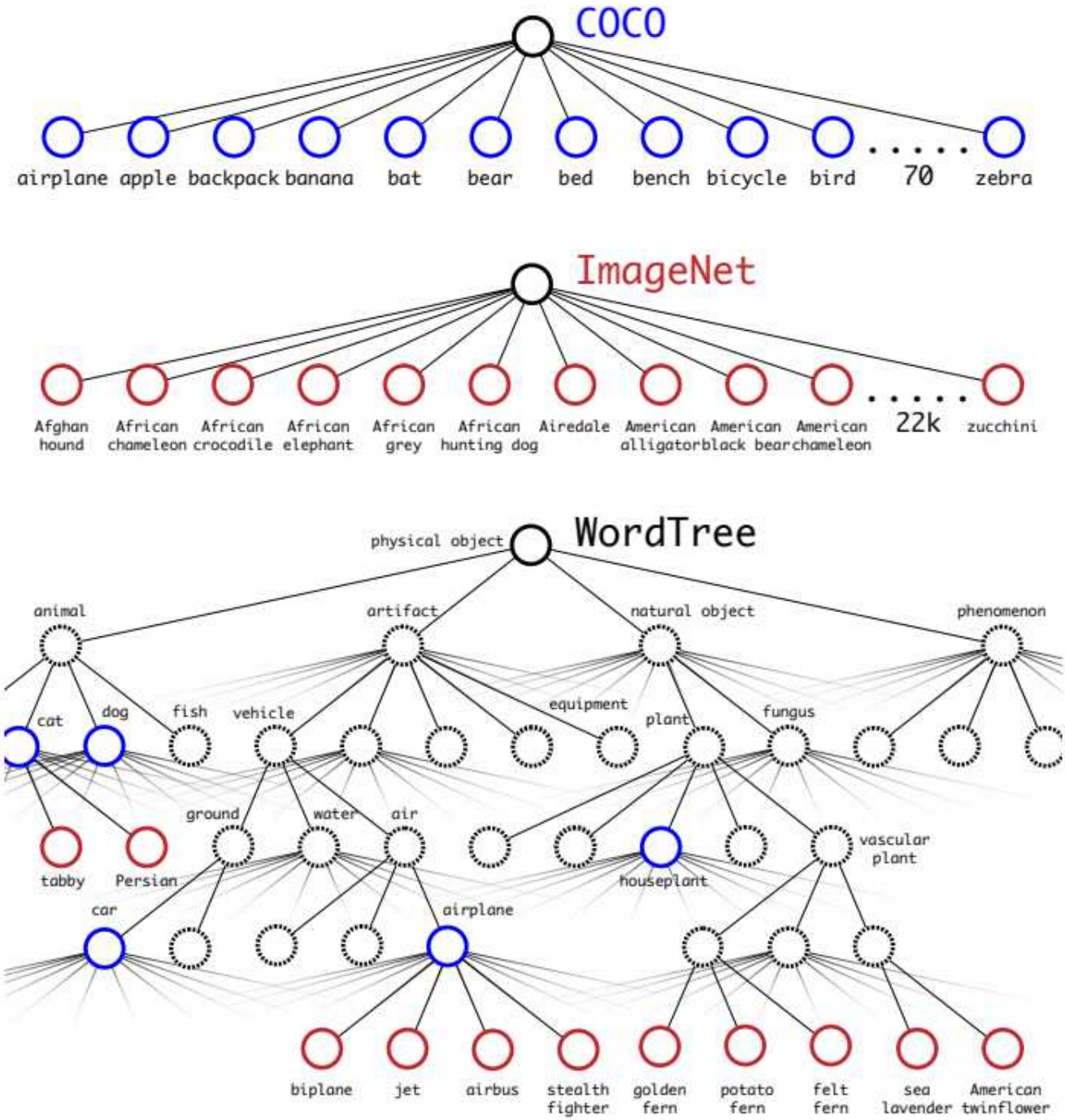
在训练检测时，作者把分类网络改成检测网络，去掉原先网络的最后一个卷积层，取而代之的是使用3个3×3x1024的卷积层，并且每个新增的卷积层后面接1×1的卷积层，数量是我们要检测的类的数量。

