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

目录

[整体框架 2](#_Toc5955886)

[网络结构 3](#_Toc5955887)

[1.Conv layers 3](#_Toc5955888)

[2.RPN(Region Proposal Networks) 4](#_Toc5955889)

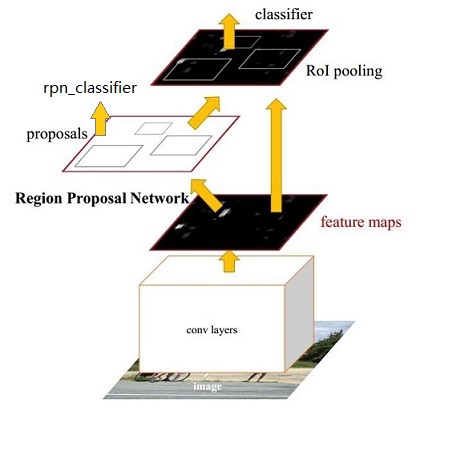
[2.1 Anchors的生成规则 4](#_Toc5955890)

[2.2.RPN工作原理解析 6](#_Toc5955891)

[3.ROI Pooling 11](#_Toc5955892)

[4全连接层 11](#_Toc5955893)

# **整体框架**

****

1)Conv layers提取特征图

作为一种CNN网络目标检测方法，Faster RCNN首先使用一组基础的conv + Relu + pooling层提取input image的feature maps,该feature maps会用于后续的RPN层和全连接层

2)RPN(Region Proposal Networks)

RPN网络主要用于生成region proposals，首先生成一堆Anchor box，对其进行裁剪过滤后通过softmax判断anchors属于前景(foreground)或者后景(background)，即是物体or不是物体，所以这是一个二分类；同时，另一分支bounding box regression修正anchor box，形成较精确的proposal（注：这里的较精确是相对于后面全连接层的再一次box regression而言）

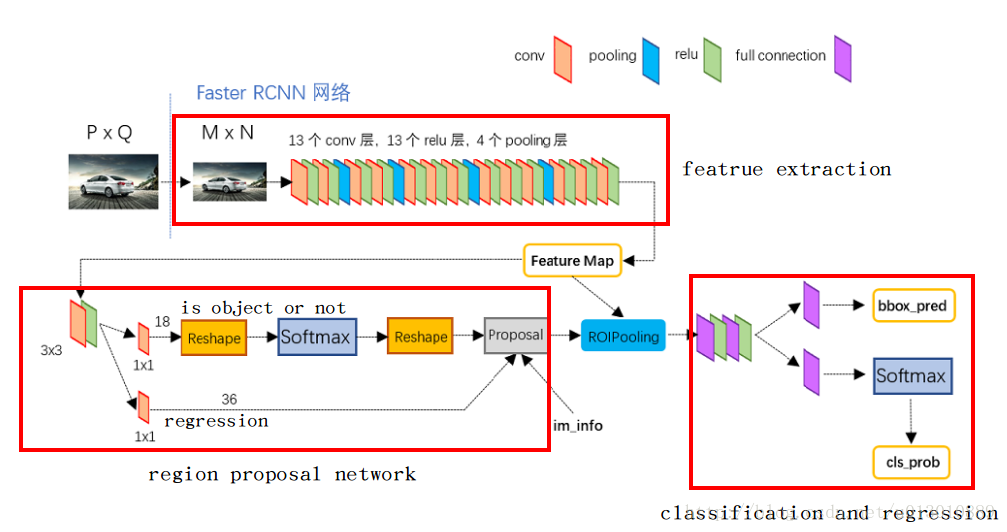
3)Roi Pooling

该层利用RPN生成的proposals和VGG16最后一层得到的feature map，得到固定大小的proposal feature map,进入到后面可利用全连接操作来进行目标识别和定位

4)Classifier

会将Roi Pooling层形成固定大小的feature map进行全连接操作，利用Softmax进行具体类别的分类，同时，利用L1 Loss完成bounding box regression回归操作获得物体的精确位置.

