Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Abstract

提出了一种用于目标检测的快速区域卷积网络方法(Fast R-CNN)。*Fast R-CNN*建立在之前使用深度卷积网络对候选目标进行有效分类的工作之上。与之前的工作相比，*Fast R-CNN*采用了一些创新来提高训练和测试速度，同时也提高了检测精度。*Fast R-CNN*训练非常深的*VGG16*网络*9* ×比*R-CNN*快，在测试时快*213* ×，并在*PASCAL VOC 2012*上实现了更高的*mAP*。与*SPPnet*相比，*Fast R-CNN*的训练速度更快*VGG16 3* ×，测试速度更快*10* ×，而且更准确。*Fast R-CNN*是用*Python*和*c++*实现的*(*使用*Caffe)*，可以在*MIT*开源许可证下获得，地址是*https://github.com/rbgirshick/fast-rcnn*。

1. 简介

最近，深度卷积网络[14, 16]显著提高了图像分类[14]和目标检测[9, 19]的精度。与图像分类相比，目标检测是一项更具挑战性的任务，需要更复杂的方法来解决。由于这种复杂性，目前的方法(*e.g.*, [9, 11, 19, 25])在多级管道中训练模型是缓慢和不优雅的。

检测需要精确定位物体，这带来了两个主要挑战。首先，必须处理大量候选对象位置(通常称为“建议”)。其次，这些候选对象只提供了粗略的定位，必须进行细化才能实现精确定位。这些问题的解决方案通常会牺牲速度、准确性或简单性。

本文简化了最先进的基于卷积网络的目标检测器的训练过程[9, 11]。本文提出一种单阶段训练算法，联合学习对候选目标进行分类并细化其空间位置。

由此产生的方法可以训练一个非常深的检测网络(VGG16 [20]) 9 ×比R-CNN [9]快，3 ×比SPPnet [11]快。在运行时，检测网络在0.3秒内处理图像(不包括目标建议时间)，同时在PASCAL VOC 2012 [7]上实现最高精度，mAP为66%(R-CNN为62%)。[1]

1.1. R-CNN和SPPnet

基于区域的卷积网络方法(R-CNN) [9]通过使用深度卷积网络对建议目标进行分类，实现了出色的目标检测精度。然而，R-CNN有明显的缺点:

1. 培训是一个多阶段的管道。R-CNN首先使用日志损失对目标建议进行微调。然后，将svm与卷积网络特征拟合。这些svm充当目标检测器，取代通过微调学习到的softmax分类器。在第三个训练阶段，学习边界框回归器。
2. 训练在空间和时间上都很昂贵。对于SVM和边界框回归器训练，从每幅图像中的每个候选对象中提取特征并写入磁盘。使用非常深的网络，例如VGG16，对于VOC07训练集的5k图像，这个过程需要2.5 gpu天。这些功能需要数百gb的存储空间。
3. 目标检测很慢。在测试时，从每个测试图像的每个候选目标中提取特征。使用VGG16进行检测需要47秒/张图像(在GPU上)。

R-CNN很慢，因为它为每个目标建议执行卷积网络前向传递，而没有共享计算。提出空间金字塔池化网络(SPPnets) [11]通过共享计算来加速R-CNN。SPPnet方法为整个输入图像计算卷积特征映射，然后使用从共享特征映射中提取的特征向量对每个目标建议进行分类。通过将建议框内的特征映射部分最大池化到固定大小的输出(*e.g.*, 6×6)，来提取建议框的特征。多个输出大小被池化，然后连接起来，就像空间金字塔池化[15]。SPPnet在测试时将R-CNN加速10到100 ×。由于提取候选特征的速度更快，训练时间也减少了3 ×。

SPPnet也有明显的缺点。与R-CNN一样，训练是一个多阶段的管道，包括提取特征、使用对数损失对网络进行微调、训练svm，最后拟合边界框回归器。特性也被写入磁盘。但与R-CNN不同的是，[11]中提出的微调算法不能更新空间金字塔池化之前的卷积层。不出所料，这种限制(固定的卷积层)限制了非常深的网络的准确性。

1.2. 贡献

本文提出一种新的训练算法，修复了R-CNN和SPPnet的缺点，同时提高了它们的速度和精度。我们称这种方法为*Fast R-CNN*，因为它的训练和测试速度相对较快。Fast R-CNN方法有以下几个优点:

