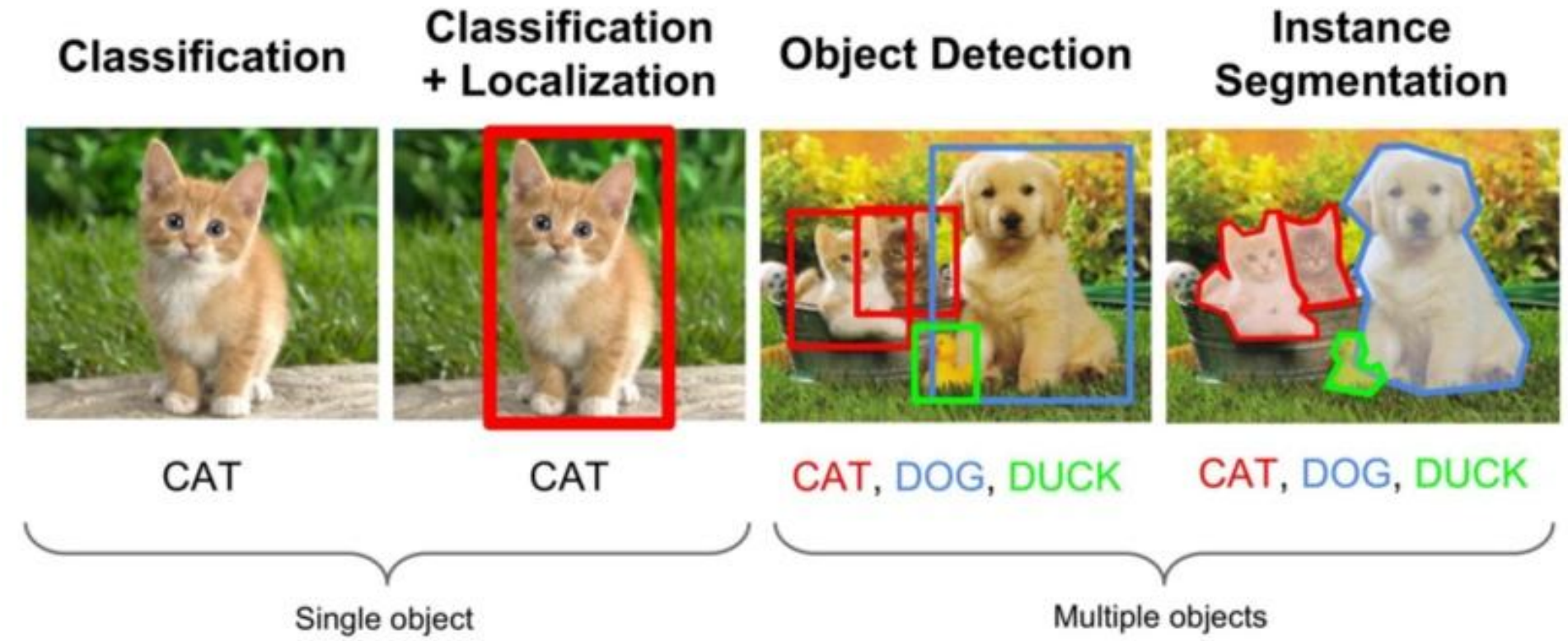
Objecct Detection with CNN

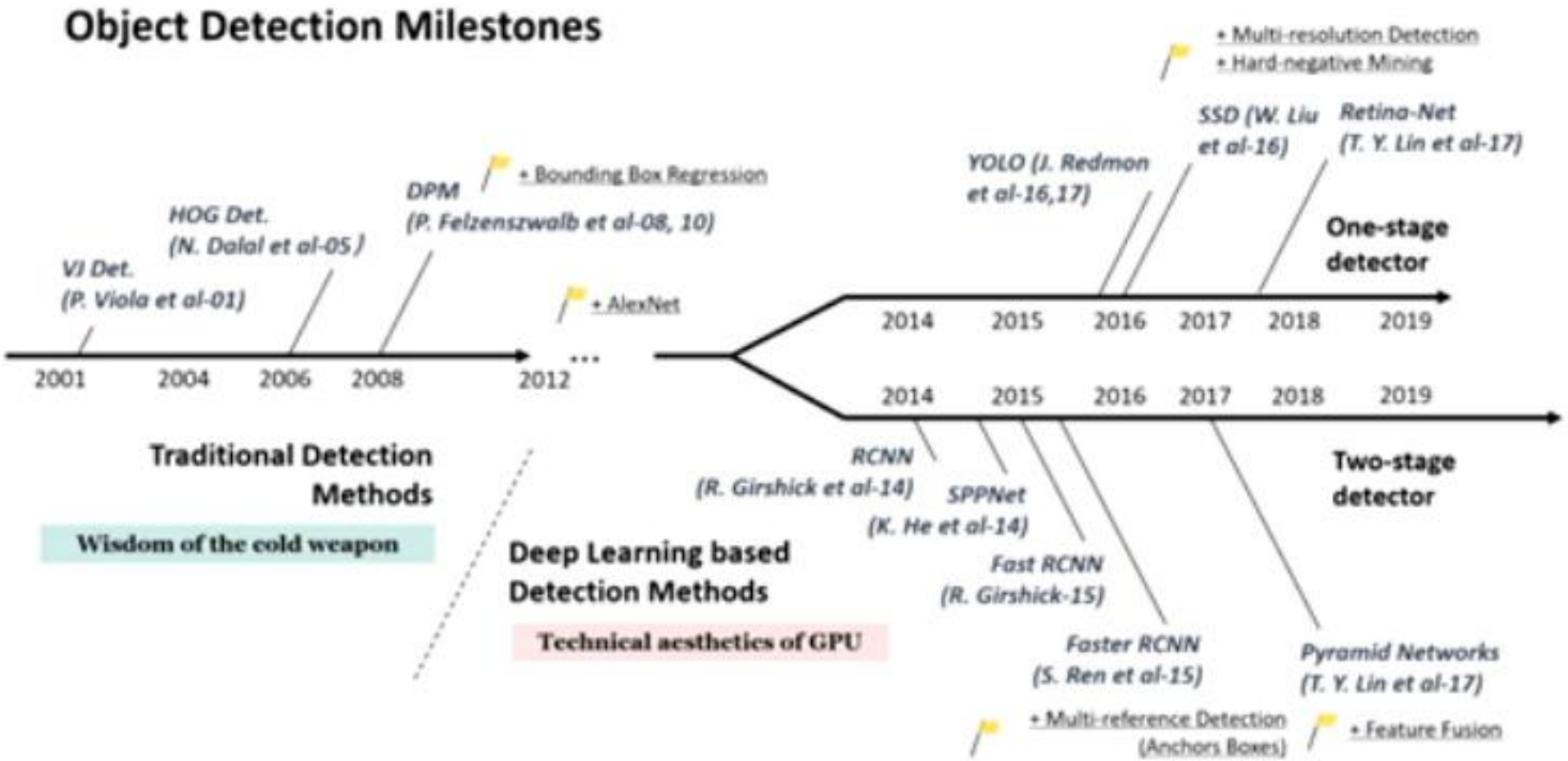
目标检测任务涉及计算机视觉应用中的两个最基本的问题：
“**这个物体是什么**”以及“**这个物体在哪里**”。



按照应用对目标检测进行分类，可以分成**两个研究主题**:
通用目标检测（General Object Detection）和**检测应用**（Detection Application）。
通用目标检测以探索**统一框架**下监测不同类型物体的方法为目标，从而模仿人类的视觉和认知;
检测应用指的是在**特定应用场景**下的检测，例如人脸识别、文本识别等。

近年来，**深度学习技术**的蓬勃发展为目标检测注入了新的动力，使得目标检测成为一个学术界的研究热点。
现如今，目标检测在**自动驾驶领域**、**机器人视觉领域**、**视频监控领域**等都有广泛的应用。

目标检测介绍| 发展历程



2014年前：**传统目标检测**时期

2014年后：基于**深度学习**的目标检测时期

传统目标检测的方法可以分为**三个阶段**:

（1）区域选择：这一步是为了**对目标的位置进行定位**。最初采用**滑动窗口**（slide window）的策略对整幅图像进行遍历，而且需要设置不同的尺度和不同的长宽比。这种穷举策略的缺点是：时间复杂度太高，产生冗余窗口太多，这也严重影响后续特征提取和分类的速度和性能。

（2）特征提取：设计一个具有鲁棒性的特征并不容易。但是，提取特征的好坏直接影响到分类的准确性。其中，这个阶段常用的特征有 SIFT、HOG等。

（3）分类：根据第二步提取到的特征**对目标进行分类**，分类器主要有 SVM，AdaBoost 等。

由于深卷积网络能够学习图像的鲁棒性和高层次特征表示，**在2014年，RCNN被提出了**，从那时起，目标检测开始以前所未有的速度发展。

基于深度学习的阶段，目标检测任务可分为两个关键子任务：**目标分类**和**目标定位**。

（1）目标分类任务负责判断输入图像或所选择图像区域（Proposals）中是否有感兴趣类别的物体出现，输出**一系列带分数的标签**表明物体出现在图像区域中的**可能性**；

（2）目标定位任务负责确定图像区域中物体的位置和范围，输出物体的包围盒、物体中心或物体的闭合边界等，通常使用方形包围盒，即**Bounding Box**用来表示物体的位置信息。

目标检测可以**分为两类**：**双阶段检测（two-stage detection）**和**单阶段检测（one-stage detection）**。

（1）双阶段检测将检测框定为一个**“从粗到细”**的过程。问题分为两个阶段，首先是**产生候选区域（Region Proposals）**，包含目标大概的位置信息，然后对候选区域进行**分类和位置精修。典型算法**有**RCNN**（2014年）、SPPNet（2014年）、Fast-RCNN（2015年）、FPN（2017年）。

（2）单阶段检测将其定义为**“一步到位”**，即**不需要产生候选区域**，可以直接产生物体的类别概率和位置坐标值。**典型算法**有**You Only Look Once（YOLO）**（2015年）、**Single Shot MultiBox Detector（SSD）**（2015年）、RetinaNet（2017年）。

一般情况下，Two-Stage算法在**准确度**上有优势，而One-Stage算法在**速度**上有优势。

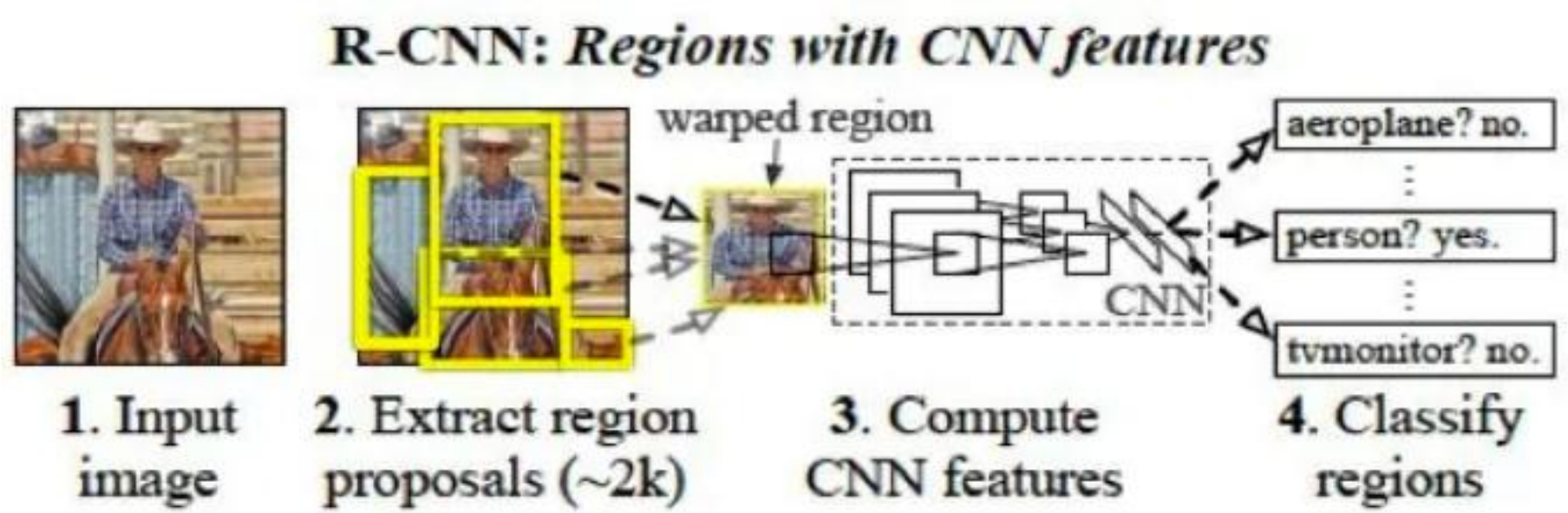
R-CNN | 起源和内容概要

- R-CNN算法是由2014 CVPR会议论文里提出，特点就是将区域特征和卷积神经网络方法结合起来，属于Two-stage方法。

01 候选区域的筛选

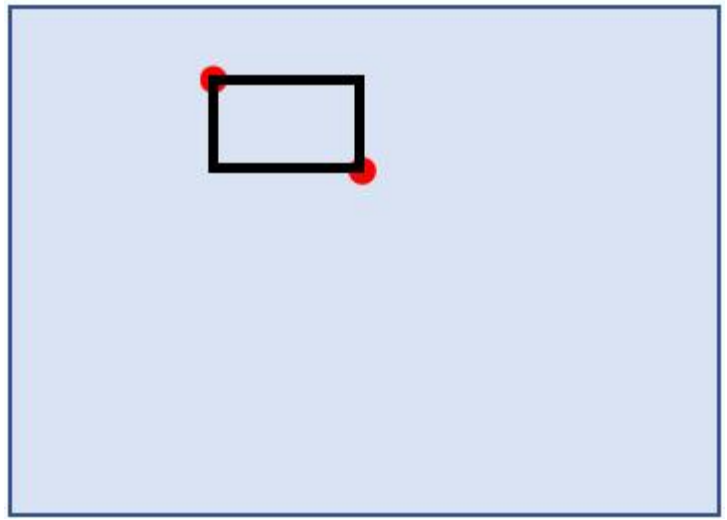
02 CNN提取特征向量

03 SVM分类



R-CNN | possible boxes

- 存在多少可能的窗口?
  - Every pair of pixels = 1 box
  - $\binom{N}{2}$ = $O(N^2)$
  - For 800 x 600 image, N = 480K
  - 2.3×10^{11}