# **网络结构**

****

通过上图开始逐层分析

## 1.Conv layers

Faster RCNN首先是支持输入任意大小的图片的，比如上图中输入的P\*Q，进入网络之前对图片进行了规整化尺度的设定，如可设定图像短边不超过600，图像长边不超过1000，我们可以假定M\*N=1000\*600（如果图片少于该尺寸，可以边缘补0，即图像会有黑色边缘）

①   13个conv层：kernel\_size=3,pad=1,stride=1;

卷积公式：https://images2018.cnblogs.com/blog/75922/201803/75922-20180306112055150-29301770.png

             所以，conv层不会改变图片大小（即：输入的图片大小=输出的图片大小）

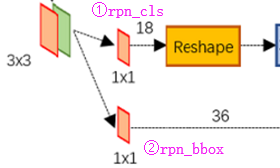
②   13个relu层：激活函数，不改变图片大小

③   4个pooling层：kernel\_size=2,stride=2;pooling层会让输出图片是输入图片的1/2

  经过Conv layers，图片大小变成(M/16)\*(N/16)，即：60\*40(1000/16≈60,600/16≈40)；则，Feature Map就是60\*40\*512-d(注：VGG16是512-d,ZF是256-d)，表示特征图的大小为60\*40，数量为512

## 2.RPN(Region Proposal Networks)

Feature Map进入RPN后，先经过一次3\*3的卷积，同样，特征图大小依然是60\*40,数量512，这样做的目的应该是进一步集中特征信息，接着看到两个全卷积,即kernel\_size=1\*1,p=0,stride=1;



如上图中标识：

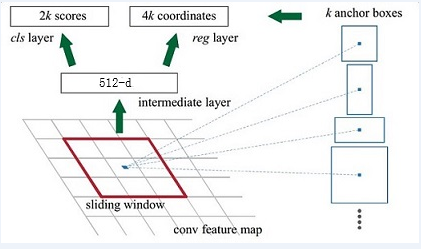
① rpn\_cls：60\*40\*512-d ⊕ 1\*1\*512\*18 ==> 60\*40\*9\*2