[1] 所有计时使用一个Nvidia K40 GPU超频到875 MHz。

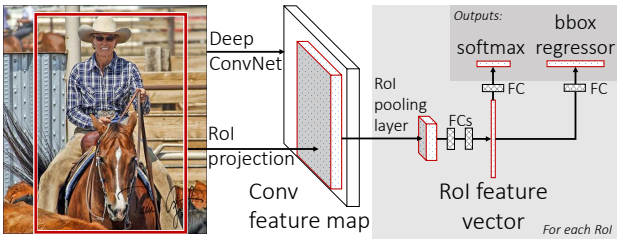

Figure 1. 快速R-CNN架构。将输入图像和多个感兴趣区域(RoI s)输入到全卷积网络中。每个RoI被汇集到一个固定大小的特征图中，然后通过全连接层(FCs)映射到一个特征向量。每个RoI网络有两个输出向量:softmax概率和每个类的边界框回归偏移量。该架构是端到端的训练，具有多任务损失。

1. 检测质量(mAP)高于R-CNN、SPPnet
2. 训练是单阶段的，使用了多任务损失
3. 训练可以更新所有网络层
4. 特性缓存不需要磁盘存储

Fast R-CNN是用Python和c++编写的(Caffe [13])，可以在MIT开源许可证下获得，地址是https://github.com/rbgirshick/fast-rcnn。

2. 快速R-CNN架构和训练

Fig. 1 说明了Fast R-CNN架构。Fast R-CNN网络将整个图像和一组目标建议作为输入。该网络首先用几个卷积*(conv)*和最大池化层处理整个图像，以产生conv特征图。然后，对于每个目标建议，感兴趣区域*(RoI)*池化层从特征图中提取固定长度的特征向量。每个特征向量被馈送到一个全连接*(fc)*层序列中，最终分支成两个兄弟输出层:一个在K对象类上产生softmax概率估计加上一个全能的"背景"类，另一个层为每个K对象类输出四个实数。每组4值对一个K类的精确边界框位置进行编码。

2.1. RoI池化层

RoI池化层使用最大池化将任何有效感兴趣区域内的特征转换为具有$H \times W$ (*e.g.*, 7×7)固定空间范围的小特征图，其中H和W是独立于任何特定RoI的层超参数。在本文中，RoI是一个矩形窗口转换成一个特征图。每个RoI由一个四元组(r, c, h, w)定义，它指定其左上角(r, c)以及其高度和宽度(h, w)。

RoI最大池化工作通过将$h \times w$ RoI窗口划分为近似大小$h/H \times w/W$的子窗口$H \times W$网格，然后将每个子窗口中的值最大池化到相应的输出网格单元。池化独立应用于每个特征映射通道，如标准的最大池化。RoI层只是SPPnets [11]中使用的空间金字塔池化层的特殊情况，其中只有一个金字塔层。我们使用[11]中给出的池化子窗口计算。

2.2. 从预训练网络初始化

我们用三个预训练的ImageNet [4]网络进行实验，每个网络都有5个最大池化层和5到13个转换层(有关网络细节，请参阅Section 4.1)。当预训练网络初始化Fast R-CNN网络时，它将经历三个转换。

首先，最后一个最大池化层被RoI池化层取代，该层通过设置H和W与网络的第一个全连接层(*e.g.*, $H = W = 7$为VGG16)兼容来配置。

其次，网络的最后一个全连接层和softmax(为1000路ImageNet分类进行训练)被替换为前面描述的两个兄弟层($K + 1$类别和特定类别边界框回归器上的全连接层和softmax)。

第三，修改网络以接受两个数据输入:图像列表和这些图像中的RoI s列表。

2.3. 检测微调

通过反向传播训练所有网络权值是Fast R-CNN的一个重要能力。首先，让我们阐明为什么SPPnet无法更新空间金字塔池化层以下的权重。

根本原因是当每个训练样本(*i.e.* RoI)来自不同的图像时，通过SPP层的反向传播非常低效，这正是R-CNN和SPPnet网络的训练方式。效率低下的原因是每个RoI可能具有非常大的感受野，通常跨越整个输入图像。由于前向传递必须处理整个感受野，因此训练输入很大(通常是整个图像)。

