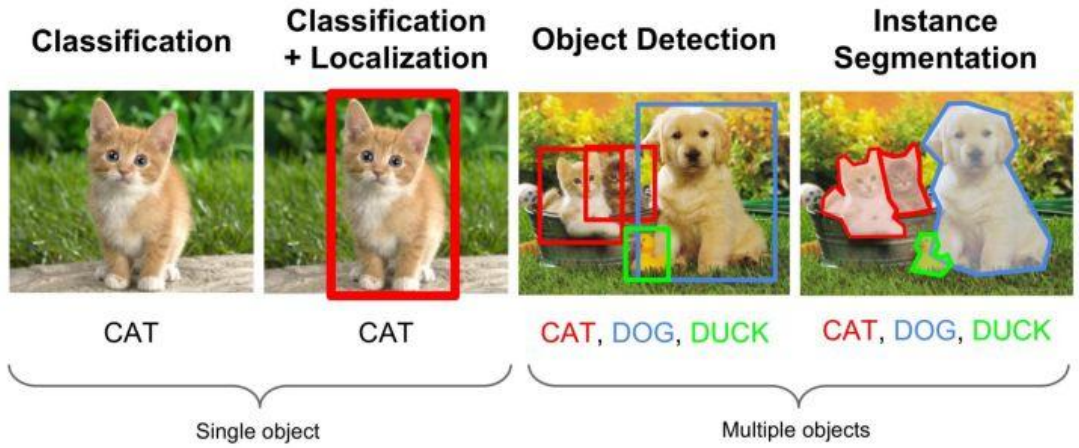
综述|基于深度学习的目标检测(一)

概述

图像分类，检测及分割是计算机视觉领域的三大任务。图像分类模型（详情见这里）是将图像划分为单个类别，通常对应于图像中最突出的物体。但是现实世界的很多图片通常包含不只一个物体，此时如果使用图像分类模型为图像分配一个单一标签其实是非常粗糙的，并不准确。对于这样的情况，就需要目标检测模型，目标检测模型可以识别一张图片的多个物体，并可以定位出不同物体（给出边界框）。目标检测在很多场景有用，如无人驾驶和安防系统。

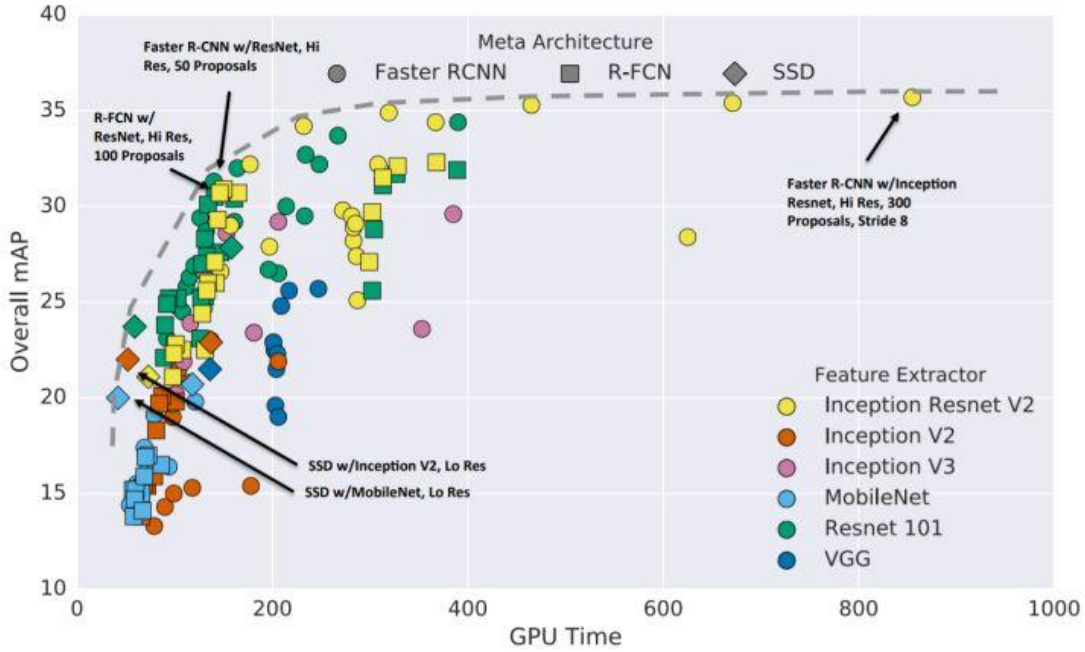


图像分类，目标检测与实例分割的对比

目前主流的目标检测算法主要是基于深度学习模型，其可以分成两大类：（1）two-stage 检测算法，其将检测问题划分为两个阶段，首先产生候选区域（region proposals），然后对候选区域分类（一般还需要对位置精修），这类算法的典型代表是基于 region proposal 的 R-CNN 系算法，如 R-CNN，Fast R-CNN，Faster R-CNN 等；（2）one-stage 检测算法，其不需要 region proposal 阶段，直接产生物体的类别概率和位置坐标值，比较典型的算法如 YOLO 和 SSD。目标检测模型的主要性能指标是检测准确度和速度，对于准确度，目标检测要考虑物体的定位准确性，而不单单是分类准确度。一般情况下，two-stage 算法在准确度上有优势，而 one-stage 算法在速度上有优势。不过，随着研究的发展，两类算法都在两个方面做改进。