逐像素对其9个Anchor box进行二分类

② rpn\_bbox：60\*40\*512-d ⊕ 1\*1\*512\*36==>60\*40\*9\*4

逐像素得到其9个Anchor box四个坐标信息（其实是偏移量，后面介绍）

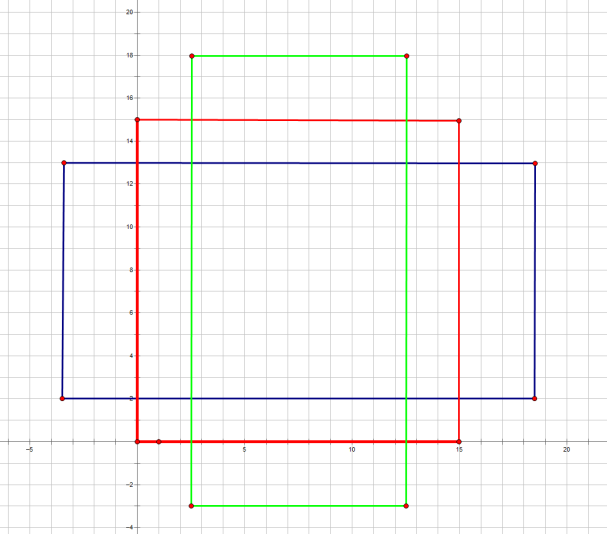
如下图所示：



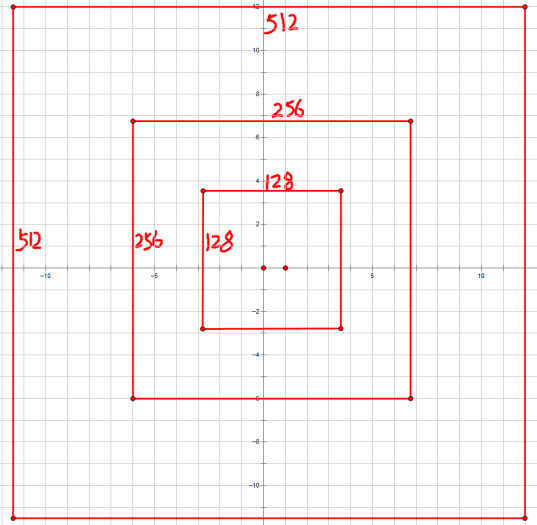
### 2.1 Anchors的生成规则

      前面提到经过Conv layers后，图片大小变成了原来的1/16，令feat\_stride=16，在生成Anchors时，我们先定义一个base\_anchor，大小为16\*16的box(因为特征图(60\*40)上的一个点，可以对应到原图（1000\*600）上一个16\*16大小的区域)，源码中转化为[0,0,15,15]的数组，参数ratios=[0.5, 1, 2]scales=[8, 16, 32]

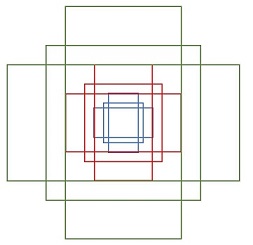
   先看[0,0,15,15],面积保持不变，长、宽比分别为[0.5, 1, 2]是产生的Anchors box



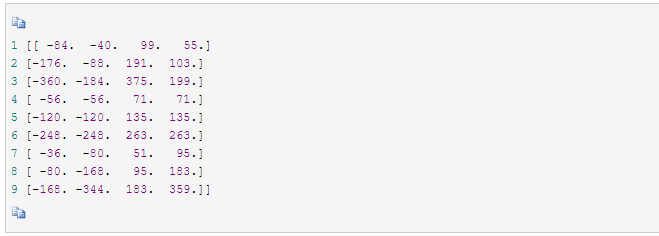
如果经过scales变化，即长、宽分别均为 (16\*8=128)、(16\*16=256)、(16\*32=512)，对应anchor box如图



综合以上两种变换，最后生成9个Anchor box



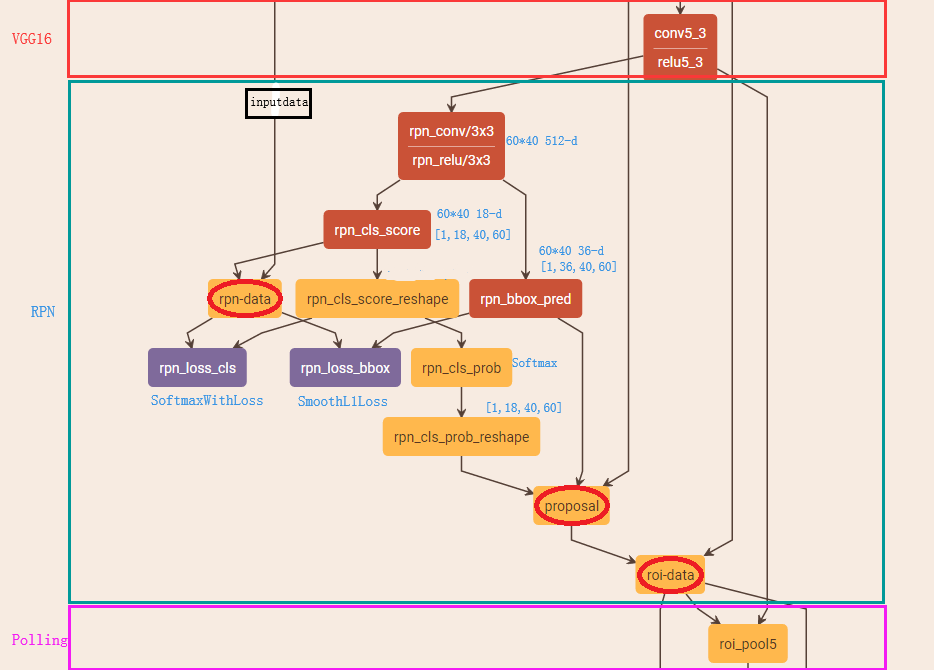
所以，最终base\_anchor=[0,0,15,15]生成的9个Anchor box坐标如下：