2.3更强

论文提出了一种联合训练的机制：使用识别数据集训练模型识别相关部分，使用分类数据集训练模型分类相关部分。

众多周知，检测数据集的标注要比分类数据集打标签繁琐的多，所以ImageNet分类数据集比VOC等检测数据集高出几个数量级。所以在YOLOv1中，边界框的预测其实并不依赖于物体的标签，YOLOv2实现了在分类和检测数据集上的联合训练。对于检测数据集，可以用来学习预测物体的边界框、置信度以及为物体分类，而对于分类数据集可以仅用来学习分类，但是其可以大大扩充模型所能检测的物体种类。

作者选择在COCO和ImageNet数据集上进行联合训练，遇到的第一问题是两者的类别并不是完全互斥的，比如"Norfolk terrier"明显属于"dog"，所以作者提出了一种层级分类方法（Hierarchical classification），根据各个类别之间的从属关系（根据WordNet）建立一种树结构WordTree，结合COCO和ImageNet建立的词树（WordTree）如下图所示：



图七：WordTree

WordTree中的根节点为"physical object"，每个节点的子节点都属于同一子类，可以对它们进行softmax处理。在给出某个类别的预测概率时，需要找到其所在的位置，遍历这个路径，然后计算路径上各个节点的概率之积。

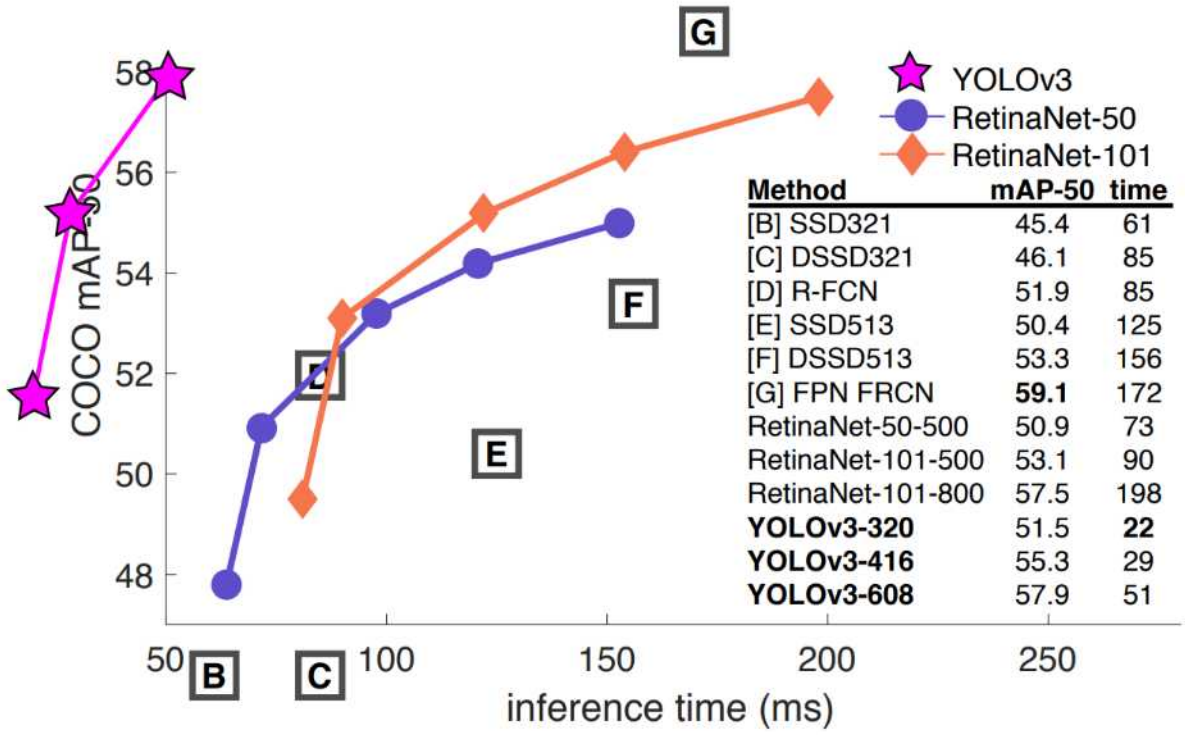
在训练时，如果是检测样本，按照YOLOv2的loss计算误差，而对于分类样本，只计算分类误差。在预测时，YOLOv2给出的置信度就是 ，同时会给出边界框位置以及一个树状概率图。在这个概率图中找到概率最高的路径，当达到某一个阈值时停止，就用当前节点表示预测的类别。