Google 在 2017 年开源了 TensorFlow Object Detection API，并对主流的 Faster R-CNN，R-FCN 及 SSD 三个算法在 MS COCO 数据集上的性能做了细致对比（见 Huang et al. 2017），如下图所示。近期，Facebook 的 FAIR 也开源了基于 Caffe2 的目标检测平台 Detectron，其实现了最新的 Mask R-CNN，RetinaNet 等检测算法，并且给出了这些算法的 Baseline Results。

不得不说，准确度（accuracy）和速度（speed）是一对矛盾体，如何更好地平衡它们一直是目标检测算法研究的一个重要方向。



Faster R-CNN，R-FCN 及 SSD 算法在 MS COCO 数据集上的性能对比

在这篇长文中，我们将对最新的目标检测算法做一个综述。在介绍目标检测算法之前，先简单介绍目标检测领域常用的数据集以及性能指标。

数据集和性能指标

目标检测常用的数据集包括 PASCAL VOC，ImageNet，MS COCO 等数据集，这些数据集用于研究者测试算法性能或者用于竞赛。目标检测的性能指标要考虑检测物体的位置以及预测类别的准确性，下面我们会说到一些常用的性能评估指标。

数据集

PASCAL VOC（The PASCAL Visual Object Classification）是目标检测，分类，分割等领域一个有名的数据集。从 2005 到 2012 年，共举办了 8 个不同的挑战赛。PASCAL VOC 包含约 10,000 张带有边界框的图片用于训练和验证。但是，PASCAL VOC 数据集仅包含 20 个类别，因此其被看成目标检测问题的一个基准数据集。

ImageNet 在 2013 年放出了包含边界框的目标检测数据集。训练数据集包含 500,000 张图片，属于 200 类物体。由于数据集太大，训练所需计算量很大，因而很少使用。同时，由于类别数也比较多，目标检测的难度也相当大。2014 ImageNet 数据集和 2012 PASCAL VOC 数据集的对比在这里。

另外一个有名的数据集是 Microsoft 公司（见 T.-Y.Lin and al. 2015)建立的 MS COCO（Common Objects in COntext）数据集。这个数据集用于多种竞赛：图像标题生成，目标检测，关键点检测和物体分割。对于目标检测任务，COCO 共包含 80 个类别，每年大赛的训练和验证数据集包含超过 120,000 个图片，超过 40,000 个测试图片。测试集最近被划分为两类，一类是 test-dev 数据集用于研究者，一类是 test-challenge 数据集用于竞赛者。测试集的标签数据没有公开，以避免在测试集上过拟合。在 COCO 2017 Detection Challenge 中，旷视科技团队凭借提出的 Light-Head R-CNN 模型夺得冠军（AP 为 0.526 ），看来还是 two-stage 算法准确度更胜一筹。

2015 COCO 数据集的分割实例. 来源: T.-Y.Lin and al. (2015)

Name	# Images (trainval)	# Classes	Last updated
ImageNet	450k	200	2015
COCO	120K	80	2014
Pascal VOC	12k	20	2012
Oxford-IIIT Pet	7K	37	2012
KITTI Vision	7K	3	2014

目标检测的主流数据集. 来源: https://tryolabs.com/blog/

性能指标

目标检测问题同时是一个回归和分类问题。首先，为了评估定位精度，需要计算 IoU（Intersection over Union，介于 0 到 1 之间），其表示预测框与真实框（ground-truth box）之间的重叠程度。IoU 越高，预测框的位置越准确。因而，在评估预测框时，通常会设置一个 IoU 阈值（如 0.5），只有当预测框与真实框的 IoU 值大于这个阈值时，该预测框才被认定为真阳性（True Positive, TP），反之就是假阳性（False Positive，FP）。

对于二分类，AP（Average Precision）是一个重要的指标，这是信息检索中的一个概念，基于 precision-recall 曲线计算出来，详情见这里。对于目标检测，首先要单独计算各个类别的 AP 值，这是评估检测效果的重要指标。取各个类别的 AP 的平均值，就得到一个综合指标 mAP（Mean Average Precision），mAP 指标可以避免某些类别比较极端化而弱化其它类别的性能这个问题。

对于目标检测，mAP 一般在某个固定的 IoU 上计算，但是不同的 IoU 值会改变 TP 和 FP 的比例，从而造成 mAP 的差异。COCO 数据集提供了官方的评估指标，它的 AP 是计算一系列 IoU 下（0.5:0.05:0.9，见说明）AP 的平均值，这样可以消除 IoU 导致的 AP 波动。其实对于 PASCAL VOC 数据集也是这样，Facebook 的 Detectron 上的有比较清晰的实现。
除了检测准确度，目标检测算法的另外一个重要性能指标是速度，只有速度快，才能实现实时检测，这对一些应用场景极其重要。评估速度的常用指标是每秒帧率（Frame Per Second，FPS），即每秒内可以处理的图片数量。当然要对比 FPS，你需要在同一硬件上进行。另外也可以使用处理一张图片所需时间来评估检测速度，时间越短，速度越快。

R-CNN

R-CNN（R. Girshick et al., 2014）是基于 region proposal 方法的目标检测算法系列开山之作，其先进行区域搜索，然后再对候选区域进行分类。在 R-CNN 中，选用 Selective search 方法（J.R.R. Uijlings and al. 2012）来生成候选区域，这是一种启发式搜索算法。它先通过简单的区域划分算法将图片划分成很多小区域，然后通过层级分组方法按照一定相似度合并它们，最后的剩下的就是候选区域（region proposals），它们可能包含一个物体。

