目标检测综述

`YOLO` `RCNN` `SSD` `RetinaNet`

这篇综述是我统计信号处理的作业，在这里分享一下，将介绍计算机视觉中的目标检测任务，论述自深度学习以来目标检测的常见方法，着重讲yolo算法，并且将yolo算法与其他的one-stage以及two-stage方法进行比较。

目录

1. 介绍

目标检测在现实中的应用很广泛，我们需要检测数字图像中的物体位置以及类别，它需要我们构建一个模型，模型的输入一张图片，模型的输出需要圈出图片中所有物体的位置以及物体所属的类别，见图1。在深度学习浪潮到来之前，目标检测精度的进步十分缓慢，靠传统依靠手

工特征的方法来提高精度已是相当困难的事。而ImageNet分类大赛出现的卷积神经网络（CNN）——AlexNet1所展现的强大性能，吸引着学者们将CNN迁移到了其他的任务，这也包括着目标检测任务，近年来，出现了很多目标检测的方法，这里将介绍YOLO2-4，RCNN5-7，SSD8，RetinaNet9系列的方法，其中YOLO，SSD，RetinaNet都是one-stage方法，原始RCNN事multi-stage方法，它的延伸FastRCNN以及FasterRCNN则是two-stage方法。RCNN系列方法是先生成候选框，然后根据候选框来进行坐标回归预测，而YOLO，SSD，RetinaNet则是直接进行回归生成坐标回归，没有经过候选框这一步。

目标检测12

2.YOLO

这一节将开始详细地叙述YOLO三部曲，先介绍YOLOv1，更好更简单地了解YOLO的原理和思想，然后分别介绍YOLOv2以及YOLOv3，看看这两个模型是使用那些技巧来提升性能。

2.1 YOLOv1

流程：首先YOLOv1会把图像看成一个sxs的栅格，这里的s是等于7，每个栅格预测2个bounding boxes以及栅格含有对象的置信度，同时每个栅格还是预测栅格所属的对象类别，然后通过一些处理方式得到最后的结果，这个处理方式后面会讲到。

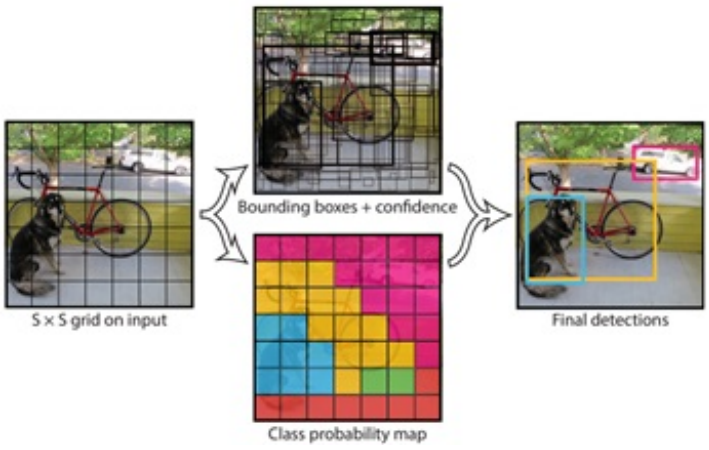

图2 YOLOv1流程

架构：然后，我们来看看YOLOv1的架构，YOLOv1由24层卷积层，4个最大池化层和2个全连接层组成，常规操作，我们关注最后的输出是7x7x30，这里是7x7代表输入图像的7x7栅格，一一对应，30的前十个代表2个bounding boxes的坐标以及对象的置信度，后20个代表VOC数据集的20个类别。

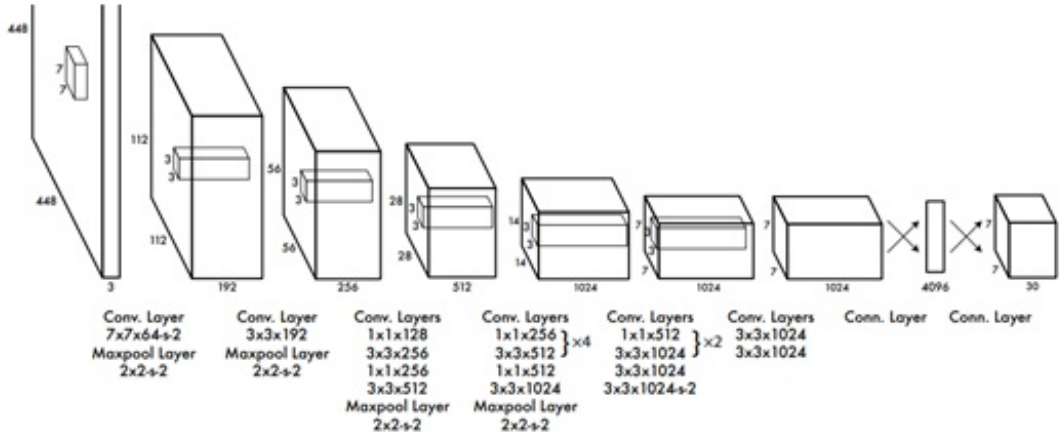

图3 YOLOv1架构

标签定义：YOLOv1是一个监督式的网络，有监督那就有标签，来看看标签是怎样定义，首先看狗，它被一个框框起来了，这个框就是真实的标签，框对应的中心在哪个栅格，就代表当前栅格是狗所在的栅格，这个栅格里就会记录狗的标签信息，自行车和小车和狗类似。

