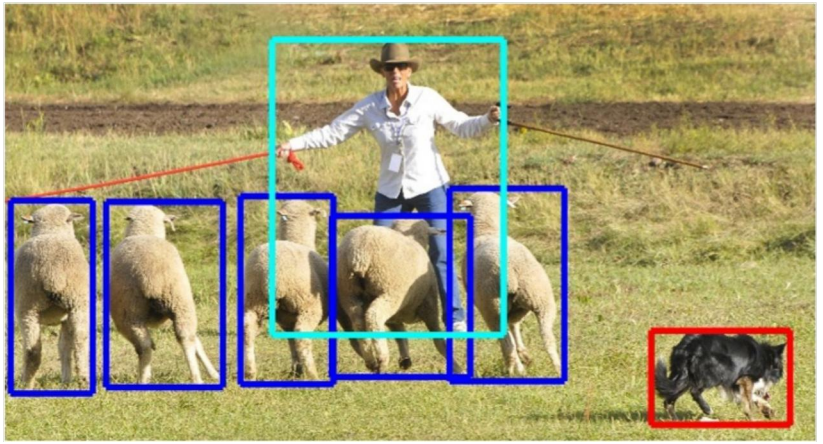
A classic review of target detection

A classic review of target detection

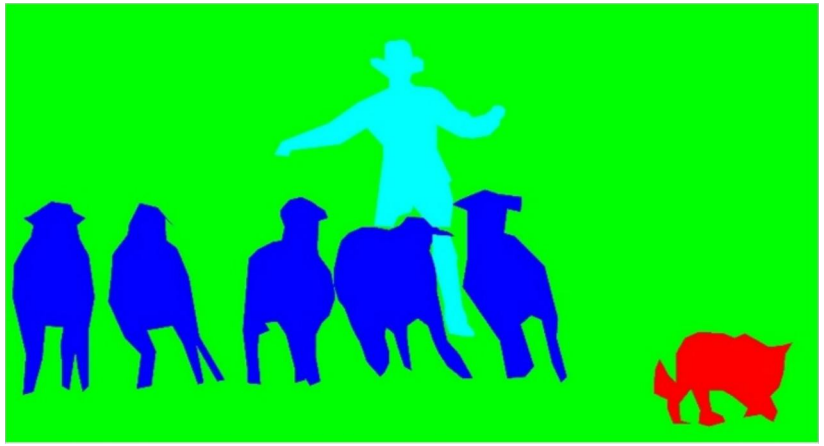
- introduce
- challenge
- develop/compare
- inspiration

Task definition

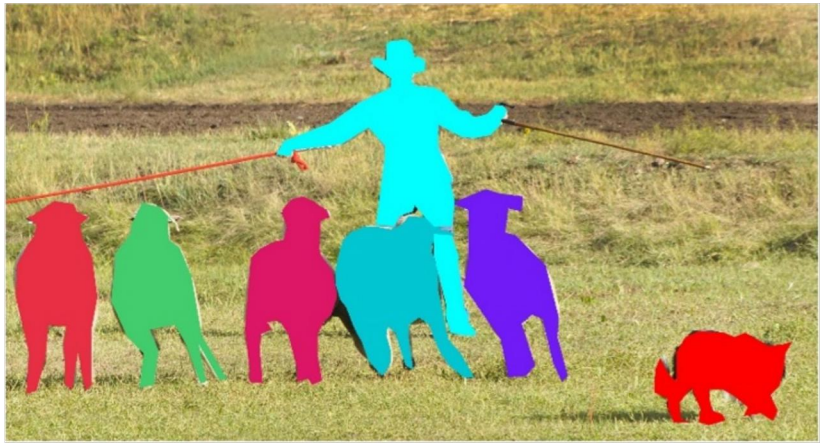
Image classification



Object detection

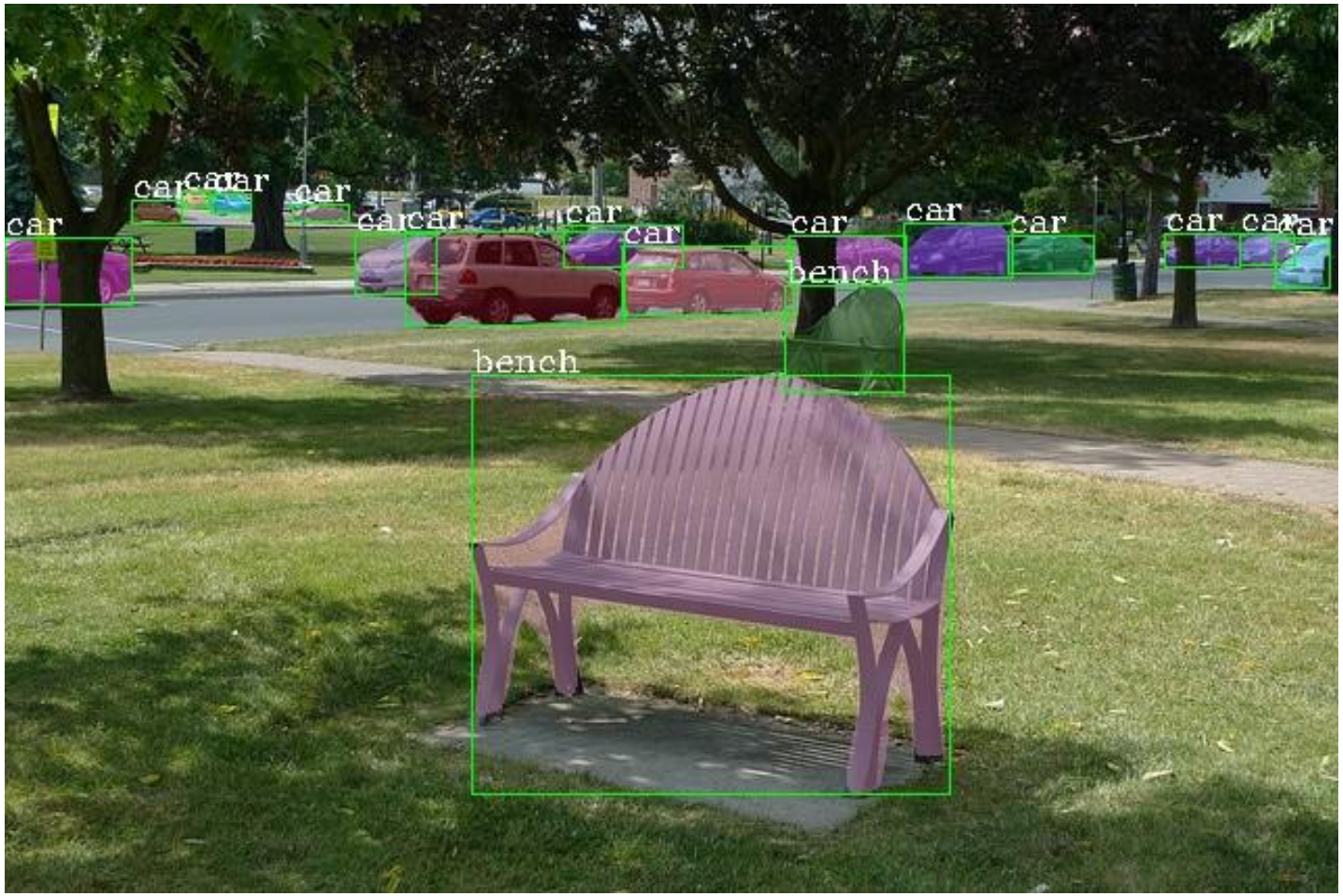


Semantic segmentation



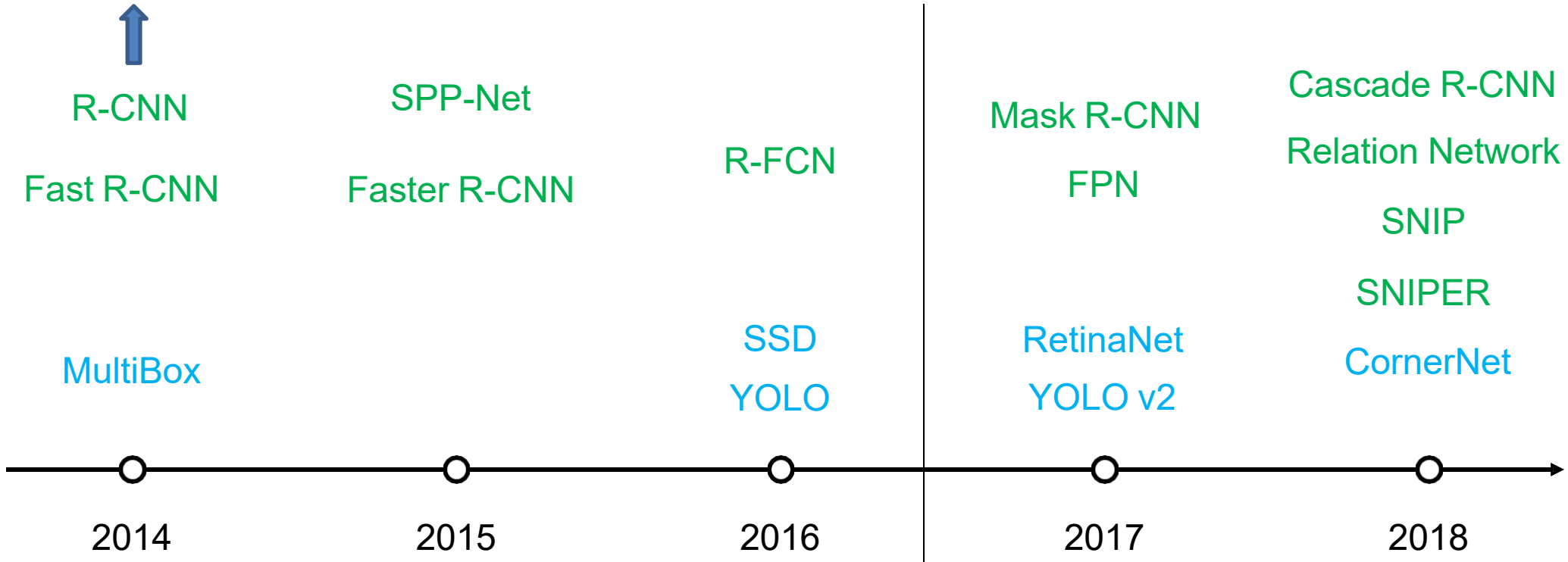
Instance segmentation

Task definition



Progress

- 将物体检测**首次**应用于深度学习中的一篇论文，引用量非常高，影响也非常广泛



- 基于深度学习目标检测的流程变得越来越**精简**
- 精度越来越**高**
- 速度也越来越**快**

recent

Challenges

1.小目标在原图中尺寸比较小。

通用目标检测模型中，一般的基础骨干神经网络（VGG系列和Resnet系列）都有几次下采样处理，如果分类和回归操作在经过几层下采样处理的特征层进行，**小目标特征的感受野映射回原图将可能大于小目标在原图的尺寸**，造成检测效果差。