2.4总结

通过对YOLOv1网络结构和训练方法的改进，提出了YOLOv2/YOLO9000实时目标检测系统。YOLOv2在YOLOv1的基础上进行了一系列的改进，在快速的同时达到state of the art。同时，YOLOv2可以适应不同的输入尺寸，根据需要调整检测准确率和检测速度（值得参考）。作者综合了ImageNet数据集和COCO数据集，采用联合训练的方式训练，使该系统可以识别超过9000种物品。除此之外，作者提出的WordTree可以综合多种数据集的方法可以应用于其它计算机数觉任务中。但是对于重叠的分类，YOLOv2依然无法给出很好的解决方案。

三、 Yolov3集大成之作

YOLOv3是到目前为止，速度和精度最均衡的目标检测网络。通过多种先进方法的融合，将YOLO系列的短板（速度很快，不擅长检测小物体等）全部补齐。达到了令人惊艳的效果和拔群的速度。



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

图八：YOLOv3与其他网络的mAP与运行时间对比

3.1多标签分类预测

在YOLO9000[14]之后，我们的系统使用维度聚类（dimension clusters ）作为anchor boxes来预测边界框，网络为每个边界框预测4个坐标，，。

在YOLOv3[15]中使用逻辑回归预测每个边界框（bounding box）的对象分数。 如果先前的边界框比之前的任何其他边界框重叠ground truth对象，则该值应该为1。如果以前的边界框不是最好的，但是确实将ground truth对

象重叠了一定的阈值以上，我们会忽略这个预测，按照进行。我们使用阈值0.5。与YOLOv2不同，我们的系统只为每个ground truth对象分配一个边界框。如果先前的边界框未分配给grounding box对象，则不会对坐标或类别预测造成损失。

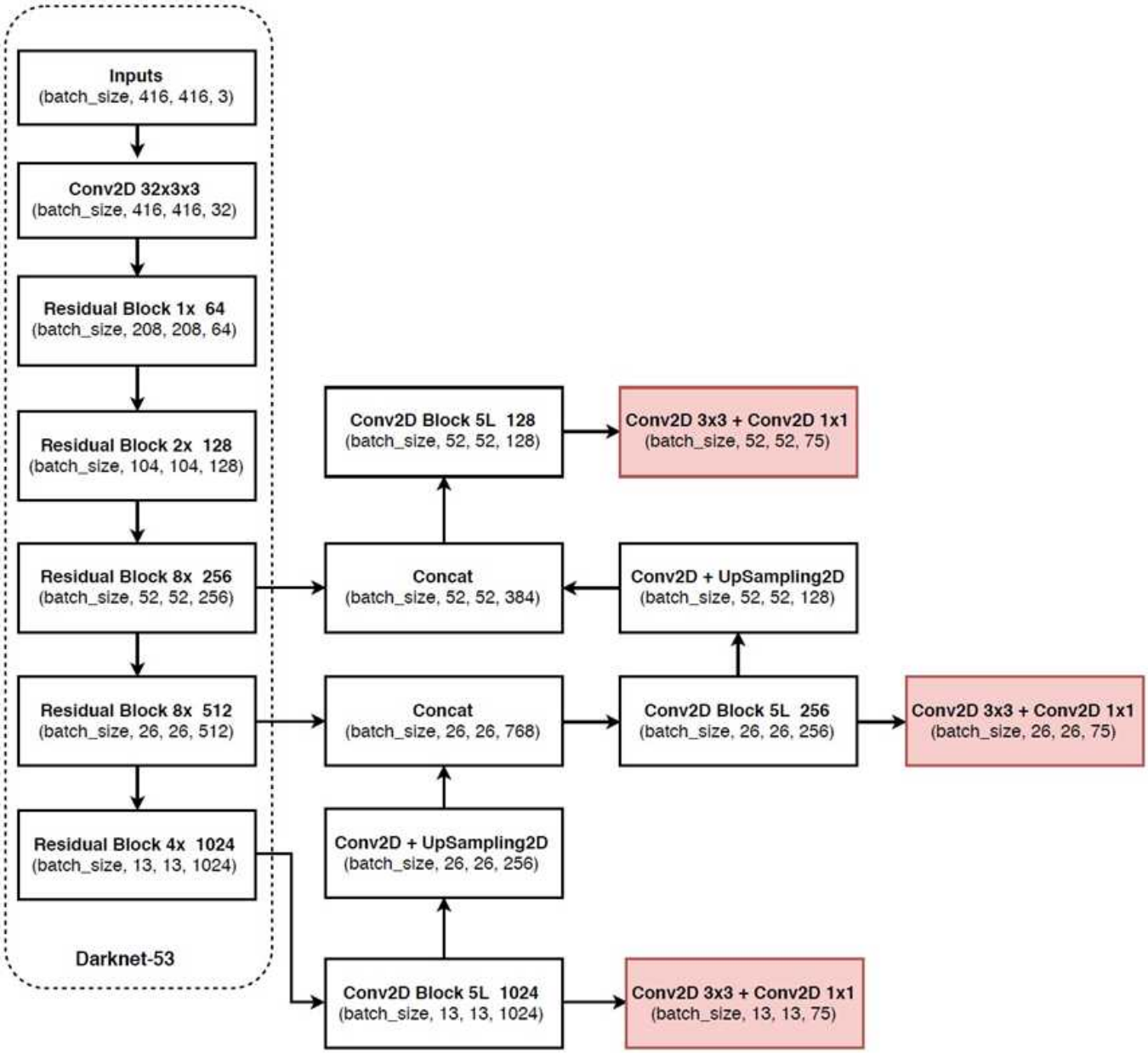
在YOLOv3中，每个框使用多标签分类来预测边界框可能包含的类。该算法不使用softmax，因为它对于高性能没有必要，因此YOLOv3使用独立的逻辑分类器。在训练过程中，我们使用二元交叉熵损失来进行类别预测。对于重叠的标签，多标签方法可以更好地模拟数据。