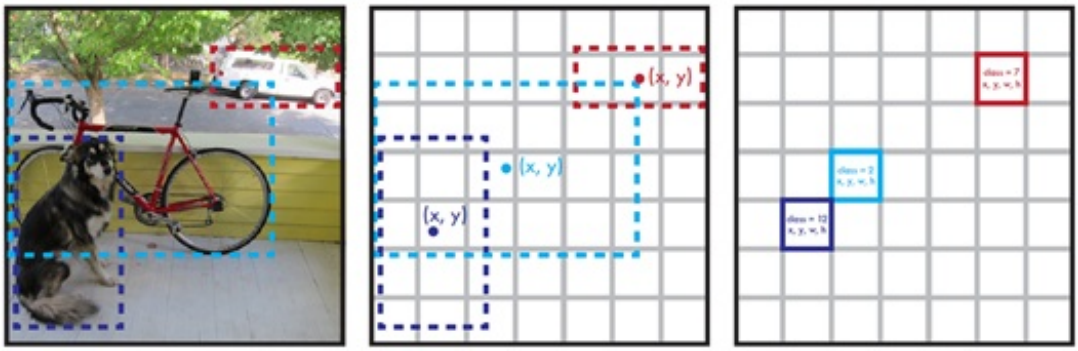

图4 标签定义

损失函数：首先，这个λ是bounding box坐标损失的权重，外层求和是多少个栅格，内层求和是每个栅格的B个Boxes，这个像一的符号，它代表当前box中是否含有真实标签对象，坐标预测我们只计算有对象的栅格，其他的栅格不进行计算，这个Ci代表当前栅格含有对象的概率，不光要计算含有对象的，也要计算没有含有对象的，最后的类别，只计算含有对象的栅格，没有包含对象的不考虑。根据这个损失进行反向传播，一步步优化YOLOv1模型。

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

图5 损失函数

交并比（IOU）：这是一个评价两个bounding box相互重合程度的指标，这个指标等于两个bounding box的交集面积除以它们并集的面积。当两个bounding box没有任何交集时，IoU为0，即IoU的最小取值，当两个bounding box完全重合时，IoU为1，即IoU的最大取值，所以IoU的取值范围是[0,1]。

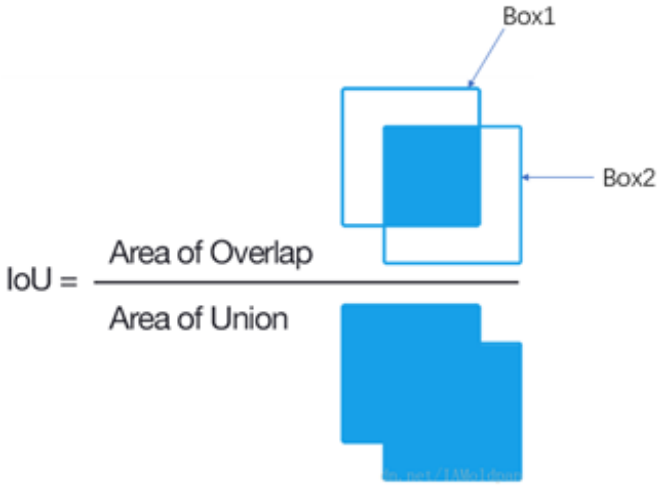

图6 交并比13

推断：给定一张图，运行YOLO后，总共有98个bounding box输出出来，可以通过非极大值抑制算法得到最后可靠的结果。大致分两步，第一步根据阈值去除那些置信度低的bounding box，然后进入一个循环，首先挑选出最大置信度的bounding box作为预测输出，然后去除那些与这个最大置信度的bounding box的IoU超过0.5的bounding box，因为我们可以看到一个对象有很多bounding box，它们很多是相交的，这样一个对象的bounding box就确定好了，然后，我们再进入循环，找出下一个对象的bounding box，最后直到没有剩余的bounding box，循环结束。

2.2 YOLOv2

从三个方面开始介绍YOLOv2，Better，Faster，Stronger。
Better：批归一化，高分辨分类器，锚盒，维度聚类，细粒度特征以及多尺度训练。

批归一化14（Batch Normalization）的效果则是将数据分布映射到相对紧凑的分布，让网络可以更快以及更好地学习，避免过拟合，使用批归一化这一操作提升了2%mAP。

