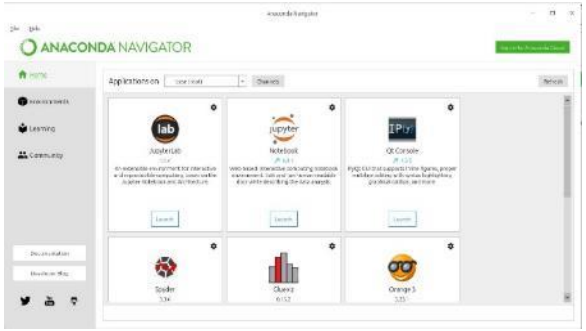
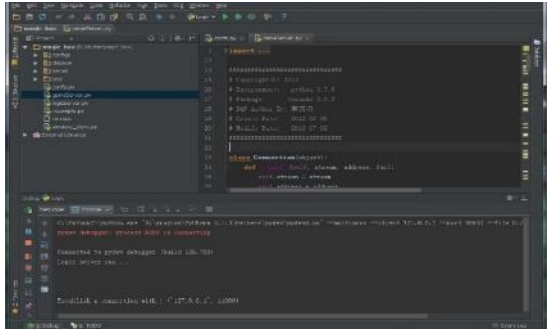
内容回顾：物体检测环境配置



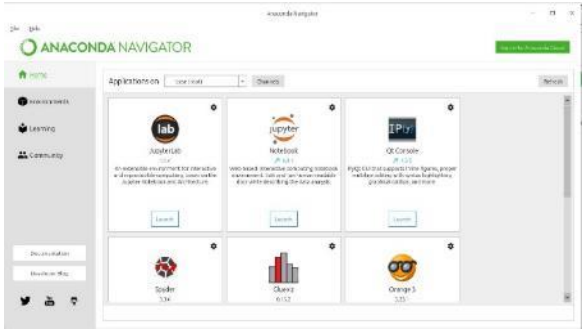
环境管理软件Anaconda

物体检测平台Detectron2



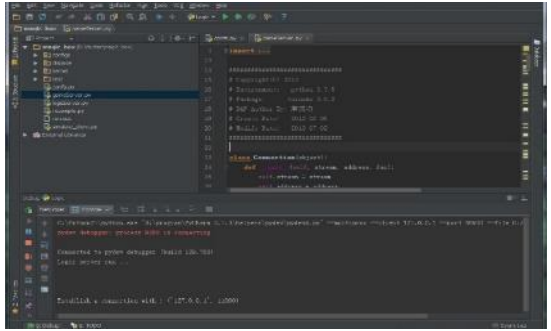
代码调试软件PyCharm

内容回顾：通用物体检测概述



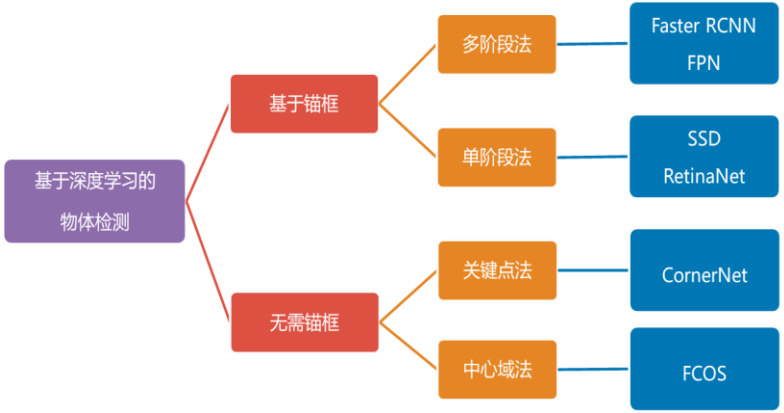
环境管理软件Anaconda

物体检测平台Detectron2



代码调试软件PyCharm

通用与特定物体检测



通用物体检测算法

数据库	图片数量	标注数量	类别数量
PASCAL VOC 2007	9963	24640	20
PASCAL VOC 2012	11540	27450	20
MS COCO	约14万	5171	80
LVIS	16.4 万	200万	1000+
OpenImages	190万	1600万	600