Selective Search 方法：上面是分割结果，下面是候选框. 来源: J.R.R. Uijlings and al. (2012)

对于一张图片，R-CNN基于selective search方法大约生成2000个候选区域，然后每个候选区域被resize成固定大小（ 227×227 ）并送入一个CNN模型中，最后得到一个4096-d的特征向量。然后这个特征向量被送入一个多类别SVM分类器中，预测出候选区域中所含物体的属于每个类的概率值。每个类别训练一个SVM分类器，从特征向量中推断其属于该类别的概率大小。为了提升定位准确性，R-CNN最后又训练了一个边界框回归模型。训练样本为 (P, G) ，其中 $P = (P_x, P_y, P_w, P_h)$ 为候选区域，而 $G = (G_x, G_y, G_w, G_h)$ 为真实框， G 是与 P 的IoU最大的真实框（只使用IoU大于0.6的样本），回归器的目标值定义为：

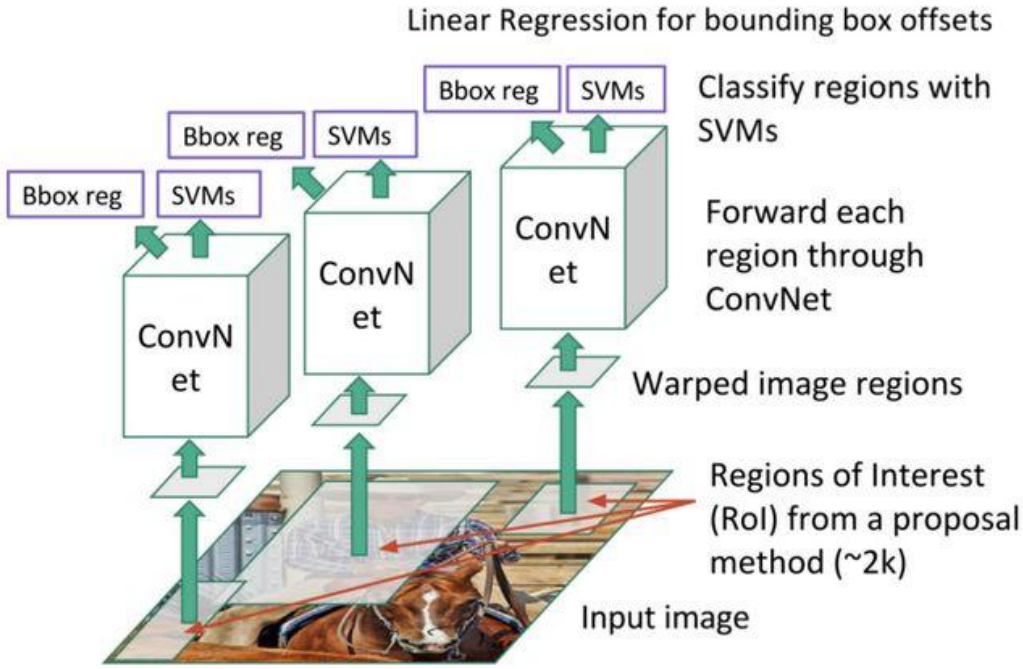
$$t_x = (G_x - P_x)/P_w, t_y = (G_y - P_y)/P_h$$

$$t_w = \log(G_w/P_w), t_h = \log(G_h/P_h)$$

在做预测时，利用上述公式可以反求出预测框的修正位置。R-CNN 对每个类别都训练了单独的回归器，采用最小均方差损失函数进行训练。

R-CNN 模型的训练是多管道的，CNN 模型首先使用 2012 ImageNet 中的图像分类竞赛数据集进行预训练。然后在检测数据集上对 CNN 模型进行 finetuning，其中那些与真实框的 IoU 大于 0.5 的候选区域作为正样本，剩余的候选区域是负样本（背景）。共训练两个版本，第一版本使用 2012 PASCAL VOC 数据集，第二个版本使用 2013 ImageNet 中的目标检测数据集。最后，对数据集中的各个类别训练 SVM 分类器（注意 SVM 训练样本与 CNN 模型的 funetuning 不太一样，只有 IoU 小于 0.3 的才被看成负样本）。

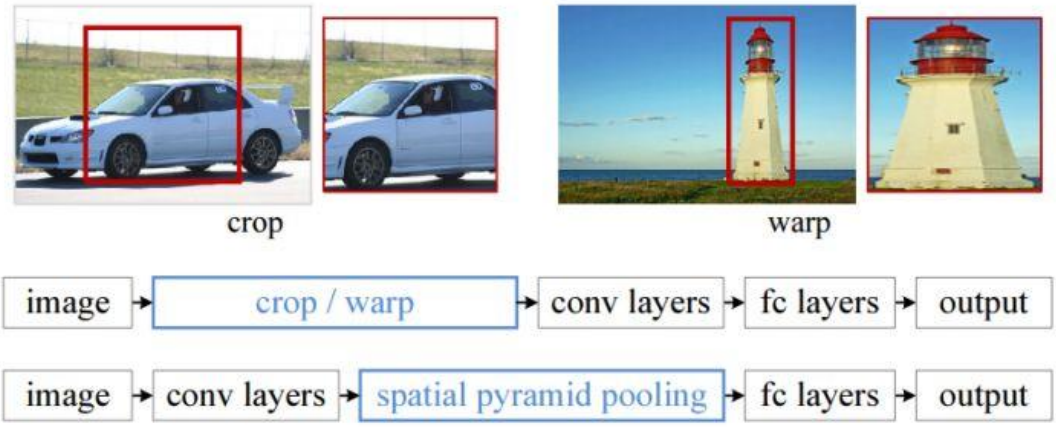
总体来看，R-CNN 是非常直观的，就是把检测问题转化为了分类问题，并且采用了 CNN 模型进行分类，但是效果却很好。最好的 R-CNN 模型在 2012 PASCAL VOC 数据集的 mAP 为 62.4%（比第二名高出了 22 个百分点），在 2013 ImageNet 上的 mAP 为 31.4%（比第二名高出 7.1 个百分点）。



