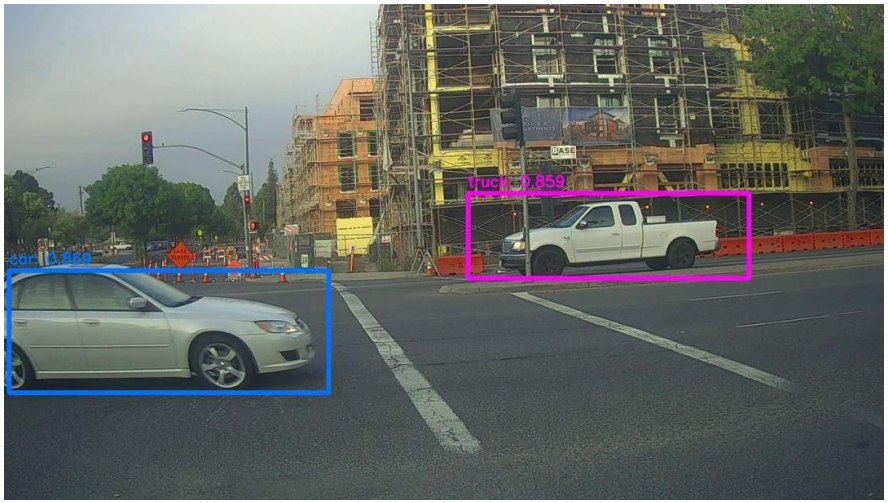
第十二讲

视频算法工程师实战



主要内容

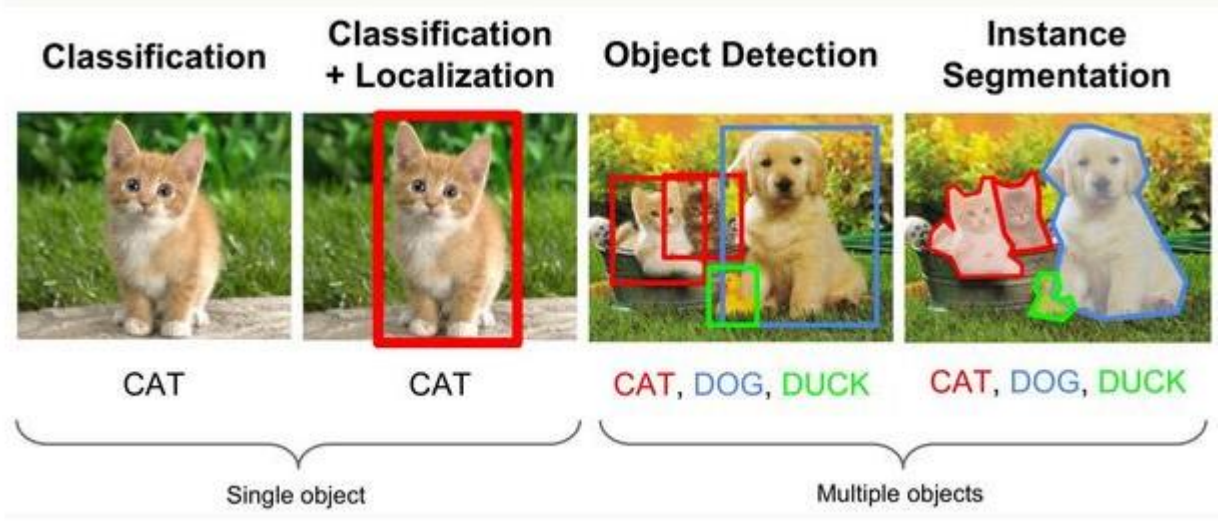
YW元微
deep168.com

- RCNN：Region-based Convolutional Neural Networks
- Fast RCNN
- Faster RCNN
- YOLO: You Look Only Once
- SSD: Single Short MultiBox Detector

计算机视觉的三大任务

YW元微
deep168.com

- 图像分类：将图像划分为单个类别，通常对应于图像中最突出的物体
- 目标检测：识别一张图片中的多个物体，并可以给出边界框
  - 应用场景：无人驾驶、安防系统
- 图像分割



基于深度学习的目标检测算法

YW元微
deep168.com

目前目标检测领域的深度学习方法主要分为两类：

- Two Stage：先由算法生成一系列作为样本的候选框，再通过卷积神经网络进行样本分类
- One Stage：不用产生候选框，直接将目标边框定位的问题转化为回归问题处理

两种方法由于方法不同，性能上有差异：

- Two Stage：检测准确率和定位精度占优
- One Stage：算法速度占优

算法代表：

- Two Stage：RCNN系列，倾向于准
- One Stage：YOLO，倾向于快

第1部分

RCNN、Fast RCNN、Faster RCNN

- Region-based Convolutional Neural Networks
- 基于候选区域（Region Proposal）的深度学习目标检测算法

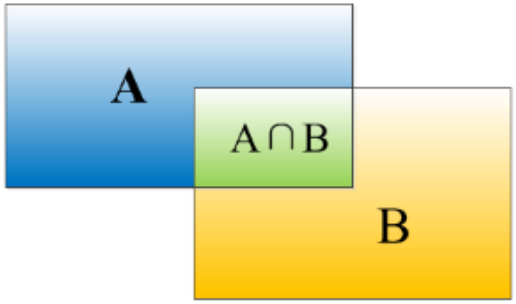
候选区域（Region Proposal）

YW元微
deep168.com

- 候选区域（Region Proposal）：预先找出图中目标可能出现的位置，通过利用图像的纹理、边缘、颜色等信息，保证在选取较少窗口（几千个甚至几百个）的情况下保持较高的召回率（IOU, Intersection-Over-Union）
- IOU：定位精度评价公式

bounding box里面的物体是车辆，对于bounding box的定位精度，我们算大不可能100%跟人工标注重合，因此存在一个精度评价公式IoU

IOU定义了两个bounding box的重叠度，如下图所示：



矩形框A、B的一个重合度IOU计算公式为：

$$IOU=(A\cap B)/(A\cup B)$$

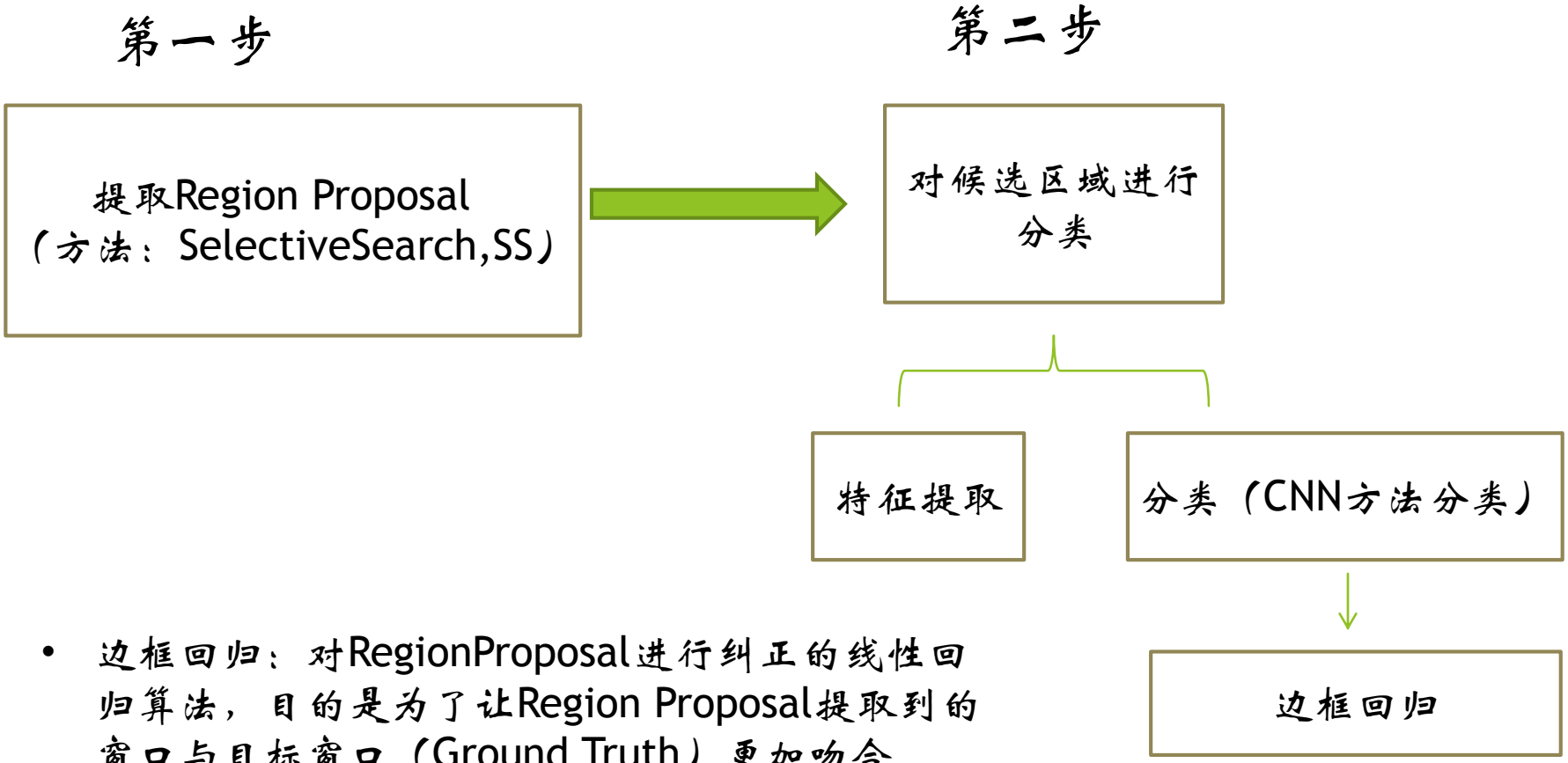
就是矩形框A、B的重叠面积占A、B并集的面积比例：

$$IOU=S_I/(S_A+S_B-S_I)$$

基于Region Proposal的目标检测算法步骤

YW元微 deep168.com

Two stage 两步走！



- 边框回归：对RegionProposal进行纠正的线性回归算法，目的是为了让Region Proposal提取到的窗口与目标窗口（Ground Truth）更加吻合

