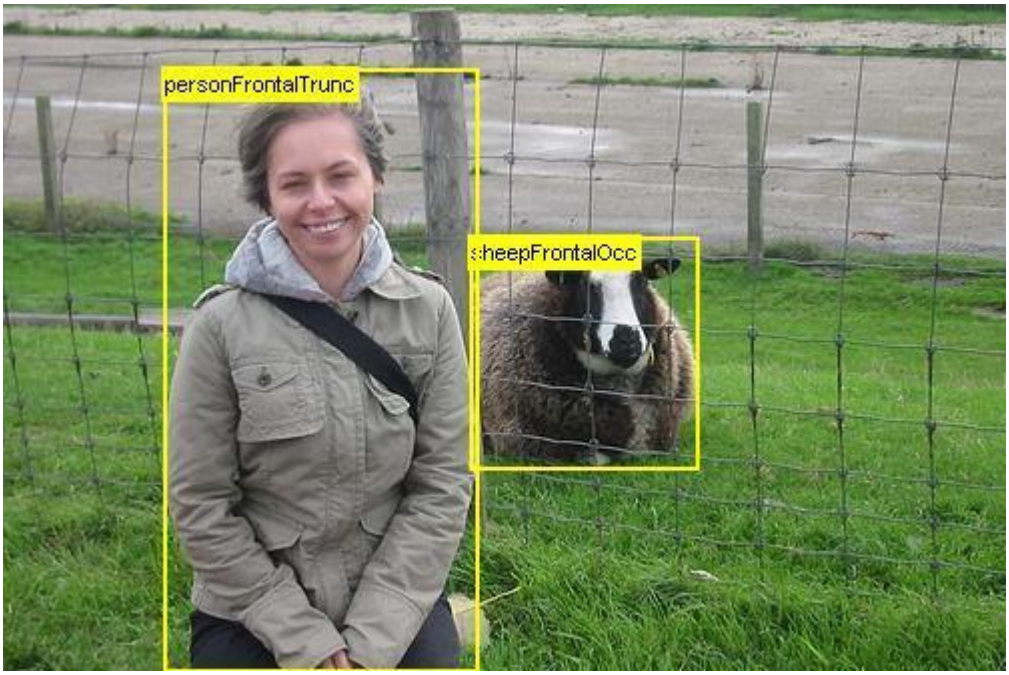
从 R-CNN 到 Mask R-CNN 综述

自从 2012 年的 ILSVRC 竞赛中基于 CNN 的方法一鸣惊人之后，CNN 已成为图像分类、检测和分割的神器。其中在图像检测的任务中，R-CNN 系列是一套经典的方法，从最初的 R-CNN 到后来的 Fast R-CNN， Faster R-CNN 和今年的 Mask R-CNN, 我们可以看到 CNN 在图像检测中是如何一点一点提高的。和本文来一道回顾 R-CNN 家族的发展史，了解这些方法的演变和这个演变过程中的那些富有创意的想法。

R-CNN 系列的四篇文章如下：

1. R-CNN: https://arxiv.org/abs/1311.2524
2. Fast R-CNN: https://arxiv.org/abs/1504.08083
3. Faster R-CNN: https://arxiv.org/abs/1506.01497
4. Mask R-CNN: https://arxiv.org/abs/1703.06870

图像的检测任务是从一个复杂场景的图像中找到不同的物体，并且给出各个物体的边界框。图像检测的三个著名的数据集是 PASCAL VOC，ImageNet 和微软 COCO. PASCAL VOC 包含 20 个物体的类别，而 ImageNet 包含一千多种物体类别，COCO 有 80 中物体类别和 150 万个物体实例。