通用物体检测数据集

目录

基于锚框的物体检测

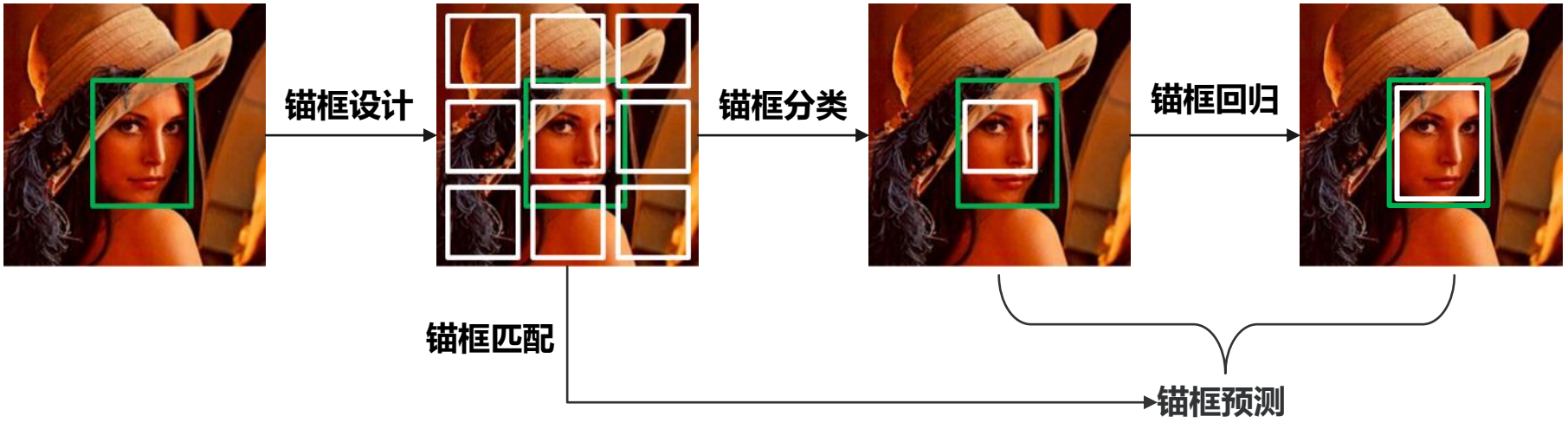
基于锚框的物体检测

基于锚框的物体检测

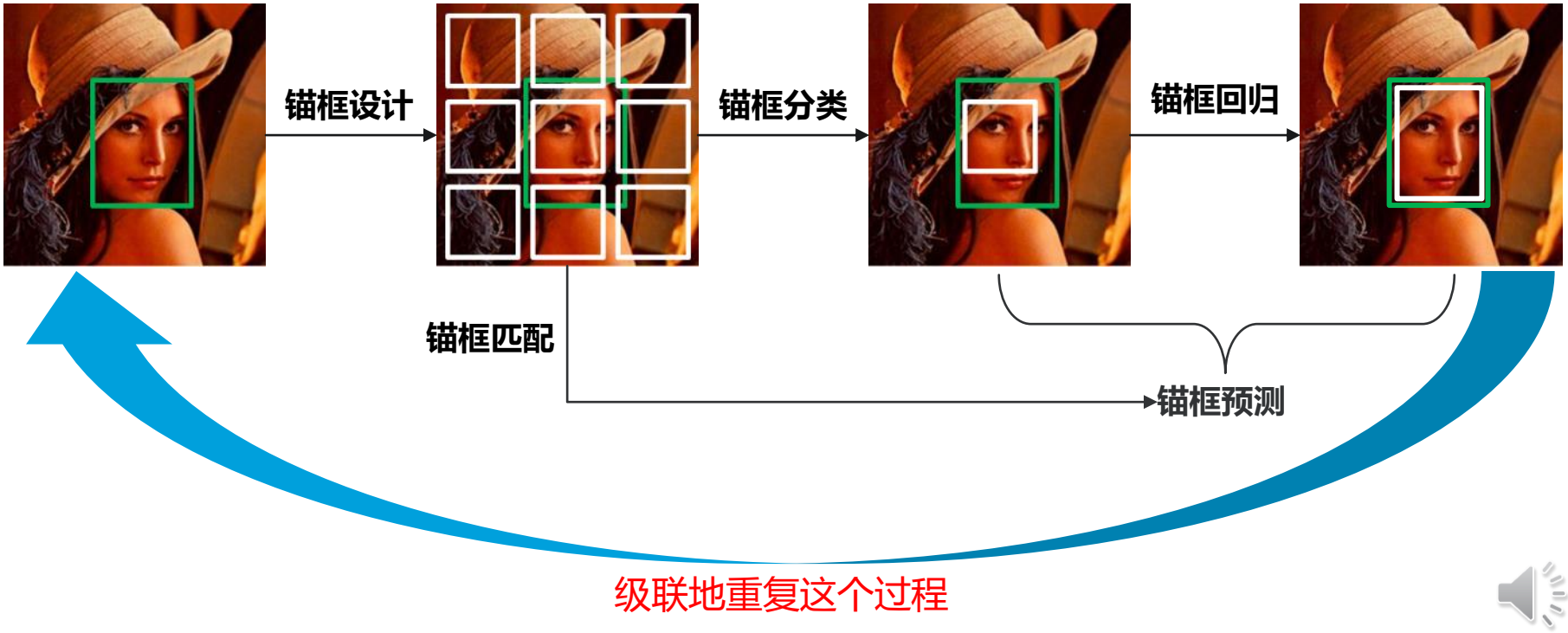
基于锚框的物体检测

基于锚框的物体检测

基于锚框的物体检测

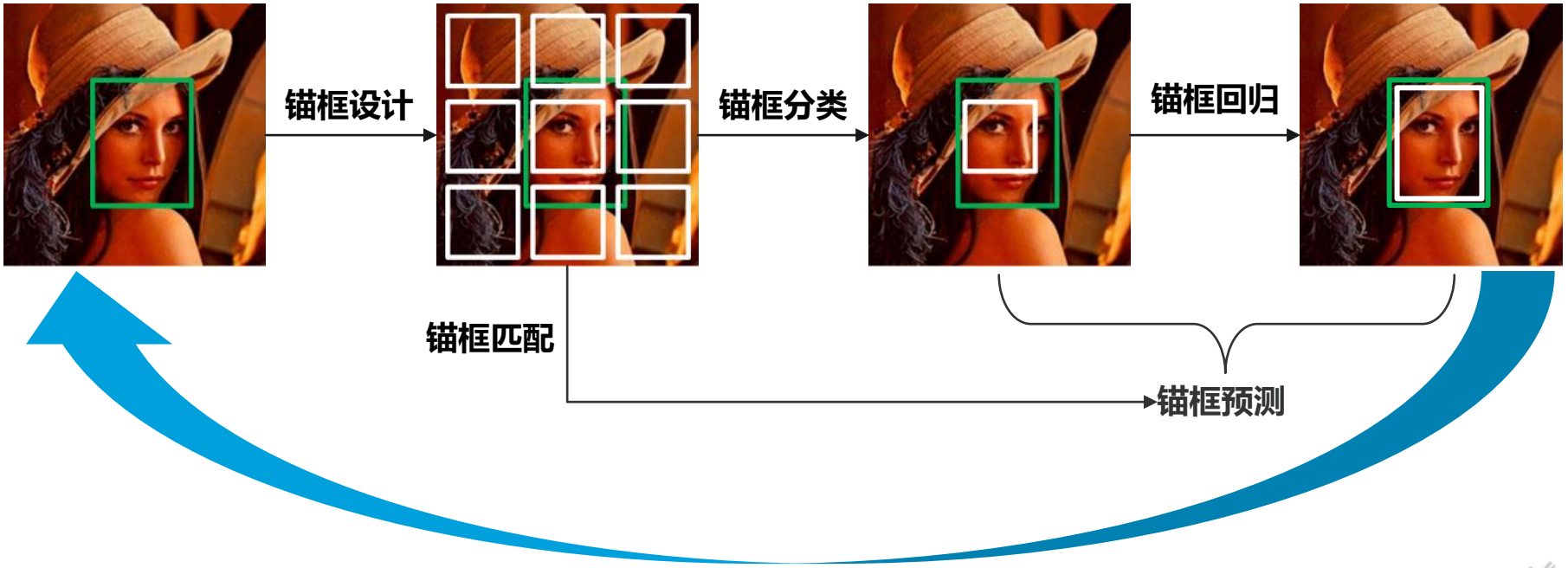


基于锚框的物体检测

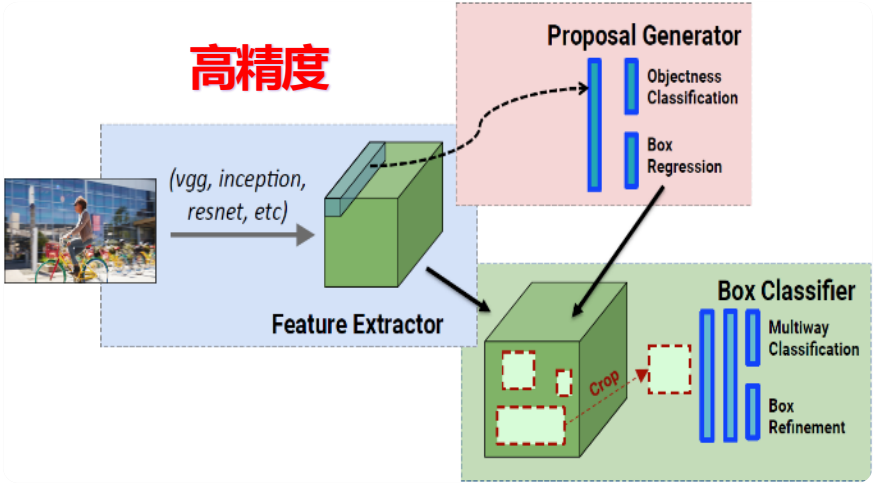


基于锚框的物体检测

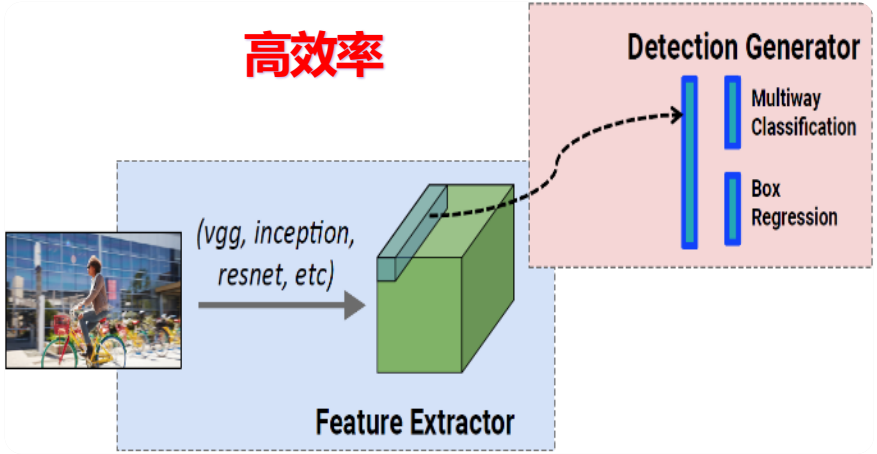
锚框机制是该类物体检测算法的核心



基于锚框的物体检测

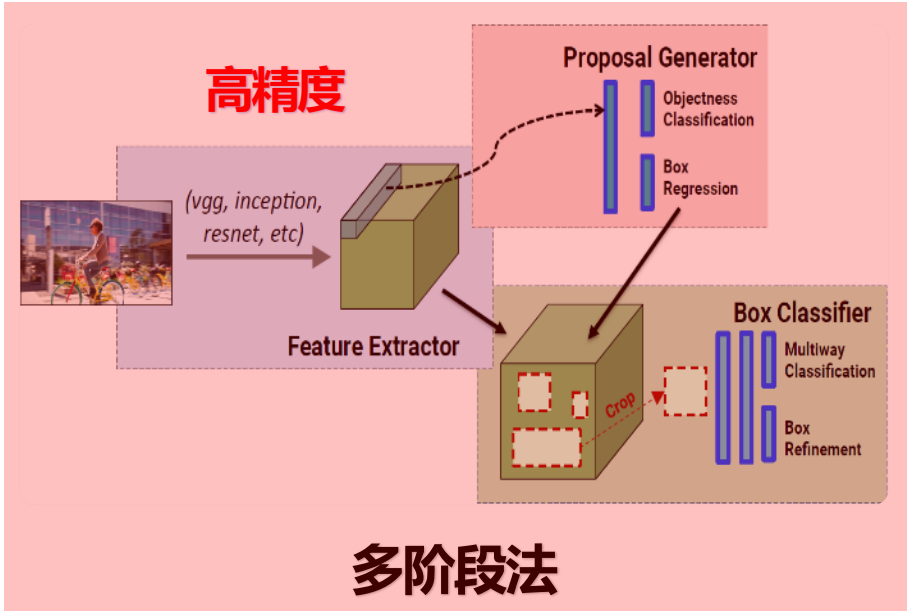


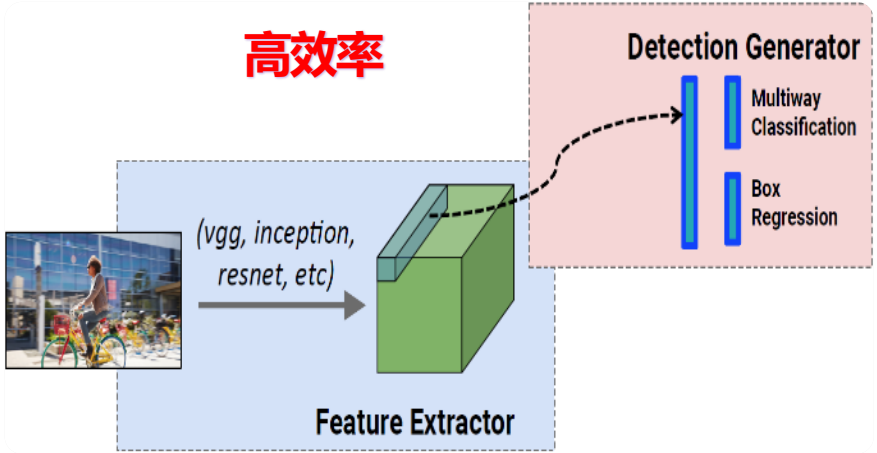
多阶段法



单阶段法

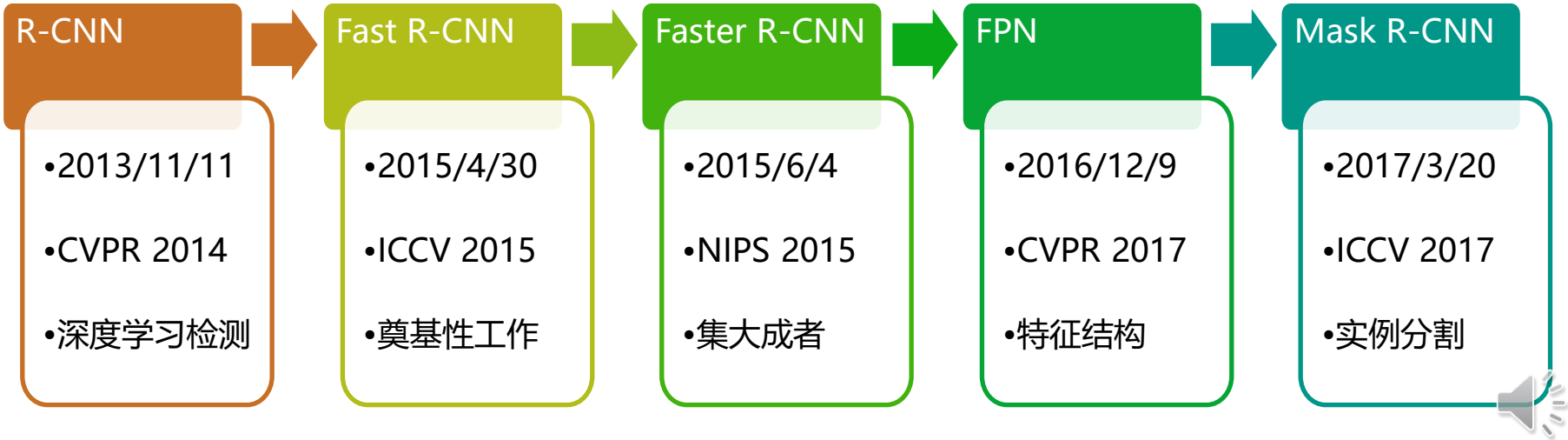
基于锚框的物体检测


高精度
Proposal Generator
Objectness Classification
Box Regression
(vgg, inception, resnet, etc)
Feature Extractor
Box Classifier
Multiway Classification
Box Refinement
Crop
多阶段法

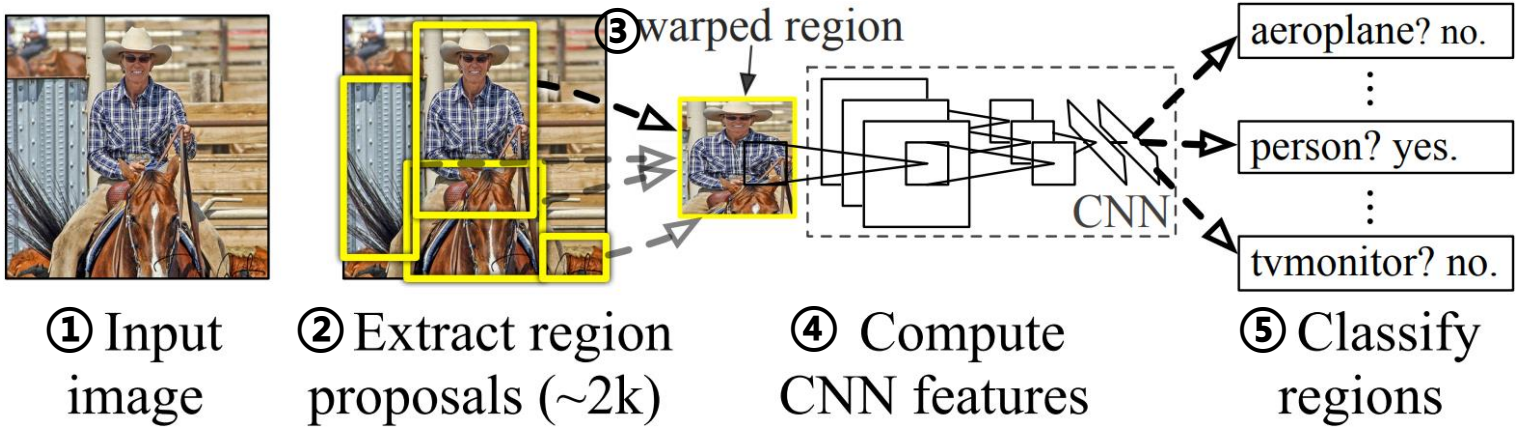

高效率
Detection Generator
Multiway Classification
Box Regression
(vgg, inception, resnet, etc)
Feature Extractor
单阶段法

基于锚框的多阶段检测算法

- 基于锚框的多阶段检测算法的集大成者是Faster R-CNN
- Faster R-CNN是一系列算法，从2013年的R-CNN出现，到2015年中旬的Faster R-CNN成熟，再到后来的开枝散叶
- 下面简单介绍R-CNN和Fast R-CNN，接着详细介绍Faster R-CNN
- 掌握了Faster R-CNN，便可以掌握绝大部分的物体检测算法



