Fast RCNN

思想

基础：RCNN

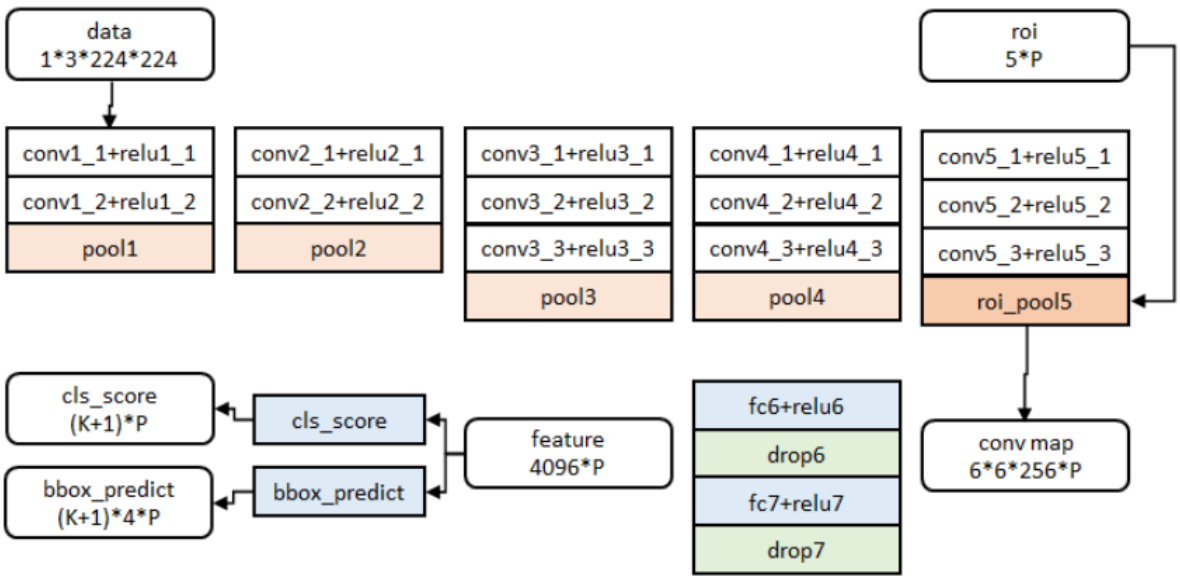
改进Fast RCNN

- Fast RCNN方法解决了RCNN方法三个问题：
  - 问题一：测试时速度慢
    - RCNN一张图片内候选框之间大量重叠，提取特征操作冗余。
    - 解决方法：将整张图像归一化后直接送入深度网络。在邻接时，才加入候选框信息，在末尾的少数几层处理每个候选框。
  - 问题二：**训练时速度慢**
    - 原因同上
    - 解决方法：将一张图像送入网络，紧接着送入从这幅图像上提取出的候选区域。
  - 问题三：**训练所需空间大**
    - RCNN中独立的分类器和回归器需要大量特征作为训练样本。
    - 把类别判断和位置精调统一用深度网络实现，不再需要额外存储。