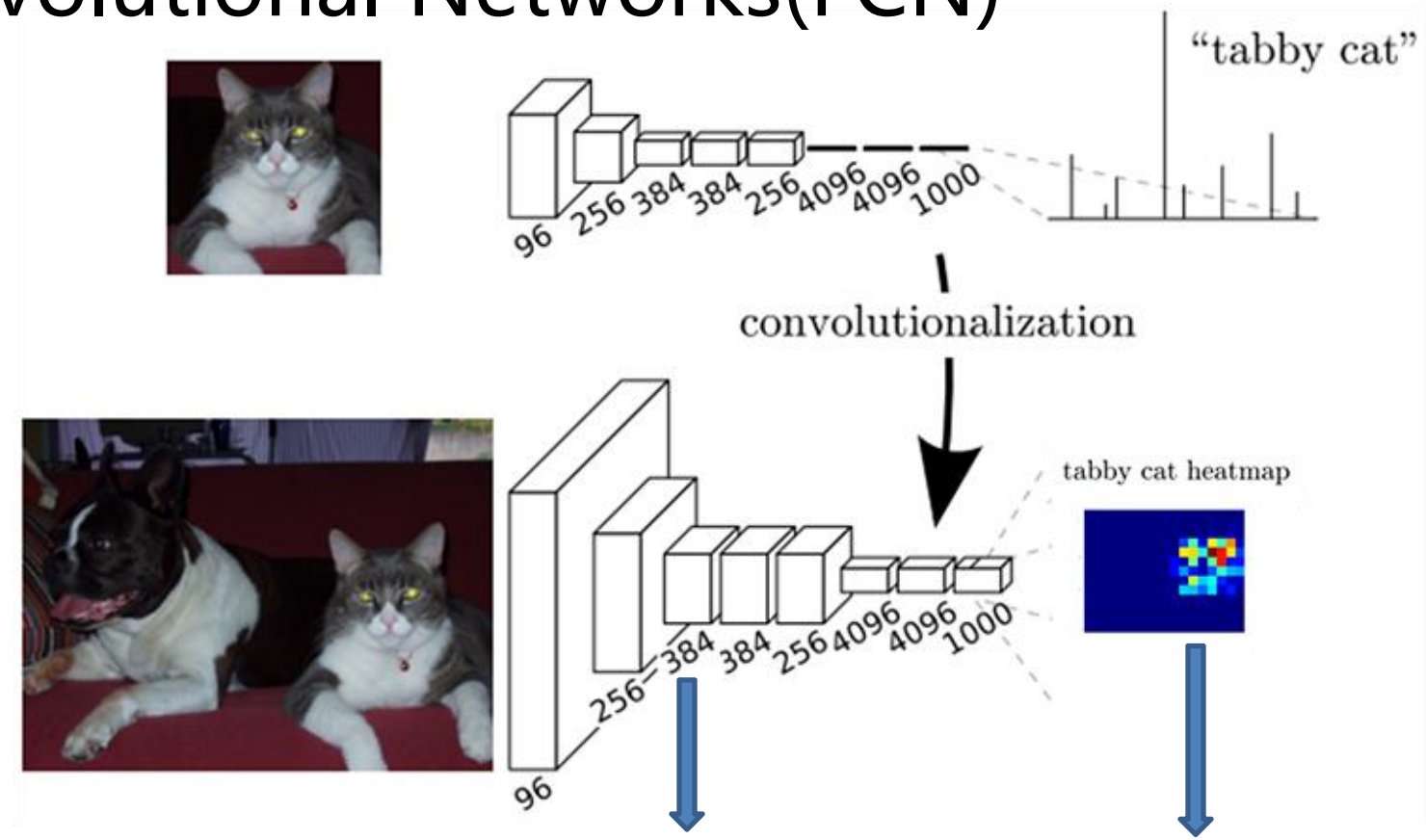
2.小目标在原图中的数量较少

小目标在原图中的数量较少，**检测器提取的特征较少**，导致小目标的检测效果差。

3.神经网络在学习中被大目标主导

大目标容易被检测，**学习中被大目标主导，小目标在整个学习过程被忽视**，导致导致小目标的检测效果差。

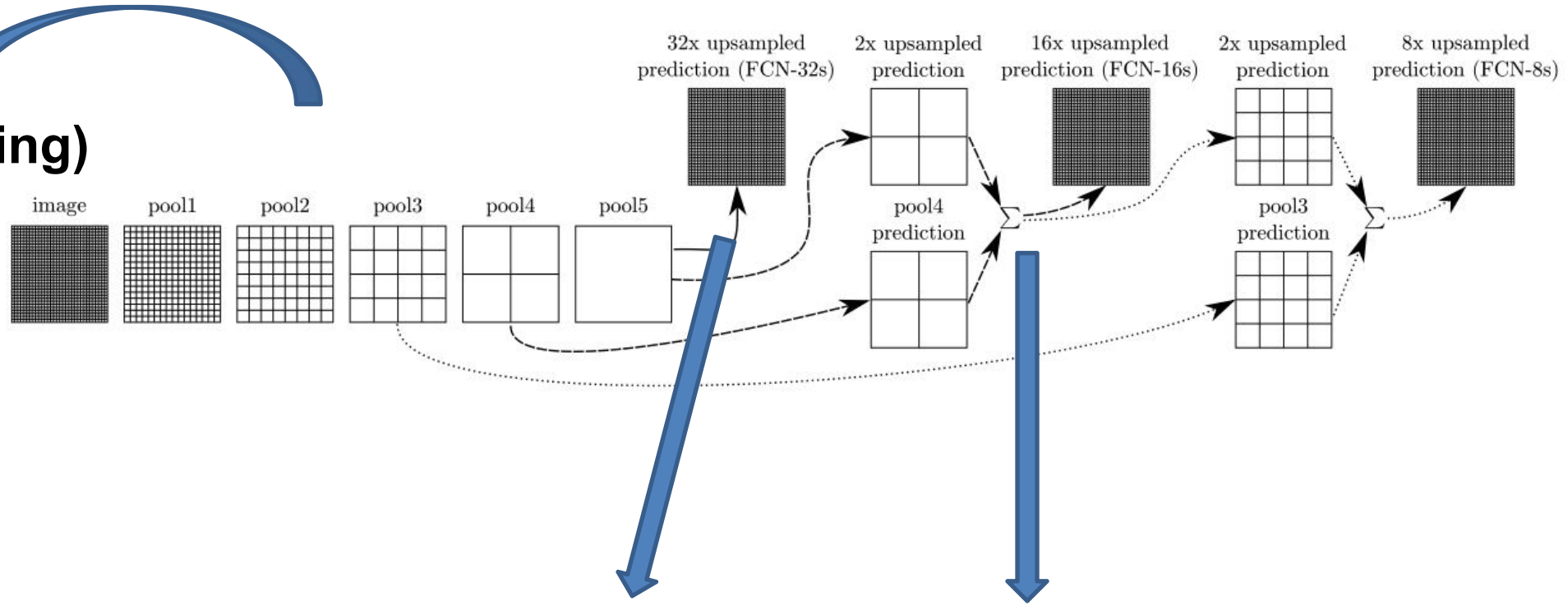
Fully Convolutional Networks(FCN)



1.**Fully connection layers-> conv lays**， 2. Get **heatmap of each pixel's category**
become fully conv network

Fully Convolutional Networks(FCN) framework

1.**5 conv(and pooling)**



2.**Deconvolution** to original size

3.**Sum and deconvolution**

在反卷积时，采用一部分**较浅层的反卷积信息辅助叠加**，更好的优化分割结果的精度

CNN vs FCN

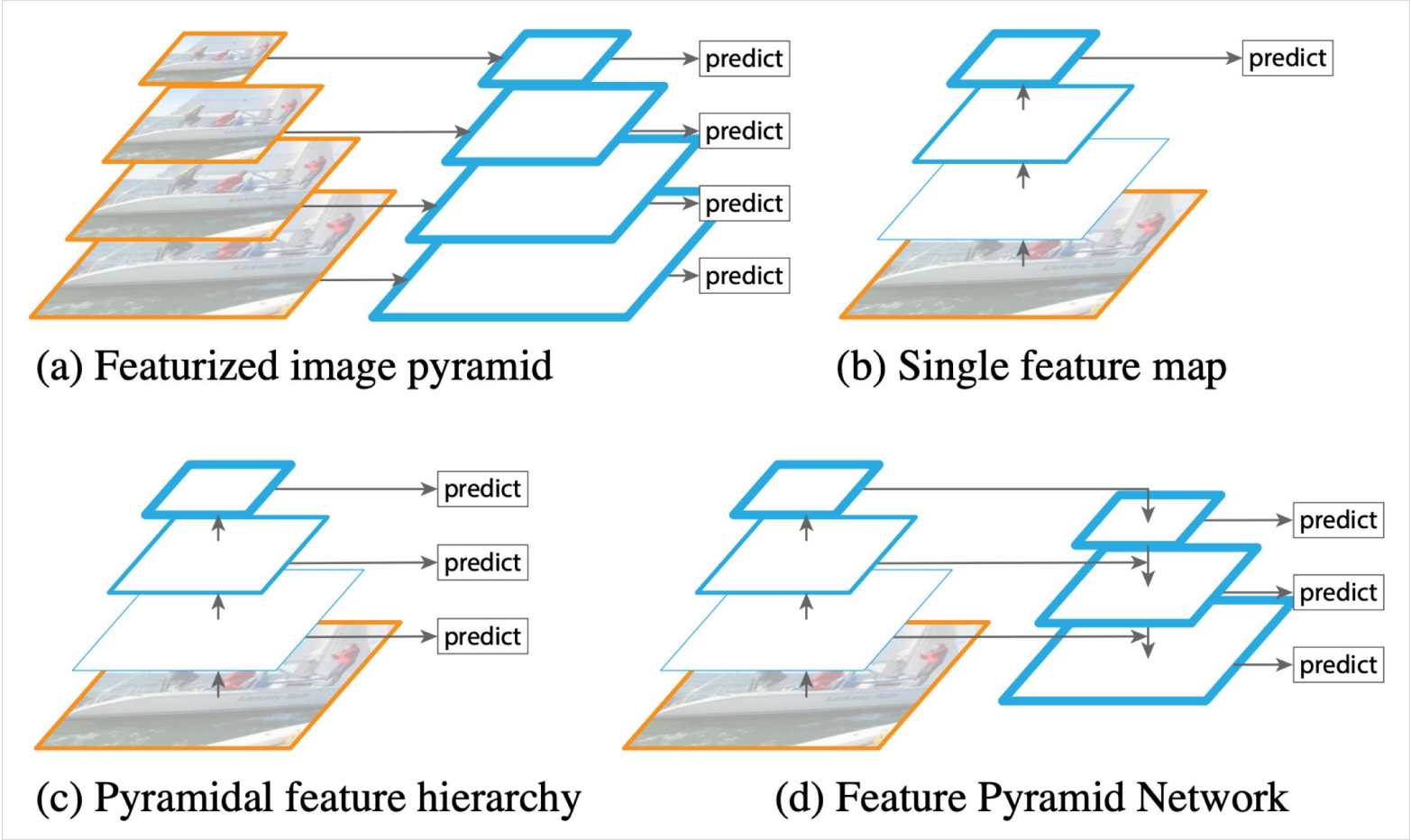
- CNN可以比较好地判断出一幅图像中包含什么类别的物体，但是因为丢失了一些物体的细节，**不能很好地给出物体的具体轮廓、指出每个像素具体属于哪个物体**，因此做到精确的分割就很有难度。

- FCN优点：1.**可以接受任意大小的输入图像**。而不用要求所有的训练图像和测试图像具有同样的尺寸。2.**更加高效**。因为避免了由于使用像素块而带来的重复存储和计算卷积的问题。

- FCN不足：1.得到的**结果还是不够精细**。上采样的结果还是比较模糊和平滑，对图像中的细节不敏感。2，**缺乏空间一致性。**对各个像素进行分类，没有充分考虑像素与像素之间的关系。

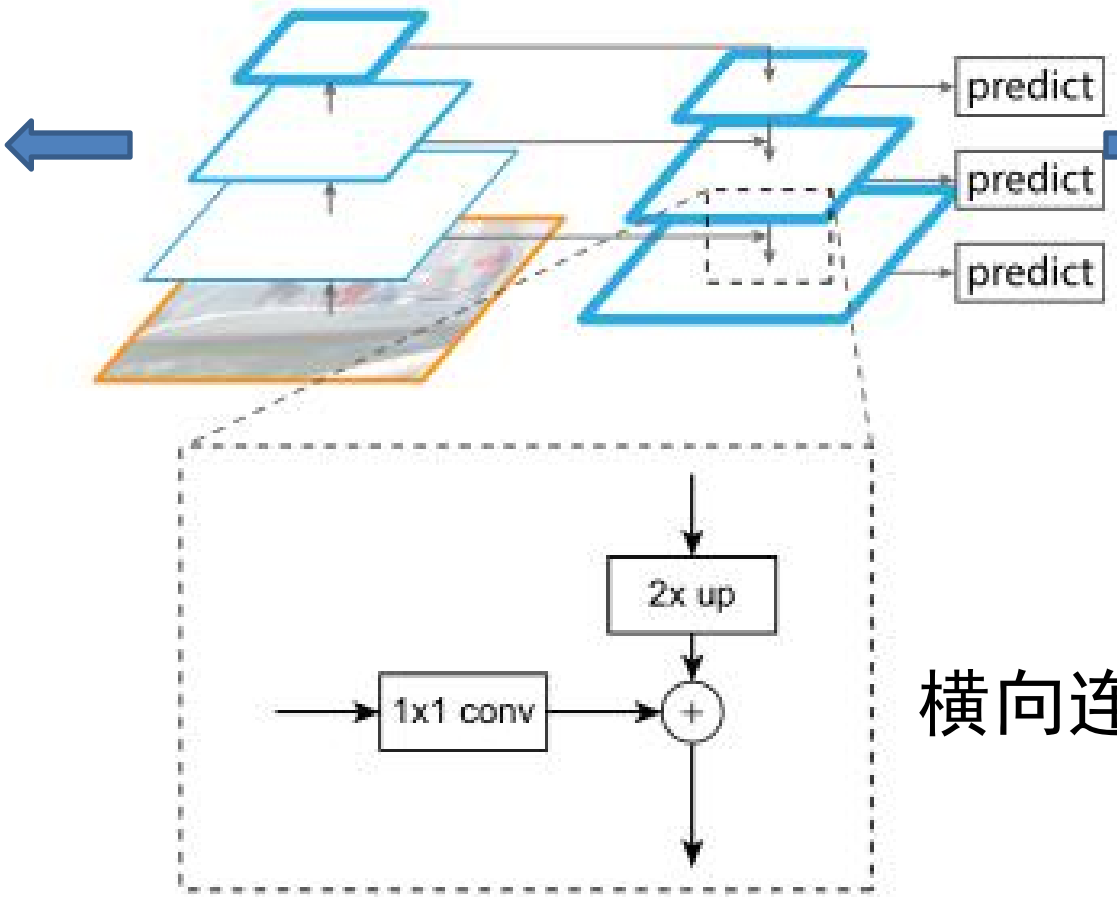
Feature Pyramid Network (FPN)

a：图像金字塔
b：单特征图预测
c：自下而上金字塔
d：特征金字塔(FPN)