基于锚框的多阶段检测算法：R-CNN



R-CNN的检测步骤：

① 输入图像：输入一张待检测的图像

② 候选区域生成：使用Selective Search算法，在输入图像上生成~2K个候选区域

③ 候选区域处理：在输入图像上裁剪出每个候选区域，并缩放到227*227大小

④ 特征提取：每个候选区域输入CNN网络提取一定维度（如4096维）的特征

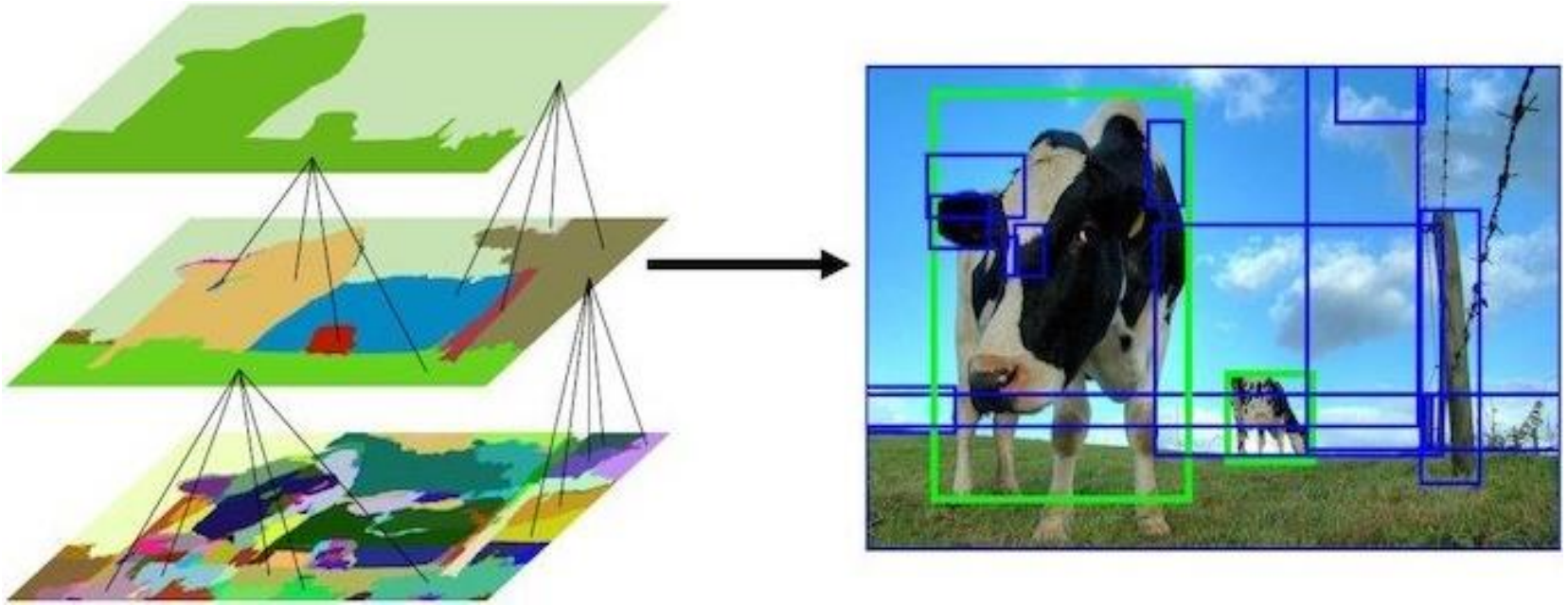
⑤ 类别判断：把提取的特征送入每一类的SVM 分类器，判别是否属于该类

⑥ 位置精修：使用回归器精细修正候选框位置

Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

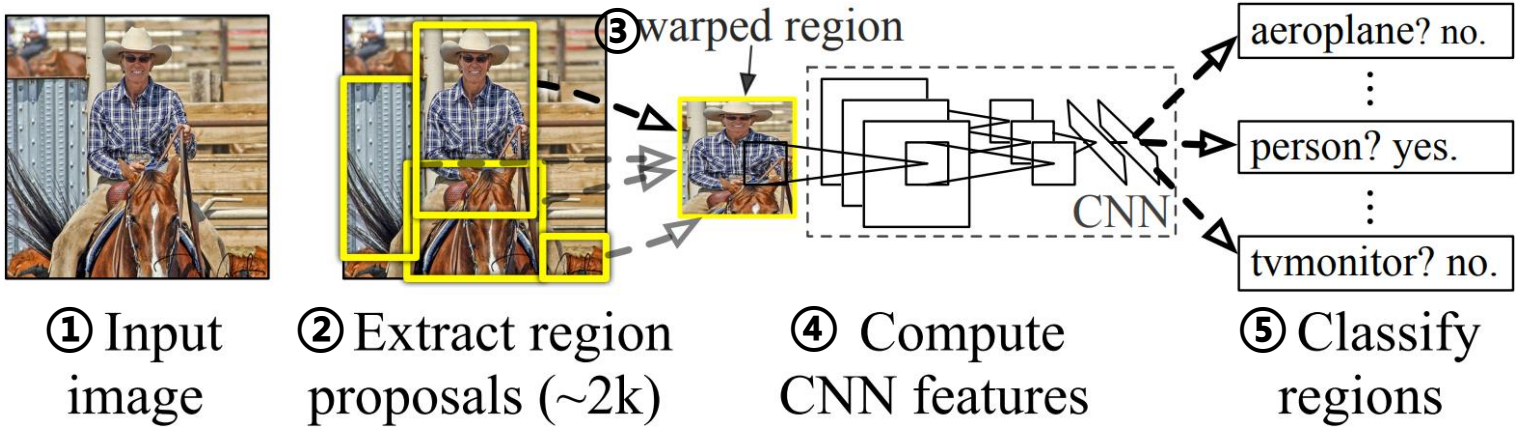
基于锚框的多阶段检测算法：R-CNN

- Selective Search算法：利用图像分割产生初始分割区域 -> 利用相似度进行区域合并
- 如何计算两个区域的相似度？计算颜色、纹理、大小和形状交叠的差异，利用不同的权重相加



Selective Search for Object Recognition, IJCV 2013

基于锚框的多阶段检测算法：R-CNN



相比传统方法，检测精度得到大幅度提升，但是速度太慢，原因是：

① 使用Selective Search生成候选区域非常耗时

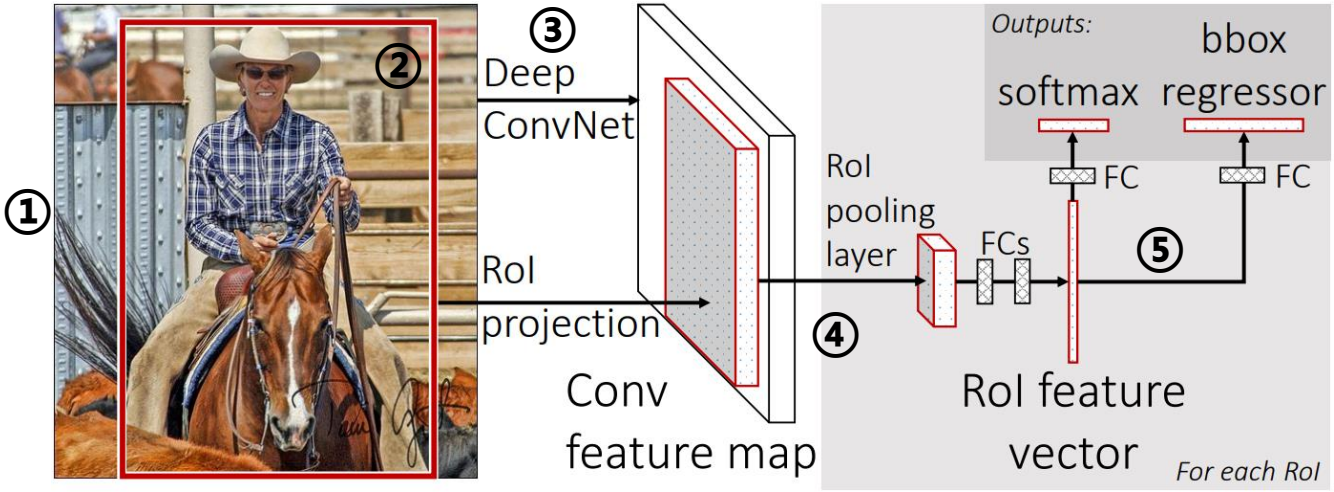
② 一张图像上有~2K个候选区域，需要使用~2K次CNN来提取特征，存在大量重复计算