R-CNN 模型结构图

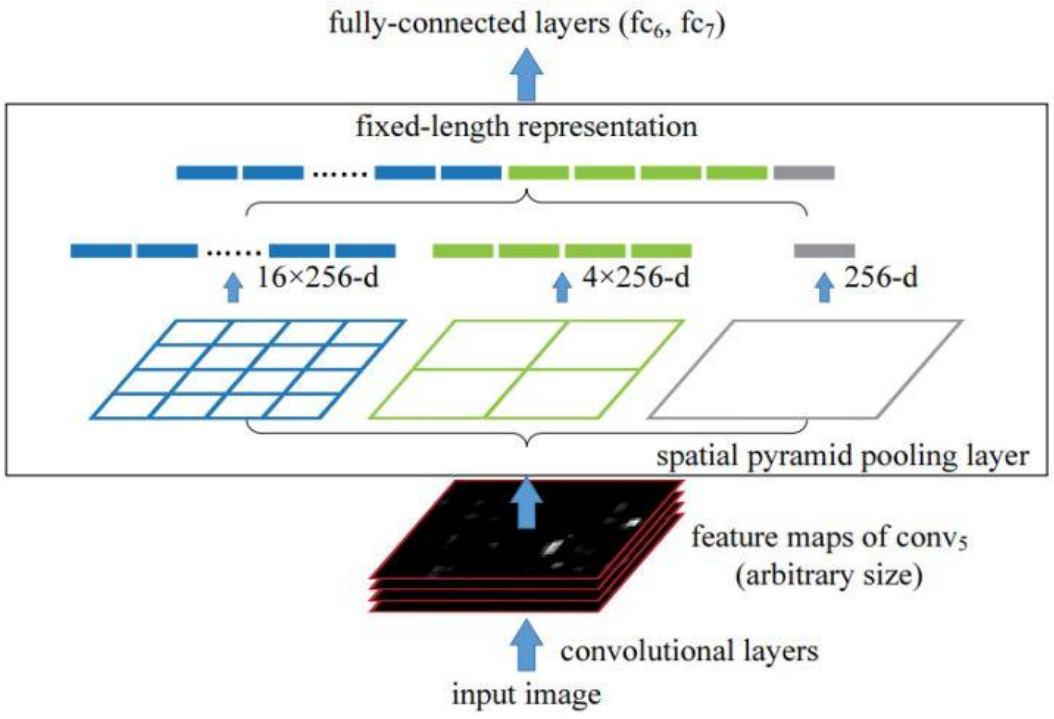
SPP-net

SPP-net（Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, He et al. 2014）提出的起因是解决图像分类中要求输入图片固定大小的问题，但是 SPP-net 中所提出的空间金字塔池化层（Spatial Pyramid Pooling Layer, SPP）可以和 R-CNN 结合在一起并提升其性能。采用深度学习模型解决图像分类问题时，往往需要图像的大小固定（比如 224X224 ），这并不是 CNN 层的硬性要求，主要原因在于 CNN 层提取的特征图最后要送入全连接层（如 softmax 层），对于变大小图片，CNN 层得到的特征图大小也是变化的，但是全连接层需要固定大小的输入，所以必须要将图片通过 resize, crop 或 wrap 等方式固定大小（训练和测试时都需要）。但是实际上真实的图片的大小是各种各样的，一旦固定大小可能会造成图像损失，从而影响识别精度。为了解决这个问题，SSP-net 在 CNN 层与全连接层之间插入了空间金字塔池化层来解决这个矛盾。