特征提取网络

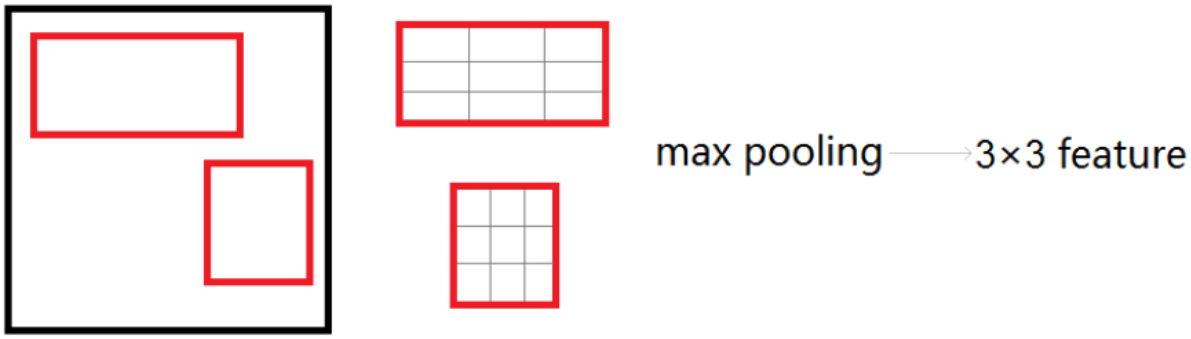
基本结构

- 图像归一化为227x227直接送入网络。
- 前五阶段是基础的`conv+relu+pooling`形式，在第五阶段结尾，输入P个候选区域。



ROI_pool层的测试（forward）

- roi_pool层将每个候选区域均匀分成MxN块，对每块进行max pooling。将特征图上大小不一的候选区域转变为大小统一的数据，送入下一层。



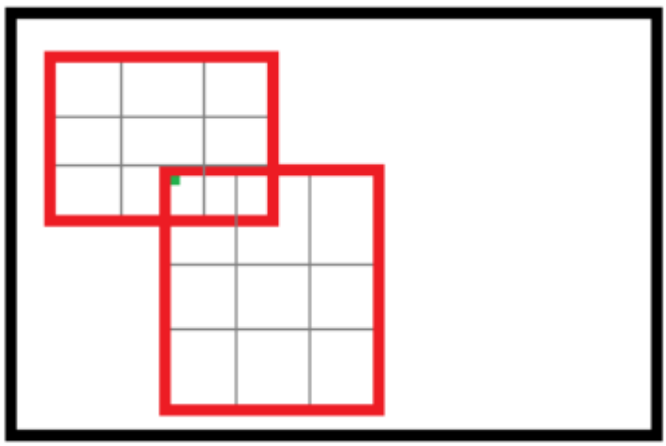
ROI_pool层的训练（backward）

- 首先考虑普通max pooling层，设`xi`为输入层的节点，`yi`为输出层的节点。

$$\frac{\partial L}{\partial x_i} = \begin{cases} 0 & \delta(i,j) = false \\ \frac{\partial L}{\partial y_j} & \delta(i,j) = true \end{cases}$$

  其中判决函数`δ(i,j)`表示`i`节点是否被`j`节点选为最大值输出。不被选中有两种可能：`xi`不在`yj`范围内，或者`xi`不是最大值。

- 对于roi max pooling，一个输入节点可能和多个输出节点相连。设`xi`为输入层的节点，`yrj`为第`r`个候选区域的第`j`个输出节点。



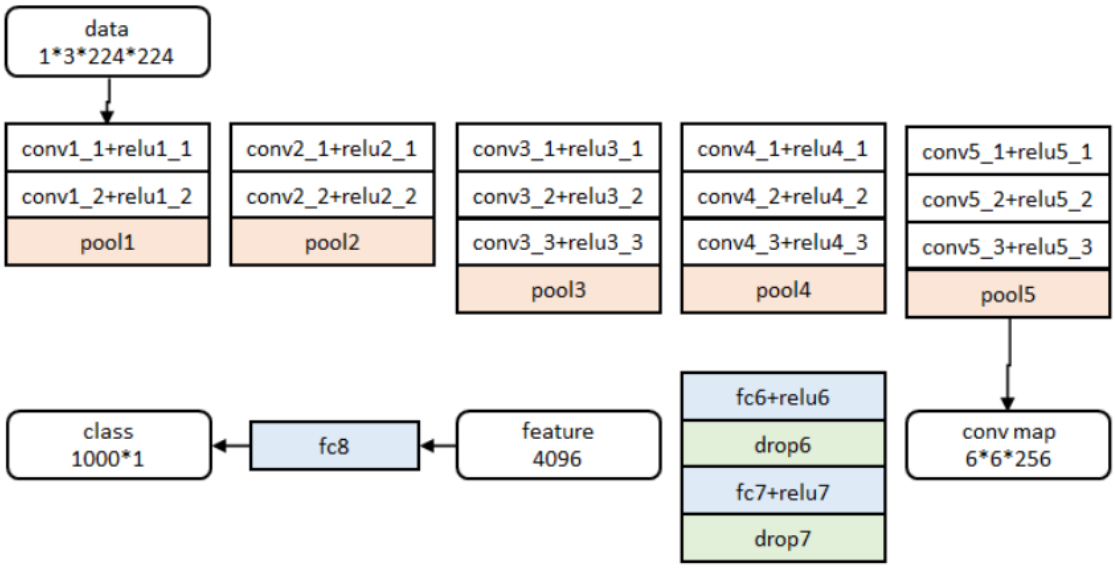
$$\frac{\partial L}{\partial x_i} = \Sigma_{r,j}\delta(i,r,j)\frac{\partial L}{\partial y_{rj}}$$

判决函数`δ(i,r,j)`表示`i`节点是否被候选区域`r`的第`j`个节点选为最大值输出。代价对于`xi`的梯度等于所有相关的后一层梯度之和。

网络参数训练

参数初始化

- 网络除去末尾部分如下图，在ImageNet上训练1000类分类器。结果参数作为相应层的初始化参数。



其余参数随机初始化。

分层数据

- 在调优训练时，每一个mini-batch中首先加入N张完整图片，而后加入从N张图片中选取的R个候选框。这R个候选框可以复用N张图片前5个阶段的网络特征。实际选择N=2，R=128。