③ 特征提取、图像分类、边框回归是三个独立的步骤，要分别训练，测试效率也较低

Fast R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

基于锚框的多阶段检测算法：Fast R-CNN



Fast R-CNN的检测步骤：

① 输入图像：输入一张待检测的图像

② 候选区域生成：使用Selective Search算法，在输入图像上生成~2K个候选区域

③ 特征提取：将整张图像传入CNN提取特征

④ 候选区域特征：利用RoIPooling**分别**生成每个候选区域的特征

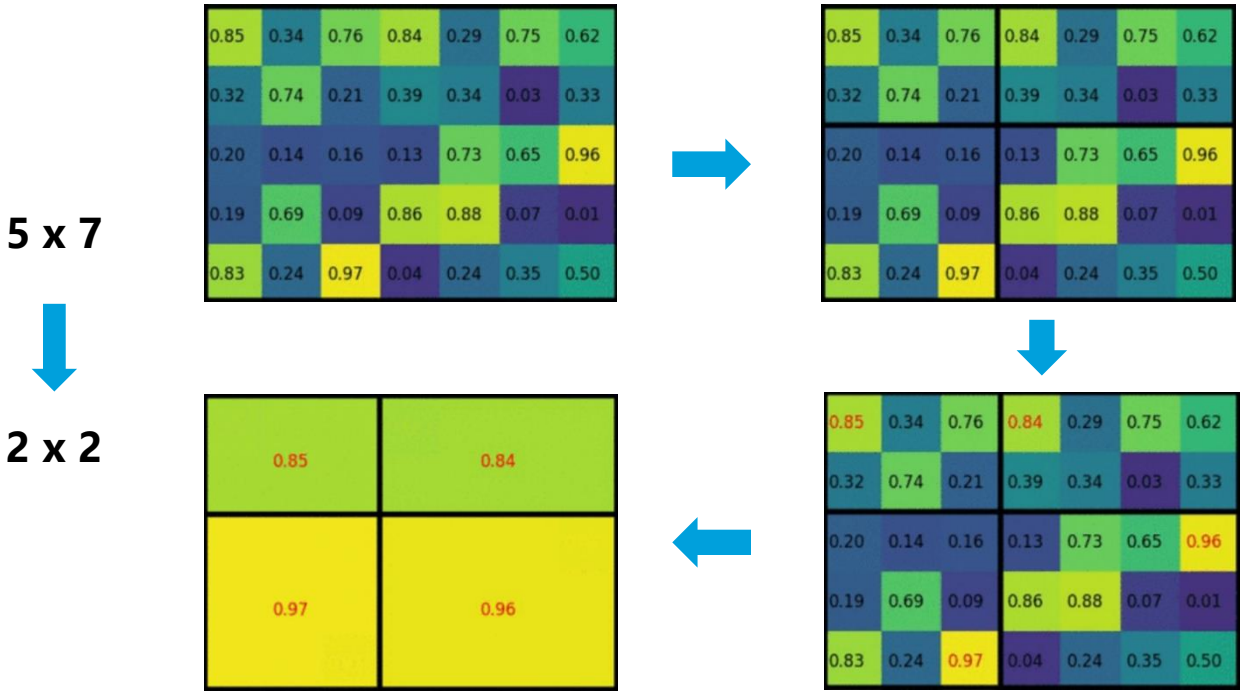
⑤ 候选区域分类和回归：利用扣取的特征，对每个候选区域进行分类和回归

* 注：步骤④⑤仍会存在一些重复计算，但是相对R-CNN少了很多

Fast R-CNN, ICCV 2015

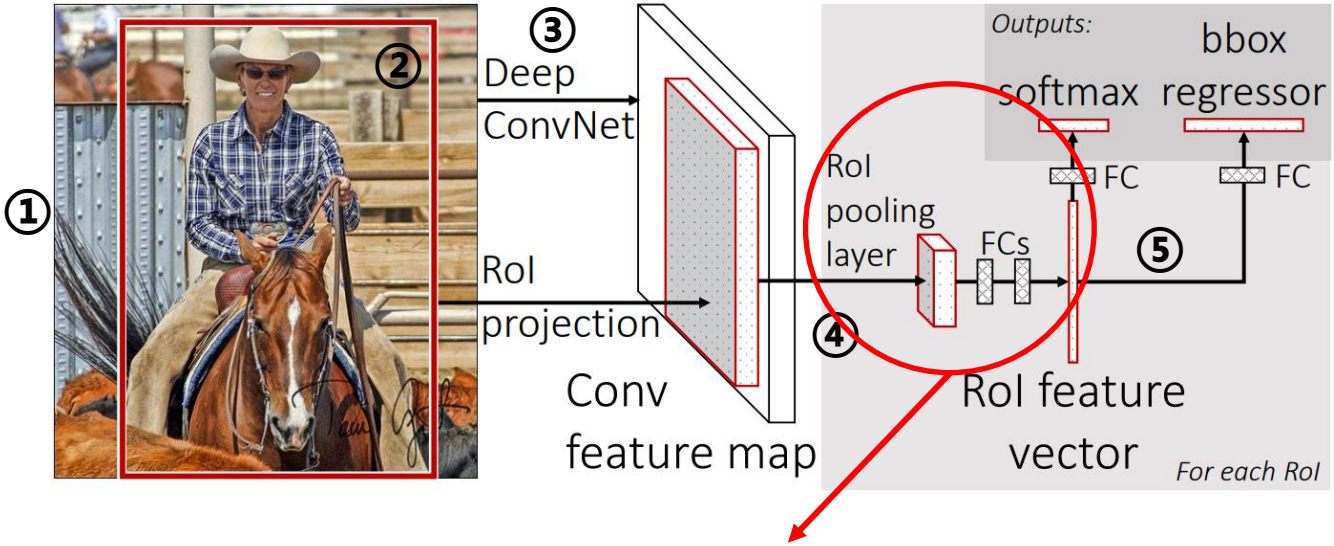
基于锚框的多阶段检测算法：Fast R-CNN

- RoIPooling：利用特征采样，把不同空间大小的特征，变成空间大小一致的特征



Fast R-CNN, ICCV 2015

基于锚框的多阶段检测算法：Fast R-CNN



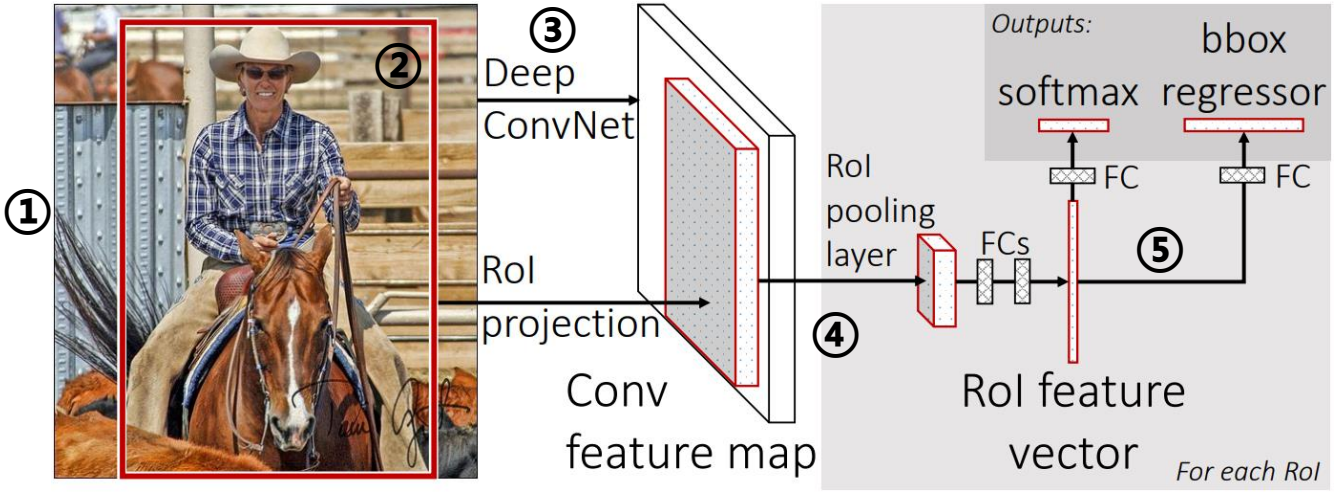
为什么要使用RoIPooling把不同大小的特征变成固定大小？

① 网络后面是全连接层（FC层），要求输入有固定的维度

② 各个候选区域的特征大小一致，可以组成batch进行处理

Fast R-CNN, ICCV 2015

基于锚框的多阶段检测算法：Fast R-CNN

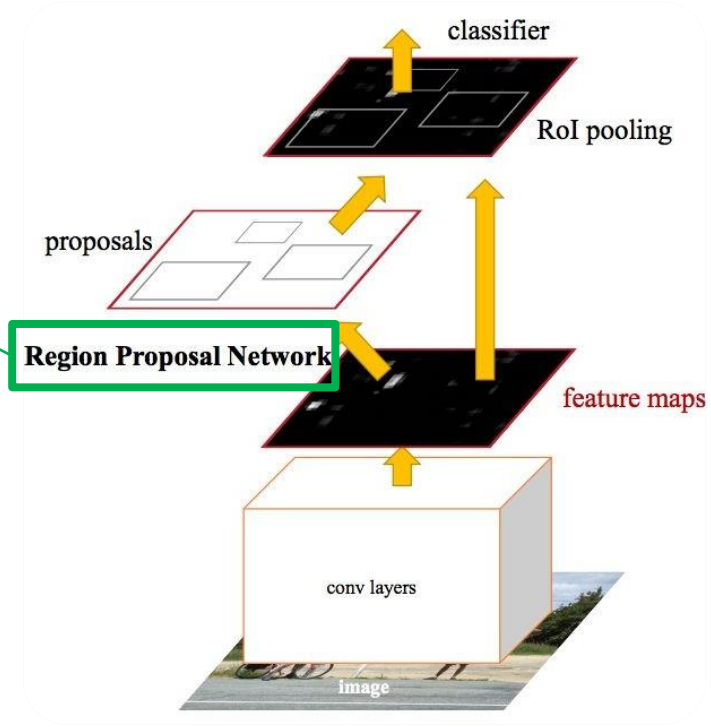


- 端到端的多任务训练
- Fast R-CNN比R-CNN快了200多倍，并且精度更高
- **生成候选区域算法（Selective Search）非常慢（耗时2s）** → **Faster R-CNN**

Fast R-CNN, ICCV 2015

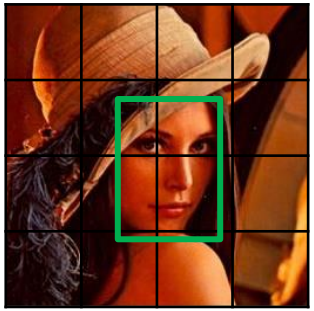
基于锚框的多阶段检测算法：Faster R-CNN

- Fast R-CNN = Selective Search + Fast R-CNN
- Faster R-CNN = RPN + Fast R-CNN
- RPN取代了耗时的Selective Search
- RPN与Fast RCNN共享卷积层
- 引入计算量小，耗时少
- 可以端到端地训练
- Faster R-CNN确定了基于锚框算法的检测流程



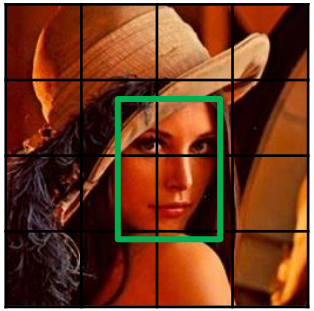
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

Faster R-CNN算法之RPN的原理和流程



4x4输入图像

Faster R-CNN算法之RPN的原理和流程

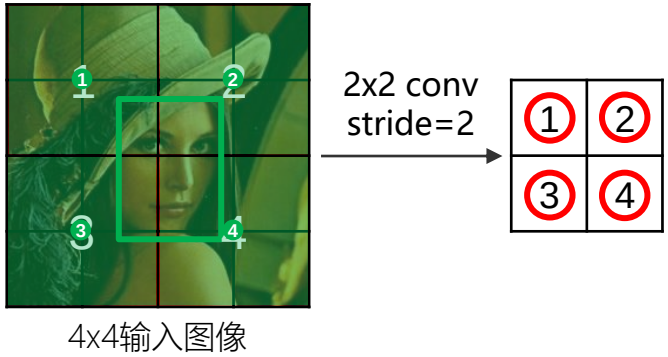


4x4输入图像

2x2 conv
stride=2

1	2
3	4

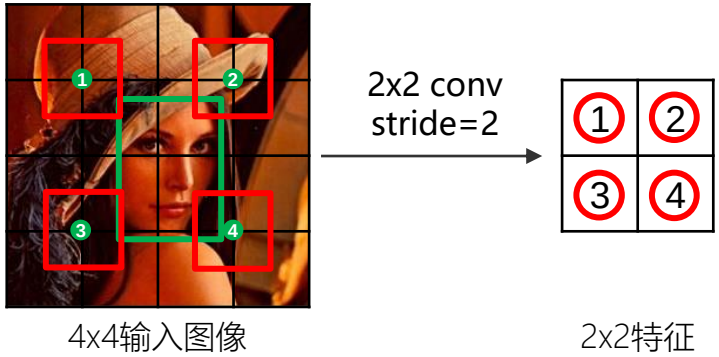
Faster R-CNN算法之RPN的原理和流程



4x4输入图像

- 理论感受野：感受野的大小、感受野的中心

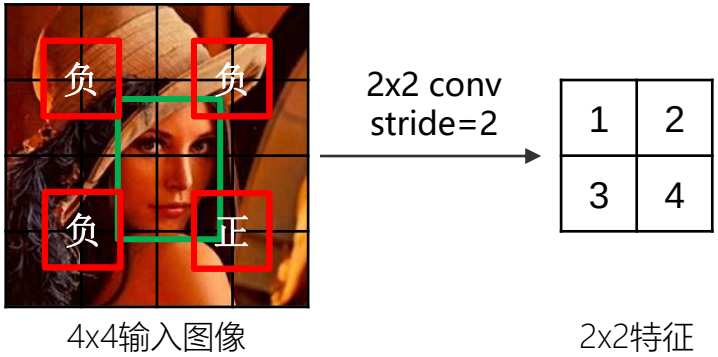
Faster R-CNN算法之RPN的原理和流程



4x4输入图像　　2x2特征

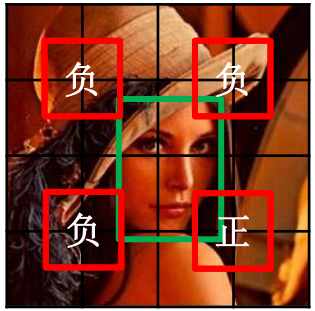
- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔

Faster R-CNN算法之RPN的原理和流程



- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔
- 锚框的匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负

Faster R-CNN算法之RPN的原理和流程



4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

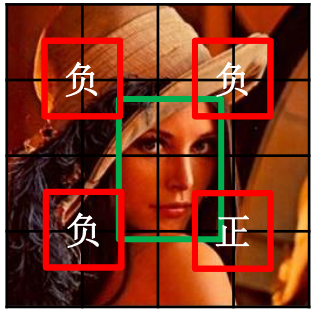
$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