选择性搜索（selective search）

YW元微
deep168.com

step0：生成区域集R

step1：计算区域集R里每个相邻区域的相似度S={s1,s2,...}

step2：找出相似度最高的两个区域，将其合并为新集，添加进R

step3：从S中移除所有与step2中有关的子集

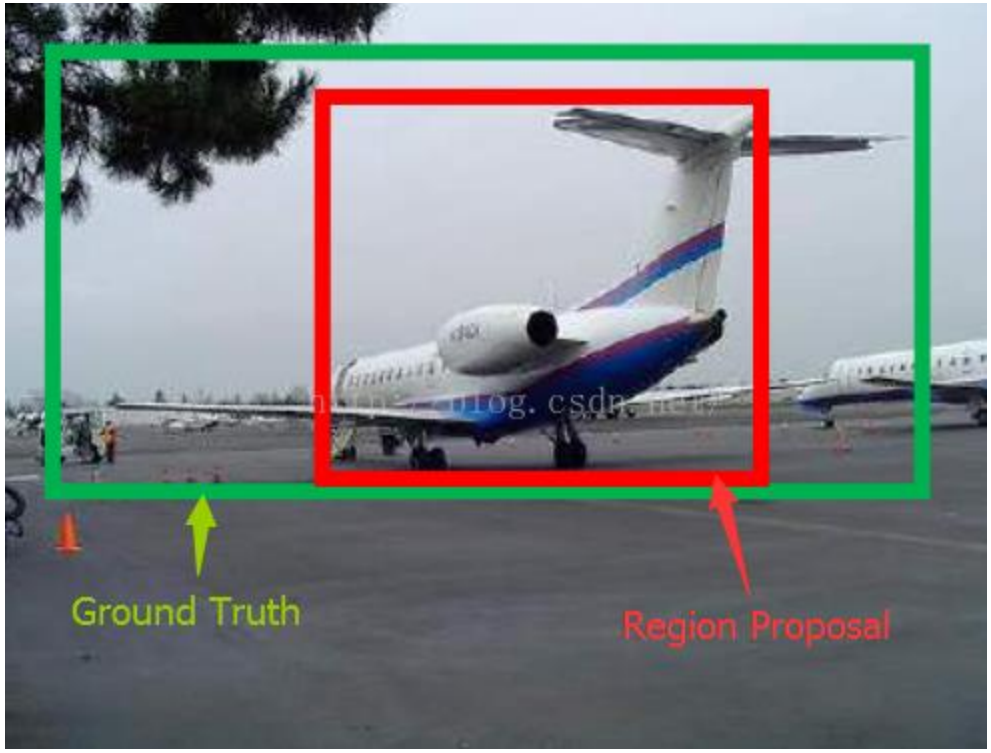
step4：计算新集与所有子集的相似度

step5：跳至step2，直至S为空

边框回归

YW元微
deep168.com

如图所示，绿色的框为飞机的Ground Truth，红色的框是提取的Region Proposal。那么即便红色的框被分类器识别为飞机，但是由于红色的框定位不准(IoU<0.5)，那么这张图相当于没有正确的检测出飞机。如果我们能对红色的框进行微调，使得经过微调后的窗口跟Ground Truth更接近，这样岂不是定位会更准确。确实，Bounding-box regression 就是用来微调这个窗口的。



边框回归

YW元微
deep168.com

回归/微调的对象是什么？

对于窗口一般使用四维向量(x, y, w, h)来表示，分别表示窗口的中心点坐标和宽高。对于图 11，红色的框 P 代表原始的 Proposal，绿色的框 G 代表目标的 Ground Truth，我们的目标是寻找一种关系使得输入原始的窗口 P 经过映射得到一个跟真实窗口 G 更接近的回归窗口$\hat{G}$。

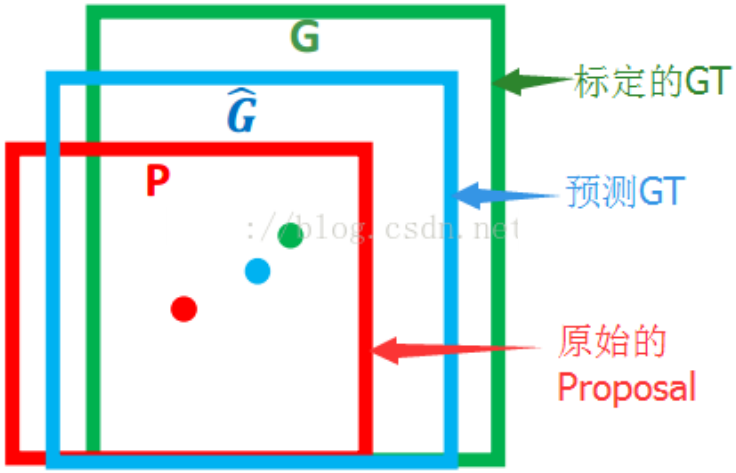


图 11　回归窗口

即：给定(P_x, P_y, P_w, P_h)，寻找一种映射f，使得：

$$f(P_x, P_y, P_w, P_h) = (\hat{G_x}, \hat{G_y}, \hat{G_w}, \hat{G_h}) \approx (G_x, G_y, G_w, G_h)$$

边框回归

YW元微
deep168.com

那么经过何种变换才能从图11中的窗口P变为窗口呢？比较简单的思路就是：

（1） 先做平移$(\Delta x, \Delta y)$，$\Delta x = P_w d_x(P)$，$\Delta y = P_h d_y(P)$。

这就是 R-CNN 论文中的：

http://blog.csdn.net/

$$\widehat{G_x} = P_w d_x(P) + P_x$$

$$\widehat{G_y} = P_h d_y(P) + P_y$$

（2） 后做尺度缩放(S_w, S_h)，$S_w = P_w d_w(P)$，$S_h = P_h d_h(\mathrm{P})$

$$\widehat{G_w} = P_w e^{d_w(P)}$$

$$\widehat{G_h} = P_h e^{d_h(\mathrm{P})}$$

所以，由上面 4 个公式可以看出，我们需要学习$d_x(P)$，$d_y(P)$，$d_w(P)$，$d_h(\mathrm{P})$这四个变换。下一步就是设计算法得到这四个映射。当输入的 Proposal 与 Ground Truth 相差较小时(RCNN 设置的是 IoU>0.6)，可以认为这种变换是一种线性变换，那么我们就可以用线性回归来建模对窗口进行微调。

边框回归

YW元微
deep168.com

线性回归就是给定输入的特征向量X，学习一组参数W，使得经过线性回归后的值跟真实值Y(Ground Truth)非常接近。即 $Y \approx WX$.

那么Bounding-box中我们的输入以及输出分别是什么呢？

输入： Region Proposal $\rightarrow P = (P_x, P_y, P_w, P_h)$

输出：需要进行的平移变换和尺度缩放

$d_x(P),\ d_y(P),\ d_w(P),\ d_h(P)$.

这四个值应该是经过 Ground Truth 和Proposal计算得到的真正需要的平移量和尺度缩放

$$t_x = \frac{(G_x - P_x)}{P_w}$$
$$t_y = \frac{(G_y - P_y)}{P_h}$$
$$t_x = \log\frac{G_w}{P_w}$$
$$t_x = \log\frac{G_h}{P_h}$$