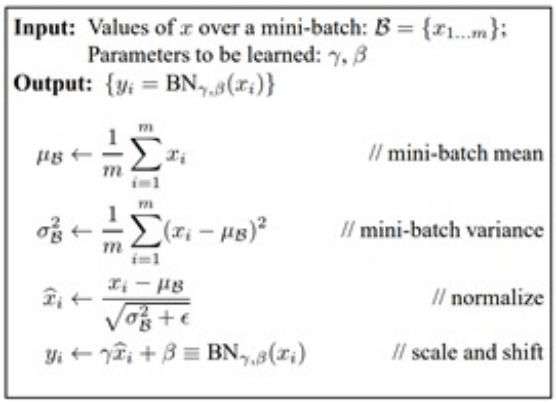

图7 批归一化

YOLOv2采用了高分辨率的分类器，在YOLOv1中，将在imagenet预训练好的网络迁移到目标检测网络中，而训练imagenet数据集的输入的大小和目标检测网络的输入尺寸是不一致的，这意味目标检测网络需要同时取学习目标检测而且还要去适应新的分辨率输入，所以YOLOv2使用目标检测输入的分辨率微调了分类网络，然后迁移到目标检测网络中去，这样，目标检测网络就可以专攻学习目标检测了，通过这一技巧，使mAP上升了4%。

在YOLOv1中，最后是使用全连接层来生成bounding box的坐标，然而使用全连接的缺点在于丢失了特征图的空间信息，造成定位不准，作者借鉴了Faster Rcnn中锚框的思想，利用锚框直接在卷积特征图滑窗采样，因为卷积不需要Reshape，所以很好的保留的空间信息，最终使特征图的每个特征点和原图的每个栅格一一对应。另外，与YOLOv1不同的是，YOLOv2是预测的是坐标相对于栅格左顶点的偏移量，通过变换公式得到最后的预测坐标。既然锚框是手动精选的先验框，设想能否一开始就选择了更好的、更有代表性的先验Boxes维度，那么网络就应该更容易学到准确的预测位置。所以作者通过K-means聚类算法将数据集中的ground truth进行了聚类。最后对模型复杂度和召回率的平衡，选择5个聚类中心，挑选了5个最具代表性的bounding box。一起提升了5%mAP。

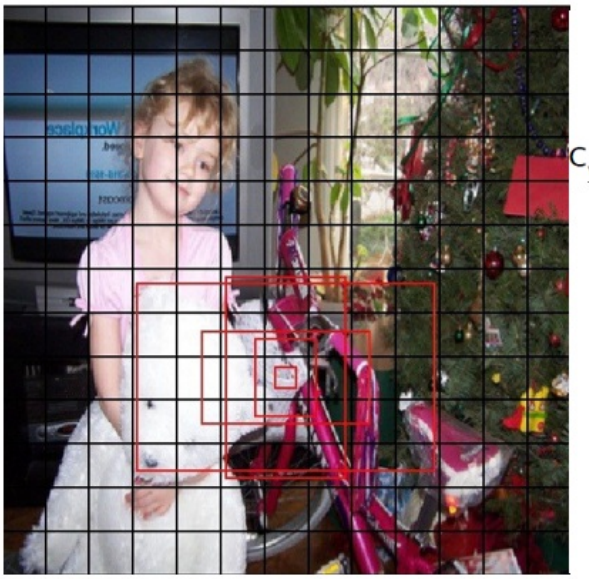

图 8 锚框[15]