Softmax
Loss

负	负
负	正

- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔
- 锚框的匹配：①正样本：最佳匹配或$IoU \geq \theta_+$；②负样本：$IoU < \theta_-$；③忽略样本：非正非负
- 锚框的分类：正样本和负样本 +二分类Softmax损失函数

Faster R-CNN算法之RPN的原理和流程



4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

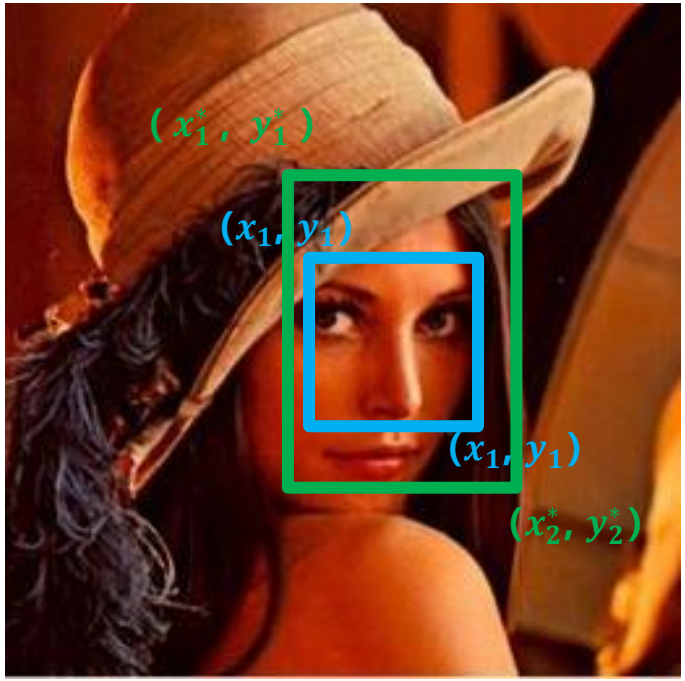
Softmax
Loss

负	负
负	正

- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔
- 锚框的匹配：①正样本：最佳匹配或$IoU \geq \theta_+$；②负样本：$IoU < \theta_-$；③忽略样本：非正非负
- 锚框的分类：正样本和负样本 +二分类Softmax损失函数
- 锚框的回归：正样本 + SmoothL1损失函数

Faster R-CNN算法之RPN的原理和流程

- 锚框的回归目标值的计算

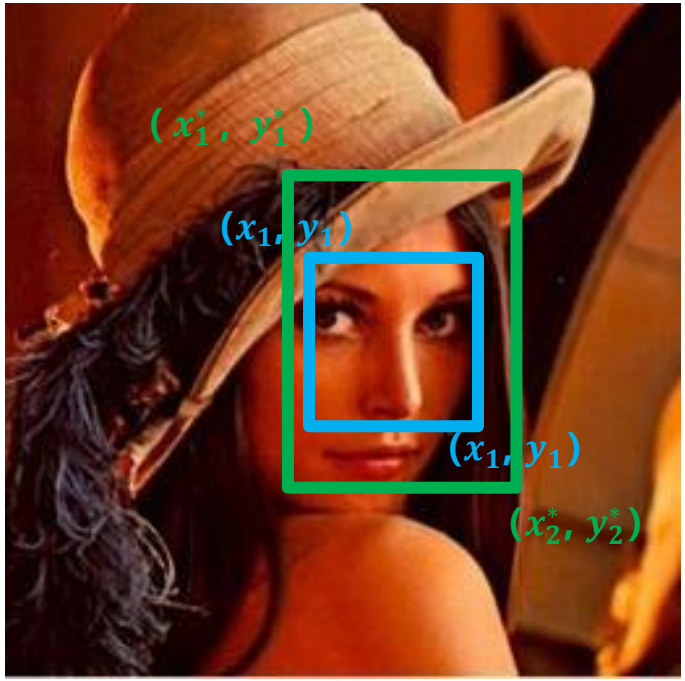


真实标注框的位置和大小

$$x_c^* = \frac{x_1^* + x_2^*}{2}$$
$$y_c^* = \frac{y_1^* + y_2^*}{2}$$
$$w^* = x_2^* - x_1^* + 1$$
$$h^* = y_2^* - y_1^* + 1$$

Faster R-CNN算法之RPN的原理和流程

- 锚框的回归目标值的计算



真实标注框的位置和大小

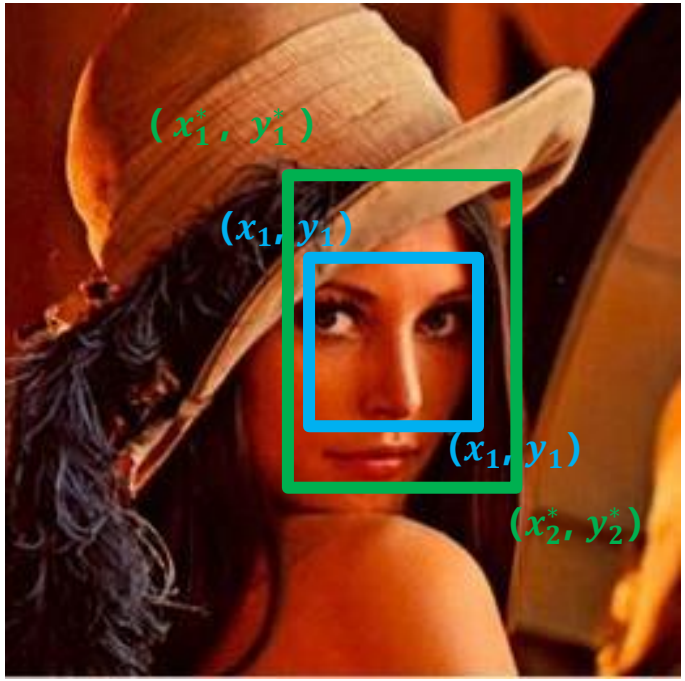
$$x_c^* = \frac{x_1^* + x_2^*}{2}$$
$$y_c^* = \frac{y_1^* + y_2^*}{2}$$
$$w^* = x_2^* - x_1^* + 1$$
$$h^* = y_2^* - y_1^* + 1$$

锚框的位置和大小

$$x_c = \frac{x_1 + x_2}{2}$$
$$y_c = \frac{y_1 + y_2}{2}$$
$$w = x_2 - x_1 + 1$$
$$h = y_2 - y_1 + 1$$

Faster R-CNN算法之RPN的原理和流程

- 锚框的回归目标值的计算



真实标注框的位置和大小

$$x_c^* = \frac{x_1^* + x_2^*}{2}$$
$$y_c^* = \frac{y_1^* + y_2^*}{2}$$
$$w^* = x_2^* - x_1^* + 1$$
$$h^* = y_2^* - y_1^* + 1$$

锚框的位置和大小

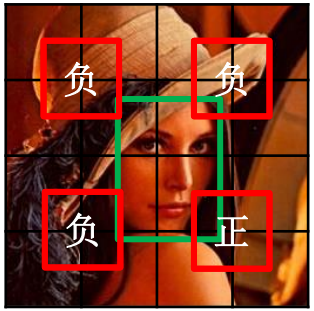
$$x_c = \frac{x_1 + x_2}{2}$$
$$y_c = \frac{y_1 + y_2}{2}$$
$$w = x_2 - x_1 + 1$$
$$h = y_2 - y_1 + 1$$

锚框的回归目标值

$$\Delta x_c^* = \frac{x_c - x_c^*}{w}, \qquad \Delta y_c^* = \frac{y_c - y_c^*}{h}$$

$$\Delta w^* = ln\frac{w^*}{w}, \qquad \Delta h^* = ln\frac{h^*}{h}$$

Faster R-CNN算法之RPN的原理和流程



4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

Softmax
Loss

负	负
负	正

回归
1x1 conv
channel=4

$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

SmoothL1
Loss

$1^*_{\triangle x}$	$2^*_{\triangle x}$	$1^*_{\triangle y}$	$2^*_{\triangle y}$	$1^*_{\triangle w}$	$2^*_{\triangle w}$	$1^*_{\triangle h}$	$2^*_{\triangle h}$
$3^*_{\triangle x}$	$4^*_{\triangle x}$	$3^*_{\triangle y}$	$4^*_{\triangle y}$	$3^*_{\triangle w}$	$4^*_{\triangle w}$	$3^*_{\triangle h}$	$4^*_{\triangle h}$

- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔
- 锚框的匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 锚框的分类：正样本和负样本 +二分类Softmax损失函数
- 锚框的回归：正样本 + SmoothL1损失函数

Faster R-CNN算法之RPN的原理和流程

- Softmax Loss的形式：

$$f(z_k) = e^{z_k} / (\sum_j e^{z_j})$$

$$l(y, z) = -\sum_{k=0}^{C} y_c log(f(z_c))$$

- SmoothL1 Loss的形式：

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2\sigma^2, if\ |x| < \frac{1}{\sigma^2} \\ |x| - \frac{0.5}{\sigma^2}, otherwise \end{cases} \xrightarrow{\sigma=1} Smooth_{L_1}(x) = \begin{cases} 0.5x^2, if\ |x| < 1 \\ |x| - 0.5, otherwise \end{cases}$$

Faster R-CNN算法之RPN的原理和流程

4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=4

1背景	2背景	1前景	2前景	1背景	2背景	1前景	2前景
3背景	4背景	3前景	4前景	3前景	4前景	3前景	4前景

第一层锚框　　第二层锚框

回归
1x1 conv
channel=8

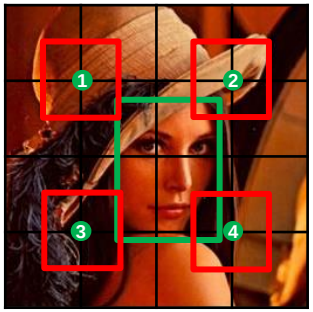
第一层锚框

$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$
$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

第二层锚框

- 理论感受野：感受野的大小、感受野的中心
- 锚框的设计：锚框的大小、锚框的长宽比、锚框的铺设间隔
- 锚框的匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 锚框的分类：正样本和负样本 +二分类Softmax损失函数
- 锚框的回归：正样本 + SmoothL1损失函数

Faster R-CNN算法之RPN的原理和流程

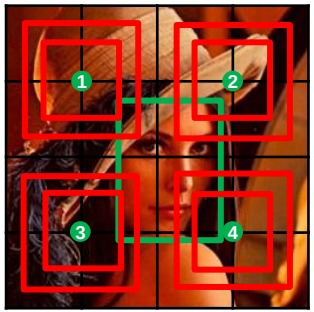


4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征



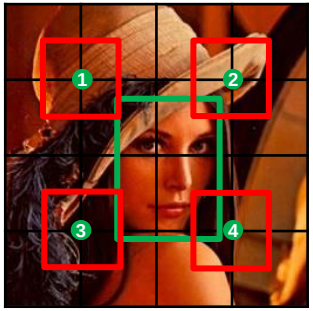
4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