R-CNN | Selective Search

- 传统方法：滑动窗口法
  - 缺陷：1.计算开销较大，2.目标位置不够精确
- SS目标：根据颜色、纹理、大小和形状的兼容性，计算相似区域的层次分组
  - 生成一个区域集R
  - 计算R里每个相邻区域的相似度S，
  - 找出相似度最高的两个相邻区域
  - 合并后重新加入区域集中
  - 将原有的相关子集删除
  - 继续计算相似度，直到相似度为0
  - 完成区域筛选。

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

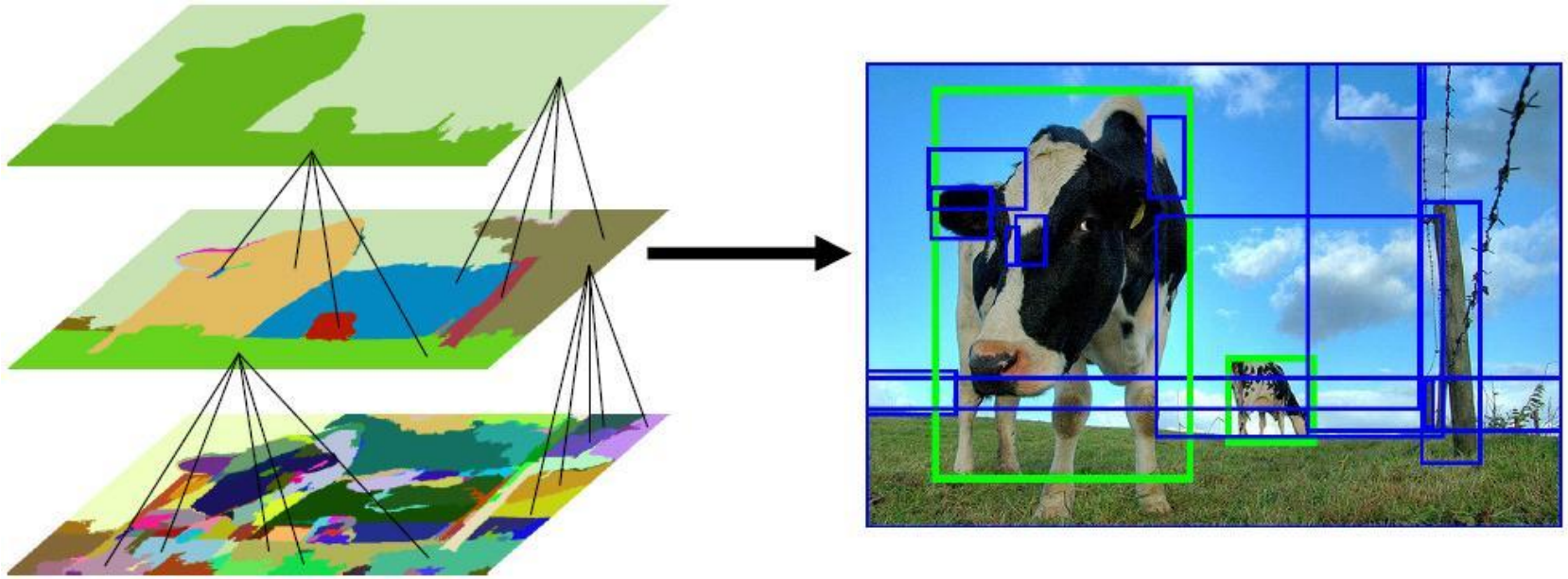
Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$
  $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$
  Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$
  Calculate similarity set S_t between r_t and its neighbours
  $S = S \cup S_t$
  $R = R \cup r_t$

Extract object location boxes L from all regions in R

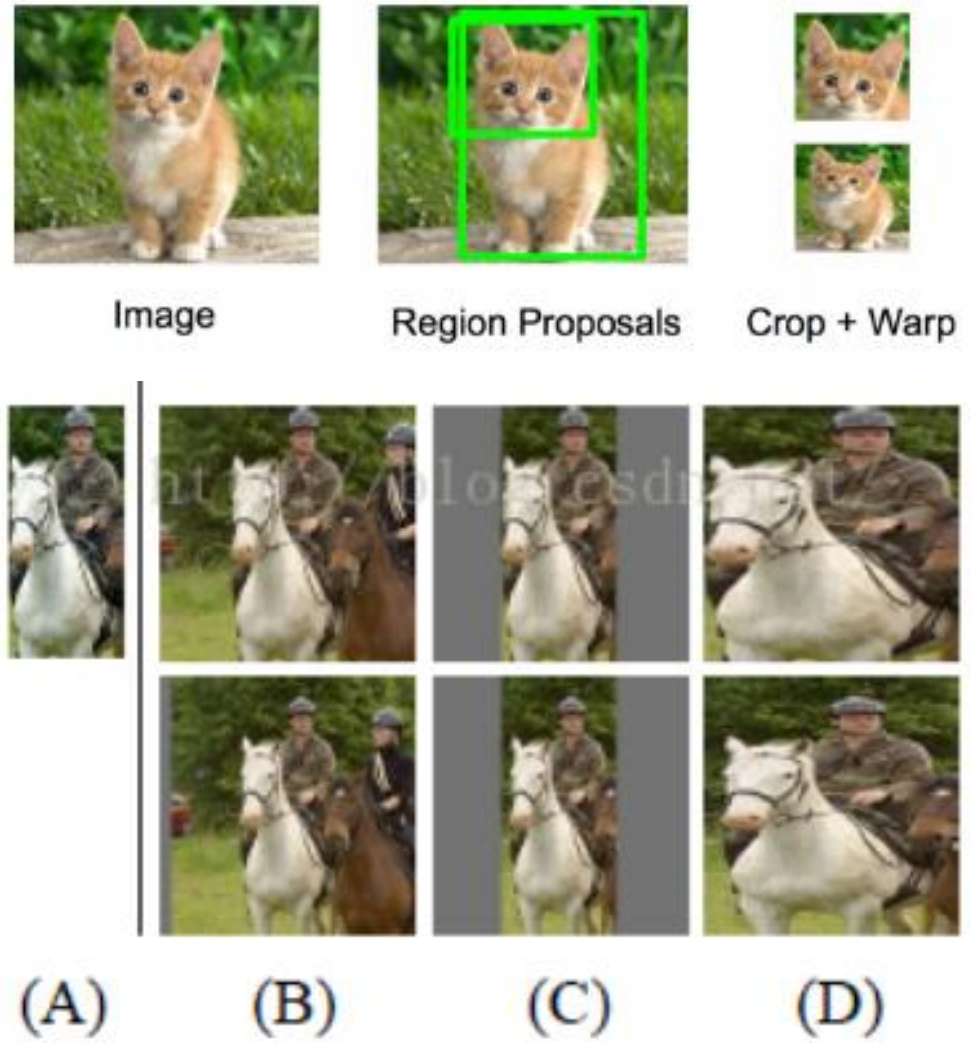
R-CNN | Object proposals

Selective Search for Object Recognition
J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders
In International Journal of Computer Vision 2013.



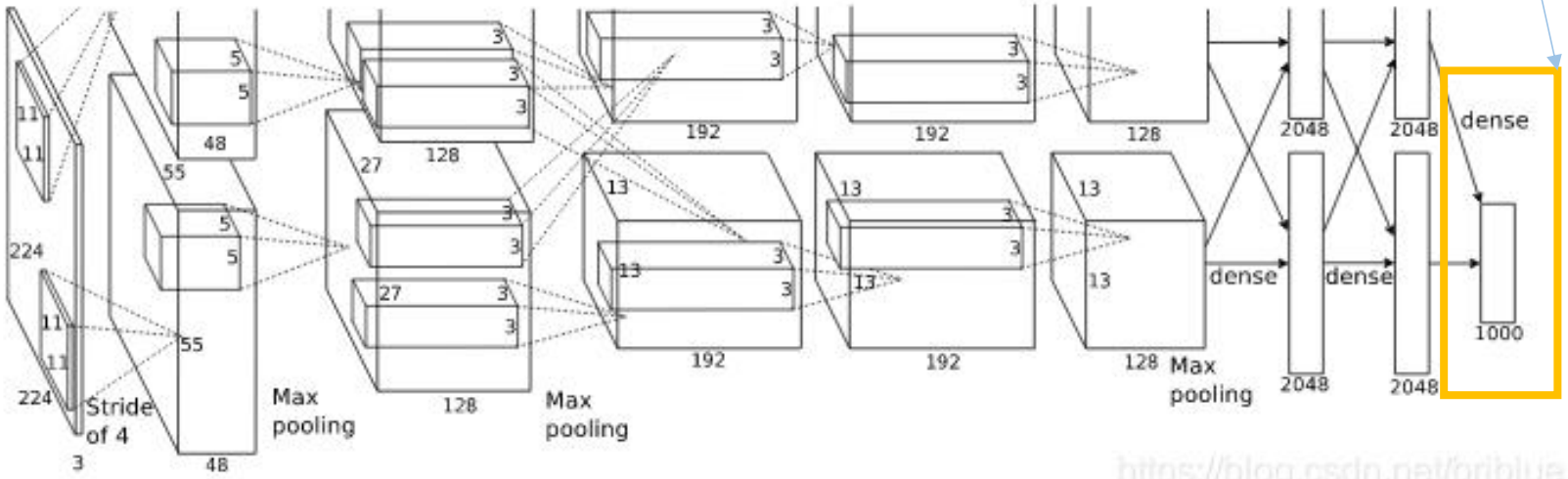
R-CNN | CNN特征提取

- 候选区域预处理（crop+wrap）
  - Paper中筛选了2000个候选框
  - **各项异性缩放**：不顾长宽比例，全部缩放到CNN输入的大小227*227
  - 各项同性缩放：拓展边界、颜色均值填充
  - 文献中发现异性缩放精度最高
- 模型预训练和微调
  - 网络架构选用相对于VGG更简单的Alexnet
  - 针对特定的任务，对模型进行微调



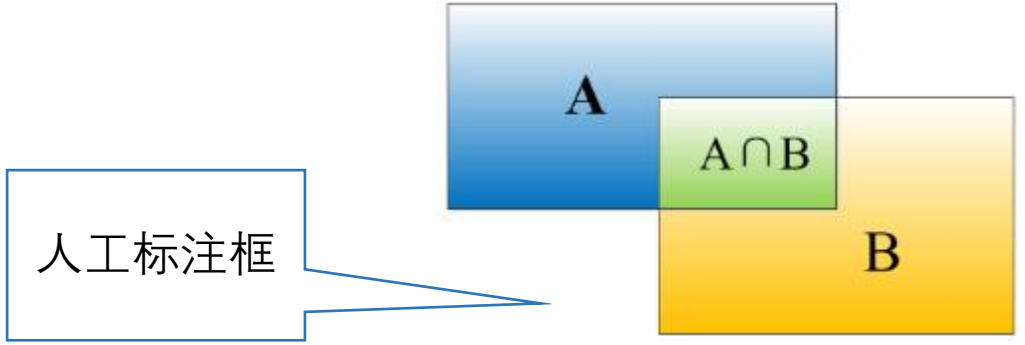
R-CNN | CNN特征提取

- Alexnet包含5个卷积层和2个全连接层
- 微调（fine-tuning）
  - 将预训练阶段的CNN模型的最后一层的给替换掉，1000替换成N+1个输出的神经元(N表示物体的类别数，1表示还有一个背景类)
  - 将处理好的候选框输入模型中训练
  - 大大提高后面SVM分类的精度
- 输入每个候选框，输出一个4096维的特征向量



R-CNN | SVM训练、分类

- CNN+SVM（正负样本定义方式各有不同）
  - CNN在训练的时候，对训练数据做了比较宽松的标注
  - SVM适用于少样本训练，对训练样本的IOU要求严格
- IOU阈值选择
  - 定义：两个bounding box的重叠度，即$IOU = (A \cap B)/(A \cup B)$
  - CNN阶段，IOU>0.5记为物体类别，否则为背景
- 提取最后一个全连接层的输出的特征向量，输入SVM中进行分类
  - 每一类物体对应一个分类器
  - IOU的阈值取0.3，大于0.3为正样本
- 输出分类结果：正负样本



R-CNN | NMS

- 非极大值抑制（NMS）
  - 目标是消除重复和冗余的重叠框
  - 本质是寻找局部最大值
- 步骤
  - 按照得分降序或升序排列
  - 选取得分最大的框
  - 和其余的框比较重叠度IOU
  - 丢弃超过阈值的框，留下最大得分框
  - 对于剩下框执行相同的操作步骤

R-CNN | Bounding box回归

- 修正候选框的位置
- 利用CNN输出的特征预测边框位置(x, y, w, h)、
- 回归的目标值是预测的框和SS中的差值(dx, dy, dw, dh)