FPN framework

由上到下



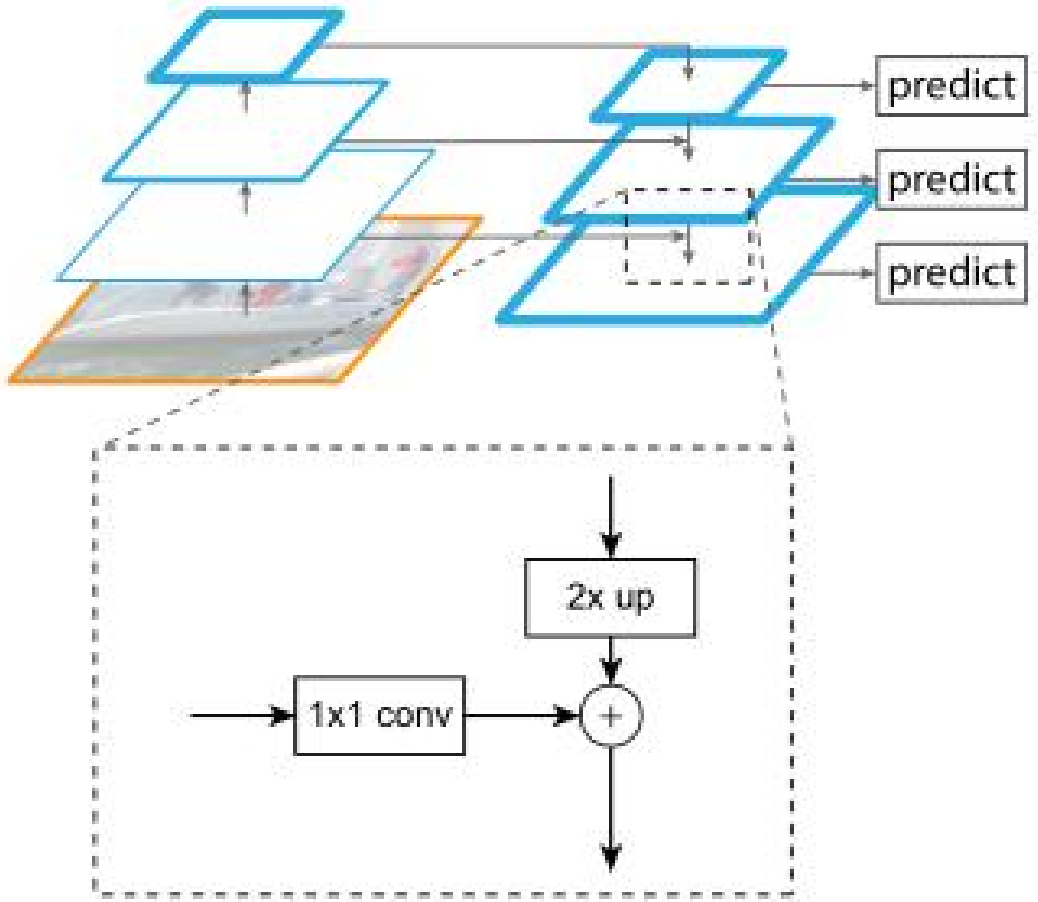
由下到上

Multi-level prediction

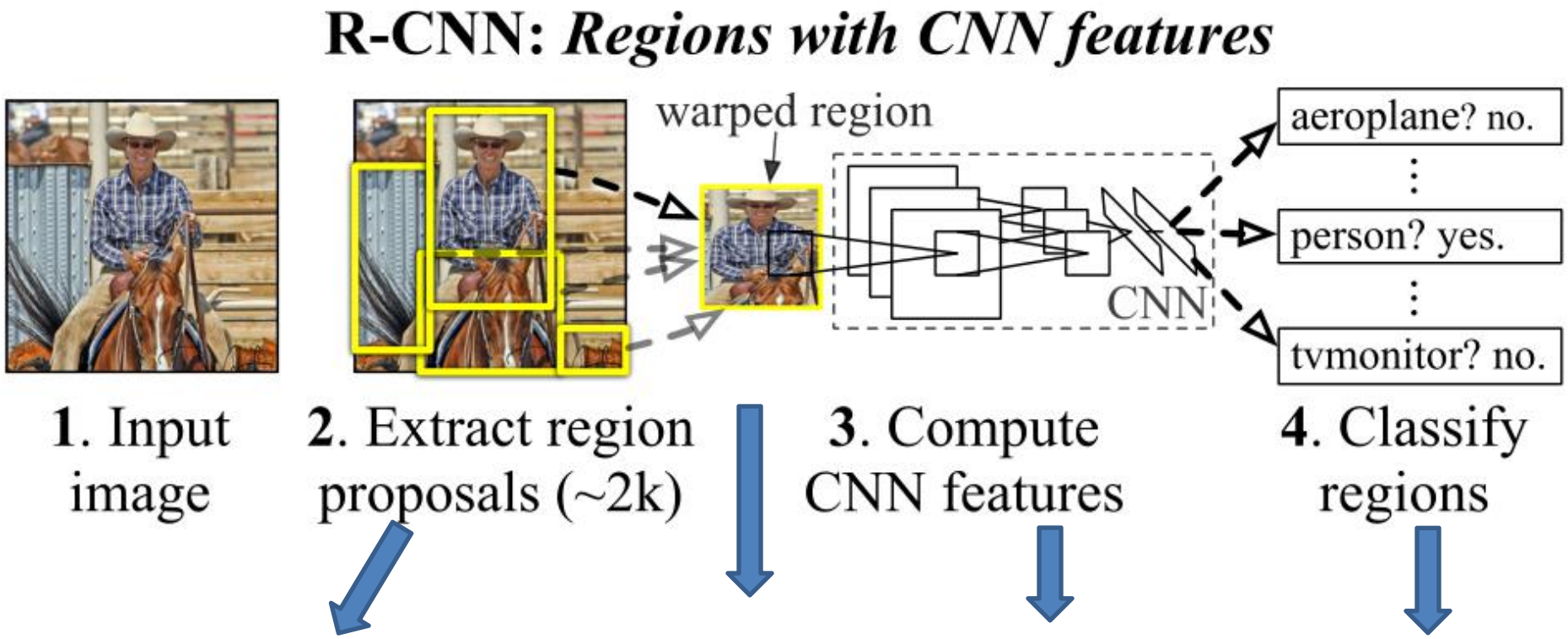
横向连接

FPN contribution

- **多尺度特征融合**
- **结合深层网络的语义信息和浅层网络的细节，提高了准确率。**



R-CNN framework



R-CNN

基本步骤

1.候选区域生成： 一张图像生成1K~2K个候选区域 （采用Selective Search 方法）

2.特征提取： 对每个候选区域，使用深度卷积网络提取特征 （CNN）

3.类别判断： 特征送入每一类的SVM 分类器，判别是否属于该类

4.位置精修： 使用回归器精细修正候选框位置

R-CNN的不足

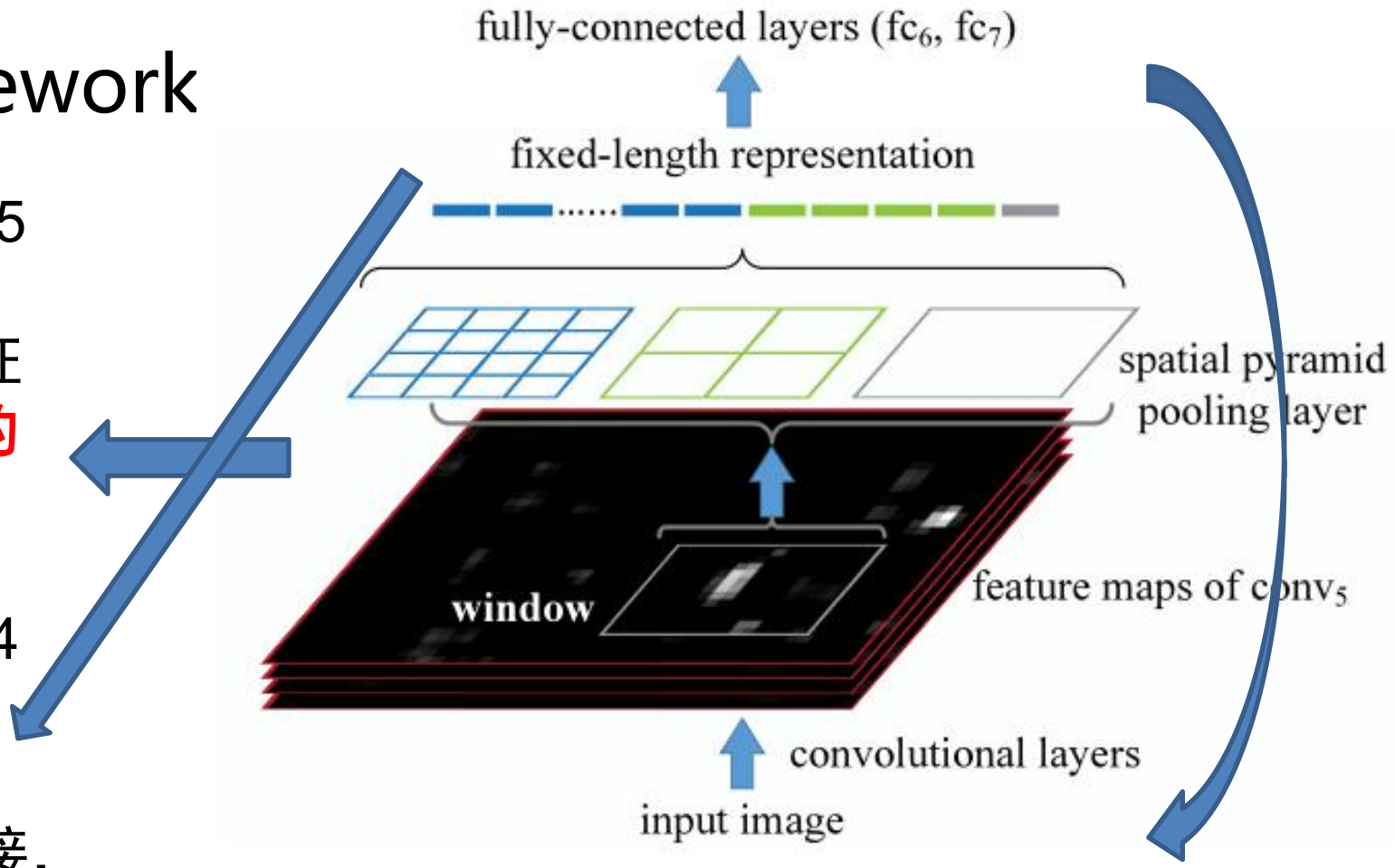
- R-CNN：速度**太慢**。 47s/image,生成2000个Proposal，将**每个Proposal当成一张图像进行后续处理**(CNN提特征+SVM分类)。

SPPNet-> Motivation/Challenge

- Motivation：完全可以对整张图像提取一次卷积层特征，然后只需要**将Region Proposal在原图的位置映射到卷积层特征图**。
- Challenge：**每个Region Proposal的尺度不一样**，而全连接层是使用图像的**全局**信息，要输入必须是**固定**的长度。

SPPNet framework

1. **SPP池化**：经过conv5得到特征图，使用一个224*224的窗口在此特征图上滑动，得到**256维的13x13**的feature map，对其**spatial pyramid pooling**得到1*1,2*2,4*4网格大小的三张子图。