本文提出一种更有效的训练方法，在训练过程中利用特征共享。在Fast R-CNN训练中，对随机梯度下降(SGD)小批次进行分层采样，首先通过采样N图像，然后从每个图像中采样R/NRoI s。关键是，RoI s从同一图像共享计算和内存中的前向和后向传递。使N变小可以减少小批量计算。例如，当使用$N = 2$和$R = 128$时，所提出的训练方案比从128不同的图像中采样一个RoI大约快64 $\times$ (*i.e.*, R-CNN和SPPnet策略)。

关于这种策略的一个问题是，它可能会导致训练收敛缓慢，因为来自同一图像的RoI s是相关的。这个问题似乎不是一个实际问题，我们使用比R-CNN更少的SGD迭代次数在$N = 2$和$R = 128$上取得了良好的结果。

除了分层采样之外，Fast R-CNN使用一个精简的训练过程，其中一个微调阶段联合优化softmax分类器和边界框回归器，而不是在三个单独的阶段训练softmax分类器、svm和回归器[9, 11]。这个过程的组成部分(损失、小批量采样策略、通过RoI池化层的反向传播和SGD超参数)如下所述。

多任务丢失。 Fast R-CNN网络有两个兄弟输出层。第一个输出$K + 1$类别上的离散概率分布(每个RoI) $p = (p_0, \ldots, p_K)$。照例，p是由softmax在全连接层的$K + 1$输出上计算出来的。第二个兄弟层输出每个K对象类的边界框回归偏移量$t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$，由$k$索引。我们使用[9]中给出的$t^k$的参数化，其中$t^k$指定相对于

目标建议的缩放不变的平移和对数空间的高度/宽度偏移。

每个训练RoI被标记为基础真实类u和基础真实边界框回归目标v。我们在每个标签RoI上使用多任务损失L来联合训练分类和边界框回归:

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

其中$L_{\text{cls}}(p, u) = -\log p_u$是真实类$u$的日志丢失。

第二个任务损失L_{loc}是在类u、$v = (v_{\text{x}}, v_{\text{y}}, v_{\text{w}}, v_{\text{h}})$的真正边界框回归目标的元组上定义的，以及类$u$的预测元组$t^u = (t^u_{\text{x}}, t^u_{\text{y}}, t^u_{\text{w}}, t^u_{\text{h}})$上定义的。艾弗森括号指示函数$[u \geq 1]$在$u \geq 1$时求值为1，否则求值为0。按照惯例，笼统的背景类标记为$u = 0$。对于背景RoI s没有ground-truth边界框的概念，因此L_{loc}被忽略。对于边界框回归，我们使用损失函数

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t^u_i - v_i), \quad (2)$$

在...

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \quad (3)$$

是一个鲁棒的L_1损失，对异常值不像R-CNN和SPPnet中使用的L_2损失那么敏感。当回归目标无界时，使用L_2损失进行训练可能需要仔细调整学习率，以防止梯度爆炸。Eq. 3消除了这种敏感性。

Eq. 1中的超参数λ控制两个任务损失之间的平衡。我们对地面真实值回归目标v_i进行归一化，使其具有零均值和单位方差。所有实验都使用$\lambda = 1$。

我们注意到[6]使用相关损失来训练一个与类别无关的对象候选网络。与我们的方法不同，[6]提倡将本地化和分类分开的双网络系统。OverFeat [19], R-CNN [9]和SPPnet [11]也训练分类器和边界框定位器，然而这些方法使用分阶段训练，本文表明这对于Fast R-CNN (Section 5.1)是次优的。

小批量抽样。 在微调过程中，每个SGD小批量都是从$N = 2$图像中构建的，这些图像均匀随机地选择(通常的做法是，我们实际上迭代数据集的排列)。我们使用小批量$R = 128$，从每个图像中采样64RoI s。与[9]一样，我们从具有交集/并集(IoU)的对象建议中提取25%的RoI s，与至少为0.5的真实边界框重叠。这些RoI s包含标记为前景对象类*i.e.* $u \geq 1$的示例。剩余的RoI s从对象建议中采样，这些建议在区间$[0.1, 0.5)$中具有最大的IoU，位于[11]之后。这些是背景示例，标签为$u = 0$。0.1的较低阈值似乎可以作为一种启发式方法，用于挖掘[8]的困难示例。在训练过程中，图像按概率0.5水平翻转。没有使用其他数据增强。

