# R-CNN系列

## R-CNN回顾总结

### 1.1 基本概念回顾

#### 1.1.1 目标区域选取规则

首先来看下目标框即推荐区域的提取。在2014年R-CNN被提出之前，目标框的提取主要分为三种。

第一种就是滑动窗口。滑动窗口本质上就是穷举法，利用不同的尺度和长宽比把所有可能的大大小小的块都穷举出来，然后送去识别，识别出来概率大的就留下来。很明显，这样的方法复杂度太高，产生了很多的冗余候选区域，在现实当中不可行。

第二种是规则块。在穷举法的基础上进行了一些剪枝，只选用固定的大小和长宽比。但是对于普通的目标检测来说，规则块算法依然需要访问很多的位置，复杂度高。

第三种是选择性搜索（selective search，ss）。从机器学习的角度来说，前面的方法召回是不错了，但是精度差强人意，所以问题的核心在于如何有效地去除冗余候选区域。其实冗余候选区域大多是发生了重叠，选择性搜索利用这一点，自底向上合并相邻的重叠区域，从而减少冗余。

这里重点讲一下selective search，使用了Selective Search1方法从一张图像生成约2000-3000个候选区域。基本思路如下：

* 使用一种过分割手段，将图像分割成小区域
* 查看现有小区域，合并可能性最高的两个区域。重复直到整张图像合并成一个区域位置
* 输出所有曾经存在过的区域，所谓候选区域（这里的合并，只是生成候选区域，与后边的分类过程不一样，后边的微调用到的非极大值抑制剔除掉冗余的bbox不是同一个内容）
* 候选区域生成和后续步骤相对独立，实际可以使用任意算法进行。

合并的规则是优先合并以下四种区域：

* 颜色（颜色直方图）相近的
* 纹理（梯度直方图）相近的
* 合并后总面积小的
* 合并后，总面积在其BBOX中所占比例大的

注意：第三条，保证合并操作的尺度较为均匀，避免一个大区域陆续“吃掉”其他小区域。例：设有区域a-b-c-d-e-f-g-h。较好的合并方式是：ab-cd-ef-gh -> abcd-efgh -> abcdefgh。不好的合并方法是：ab-c-d-e-f-g-h ->abcd-e-f-g-h ->abcdef-gh -> abcdefgh。

述四条规则只涉及区域的颜色直方图、纹理直方图、面积和位置。合并后的区域特征可以直接由子区域特征计算而来，速度较快。因此在R-CNN中选取了选择性搜索算法来生成后序后续送入CNN的推荐区域（Region Proposal）。

#### 1.1.2 IoU（Intersection-over-Union），即交并比:

是目标检测中使用到的一个重要概念，是一种测量在特定数据集中检测相应物体准确度的一个标准。IoU表示的是预测的候选框（candidate bounding box）与原标记框（ground truth bounding box）的交叠率（或者重叠度），也就是它们的交集与并集的比值。相关度越高该值。最理想情况是完全重叠，即比值为1，如下图1.1所示：

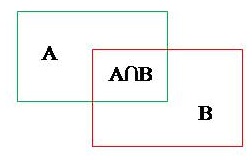


图1.1 IoU示意图

那么，IoU的计算公式为：

（1.1）

#### 1.1.3 非极大抑制NMS（non maximum suppression）

从其名称就可以看出，NMS想要做到的是找到局部的极大值，抑制非极大值，主要用于在图像检测中剔除掉检测出来的冗余的bbox。对于目标检测算法，最终我们会得到一系列的bbox以及对应的分类score，NMS所做的工作就是将同一个类别下的bbox按照分类score以及IoU阈值做筛选，剔除掉冗余的bbox，NMS的具体过程为：

* 在算法得到一系列bbox后，按照类别划分；
* 对于每一个分类，根据分类score对该类别下所有的bbox做降序排列，最终得到一个排好序的列表list\_i；
* 从列表list\_i中取出最大score的bbox\_x，并将其与list\_i中所有其他的bbox\_y计算IoU，若IoU大于某个阈值T，则剔除bbox\_y，最终保留bbox\_x；
* 从剩余的list\_i中重复上述的选择操作，直到list\_i中的bbox都完成筛选；
* 对其余的类别的列表重复上述两步的操作；

### 1.2 R-CNN算法流程图

图1.2解释了R-CNN的算法流程，RCNN的整个过程是这样的：

* 通过selective search算法为每一张待检测图片提取出2000左右的候选框；
* 每张图片中的候选框调整到227\*277大小，然后分别输入到CNN中提取特征；
* 提取到的特征后，利用svm对这些特征进行分类识别；
* 利用NMS（非极大值抑制）算法对结果进行抑制处理。