2.对三张子图进行全连接，得到**固定长度**的池化特征向量。

3. fc6的结果用于**分类**，fc7的结果用于**边框回归**。

SPPNet->contribution

- **不管**输入的图片是什么尺度，都能够正确的传入网络。
- **对整张图片进行一次特征提取**，**大大**加快了速度。

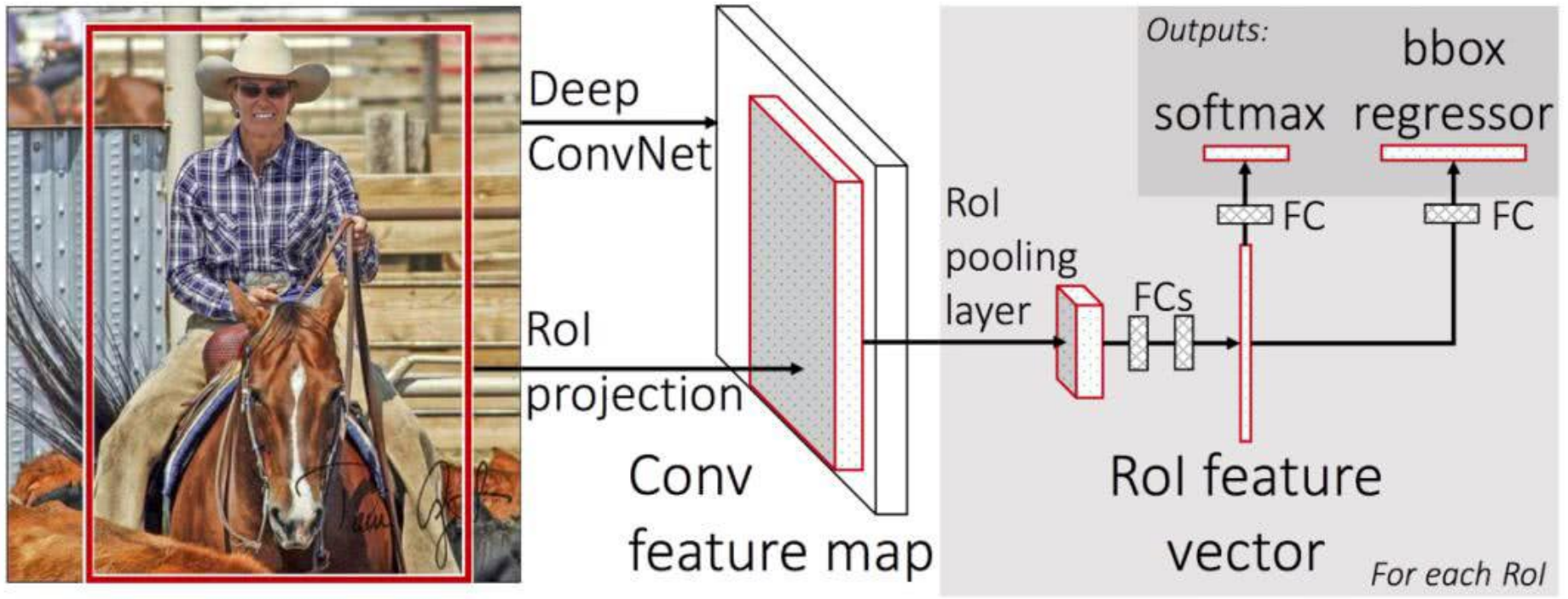
SPP-Net

基本步骤:
1.候选区域生成：使用Selective Search 获得获选区域
2.特征提取： 对每个候选区域，使用多尺寸识别网络提取特征
3.类别判断： 特征送入每一类的SVM 分类器，判别是否属于该类
4.位置精修： 使用回归器精细修正候选框位置

Fast R-CNN framework

1.**Selective search**
->proposal

2.Take an **entire** image into CNN



3.Proposal **mapping** to feature map

4.**RoI pooling**，生成固定尺寸的feature map

5.Softmax Loss和Smooth Loss**联合训练**

Fast R-CNN

基本步骤:
1.候选区域生成：在图像中确定约1k-2k个候选框 (使用选择性搜索Selective Search)
2.特征提取：对整张图片输进CNN，得到feature map
3.抽取特征框：找到每个候选框在feature map上的映射patch，将此patch作为每个候选框的卷积特征输入到SPP layer和之后的层
4.类别判断：对候选框中提取出的特征，使用Softmax分类器判别是否属于一个特定类
5.位置精修：对于属于某一类别的候选框，用回归器进一步调整其位置

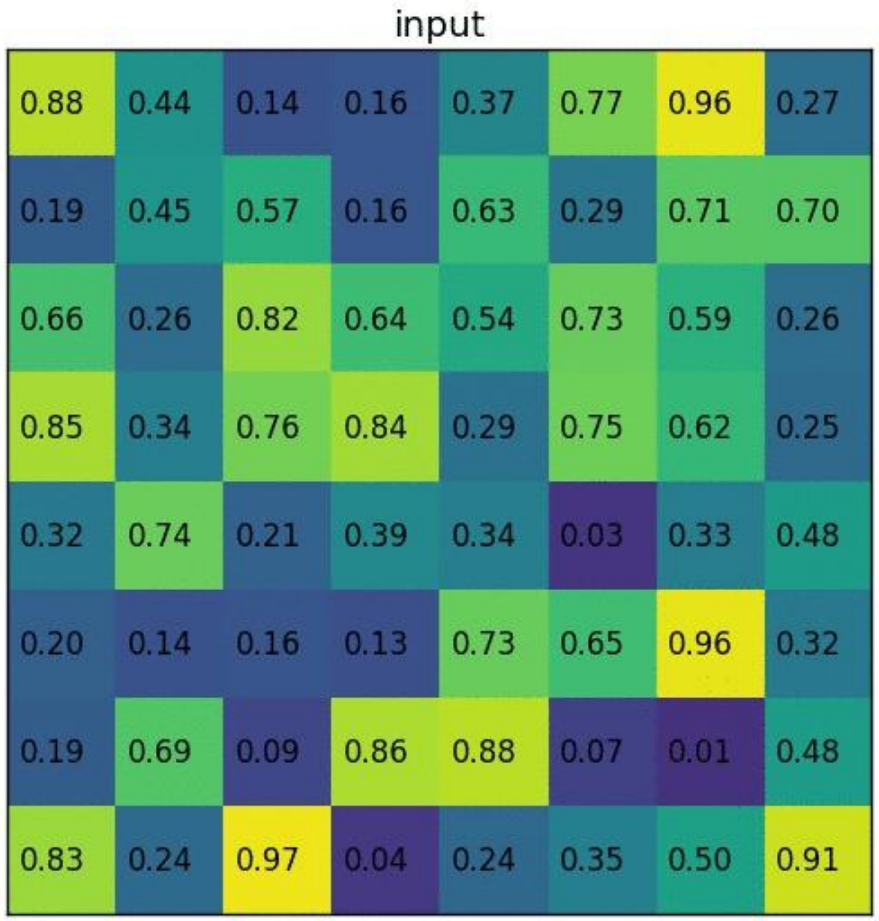
Fast R-CNN->RoI Pooling

1.输入固定大小的feature map。

2. region proposal 投影之后位置（左上角，右下角坐标）。

3.将其**划分为（2x2）**个sections（指定输出的大小为2x2）。

4.对**每个section做max pooling**。



Fast R-CNN contribution

- R-CNN：中**先采用SVM**对特征进行类别判断，**再**进行bounding-box的回归得到位置信息。**整个过程是个串行的**，这极大地影响了网络的检测速度。

- Fast R-CNN：则将，损失函数使用了**多任务损失函数**(multi-task loss)，将边框回归直接加入到CNN网络中训练。实现了特征的共享也进一步提升了速度。

Fast R-CNN RoI Pooling vs SPPNet

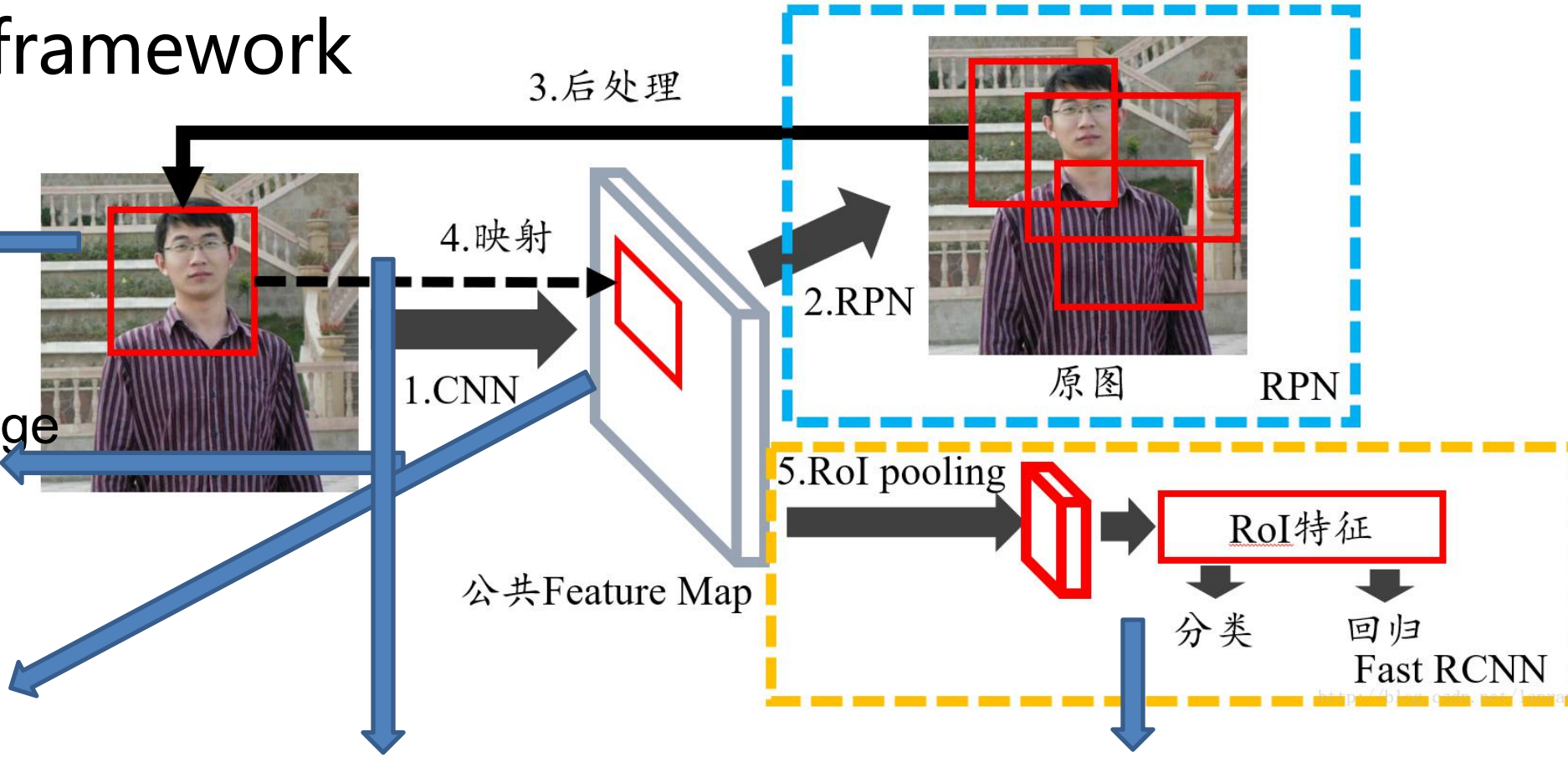
- SPP针对同一个输入使用了**多个不同尺寸**的池化操作，把不同尺度的结果拼接作为输出。
- RoI Pooling可看作**单尺度**的SPP，对于一个输入只进行一次池化操作。