边框回归

YW元微
deep168.com

那么目标函数可以表示为

得到的预测值 → $d_*(P) = w_*^T \Phi_5(P)$

要学习的参数 x, y, w, h 也就是每一个变换对应一个目标函数

是输入Proposal的特征向量

我们要让预测值跟真实值差距最小，得到损失函数为：

$$\text{Loss} = \sum_i^N (t_*^i - \hat{w}_*^T \Phi_5(P^i))^2$$

边框回归

YW元微
deep168.com

函数优化目标为：

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2$$

利用梯度下降法就可以得到

测试阶段

对于测试图像，我们首先经过 CNN 提取特征 $\Phi_5(P)$

预测的变化就是 $d_*(P) = w_*^T \Phi_5(P)$

最后根据以下4个公式对窗口进行回归：

$$\widehat{G_x} = P_w d_x(P) + P_x$$

$$\widehat{G_y} = P_h d_y(P) + P_y$$

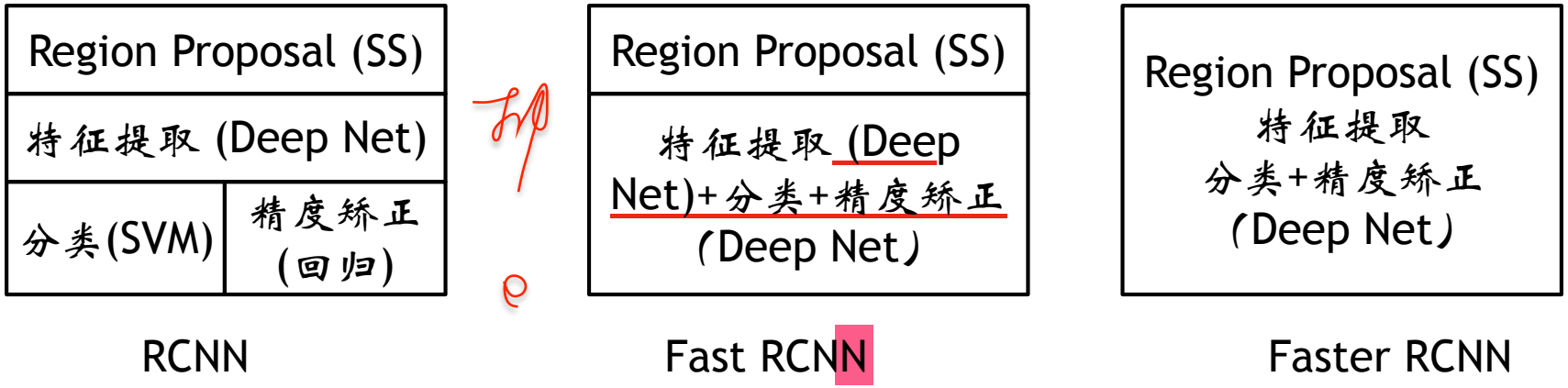
$$\widehat{G_w} = P_w e^{d_w(P)}$$

$$\widehat{G_h} = P_h e^{d_h(P)}$$

RCNN、Fast RCNN、Faster RCNN

YW元微 deep168.com

- 三者比较



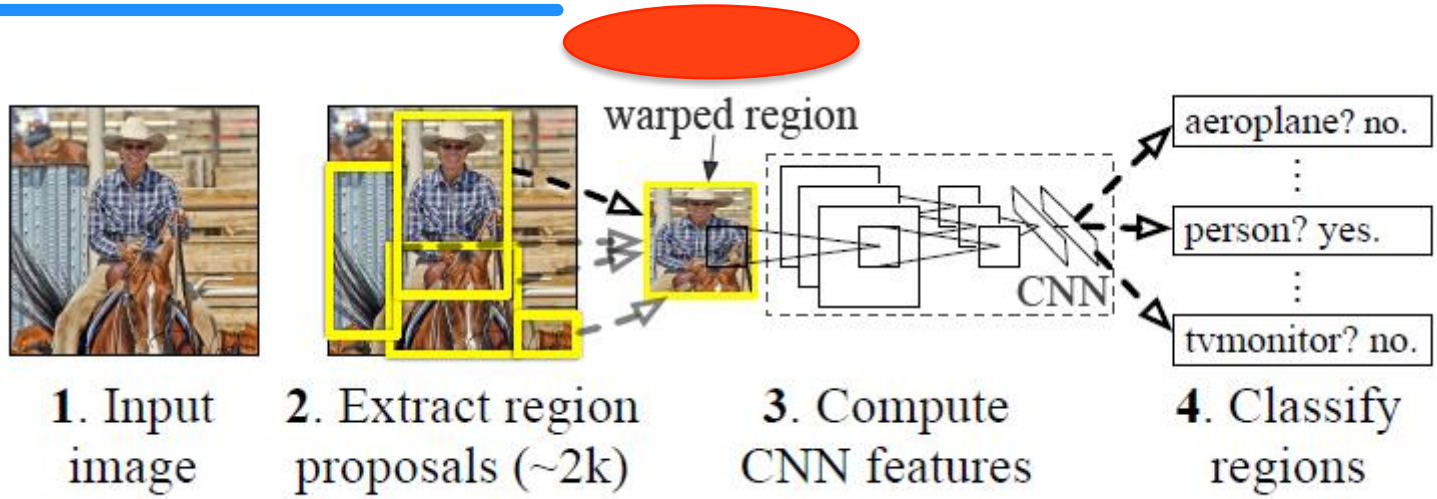
RCNN、Fast RCNN、Faster RCNN

YW元微 deep168.com

	使用方法	缺点	改进
R-CNN (Region-based Convolutional Neural Networks), 2014年提出	1、SS提取RP； 2、CNN提取特征； 3、SVM分类； 4、BB盒回归。	1、 训练步骤繁琐（微调网络+训练SVM+训练bbox）； 2、 训练、测试均速度慢； 3、 训练占空间	1、 从34.3%直接提升到了66%； 2、 引入RP+CNN
Fast R-CNN (Fast Region-based Convolutional Neural Networks), 2015年提出	1、SS提取RP； 2、CNN提取特征； 3、softmax分类； 4、多任务损失函数边框回归。	1、 依旧用SS提取RP(耗时2-3s，特征提取耗时0.32s)； 2、 无法满足实时应用，没有真正实现端到端训练测试； 3、 利用了GPU，但是区域建议方法是在CPU上实现的。	1、 由66.9%提升到70%； 2、 每张图像耗时约为3s。
Faster R-CNN (Fast Region-based Convolutional Neural Networks) 2015年提出	1、RPN提取RP； 2、CNN提取特征； 3、softmax分类； 4、多任务损失函数边框回归。	1、 还是无法达到实时检测目标； 2、 获取region proposal，再对每个proposal分类计算量还是比较大。	1、 提高了检测精度和速度； 2、 真正实现端到端的目标检测框架； 3、 生成建议框仅需约10ms。

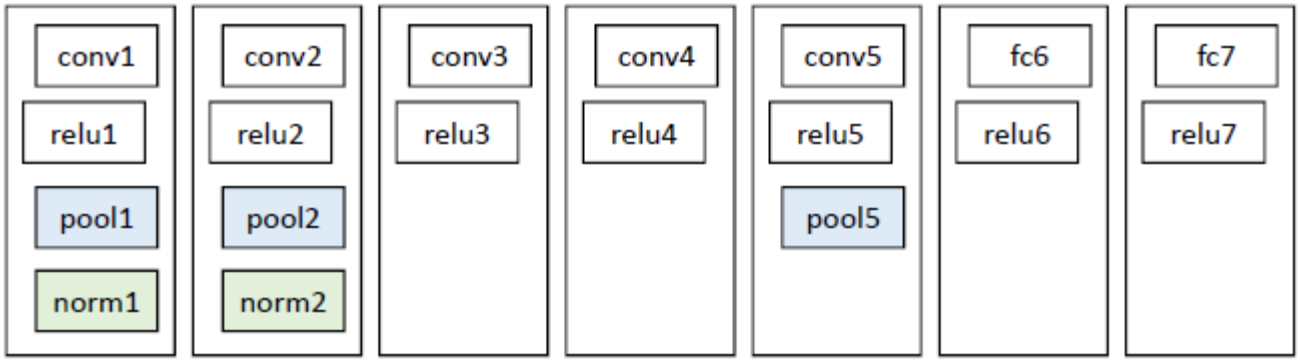
RCNN算法流程

YW元微
deep168.com



RCNN算法分为4个步骤：

1. 对一张图像生成1K~2K个**候选区域**
2. 对每个候选区域，使用深度网络（结构如下图）**提取特征**
3. 特征送入每一类的**SVM分类器**，判别是否属于该类
4. 使用回归器(Bounding box, bbox)**精度修正**候选框位置



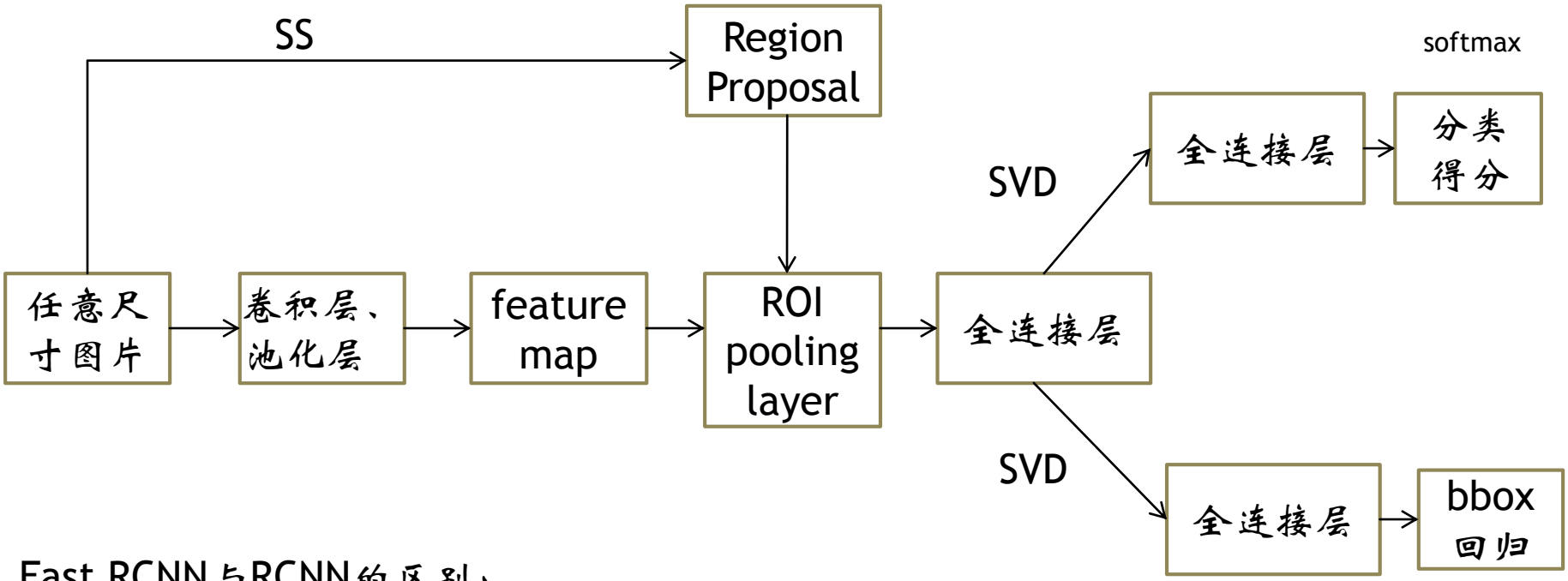
RCNN存在的问题——改进Fast RCNN

YW元微 deep168.com

RCNN问题		Fast RCNN改进方法
测试时速度慢	RCNN一张图像内候选框之间大量重叠，提取特征操作冗余。	将整张图像归一化后直接送入深度网络。在邻接时，才加入候选框信息，在末尾的少数几层处理每个候选框。
训练时速度慢	同上	在训练时，本文先将一张图像送入网络，紧接着送入从这幅图像上提取出的候选区域。这些候选区域的前几层特征不需要再重复计算。
训练所需空间大	RCNN中独立的分类器和回归器需要大量特征作为训练样本。	本文把类别判断和位置精调统一用深度网络实现，不再需要额外存储。

Fast RCNN目标检测流程

YW元微 deep168.com



Fast RCNN与RCNN的区别：

- Fast RCNN的Region Proposal是在feature map之后做的，这样就可以不用对所有区域进行单独CNN forward计算；
- 最后一个卷积层后加了一个ROI pooling layer；
- 损失函数使用了multi-task loss（多任务损失）函数，将边框回归直接加到CNN网络中训练；
- fast RCNN直接用softmax替代RCNN的SVM进行分类