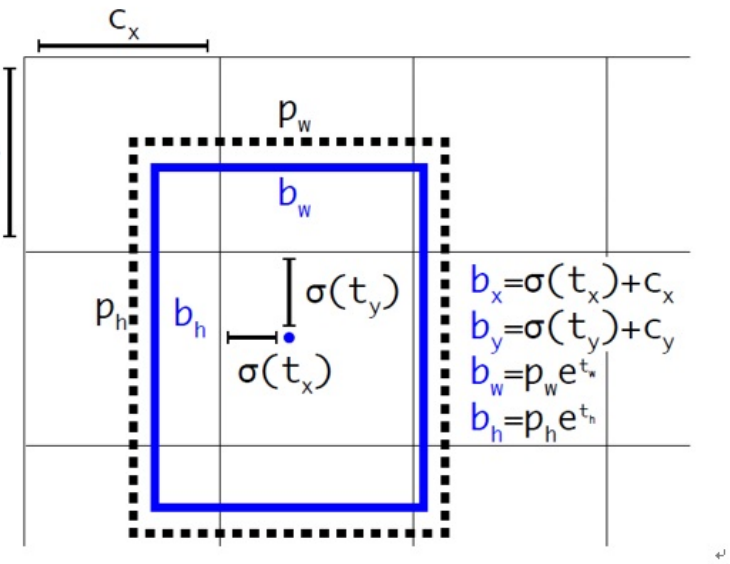

图 9 标签与锚框的相对偏移

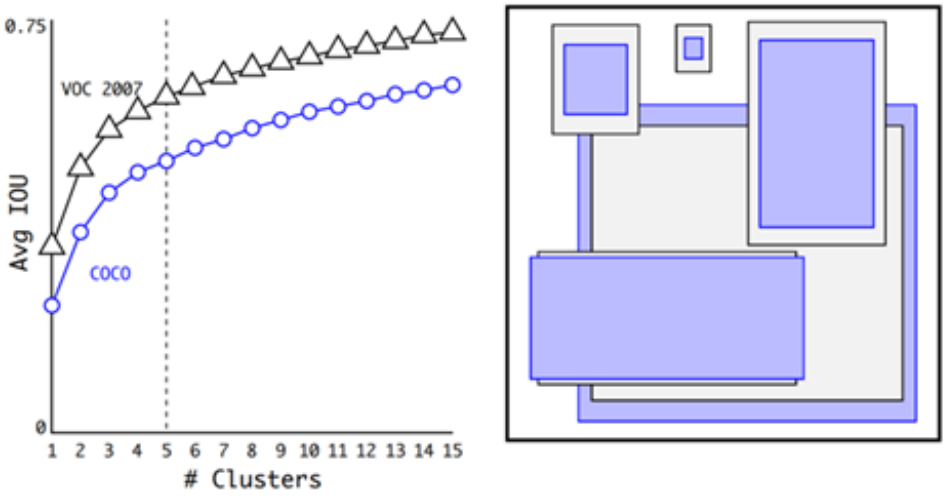

图10 锚盒K-means聚类

细粒度特征对于检测小物体是有很大的影响，随着图像尺寸的一步步缩小，图片中的某些小物体的特征是会由于特征图的不断缩小而发生信息丢失，作者通过引入了一个Passthrough Layer，把浅层特征图连接到深层特征图，也就是图中这个26x26x512的特征图通过隔行隔列采样，变换成13x13x2048的特征图，然后和13x13x1024的特征图进行按通道concat，通过这一操作，可以使mAP提升一个百分点。

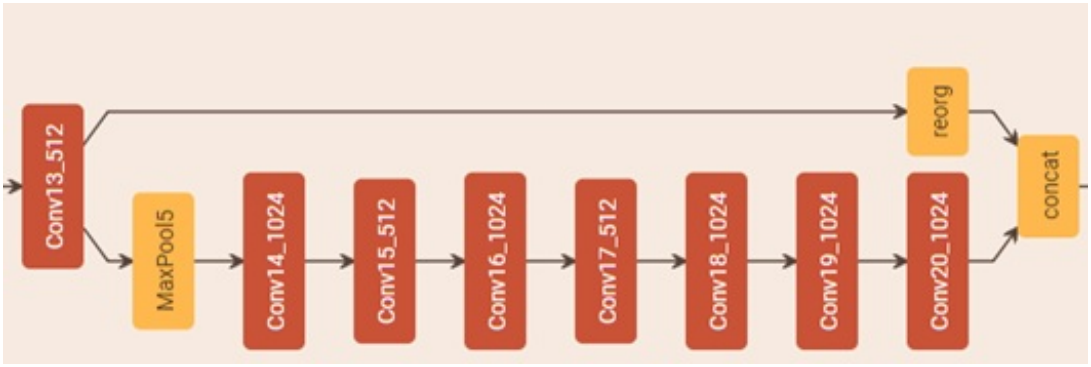

图11 细粒度特征的引入16

然后使为了让YOLOv2对不同尺寸图片的具有鲁棒性，引入了多尺寸的训练，每10batch，选择新的图像尺寸对网络进行训练，最后使精度提升了2个百分点。

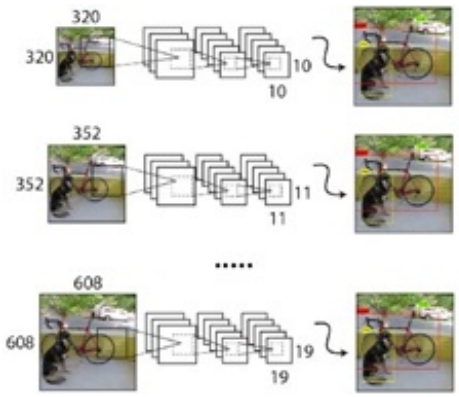

图12 多尺度训练

通过一系列的改进，YOLOv2相比于YOLOv1，一下子提高了15个点。

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

图13 better总览图

Faster：YOLOv2简化了网络，只使用了19卷积层和5个池化层（Darknet-19），不仅通过一系列的改进使精度高了，速度方面依然比YOLOv1还要快。

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 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 × 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			