Faster -RCNN framework

1. Input the image

2. Take an entire image into CNN

3. **RPN** -> proposal



4.Proposal mapping to feature，**RoI pooling**

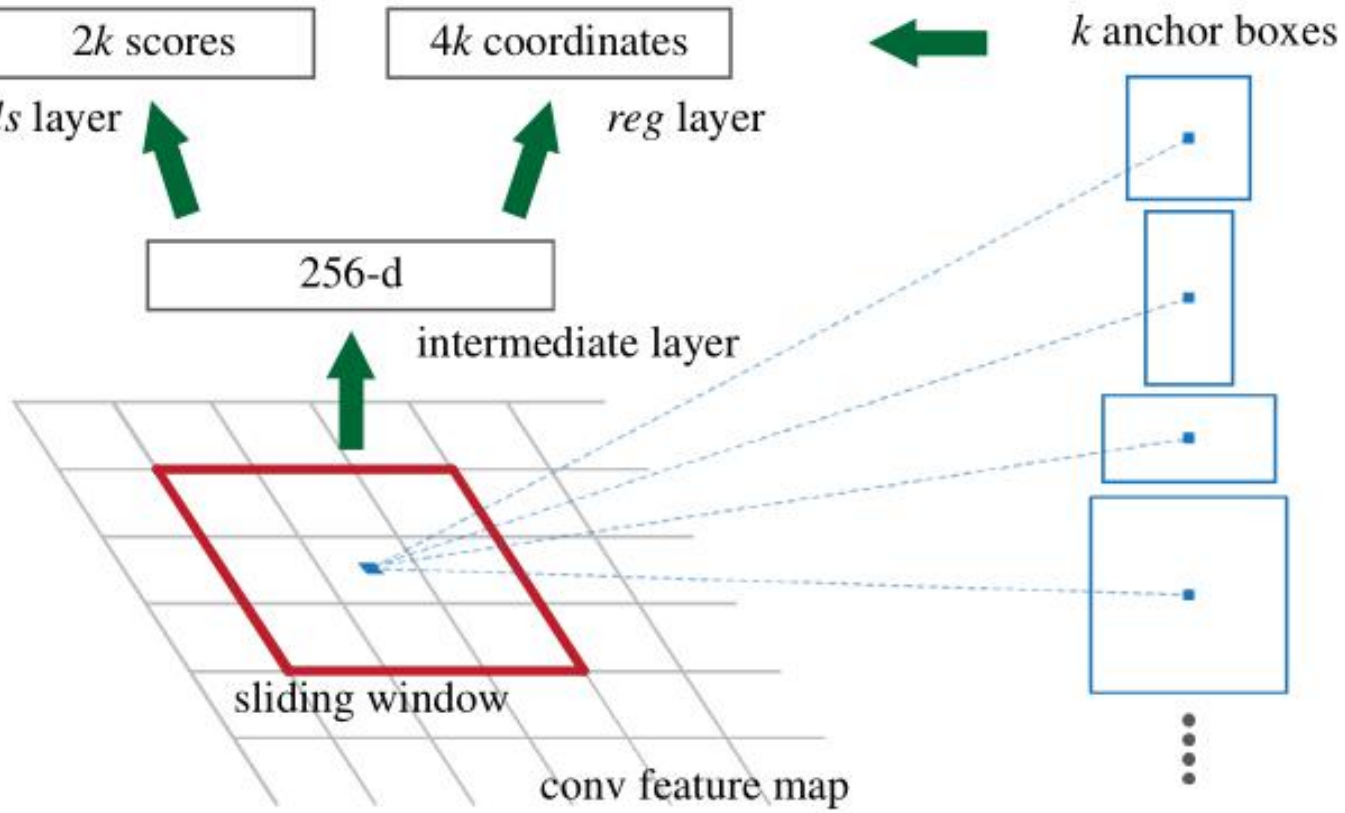
5. Softmax Loss和 Smooth Loss联合训练

Faster R-CNN

基本步骤:
1.特征提取：对整张图片输进CNN，得到feature map
2.候选区域生成：卷积特征输入到RPN，得到候选框的特征信息
3.类别判断：对候选框中提取出的特征，使用分类器判别是否属于一个特定类
4.位置精修：对于属于某一类别的候选框，用回归器进一步调整其位置

Faster R-CNN-> RPN

- 3*3 **window** sliding 取一个中心
- **3** scale：
  128*128,256*256,512*512
- **3** aspect ratio：
  1：1， 1：2， 2：1
- W*H feature map
  -> ***W*H*k anchor***
  ->**2* W*H*k scores**
  ->**4* W*H*k coordinates**



输出:每一个anchor输出种类(前景后景)
置信度 + 位置偏移量 t2

RPN ->anchor-> foreground/background

foreground->
1.The highest IoU
2.IoU > 0.7

background->
1.IoU < 0.3

Other anchor->
do not contribute to the training objective。

前提：覆盖到feature map边界线上的anchor不参与训练。

RPN ->loss function

- **Union loss function**

t_i：anchor
ti*：GT

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

- details

Log loss

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$$

$$L_{\text{loc}}(t^u, v) = \sum_{i\in\{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

Smooth：鲁棒性函数

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

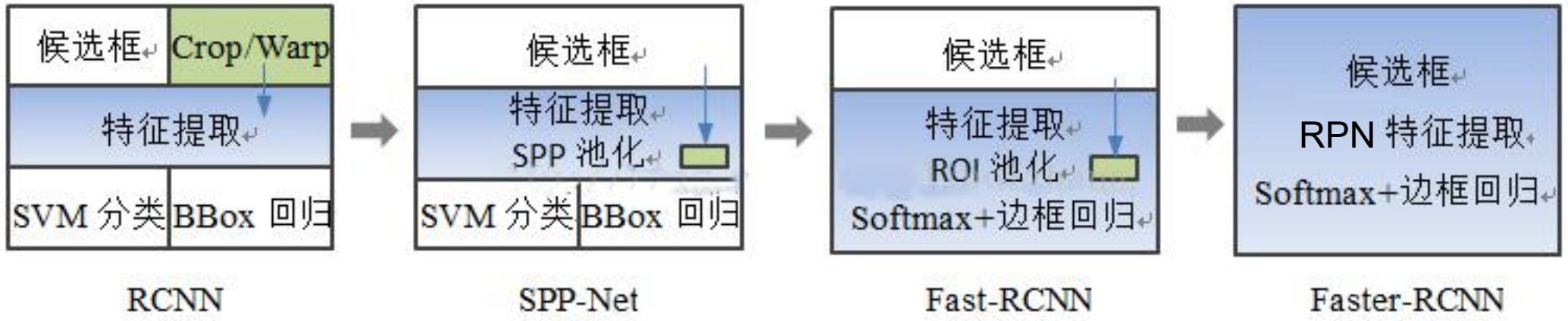
$$t_{\text{x}} = (x - x_{\text{a}})/w_{\text{a}}, \quad t_{\text{y}} = (y - y_{\text{a}})/h_{\text{a}},$$
$$t_{\text{w}} = \log(w/w_{\text{a}}), \quad t_{\text{h}} = \log(h/h_{\text{a}}),$$
$$t_{\text{x}}^* = (x^* - x_{\text{a}})/w_{\text{a}}, \quad t_{\text{y}}^* = (y^* - y_{\text{a}})/h_{\text{a}},$$
$$t_{\text{w}}^* = \log(w^*/w_{\text{a}}), \quad t_{\text{h}}^* = \log(h^*/h_{\text{a}}),$$

Faster R-CNN vs Fast R-CNN

- 使用**RPN**(Region Proposal Network)代替原来的Selective Search方法产生建议窗口。

- Train stage：
  输出约2000个proposal。
  但只会抽取其中**256**个proposal来训练RPN的cls+reg结构，速度更**快**。

RCNN、SPP-Net、Fast-RCNN、 Faster-RCNN对比

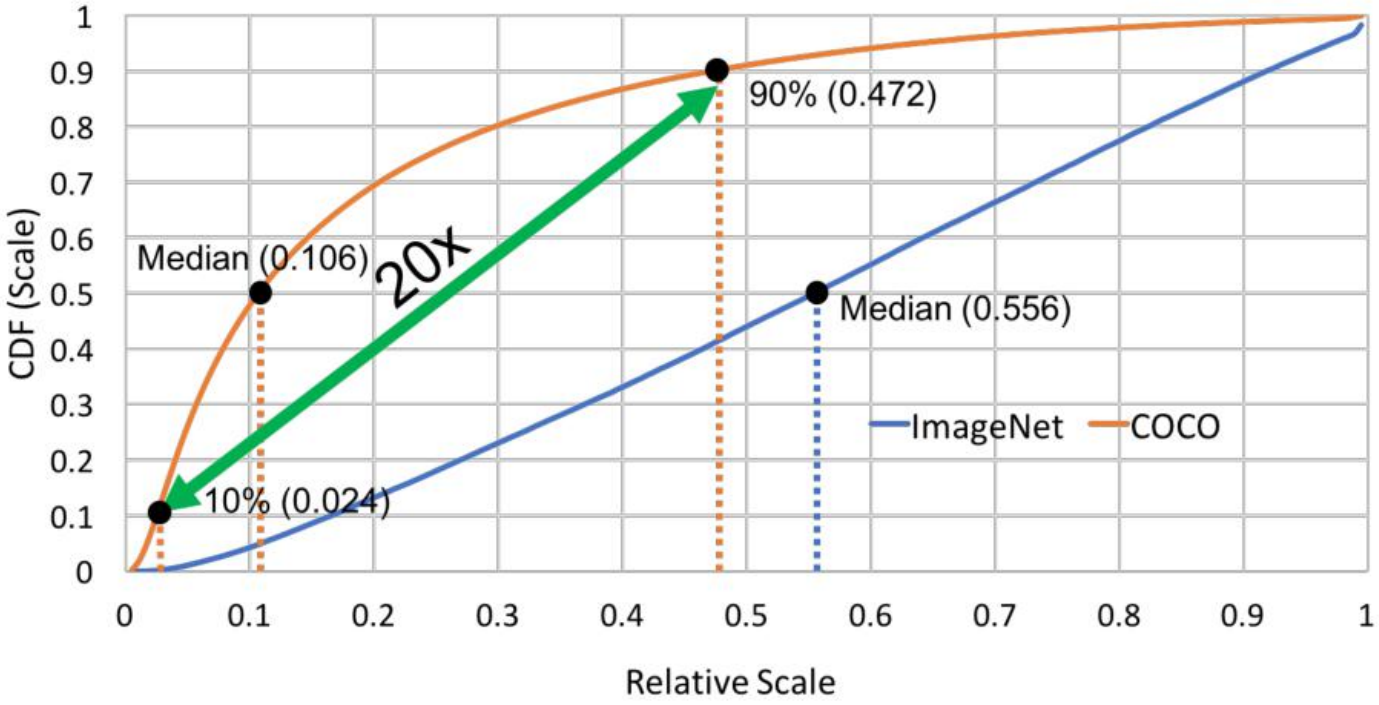


RCNN、SPP-Net、Fast-RCNN、 Faster-RCNN总结

R-CNN	SPP-net	Fast R-CNN	Faster R-CNN
Selective Search + CNN + **SVM**	**RoI Pooling**	Selective Search + CNN + **RoI**	**RPN** + CNN + RoI

Scale Invariance in Object Detection – SNIP

- COCO数据集中的object**尺寸变化范围非常大**
- **90%的物体在0-0.5scale中**

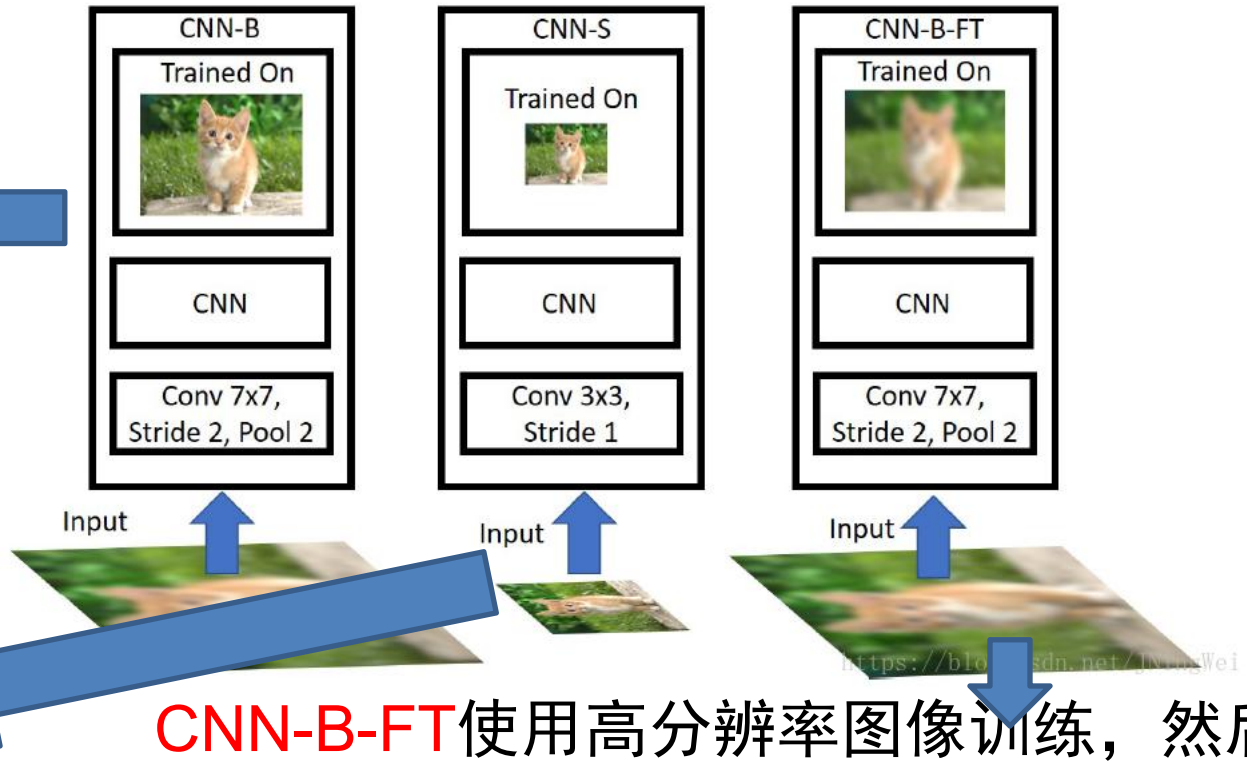


SNIP-> Motivation

1.**Upsample**是必要的吗？	现在检测网络为了提高对小目标的效果，都采用**upsample**的方法，这个真的是必要的吗？是否能直接用低分辨率的图像不做upsample来训练在ImageNet上用低像素图片预训练一个缩放倍数较小的CNN。
2.是否需要**所有scale**的object都参与训练？	用ImageNet做预训练模型的时候训练检测器的时候，**是否需要所有尺寸的object都参与进来？**用所有的gt来训练效果真的更好吗？ 能否通过挑选样本的方式来增加网络效果的，比如upsample调整了大小以后，**只用64x64~256x256范围的ground truth来训练**？