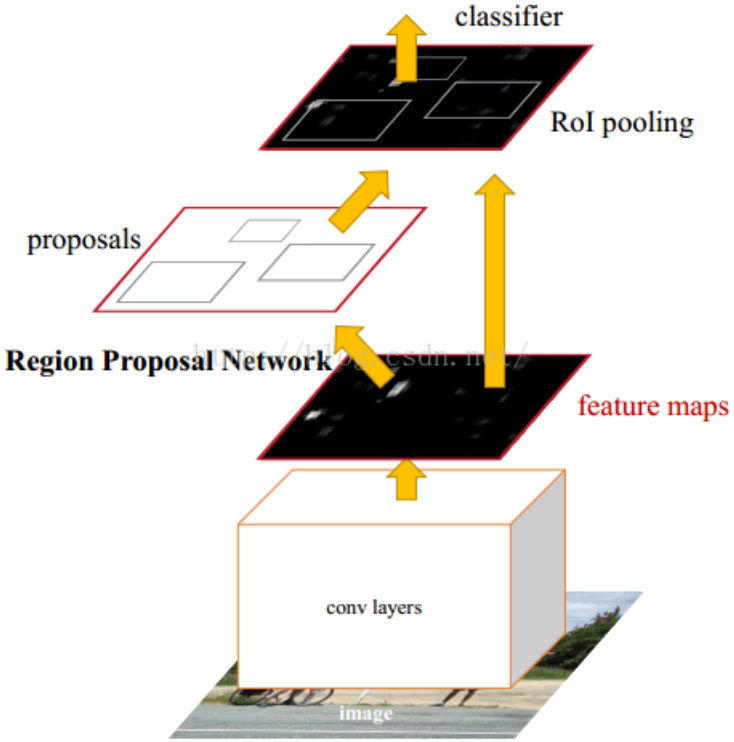
Faster RCNN

YW元微
deep168.com

- Faster RCNN：区域生成网络RPNs+Fast RCNN
- 用区域生成网络代替Fast RCNN中的Selective Search方法
- Faster RCNN解决的三个问题：
  1. 如何设计区域生成网络；
  2. 如何训练区域生成网络；
  3. 如何让区域生成网络和Fast RCNN网络共享特征提取网络
- 在整个Faster RCNN算法中，有三种尺度：
  1. 原图尺度：原始输入的大小。不受任何限制，不影响性能；
  2. 归一化尺度：输入特征提取网络的大小，在测试时设置，源码中opts.test_scale=600。anchor在这个尺度上设定。这个参数和anchor的相对大小决定了想要检测的目标范围。
  3. 网络输入尺度：输入特征检测网络的大小，在训练时设置，源码中为224*224。

Faster RCNN框架

YW元微 deep168.com



Faster RCNN算法由2大模块组成：

1. RPN候选框提取模块；
2. Fast RCNN检测模块。

其中，RPN是全卷积神经网络，用于提取候选框；Fast RCNN基于RPN提取的proposal检测并识别proposal中的目标。

RPN介绍

YW元微
deep168.com

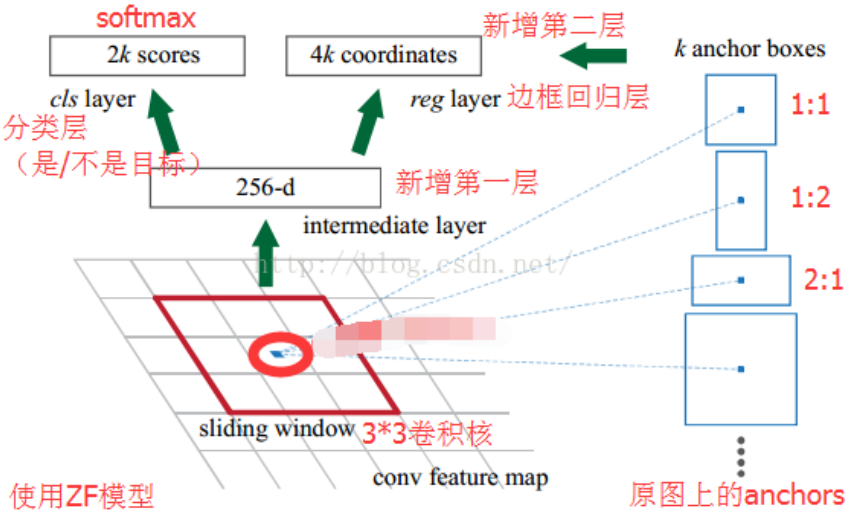
- RPN(Region Proposal Network)：将Region Proposal（区域检测）也由CNN来做，它能和整个检测网络共享全图的卷积特征，使得区域检测几乎不花时间。
- RCNN解决的是——为什么不用CNN做分类？
- Fast RCNN解决的是——为什么不一起输出bounding box和label呢？
- Faster RCNN解决的是——为什么还要用selectibe search(SS)进行Region Proposal（区域检测）
- RPN的核心思想：使用CNN直接产生Region Proposal，使用的方法本质上就是滑动窗口（只需在最后的卷积层上滑动一遍），因为anchor机制和边框回归可以得到多尺度多长宽比的Region Proposal。
- RPN网络也是全卷积网络（FCN，fully-convolutional network），可以针对生成检测建议框的任务端到端地训练，能够同时预测出object的边界和分数。只是在CNN上额外增加了2个卷积层（全卷积层cls和reg）。

RPN具体流程

YW元微 deep168.com

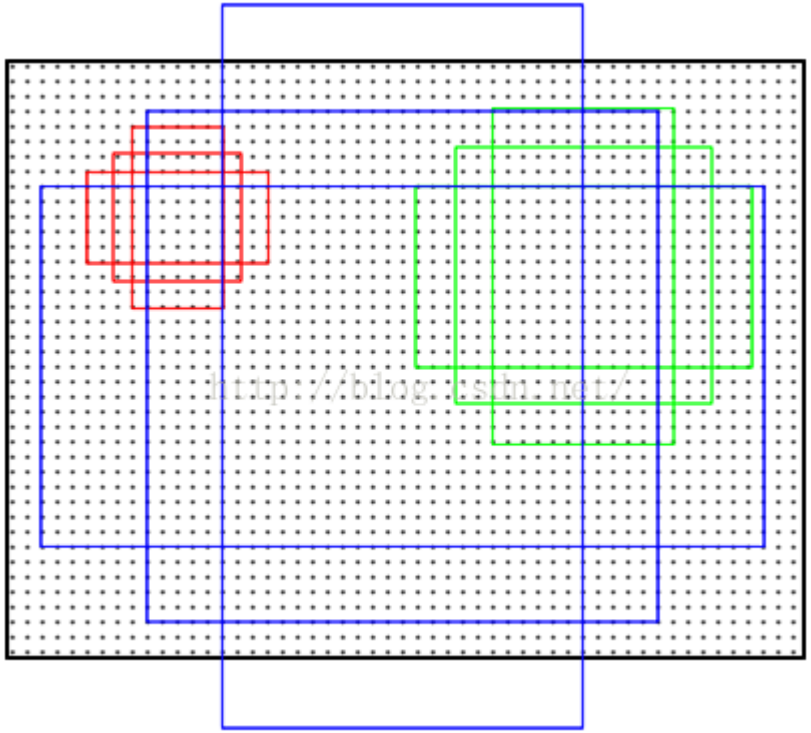
使用一个小网络在最后卷积得到的特征图上进行滑动扫描，这个滑动网络每次与特征图上n*n（论文中n=3）的窗口全连接（图像的有效感受野很大，ZF是171像素，VGG是228像素），然后映射到一个低维向量（256d for ZF / 512d for VGG），最后将这个低维向量送入到两个全连接层，即bbox回归层（reg）和box分类层（cls）。sliding window的处理方式保证reg-layer和cls-layer关联了conv5-3的全部特征空间。

- reg层：预测proposal的anchor对应的proposal的（x,y,w,h）
- cls层：判断该proposal是前景（object）还是背景（non-object）。



RPN具体流程

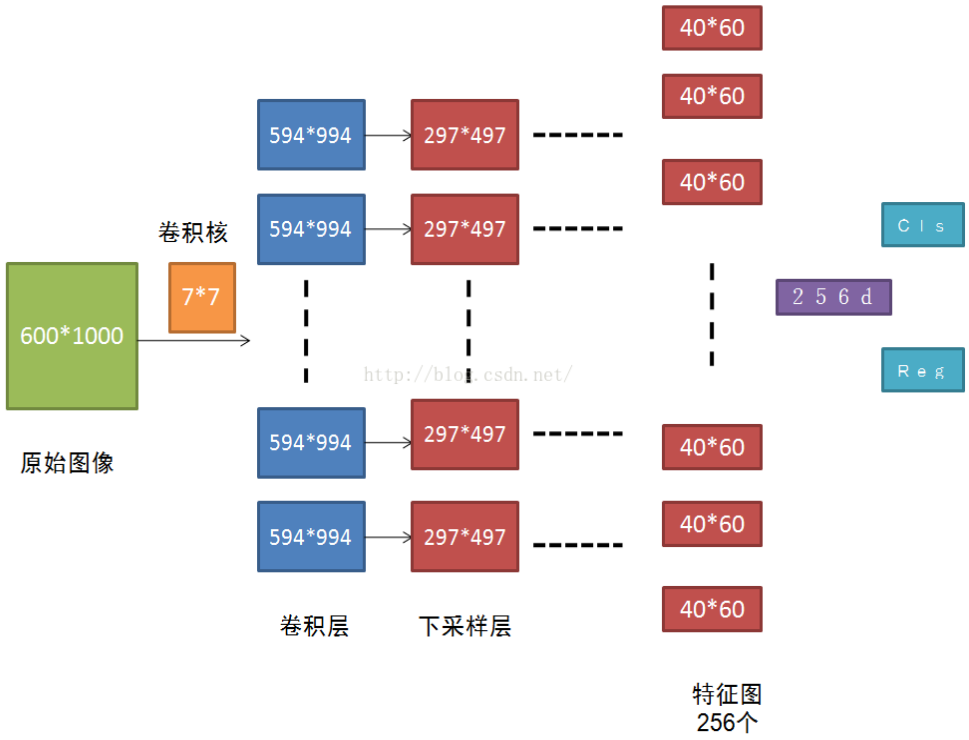
YW元微
deep168.com



三种尺度（面积）{128^2，256^2，512^2}，三种比例{1:1, 1:2, 2:1}，所以每一个位置都有 3*3=9 个 anchor。

RPN具体流程

YW元微
deep168.com



- 原图600*1000经CNN卷积后，在CNN最后一层（conv5）得出的是40*60大小的特征图，对应文中说的典型值为2400。若特征图大小为W*H，则需要W*H*K个anchor，本文中需要40*60*9≈2k个。

RPN的LOSS函数

YW元微
deep168.com

Loss functions:

在计算Loss值之前，作者设置了anchors的标定方法.正样本标定规则：

1) 如果Anchor对应的reference box与ground truth的IoU值最大，标记为正样本；

2) 如果Anchor对应的reference box与ground truth的IoU>0.7，标记为正样本.