2000个左右的proposal都要使用CNN来提取特征，计算量是很大的。然后利用这些特征进行分类时，同样需要很大的内存。

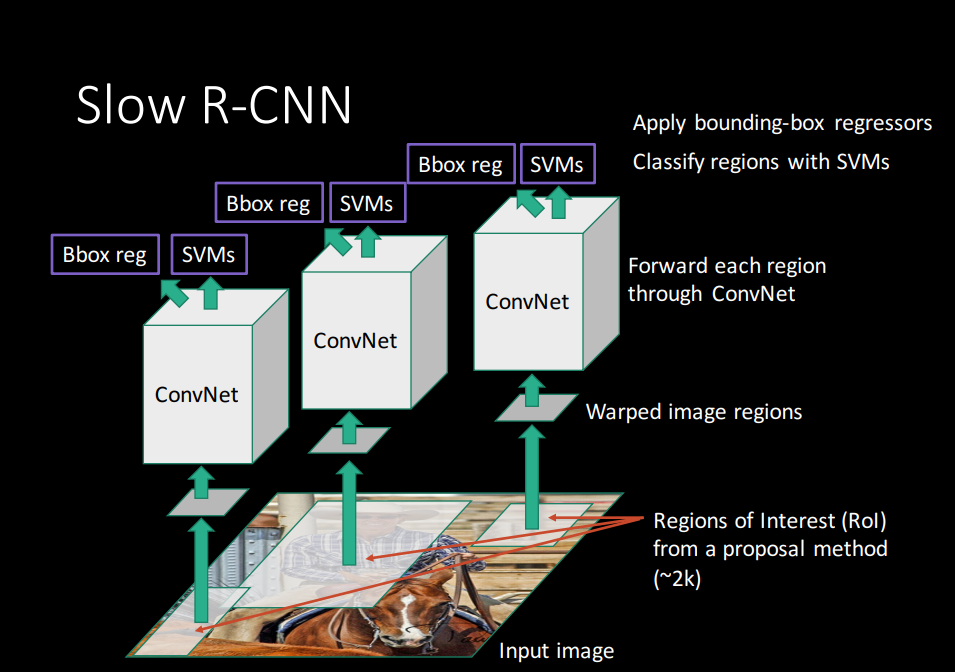


图1.1 R-CNN算法流程图

## 2. SPPnet回顾总结

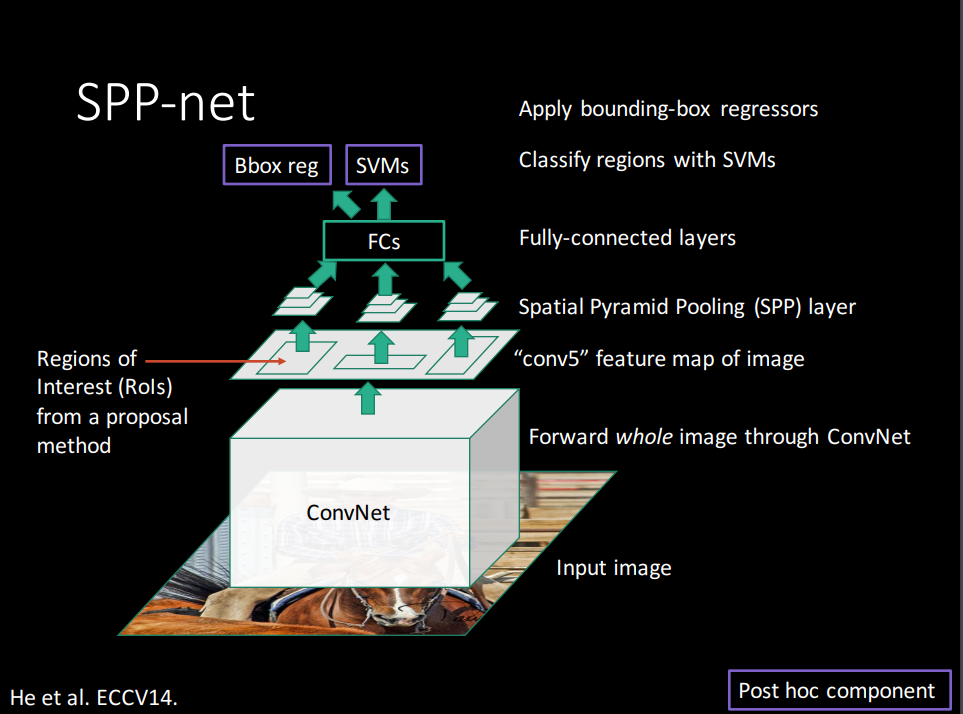
在R-CNN之后，虽然在Fast RCNN之前有提出过SPPnet算法来解决RCNN中重复卷积的问题，但是SPPnet依然存在和RCNN一样的一些缺点比如：训练步骤过多，需要训练SVM分类器，需要额外的回归器，特征也是保存在磁盘上。因此Fast RCNN相当于全面改进了原有的这两个算法，不仅训练步骤减少了，也不需要额外将特征保存在磁盘上。

SPPNet的整个过程：

* 和R-CNN一样使用selective search算法为每一张待检测的图片提取出2000左右的候选框，这一点和RCNN相同；
* 特征提取阶段，对整个图片进行卷机，经过5个卷积层和2个降采样层（这两个降采样层分别跟在第一和第二个卷积层后面）后，进入ROIPooling层后，将卷积后的结果输入到SPPNet中，然后提取该卷积结果中的ROI，也就是找到原图像ROI在feature map中的映射。然后对各个候选框对应的feature map上的块做空间金字塔池化（如图2.2），将每个ROI区域，分别分成1\*1，2\*2和4\*4总共21块区域，然后为每一个小的区域提取一个特征，总共下来提取出了一个维度是21的特征向量；
* 使用SVM算法对得到的特征向量分类识别；
* 使用NMS做极大值抑制，bounding box进行精修。