通过**RoI**池化层进行反向传播。 反向传播通过RoI池化层路由衍生品。为清晰起见，我们假设每个小批量($N = 1$)只有一个图像，尽管扩展到$N > 1$是直接的，因为向前传递独立处理所有图像。

让$x_i \in \mathbb{R}$是RoI池化层的i -th激活输入，让y_{rj}是该层的j -th输出r -th RoI。RoI池化层计算$y_{rj} = x_{i^*(r,j)}$，其中$i^*(r, j) = \text{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$。$\mathcal{R}(r, j)$是输出单元$y_{rj}$ Max池所在的子窗口中的输入索引集。单个x_i可以分配给几个不同的输出y_{rj}。

RoI池化层的反向函数通过遵循argmax开关来计算损失函数相对于每个输入变量x_i的偏导数:

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}}. \quad (4)$$

换句话说，对于每个小批量RoI r和每个池化输出单元y_{rj}，如果i是通过最大池化为y_{rj}选择的argmax，则偏导数$\partial L/\partial y_{rj}$将累加。在反向传播中，偏导数$\partial L/\partial y_{rj}$已经由RoI池化层顶部的层的反向函数计算出来了。

SGD超参数。 用于softmax分类和bounding-box回归的全连接层分别从标准差为0.01和0.001的零均值高斯分布中初始化。偏差初始化为0。所有层使用1作为权重，2作为偏差和0.001作为全局学习率。当在VOC07或VOC12 trainval上训练时，我们运行SGD进行30k小批量迭代，然后将学习率降低到0.0001并训练另一个10k迭代。当我们在更大的数据集上训练时，我们运行SGD进行更多的迭代，如下所述。动量0.9和参数衰减0.0005(关于权重和偏差)被使用。

2.4. 尺度不变性

本文探索了两种实现尺度不变目标检测的方法:(1)通过"蛮力"学习和(2)使用图像金字塔。这些策略遵循[11]中的两种方法。在暴力方法中，每个图像在训练和测试期间都以预定义的像素大小处理。网络必须直接从训练数据中学习尺度不变的目标检测。

相比之下，多尺度方法通过图像金字塔为网络提供近似的尺度不变性。在测试时，使用图像金字塔近似地对每个候选目标进行尺度归一化。在多尺度训练期间，我们每次采样图像时随机采样一个金字塔尺度，随后[11]，作为数据增强的一种形式。由于GPU内存的限制，我们仅对较小的网络进行多尺度训练。

3. 快速**R-CNN**检测

一旦对Fast R-CNN网络进行微调，检测等同于运行前向传递(假设目标建议是预先计算的)。该网络将图像(或图像金字塔，编码为图像列表)和R对象建议列表作为输入进行评分。在测试时，R通常在2000附近，尽管我们将考虑它更大的情况(≈45 k)。当使用图像金字塔时，每个RoI被分配到尺度，这样缩放后的RoI最接近区域[11]中的224^2像素。

对于每个测试RoI r，正向传递输出一个类后验概率分布p和一组相对于r的预测边界框偏移量(每个K类获得自己的精确边界框预测)。我们使用估计的概率Pr(class $= k \mid r$) $\triangleq p_k$为每个对象类k分配检测置信度r。然后，我们使用R-CNN [9]中的算法和设置独立地对每个类进行非极大值抑制。

3.1. 截断SVD，加快检测速度

对于全图像分类，计算全连接层的时间比计算转换层的时间少。相反，对于检测来说，要处理的RoI s的数量很大，几乎一半的前向传递时间花费在计算完全连接的层上(参见Fig. 2)。通过使用截断SVD压缩大型全连接层，可以很容易地加速[5, 23]。

在这种技术中，由参数化的层$u \times v$ 权重矩阵W 近似分解为

$$W \approx U\Sigma_t V^T \tag{5}$$

使用SVD。在这个因子分解中，U 是一个$u \times t$ 矩阵组成第一个t 的左奇异向量W，Σ_t 是一个$t \times t$ 包含顶部的对角矩阵t 的奇异值W和V 是$v \times t$ 矩阵组成第一个t的右奇异向量W。截断SVD减少了从的参数计数uv to $t(u+v)$，如果t 比$\min(u, v)$。要压缩一个网络，单一的全连接层对应W 被两个完全连接的层取代，它们之间没有非线性。第一层使用权重矩阵$\Sigma_t V^T$ (没有偏见)和第二次使用U (与原始偏差相关W）．这种简单的压缩方法可以很好地提高速度，当元素的数量为RoIS很大。