3.2跨尺度预测

YOLOv3采用多个尺度融合的方式做预测。原来的YOLO v2有一个层叫：passthrough layer，假设最后提取的feature map的size是13*13，那么这个层的作用就是将前面一层的26*26的feature map和本层的13*13的feature map进行连接，有点像ResNet。这样的操作也是为了加强YOLO算法对小目标检测的精确度。这个思想在YOLO v3中得到了进一步加强，在YOLO v3中采用类似FPN的上采样（upsample）和融合做法（最后融合了3个scale，其他两个scale的大小分别是26*26和52*52），在多个scale的feature map上做检测，对于小目标的检测效果提升还是比较明显的。虽然在YOLO v3中每个网格预测3个边界框，看起来比YOLO v2中每个grid cell预测5个边界框要少，但因为YOLO v3采用了多个尺度的特征融合，所以边界框的数量要比之前多很多。

3.3网络结构改变

YOLO v3使用新的网络来实现特征提取。在Darknet-19中添加残差网络的混合方式，使用连续的3×3和1×1卷积层，但现在也有一些shortcut连接，YOLO v3将其扩充为53层并称之为Darknet-53。



图九：Darknet-53网络结构

这个新网络比Darknet-19功能强大得多，而且比ResNet-101或ResNet-152更有效。

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

图十：ImageNet结果

每个网络都使用相同的设置进行训练，并以256×256的单精度测试进行测试。运行时间是在Titan X上以256×256进行测量的。因此，Darknet-53可与state-of-the-art的分类器相媲美，但浮点运算更少，速度更快。Darknet-53比ResNet-101更好，速度更快1：5倍。 Darknet-53与ResNet-152具有相似的性能，速度提高2倍。

Darknet-53也可以实现每秒最高的测量浮点运算。这意味着网络结构可以更好地利用GPU，从而使其评估效率更高，速度更快。

3.4总结

YOLO检测算法进行目标检测，取得了较高的检测速度和检测准确率。该算法不仅对于实物有着很好的效果，对于其他目标，如艺术作品等同样具有很好的兼容性。YOLO算法相比其他算法更符合工业界对目标检测算法实时性的要求，简单易实现，对于嵌入式很友好。

YOLO系列不断吸收目标检测同类算法的优点，将其应用于自身，不断进步，可谓成长中的算法。