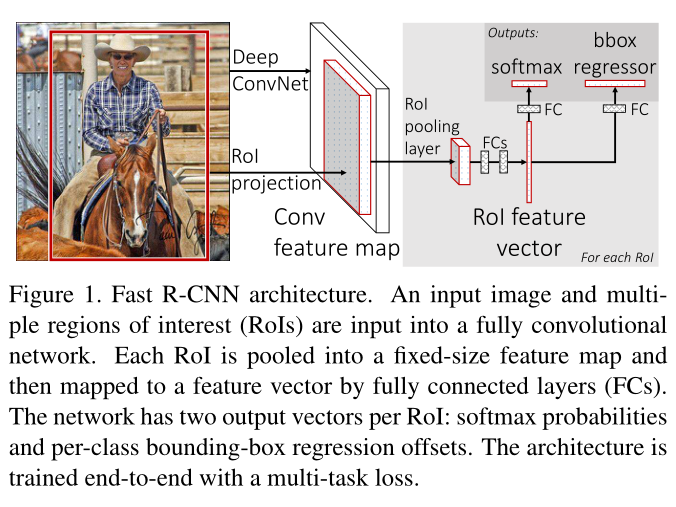
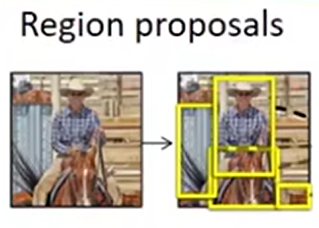
**Faster R-CNN论文的讲解**

是在Fast R-CNN论文的基础上进一步加速的，我们先来回顾一下Fast R-CNN的算法：



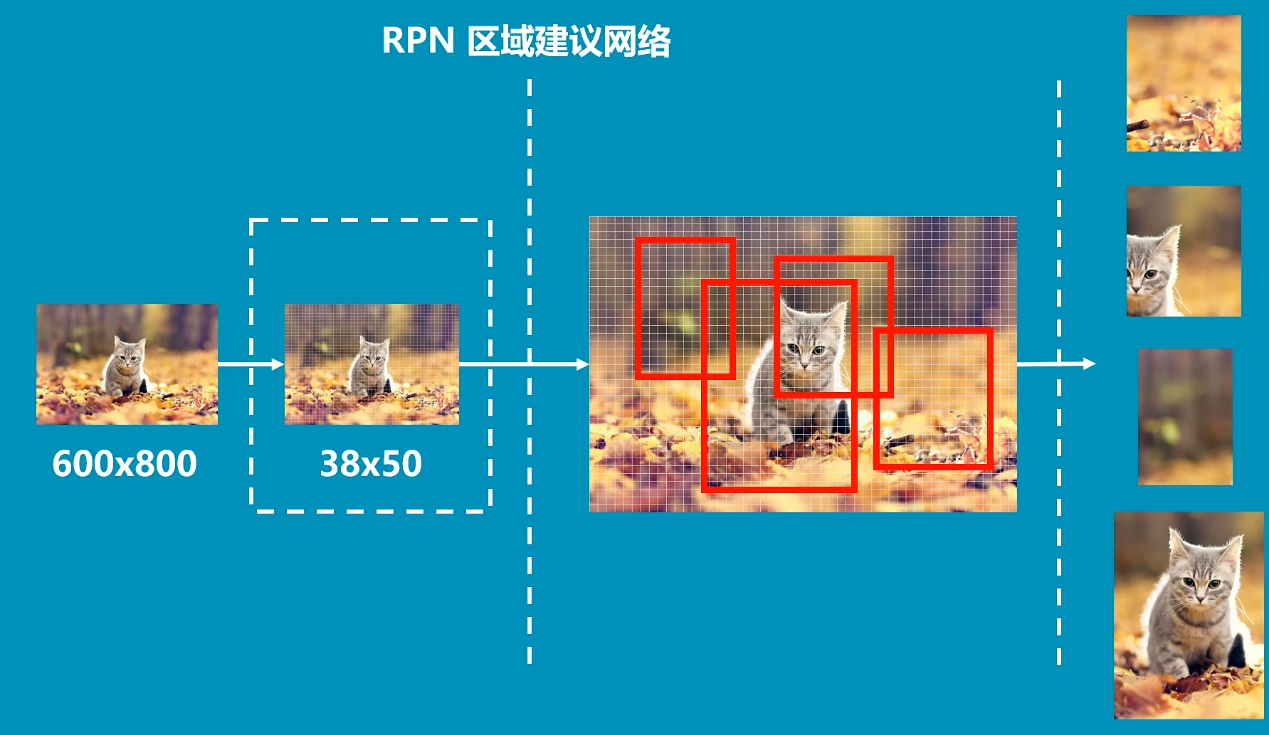
接收一整张图片后，通过一些卷积层来得到一整张图像的特征图，通过ROI映射得到候选区域的特征图。然后通过ROI池化层将候选区域的特征图转化成特定大小的候选区域特征图，然后识别出这个候选区域的类别和坐标。