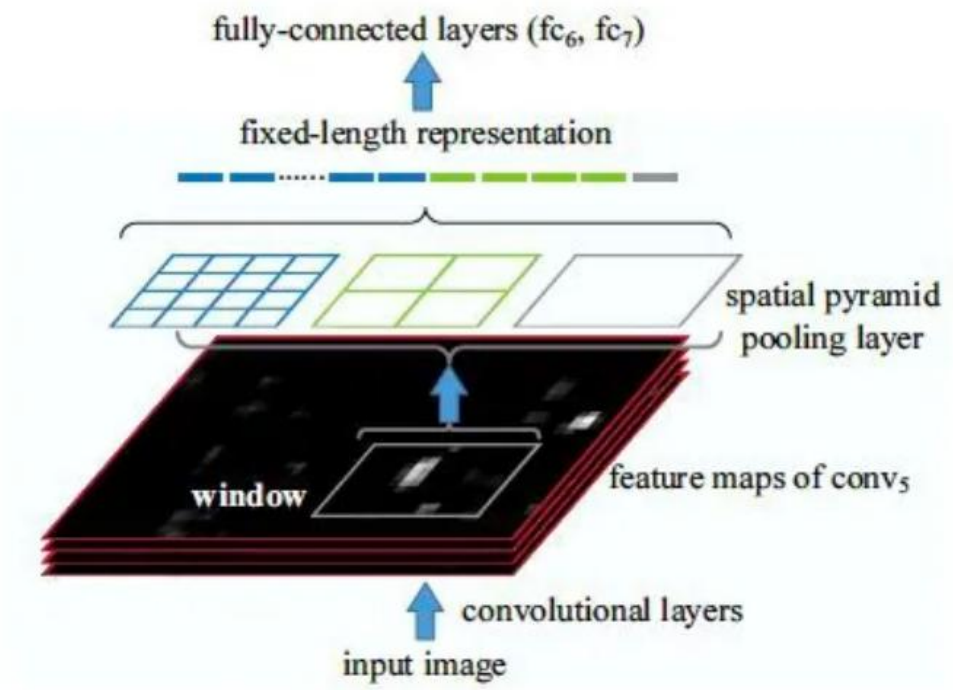
R-CNN results on PASCAL

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%
R-CNN	54.2%	50.2%
R-CNN + bbox regression	58.5%	53.7%

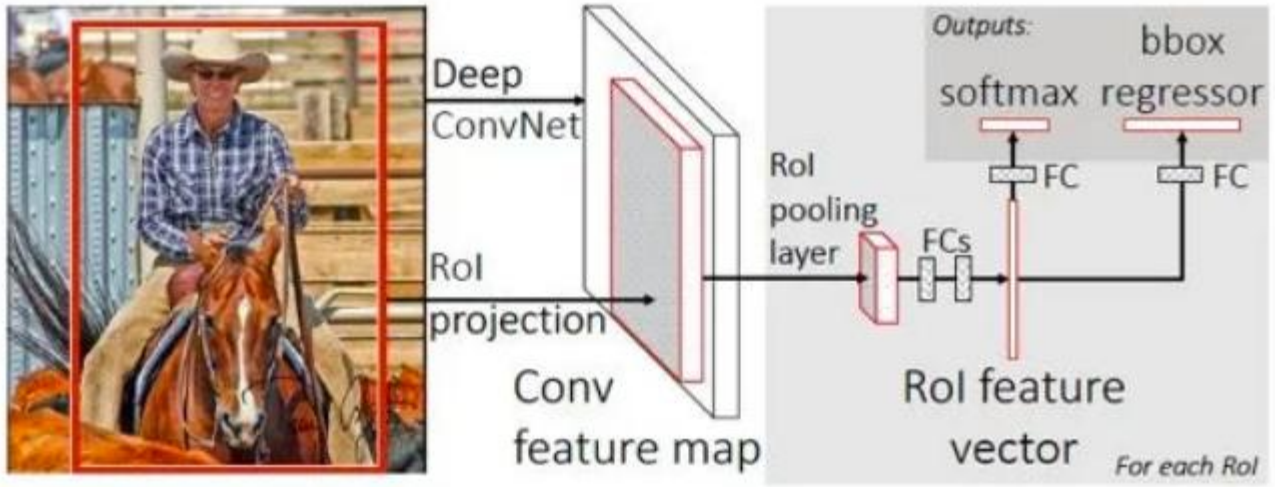
Slide credit : Ross Girshick

R-CNN | 扩展方法

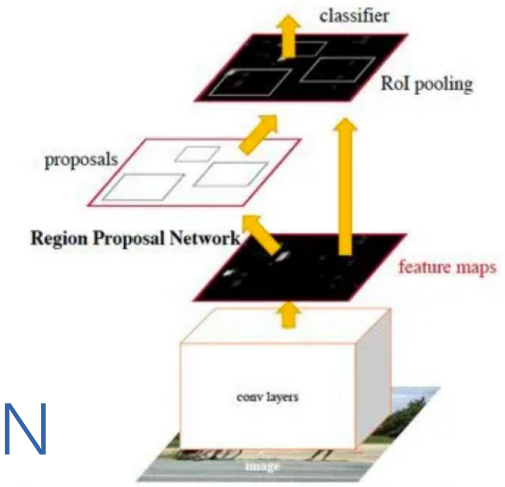
SSP-net

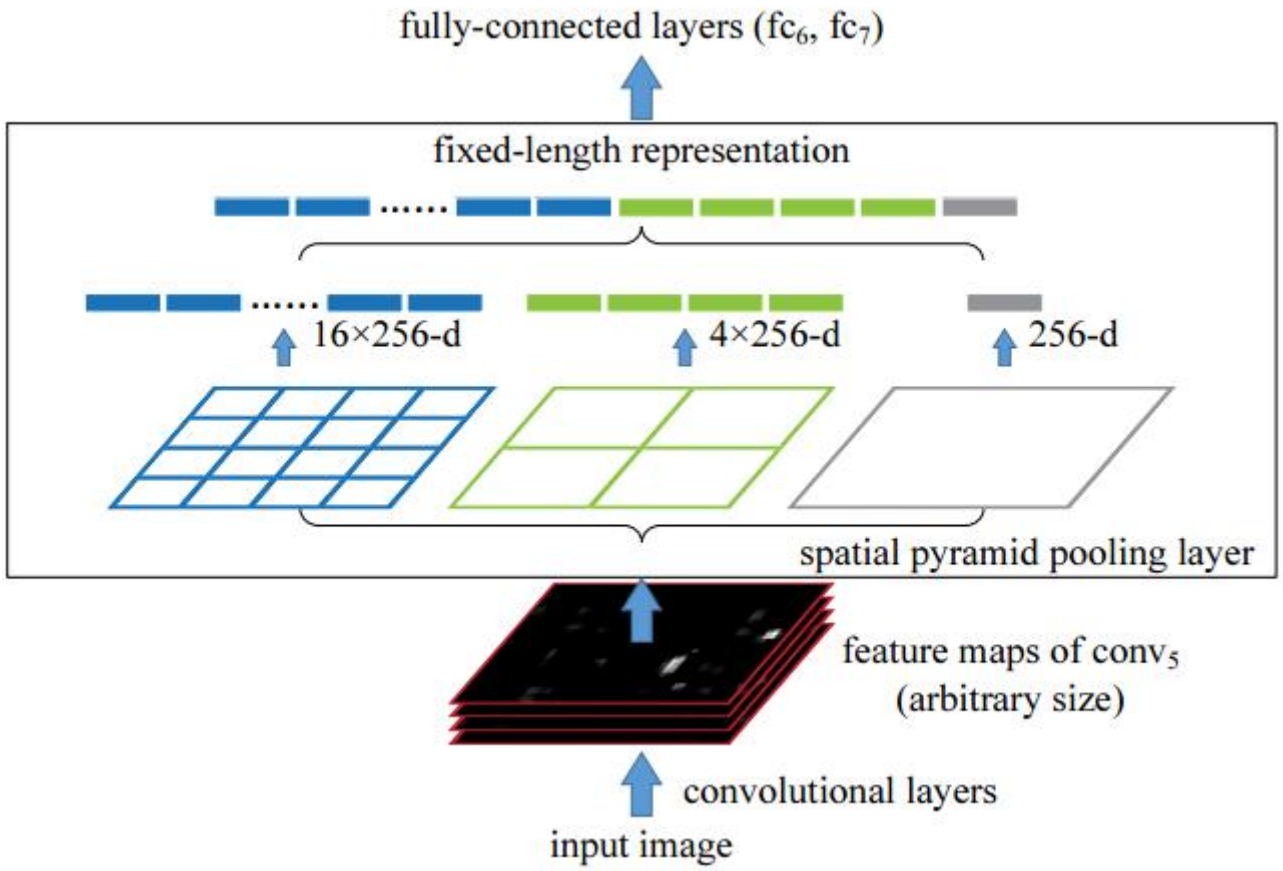


Fast-CNN



Faster-CNN

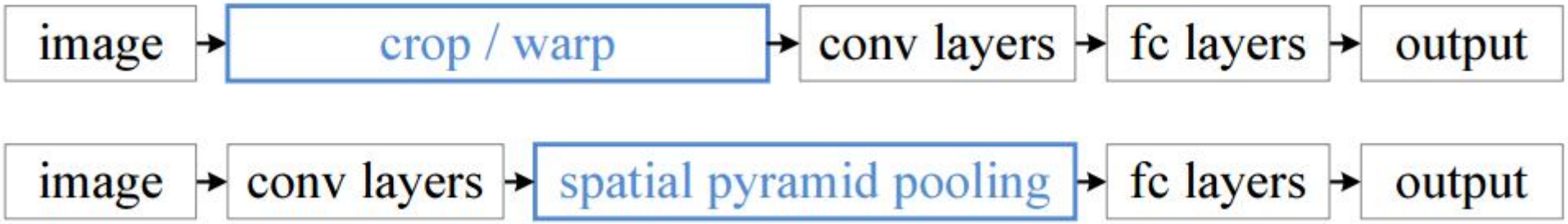



fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$
(arbitrary size)
convolutional layers
input image

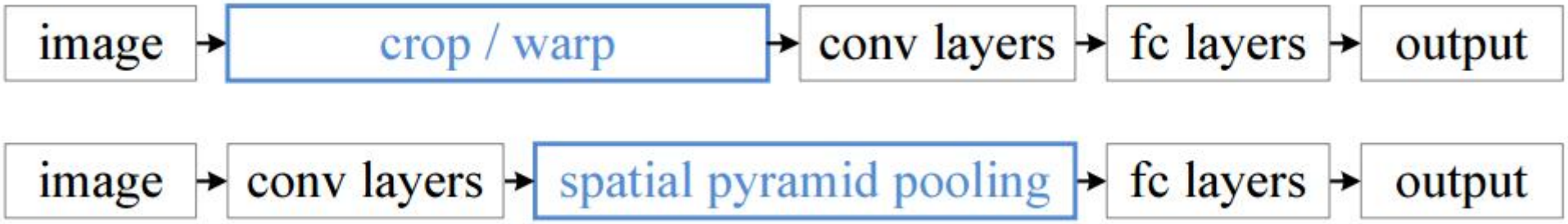
SSP-net

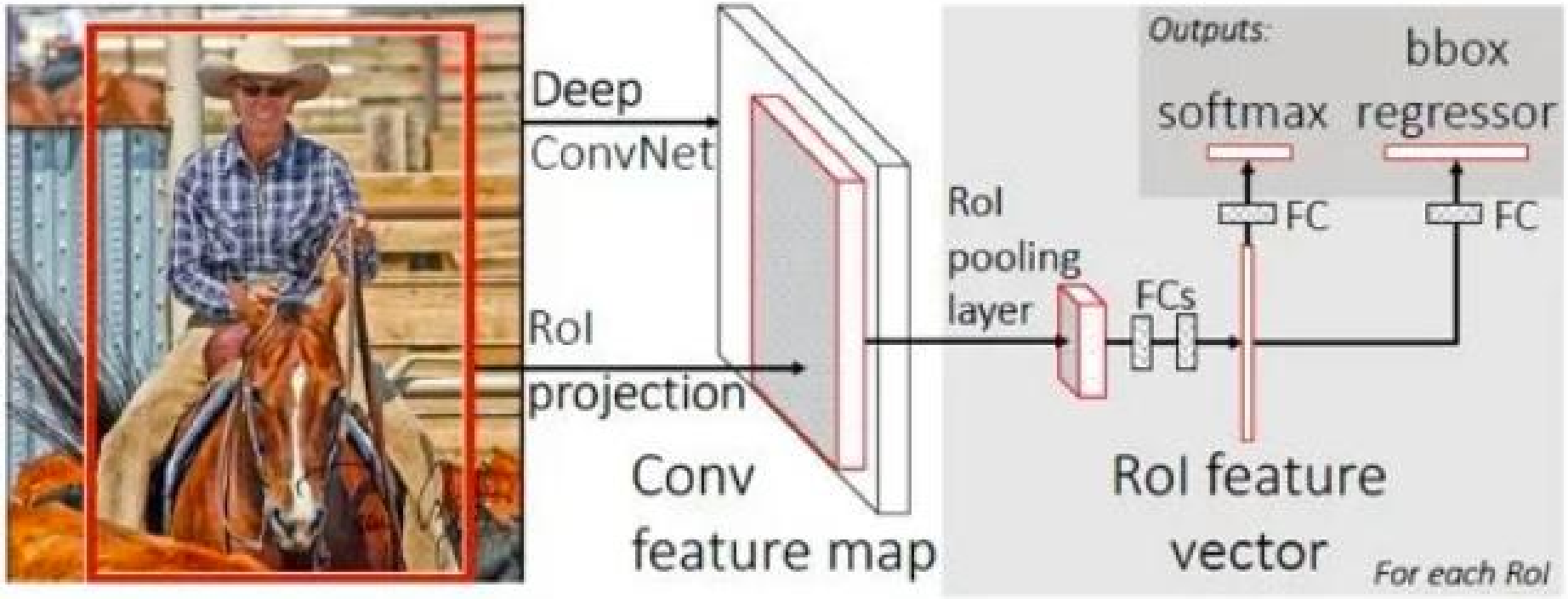
空间金字塔池化

当网络输入的是一张任意大小的图片，这个时候我们可以一直进行卷积、池化，直到网络的倒数几层的时候，也就是我们即将与全连接层连接的时候，就要使用金字塔池化，使得任意大小的特征图都能够转换成固定大小的特征向量，这就是空间金字塔池化的意义（多尺度特征提取出固定大小的特征向量）。