PASCAL VOC 目标检测

COCO 目标检测和实例分割

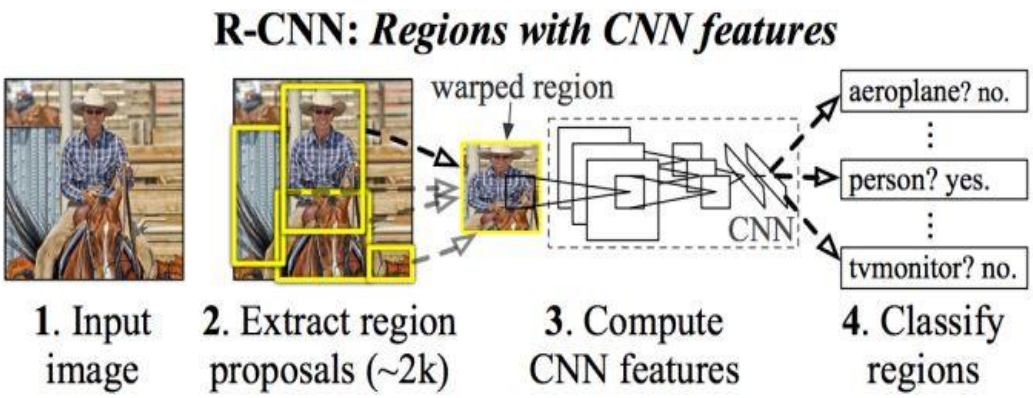
1， R-CNN

R-CNN 的思路是很直观的三步曲：1，得到若干候选区域；2， 对每个候选区域分别用 CNN 分类；3，对每个候选区域分别进行边框预测。

在 R-CNN 出现之前，目标检测的流行思路是先从图像中得到一些候选区域，再从候选区域中提取一些特征，然后利用一个分类器对这些特征进行分类。分类的结果和候选区域的边界框就可以作为目标检测的输出。一种得到候选区域的方法是 Selective Search, 该方法可以得到不同尺度的候选区域，每个候选区域是一个联通的区域。如下图中，左边得到的是较小的候选区域，右边是较大的候选区域，在该尺度下包含了整个人的区域。

Selective Search 得到不同尺度的候选区域

R-CNN 的想法是，既然 CNN 在图像分类任务中的表现很好，可以自动学习特征，何不用它来对每个候选区域进行特征提取呢？于是 R-CNN 在 Selective Search 得到的候选区域的基础上，将每个候选区域缩放到一个固定的大小，并且作为 AlexNet（ImageNet 2012 图像分类竞赛的冠军）的输入，依次对这些区域进行特征提取，然后再使用支持向量机对各个区域的特征分别进行分类。R-CNN 的过程如下：



R-CNN 示意图

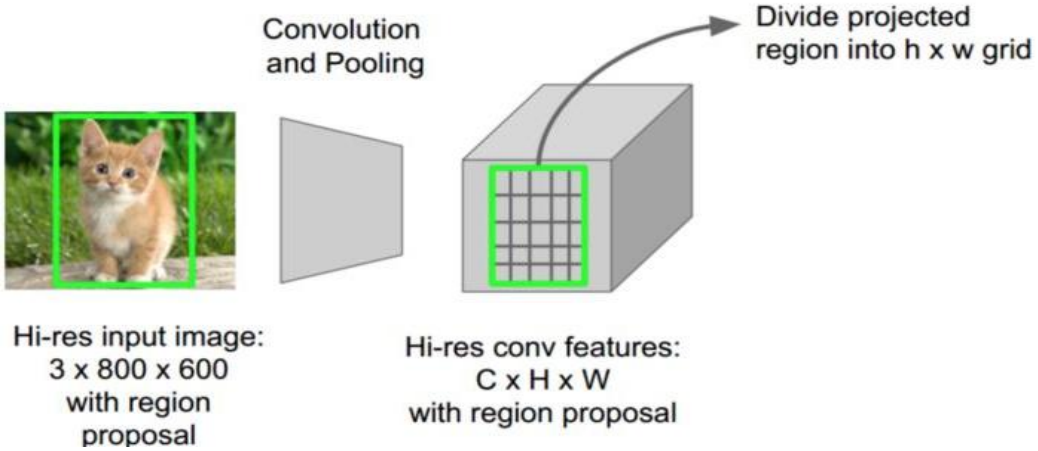
这样就能得到一个检测的结果了。但是 Slelective Search 得到的候选区域并不一定和目标物体的真实边界相吻合，因此 R-CNN 提出对物体的边界框做进一步的调整，使用一个线性回归器来预测一个候选区域中物体的真实边界。该回归器的输入就是候选区域的特征，而输出是边界框的坐标。

R-CNN 的效果很不错，比 VOC 2012 的最好的结果提高了 30%的准确度。但是问题就是太慢，主要有三方面原因：1，候选区域的生成是一个耗时的过程；2，对候选区域特征提取需要在单张图像上使用 AlexNet 2000 多次； 3， 特征提取、图像分类、边框回归是三个独立的步骤，要分别进行训练，测试过程中的效率也较低。