SPPNet和RCNN最大的不同就在于第二步处理上，对一张图片只做一次特征提取，显然会提高效率，速度得到很大的提升。

SPPnet的算法流程图如下图2.1所示：



原图中ROI在卷积后的feature map中的映射ROI区域

图2.1 SPPnet算法流程图

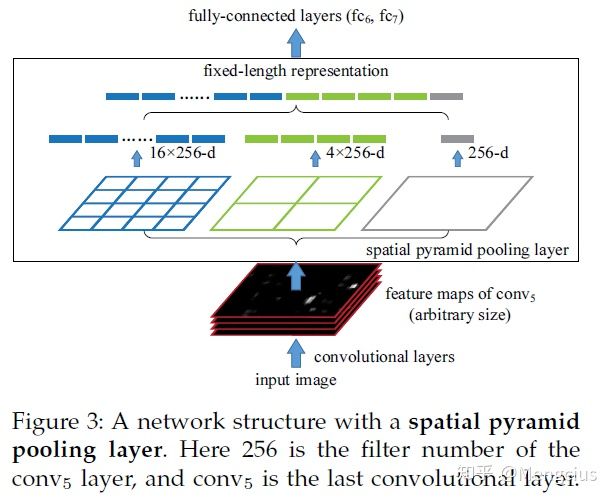


图2.2 空间金字塔池化

基于VGG16的Fast RCNN算法在训练速度上比RCNN快了将近9倍，比SPPnet快大概3倍；测试速度比RCNN快了213倍，比SPPnet快了10倍。在VOC2012上的mAP在66%左右。

总结：

在RCNN的基础上做了改进

* 取消了crop/warp图像归一化过程，解决图像变形导致的信息丢失以及存储问题，精度提高；
* 引入空间金字塔池化层，只对原图提取一次特征，SPP放在卷积层之后，有效解决了卷积层的重复计算问题，速度提高了30倍以上；
* 在feature map上提取ROI特征，即找到与原图上候选框对应的映射块，映射块是SPP的输入，这样就只需要在整幅图像上做一次卷积；
* 分别训练三个模型：CNN模型（提取图像特征）、SVM分类器（预测类别）、回归模型（修正边界，L2损失）。

缺点：

* 和RCNN一样，训练多级流水线，分别隔离训练三个模型：CNN fine-tuning模型（提取图像特征）、SVM分类器（预测类别）、回归模型（修正边界），大量的中间结果需要转存，无法整体训练参数；
* SPP-Net在无法同时Tuning在SPP-Layer两边的卷积层和全连接层，很大程度上限制了深度CNN的效果；
* 与R-CNN不同，微调算法不能更新在空间金字塔池之前的卷积层。不出所料，这种限制（固定的卷积层）限制了深层网络的精度；
* Proposal Region仍然很耗时；
* Backbone（主干）网络参数沿用了分类网络的初始参数，没有针对检测问题进行优化。

## 3. Fast R-CNN

### 3.1 Fast R-CNN优点

相对于R-CNN和SPPnet方法，Fast R-CNN有以下几个优点：

* Fast RCNN具有更高的目标检测的精度（mAP）；
* 训练过程采用多任务的损失函数；
* 训练可以更新所有网络层的参数，而SPPnet很难更新，效率很低；
* 不需要额外的磁盘空间存储特征，R-CNN和SPPnet都需要大量的中间存储，需要很大的存储空间。

针对第三个优点，这里做一下简单解释，Fast RCNN能够使用反向传播来更新训练所有的网络权重。SPPnet不能更新所有的权重，不能更新spp之前层的参数。（注意：这里不是说不能更新，而是由于在finetune的过程中反向传播非常低效。）

根本原因是当每个训练样本（即RoI）来自不同的图像时，通过SPP层的反向传播是非常低效的，这正是训练R-CNN和SPPnet网络的方法。低效的部分是因为每个RoI可能具有非常大的感受野，通常跨越整个输入图像。由于正向传播必须处理整个感受野，训练输入很大（通常是整个图像）。

RoI-centric sampling和image-centric sampling的区别：SPP-net是先把所有图像用SS计算的RoIs存起来，再从中每次随机选128个RoIs作为一个batch进行训练，这128个RoIs最坏的情况来自128张不同的图像，那么，要对128张图像都送入网络计算其特征，同时内存就要把128张图像的各层feature maps都记录下来（反向求导时要用），所以时间和空间上开销都比较大；而Fast R-CNN虽然也是SS计算RoIs，但每次只选2张图像的RoIs(一张图像上约2000个RoIs)，再从中选128个作为一个batch，那么训练时只要计算和存储2张图像的Feature maps，所以时间和内存开销更小。

### 3.2 算法流程图详解

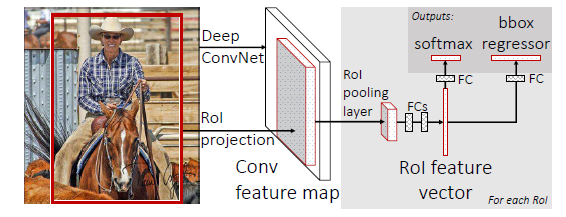


图3.2 Fast R-CNN算法流程图

1. Fast R-CNN网络将整个图像和一组候选框（候选框大概1k-2k，与R-CNN类似）作为输入。
2. 网络首先使用卷积层和最大池化层来处理整个图像，以产生卷积特征图。然后，找到原图像的每个候选框映射在feature map上的目标区域，RoI池化层从特征图中提取固定长度的特征向量（如图3.3所示）。
3. 每个特征向量被送入一系列全连接（fc）层中，其最终分支成两个同级输出层 ：一个输出K个类别加上1个背景类别的Softmax概率估计（代替SVM分类器，分类结果略好）；另一个为K个类别的每一个类别输出四个实数值。每组4个值表示K个类别的一个类别的检测框位置的修正。