训练数据构成

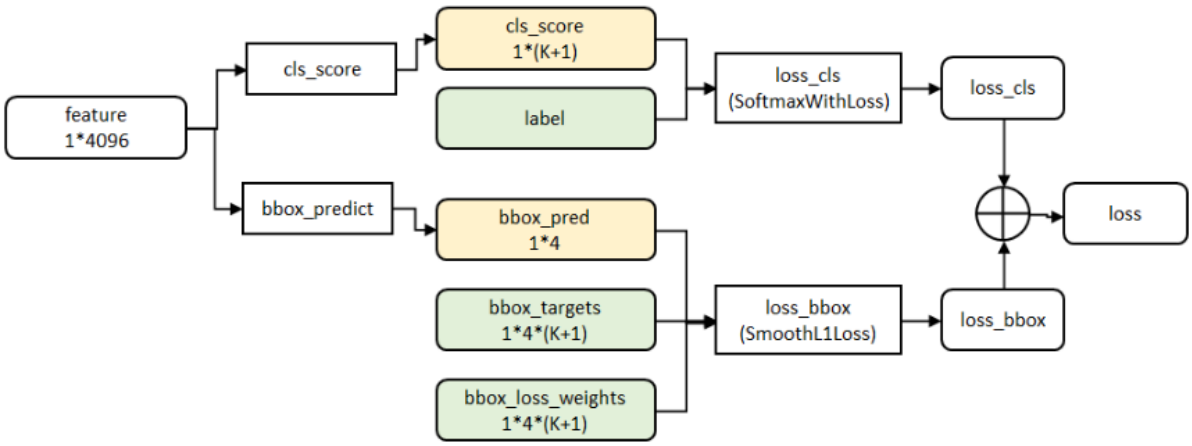
- N张完整图片以50%概率水平翻转。
- R个候选框的构成方式如下：

类别	比例	方式
前景	25%	与某个真值重叠在[0.5,1]的候选框
背景	75%	与真值重叠在[0.1,0.5）的候选框

分类与位置调整

数据结构

- 第五阶段的特征输入到两个并行的全连接层中（称为multi-task）。



- **cls_score层**用于分类，输出K+1维数组P，表示属于K类和背景的概率。
- **bbox_predict层**用于调整候选区域的位置，输出4*K维数组t，表示分别属于K类时，应该平移缩放的参数。

代价函数

- **loss_cls层**评估分类代价。由真实分类u对应的概率决定：

$$L_{cls} = -\log p_u$$

- **loss_bbox**评估检测框定位代价。比较真实分类对应的预测参数`tu`和真实平移缩放参数为`v`的差别：

$$L_{loc} = \Sigma_{i=1}^{4} g(t_i^u - v_i)$$

- g为Smooth L1误差，对outlier不敏感：

$$g(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

- 总代价为两者加权和，如果分类为背景则不考虑定位代价：

$$L = \begin{cases} L_{cls} + \lambda L_{loc} & u为前景 \\ L_{cls} & u为背景 \end{cases}$$

全连接层提取

- 分类和位置调整都是通过全连接层实现的，设前一级数据为x，后一级为y，全连接层参数为W，尺寸`u * v`。一次前向传播为：

$$y = Wx$$

计算复杂度为`u x v`。

- 将W进行SVD分解，兵营前t个特征值近似：

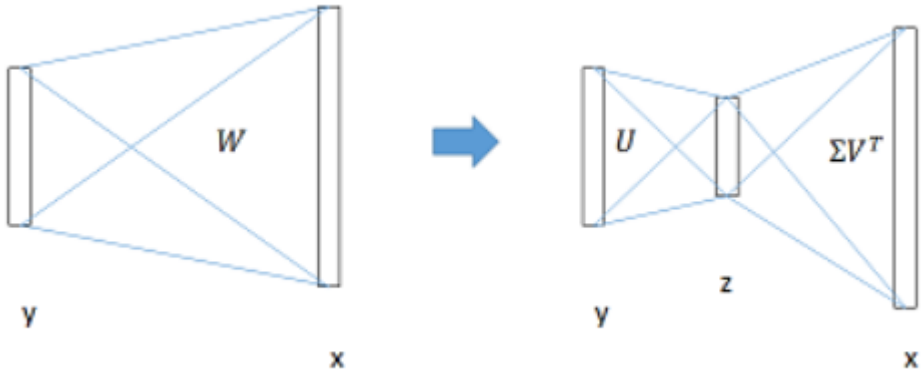
$$W = U\Sigma V^T \approx U(:, 1:t) \cdot \Sigma(1:t, 1:t) \cdot V(:, 1:t)^T$$

原来的前向传播分解成两步：

$$y = Wx = U \cdot (\Sigma \cdot V^T) \cdot x = U \cdot z$$

计算复杂度变为`u x t + v x t`。

在实现时，相当于把一个全连接层拆分成两个，中间以一个低维数据相连。



实验与结论

- 网络末端**同步训练***的分类和位置调整，提升准确度。
- 使用**多尺度**的图像金字塔，性能几乎没有提高。
- **倍增训练**数据，能够有2%~3%的准确度提升。
- 网络直接输出各类概率（**softmax**），比SVM分类器性能略好。
- **更多候选窗**不能提升性能。