2， Fast R-CNN

为了解决 R-CNN 中效率低的问题，Fast R-CNN 想，一张图像上要使用 AlexNet 2000 多次来分别得到各个区域的特征，但很多区域都是重合的，可否避免这些重复计算，只在一张图像上使用一次 AlexNet，然后再得到不同区域的特征呢？

于是Fast R-CNN提出了一个ROI Pooling的方法，先对输入图像使用一次CNN前向计算，得到整个图像的特征图，再在这个特征图中分别取提取各个候选区域的特征。由于候选区域的大小不一样，而对应的特征需要要具有固定的大小，因此该方法对各个候选区域分别使用POI Pooling, 其做法是：假设第 i 个候选区域ROI的大小为 $h_i \times w_i$，要使输出的大小为 $h \times w$，那么就将该ROI分成 $h \times w$ 个格子，每一个格子的大小为 $(h_i/h) \times (w_i/w)$，然后对每一格子使用max-pooling得到大小为 $h \times w$ 的特征图像。

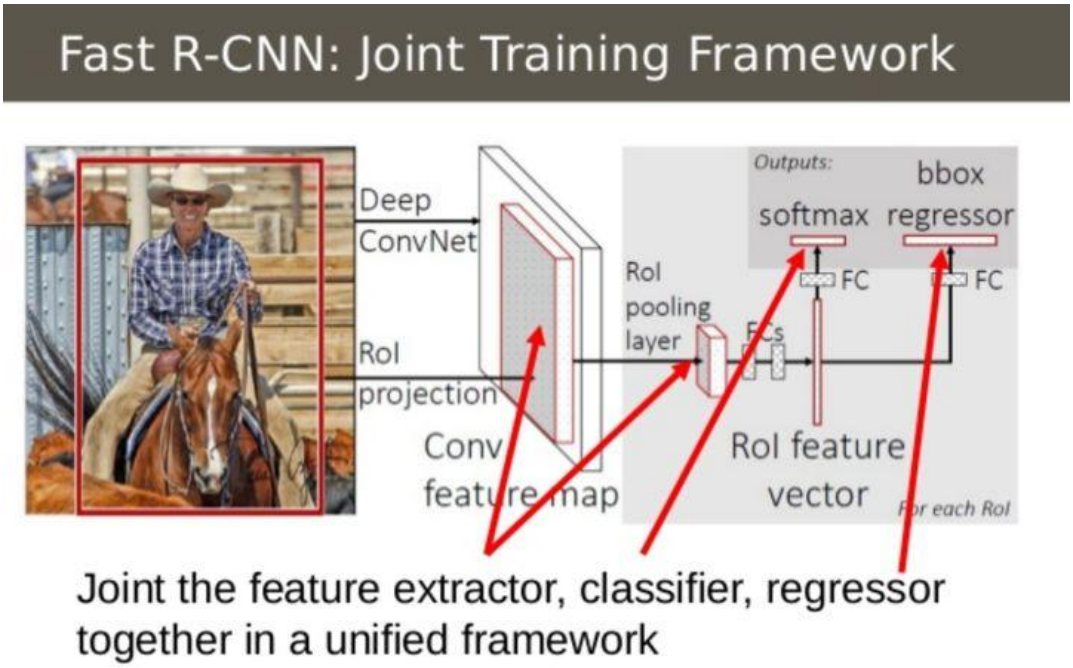


Fast R-CNN 示意图，将每个候选区域分成 h x w 大小的格子做 pooling

Fast R-CNN 的第二点创意是把之前独立的三个步骤（特征提取、分类和回归）放到一个统一的网络结构中。该网络结构同时预测一个候选区域的物体类别和该物体的边界框，使用两个全连接的输出层分别进行类别预测和边框预测(如下图所示)，将这两个任务进行同时训练，利用一个联合代价函数：