4. 主要结果

三个主要结果支持本文的贡献:

1. VOC07、2010和2012的最新地图
2. 与R-CNN、SPPnet相比的快速训练和测试
3. 在VGG16中对conv层进行微调可以改进mAP

4.1. 实验装置

实验使用了三个预先训练的ImageNet模型，这些模型可以在线获得。[2] 第一个是来自R-CNN [9]的CaffeNet(本质上是AlexNet [14])。我们也把这个CaffeNet称为**S**模型，意思是"小"。第二个网络是来自[3]的VGG_CNN_M_1024，与**S**具有相同的深度，但更宽。我们称这种网络模型为**M**，即"媒介"。最终的网络是来自[20]的非常深的VGG16模型。由于这个模型是最大的，我们称之为**L**模型。在本节中，所有实验都使用单尺度训练和测试($s = 600$;详情见Section 5.2)。

4.2. VOC 2010和2012的结果

在这些数据集上，我们将Fast R-CNN(简称*FRCN*)与来自公共排行榜(Table 2, Table 3)的comp4(外部数据)跟踪上的顶级方法进行比较。[3] 对于NUS_NIN_c2000和婴儿学习方法，目前没有相关的出版物，我们无法找到有关所使用的卷积网络架构的确切信息;它们是网络中网络设计的变种[17]。所有其他方法都从相同的预训练VGG16网络中初始化。

Fast R-CNN在VOC12上取得了最高的结果，mAP达到了65.7%(以及68.4%的额外数据)。它也比其他方法快两个数量级，这些方法都是基于'slow ' R-CNN管道。在VOC10上，SegDeepM [25]取得了比Fast R-CNN更高的mAP (67.2%vs. 66.1%)。SegDeepM在VOC12 trainval加上分割标注的基础上进行训练;它旨在通过使用马尔可夫随机场来推理O $_2$ P [1]语义分割方法中的R-CNN检测和分割，从而提高R-CNN的准确性。可以将Fast R-CNN替换为SegDeepM，从而获得更好的结果。当使用放大的07++12训练集(见Table 2标题)时，Fast R-CNN的mAP增加到68.8%，超过了SegDeepM。

4.3. 2007年度VOC测试结果

在VOC07上，我们将Fast R-CNN与R-CNN和SPPnet进行了比较。所有方法都从相同的预训练VGG16网络开始，并使用边界框回归。VGG16 SPPnet的结果由[11]的作者计算。SPPnet在训练和测试过程中使用了5个量表。Fast R-CNN在SPPnet上的改进表明，即使Fast R-CNN使用单尺度训练和测试，微调卷积层也可以大大提高mAP(从63.1%提高到66.9%)。R-CNN的mAP达到了66.0%。次要的一点是，SPPnet在训练时没有使用PASCAL标记为"困难"的示例。删除这些示例将Fast R-CNN mAP提高到68.1%。所有其他实验都使用"困难"的例子。

4.4. 训练和测试时间

快速的训练和测试时间是我们的第二个主要成果。Table 4比较了Fast R-CNN, R-CNN和SPPnet在VOC07上的训练时间(小时)，测试率(秒每图像)和mAP。对于VGG16, Fast R-CNN处理图像的速度比没有截断SVD的R-CNN快146 ×，比它快213 ×。培训时间减少9 ×，从84小时减少到9.5小时。与SPPnet相比，Fast R-CNN的训练速度提高了VGG16 2.7 × (在9.5小时vs. 25.5小时)，在没有截断SVD的情况下测试速度提高了7 ×，使用它测试速度提高了10 ×。Fast R-CNN还消除了数百gb的磁盘存储，因为它没有缓存功能。

截断SVD。 截断SVD可以将检测时间减少30%以上，而mAP只下降了很小(0.3个百分点)，并且在模型压缩后不需要进行额外的微调。Fig. 2说明了如何使用来自VGG16的fc6层的25088×4096矩阵的顶部1024奇异值和来自4096 × 4096 fc7层的顶部256奇异值在mAP中几乎没有损失的情况下减少运行时。如果在压缩后