**特征图大小为60\*40，所以会一共生成60\*40\*9=21600个Anchor box(分别以feature map的每一个点为中心，在原图上画出9个尺寸不一anchor。)**

  源码中，通过width:(0~60)\*16,height(0~40)\*16建立shift偏移量数组，再和base\_anchor基准坐标数组累加，得到特征图上所有像素对应的Anchors的坐标值，是一个[216000,4]的数组

### 2.2.RPN工作原理解析

为了进一步更清楚的看懂RPN的工作原理，将Caffe版本下的网络图贴出来，对照网络图进行讲解会更清楚

主要看上图中框住的‘RPN’部分的网络图，其中‘rpn\_conv/3\*3’是3\*3的卷积，上面有提到过，接着是两个1\*1的全卷积，分别是图中的‘rpn\_cls\_score’和‘rpn\_bbox\_pred’，在上面同样有提到过。接下来，分析网络图中其他各部分的含义

#### 2.2.1.RPN-data



这一层主要是为特征图60\*40上的每个像素生成9个Anchor box，并且对生成的Anchor box进行过滤和标记，参照源码，过滤和标记规则如下：

①    去除掉超过1000\*600这原图的边界的anchor box

②    如果anchor box与ground truth的值最大，标记为正样本，label=1

③    如果anchor box与ground truth的>0.7，标记为正样本，label=1

④    如果anchor box与ground truth的<0.3，标记为负样本，label=0

     剩下的既不是正样本也不是负样本，不用于最终训练，label=-1

除了对anchor box进行标记外，另一件事情就是计算anchor box与ground truth之间的偏移量

   令：ground truth:标定的框也对应一个中心点位置坐标和宽高

    anchor box: 中心点位置坐标和宽高

    所以，偏移量：

通过ground truth box与预测的anchor box之间的差异来进行学习，从而是RPN网络中的权重能够学习到预测box的能力.

#### 2.2.2**rpn\_loss\_cls、rpn\_loss\_bbox、rpn\_cls\_prob**

下面集体看下这三个，其中‘rpn\_loss\_cls’、‘rpn\_loss\_bbox’是分别对应softmax，smooth L1计算损失函数，‘rpn\_cls\_prob’计算概率值(可用于下一层的nms非最大值抑制操作)

补充：

     ①   Softmax公式，https://images2018.cnblogs.com/blog/75922/201803/75922-20180306113408600-2089053418.png计算各分类的概率值

      ② Softmax Loss公式，https://images2018.cnblogs.com/blog/75922/201803/75922-20180306113423457-935709096.pngRPN进行分类时，即寻找最小Loss值

在’rpn-data’中已经为预测框anchor box进行了标记，并且计算出与gt\_boxes之间的偏移量,利用RPN网络进行训练。

RPN训练设置：在训练RPN时，一个Mini-batch是由一幅图像中任意选取的256个proposal组成的，其中正负样本的比例为1：1。如果正样本不足128，则多用一些负样本以满足有256个Proposal可以用于训练，反之亦然