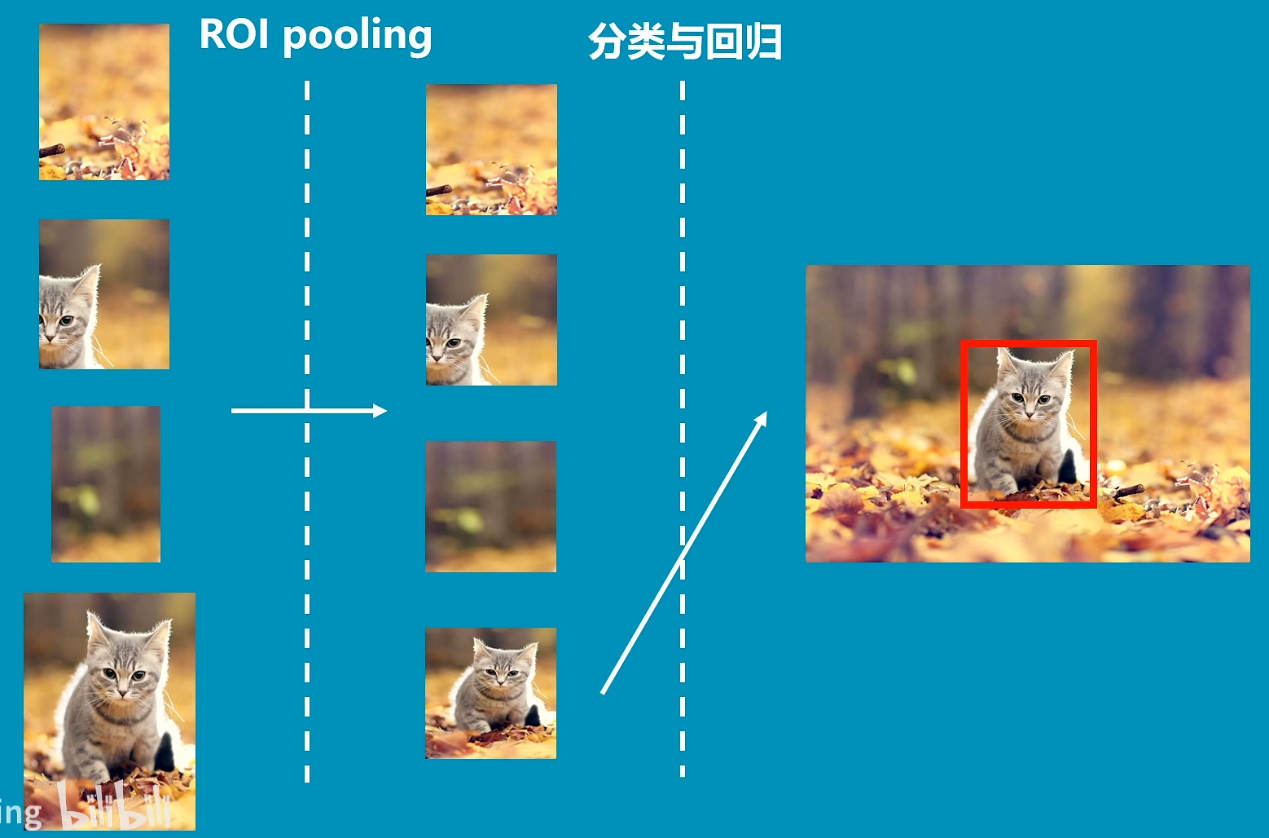
Fast R-CNN使用非常深的网络实现了接近实时的速率，但是这个实时有一个前提条件，因为它忽略了在region proposal上花费的时间。region proposal直接翻译就是区域建议，它指的就是候选区域的生成过程，如下图所示：