图14 YOLOv2

Stronger：强大之处体现在YOLO9000，YOLO9000是在YOLOv2的基础上提出的一种可以检测超过9000个类别的模型，其主要贡献点在于提出了一种分类和检测的联合训练策略。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。但是遇到的一个问题是两个数据集的类别不是完全互斥的，比如ImageNet光狗的品种就有100多种，它们与COCO数据集中狗的类别产生冲突，两者是包含关系，针对这一问题，作者提出了一种层级分类方法，主要思路是根据各个类别之间的从属关系建立一种树结构，也就是WordTree。

论文中给出了COCO数据集和ImageNet数据集联合生成的树结构，蓝色的是COCO数据集的类别，橘色的是imageNet的类别，图15给出一个例子，比如imagenet有这两种不同类型的小猎狗，它们输入小猎狗这一分支，也属于猎狗分支，还属于COCO数据集中狗这一分支，还属于犬类这一分支。这就是wordtree的构造形式。

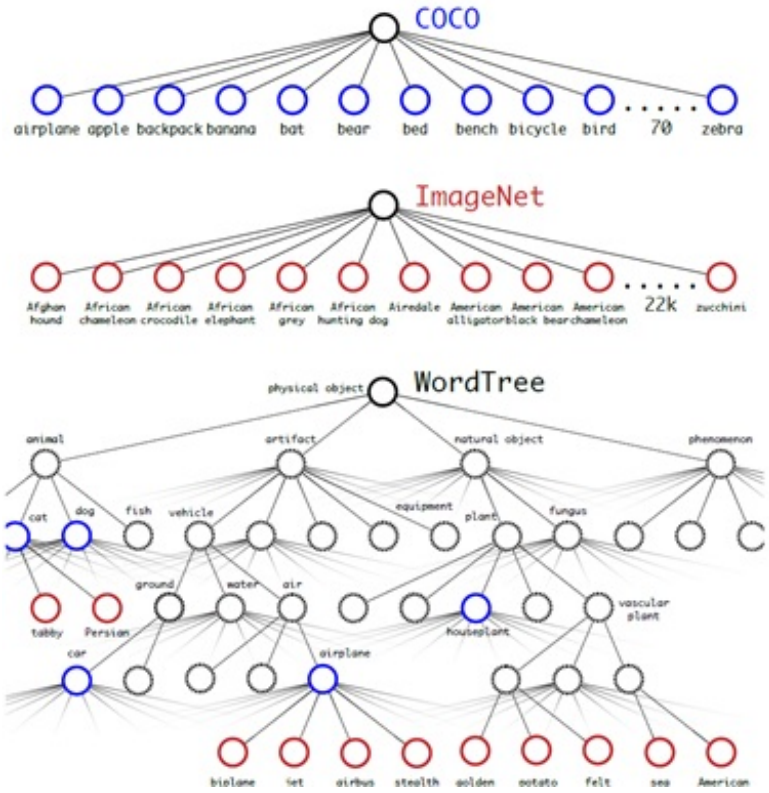

图15 COCO与ImageNet联合生成的wordtree

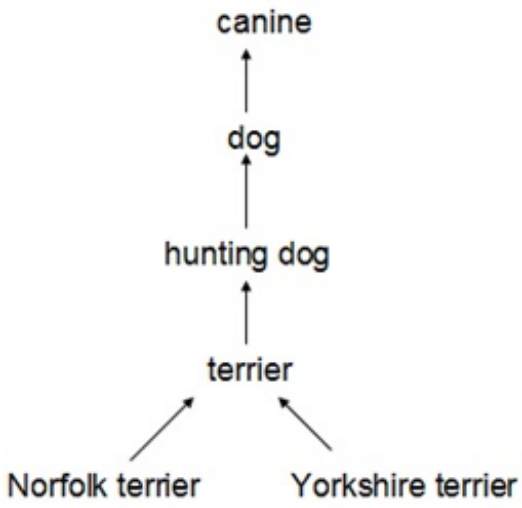

图16 wordtree例子

每个节点的子节点都属于同一子类，所以可以对它们分类型进行softmax处理。这样就解决了原始分类的冲突问题。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给出的置信度是根结点的置信度 ，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。