事实上，采用第2个规则基本上可以找到足够的正样本，但是对于一些极端情况，例如所有的Anchor对应的reference box与groud truth的IoU不大于0.7,可以采用第一种规则生成.

3) 负样本标定规则：如果Anchor对应的reference box与ground truth的IoU<0.3，标记为负样本.

4) 剩下的既不是正样本也不是负样本，不用于最终训练.

5) 训练RPN的Loss是有classification loss（即softmax loss）和regression loss 按一定比重组成的.

RPN的LOSS函数

YW元微 deep168.com

Loss functions:

计算softmax loss需要的是anchors对应的groundtruth标定结果和预测结果，计算regression loss需要三组信息：

i. 预测框，即RPN网络预测出的proposal的中心位置坐标x,y和宽高w,h；

ii. 锚点reference box:

之前的9个锚点对应9个不同scale和aspect_ratio的reference boxes，每一个reference boxes都有一个中心点位置坐标x_a,y_a和宽高w_a,h_a；

iii. ground truth:标定的框也对应一个中心点位置坐标x*,y*和宽高w*,h*.因此计算总Loss方式如下：

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$

reference boxes

$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

ground truth

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

classification loss

$$+\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

regression loss

第2部分

YOLO v1、SSD、YOLO v2、YOLO v3

- YOLO：You Only Look Once: Unified, Real-Time Object Detection
- SSD：Single Short MultiBox Detector

目标检测算法分类

YW元微
deep168.com

- Two stage：提取区域候选（Region Proposal)+分类
  - 代表算法：RCNN, Fast RCNN, Faster RCNN
- One stage：将目标检测的问题转化成一个回归问题，给定输入图像，直接在图像的多个位置上回归出目标的bounding box以及其分类的类别。
  - 代表算法：YOLO、SSD

YOLO背景介绍

YW元微
deep168.com

- YOLO是由oseph Redmon和Ali Farhadi等人于2015年提出的基于单个神经网络的目标检测系统；
- 2017年CVPR上， Joseph Redmon和Ali Farhadi又发表的YOLO v2，进一步提高了检测的精度和速度；
- 2018年，华盛顿大学的 Joseph Redmon 和 Ali Farhadi 提出 YOLO 的最新版本 YOLOv3，这个新模型在取得相当准确率的情况下实现了检测速度的很大提升，一般它比 R-CNN 快 1000 倍、比 Fast R-CNN 快 100 倍，快SSD三倍。

YOLO主页 https://pjreddie.com/darknet/yolo/

YOLO算法简介

YW元微
deep168.com

- YOLO是一个可以一次性预测多个Box位置和类别的卷积神经网络，能够实现端到端的目标检测和识别，其最大的优势就是速度快。事实上，目标检测的本质就是回归，因此一个实现回归功能的CNN并不需要复杂的设计过程。YOLO没有选择滑动窗口（silding window）或提取proposal的方式训练网络，而是直接选用整图训练模型。这样做的好处在于可以更好的区分目标和背景区域，相比之下，采用proposal训练方式的Fast-R-CNN常常把背景区域误检为特定目标。

YOLO的核心思想

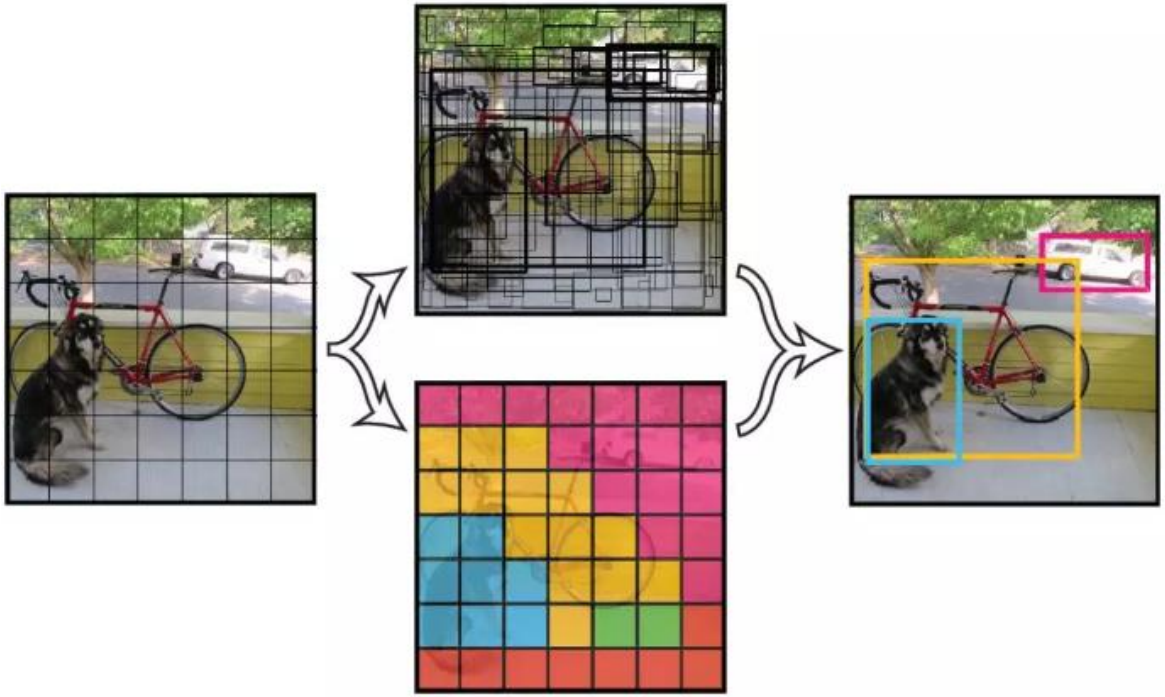
YW元微
deep168.com

- YOLO 的核心思想就是利用整张图作为网络的输入，直接在输出层回归 bounding box（边界框） 的位置及其所属的类别。
- faster-RCNN 中也直接用整张图作为输入，但是 faster-RCNN 整体还是采用了RCNN 那种提取区域候选（Region Proposal)+分类的思想，只不过是将提取 proposal 的步骤放在 CNN 中实现了，而 YOLO 则采用直接回归的思路。

YOLO的实现方法

YW元微
deep168.com

- 将一幅图像分成SxS个网格（grid cell），如果某个object的中心落在这个网格中，则这个网格就负责预测这个object。



YOLO的实现方法

YW元微
deep168.com

- 每个网格要预测 B 个 bounding box，每个 bounding box 除了要回归自身的位置之外，还要附带预测一个 confidence 值。
- 这个 confidence 代表了所预测的 box 中含有 object 的置信度和这个 box 预测的有多准这两重信息，其值是这样计算的：

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

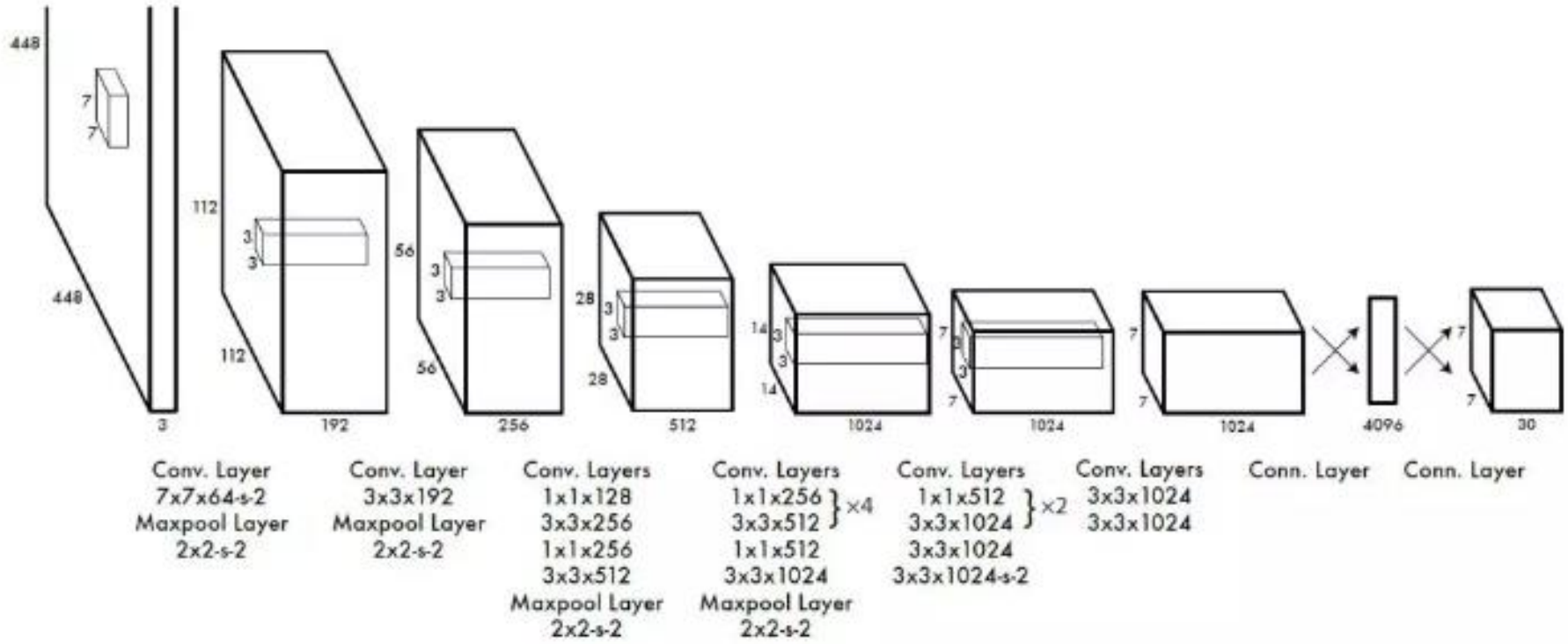
- 其中如果有 object 落在一个 grid cell 里，第一项取 1，否则取 0。第二项是预测的 bounding box 和实际的 groundtruth 之间的 IoU 值。
- 每个 bounding box 要预测 (x, y, w, h) 和 confidence 共5个值，每个网格还要预测一个类别信息，记为 C 类。则 SxS个网格，每个网格要预测 B 个 bounding box 还要预测 C 个 categories。输出就是 S x S x (5*B+C) 的一个 tensor。
- 注意：class 信息是针对每个网格的，confidence 信息是针对每个 bounding box 的。