Faster R-CNN算法之RPN的原理和流程

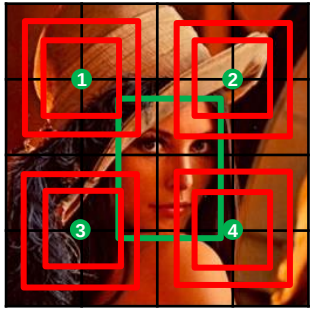


4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

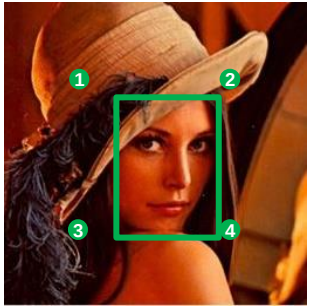


4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征



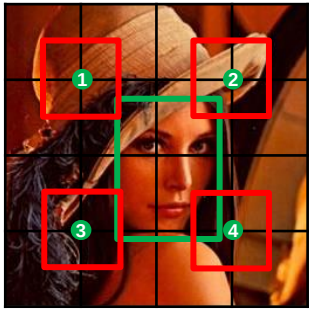
32x32输入图像

VGG16/ResNet101
total stride=**16**

1	2
3	4

2x2特征

Faster R-CNN算法之RPN的原理和流程

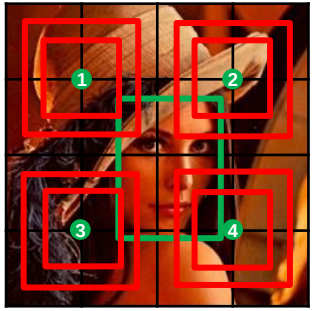


4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

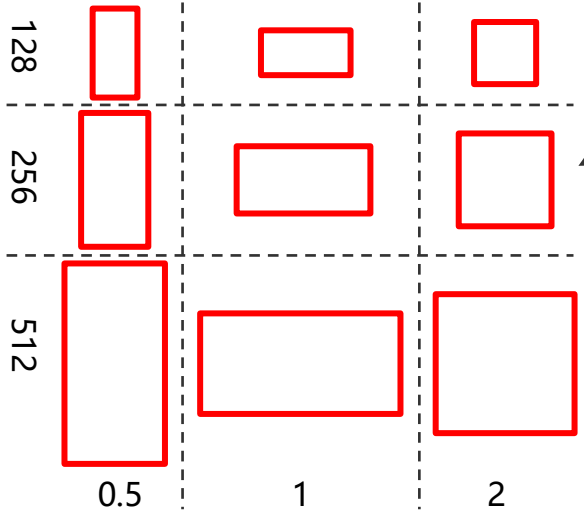


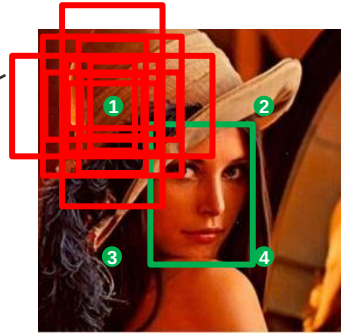
4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征





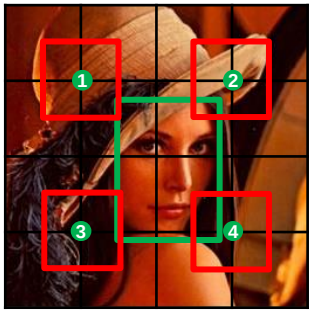
32x32输入图像

VGG16/ResNet101
total stride=**16**

1	2
3	4

2x2特征

Faster R-CNN算法之RPN的原理和流程

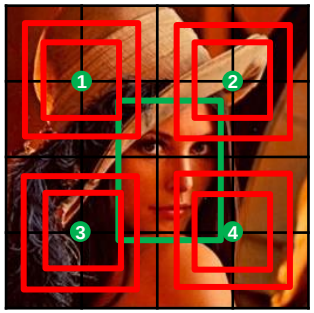


4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

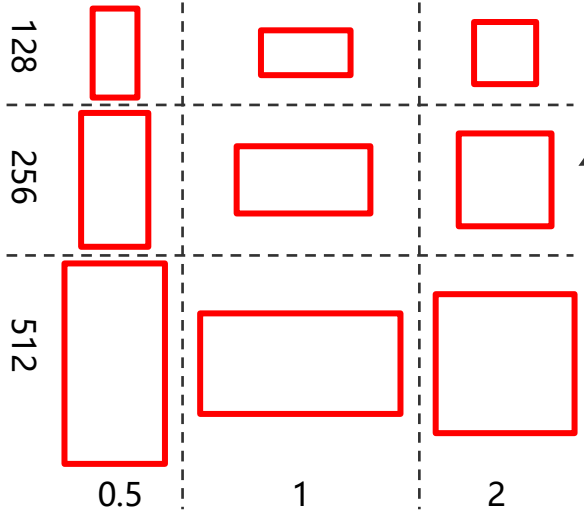


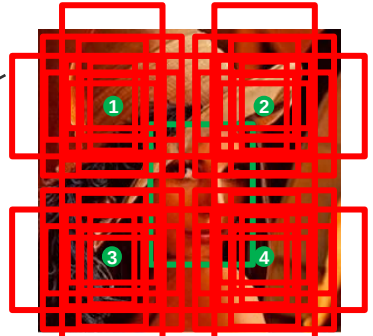
4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征





32x32输入图像

VGG16/ResNet101
total stride=**16**

1	2
3	4

2x2特征

Faster R-CNN算法之RPN生成候选区域

4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

回归
1x1 conv
channel=4

$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

Faster R-CNN算法之RPN生成候选区域

4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

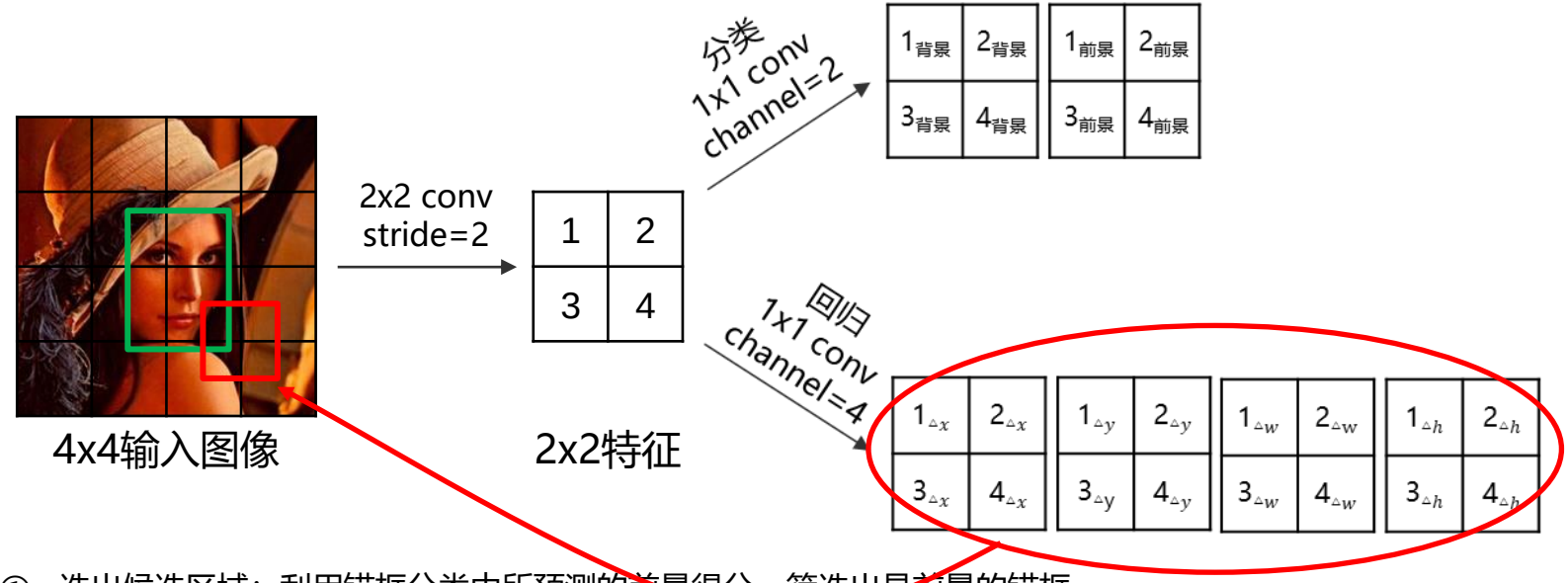
$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

回归
1x1 conv
channel=4

$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

① 选出候选区域：利用锚框分类中所预测的前景得分，筛选出是前景的锚框

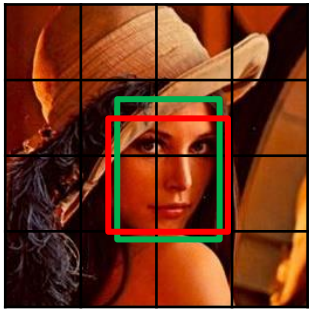
Faster R-CNN算法之RPN生成候选区域



① 选出候选区域：利用锚框分类中所预测的前景得分，筛选出是前景的锚框

② 调整候选区域：利用锚框回归中所预测的偏移值，调整前景锚框的位置和长宽

Faster R-CNN算法之RPN生成候选区域



4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

回归
1x1 conv
channel=4

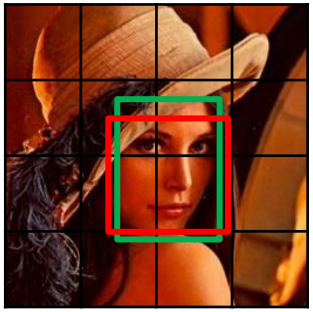
$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

① 选出候选区域：利用锚框分类中所预测的前景得分，筛选出是前景的锚框

② 调整候选区域：利用锚框回归中所预测的偏移值，调整前景锚框的位置和长宽

③ 最终候选区域：利用非极大值抑制（NMS）去掉冗余的候选区域，输出最终的候选区域

Faster R-CNN算法之RPN生成候选区域



4x4输入图像

2x2 conv
stride=2

1	2
3	4

2x2特征

分类
1x1 conv
channel=2

$1_{背景}$	$2_{背景}$	$1_{前景}$	$2_{前景}$
$3_{背景}$	$4_{背景}$	$3_{前景}$	$4_{前景}$

回归
1x1 conv
channel=4

$1_{\triangle x}$	$2_{\triangle x}$	$1_{\triangle y}$	$2_{\triangle y}$	$1_{\triangle w}$	$2_{\triangle w}$	$1_{\triangle h}$	$2_{\triangle h}$
$3_{\triangle x}$	$4_{\triangle x}$	$3_{\triangle y}$	$4_{\triangle y}$	$3_{\triangle w}$	$4_{\triangle w}$	$3_{\triangle h}$	$4_{\triangle h}$