对于Fast R-CNN来讲，除了后面的检测流程，还需要一个单独的步骤，就是从原图中提取候选框的过程。我们使用SS算法在每张图像上得到接近200张候选框，有了这些候选框之后，我们才能利用ROI映射得到候选框的特征图。

整体来说，网络运行包含两步，一步是粗略的筛选，一步是精细的调整：





单单看后面的检测流程，确实可以达到一个非常接近实时的检测效率，但是region proposal也是不可忽略的过程，而恰恰这个步骤又很耗时，因此，proposals成为了一个测试时的计算瓶颈。

因此，整篇Faster R-CNN，最核心的一个问题就是解决region proposals的问题，如何减少这一步的时间，让整个算法真正达到实时的部署。

之前在讲R-CNN、SPP-Net以及Fast R-CNN的论文时，region proposal这一步用的都是一个叫做Selective Search的算法，它的速度很慢，在一个CPU上，每一张图片产生候选区域需要花费两秒钟。这个算法太慢了，我们可以考虑换一个算法去实现它，换成EdgeBoxes算法，它只需要花费0.2秒，时间还是有点长。我们上面考虑的两个算法都是比较传统的方法，一般是在CPU上去实现的。

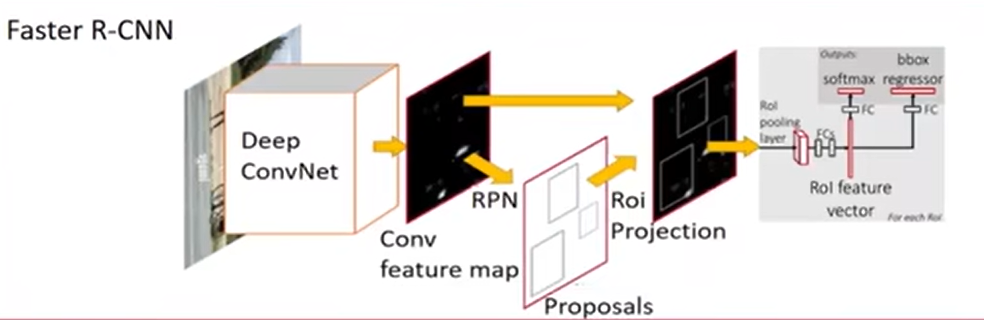
有学者提出了一个想法，既然卷积神经网络那么方便，能用GPU加速计算，那么是否可以利用CNN和GPU和加速实现region proposal这一步呢？

这确实是一个有效的解决思路，但是它忽略了共享计算。也就是说，要用CNN去实现，需要先提取图片特征，然后得到这些候选区域，而后面的检测流程也是先从一张图像上得到图像特征。

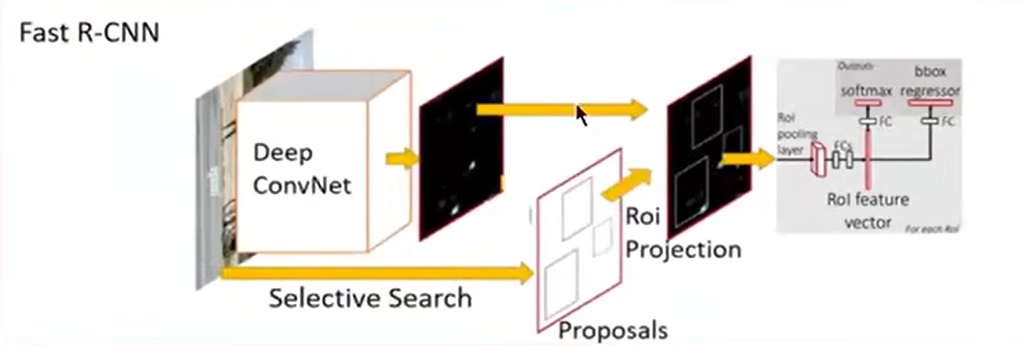
那么这两个CNN特征提取的步骤是否可以合到一起，从而实现共享计算呢？这样就可以提取一次特征实现两个功能，不需要使用两次CNN来提取特征。