假设输入feature map的尺寸为H x W x C，使用一个H x W尺寸的pooling层进行处理，那么每一个通道C变成了一个值，整个输入得到了一个C维的输出；再分别用H/2 x W/2和H/4 x W/4尺寸的pooling层处理，得到了4xC和16xC维的输出，把三个结果concat在一起变成了一个21xC维的输出，其大小和输入的H与W无关。

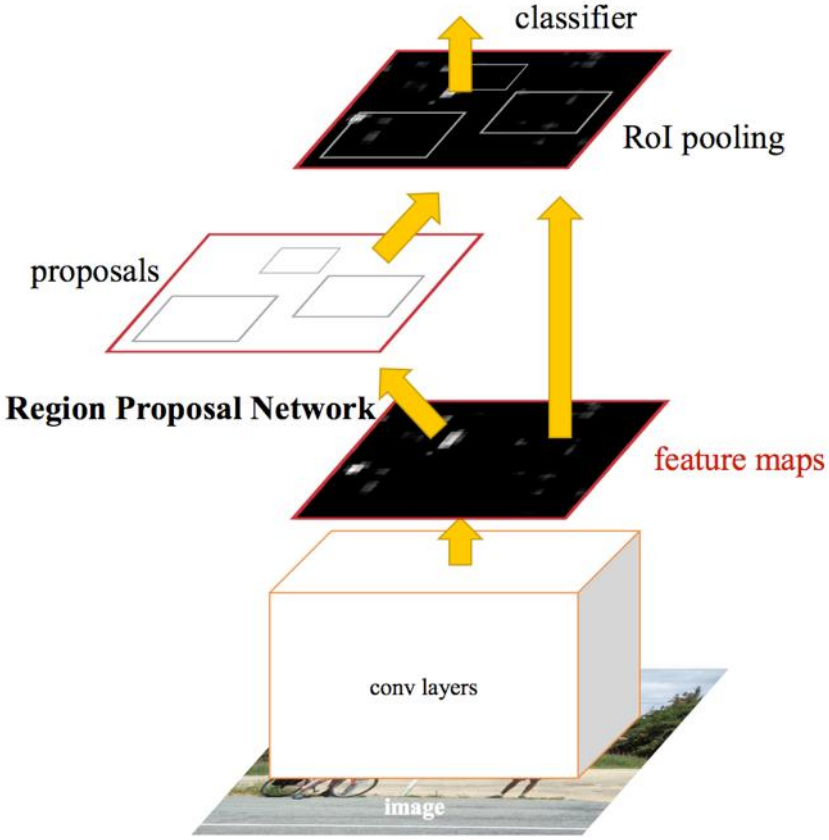



Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

Fast-RCNN

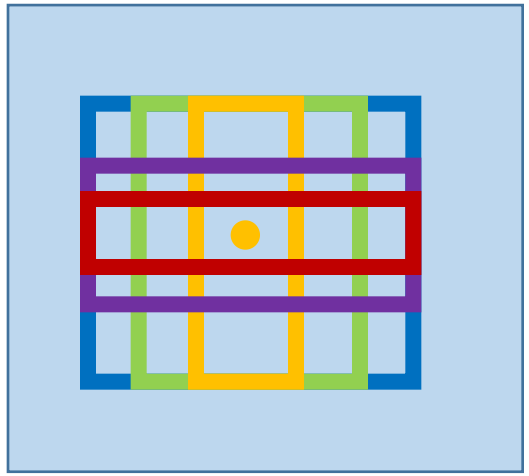
Fast R-CNN

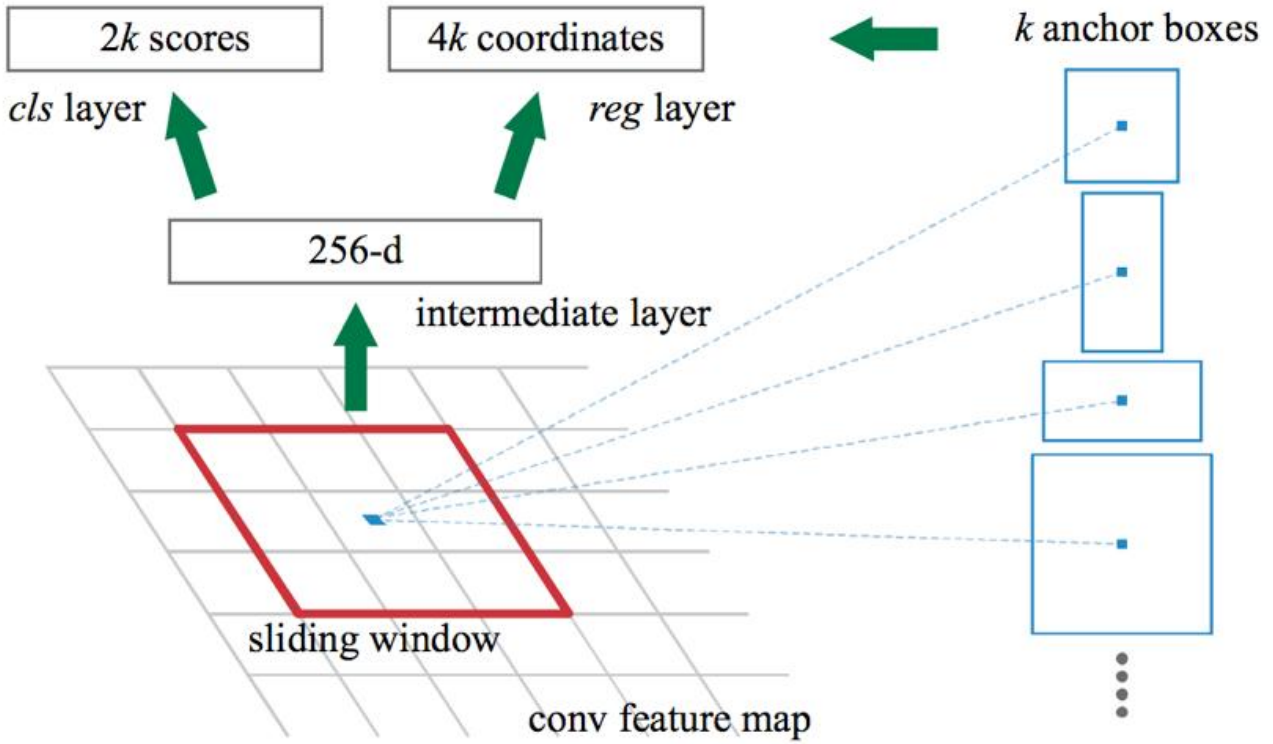
	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Speedup	146x	1x
mean AP	66.9	66.0


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Faster-CNN

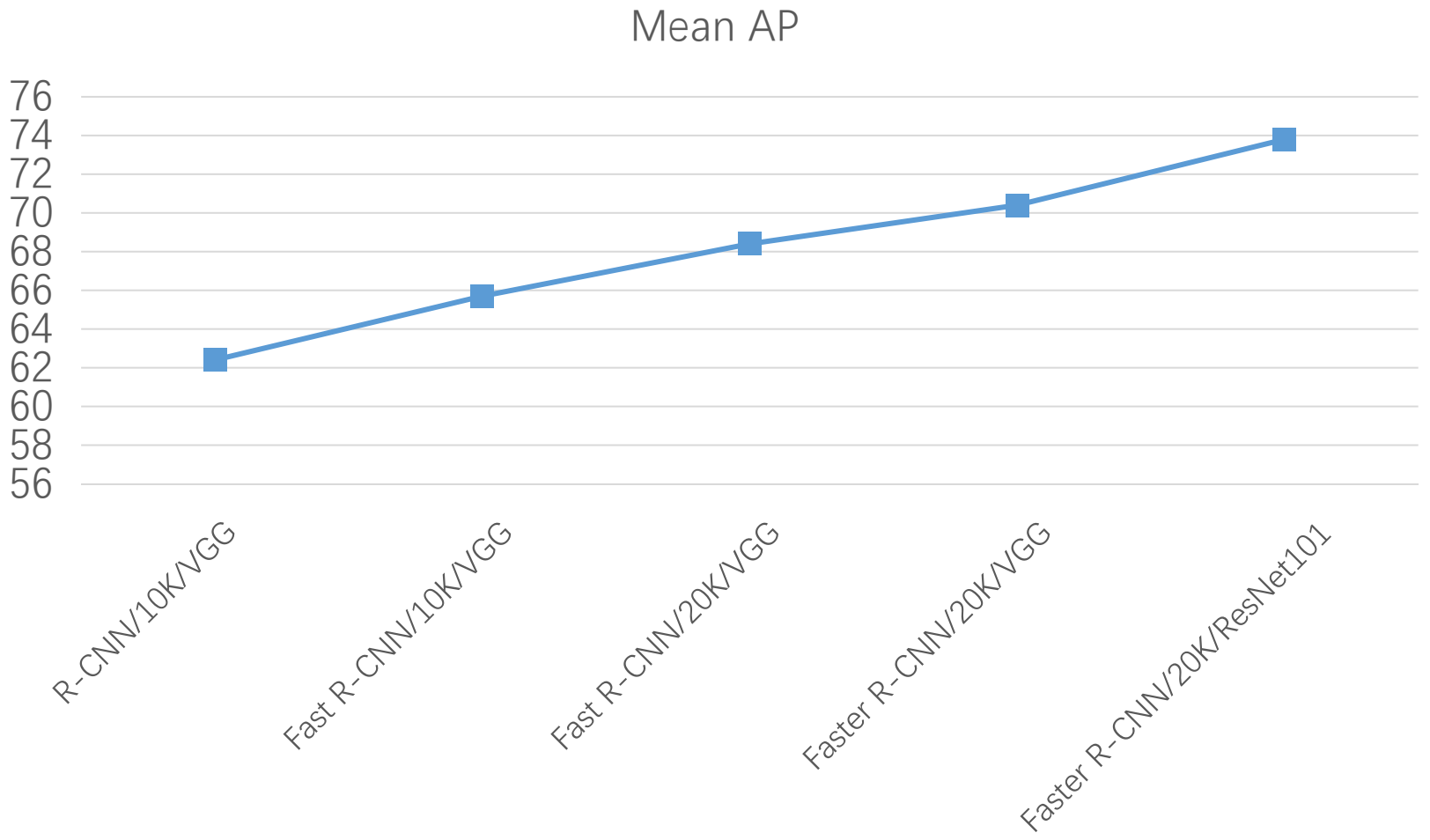
- 每个位置，考虑不同大小和长宽比例的boxes



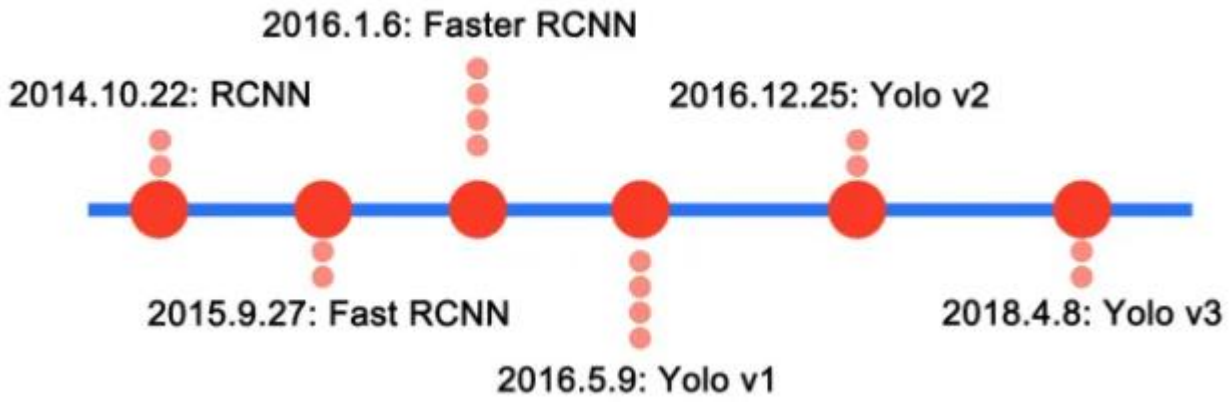


Method	Training data	mean AP (PASCAL VOC 2012 Test)
Fast R-CNN	VOC 12 Train (10K)	65.7
Fast R-CNN	VOC07 Trainval + VOC 12 Train	68.4
Faster R-CNN	VOC 12 Train (10K)	67.0
Faster R-CNN	VOC07 Trainval + VOC 12 Train	70.4