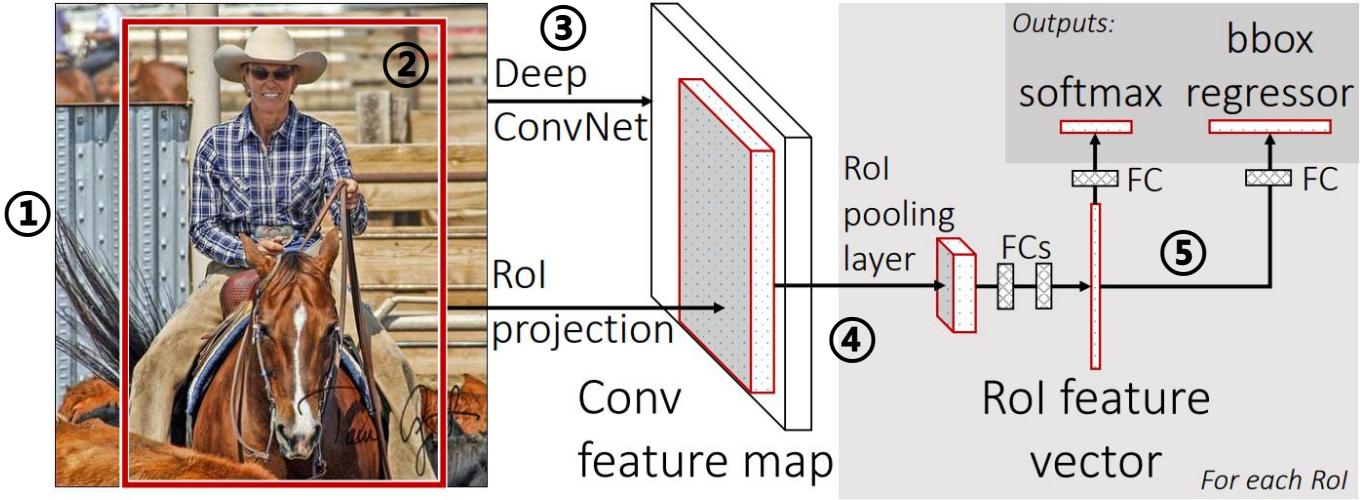
① 选出候选区域：利用锚框分类中所预测的前景得分，筛选出是前景的锚框

② 调整候选区域：利用锚框回归中所预测的偏移值，调整前景锚框的位置和长宽

③ 最终候选区域：利用非极大值抑制（NMS）去掉冗余的候选区域，输出最终的候选区域

④ 增强候选区域（**仅训练时**）：把真实标注框也当做候选区域，放入候选区域子集中去

Faster R-CNN算法之Fast R-CNN原理和流程



Fast R-CNN的检测步骤：

① 输入图像：输入一张待检测的图像

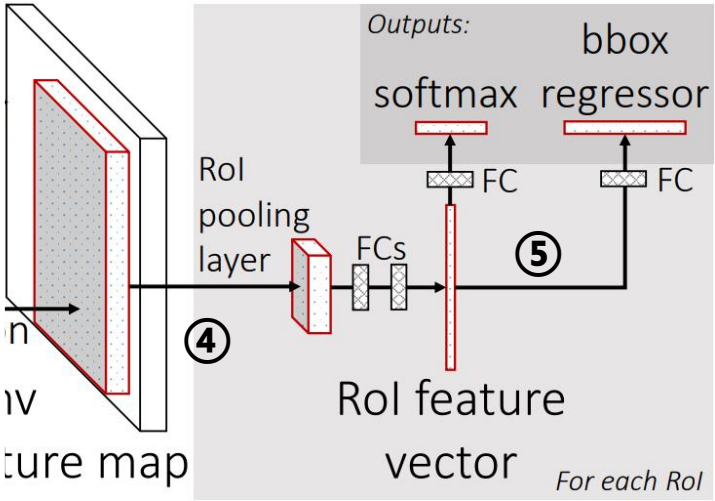
② 候选区域生成：使用Selective Search算法，在输入图像上生成~2K个候选区域

③ 特征提取：将整张图像传入CNN提取特征

④ 候选区域特征：利用RoIPooling生成每个候选区域的特征

⑤ 候选区域分类和回归：利用扣取的特征，对每个候选区域进行分类和回归

Faster R-CNN算法之Fast R-CNN原理和流程



Fast R-CNN的检测步骤：

① 输入图像：输入一张待检测的图像

② 候选区域生成：使用Selective Search算法，在输入图像上生成~2K个候选区域

③ 特征提取：将整张图像传入CNN提取特征

④ 候选区域特征：利用RoIPooling生成每个候选区域的特征

⑤ 候选区域分类和回归：利用扣取的特征，对每个候选区域进行分类和回归

Faster R-CNN算法之Fast R-CNN原理和流程

128x128输入图像

8x8特征

Faster R-CNN算法之Fast R-CNN原理和流程

128x128输入图像　　8x8特征

- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负

Faster R-CNN算法之Fast R-CNN原理和流程

128x128输入图像　　8x8特征

- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去

Faster R-CNN算法之Fast R-CNN原理和流程

128x128输入图像　　8x8特征

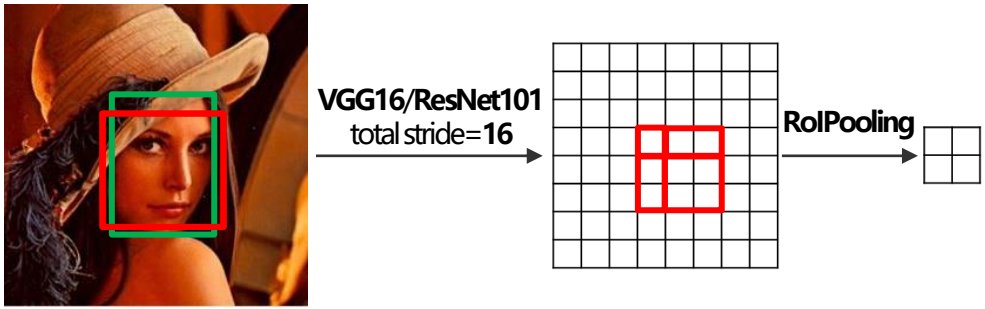
- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2

Faster R-CNN算法之Fast R-CNN原理和流程

128x128输入图像　　8x8特征

- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2
- 子区域取整调整：子区域的位置不为整数的，根据四舍五入的规则进行取整

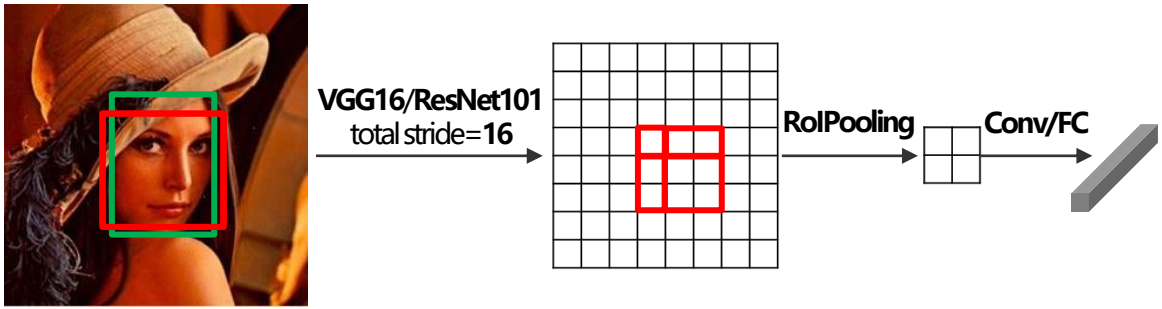
Faster R-CNN算法之Fast R-CNN原理和流程



128x128输入图像　　8x8特征

- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2
- 子区域取整调整：子区域的位置不为整数的，根据四舍五入的规则进行取整
- 子区域取最大值：在每个子区域里取最大值，生成2x2的特征

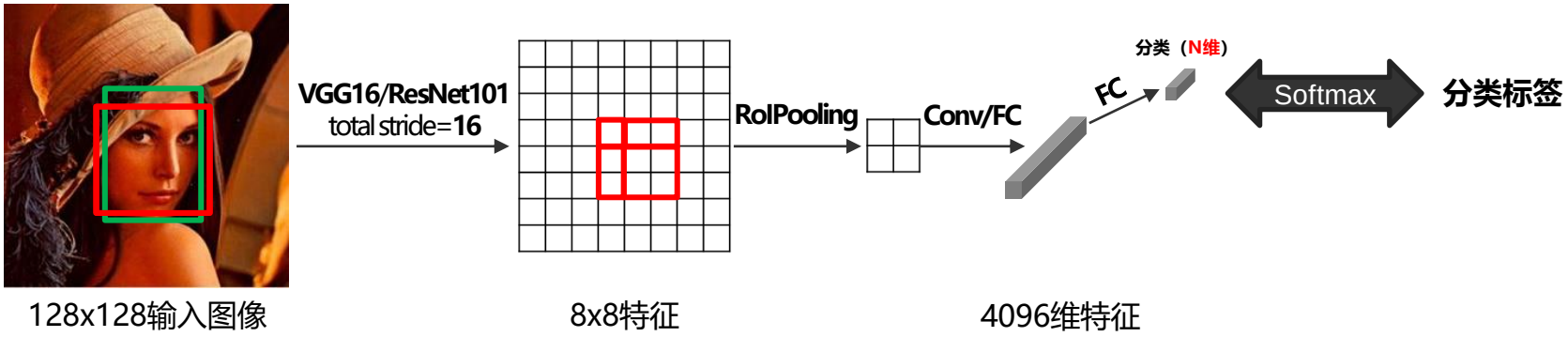
Faster R-CNN算法之Fast R-CNN原理和流程



128x128输入图像　　8x8特征　　4096维特征

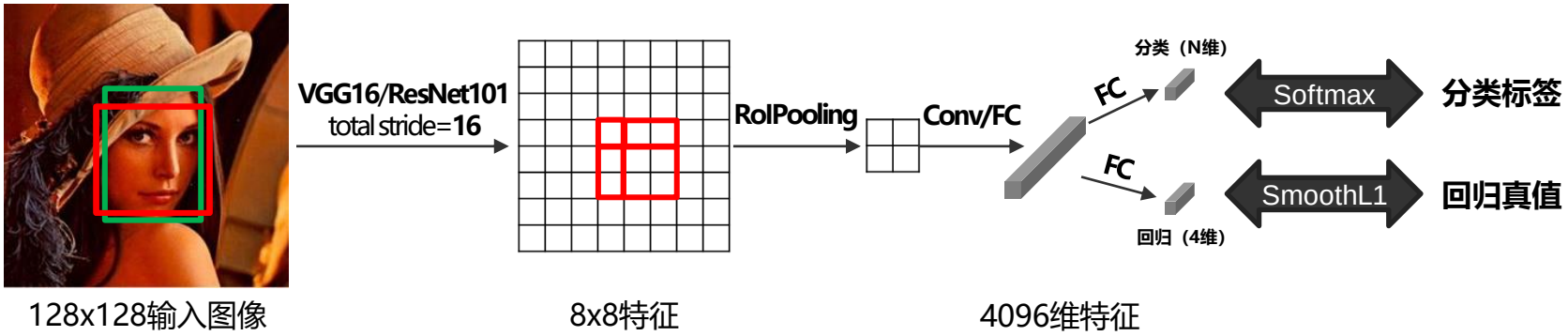
- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2
- 子区域取整调整：子区域的位置不为整数的，根据四舍五入的规则进行取整
- 子区域取最大值：在每个子区域里取最大值，生成2x2的特征
- 候选区域特征：把2x2的特征输入到Conv或FC子网络，以输出候选区域的最终特征

Faster R-CNN算法之Fast R-CNN原理和流程



- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2
- 子区域取整调整：子区域的位置不为整数的，根据四舍五入的规则进行取整
- 子区域取最大值：在每个子区域里取最大值，生成2x2的特征
- 候选区域特征：把2x2的特征输入到Conv或FC子网络，以输出候选区域的最终特征
- 候选区域分类：正样本和负样本 + 多分类Softmax损失函数

Faster R-CNN算法之Fast R-CNN原理和流程



- 候选区域匹配：①正样本：最佳匹配或IoU≥θ_+；②负样本：IoU < θ_-；③忽略样本：非正非负
- 候选区域映射：根据下采样倍数，把候选区域映射到特征上去
- 候选区域分块：把候选区域等分成指定大小的子区域，例如2x2
- 子区域取整调整：子区域的位置不为整数的，根据四舍五入的规则进行取整
- 子区域取最大值：在每个子区域里取最大值，生成2x2的特征
- 候选区域特征：把2x2的特征输入到Conv或FC子网络，以输出候选区域的最终特征
- 候选区域分类：正样本和负样本 + 多分类Softmax损失函数
- 候选区域回归：正样本 + SmoothL1损失函数