YOLO的实现方法

YW元微
deep168.com

- 举例说明：在 PASCAL VOC 中，图像输入为 448x448，取 S=7，B=2，一共有20 个类别（C=20），则输出就是 7x7x30 的一个 tensor。
- 整个网络结构如下图所示：

S x S x (5*B+C)



YOLO的实现方法

YW元微 deep168.com

- 在 test 的时候，每个网格预测的 class 信息和 bounding box 预测的 confidence信息相乘，就得到每个 bounding box 的 class-specific confidence score:

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

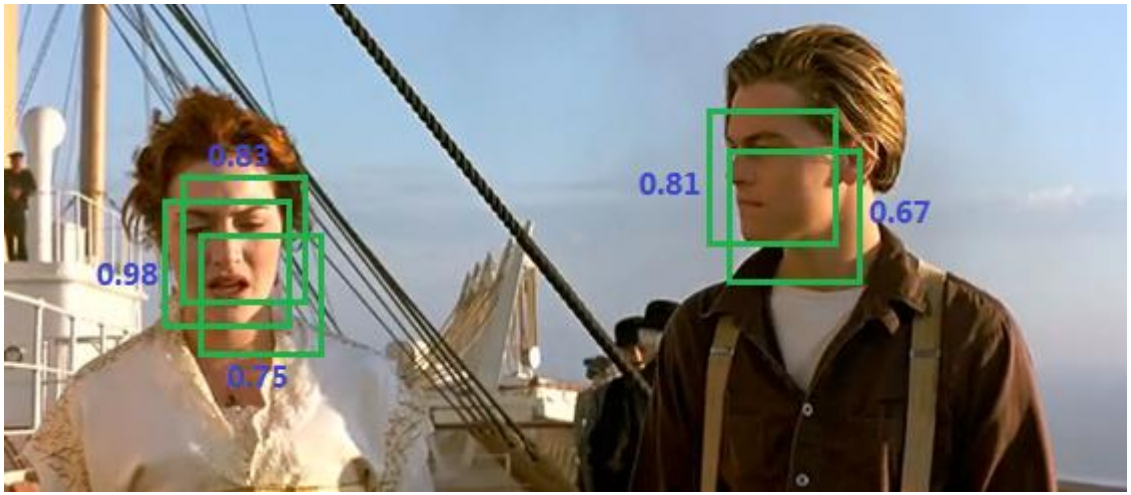
- 等式左边第一项就是每个网格预测的类别信息，第二、三项就是每个 bounding box 预测的 confidence。这个乘积即 encode 了预测的 box 属于某一类的概率，也有该 box 准确度的信息。
- 得到每个 box 的 class-specific confidence score 以后，设置阈值，滤掉得分低的 boxes，对保留的 boxes 进行 NMS 处理，就得到最终的检测结果。
- 注意：（1）由于输出层为全连接层，因此在检测时，YOLO 训练模型只支持与训练图像相同的输入分辨率；

（2）虽然每个格子可以预测 B 个 bounding box，但是最终只选择只选择 IOU 最高的 bounding box 作为物体检测输出，即每个格子最多只预测出一个物体。当物体占画面比例较小，如图像中包含畜群或鸟群时，每个格子包含多个物体，但却只能检测出其中一个。这是 YOLO 方法的一个缺陷。

NMS——非极大值抑制

YW元微
deep168.com

以下图为例，由于滑动窗口，同一个人可能有好几个框(每一个框都带有一个分类器得分)



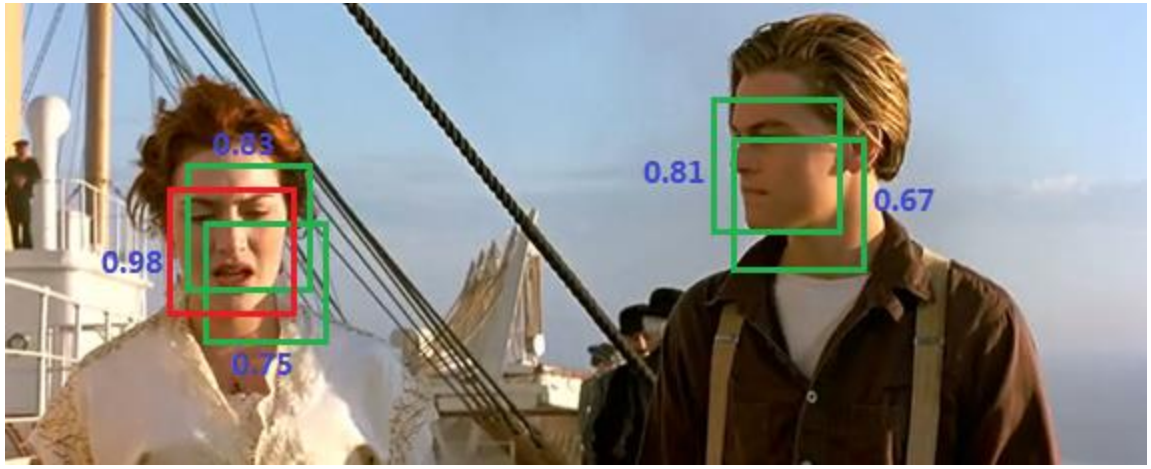
而我们的目标是一个人只保留一个最优的框；

于是我们就要用到非极大值抑制，来抑制那些冗余的框： 抑制的过程是一个迭代-遍历-消除的过程。

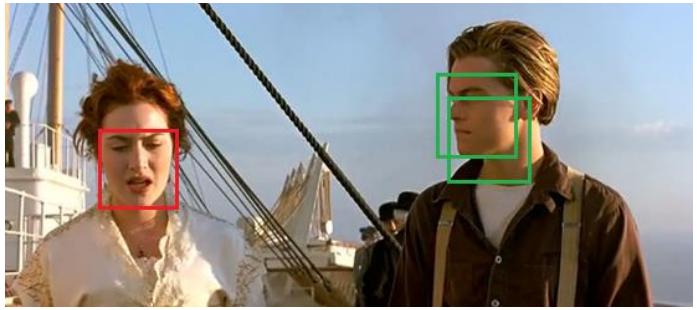
NMS——非极大值抑制

YW元微
deep168.com

（1）将所有框的得分排序，选中最高分及其对应的框：



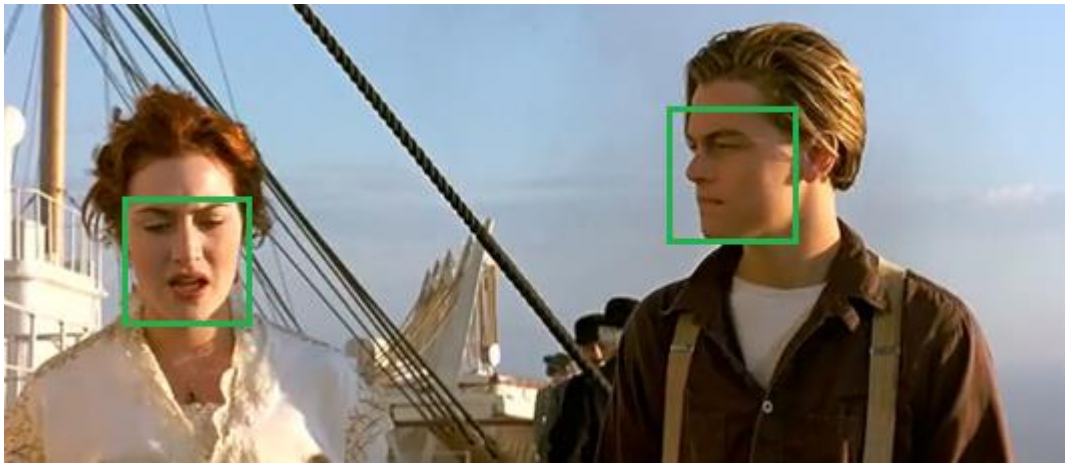
（2）遍历其余的框，如果和当前最高分框的重叠面积(IOU)大于一定阈值，我们就将框删除。