$$L(p,u,t^u,v)=L_{cls}(p,u)+\lambda[u\geq 1]L_{reg}(t^u,v)$$

公式中的两项分别是 classification loss 和 regression loss。

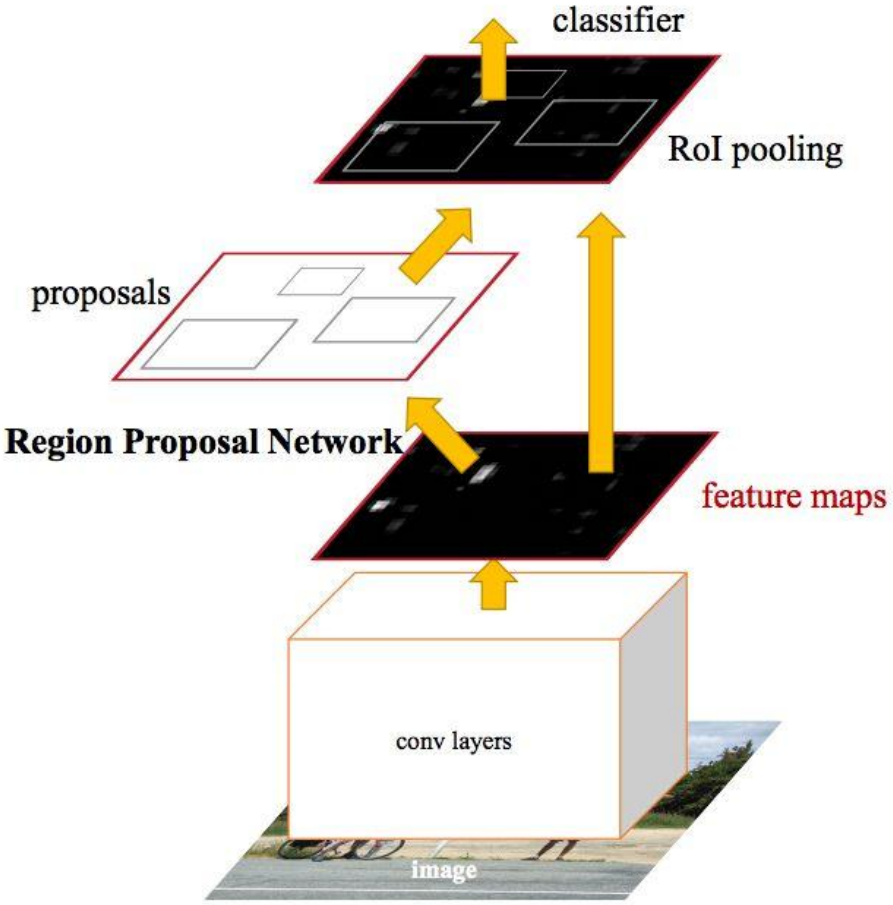


特征提取-分类-回归联合网络

使用 VGG16 作为特征提取网络，Fast R-CNN 在测试图像上的处理时间比 R-CNN 快了 200 多倍，并且精度更高。如果不考虑生成候选区域的时间，可以达到实时检测。生成候选区域的 Selective Search 算法处理一张图像大概需要 2s 的时间，因此成为该方法的一个瓶颈。

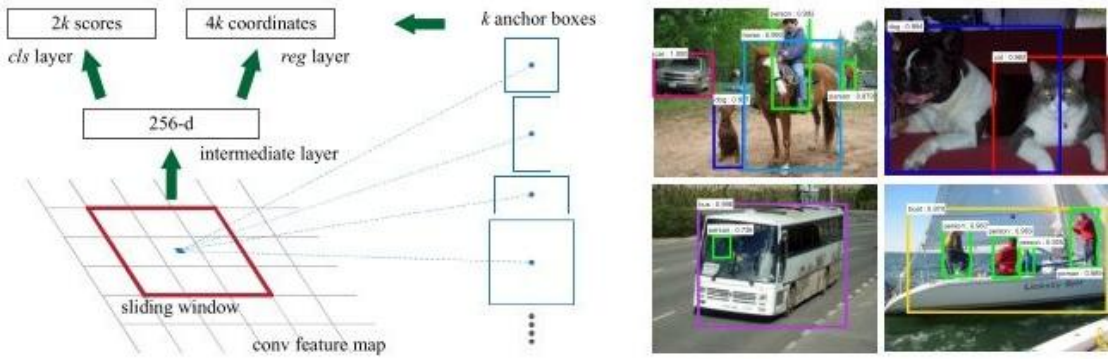
3， Faster R-CNN

上面两种方法都依赖于 Selective Search 生成候选区域，十分耗时。考虑到 CNN 如此强大，Faster R-CNN 提出使用 CNN 来得到候选区域。假设有两个卷积神经网络，一个是区域生成网络，得到图像中的各个候选区域，另一个是候选区域的分类和边框回归网路。这两个网络的前几层都要计算卷积，如果让它们在这几层共享参数，只是在末尾的几层分别实现各自的特定的目标任务，那么对一幅图像只需用这几个共享的卷积层进行一次前向卷积计算，就能同时得到候选区域和各候选区域的类别及边框。