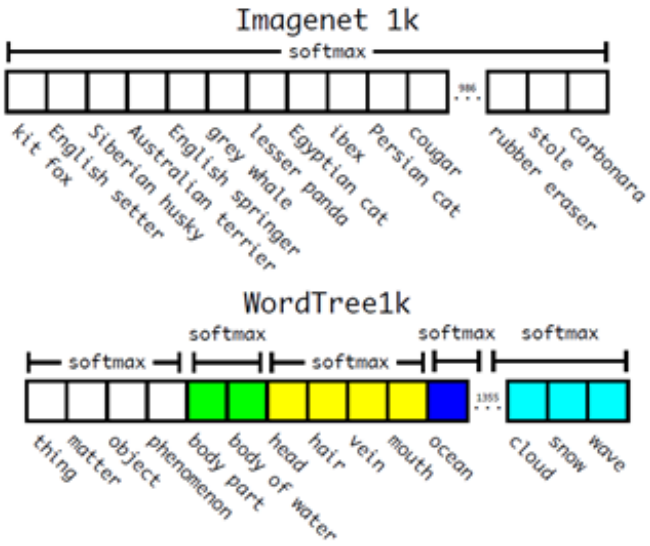

图17 wordtree的softmax处理

2.3 YOLOv3

YOLOv3给出的是一个科技报告，在保持实时性的基础上，对YOLOv2进行了几点改进，主要有三点：采用逻辑回归预测置信度和进行分类，从三个尺度上预测b-box的坐标以及特征提取器发生变化。

逻辑回归的使用：在YOLOv2中，每个cell是直接通过网络回归预测b-box坐标和置信度的，YOLOv3则将置信度和坐标分开预测，坐标预测还是通过网络进行预测，而置信度则是单独通过逻辑回归进行预测。在分类上，没有使用softmax多分类，作者也指出softmax最终对性能也没有提升，而且softmax假设是每个box只有一个类，这对迁移到更大有多种类别标签的数据集是没有好处的，所以作者使用多个逻辑回归来预测分类，使用二元交叉熵计算分类损失。

特征提取器：YOLOv3重新训练了一个新的特征提取器——DarkNet-53，使用了残差网络，相比最先进的特征提取器，性能相当，但浮点数运算更少，速度更快，下采样没有使用池化操作，而是通过卷积步长来实现。图18是DarkNet-53的结构图。

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

图18 DarkNet-53结构图

多尺度预测坐标：作者借由FPN的思想，引用中间层的输出与后层输出进行融合，进行三个尺度预测，每个尺度的每个cell预测3个坐标，以上面为例，下采样32倍，最后一起的输出是8x8x1024，通过卷积层和逻辑回归得到8x8x255（255=3x（5+80），5是坐标加置信度，80是coco类别），这就是第一个尺度的预测输出，第二个尺度是8x8x1024通过上采样与卷积通过缩放变成16x16x512，然后与上一个stage的16x16x512进行concat，然后通过相似的方式生成16x16x255，类似的操作得到，得到32x32x255。

3.其他方法

R-CNN：将深度学习应用到目标检测的开创性工作之一，处理过程如图19所示，具体有四步，第一步是使用选择性搜索对输入图像提取不同尺寸不同形状大小的候选区域，第二步是选取一个预训练好的深度学习分类模型，将输出层截取掉，将候选区域形变为网络输入需要的固定形状，得到每个候选区域的特征图。第三步是将特征图与类别标签联合，通过多个SVM分类器来进行分类。第四步是将特征图与位置标签联合，通过线性回归模型预测真实边界框。

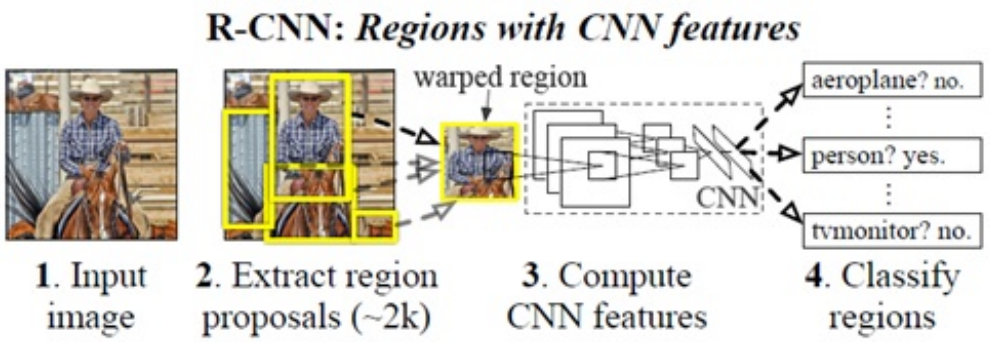

图19 R-CNN流程图

FastRCNN：RCNN是很慢的，每个候选区域都要通过前向传播，许多候选区域是相互重叠的，FastRCNN还是通过选择性搜索得到候选框，但FastRCNN是将输入图像直接通过预训练模型，将候选框映射到特征图中进行提取感兴趣区域，然后不同大小的区域通过RoI Pooling层得到相同大小的特征向量，最后通过两个全连接层得到类别和边界框的预测。具体如下图20所示。