The R-CNN family of detectors



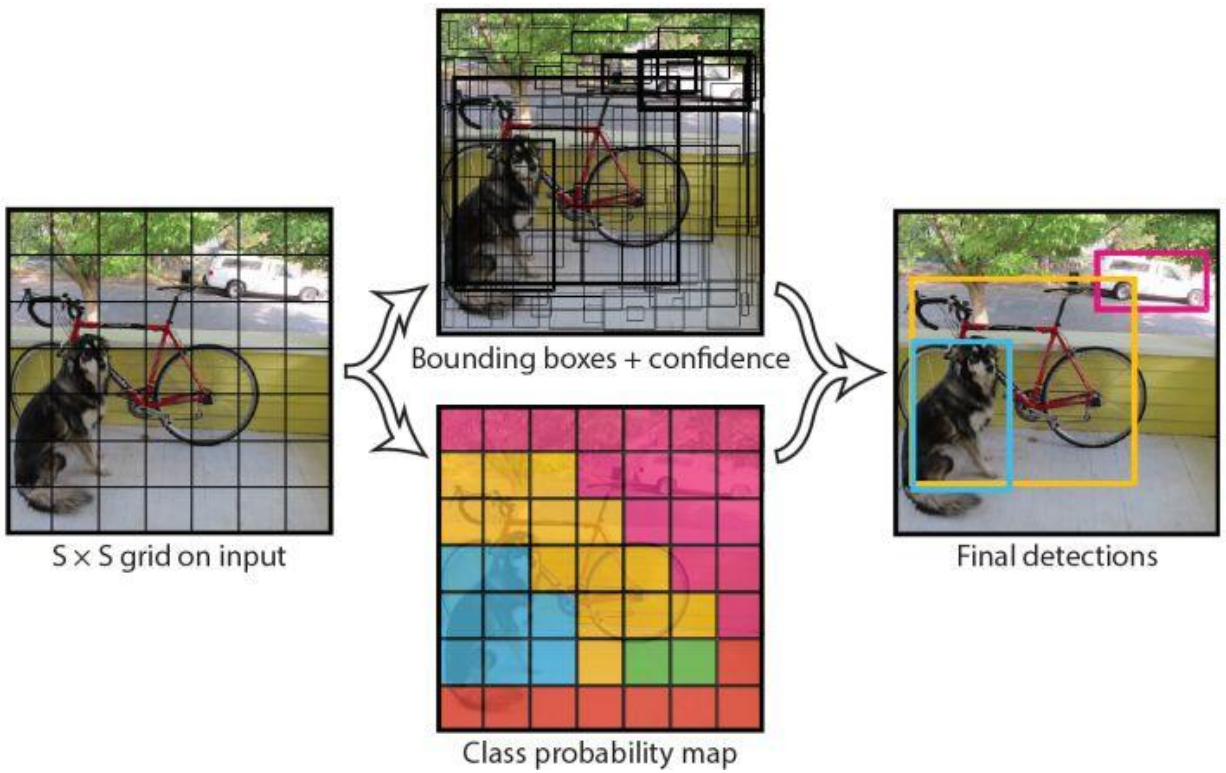
YOLO介绍


2016.1.6: Faster RCNN
2014.10.22: RCNN
2016.12.25: Yolo v2
2015.9.27: Fast RCNN
2018.4.8: Yolo v3
2016.5.9: Yolo v1

YOLO介绍 | 简介

YOLO是由Joseph Redmon提出的一种一种基于深度神经网络的对象识别和定位算法，其最大的特点是运行速度很快，可以用于实时系统，属于one-stage方法。

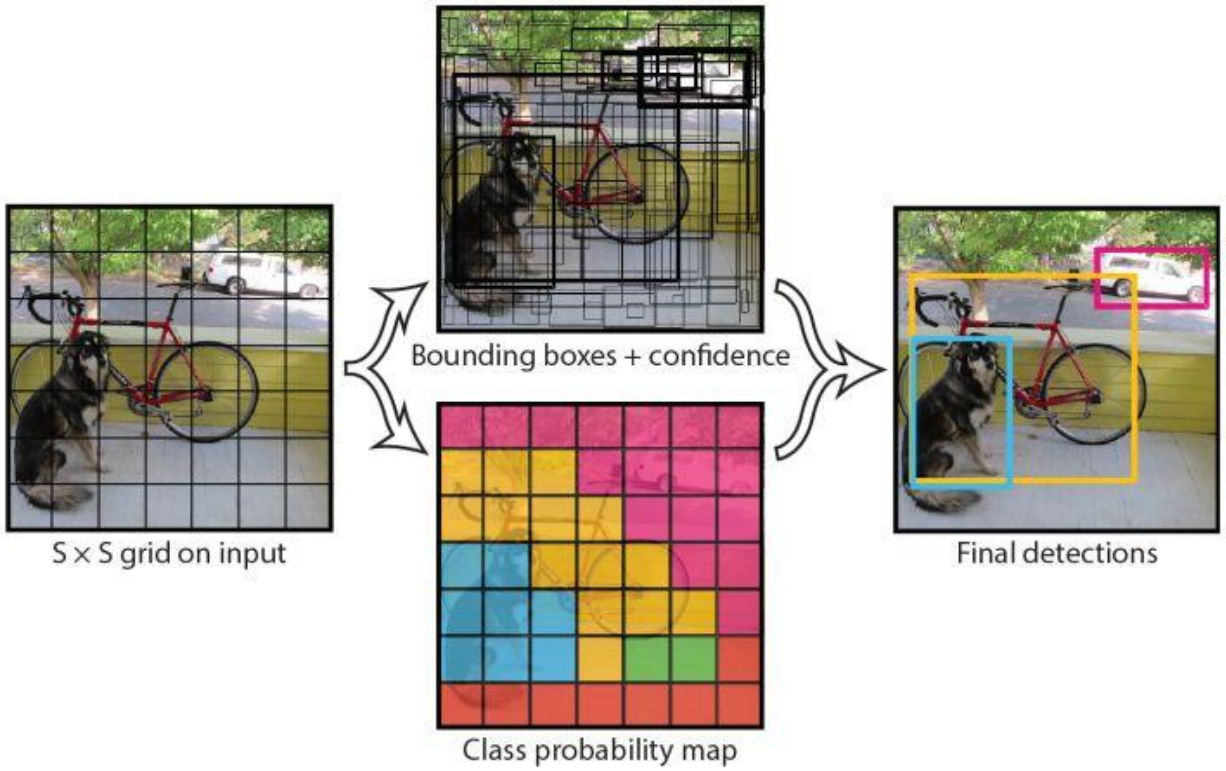
目前YOLO已经有YOLOv1，YOLOv2，YOLOv3，YOLOv4，YOLOv5等多个版本。



YOLO介绍 | YOLOv1

核心思想：将整张图片作为网络的输入，直接在输出层对Bounding Box的位置和类别进行回归。

实现方法：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。

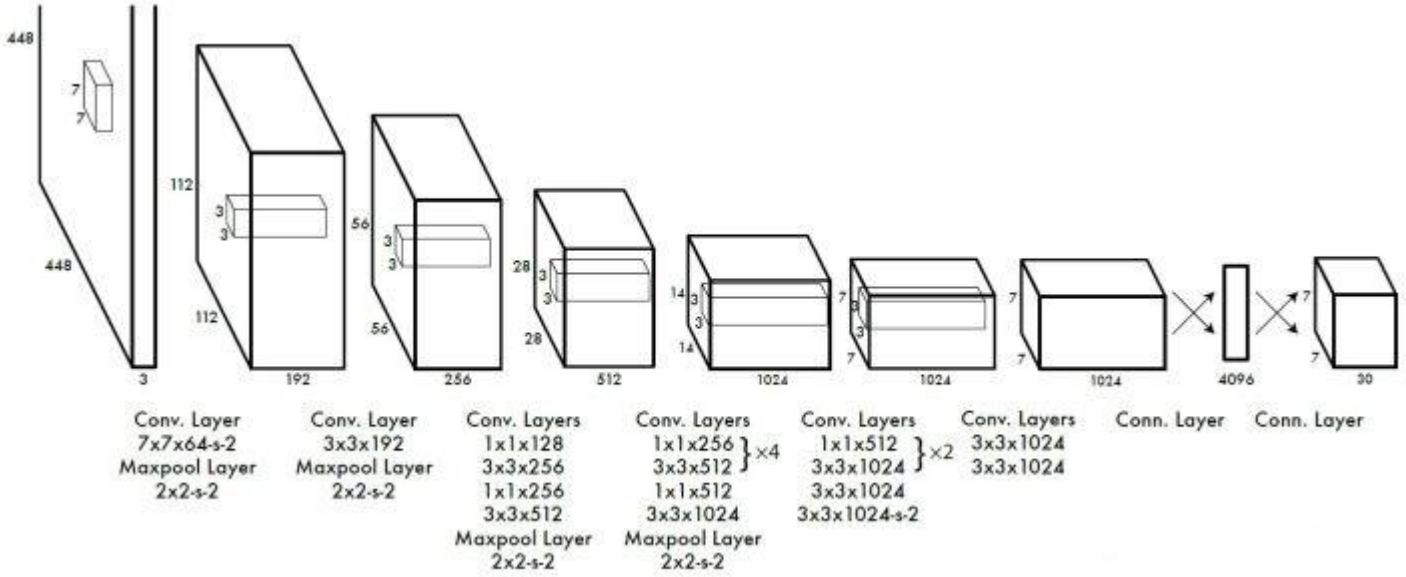


YOLO介绍 | YOLOv1

每个网格需要预测B个Bounding Box的位置信息和confidence信息，一个Bounding Box对应着四个位置信息和一个confidence信息。confidence代表了所预测的box中含有object的置信度和这个box预测的有多准两重信息：

$$\Pr(Object) * IOU_{pred}^{truth}$$

SxS个网格，每个网格要预测B个bounding box，还要预测一个类别信息，记为C类。每个bounding box要预测(x, y, w, h)和confidence共5个值，输出就是S x S x (5*B+C)的一个tensor。



PASCLA VOC数据集

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:

$$\Pr(Class_i \mid Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth}$$

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。

损失函数：sum-squared error loss

缺点：1. 8维的localization error和20维的classification error同等重要是不合理的；

2. 如果一个网格中没有object，那么就会将这些网格中的box的confidence push到0，相比于较少的有object的网格，会导致网络不稳定甚至发散。

方法：1. 更重视8维的坐标预测，给这些损失前面赋予更大的loss weight。

2. 对没有object的box的confidence loss，赋予小的loss weight。

3. 有object的box的confidence loss和类别的loss的loss weight正常取1。

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_2)^2 + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{\omega_i} - \sqrt{\hat{\omega}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

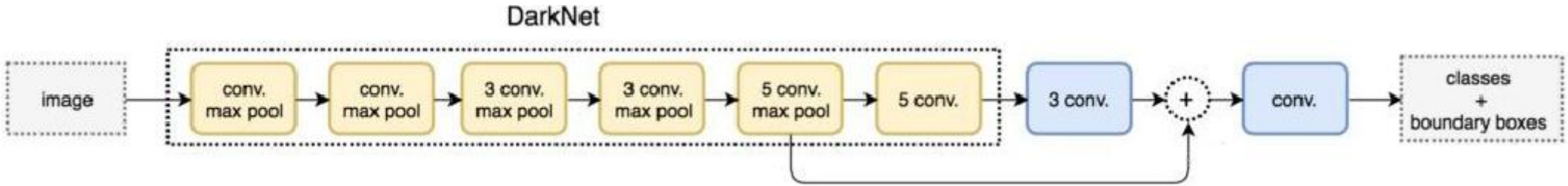
YOLOv1的优缺点：

优点：1. 速度快，处理速度可以达到45fps。
2. 泛化能力强，可以广泛适用于其他测试集。
3. 背景预测错误率低，因为是整张图片放到网络里面进行预测。

缺点：1. 输入尺寸固定：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。其它分辨率需要缩放成此固定分辨率；
2. 占比小的目标检测效果不好：每个网格最多只预测出一个物体。当物体占画面比例较小时，每个格子包含多个物体，但却只能检测出其中一个。

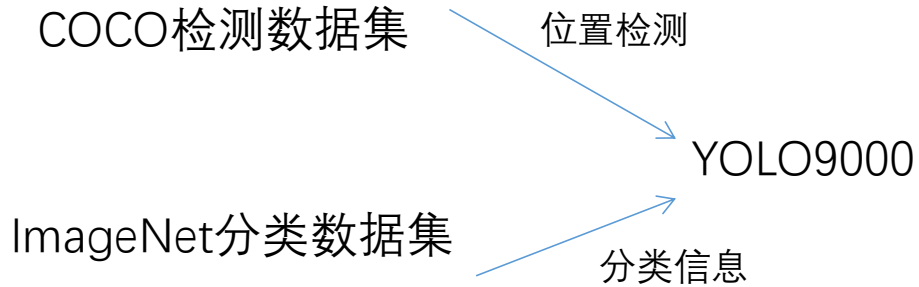
YOLO介绍 | YOLOv2

为提高物体定位精准性和召回率，YOLO 作者在2017年提出了YOLOv2，相比 v1 提高了训练图像的分辨率，从预测更准确，速度更快，识别对象更多这三个方面进行了改进。其中识别更多对象也就是扩展到能够检测9000种不同对象，称之为YOLO9000。



训练方法：联合训练算法

同时在检测数据集和分类数据集上训练物体检测器（Object Detectors），用检测数据集的数据学习物体的准确位置，用分类数据集的数据来增加分类的类别量、提升健壮性。