因此，Faster R-CNN这篇论文的思想就出来了，它使用一个深度卷积神经网络来实现计算proposals的过程，并为它起了一个名字，叫region proposal network。

RPN可以与原来的Fast R-CNN共享卷积层，从而使得候选区域生成时间降低到每张图像10毫秒。

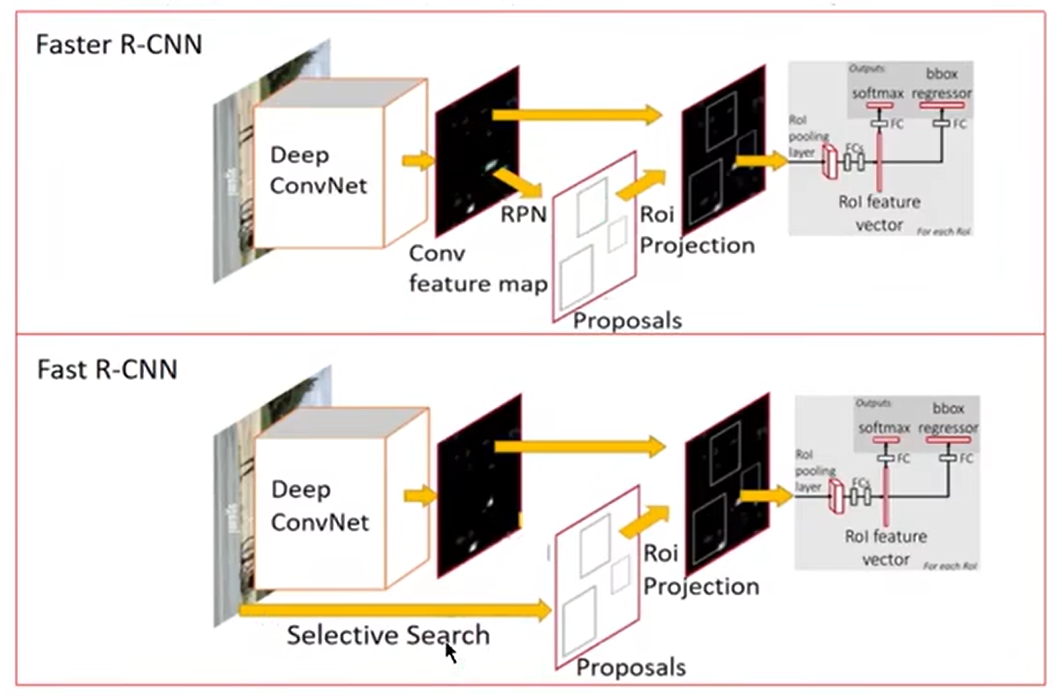


输入一张图片后，会通过深度卷积神经网络得到卷积特征图，把卷积特征图传给RPN模块来生成候选区域的位置，再通过ROI映射和原始的特征图来得到ROI区域的特征，得到每一个候选区域特征后就可以用于后续的分类和回归。