NMS——非极大值抑制

YW元微
deep168.com

（3）从未处理的框中继续选一个得分最高的，重复上述过程。



YOLO的缺点

YW元微
deep168.com

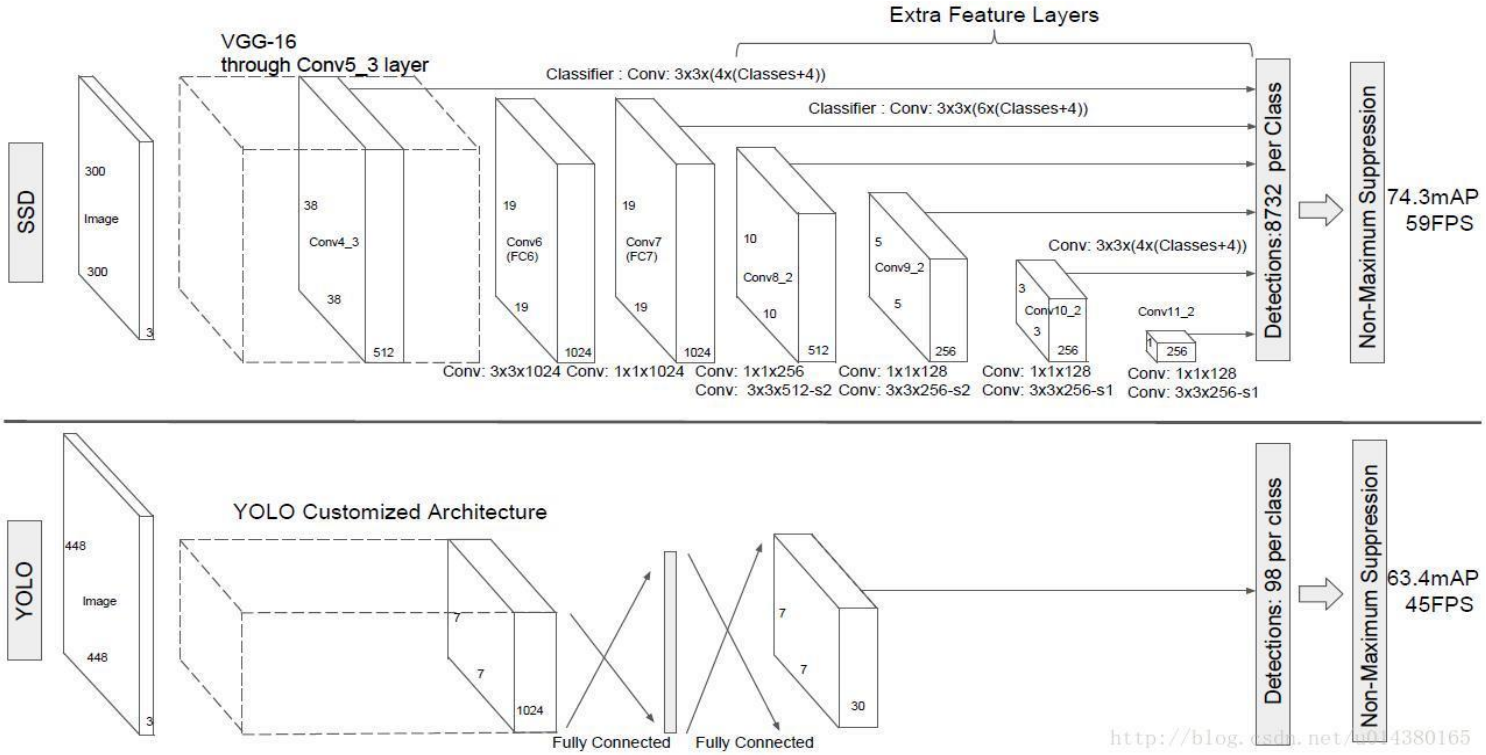
- YOLO 对相互靠的很近的物体，还有很小的群体检测效果不好，这是因为一个网格中只预测了两个框，并且只属于一类。
- 同一类物体出现的新的不常见的长宽比和其他情况时，泛化能力偏弱。
- 由于损失函数的问题，定位误差是影响检测效果的主要原因。尤其是大小物体的处理上，还有待加强。

论文： **You Only Look Once: Unified, Real-Time Object Detection**
https://arxiv.org/abs/1506.02640
论文中文翻译：https://blog.csdn.net/hrsstudy/article/details/70767950
代码下载： https://github.com/pjreddie/darknet

SSD (Single Shot Multibox Detector)

YW元微 deep168.com

- SSD比Faster RCNN快，比YOLO v1准确率高
- SSD与YOLO v1结构对比如下图：



- YOLO算法的输入是448*448*3，输出是7*7*30，这7*7个grid cell一共预测98个bounding box。SSD算法是在原来VGG16的后面添加了几个卷积层来预测offset和confidence（相比之下YOLO算法是采用全连接层），算法的输入是300*300*3，采用conv4_3，conv7，conv8_2，conv9_2，conv10_2和conv11_2的输出来预测location和confidence。

SSD

YW元微
deep168.com

- SSD算法是一种直接预测bounding box的坐标和类别的object detection算法，没有生成proposal的过程。
- 针对不同大小的物体检测，传统的做法是将图像转换成不同的大小，然后分别处理，最后将结果综合起来，而**ssd利用不同卷积层的feature map进行综合也能达到同样的效果。**
- 算法的主网络结构是VGG16，将两个全连接层改成卷积层再增加4个卷积层构造网络结构。对其中5个不同的卷积层的输出分别用两个3*3的卷积核进行卷积，**一个输出分类用的confidence，每个default box生成21个confidence（这是针对VOC数据集包含20个object类别而言的）；一个输出回归用的localization，每个default box生成4个坐标值（x，y，w，h）**。另外这5个卷积层还经过priorBox层生成default box（生成的是坐标）。上面所述的5个卷积层中每一层的default box的数量是给定的。最后将前面三个计算结果分别合并然后传递给loss层。

SSD

YW元微
deep168.com

- 算法的结果：对于300*300的输入，SSD可以在VOC2007 test上有74.3%的mAP，速度是59 FPS(Nvidia Titan X)，对于512*512的输入，SSD可以有76.9%的mAP。相比之下Faster RCNN是73.2%的mAP和7FPS，YOLO是63.4%的mAP和45FPS。即便对于分辨率较低的输入也能取得较高的准确率。可见SSD并非像传统的做法一样以牺牲准确率的方式来提高检测速度。作者认为自己的算法之所以在速度上有明显的提升，得益于去掉了bounding box proposal以及后续的pixel或feature的resampling步骤。

SSD算法详解

YW元微 deep168.com

- SSD算法在训练的时候只需要一张输入图像及其每个object的ground truth boxes。
- 基本的网络结构是基于VGG16，在ImageNet数据集上预训练完以后用两个新的卷积层代替fc6和fc7，另外对pool5也做了一点小改动，还增加了4个卷积层构成本文的网络。VGG结构如下：

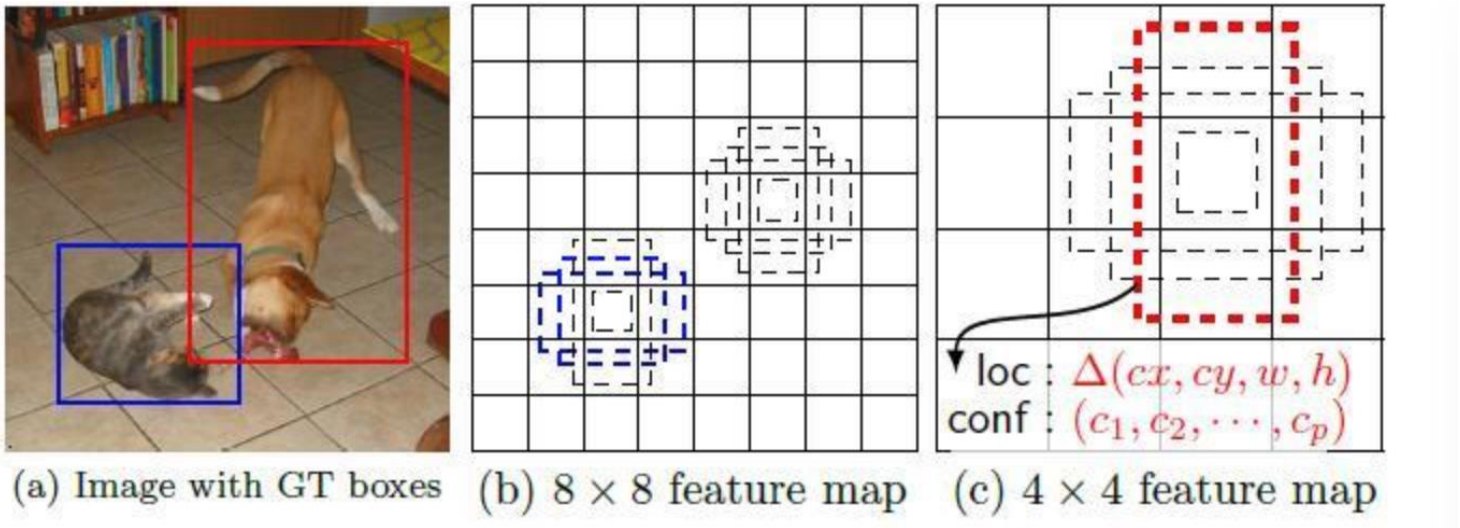
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

http://blog.csdn.net/u014380165

SSD算法详解

YW元微 deep168.com

- SSD的一个核心是作者同时采用lower和upper的feature maps做检测。如下图，有8*8和4*4两种大小的feature maps，而feature map cell就是其中的每一个小格。
- default box，是指在feature map的每个小格(cell)上都有一系列固定大小的box，如下图有4个。这里用到的default box和Faster RCNN中的anchor很像，在Faster RCNN中anchor只用在最后一个卷积层，但是在本文中，default box是应用在多个不同层的feature map上。