SPP-net 与普通网络的结构对比

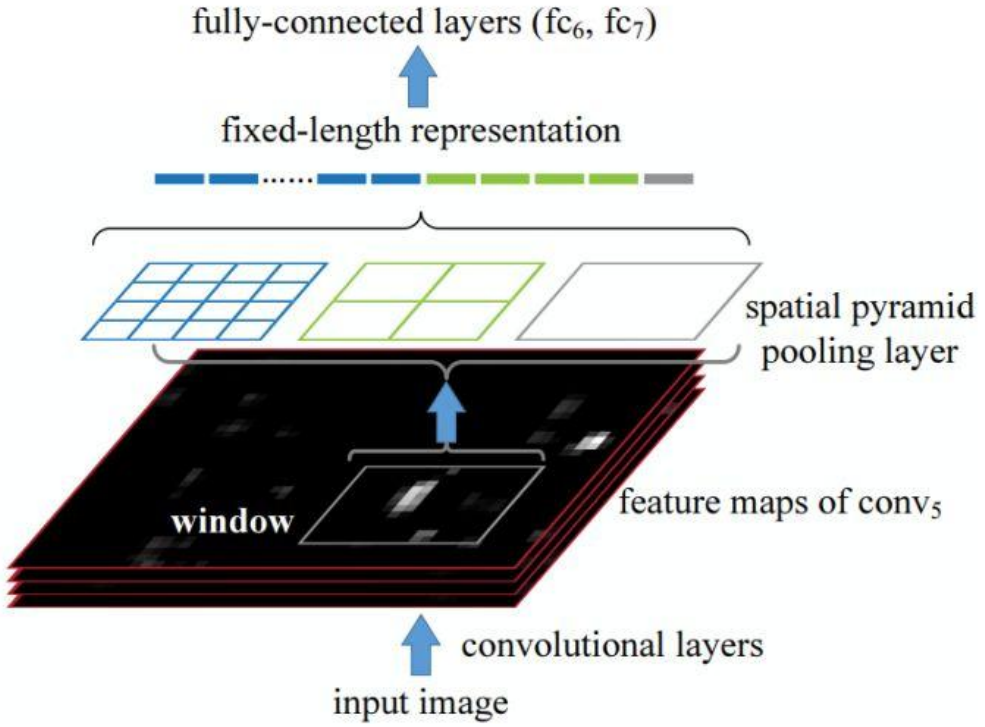
SPP层原理如下所所示，假定CNN层得到的特征图大小为 $a \times a$ （比如 13×13 ，随输入图片大小而变化），设定的金字塔尺度为 $n \times n$ bins（对于不同大小图片是固定的），那么SPP层采用一种滑动窗口池化，窗口大小 $win\_size = \lceil a/n \rceil$ ，步为 $stride = \lfloor a/n \rfloor$ ，采用max pooling，本质上将特征图均分为 $n \times n$ 个子区域，然后对各个子区域max pooling，这样不论输入图片大小，经过SPP层之后得到是固定大小的特征。一般设置多个金字塔级别，文中使用了 4×4 ， 2×2 和 1×1 三个尺度。每个金字塔都得一个特征，将它们连接在一起送入后面的全连接层即可，这样就解决了变大小图片输入的问题了。SPP-net在ImageNet ILSVRC 2014图像分类大赛中夺得了第三名。



SPP-net 中的空间金字塔池化层

那么 SPP-net 和 R-CNN 有什么关系呢？在 R-CNN 中，由于每个候选区域大小是不同，所以需要先 resize 成固定大小才能送入 CNN 网络，SPP-net 正好可以解决这个问题。继续上前一步，就是 R-CNN 每次都要挨个使用 CNN 模型计算各个候选区域的特征，这是极其费时的，不如直接将整张图片送入 CNN 网络，然后抽取候选区域的对应的特征区域，采用 SPP 层，这样

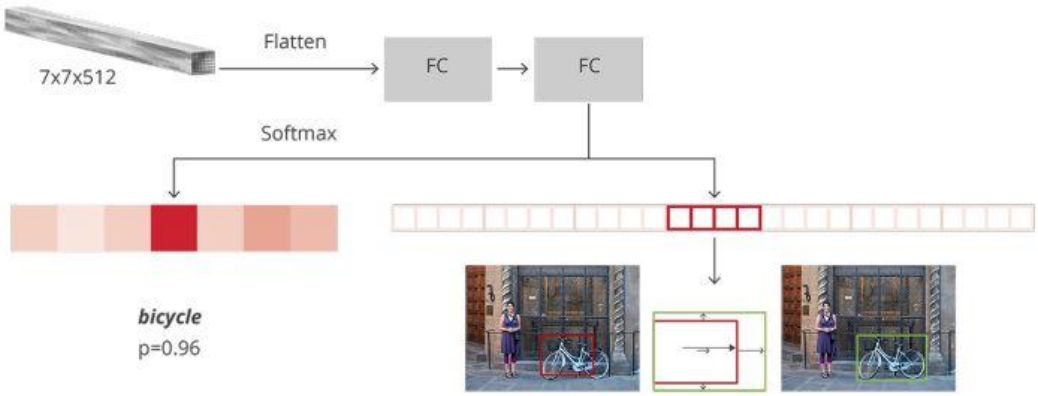
可以大大减少计算量，并提升速度。基于 SPP 层的 R-CNN 模型在准确度上提升不是很大，但是速度却比原始 R-CNN 模型快 24-102 倍。这也正是接下来 Fast R-CNN 所改进的方向。