具体的流程还可以用图3.3来解释：

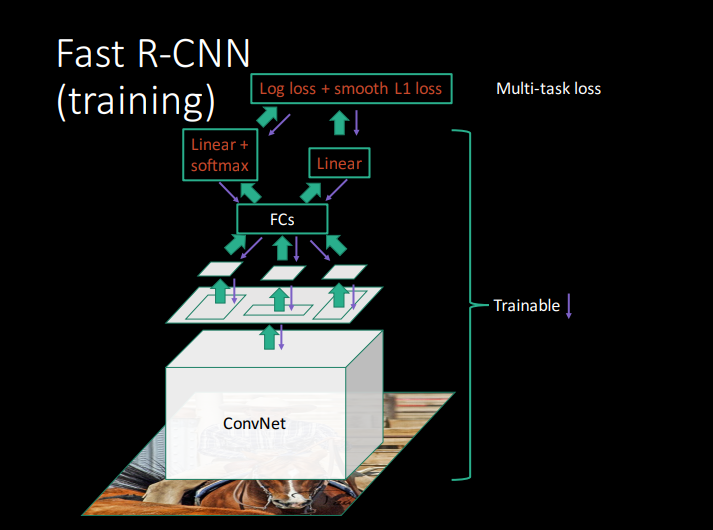


图3.3 FastR-CNN算法流程图

### 3.3 相关知识解释

#### 3.3.1 RoI pooling层

RoI池化层使用最大池化将任何有效的RoI内的特征转换成具有H×W（例如，7×7）的固定尺度的小特征图，其中H和W是层的超参数，独立于任何特定的RoI。在本文中，RoI是卷积特征图中的一个矩形窗口。每个RoI由指定其左上角(r, c)及其高度和宽度(h, w)的四元组(r, c, h, w)定义。

RoI最大池化通过将大小为h×w的RoI窗口分割成H×W个网格，子窗口大小约为h/H×w/W，然后对每个子窗口执行最大池化，并将输出合并到相应的输出网格单元中具体操作可看图3.4。同标准的最大池化一样，池化操作独立应用于每个特征图通道。对原矩形的feture map中的每一个长方形区域进行最大池化，这样就将原来的矩形feture map变换成了正方形的feature map。

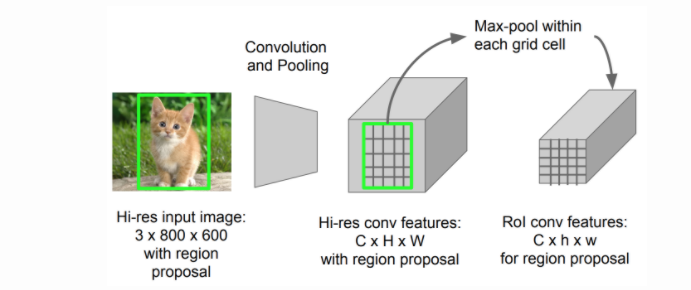
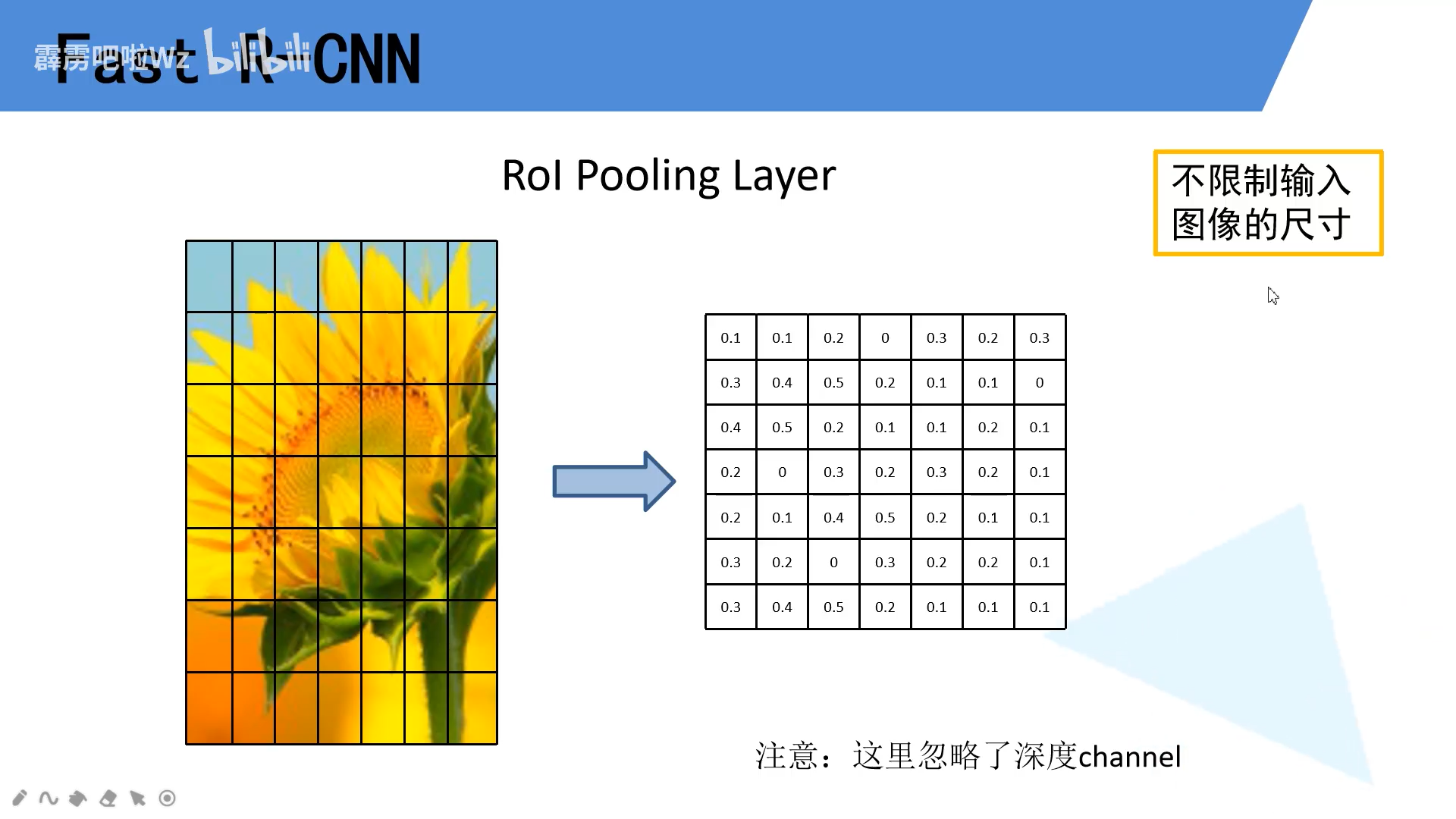