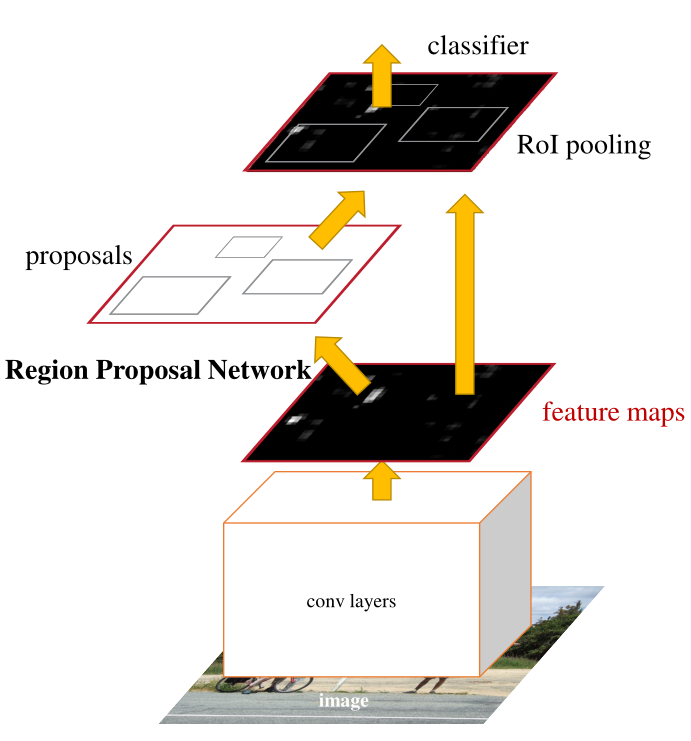
我们把Fast R-CNN也画成这个样子。

它们两个最大的区别是Fast R-CNN在原图上单独使用一个耗时的SS算法来提取候选区域。



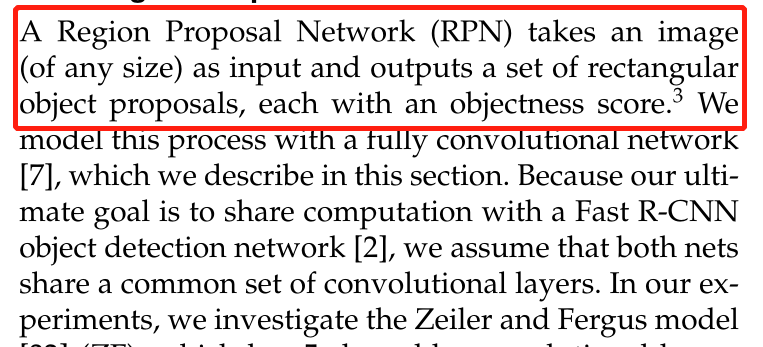
RPN如何产生候选区域的：

我们看一下论文中的图示：



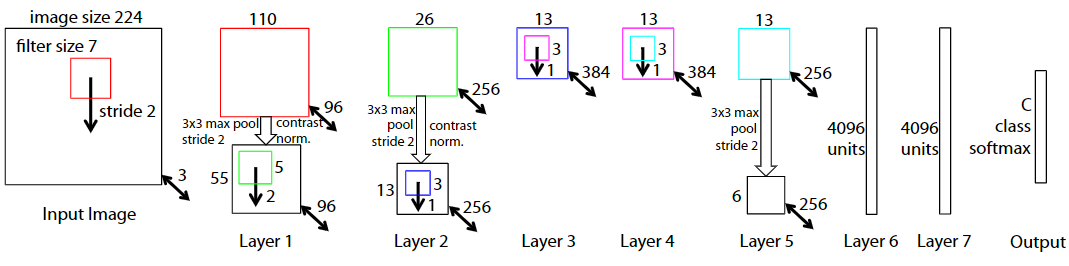
这里容易出现误解，RPN网络在图示的网络结构中，它标在了黄色箭头的旁边，认为只包含从特征图提取特征的过程，但这其实不是很准确，因为RPN最根本的作用是根据CNN来实现region proposal的过程。因此，它的真实过程应该从输入一张图片开始到生成候选区域结束，只不过最开始的那个卷积层可以和Fast R-CNN实现共享计算。