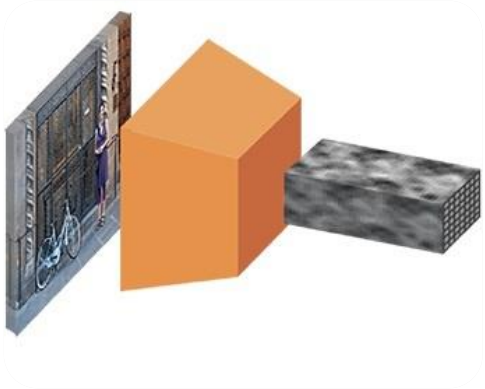
Faster R-CNN算法的整体流程

Faster R-CNN算法的整体流程

Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

Faster R-CNN算法的整体流程

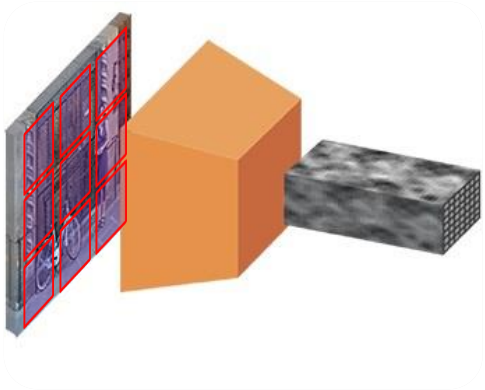


Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

Faster R-CNN算法的整体流程



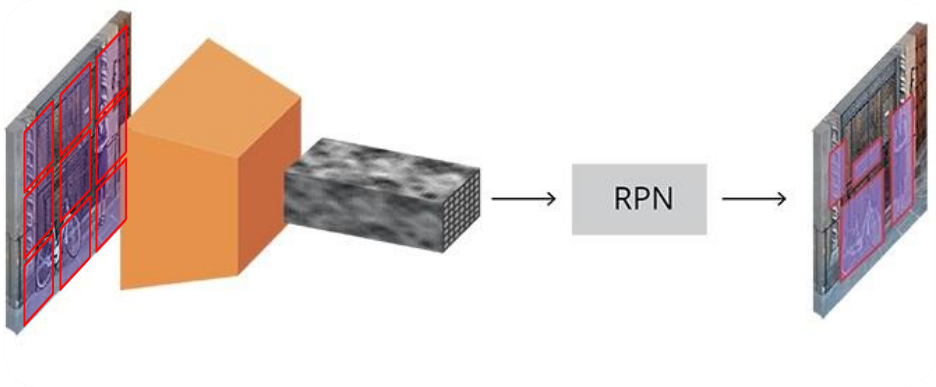
Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

③ 根据检测层预设一系列大小和比例的锚框（9个）

Faster R-CNN算法的整体流程



Faster R-CNN中RPN步骤：

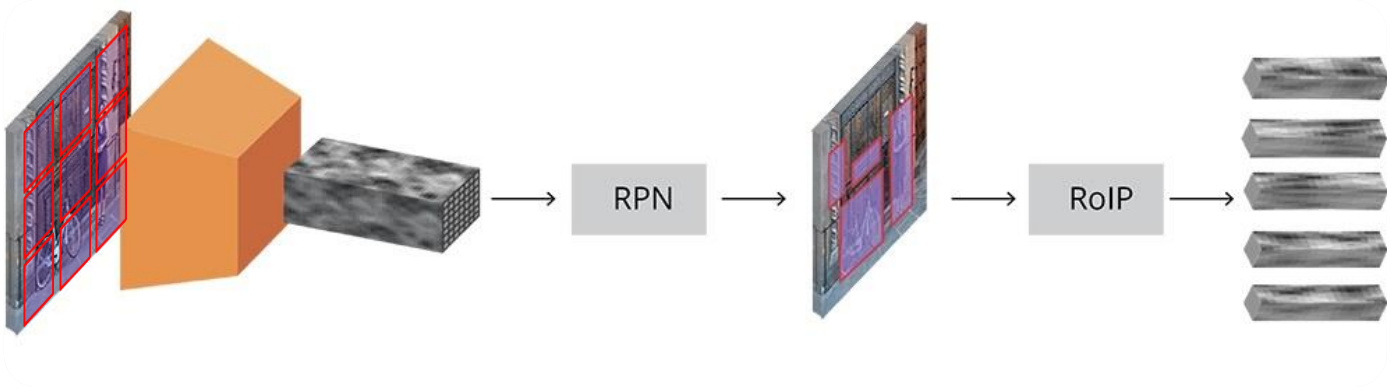
① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

③ 根据检测层预设一系列大小和比例的锚框（9个）

④ 对锚框进行二分类和回归得到若干候选区域

Faster R-CNN算法的整体流程



Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

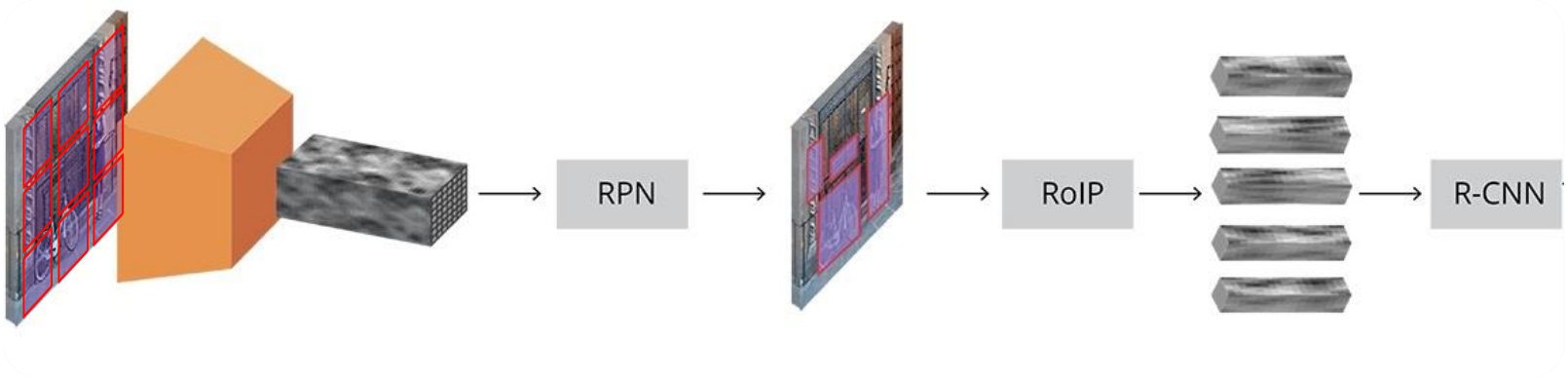
③ 根据检测层预设一系列大小和比例的锚框（9个）

④ 对锚框进行二分类和回归得到若干候选区域

Faster R-CNN中Fast R-CNN步骤：

① 利用RoIPooling在检测层的特征上提取每个候选区域对应的特征

Faster R-CNN算法的整体流程



Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

③ 根据检测层预设一系列大小和比例的锚框（9个）

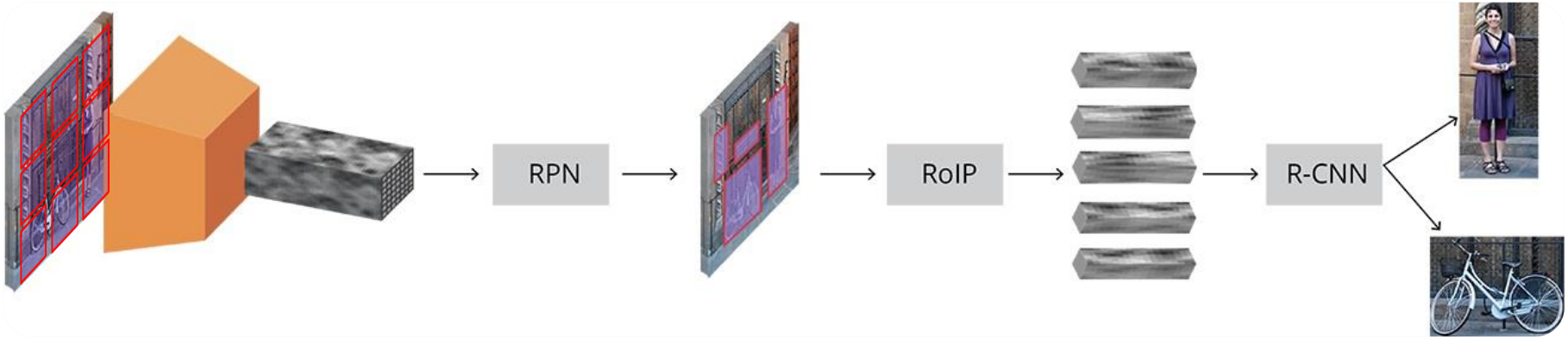
④ 对锚框进行二分类和回归得到若干候选区域

Faster R-CNN中Fast R-CNN步骤：

① 利用RoIPooling在检测层的特征上提取每个候选区域对应的特征

② 输入CNN/FC子网络来增强候选区域的特征

Faster R-CNN算法的整体流程



Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

③ 根据检测层预设一系列大小和比例的锚框（9个）

④ 对锚框进行二分类和回归得到若干候选区域

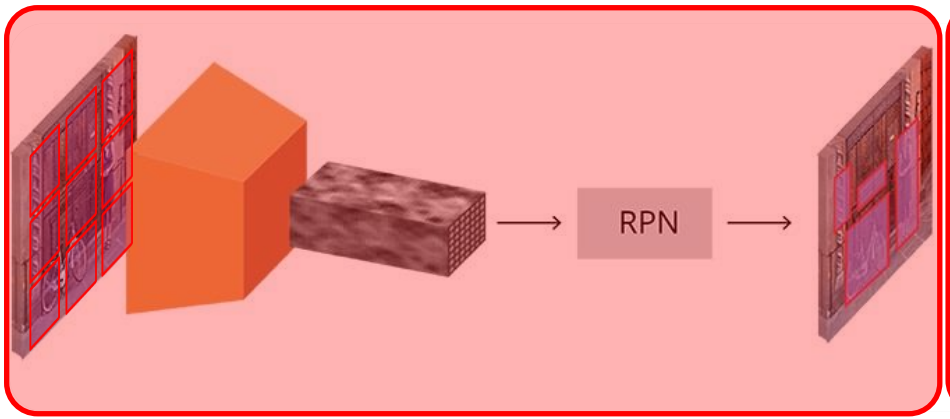
Faster R-CNN中Fast R-CNN步骤：

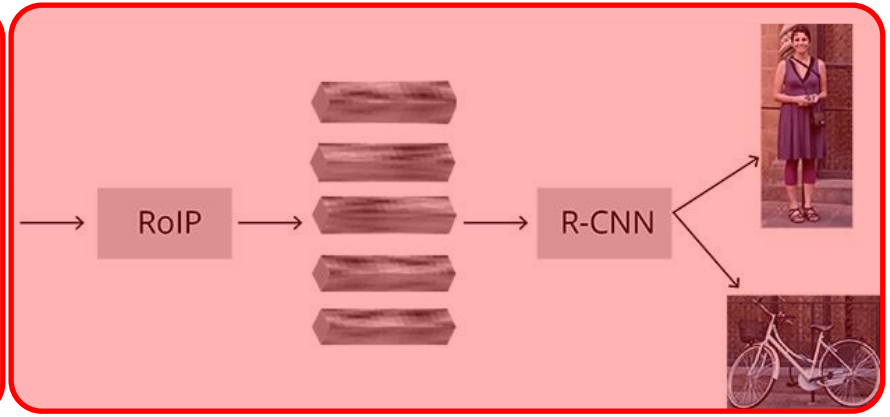
① 利用RoIPooling在检测层的特征上提取每个候选区域对应的特征

② 输入CNN/FC子网络来增强候选区域的特征

③ 对候选区域进行多分类和回归得到检测结果

Faster R-CNN算法的整体流程





Faster R-CNN中RPN步骤：

① 整张图传入VGG16或ResNet提取特征

② 选择下采样倍数为16的特征层作为检测层

③ 根据检测层预设一系列大小和比例的锚框（9个）

④ 对锚框进行二分类和回归得到若干候选区域

Faster R-CNN中Fast R-CNN步骤：

① 利用RoIPooling在检测层的特征上提取每个候选区域对应的特征

② 输入CNN/FC子网络来增强候选区域的特征

③ 对候选区域进行多分类和回归得到检测结果

R-CNN系列算法总结

算法名称	R-CNN	Fast R-CNN	Faster R-CNN
候选区域	Selective Search	Selective Search	RPN
分类方式	SVM	SoftmaxLoss	SoftmaxLoss
回归方式	边缘提取	SmoothL1Loss	SmoothL1Loss
重复计算	非常多	较多	比较少
训练方式	每个步骤都是独立训练	除候选区域生成外，其他步骤是端到端训练	端到端训练
检测速度	~0.002 FPS	~0.5 FPS	~5 FPS