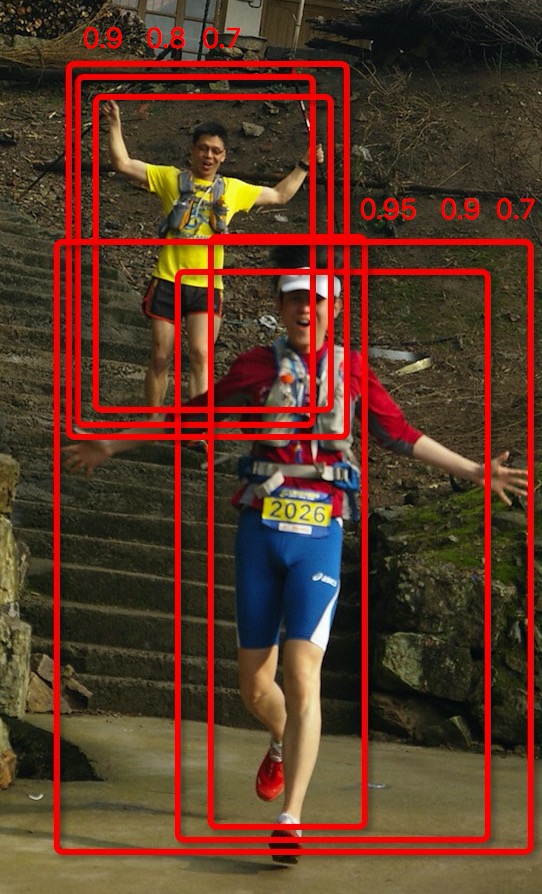
#### ****2.2.3.proposal****



在输入中我们看到’rpn\_bbox\_pred’，记录着训练好的四个回归值△x, △y, △w, △h。

源码中，会重新生成60\*40\*9个anchor box，然后累加上训练好的△x, △y, △w, △h,从而得到了相较于之前更加准确的预测框region proposal，进一步对预测框进行越界剔除和使用nms非最大值抑制，剔除掉重叠的框；比如，设定为0.7的阈值，即仅保留覆盖率不超过0.7的局部最大分数的box（粗筛）。最后留下大约2000个anchor，然后再取前N个box（比如300个）；这样，进入到下一层ROI Pooling时region proposal大约只有300个

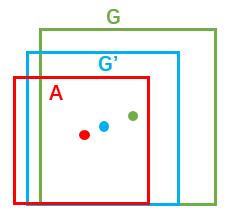
用下图一个案例来对NMS算法进行简单介绍



如上图所示，一共有6个识别为人的框，每一个框有一个置信率。   
现在需要消除多余的:

* 按置信率排序: 0.95, 0.9, 0.9, 0.8, 0.7, 0.7
* 取最大0.95的框为一个物体框
* 剩余5个框中，去掉与0.95框重叠率IoU大于0.6(可以另行设置)，则保留0.9, 0.8, 0.7三个框
* 重复上面的步骤，直到没有框了，0.9为一个框
* 选出来的为: 0.95, 0.9

所以，整个过程，可以用下图形象的表示出来



其中，红色的A框是生成的anchor box,而蓝色的G’框就是经过RPN网络训练后得到的较精确的预测框，绿色的G是ground truth box

#### 2.2.4. **roi\_data**



为了避免定义上的误解，我们将经过‘proposal’后的预测框称为region proposal（其实，RPN层的任务其实已经完成，roi\_data属于为下一层准备数据）

主要作用：

①RPN层只是来确定region proposal是否是物体(是/否),这里根据region proposal和ground truth box的最大重叠指定具体的标签(就不再是二分类问题了，参数中指定的是81类)

②计算region proposal与ground truth boxes的偏移量，计算方法和之前的偏移量计算公式相同

经过这一步后的数据输入到ROI Pooling层进行进一步的分类和定位.

## 3.**ROI Pooling**



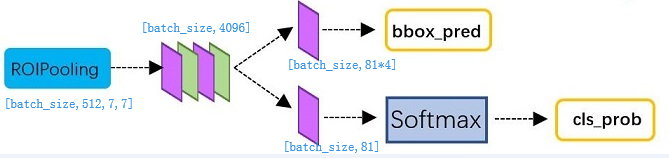
从上述的Caffe代码中可以看到，输入的是RPN层产生的region proposal(假定有300个region proposal box)和VGG16最后一层产生的特征图(60\*40 512-d)，遍历每个region proposal，将其坐标值缩小16倍，这样就可以将在原图(1000\*600)基础上产生的region proposal映射到60\*40的特征图上，从而将在feature map上确定一个区域(定义为RB\*)。

在feature map上确定的区域RB\*，根据参数pooled\_w:7,pooled\_h:7,将这个RB\*区域划分为7\*7，即49个相同大小的小区域，对于每个小区域，使用max pooling方式从中选取最大的像素点作为输出，这样，就形成了一个7\*7的feature map

以此，参照上述方法，300个region proposal遍历完后，会产生很多个7\*7大小的feature map，故而输出的数组是：[300,512,7,7],作为下一层的全连接的输入