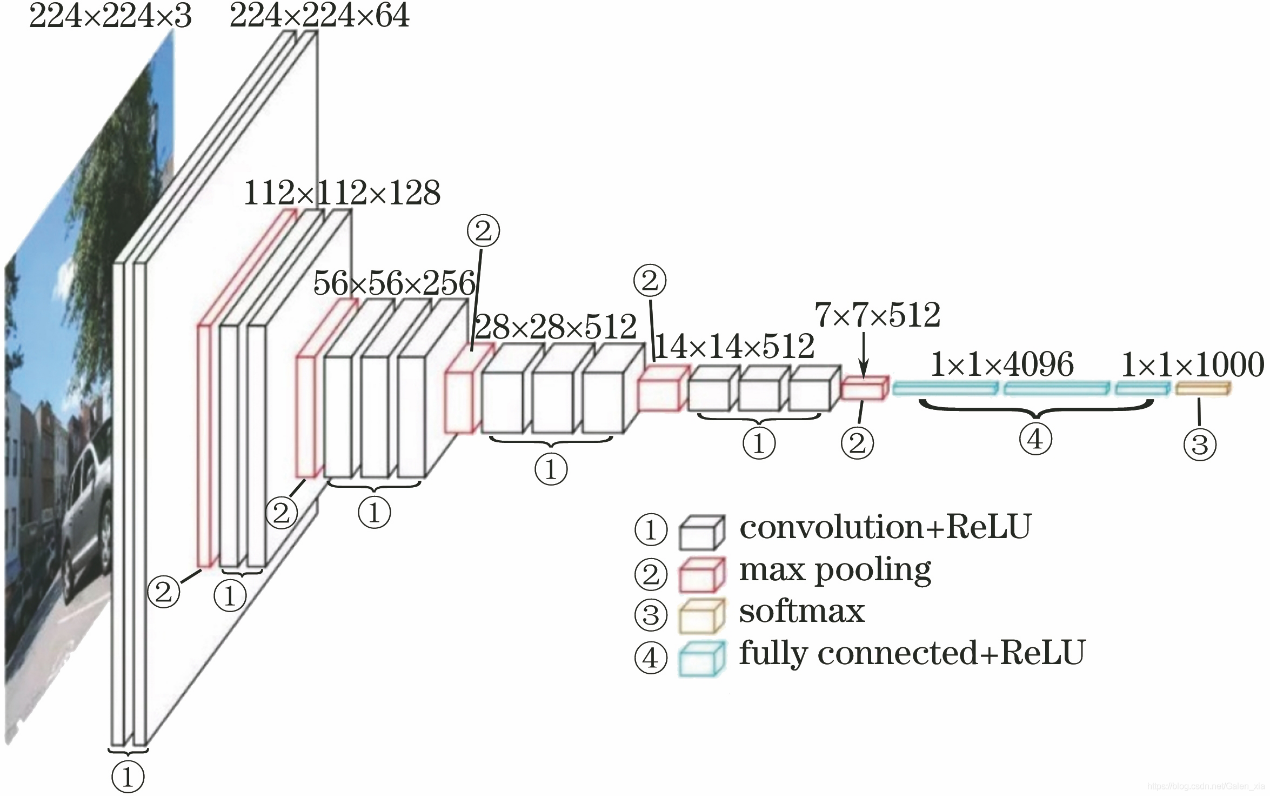
在整篇论文中，作者是按照整个流程作为RPN网络，如下图所示：



所以论文里面讲RPN网络接收的是一整张图片作为输入，然后输出一些列的矩形候选框。

在最前面的卷积神经网络中，作者验证了ZF模型和VGG16模型。ZF模型比较小，只有5个共享的卷积层。VGG16比较大，它有13个共享的卷积层。

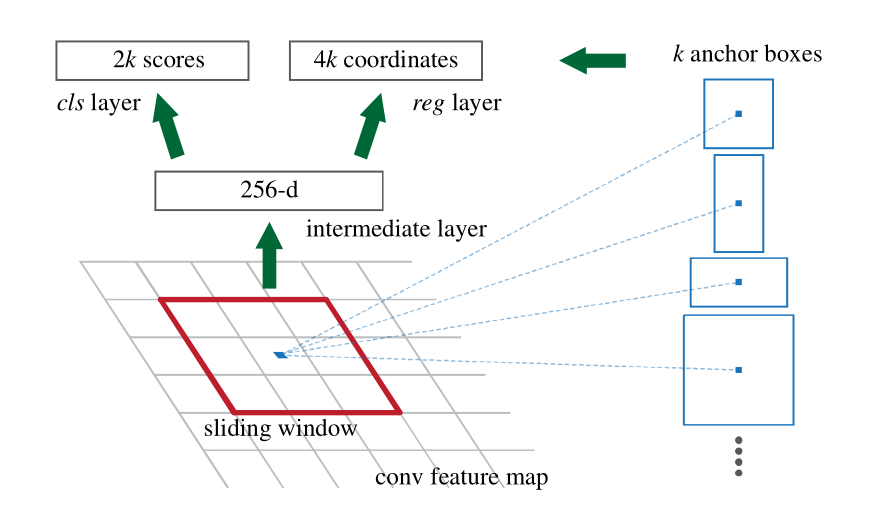




它们的目的主要是为了得到后面的特征图，所以选择ZF或者VGG16区别不明显。

为了生成候选框，Faster R-CNN使用一个卷积网络在共享特征的最后一层特征图上进行滑窗操作，每次滑窗，在特征图上截取大小的特征块(其中表示滑窗卷积网络输入窗口的宽和高，表示共享卷积特征最后一层特征图的通道数)。

