# Fast Region-based Convolutional Network method (Fast R-CNN) for object detection

为何有了 R-CNN 和 SPP-Net 之后还要提出 Fast RCNN（简称 FRCN）？因为前者有三个缺点

- 训练的时候，pipeline 是隔离的，先提 proposal，然后 CNN 提取特征，之后用 SVM 分类器，最后再做 bbox regression。FRCN 实现了从 CNN 到 bbox regression 的联合训练。
- 训练时间和空间开销大。RCNN 中 ROI 的运算开销大（每个都进行一次特征提取），所以 FRCN 用了整个 image 的训练方式来通过卷积的 share 特性来降低运算开销；RCNN 提取特征给 SVM 训练时候需要中间要大量的磁盘空间存放特征，FRCN 去掉了 SVM 这一步，所有的特征都暂存在显存中，就不需要额外的磁盘空间了。
- 测试时间开销大。依然是因为 ROI 的原因，这点 SPP-Net 已经改进，然后 FRCN 进一步通过 single scale testing 和 SVD 分解全连接来提速。

整体框架如 Figure 1，如果以 AlexNet（5 个卷积和 3 个全连接）为例，大致的训练过程可以理解为：

1. selective search 在一张图片中得到约 2k 个 object proposal(这里称为 RoI)
2. 缩放图片的 scale 得到图片金字塔，FP 得到 conv5 的特征金字塔。

3. 对于每个 scale 的每个 ROI，求取映射关系，在 conv5 中 crop 出对应的 patch。并用一个单层的 SPP layer（这里称为 RoI pooling layer）来统一到一样的尺度（对于 AlexNet 是 6x6）。
4. 继续经过两个全连接得到特征，这特征有分别 share 到两个新的全连接，连接上两个优化目标。第一个优化目标是分类，使用 softmax，第二个优化目标是 bbox regression，使用了一个 smooth 的 L1-loss.

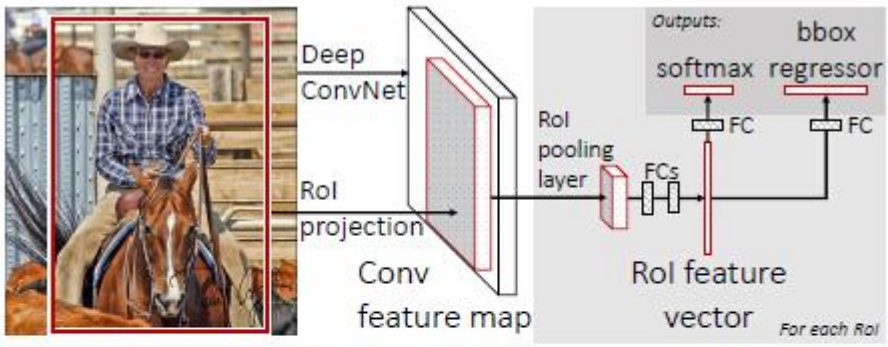


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully-convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully-connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

整体框架大致如上述所示了，对比回来 SPP-Net，可以看出 **FRCN 大致就是一个 joint training 版本的 SPP-Net**，改进如下：

1. SPP-Net 在实现上无法同时 tuning 在 SPP layer 两边的卷积层和全连接层。
2. SPP-Net 后面的需要将第二层 FC 的特征放到硬盘上训练 SVM，之后再额外训练 bbox regressor。

## RoI pooling layer

RoI pooling layer 的作用主要有两个，一个是将 image 中的 roi 定位到 feature map 中对应 patch ,另一个是用一个单层的 SPP layer 将这个 feature map patch 下采样为大小固定的 feature 再传入全连接层。

## Multi-task loss

FRCN 有两个 loss，以下分别介绍。

对于分类 loss，是一个 N+1 路的 softmax 输出，其中的 N 是类别个数，1 是背景。为何不用 SVM 做分类器了？在 5.4 作者讨论了 softmax 效 果比 SVM 好。

对于回归 loss，是一个 4xN 路输出的 regressor，也就是说对于每个类别都会训练一个单独的 regressor 的意思，一个平滑的 L1，形式如下：

$$L_{\text{loc}}(t, t^*) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i, t_i^*),$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

最后在 5.1 的讨论中，作者说明了 Multitask loss 是有助于网络的 performance 的。