[2] https://github.com/BVLC/caffe/wiki/Model-Zoo

[3] http://host.robots.ox.ac.uk:8080/leaderboard (2015年4月18日通过)

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007**测试检测平均精度(%)。所有方法都使用VGG16。训练集键:**07**:VOC07 trainval, **07\ diff**:**07**没有"困难"的例子，**07+12:07**和VOC12 trainval的结合。† SPPnet的结果由[11]的作者编写。

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010**测试检测平均精度(%)。BabyLearning使用了一个基于[17]的网络。所有其他方法使用VGG16。训练集键:**12**:VOC12训练，道具。:专有数据集，**12+seg**: **12**带分割注释，**07++12**:VOC07 trainval, VOC07 test和VOC12 trainval的联合。

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012**测试检测平均精度(%)。BabyLearning和NUS__NIN__c2000使用基于[17]的网络。所有其他方法使用VGG16。训练集键:见Table 2, 。:未知。

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. 相同模型在Fast R-CNN、R-CNN和SPPnet中的运行时比较。Fast R-CNN使用单尺度模式。SPPnet使用[11]中指定的五个刻度。†时间由[11]作者提供。时间是在Nvidia K40 GPU上测量的。

再次微调，则可以在mAP下降较小的情况下进一步加速。

4.5. 哪些层需要微调?

对于SPPnet论文[11]中考虑的较少深度的网络，只微调全连接层似乎就足以获得良好的精度。我们假设这个结果不适用于非常深的网络。为了验证微调转换层对VGG16的重要性，我们使用Fast R-CNN进行微调，但冻结13个转换层，以便只有全连接层进行学习。这种消融模拟了单尺度SPPnet训练，并

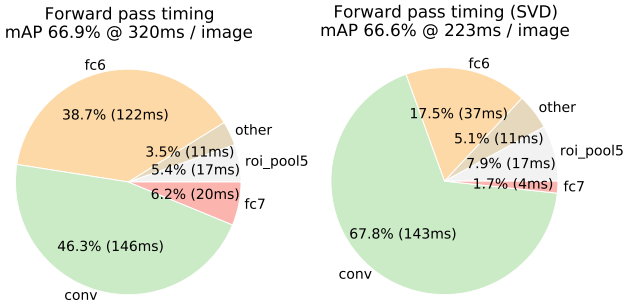

Figure 2. VGG16在截断SVD前后的时间。在SVD之前，全连接层fc6和fc7占用了45%的时间。

将*mAP*从*66.9*%降低到*61.4*% (Table 5)。这个实验验证了我们的假设:通过RoI池化层进行训练对于非常深的网络很重要。

	layers that are fine-tuned in model **L**			SPPnet **L**
	≥ fc6	≥ conv3_1	≥ conv2_1	≥ fc6
VOC07 mAP	61.4	66.9	**67.2**	63.1
test rate (s/im)	**0.32**	**0.32**	**0.32**	2.3

Table 5. 对VGG16进行微调的限制层的效果。微调≥ fc6模拟SPPnet训练算法[11]，但使用单一尺度。SPPnet **L**结果使用5个尺度，以显著的速度成本(7 ×)获得。

这是否意味着所有的conv层都应该进行微调?简

而言之，没有。在较小的网络(**S**和**M**)中，我们发现conv1是通用的和任务独立的(众所周知的事实[14])。是否允许conv1学习对mAP没有任何意义。对于VGG16，我们发现只需要更新从conv3_1及以上的层(13个conv层中的9个)。这一观察是务实的:(1)与从conv3_1学习相比，从conv2_1更新将训练速度降低1.3 × (12.5小时vs. 9.5小时); (2)从conv1_1更新会占用GPU内存。从conv2_1 up学习时，mAP的差异只有+0.3点(Table 5，最后一列)。本文中所有Fast R-CNN的结果都使用VGG16微调层conv3_1和up;所有使用模型**S**和**M**的实验都对conv2及以上层进行微调。

5. 设计评估

我们进行了实验，以了解R-CNN与R-CNN和SPPnet相比有多快，以及评估设计决策。遵循最佳实践，我们在PASCAL VOC07数据集上进行了这些实验。

5.1. 多任务训练有帮助吗?