Test guess strategy

CNN-B使用高分辨率图像训练，分类经过降采样和upsample的图片。(模拟训练数据的分辨率（resolution）和验证数据的分辨率不一致)

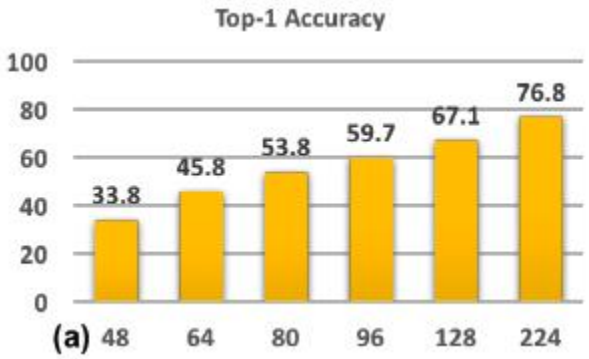


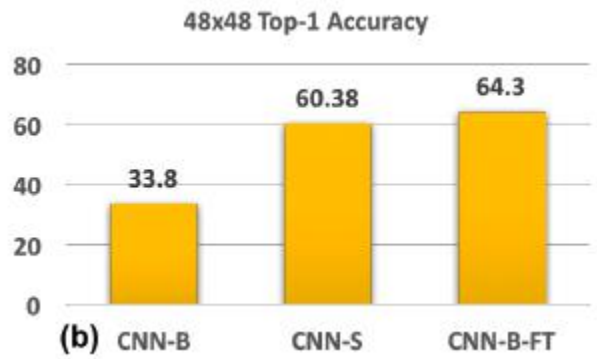
CNN-S使用低分辨率图像训练，分类经过降采样的图片。(模拟训练数据分辨率和验证数据分辨率一致时的效果)

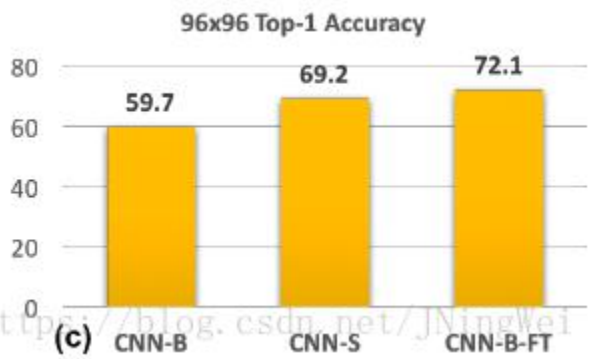
CNN-B-FT使用高分辨率图像训练，然后在低分辨率图像上fine-tune，分类经过降采样和upsample的图片。(验证基于高分辨率图像训练的模型是否能有效提取低分辨率图像的特征)

	训练	测试	
CNN-B	48*48	224*224	最不好
CNN-S	224	224	中等
CNN-B FT	448	224	最好

Guess experiment result





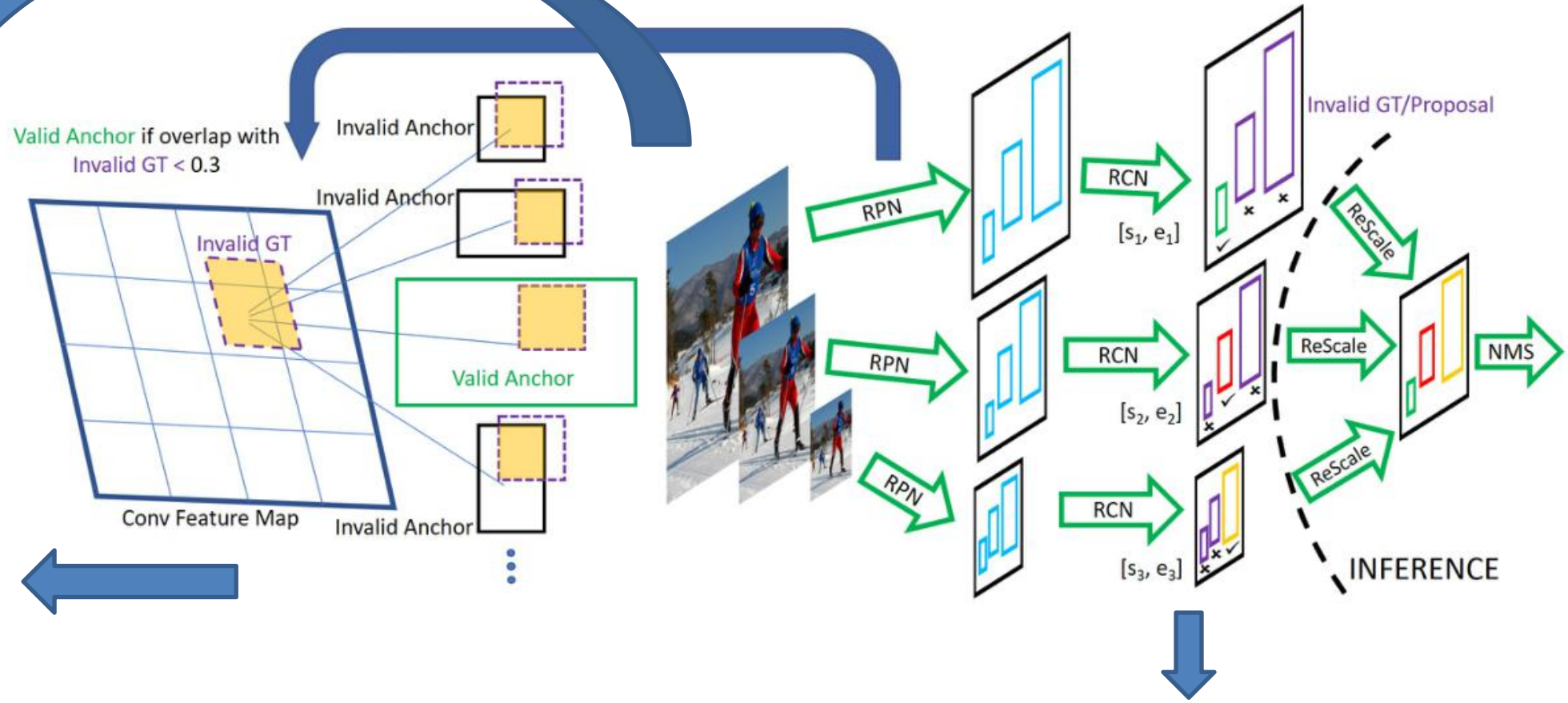


Upsample可以在一定程度上提高小目标的检测效果，但会让**本来正常大小或者本来就大的目标过大，性能下降**(CNN-B).

使用**高分辨率图像预训练与upsample微调**得到的模型是更好的选择（CNN-B-FT）

SNIP framework

1.Sampling
Sub-Images

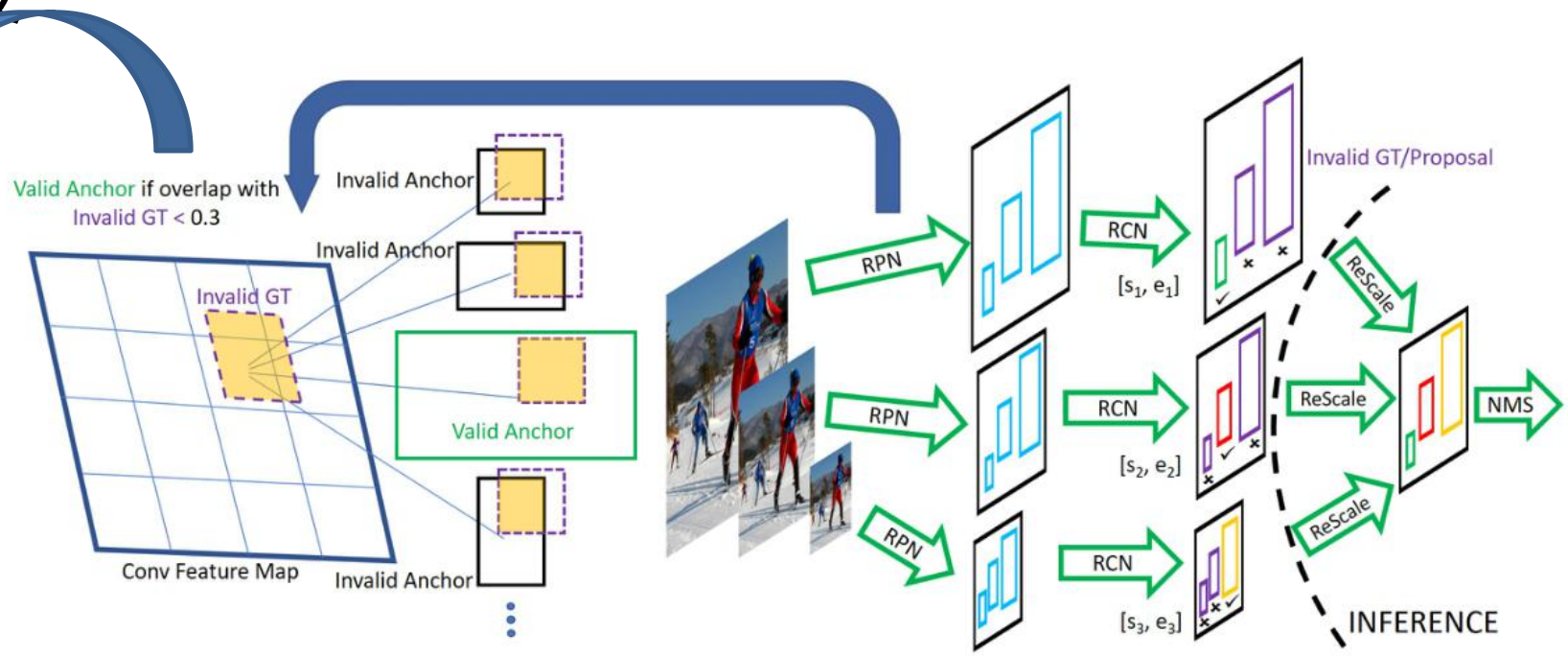


2.去除**invalid anchor**

3.**Scale Normalization for Image Pyramids**

SNIP-> RPN阶段

与invalid GT超过**0.3**区域重叠的anchor在训练中被**去除**,得到valid anchor

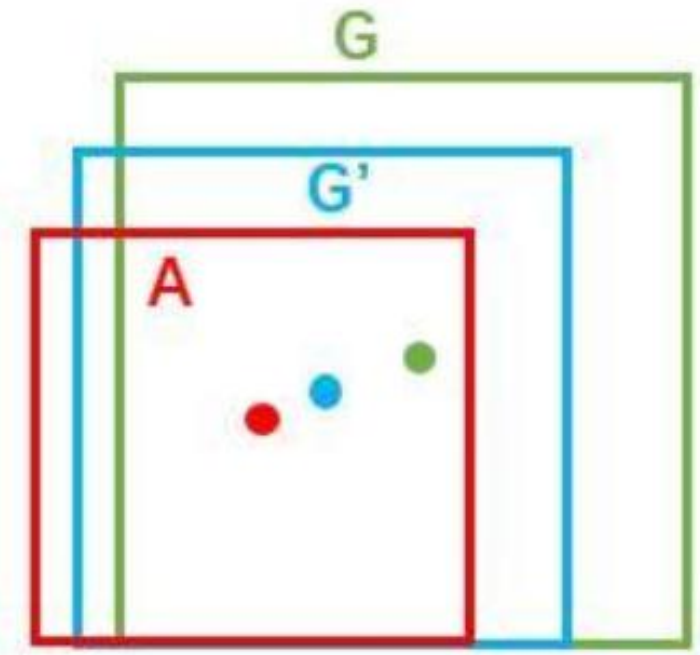


对于大尺度的feature map，只负责预测被放大的小物体；对于小尺度的feature map，只预测被缩小的大物体。

SNIP对只有与预训练数据分辨率相近的目标实例（224×224）才被用来来训练检测器。在每个特定分辨率下只保留RoI[Smin,Smax]的anchor，其他anchor被标记为invalid anchor。