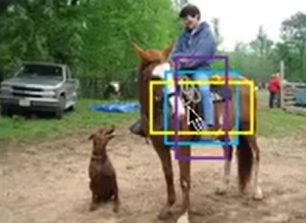
经过滑窗卷积网络后输出一个低维特征，并分别送入两个并列的全连接层，其中一个全连接层用于分类，判断当前滑窗对应的图像区域是否包含目标，另一个全连接层用于回归候选框的位置，判断当前候选框相对于当前滑窗所对应图像区域的位置偏移量及缩放尺度。具体的执行过程见下图所示：



在一个通道的特征图上，通过一个的滑动窗口，经过一个中间层产生256维的低维特征，传给分类层，产生类别，再传给回归层产生坐标。中间层可以通过的卷积层来实现，分类层可以通过的卷积层（或全连接层）加一个softmax层实现；回归层可以用一个的卷积层（或全连接层）来实现。

在每个滑窗位置，会同时预测多个候选框。假设需要预测的候选框的个数为，则用于回归候选框位置的全连接层(回归层)有个输出，分别是规范化的水平方向及垂直方向的位置偏移量和缩放尺度；用于类别判别的全连接层(分类层)有个输出，分别是当前预测框框中目标或未框中目标的概率。这个待预测的候选框中，每一个框的偏移和缩放都基于一个固定的参考框。个待预测框对应个参考框，每个参考框相对于滑窗的感受野的位置都是固定的，即都有固定的大小和宽高比。这样的参考框，我们称之为锚(anchor)。

对于任意一个滑窗位置，各个锚的中心点和滑动窗口在原图上感受野的中心点重合，设定若干锚需要取到的面积尺度和宽高比，则对于每个滑窗位置，错的个数都是固定的(比如，面积设定为、和)三个尺度，宽高比设定为1:1、1：2和2：1，则每个滑窗位置对应3×3=9个锚)。通常情况下，每个滑窗位置会设置3个面积尺度和3个宽高比，每个滑窗位置包含=3×3=9个锚，如果一个特征图的宽和高分别为和，则总的锚个数为。



这种定义锚的方法有一个显著特征：检测结果相对于目标在图像中的位置具有平移不变性。道理很简单，在原始图像上，如果一个目标可以被某个滑窗位置对应的一个锚框中，把该目标平移到图像中的另一个位置后，也能够被另一个滑窗位置对应的锚框中。

**损失函数：**

在训练候选框提取网络的时候，锚被分成两类，框中目标的锚被标记为正样本(positive)，未框中目标的锚被标记为负样本(negative)。

所谓正样本，是通过锚与真值相交的情况来定义的，具体而言，基于两种方式实现。对于每一个真值，所有锚与这个真值要么相交，要么不相交，在相交的情形中：与这个真值有最大交并比的那个锚标记为正样本；与这个真值的交并比＞0.7的那些锚也标记为正样本。所谓负样本，指的是与所有真值的交并比<0.3的锚。除此之外的其他情况的锚不需要标记，在模型训练的过程中不会被使用。

基于这样的定义，可以给出RPN损失函数的公式：



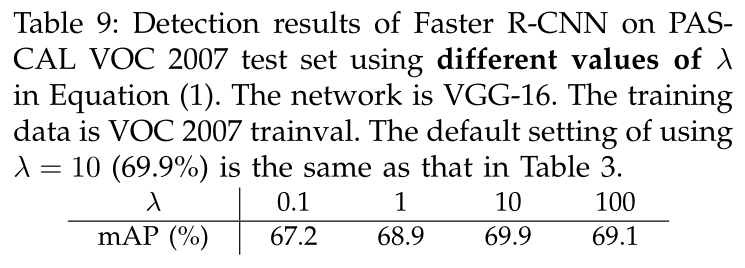
这里，是一个小批中的一个锚的索引，是一个锚是一个对象的预测概率。如果锚为正，地面真实标签，如果锚为负，则为0。是表示预测边界框的4个参数化坐标的向量，是与正锚相关的GT的向量。

分类损失是两个类的损失。对于回归损失，我们使用，其中是Fast R-CNN中定义的稳健损失函数（平滑）。术语意味着回归损失仅对正锚点被激活（），否则将被禁用（）。层和层的输出分别由和组成。