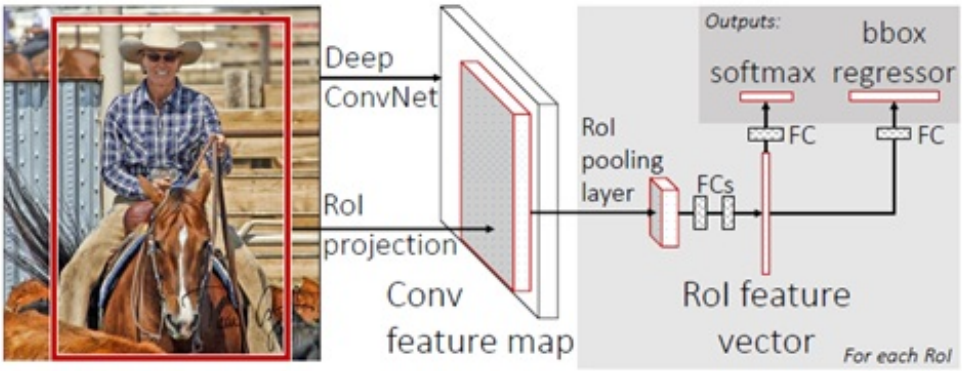

图19 FastRCNN流程图

FasterRCNN：FastRCNN需要通过选择性搜索得到许多候选框，才能得到较准确的精度，FasterRCNN针对这一问题，提出将选择性搜索替换成区域候选网络（RPN），通过网络自动学习提取好的候选区域，从而可以减少候选区域的数目，提高速度并保证了精度。具体做法是将特征提取的网络输出通过一个填充为1的3x3的卷积层变换为通道为512的特征图，这样特征图的每个单元都有512维的特征向量，以每个单元为中心，生成9个不同的锚盒（3个大小，3个不同高宽比）并标注它们，使用单元的特征向量预测锚框的二元类别（foreground-background）以及位置坐标，最后使用非极大值抑制去除相似重复的目标边界框。RPN的大致流程如图20所示。

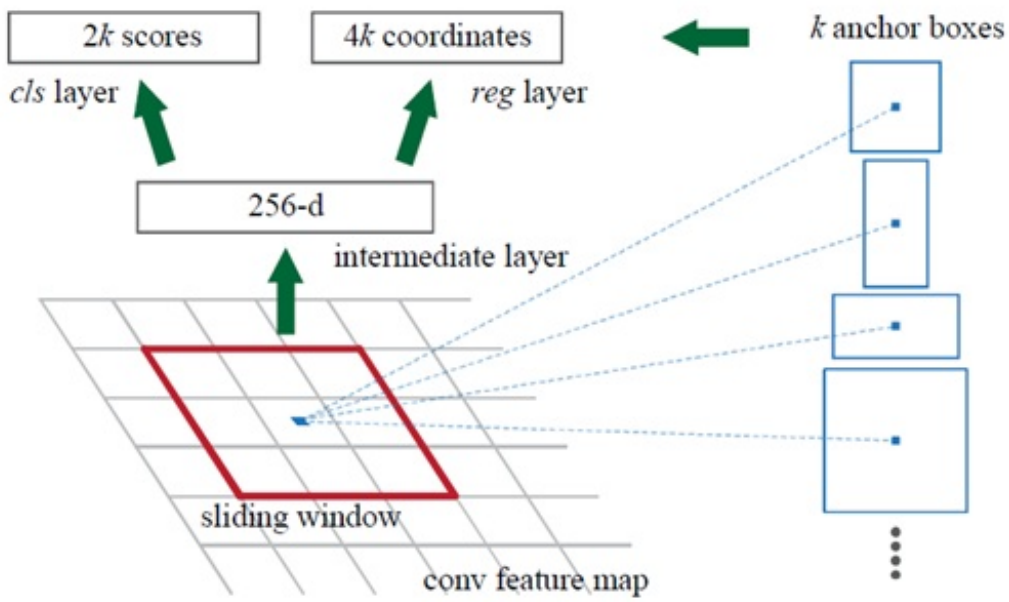

图20 RPN流程图

SSD：SSD全称是单发多框检测，它的具体流程如图21所示。首先还是通过截断全连接层的VGG网络提取特征，得到Conv6特征图，其中VGG中间层Conv4_3的特征图后面会用到，然后将Conv特征图不断地减半，得到了5个不同大小的特征图，这样，分别在Conv4_3的特征图以及这5个特征图生成锚盒，预测类别与边界框，方法与FasterRCNN类似，对于宽高大的特征图，感受野小，锚框多，适合检测小的物体，而对于宽高小的特征图，感受野大，锚框少，则适合检测大的物体，所以SSD还是一种多尺度的目标检测网络。