图3.4 RoI pooling示意图



#### 3.3.2 微调

论文提出了一种更有效的训练方法，利用训练期间的特征共享。在Fast RCNN网络训练中，随机梯度下降（SGD）的小批量是被分层采样的，首先采样N个图像，然后从每个图像采样R/N个RoI。关键的是，来自同一图像的RoI在向前和向后传播中共享计算和内存。减小N，就减少了小批量的计算。例如，当N=2和R=128（实际操作中，R大约为2k个）时，得到的训练方案比从128幅不同的图采样一个RoI（即R-CNN和SPPnet的策略）快64倍。

这个策略的一个令人担心的问题是它可能导致训练收敛变慢，因为来自相同图像的RoI是相关的。这个问题似乎在实际情况下并不存在，当N=2和R=128时，我们使用比R-CNN更少的SGD迭代就获得了良好的结果。

除了分层采样，Fast R-CNN使用了一个精细的训练过程，在微调阶段联合优化Softmax分类器和检测框回归，而不是分别在三个独立的阶段训练softmax分类器，SVM和回归器。 下面将详细描述该过程（损失，小批量采样策略，通过RoI池化层的反向传播）

#### 3.3.3损失函数

一个Fast RCNN网络有两个输出层, 第一个输出为K+1个类别的离散概率分布，而第二个输出为 bbox回归的偏置, 每一个正在训练的ROI均利用一个ground truth类别u与ground truth框V, 采 用多任务损失进行分类与边框回归:

第一部分是类别的log损失，第二部分是为止损失的回归损失， 为ground truth，为预测值，方括号是一个指示函数，满足条件为1，否则为0，按照惯例，为背景类，此时忽略回归损失，对于检测框的回归采用了smooth-L1损失，没有使用L2损失。

其中：

这个损失函数相比于L2损失相对于异常值更加鲁棒

#### 3.3.4 尺度不变性（scale invarient）

对于尺度不变性实现，一般采用两种方式：

* "brute force" learning（暴力学习）
* 图像金字塔

在暴力学习的方法中，每张图片处理为相同大小的网络输入，然后网络从特定尺寸的训练数据中学习尺度不变性的目标检测。（R-CNN的方法）

图像金字塔方法：多尺度方法通过图像金字塔向网络提供近似尺度不变性。 在测试时，图像金字塔用于大致缩放-规一化每个候选框。在多尺度训练期间，每次图像采样时随机采样金字塔尺度。

Faste R-CNN中的金字塔仅使用了一个层级，即直接从RoI区域提取出来一个固定大小的特征向量，而SPPnet确是将每一个RoI区域分成21个层级，每个层级提取一个特征，在这个地方，两种方法差别很大。

### 3.4 用到的网络

#### 3.4.1 AlexNet

AlexNet的网络结构如图3.5所示：

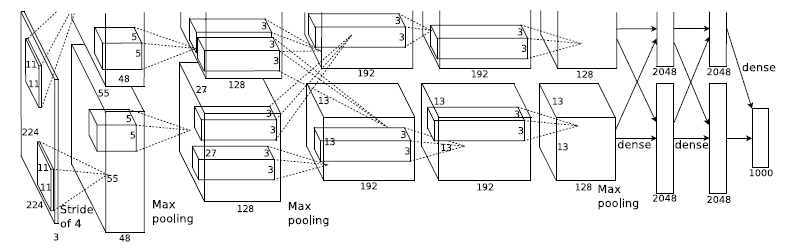


图3.5 AlexNet的网络结构

网络结构如上图所示，共有5个卷积层，3个最大池化层，3个全连接层。如今去看，网络并不能算深，毕竟ResNet网络已经可以达到上百层，但是在当时，实际上用了两块GPU训练网络。接下来将详细介绍其中的具体细节。