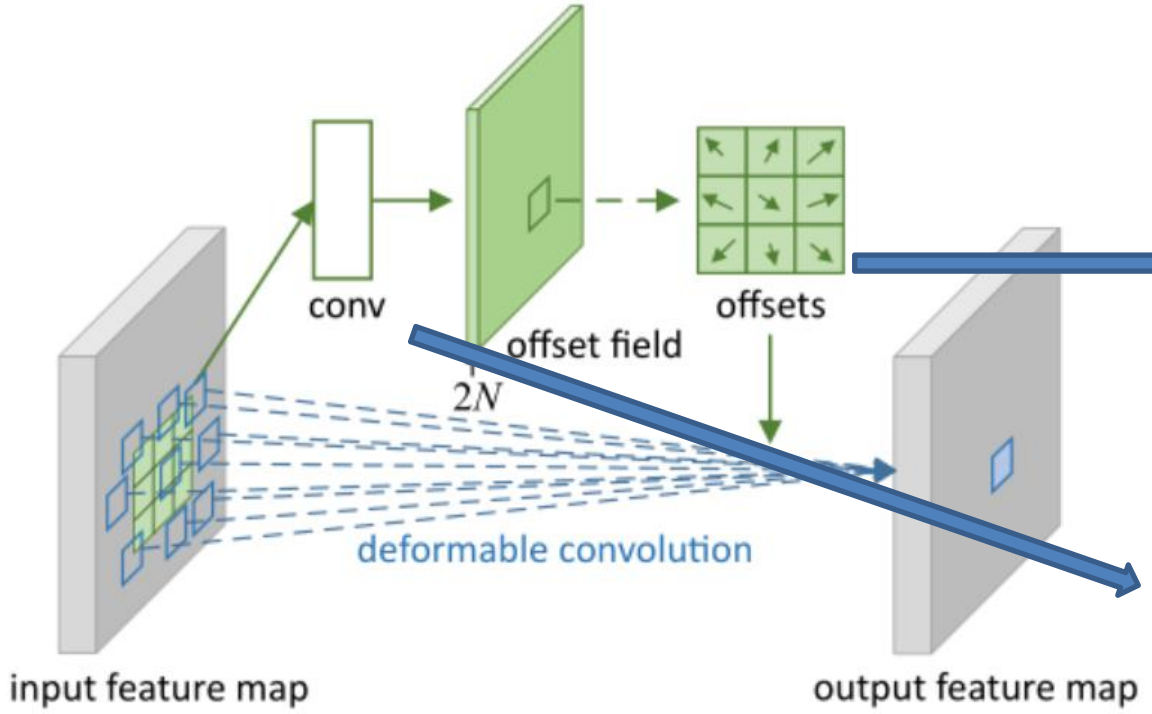
SNIP->Regression,fine tuning

- 寻找一种关系：
- 使得原始的proposal经过映射得到一个与GT box G更接近的regression box G’
- **A->G’**



SNIP-> Deformable RFCN detector

- 使用Deformable RFCN detector而不是常见的一般卷积。



Regular convolution

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n)$$

Deformable Convolution

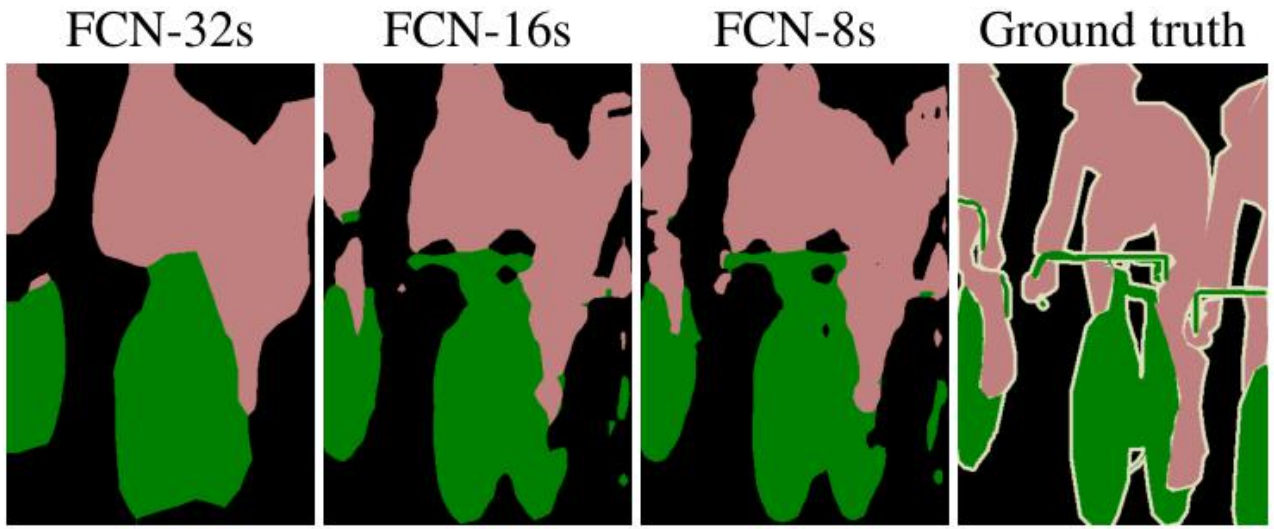
$$y(p_0) = \sum_{p \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n)$$

where Δp_n is generated by a sibling branch of regular convolution

Region-based Fully Convolutional Networks： R-FCN

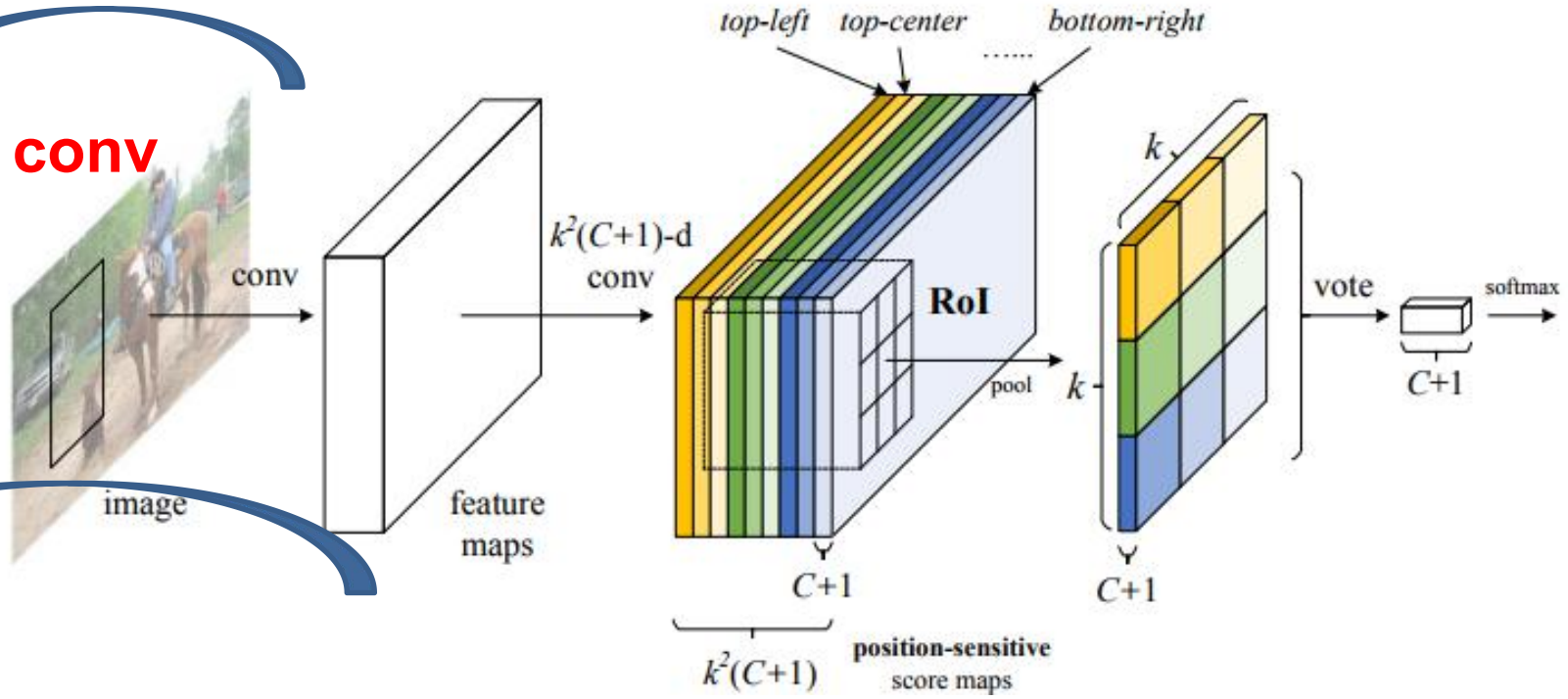
- Motivation：
- deeper layer
  -> translation invariance -> boost
  -> translation variance -> recoil

当卷积网络变深后，feature map会越来越小
输入图片中物体的小偏移，经过N多层pooling后在最后的小feature map上会感知不到。



R-FCN framework

1.Backbone architecture:
ResNet 100 + **1*1*1024 kernel conv**
-> 2048-d to 1024-d
（Dimension reduction）



2.**k^2(C+1)-d conv**：
C+1 layer per section
Total is k^2(sections) *
(C+1)(categories + background)

3.RoI Pooling：
Pooling from C+1 categories

4.Vote on a RoI

SNIP核心思想

- 对于大尺度的feature map，只负责预测被放大的小物体；
- 对于小尺度的feature map，只预测被缩小的大物体。

SNIPER: Efficient Multi-Scale Training

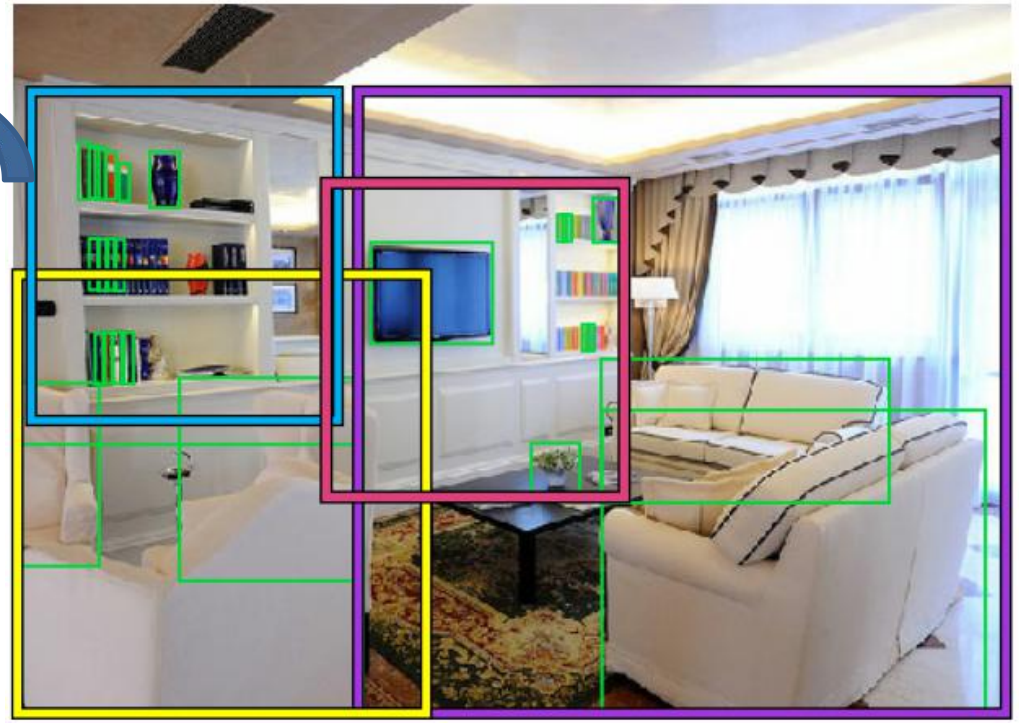
- Motivation：
- 在**高分辨率图片**上进行训练这一常用技巧是不是提升精度所**必须的**？

- 在缩放原图时，object也会跟着缩放，这就导致原来比较大的object变得**更大**了，原来更小的object变得**更小**了，这对**尺度极端物体**的检测是不利的。