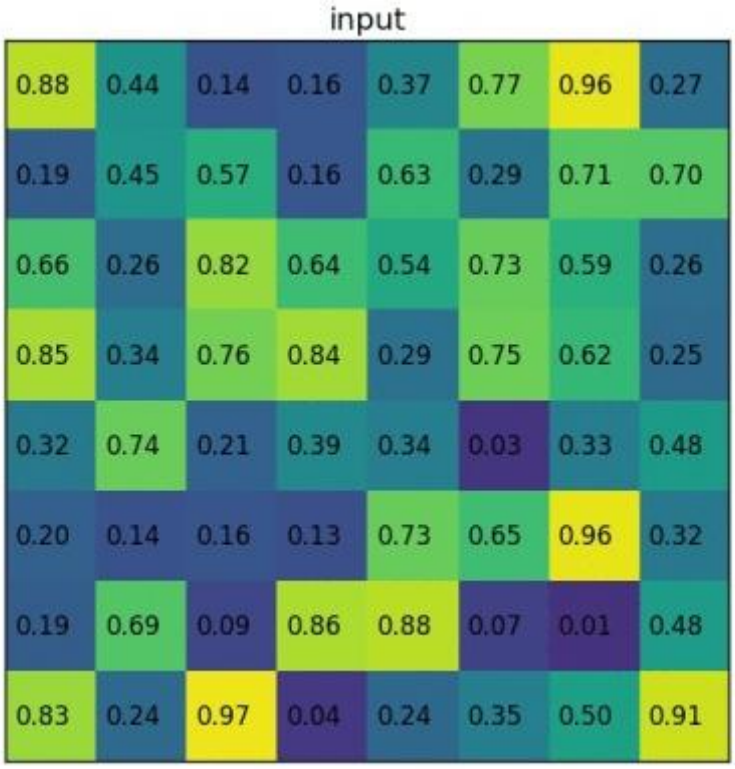
SPP 层用于 R-CNN 模型

Fast R-CNN

Fast R-CNN（Fast Region-based Convolutional Network, R. Girshick 2015）的提出主要是为了减少候选区域使用 CNN 模型提取特征向量所消耗的时间，其主要借鉴了 SPP-net 的思想。在 R-CNN 中，每个候选区域都要单独送入 CNN 模型计算特征向量，这是非常费时的，而对于 Fast R-CNN，其 CNN 模型的输入是整张图片，然后结合 RoIs（Region of Interests）pooling 和 Selective Search 方法从 CNN 得到的特征图中提取各个候选区域的所对应的特征。对于每个候选区域，使用 RoI pooling 层来从 CNN 特征图中得到一个固定长和宽的特征图（长和宽是超参数，文中选用 7X7 ），RoI pooling 的原理很简单，其根据候选区域按比例从 CNN 特征图中找到对应的特征区域，然后将其分割成几个子区域（根据要输出的特征图的大小），然后在每个子区域应用 max pooling，从而得到固定大小的特征图，这个过程是可导的（见 RoI pooling 层的 Caffe 官方实现，RoI pooling 层在大部分深度学习框架中是没有官方实现的，需要自己扩展，Tensorflow 上的开源实现可以参考 deepsense-ai/roi-pooling，但是在 TensorFlow 中可以基于一种 crop+resize+pooling 的方式来简化实现，可以参考一下 Luminoth 上的实现），RoI pooling 层相比 SPP 层看起来主要是只使用一个金字塔级别。然后 RoI pooling 层得到的特征图送入几个全连接层中，并产生新的特征向量，这些特征向量分别用于一个 softmax 分类器（预测类别）和一个线性回归器上（用于调整边界框位置）来进行检测。在实现上是使用两个不同的全连接层，第一个全连接层有 N+1 个输出（是 N 类别总数， 1 是背景），表示各个类别的概率值；第二个全连接层有 4N 个输出，表示坐标回归值 $(T_{X,}\ t_y,\ t_w,\ t_h)$，这个与 R-CNN 是一样的，每个类别都预测 4 个位置坐标值。



Fast R-CNN 的分类与回归预测. 来源：https://tryolabs.com/blog/

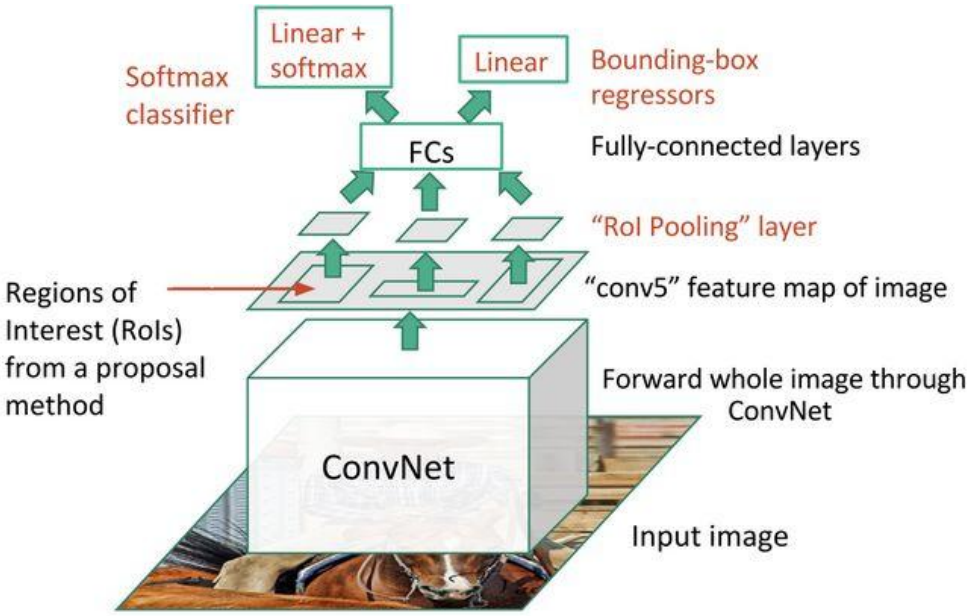


RoI pooling 原理图，特征图大小为 8x8，候选区域为 5x7，输出 2x2. 来源：https://blog.deepsense.ai/region-of-interest-pooling-explained/