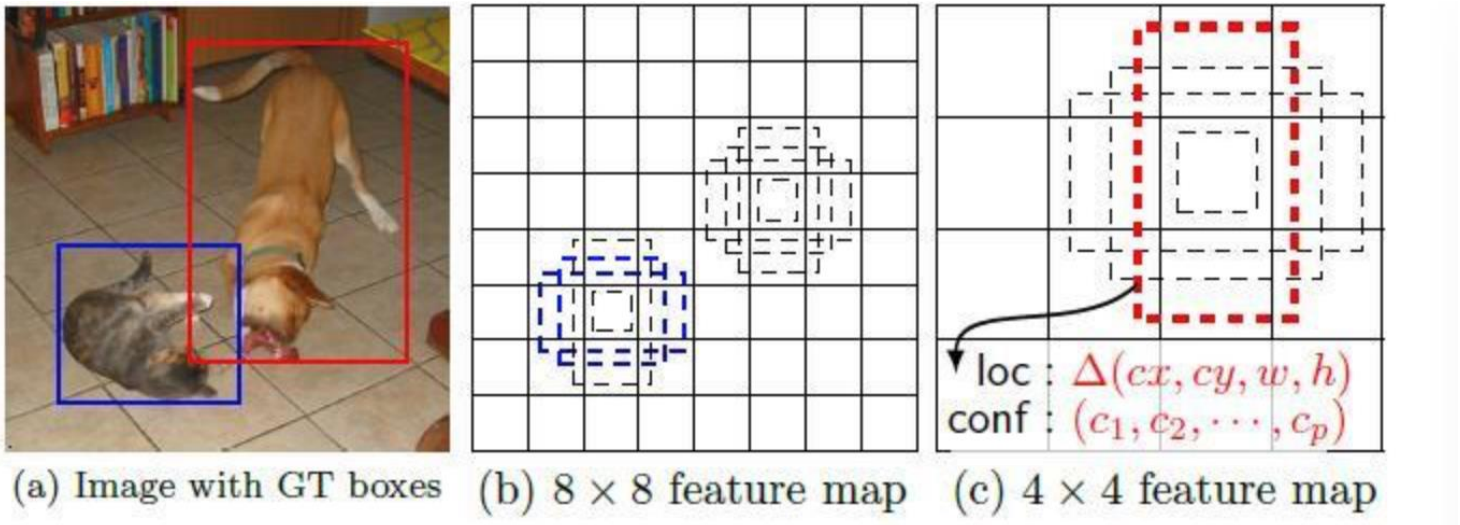


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

SSD算法详解

YW元微 deep168.com

- 假设每个feature map cell有k个default box，那么对于每个default box都需要预测c个类别score和4个offset，那么如果一个feature map的大小是m*n，也就是有m*n个feature map cell，那么这个feature map就一共有k*m*n个default box，每个default box需要预测4个坐标相关的值和c+1个类别概率（**实际代码是分别用不同数量的3*3卷积核对该层feature map进行卷积，比如卷积核数量为（c+1）*k对应confidence输出，表示每个default box的confidence，就是类别；卷积核数量为4*k对应localization输出，表示每个default box的坐标**）。



(a) Image with GT boxes (b) 8 × 8 feature map (c) 4 × 4 feature map

实验表明default box的shape数量越多，效果越好。

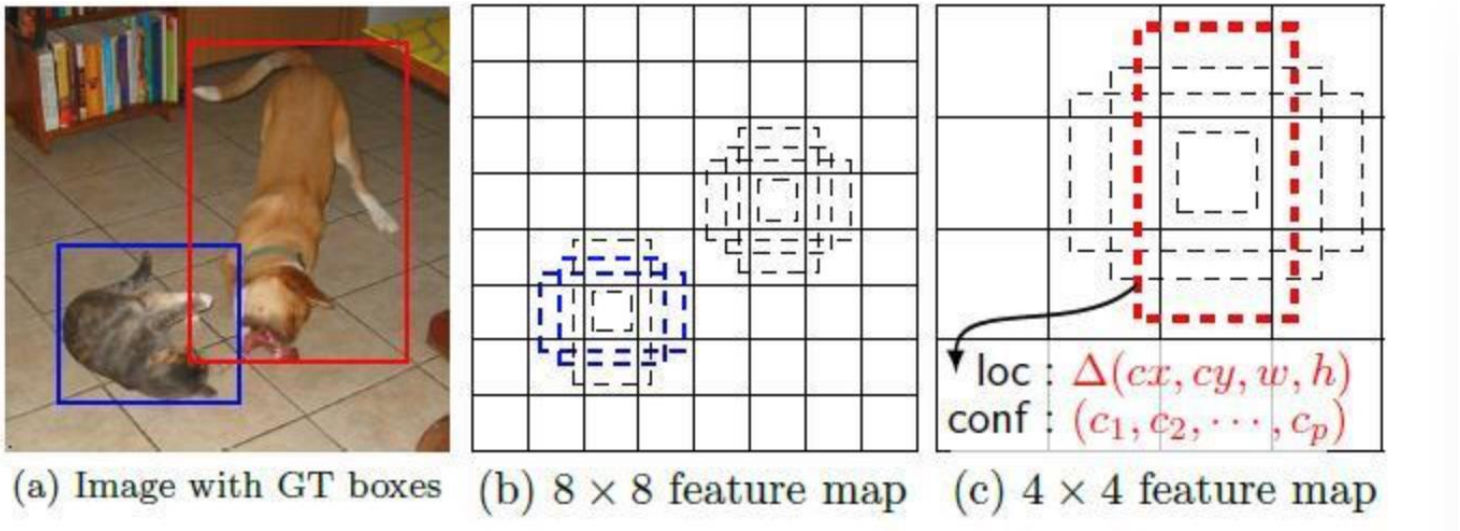
所以这里用到的default box和Faster RCNN中的anchor很像，在Faster RCNN中anchor只用在最后一个卷积层，但是在本文中，default box是应用在多个不同层的feature map上。

SSD算法详解

YW元微 deep168.com

- 在训练阶段，算法在一开始会先将这些default box和ground truth box进行匹配，比如蓝色的两个虚线框和猫的ground truth box匹配上了，一个红色的虚线框和狗的ground truth box匹配上了。所以一个ground truth可能对应多个default box。在预测阶段，直接预测每个default box的偏移以及对每个类别相应的得分，最后通过NMS得到最终的结果。

  图（c）说明对于每个default box，同时预测它的坐标offset和所有类的cor



SSD算法详解

YW元微 deep168.com

- 那么default box的scale（大小）和aspect ratio（横纵比）要怎么定呢？假设我们用m个feature maps做预测，那么对于每个featuer map而言其default box的scale是按以下公式计算的：

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m]$$

http://blog.csdn.net/u014380165

- 这里smin是0.2，表示最底层的scale是0.2,；smax是0.9，表示最高层的scale是0.9。
- 至于aspect ratio，用ar表示为下式：注意这里一共有5种aspect ratio：

$$a_r = \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$$

- 因此，对于每个feature map cell而言，一共有6种default box。
可以看出这种default box在不同的feature层有不同的scale，在同一个feature层又有不同的aspect ratio，因此基本上可以覆盖输入图像中的各种形状和大小的object。
- 显然，当default box和grount truth匹配上了，那么这个default box就是positive example（正样本），如果匹配不上，就是negative example（负样本），显然这样产生的负样本的数量要远远多于正样本。于是作者将负样本按照confidence loss进行排序，然后选择排名靠前的一些负样本作为训练，使得最后负样本和正样本的比例在3:1左右。

SSD

YW元微
deep168.com

- 论文： Single Short MultiBox Detector
- https://arxiv.org/abs/1512.02325
- 中文翻译：http://lib.csdn.net/article/deeplearning/57860
- 代码：https://github.com/weiliu89/caffe/tree/ssd

YOLO v2

YW元微
deep168.com

- YOLO v1的升级版，检测速度要快过其他检测系统（FasterR-CNN，ResNet，SSD），使用者可以在速度与精确度之间进行权衡。
- YOLO相较于其他的state-of-the-art的检测系统有一些缺陷，主要表现在两点：
  1. 和Fast R-CNN相比，YOLO会产生较多的bounding boxes的定位错误；
  2. 和基于region proposal的检测系统相比，YOLO的Recall较低。
- YOLO v2在v1的基础上，作出如下改进：
  1. 使用新的网络结构（darknet19）和技巧（Batch Normalization、High Resolution Classifier、Convolutional With Anchor Boxes等），提高了检测速度和检测精度；
  2. 提出了一种联合训练方法，可以同时使用检测数据集和分类数据集来训练检测模型，用分层的观点对物体分类，用检测数据集学习准确预测物体的位置，用分类数据集来增加可识别的类别量，提升鲁棒性。

YOLO v2

YW元微 deep168.com

- 机器视觉的发展有着神经网络越来越大、越来越深的趋势。现在的检测系统，更好的检测性能往往伴随着更大的神经网络或者是多个检测模型的集成。YOLO的目标是高精度实时检测，所以期望在不增大网络、精度不下降的前提下来对定位错误和低Recall进行改善，为此作者尝试了一系列方法（如图所示）：

	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**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

YOLO v2

YW元微
deep168.com

实验结论：

1. 将dropout层去掉，在每个卷积层添加了bach-normalization，mAP提高了2%；
2. 高分辨率分类器，输入为高分辨率数据由224*224变为448*448，mAP提高了4%；
3. 删除一个pooling层，得到高分辨率数据，在卷积层之后使用anchor代替全连接层，把输入448*448转为416*416，这样得到的feature map为奇数，feature map会有一个中心单元，物体落在中心位置的可能是比较大的，这样可以使用一个中心单元代替四个来预测物体；
4. 预测输出有变化： yolov1中，是每个grid cell负责预测各个类别的条件概率，而每个box会有coord和confidence值。在yolov2中，每个box都会有coords、confidence、和各类别的条件概率。
5. Anchor box大小是通过k-means在训练集中学习到的，重新设计聚类的距离计算来代替欧式距离，从而获得了更好的IOU scores。
6. YOLOv2网络只用到了卷积层和池化层，因此可以进行动态调整输入图像的尺寸。根据自己的要求调节精度和速度。

YOLO v2总结

YW元微
deep168.com

更快：

- YOLO 使用的是 GoogLeNet 架构，比 VGG-16 快，YOLO 完成一次前向过程只用 85.2 亿次运算，而 VGG-16 要 306.9 亿次，但是 YOLO 精度稍低于 VGG-16。
- YOLO v2 基于一个新的分类模型，有点类似于 VGG。YOLO v2 使用 3*3 的 filter，每次池化之后都增加一倍 Channels 的数量。YOLO v2 使用全局平均池化，使用 Batch Normilazation 来让训练更稳定，加速收敛，使模型规范化。最终的模型-Darknet19，有 19 个卷积层和 5 个 maxpooling 层，处理一张图片只需要 55.8 亿次运算，在 ImageNet 上达到 72.9% top-1 精确度，91.2% top-5 精确度。
- 在训练时，把整个网络在更大的448*448分辨率上Fine Turnning 10个epoches，初始学习率设置为0.001，这种网络达到达到76.5%top-1精确度，93.3%top-5精确度。

YOLO v2总结

YW元微
deep168.com

更强：

- 使用 WordTree 来混合来自不同的资源的训练数据，并使用联合优化技术同时在 ImageNet 和 COCO 数据集上进行训练，YOLO9000 进一步缩小了检测数据集与识别数据集之间的大小代沟。

详细细节参考论文： YOLO9000: Better, Faster, Stronger

谢谢观看！