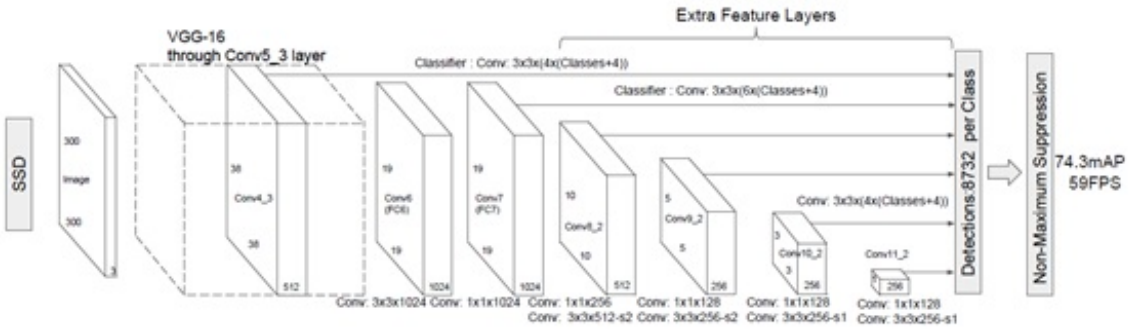

图21 SSD流程图

RetinaNet：上面介绍了one-stage的检测器（SSD，YOLO）以及two-stage的检测器（RCNN系列），但往往two-stage的检测器性能要强于one-stage，但速度要慢，RetinaNet这篇文章就指出one-stage性能差的一个重要原因是由b-box的foreground-background类别不平衡问题引起的。Two-stage方法会筛选b-box（RPN减少了

background的数目）并且训练过程会设置foreground-background的比例，类别不平衡的问题要轻许多，而one-stage的方法则是直接回归最后的边界框预测，对于一个二分类器来讲，某一个类别数目过大，会导致分类器偏向数目大的类别（也就是忽略了重要有物体的foreground，偏向图像的背景），这个问题会导致目标检测的精度很低。针对这一问题，这篇文章提出了Focal Loss，通过简单的loss函数改变，来解决类别不平衡的问题。公式如图22所示。由标准的交叉熵损失修改而成。α_t和γ参数减少了好分类的样本的损失，并让分类器更集中解决更困难样本。

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t).$$

图22 Focal loss

4.实验结果比较

目标检测算法一般是在VOC10和COCO11数据集中进行测试的，下表1给出上述算法在这两个数据集中的测试结果。限于篇幅，还有很多方法没有讲，特别是更先进的two-stage算法，同时期的two-stage 方法是要比one-stage方法在性能上是要强的，这里只介绍最基本的two-stage方法。

表1 VOC2007与COCO test-dev的测试结果

Method	VOC2007 mAP	COCO mAP	time(ms)
RCNN	53.6	None	13000
FastRCNN	70	None	2000
FasterRCNN+VGG	73.2	None	143
YOLOv1	66.4	None	48
YOLOv2	**78.6**	21.6	25
SSD	74.3	28	61
RetinaNet-50	None	**32.5**	73
YOLOv3	None	28.2	**22**

5.总结

对于目标检测这一任务而言，如果更关注性能指标，则可以参考two-stage系列的方法，而如果更关注实时性的要求，则可以关注one-stage的方法，这篇报告特别讲述了one-stage的YOLO算法，YOLO是以实时性为基础的快速目标检测算法，这种方法很快，也有许多工作，面向GPU和CPU实时性的目标检测算法都有人做出来，YOLO的含义也很有意思，YOLO——you only look once，其实还有一种说法，you only live once，别犹豫了，用起来吧。

参考文献

[1] Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2).

[2] Redmon J , Divvala S , Girshick R , et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2015.

[3] Redmon J , Farhadi A . [IEEE 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Honolulu, HI (2017.7.21-2017.7.26)] 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - YOLO9000: Better, Faster, Stronger[C]// IEEE Conference on Computer Vision & Pattern Recognition. IEEE, 2017:6517-6525.

[4] Redmon, J., and Farhadi, A.: 2016, arXiv e-prints, arXiv:1612.08242.

[5] Girshick R , Donahue J , Darrell T , et al. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 38(1):142-158.

[6] Girshick R . Fast R-CNN[J]. Computer Science, 2015.

[7] Ren S , He K , Girshick R , et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. 2015.

[8] Liu W , Anguelov D , Erhan D , et al. SSD: Single Shot MultiBox Detector[J]. 2015.

[9] Lin T Y , Goyal P , Girshick R , et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99):2999-3007.

[10] Everingham M , Eslami S M A , Van Gool L , et al. ThePascalVisual Object Classes Challenge: A Retrospective[J]. International Journal of Computer Vision, 2015, 111(1):98-136.

[11] Lin T Y , Maire M , Belongie S , et al. Microsoft COCO: Common Objects in Context[J]. 2014.
[12] http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf
[13] https://blog.csdn.net/iamoldpan/article/details/78799857
[14] Ioffe S , Szegedy C . Batch normalization: accelerating deep network training by reducing internal covariate shift[C]// International Conference on International Conference on Machine Learning. JMLR.org, 2015.
[15] https://towardsdatascience.com/training-object-detection-yolov2-from-scratch-using-cyclic-learning-rates-b3364f7e4755
[16] http://ethereon.github.io/netscope/#/gist/d08a41711e48cf111e330827b1279c31