Fast R-CNN 与 R-CNN 的另外的一个主要区别点是采用了 softmax 分类器而不是 SVM 分类器，而且训练过程是单管道的，因为 Fast R-CNN 将分类误差和定位误差合并在一起训练，定位误差采用 smooth L1 而不是 R-CNN 中的 L2。这里说点题外话，就是 R-CNN 训练是多管道的，除了对 CNN 模型预训练，R-CNN 还先对 CNN 模型 funetuning，使用的是 softmax 分类器，但是最后却又训练 SVM 分类器（原因可以见原论文），直觉上感觉有点多此一举，所以现在 Fast R-CNN 直接采用 softmax 分类器了。Fast R-CNN 训练采用 mini-batch sampling，每个 mini-batch 大小为 128，从 N=2 个图片中构建，其中 25%来自正样本（IoU>=0.5）,75%从负样本中抽样得到（背景，IoU $\in$[0.1, 0.5] ），这里的 IoU 阈值属于超参数。在图像分类中，当我们说 batch_size=32 时，是指的是 32 个图片，在 Fast R-CNN 中并不是这样，因为一个图片含有很多 RoIs，每个 batch 使用的图片非常少（内存限制），所以有时候你会看到 Fast R-CNN 训练时直接从一个图片中构建 batch，这实现起来更容易一些。

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u > 0]L_{loc}(t^u, v)$$

最好的 Fast R-CNN 模型在 2007 PASCAL VOC 测试集上的 mAp 为 70%，在 2010 PASCAL VOC 测试集上的 mAP 为 68.8%，而在 2012 PASCAL VOC 测试集上的 mAP 为 68.4%，准确度相比 R-CNN 略有提升，其实主要是速度更快。

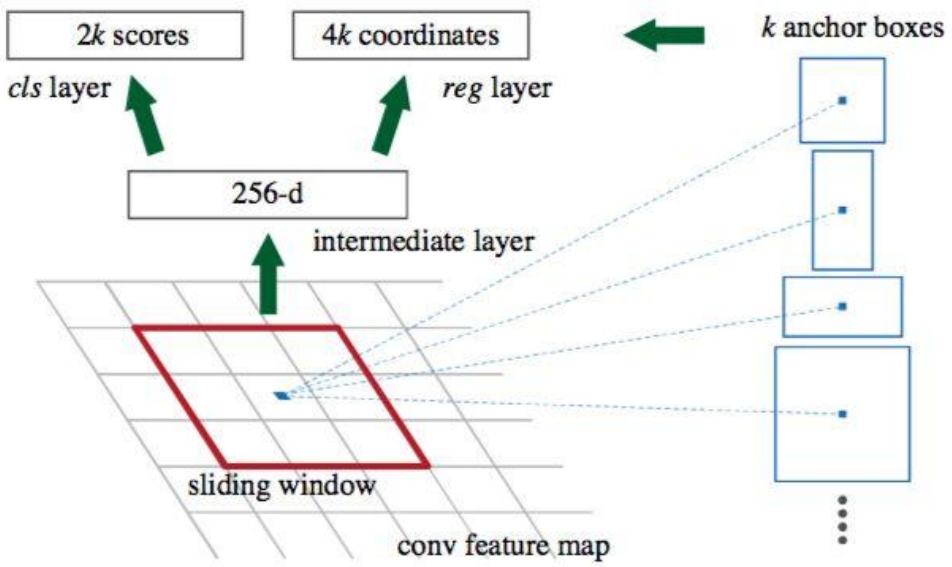


Fast R-CNN 模型结构图

Faster R-CNN

对于 Fast R-CNN，其仍然需要 selective search 方法来生产候选区域，这是非常费时的。为了解决这个问题，Faster R-CNN 模型（The Faster Region-based Convolutional Network, S. Ren and al. 2016）引入了 RPN (Region Proposal Network)直接产生候选区域。Faster R-CNN 可以看成是 RPN 和 Fast R-CNN 模型的组合体，即 **Faster R-CNN = RPN + Fast R-CNN**。

对于 RPN 网络，先采用一个 CNN 模型（一般称为特征提取器）接收整张图片并提取特征图。然后在这个特征图上采用一个 N X N （文中是 3 X 3 ）的滑动窗口，对于每个滑窗位置都映射一个低维度的特征（如 256-d）。然后这个特征分别送入两个全连接层，一个用于分类预测，另外一个用于回归。对于每个窗口位置一般设置 k 个不同大小或比例的先验框（anchors, default bounding boxes），这意味着每个位置预测 k 个候选区域（region proposals）。对于分类层，其输出大小是 2k ，表示各个候选区域包含物体或者是背景的概率值，而回归层输出 4k 个坐标值，表示各个候选区域的位置（相对各个先验框）。对于每个滑窗位置，这两个全连接层是共享的。因此，RPN 可以采用卷积层来实现：首先是一个 n x n 卷积得到低维特征，然后是两个 1 x 1 的卷积，分别用于分类与回归。