具体的参数及神经元个数如图3.6所示：

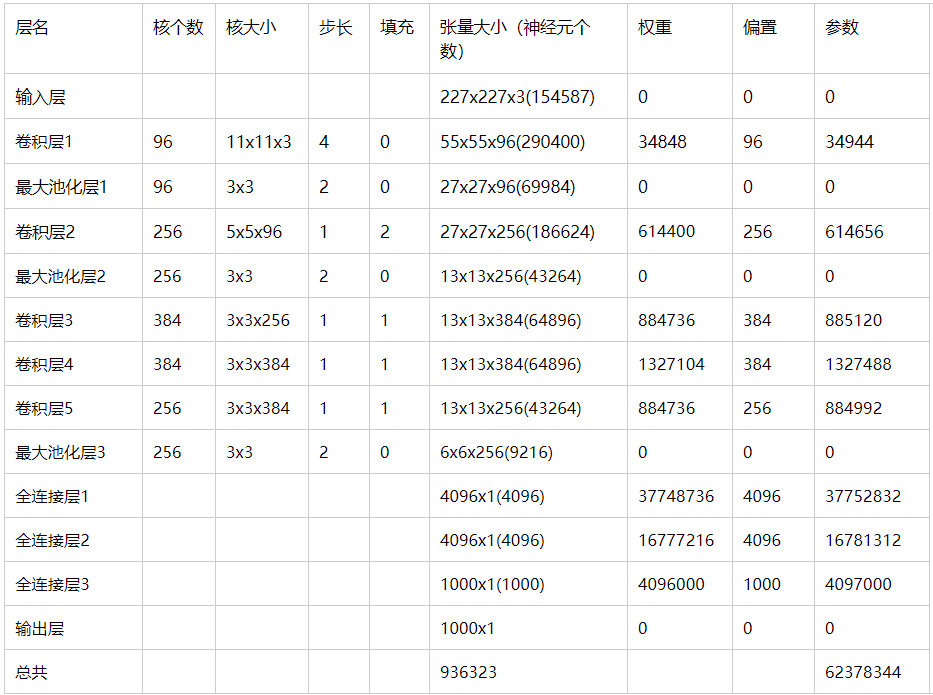


图3.6 参数及神经元个数

#### 3.4.2 VGG16网络

VGG-16的网络结构如图3.7D所示：

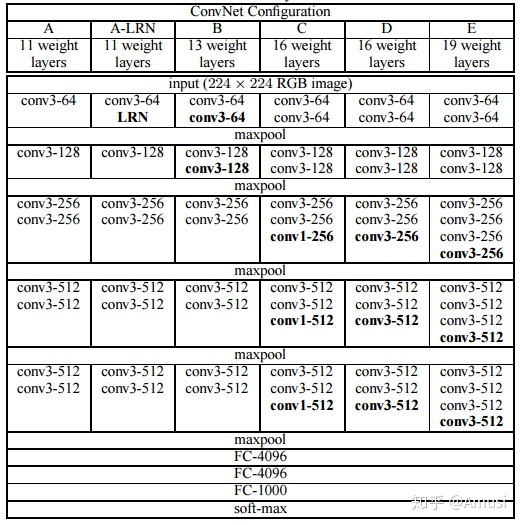
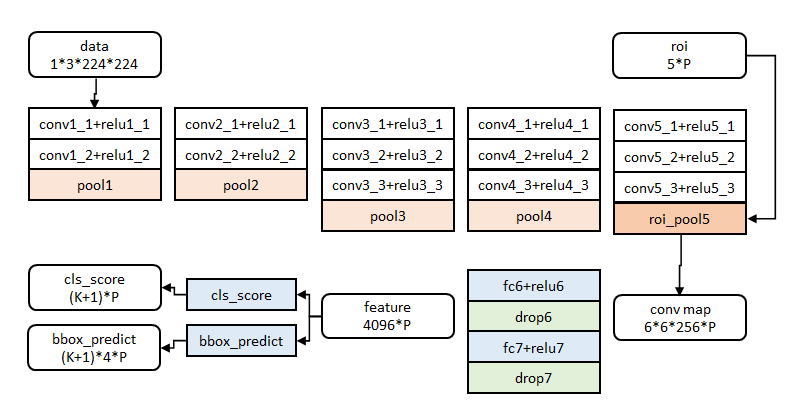


图3.7 不同深度的网络结构

* VGG16包含了16个隐藏层（13个卷积层和3个全连接层），如上图中的D列所示；
* VGG19包含了19个隐藏层（16个卷积层和3个全连接层），如上图中的E列所示；

VGG网络的结构非常一致，从头到尾全部使用的是3x3的卷积和2x2的max pooling。



图像归一化为224×224直接送入网络。前五阶段是基础conv+relu+pooling形式，在第五阶段结尾，输入P个候选区域（图像序号×1+几何位置×4，序号用于训练。

文中给出了大中小三种网络，此处示出最大的一种。三种网络基本结构相似，仅conv+relu层数有差别，或者增删了norm层

## 4. Faster R-CNN

论文的主要贡献在于提出了RPN（region proposal network)。

### 4.1 算法结构图和主要组成部分

#### 4.1.1 Faster R-CNN算法结构图

RPN是一个全卷积网络，能够同时预测每个位置目标的边界（object bound）和目标的得分，fast rcnn使用RPN，通过端到端的训练可以生成高质量的region proposal，将RPN与fast rcnn整合在一起构成了faster rcnn算法，类似于attention的机制，RPN可以告诉算法，where to look。Faster R-CNN的算法结构图如图4.1所示：