SNIPER: Efficient Multi-Scale Training framework

1.Chip **Generation**

2. **Positive** Chip Selection



3. **Negative** Chip Selection

4. Label Assignment

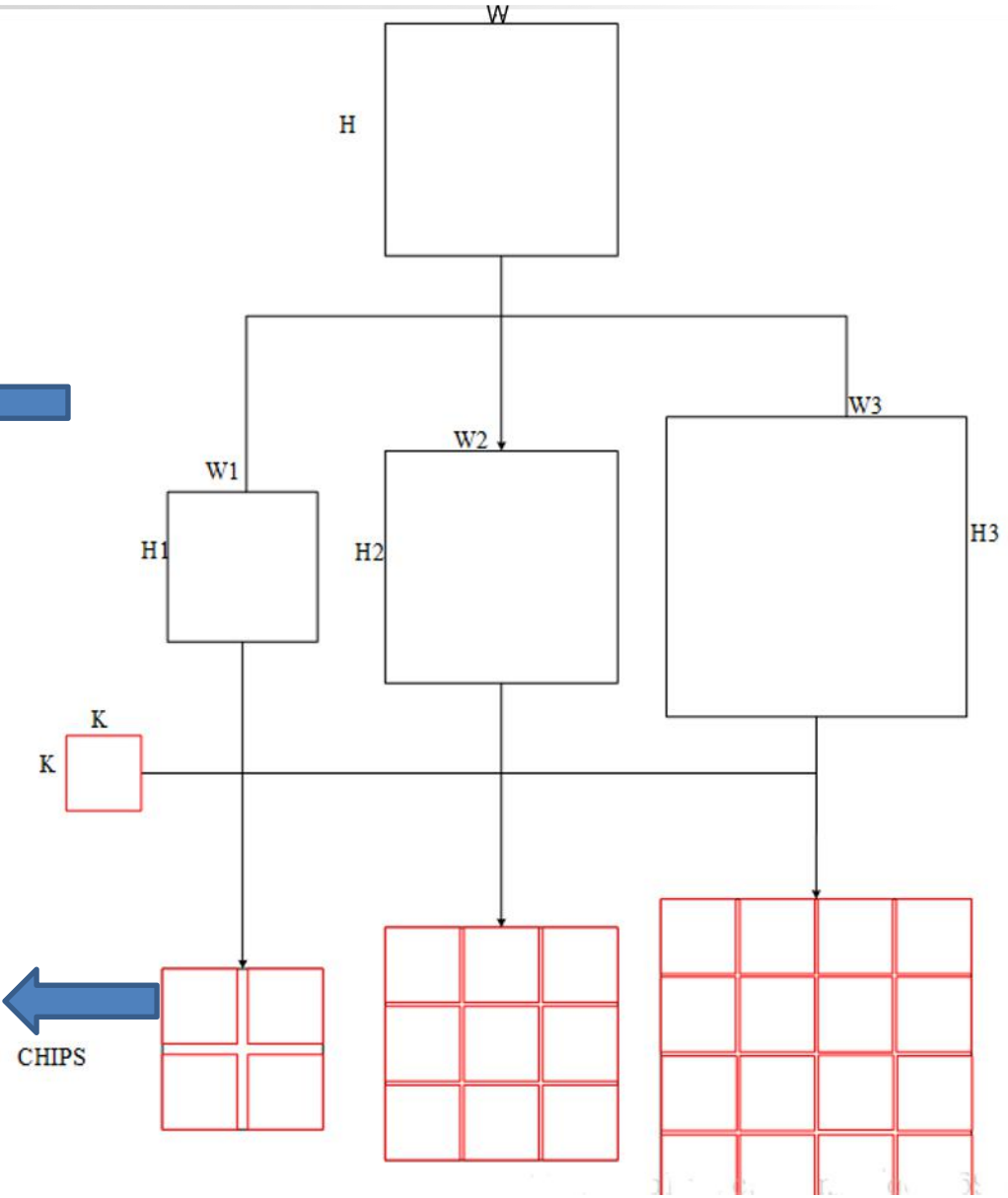
SNIPER-> Chip Generation

1.**resize** 3 scale(512/ms, 1.667, 3)

2. **K*K**的chip，按d个像素等距离放置在图片。

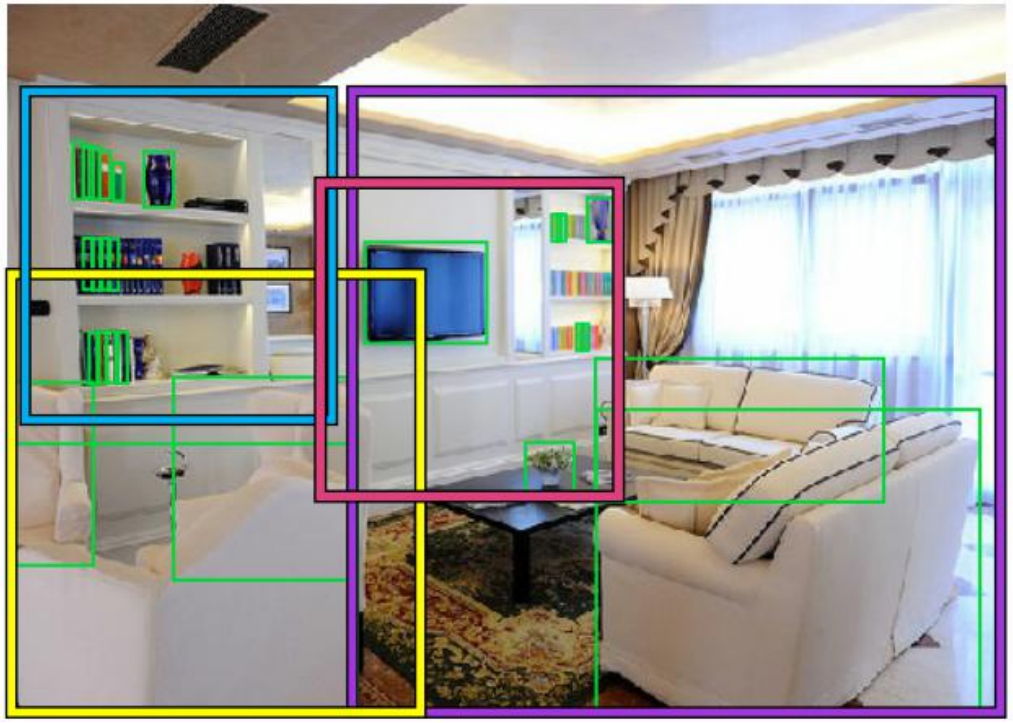
3.得到chips。

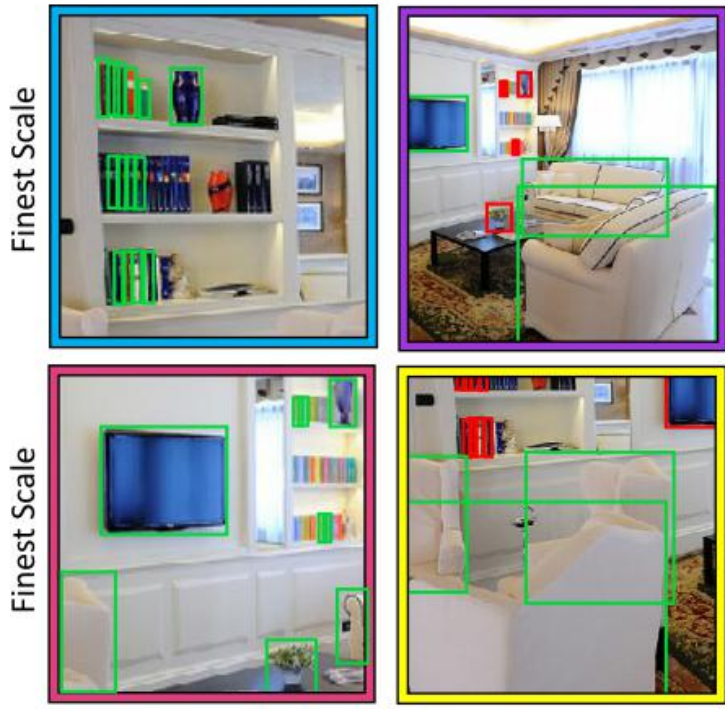
4.每个scale都会生成chips，而且chip的大小是固定的，变化的是图片的尺寸，与anchor的思想刚好相反，这种思路非常具有**启发性**。



SNIPER-> Positive Chip Selection

1. 每个scale**在 Ri 范围内**的ground truth box 称为有效的，其余为无效的。





2.从所有chips中选取包含（完全包含）有效 gt box**最多**的chip，作为Positive Chip。

3.前提：每个scale的所有chips包含了整张图片**全部**的gt box。

SNIPER-> Negative Chip selection

- 若参与训练的图片**只包含 postive chip**，那么由于有大部分的区域背景在postive chip，没人告诉网络“这样的背景时错误的”。
- 算法对背景的识别效果**不好**，导致假正例的比率比较高，容易把背景误判成目标。
- 因此需要**生成 negative chip送入训练**来提高网络对背景的识别效果。
- **保持正负样本比平衡**。

SNIPER-> Negative Chip selection

1. RPN，生成一系列proposals。

2.对每个scale i，**移除**该scale下被 positive chip 覆盖到的那些proposals。

3. 每个scale的chip**选取**包含至少M个proposals的chips作为**negative chip。**

4.每张图片训练时，在每个epoch中输入**全部positive chip**和选择**固定数量的negative chip**。

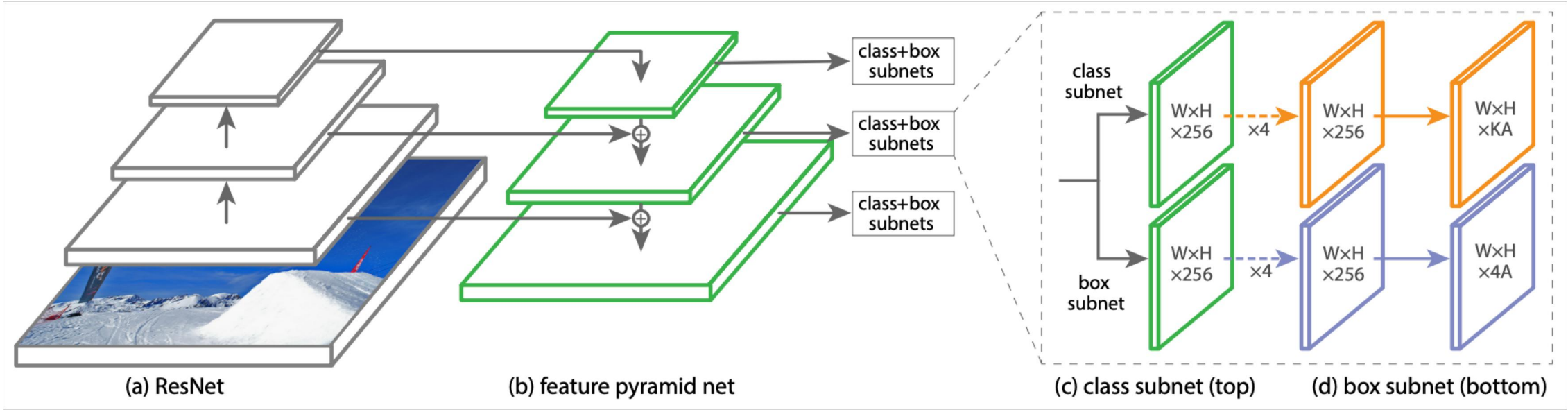
SNIPER vs SNIP

- 同时引入了positive chip和**negative chip**，保证正负样本比例适当。
- SNIPER通过跳过easy chip，**只训练挑选好的chips**，相比SNIP减少了计算量。
- 在低分辨率图片上做训练，**减少了内存占用**，性能可从batch normalization中受益。

RetinaNet

FPN

heavier head than
SSD / Faster R-CNN



RetinaNet

基本步骤:
1.主干网络使用ResNet101
2.使用金字塔结构进行特征融合，在每一个特征图上进行预测
2.两个分支预测网络分别为类别的判断和框的回归。

RetinaNet

Focal Loss

- Problem: **class imbalance**
  - inefficient training
  - loss is **overwhelmed by negative samples(**

Model	Solution
Two-stage detectors	1) proposal 2) mini-batch sampling
SSD	Hard negative mining
RetinaNet	Focal loss

RetinaNet

Focal Loss

- Solution: **high confidence -> small loss**

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

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

RetinaNet

- **Focal loss能动态地缩放交叉熵**

- 当某样本类别比较明确些，它对整体loss的贡献就比较少；而若某样本类别不易区分，则对整体loss的贡献就相对偏大， easy example可以通过权重进行抑制。

- 使得模型在训练时**更专注难分类的样本**，从而改善样本的类别**不均衡**问题，改善模型的优化方向。

One-Stage

目标检测/检测方法	YOLO	SSD	YOLOv2	YOLOv3	RetinaNet	RFB Net	FCOS
主干网络	GoogleNet	VGG16	DarkNet-19	DarkNet-53	ResNet	VGG16	ResNet101
Anchor-based		√	√	√	√	√	
Multi-scale		√	√	√	√	√	√
是否有分支预测网络					√		√
其他		多层次特征分支辅助预测	1.reorg多层次的特征。 2.Route不同层次特征。 3.在anchor选取前进行维度聚类统计	多层特征分支辅助预测	使用了Focal loss损失	使用了RFB-S和RFB模块	1.逐像素预测。 2.使用了不同的框描述。 3. 提出center-ness去除低质量框

Two-Stage

目标检测步骤/检测方法	R-CNN	SPP-Net	Fast R-CNN	Faster R-CNN	R-FCN
输入大小	固定大小	任意大小	任意大小	任意大小	任意大小
候选区域生成	Selective Search	Selective Search	Selective Search	RPN	RPN
特征提取	AlexNet	Alexnet	VGG16	ResNet101	ResNet101
类别判断	线性SVM	线性SVM	SoftMax分类	SoftMax分类	位置敏感池化结果