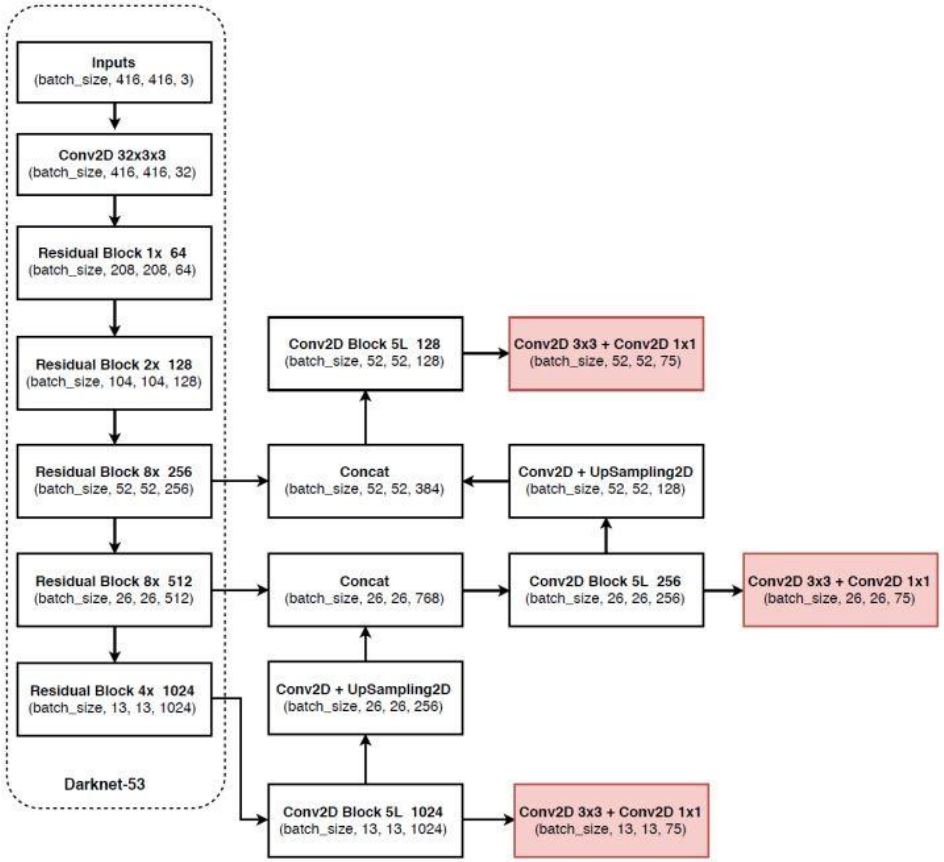
YOLO介绍 | YOLOv3

YOLO 作者在2018年提出了YOLOv3。

特征提取器：Darknet-53。

采用**FPN架构**来实现多尺度检测：输入416×416时：(13×13),(26×26),(52×52)

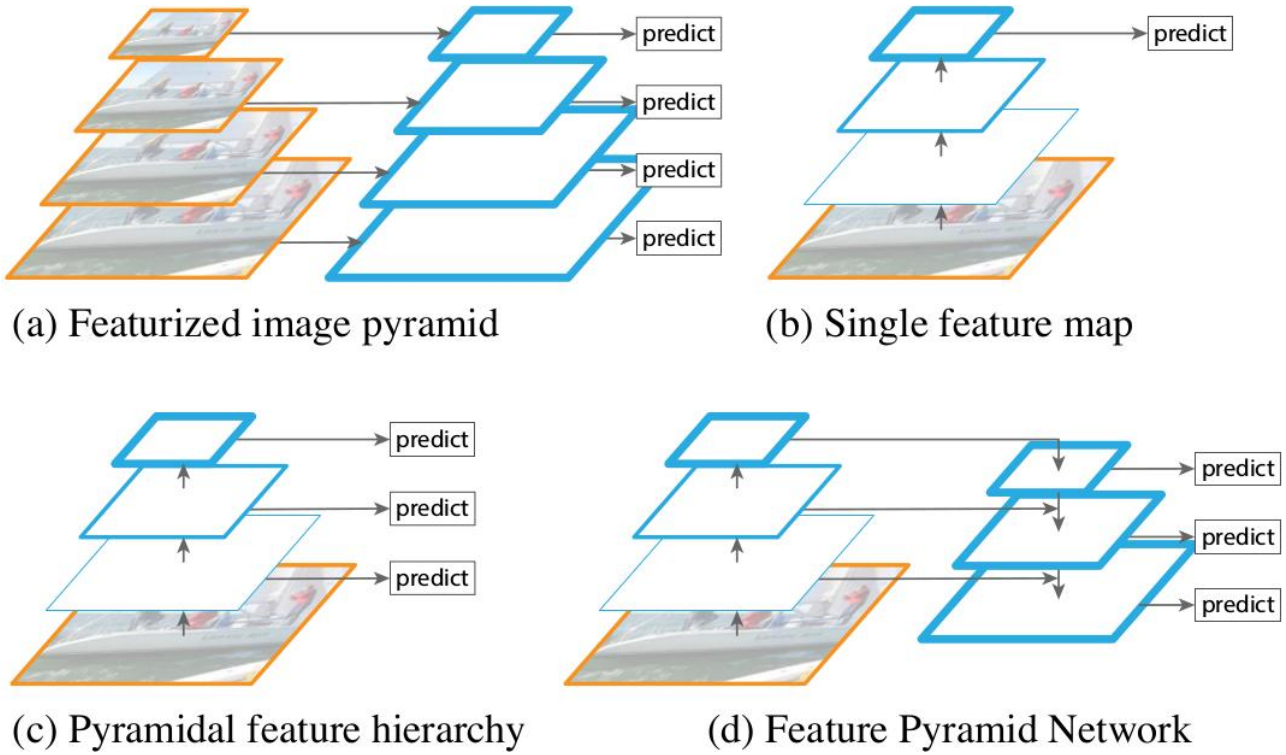
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 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			



YOLO 作者在2018年提出了YOLOv3。

特征提取器：Darknet-53。

采用**FPN架构**来实现多尺度检测：输入416×416时：(13×13),(26×26),(52×52)

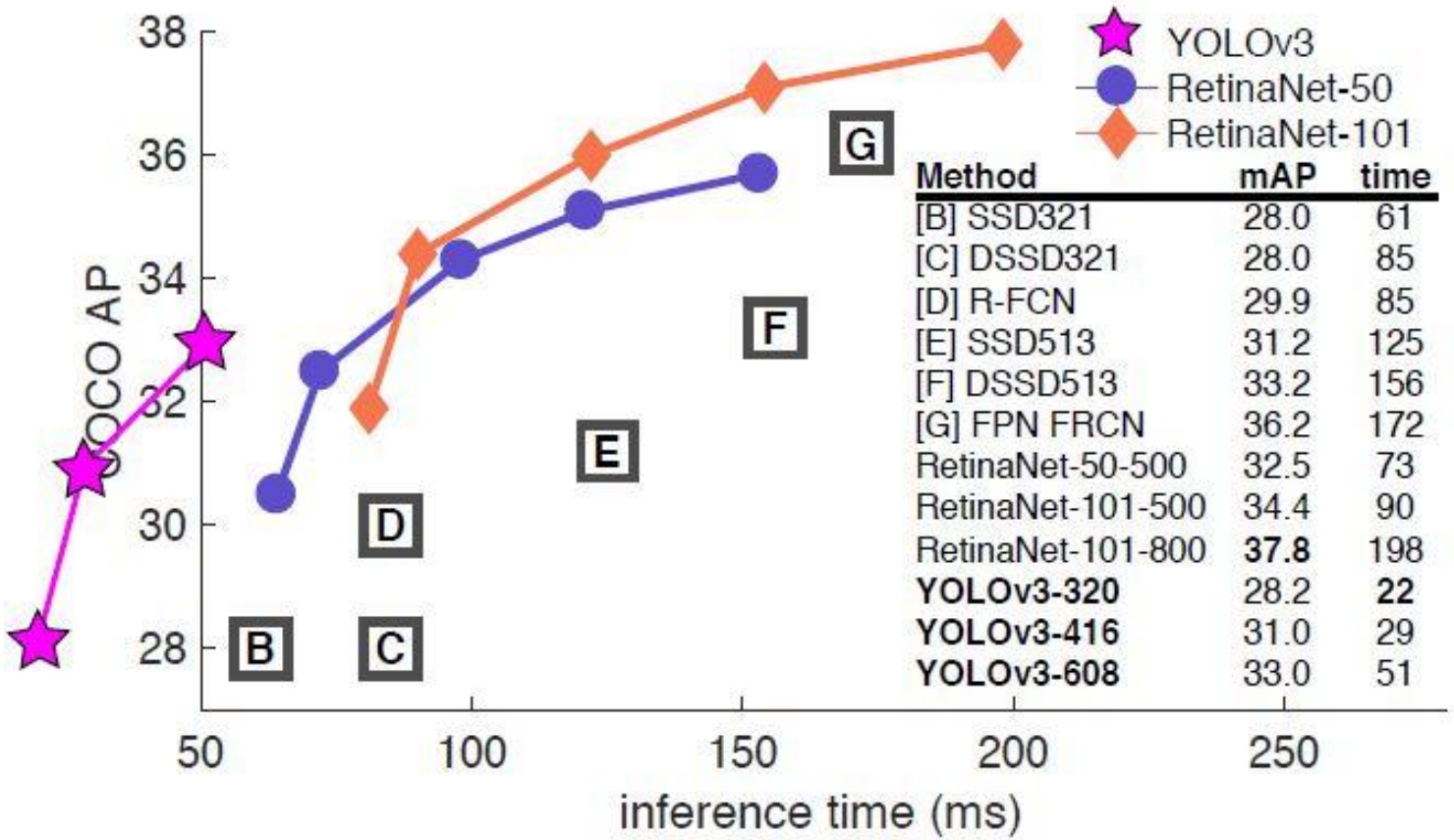


(a) Featurized image pyramid

(b) Single feature map

(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

YOLOv3在COCO测试集与其它检测算法对比图

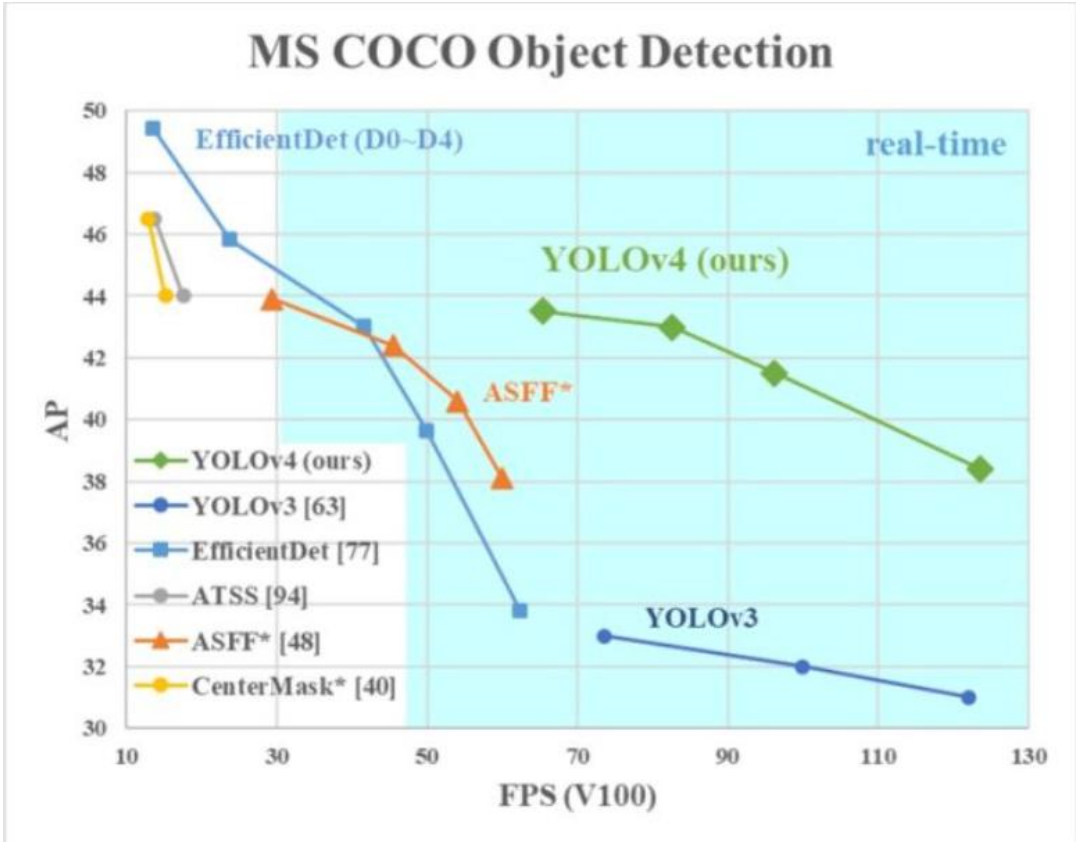
YOLO介绍 | YOLOv4

2020年4月23日，Alexey Bochkovskiy 提出了YOLOv4。

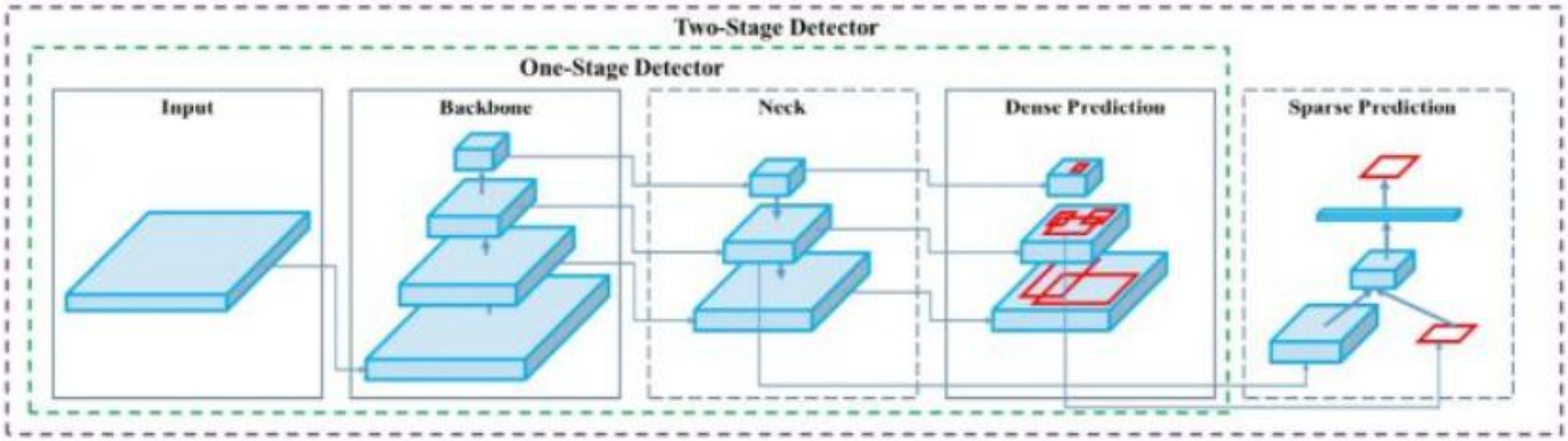
特点：1. 研究设计了一个简单且高效的目标检测算法，降低了训练门槛，可以在一块1080Ti或者2080Ti上训练超快速和准确的目标检测器。