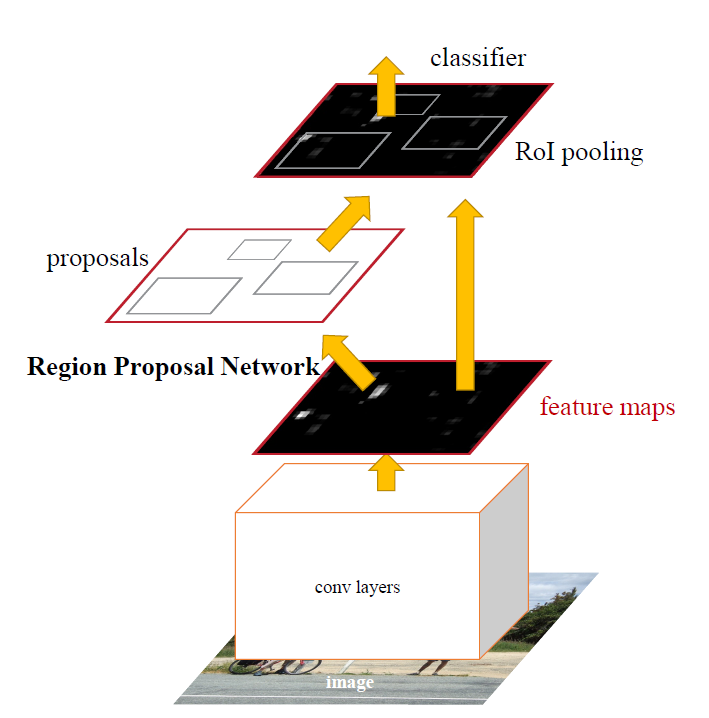


图4.1 Faster R-CNN的算法结构图

#### 4.1.2 Faster RCNN 关键步骤：

* Conv layers。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image的feature maps。该feature maps被共享用于后续RPN层和全连接层；
* Region Proposal Networks。RPN网络用于生成region proposals。该层通过softmax判断anchors属于foreground或者background，再利用bounding box regression修正anchors获得精确的proposals；
* Roi Pooling。该层收集输入的feature maps和proposals，综合这些信息后提取proposal feature maps，送入后续全连接层判定目标类别；
* Classification。利用proposal feature maps计算proposal的类别，同时再次bounding box regression获得检测框最终的精确位置。

Faster RCNN主要由以下四个部分组成：

* conv layer 输入为原始图片，用于提取图片的feature map
* RPN网络，输入为features map，用于生成region proposal，该层为features map 上每个像素生成若干个anchors（9个），随后通过softmax 判断每个anchor是属于foreground（目标）或者background（背景），再利用bounding box regression修正anchors获得精确的proposal位置。
* RoI pooling，该层输入为proposal位置信息和features map，通过proposal的位置信息在features map 上提取region features map候选区，然后通过pooling产生一个固定大小7\*7的特征图，然后将特征图展平送入全连接层进行目标判别。
* classification，利用proposal feature maps计算proposal的类别，同时再次进行一次bounding box regression，对proposal位置进行精修，随后将结果输出。

下图4.2是具体的算法流程图：

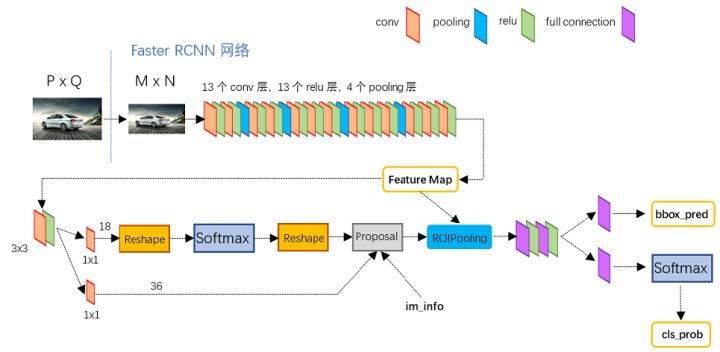


图4.2 Faster R-CNN算法流程图

将一副任意大小PxQ的图像，首先缩放至固定大小MxN，然后将MxN图像送入网络；而卷积层 Conv layers中包含了13个conv层+13个relu层+4个pooling层；RPN网络首先经过3x3卷积，再分别生成foreground anchors与bounding box regression偏移量，然后计算出proposals；而Roi Pooling层则利用proposals以及feature maps，提取proposal feature送入后续全连接和softmax网络作classification。

下图是VGG卷积层：

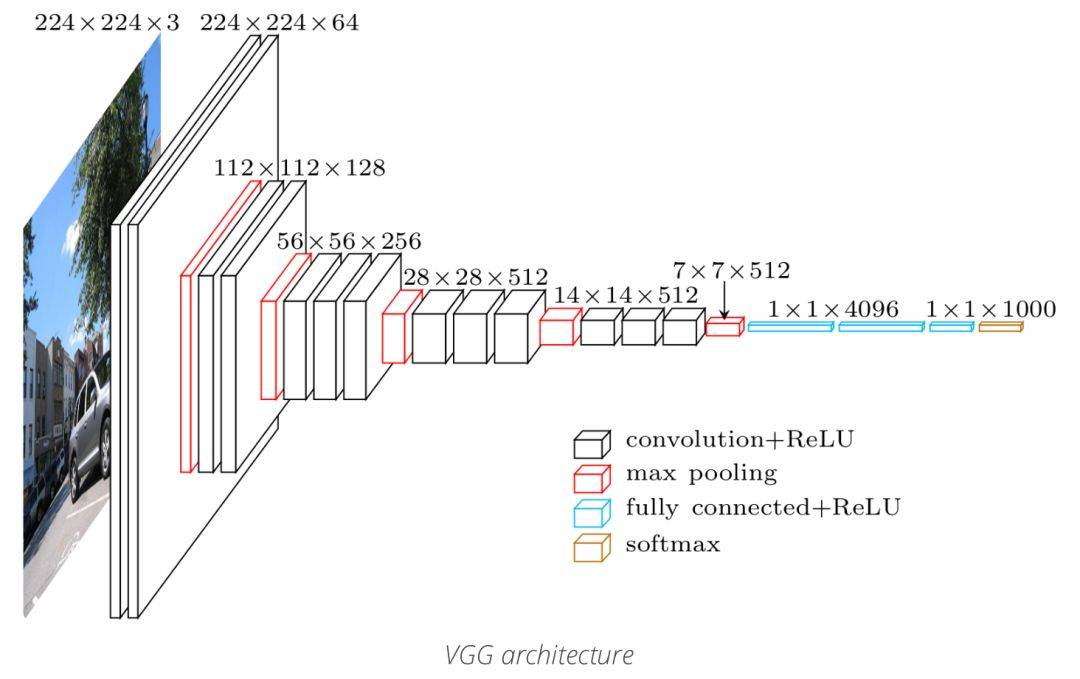


图4.3 VGG卷积层

Conv layers部分共有13个conv层，13个relu层，4个pooling层。

* + - 所有的conv层都是： kernel\_size=3 ， pad=1 ，stride=1，因此conv层不改变原图大小
    - 所有的pooling层都是： kernel\_size=2 ，pad=0 ， stride=2，pooling 层将原图缩小为原来的一半
    - 经过Conv layer后，一个MxN大小的矩阵将变为(M/16)x(N/16)