这两项用和归一化，并使用平衡参数进行加权。

在我们当前的实现中，我们一般取（小批量的大小），（锚点位置的数量）。默认情况下，我们设置，因此cls和reg项的权重大致相同。

我们通过实验表明，结果在很大范围内对λ的值不敏感。我们还注意到，上述的标准化是不需要的，可以进行简化。



对于边界盒回归，我们采用下面4个坐标的参数化:



其中, ,和表示盒子的中心坐标以及它的宽度和高度。变量、和分别用于预测框、锚框和GT框(、、也是如此) 可以认为这是从锚框到附近的GT框的边界框回归。

**训练候选框提取网络：**

训练候选框提取网络仍然采用随机梯度下降法(SGD)，为了提升训练速度，每个训练批次的锚都来自同一图像上的随机采样。一般情况下，随机选取锚在概率上会比较偏向未框中目标，为了保证训练样本的均衡，需要人为地尽量保持框中目标的锚(正样本)与未框中目标的锚(负样本)在数量上保持1：1的比例。比如一个小批次如果包含256个锚，就要求这256个锚中，有128个为正样本，另外128个为负样本，当实际情况下确实无法选到128个正样本时，再使用负样本补齐差额。

实际训练时，一个常用的做法是基于ImageNet进行预训练。对于新增的网络层，一般使用均值为0，标准差为0.01的高斯分布进行初始化，然后对整个网络基于新的样本数据进行调优训练。

**RPN和Fast R-CNN共享特征的方法：**

在检测过程中，RPN的作用是输出候选框，Fast RCNN的作用是基于候选框进行目标检测，两个网络的训练和预测过程目前彼此独立。为了更进一步地提升算法的整体效率，可以让这两个网络共享基础的卷积特征，然后设计另一个一致网络来处理候选框提取和目标检测这两个过程。为了实现这个目标，可以采取如下3种方法。

**1.交替训练法**

先训练RPN，然后基于RPN输出的候选框，以RPN的卷积层参数作为初值训练Fast R-CNN，再用Fast R-CNN的卷积层参数作为初值训练RPN。如此反复迭代，得到共享基础卷积层的RPN和Fast R-CNN。交替训练法在实际应用过程中，大致分为4个步骤：

第一步，基于ImageNet的预训练模型，端到端地训练用于提取候选框的RPN；第二步，同样基于ImageNet的预训练模型，使用第一步RPN输出的候选框，训练Fast R-CNN，此时RPN和Fast R-CNN两个网络尚无共享的卷积层；

第三步，使用Fast R-CNN初始化RPN，在训练RPN的时候，固定底层共享的卷积层网络参数，仅迭代调优RPN中共享卷积层以外的网络参数；

第四步，训练FastR-CNN，同样固定底层共享的卷积层网络参数，仅迭代调优Fast R-CNN中共享卷积层以外的网络参数。

通过上述4个步骤，就得到了一个统一的共享底层卷积特征的用于目标检测的整体网络。上述交替训练的过程可以重复多次，但实验表明，多次迭代对最终的算法性能并没有明显的提升。

**2.近似联合训练法**

训练网络的时候，将RPN和Fast R-CNN合并在一起，共用底层的卷积层。每次SGD迭代，网络前向传播预测的候选框被视为固定的，用作训练Fast R-CNN的候选框。反向传播时，共享的卷积层将RPN和Fast R-CNN回传的梯度相融合，使用融合后的梯度进行网络参数更新。虽然这种方法实现起来比较容易，但只是一种近似的处理，各个候选框的坐标其实也是变量，也需要参与反向传播的求导过程。在实际实验中，通过近似联合训练法可以得到与交替训练法近似的结果，但耗费的时间只有交替训练法的25%～50%，因此也是一种经常被采用的方法。

**3.非近似联合训练法**

如前所述，RPN预测的边界框也是输入的函数。Fast R-CNN中的RoI池层接受卷积特征和预测的边界框作为输入，因此一个理论上有效的反向传播求解器还应该包含梯度、、框坐标。上述近似联合训练忽略了这些梯度。在非近似联合训练解决方案中，我们需要一个可微的RoI池层。这是一个非常重要的问题，解决方案可以由其他论文中开发的“RoI扭曲”层给出，这超出了本文的范围。