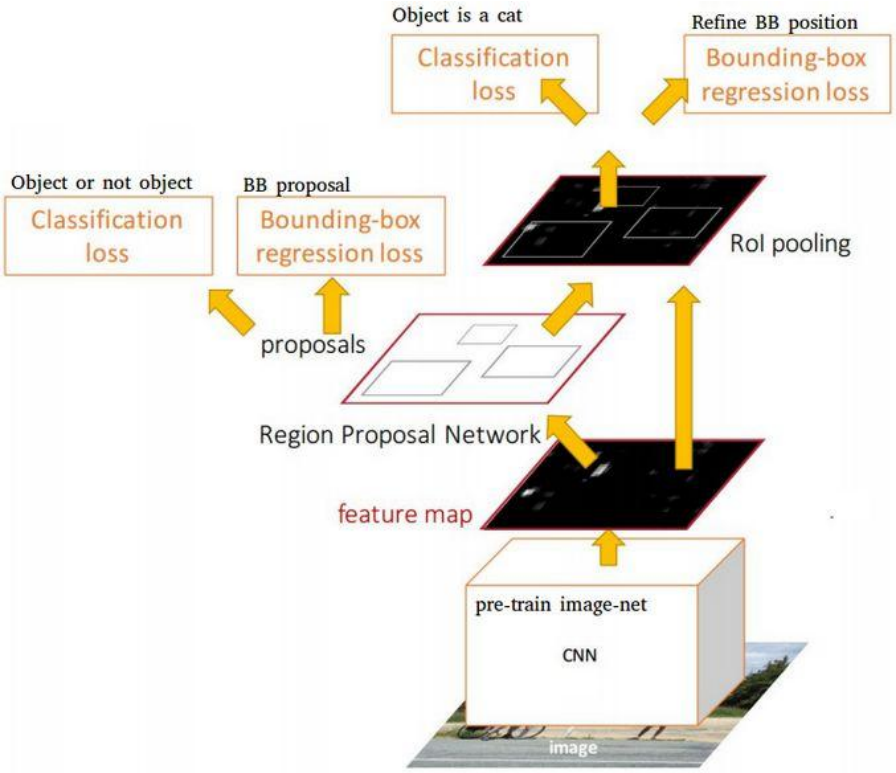
RPN 架构图

可以看到 RPN 采用的是二分类，仅区分背景与物体，但是不预测物体的类别，即 class-agnostic。由于要同时预测坐标值，在训练时，要先将先验框与 ground-truth box 进行匹配，原则为：（1）与某个 ground-truth box 的 IoU 最高的先验框；（2）与某个 ground-truth box 的 IoU 值大于 0.7 的先验框，只要满足一个，先验框就可以匹配一个 ground-truth，这样该先验框就是正样本（属于物体），并以这个 ground-truth 为回归目标。对于那些与任何一个 ground-truth box 的 IoU 值都低于 0.3 的先验框，其认为是负样本。RPN 网络是可以单独训练的，并且单独训练出来的 RPN 模型给出很多 region proposals。由于先验框数量庞大，RPN 预测的候选区域很多是重叠的，要先进行 NMS(non-maximum suppression，IoU 阈值设为 0.7）操作来减少候选区域的数量，然后按照置信度降序排列，选择 top-N 个 region proposals 来用于训练 Fast R-CNN 模型。RPN 的作用就是代替了 Selective search 的作用，但是速度更快，因此 Faster R-CNN 无论是训练还是预测都可以加速。

Faster R-CNN 模型采用一种 4 步迭代的训练策略：（1）首先在 ImageNet 上预训练 RPN，并在 PASCAL VOC 数据集上 finetuning；（2）使用训练的 PRN 产生的 region proposals 单独训练一个 Fast R-CNN 模型，这个模型也先在 ImageNet 上预训练；（3）用 Fast R-CNN 的 CNN 模型部分（特征提取器）初始化 RPN，然后对 RPN 中剩余层进行 finetuning，此时 Fast R-CNN 与 RPN 的特征提取器是共享的；（4）固定特征提取器，对 Fast R-CNN 剩余层进行 finetuning。这样经过多次迭代，Fast R-CNN 可以与 RPN 有机融合在一起，形成一个统一的网络。其实还有另外一中近似联合训练策略，将 RPN 的 2 个 loss 和 Fast R-CNN 的 2 个 loss 结合在一起，然后共同训练。注意这个过程，Fast R-CNN 的 loss 不对 RPN 产生的 region proposals 反向传播，所以这是一种近似（如果考虑这个反向传播，那就是非近似联合训练）。应该来说，联合训练速度更快，并且可以训练出同样的性能。

最好的 Faster R-CNN 模型在 2007 PASCAL VOC 测试集上的 mAP 为 78.8% ，而在 2012 PASCAL VOC 测试集上的 mAP 为 75.9%。论文中还在 COCO 数据集上进行了测试。Faster R-CNN 中的某个模型可以比采用 selective search 方法的 Fast R-CNN 模型快 34 倍。可以看到，采用了 RPN 之后，无论是准确度还是速度，Faster R-CNN 模型均有很大的提升。Faster R-CNN 采用 RPN

代替启发式 region proposal 的方法，这是一个重大变革，后面的 two-stage 方法的研究基本上都采用这种基本框架，而且和后面算法相比，Faster R-CNN 在准确度仍然占据上风。



Faster R-CNN 模型结构图

参考文章

1. Review of Deep Learning Algorithms for Object Detection [主要参考]
2. Deep Learning for Object Detection: A Comprehensive Review
3. Faster R-CNN: Down the rabbit hole of modern object detection
4. Object detection: an overview in the age of Deep Learning
5. Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD
6. R-CNN
7. SPP-net
8. Fast R-CNN
9. Faster R-CNN