## 4**全连接层**

经过roi pooling层之后，batch\_size=300, proposal feature map的大小是7\*7,512-d,对特征图进行全连接，参照下图，最后同样利用Softmax Loss和L1 Loss完成分类和定位



通过full connect层与softmax计算每个region proposal具体属于哪个类别（如人，马，车等），输出cls\_prob概率向量；同时再次利用bounding box regression获得每个region proposal的位置偏移量bbox\_pred，用于回归获得更加精确的目标检测框

即从PoI Pooling获取到7x7大小的proposal feature maps后，通过全连接主要做了：

1. 通过全连接和softmax对region proposals进行具体类别的分类
2. 再次对region proposals进行bounding box regression，获取更高精度的rectangle box

# 重难点

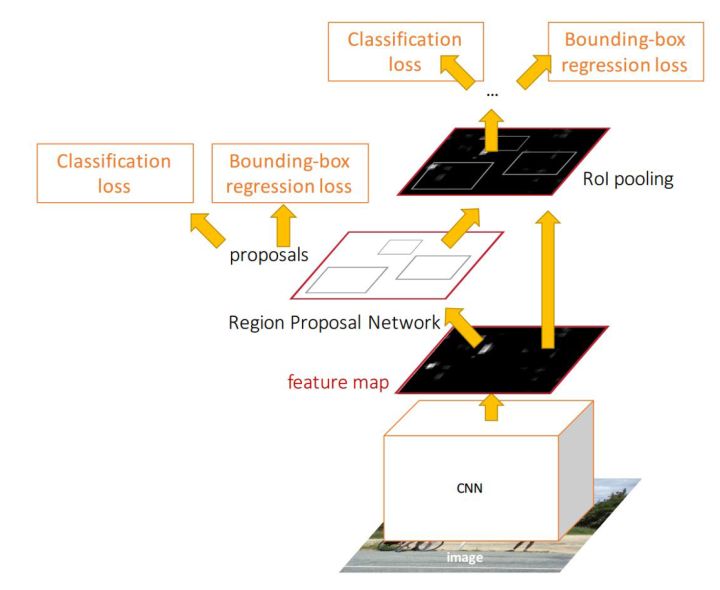
## Region proposal的作用

### 1.1 RPN介绍

首先我们明确一个定义，当前主流的Object Detection框架分为1 stage和2 stage，而2 stage多出来的这个stage就是Regional Proposal过程.

**Regional Proposal的输出到底是什么？**

以Faster R-CNN为代表的2 stage目标检测方法



图中有两个Classification loss和两个Bounding-box regression loss,那他们的区别是什么呢?

1、Input Image经过CNN特征提取，首先来到Region Proposal网络。由Regio Proposal Network输出的Classification，这并不是判定物体在COCO数据集上对应的80类中哪一类，而是输出一个Binary的值可以理解为，人工设定一个threshold=0.5。

RPN网络做的事情就是，如果一个Region的，则认为这个Region中可能是80个类别中的某一类，具体是哪一类现在还不清楚。到此为止，Network只需要把这些可能含有物体的区域选取出来就可以了，这些被选取出来的Region又叫做ROI （Region of Interests），即感兴趣的区域。当然了，**RPN同时也会在feature map上框定这些ROI感兴趣区域的大致位置，即输出Bounding-box。**

RPN网络做的事情就是，把一张图片中，我不感兴趣的区域——花花草草、大马路、天空之类的区域忽视掉，只留下一些我可能感兴趣的区域——车辆、行人、水杯、闹钟等等，然后我之后只需要关注这些感兴趣的区域，进一步确定它到底是车辆、还是行人、还是水杯（分类问题）



天空和草地都属于背景



天空和马路也都是背景

到此为止，RPN网络的工作就完成了，即我们现在得到的有：**在输入RPN网络的feature map上，所有可能包含80类物体的Region区域的信息，其他Region（非常多）我们可以直接不考虑了（不用输入后续网络）。**