2. 在训练过程中，验证了最新的Bag-of-Freebies和Bag-of-Specials对Yolov4的影响。

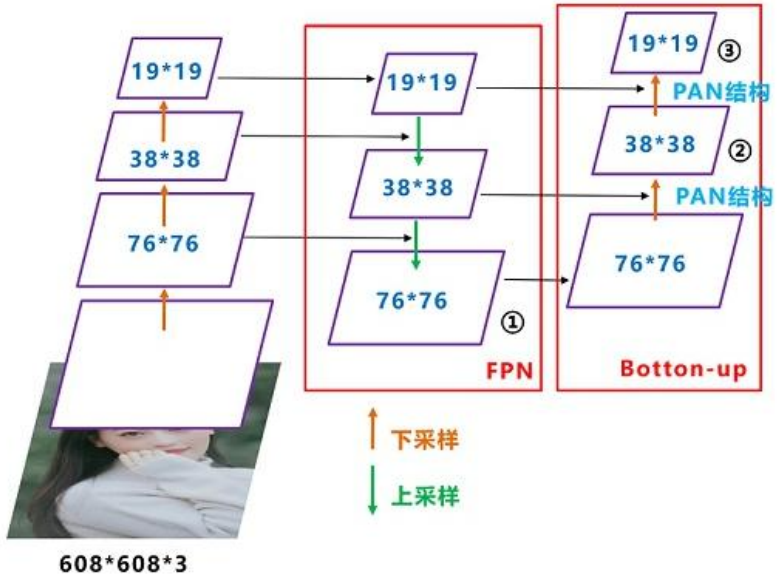
3. 简化以及优化了一些最新提出的算法，从而使Yolo-V4能够在一块GPU上就可以训练起来。



YOLO介绍 | YOLOv4



Backbone：CSPDarknet53
Neck：SPP，PAN
Head：YOLOv3

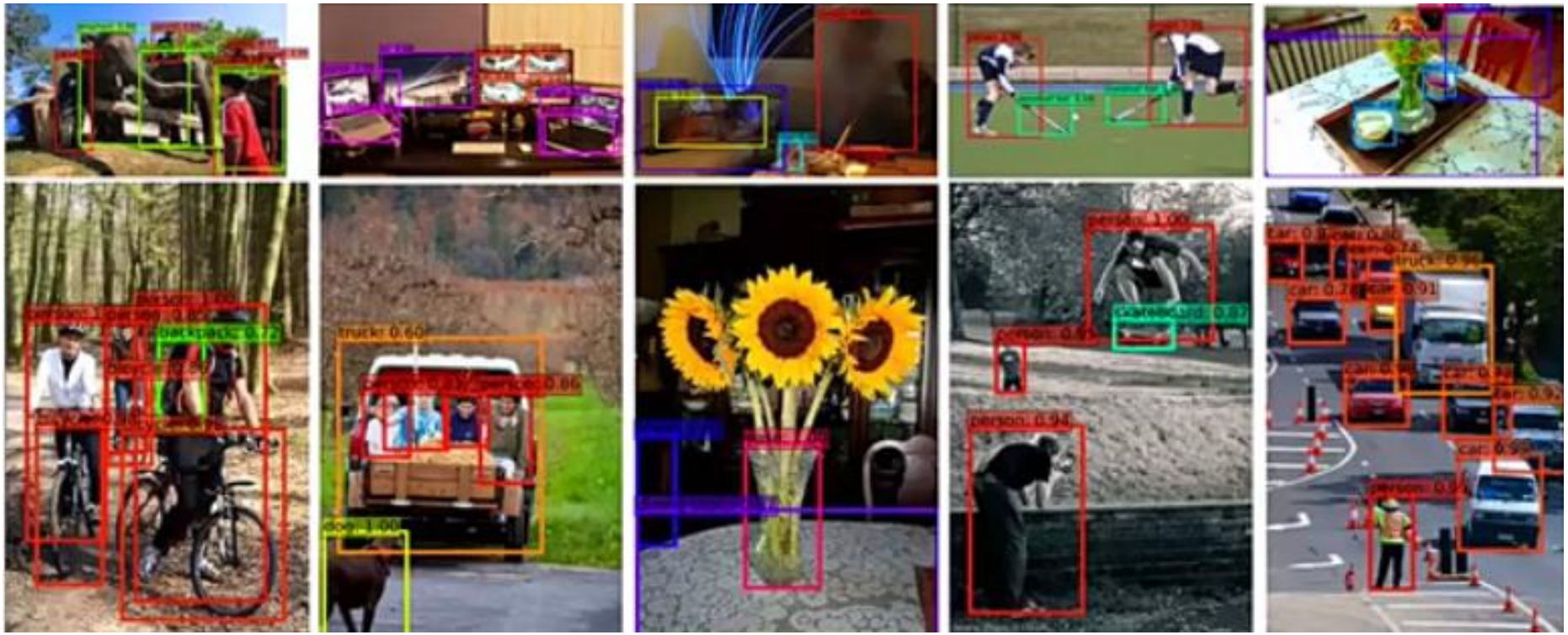


SSD算法

SSD：Single Shot MultiBox Detector

SSD网络是作者Wei Liu在ECCV2016上发表的论文。

对于输入尺寸300x300的网络使用Nvdia Titan X在VOC 2007测试集上达到了74.3%mAP超越了当时最强的Faster RCNN（73.2%mAP）。



SSD算法是Faster-RCNN和YOLO的结合：

1.采用基于回归的模式（物体的类别和位置）

2.利用基于区域的概念（多候选区域最为ROI）

SSD重要概念:

Single Shot Detection:

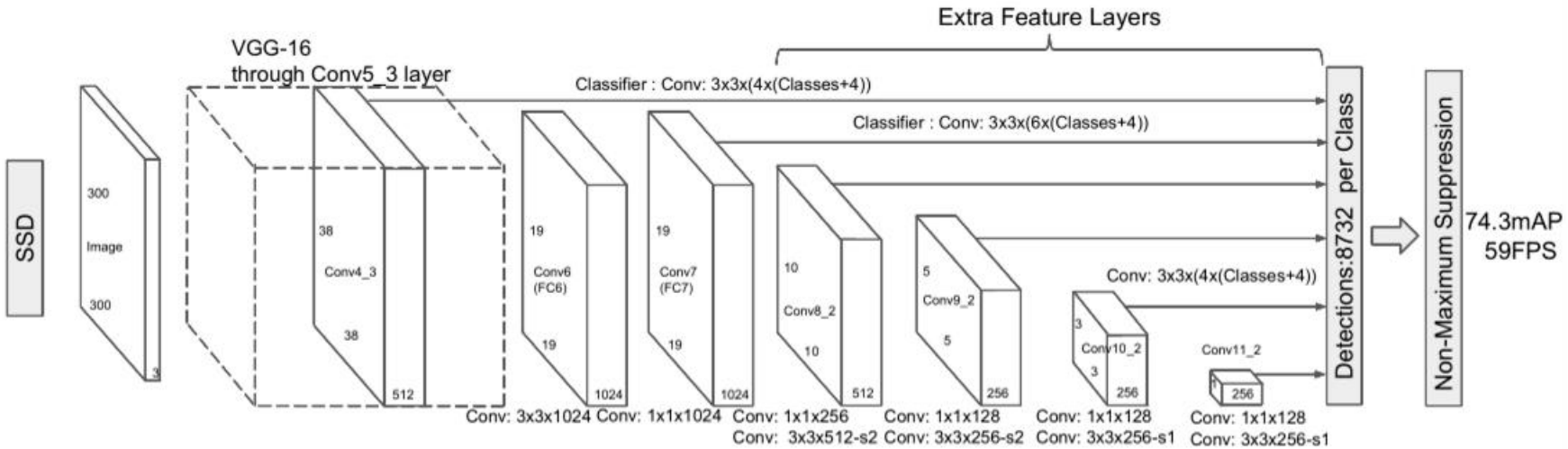
多尺度特征映射图（Multiscale Feature Maps）

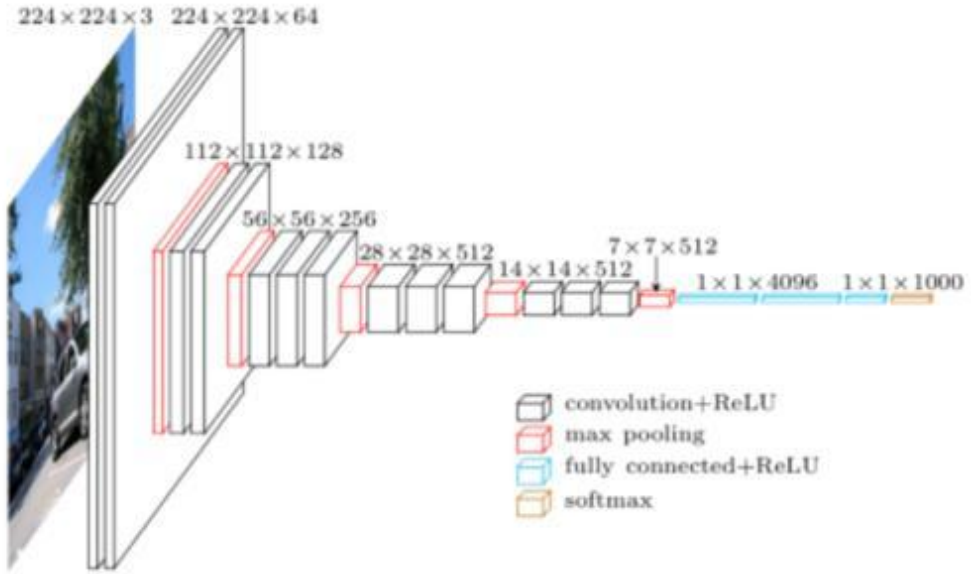
先验框（Priors）

预测矩形框

非极大值抑制（Non-maximum Suppression）

在不同特征尺度上预测不同尺度的照片



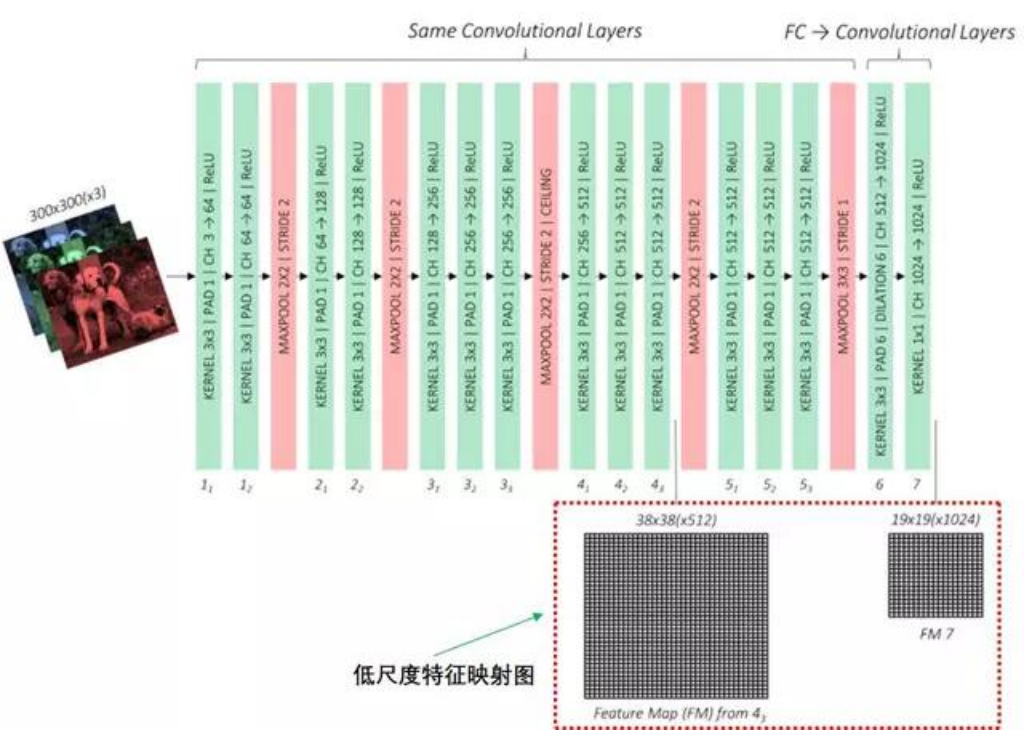


SSD网络包含了基础网络，辅助卷积层和预测卷积层：

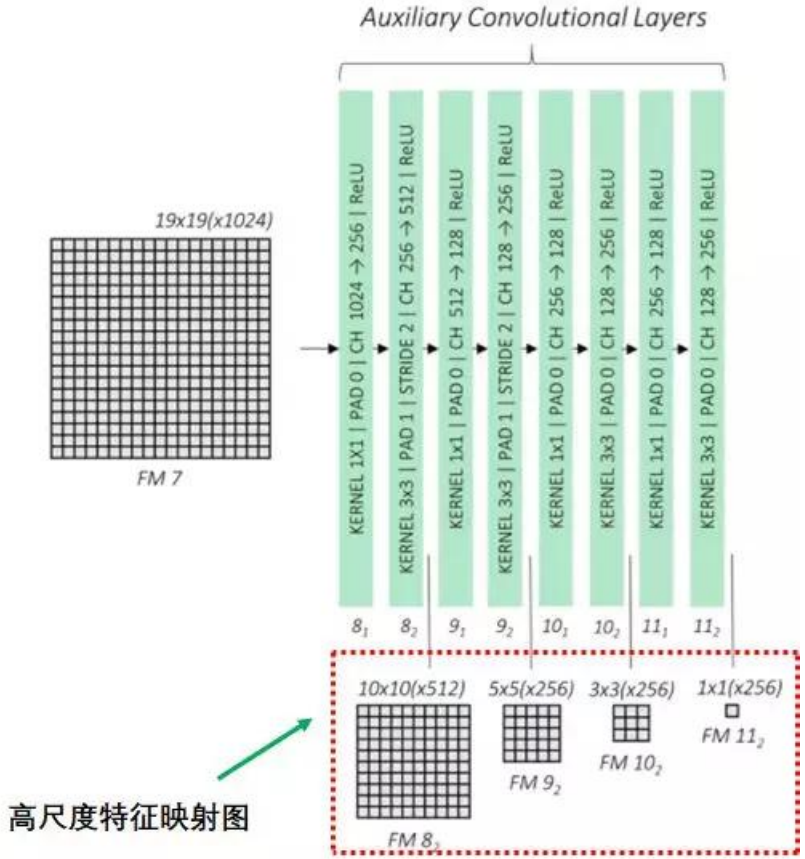
基础网络：提取低尺度的特征映射图

辅助卷积层：提取高尺度的特征映射图

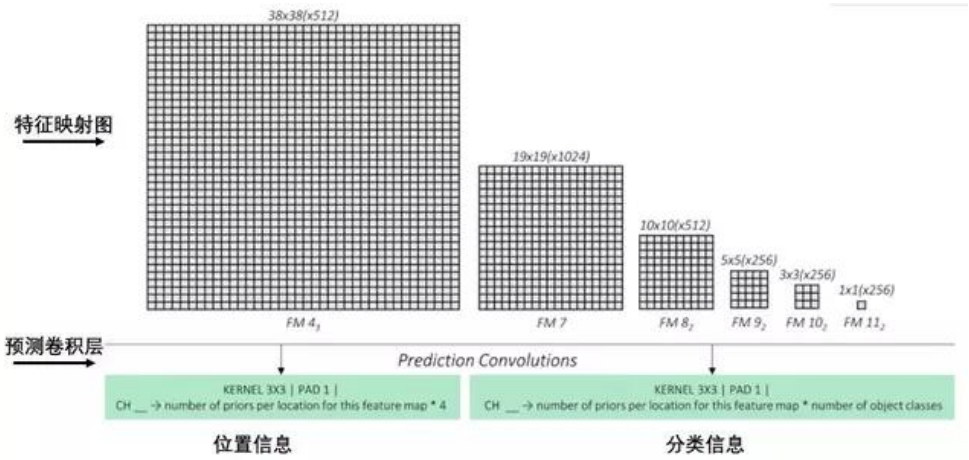
预测卷积层：输出特征映射图的位置信息和分类信息



基于VCG网络架构的基础网络

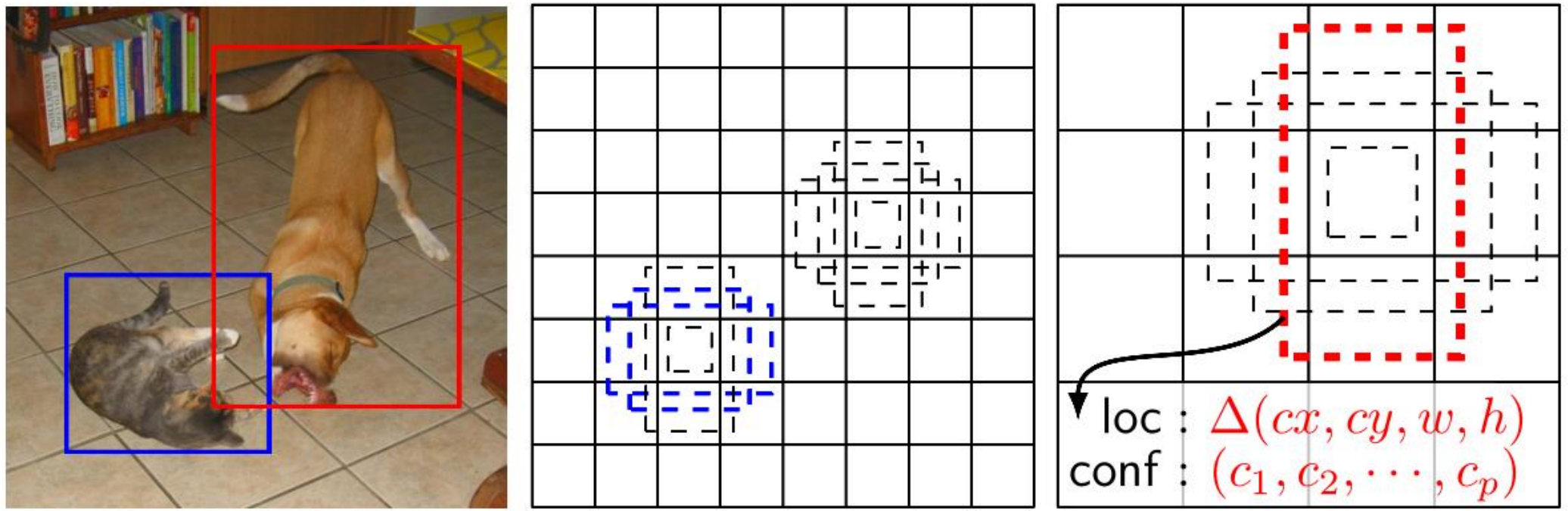


辅助卷积层连接基础网络最后的特征映射图，通过卷积神经网络输出4个高尺度的特征映射图



预测卷积层预测特征映射图每个点的矩形框信息和所属类信息

SSD借鉴了Faster R-CNN中anchors box的原理，每个单元设置尺度或者长宽比不同的先验框，预测的边界框都是以先验框为基准，在一定程度上减少了训练难度。一般情况下，每个单元会设置多个先验框，其尺度和长宽比存在差异。

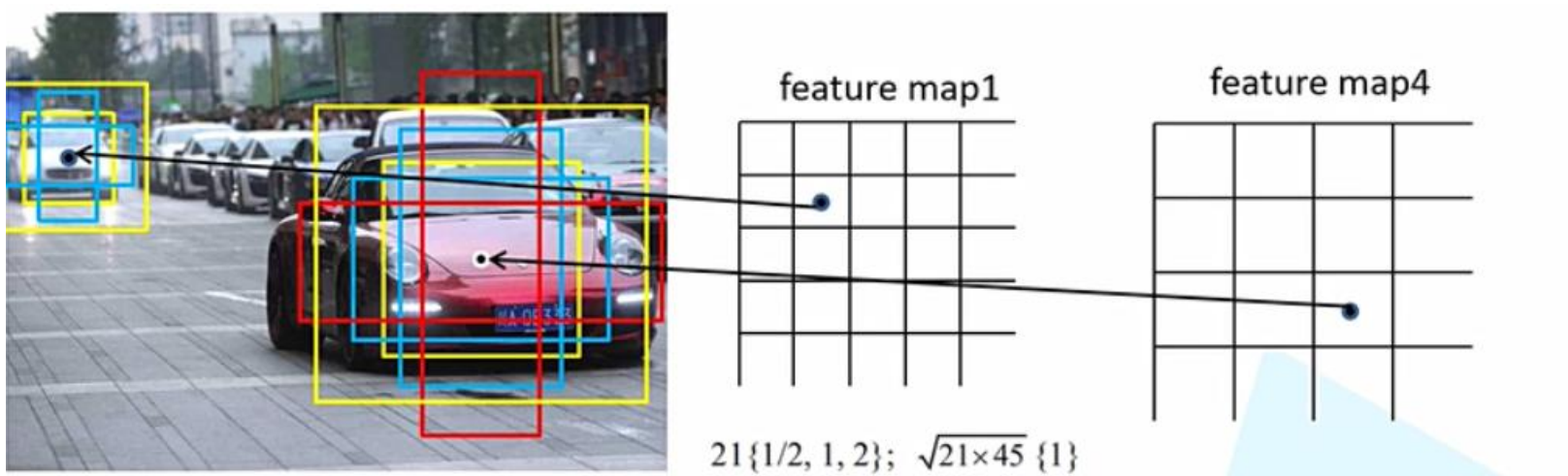


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Default Box的尺度及比例设定

特征图层	特征图层的宽和高	默认框尺寸	默认框数量
特征图层①	38×38	21{1/2, 1, 2}; $\sqrt{21\times45}$ {1}	38×38×4
特征图层②	19×19	45{1/3, 1/2, 1, 2, 3}; $\sqrt{45\times99}$ {1}	19×19×6
特征图层③	10×10	99{1/3, 1/2, 1, 2, 3}; $\sqrt{99\times153}$ {1}	10×10×6
特征图层④	5×5	153{1/3, 1/2, 1, 2, 3}; $\sqrt{153\times207}$ {1}	5×5×6
特征图层⑤	3×3	207{1/2, 1, 2}; $\sqrt{207\times261}$ {1}	3×3×4
特征图层⑥	1×1	261{1/2, 1, 2}; $\sqrt{261\times315}$ {1}	1×1×4

38x38x4+19x19x6+10x10x6+5x5x6+3x3x4+1x1+4=8732



SSD的基本步骤：

1.输入一幅图片，让图片经过卷积神经网络（CNN）提取特征，并生成 feature map。

2.抽取其中六层的feature map，然后再 feature map 的每个点上生成 default box（各层的个数不同，但每个点都有）。

3.将生成的所有 default box 都集合起来，全部丢到 NMS（非极大值抑制）中，输出筛选后的 default box，并输出。

注：文中提到，由于有8000多个default box 那么matching之后会有大量的负样本，为了数据集更加干净，正负样本比例取到 1:3。

性能评估

PASCAL VOC 2007

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

PASCAL VOC 2012

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast[6]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster[2]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Faster[2]	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO[5]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	**88.4**	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

模型分析

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Table 2: **Effects of various design choices and components on SSD performance.**

Prediction source layers from:						mAP use boundary boxes?		# Boxes
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No	
✔	✔	✔	✔	✔	✔	74.3	63.4	8732
✔	✔	✔	✔	✔		**74.6**	63.1	8764
✔	✔	✔	✔			73.8	68.4	8942
✔	✔	✔				70.7	69.2	9864
✔	✔					64.2	64.4	9025
	✔					62.4	64.0	8664

Table 3: **Effects of using multiple output layers.**

Table 1. Comparison of object detection methods on the basis of inference time and detection procedures.

Methods	Detection Procedures	Frameworks	inference time	Language	Use case
R-CNN [1]	Selective search	Caffe	47 s	MATLAB	Not used for real-time application
Fast R-CNN [1]	RoI projection	Caffe	2.3 s	Python	Faster than R-CNN
Faster R-CNN [1]	RPN	Caffe	0.2 s	Python/ MATLAB	real-time object detection
Mask R-CNN [2]	RPN	TensorFlow/ Keras	0.35 - 2 s	Python	real-time object detection
SPP-net [3]	Edge Boxes	Caffe	0.4 s	MATLAB	real-time object detection
R-FCN [7]	RPN	Caffe	0.1 s	MATLAB	real-time object detection
FPN [6]	RPN	TensorFlow	0.2 s	Python	real-time object detection
SSD [4]	-	Caffe	0.15 s	C++	real-time object detection
YOLOv3 [5]	-	Darknet	0.05 s	C	real-time object detection

Code Examples

```
%% 创建 FasterRCNN 网络并用于目标检测
openExample('vision/CreateFasterRCNNObjectDetectionNetworkExample')
openExample('deeplearning_shared/DeepLearningFasterRCNNObjectDetectionExample')

%% 使用 Yolov3 用于目标检测
openExample('deeplearning_shared/ObjectDetectionUsingYOLOV3DeepLearningExample')

%% 使用 SSD 用于目标检测
openExample('deeplearning_shared/ObjectDetectionUsingSSDDeepLearningExample')
```