#### 4.1.3 Region Proposal Networks(RPN)网络

Faster RCNN 层在Fast RCNN 的基础上，对提取候选框进行优化，具体的流程如图4.4所示：

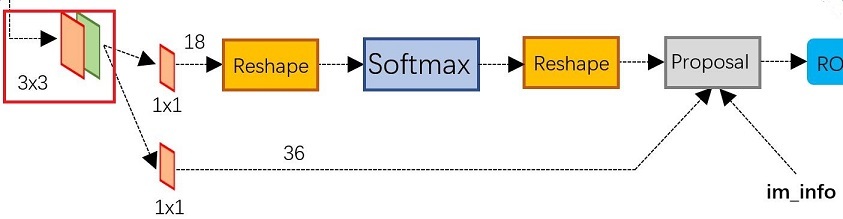


图4.4 RPN方法流程图

RPN网络分为2条线，上面一条通过softmax分类anchors获得foreground和background（检测目标是foreground），下面一条用于计算anchors的bounding box regression偏移量，以获得精确的proposal。而最后的Proposal层则负责综合foreground anchors和bounding box regression偏移量获取proposals，同时剔除太小和超出边界的proposals。其实整个网络到了Proposal Layer这里，就完成了相当于目标定位的功能。

#### 4.1.4 softmax判定foreground与background

RPN网络中利用anchors和softmax初步提取出foreground anchors作为候选区域。

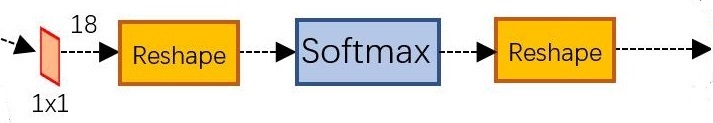


图4.5 判定过程

features map 首先做一个1\*1的卷积, 这个卷积的作用是生成一个 大小的矩阵。该矩阵用于 存储上面提到的foreground与backgr-ound信息 score 。将该特征后接softmax分类获得foreground anchors, 也就相当于初步提取了检测目标候选区域box (一般认为目标在foreground anch-ors中) 。前后两个 reshape 操作目的为便于程序实现。 clc layer输出预测区域共k个，每个有2个参数，即预测为前景的概率和背景的概率，损失用softma-x loss (cross entropy loss) 。监督信息是Y=0,1, 表示这个区域是否为groundtr-uth。确定groundtruth时, 我们需要 确定k个区域中的各个区域是不是有效的，是前景还是背景。K个区域分配标签规则:

* 与某个ground truth(GT)的loU最大的区域的分配正标签
* 与任意GT的loU大于0.7的区域分配正标签
* 与所有GT的loU都小于0.3的区域分配负标签

### 4.2 相关知识点解析

#### 4.2.1 Anchor（锚）

anchor为由一个中心点，周围生成了9个矩形，

对于特征图上的每个3\*3的滑动窗口，计算出滑动窗口中心点对应原始图上的中心点，并计算出k个anchors。矩形长宽比由三个尺寸面积{1282,2562,5122}，三种长宽比例为：1:1，1:2，2:1。如下图，基本覆盖了各种尺寸和形状，引入检测中常用到的多尺度方法。如下图4.5所示：

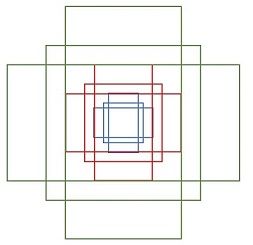


图4.5 anchor示意图

Faster RCNN遍历Conv layers计算获得的feature maps，为feature map上每一个点都配备这9种anchors作为初始的检测框。这样做获得检测框很不准确，之后将会在RPN层，以及最后进行2次的bounding box regression修正检测框位置。

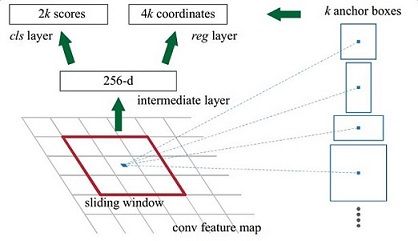


图4.6 修正过程

如上图, 对于每一个点的k个anchor来说，从conv layer提取出得特征具有256维, 对于每一个anchor, 需要分 foreground与background, 因此共有2k个score, 对于每一个anchor共有 四个坐标值。因此共 有4k个coordinates（坐标）。在训练阶段, 程序将会从这些anchor中挑选出一些合适的anchor进行训练。 因此RPN最终就是在原图尺度上，对每一个像素设置9个尺度的候选anchor。然后用cnn去判断哪些Anchor 是里面有目标的foreground anchor, 哪些是没目标的backgroud。所以, 仅仅是个二分类而已!

对于一张1000\*600\*3的图像，大约有60\*40\*9（20K）个anochors，忽略掉跨域边界的anochor后，剩下的大约有6k个anochors。对于RPN网络生成的候选框之间存在大量重叠，基于候选框的cls得分，采用非极大抑制，IoU设置为0.7这样每张图片只剩下2k个候选框。