接下来的工作就很简单了，假设输入RPN网络的feature map大小为64\*64，那么我们**提取的ROI的尺寸一定小于64\*64**，因为原始图像某一块的物体在feature map上也以同样的比例存在。我们只需要把这些Region从feature map上抠出来，由于每个Region的尺寸可能不一样，因为原始图像上物体大小不一样，所以**我们需要将这些抠出来的Region想办法resize到相同的尺寸**，这一步方法很多（Pooling或者Interpolation，一般采用Pooling，因为反向传播时求导方便）。

假设这些抠出来的ROI Region被我们resize到了14\*14或者7\*7，那我们接下来将这些Region输入普通的分类网络，即第一张Faster R-CNN的结构图中最上面的部分，即可得到整个网络最终的输出classification，这里的class（车、人、狗。。）才真正对应了COCO数据集80类中的具体类别。

同时，**由于我们之前RPN确定的box\region坐标比较粗略**，即大概框出了感兴趣的区域，所以这里我们再来一次精确的微调，根据每个box中的具体内容微微调整一下这个box的坐标，即输出第一张图中右上方的Bounding-box regression。

### 1.2Region Proposal有什么作用？

(1)COCO数据集上总共只有80类物体，如果不进行Region Proposal，即网络最后的classification是对所有anchor框定的Region进行识别分类，**会严重拖累网络的分类性能，难以收敛。**原因在于，存在过多的不包含任何有用的类别（80类之外的，例如各种各样的天空、草地、水泥墙、玻璃反射等等）的Region输入分类网络，而这些无用的Region占了所有Region的很大比例。换句话说，这些Region数量庞大，却并不能为softmax分类器带来有用的性能提升（因为无论怎么预测，其类别都是背景，对于主体的80类没有贡献）。

(2) 大量无用的Region都需要单独进入分类网络，而分类网络由几层卷积层和最后一层全连接层组成，**参数众多，十分耗费计算时间**，Faster R-CNN本来就不能做到实时，这下更慢了。

## 2. 原始图片中的ROI如何映射到到feature map

CNN计算公式:

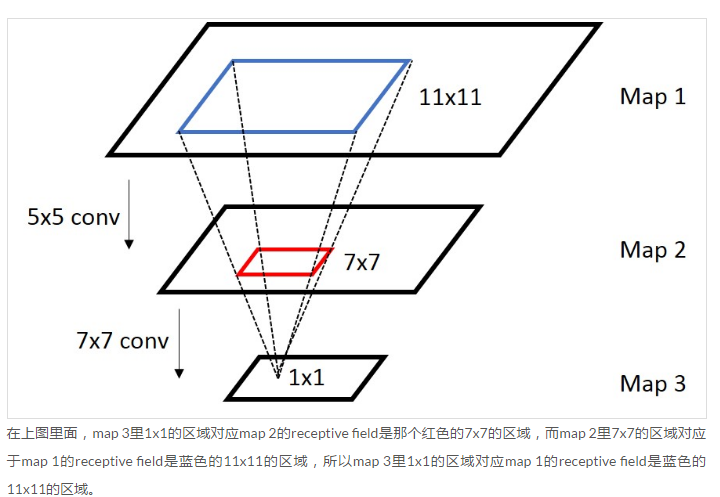
**output field size = ( input field size - kernel size + 2\*padding ) / stride + 1**

**(output field size 是卷积层的输出，input field size 是卷积层的输入)**

由此可以推出卷积层的输入

**input field size = （output field size - 1）\* stride - 2\*padding + kernel size**

卷积神经网络CNN中，某一层输出结果中一个元素所对应的输入层的区域大小，被称作感受野receptive field。感受野的大小是由kernel size，stride，padding , outputsize 一起决定的。



**公式化一下：**

(1)对于Convolution/Pooling layer:



(2)对于 Neuronlayer(ReLU/Sigmoid/..) : (显然如此)



**上面只是给出了 前一层在后一层的感受野，如何计算最后一层在原始图片上的感受野呢？ 从后向前级联一下就可以了（先计算最后一层到倒数第二层的感受野，再计算倒数第二层到倒数第三层的感受野，依次从后往前推导就可以了）**