Faster R-CNN:在卷积后的特征图上使用 Region Proposal Network

候选区域生成网络（Region Proposal Network, RPN）如下，先通过对输入图像的数层卷积得到一个特征图像，然后在特征图像上生成候选区域。它使用一个 $n \times n$（$n = 3$）的滑动窗口，将局部的特征图像转换成一个低维特征，预测 k 个的区域（cls层，$2k$ 个输出）是否为候选区域和对应的 k 个边框（reg层，$4k$ 个输出）。这里的 k 个区域被称为锚（anchor），对应着与滑动窗口具有相同的中心的不同大小和不同长宽比的矩形框。假设卷积后的特征图像大小为 $W \times H$，那么一共有 WHk 个锚。这种特征提取和候选区域生成的方法具有位移不变性。



对一个滑动窗口中的 anchor 预测候选区域

使用 RPN 得到候选区域后，对候选区域的分类和边框回归仍然使用 Fast R-CNN。这两个网络使用共同的卷积层。 由于 Fast R-CNN 的训练过程中需要使用固定的候选区域生成方法，不能同时对 RPN 和 Fast R-CNN 使用反向传播算法进行训练。该文章使用了四个步骤完成训练过程：1，单独训练 RPN；2，使用步骤中 1 得到的区域生成方法单独训练 Fast R-CNN; 3，使用步骤 2 得到的网络作为初始网络训练 RPN；4， 再次训练 Fast R-CNN，微调参数。

Faster R-CNN 的精度和 Fast R-CNN 差不多，但是训练时间和测试时间都缩短了 10 倍。

4， Mask R-CNN

Faster R-CNN 在物体检测中已达到非常好的性能，Mask R-CNN 在此基础上更进一步：得到像素级别的检测结果。 对每一个目标物体，不仅给出其边界框，并且对边界框内的各个像素是否属于该物体进行标记。

Mask R-CNN: 像素级别的目标检测

Mask R-CNN 利用 Faster R-CNN 中已有的网络结构，再添加了一个头部分支，使用 FCN 对每个区域做二值分割。

Mask R-CNN 还提出了两个小的改进使分割的结果更好。第一，对各个区域分割时，解除不同类之间的耦合。假设有 K 类物体，一般的分割方法直接预测一个有 K 个通道的输出，其中每个通道代表对应的类别。而 Mask R-CNN 预测 K 个有 2 个通道（前景和背景）的输出，这样各个类别的预测是独立的。第二，Faster R-CNN 中使用 ROI Pooling 之前的取整操作使特征图中所使用的 ROI 与原图中 ROI 的位置不完全对应。在 Fast 和 Faster R-CNN 中，在使用 ROI Pooling 之前的特征图大小是原图的 1/16。

原图的1/16。假设一个ROI在原图中的横坐标范围是 w_0 到 w_1 ,纵坐标范围是 h_0 到 h_1 ，那么在特征图中该ROI的横坐范围是 $w_0/16$ 到 $w_1/16$ ，纵坐标范围是 $h_0/16$ 到 $h_1/16$ ，取整得到 $[w_0/16]$ ， $[w_1/16]$ ， $[h_0/16]$ ， $[h_1/16]$ 。这样会导致Pooling前的特征图中的

这样会导致 Pooling 前的特征图中的 ROI 与原图中的 ROI 没有完全对齐。这不会对分类和边框回归造成大的影响，但是会影响分割的结果，因为没有保持空间对应关系。Faster R-CNN 不取整，使用双线性插值得到 与原图中 ROI 对应的特征图中的区域，保留坐标的对应关系，该方法称作 ROIAlign.

上述几种方法的代码：

R-CNN

- Caffe 版本：rbgirshick/rcnn

Fast R-CNN

- Caffe 版本： rbgirshick/fast-rcnn

Faster R-CNN

- Caffe 版本: https://github.com/rbgirshick/py-faster-rcnn
- PyTorch 版本: https://github.com/longcw/faster_rcnn_pytorch
- MatLab 版本: https://github.com/ShaoqingRen/faster_rcnn

Mask R-CNN

- PyTorch 版本: https://github.com/felixgwu/mask_rcnn_pytorch
- TensorFlow 版本: https://github.com/CharlesShang/FastMaskRCNN

注：R-CNN， Fast R-CNN 和 Faster R-CNN 已在之前的文章总结过，这里添上 Mask R-CNN。更多目标检测的方法在《深度卷积神经网络在目标检测中的进展》这篇文章中。另外，《A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN》这篇文章也做了详细的介绍。