多任务训练很方便，因为它避免了管理顺序训练任务的管道。但它也有可能改善结果，因为任务通过共享表示(ConvNet)相互影响[2]。多任务训练是否提高了Fast R-CNN的目标检测精度?

为了测试这个问题，我们训练了只使用分类损失L_{cls}的基线网络Eq. 1 (*i.e.*，设置$\lambda = 0$)。这些基线打印在Table 6中每组的第一列中的模型**S**、**M**和**L**。请注意，这些模型没有边界框回归量。接下来(每组的第二列)，我们采用用多任务损失(Eq. 1, $\lambda = 1$)训练的网络，但我们在测试时禁用边界框回归。这隔离了网络的分类精度，并允许与基线网络进行完全相同的比较。

在所有三个网络中，多任务训练相对于单独进行分类训练提高了纯分类精度。改进范围从+0.8到+1.1地图点，显示了多任务学习的一致积极影响。

最后，我们采用基线模型(仅使用分类损失进行训练)，添加边界框回归层，并使用L_{loc}训练它们，同时保持所有其他网络参数冻结。每组的第三列显示了这种分阶段训练方案的结果:mAP比第一列有所提高，但分阶段训练的表现不如多任务训练(每组第四列)。

5.2. 尺度不变性:使用蛮力还是技巧?

本文比较了实现尺度不变目标检测的两种策略:暴力学习(单尺度)和图像金字塔(多尺度)。在这两种情况下，我们将图像的比例s定义为其最短边的长度。

所有单尺度实验使用$s = 600$像素; s可能小于600的一些图像，因为我们cap最长的图像边在1000像素，并保持图像的纵横比。选择这些值是为了在微调期间VGG16适合GPU内存。较小的模型不受内存限制，可以从较大的s值中受益;然而，为每个模型优化s并不是我们主要关注的问题。我们注意到PASCAL图像平均为384 × 473像素，因此单尺度设置通常将图像的样本增加1.6倍。因此，RoI池化层的平均有效步幅为≈ 10像素。

在多尺度设置中，我们使用[11] ($s \in \{480, 576, 688, 864, 1200\}$)中指定的相同的5个尺度，以方便与SPPnet进行比较。然而，我们将最长的边限制在2000像素，以避免超过GPU内存。

Table 7 在用一个或五个尺度进行训练和测试时显示模型**S**和**M**。也许[11]中最令人惊讶的结果是单尺度检测的性能几乎与多尺度检测一样好。我们的发现证实了他们的结果:深度卷积网络擅长直接学习尺度不变性。多尺度方法仅以较大的计算时间代价增加了mAP (Table 7)。在VGG16 (模型**L**)的情况下，我们受限于根据实现细节使用单一的尺度。然而，它达到了66.9%的mAP，略高于R-CNN [10]的66.0%，即使R-CNN使用"无限"尺度，因为每个建议都被扭曲为规范的大小。

由于单尺度处理提供了速度和精度之间的最佳权衡，特别是对于非常深的模型，因此本小节之外的所有实验都使用$s = 600$像素的单尺度训练和测试。

5.3. 我们需要更多的训练数据吗?

当提供更多的训练数据时，一个好的目标检测器应该得到改进。Zhu *et al*. [24]发现DPM [8]地图仅在几百到几千个训练示例后就饱和了。为了评估Fast R-CNN，我们将VOC07训练集扩充为VOC12训练集，大致上将图像数量增加了两倍，达到16.5k。在VOC07测试集上，mAP从66.9%提高到70.0%(Table 1)。在此数据集上训练时，我们使用60k小批量迭代而不是40k。

对VOC10和2012进行了类似的实验，构建了一个由VOC07 trainval、test和VOC12 trainval的并集21.5k图像组成的数据集。当在这个数据集上训练时，我们使用100k次SGD迭代，并将学习率降低0.1×每40k次迭代(而不是每30k次)。在VOC10和2012年，mAP分别从66.1%提高到68.8%和65.7%提高到68.4%。

5.4. svm性能优于softmax吗?

Fast R-CNN使用微调期间学习的softmax分类器，而不是像R-CNN和SPPnet那样在事后训练一对其余的线性svm。为了理解这一选择的影响，我们在Fast R-CNN中使用困难的负向挖掘实现了事后SVM训练。我们使用与R-CNN相同的训练算法和超参数。

Table 8 显示softmax在所有三个网络上的表现略优于SVM，从+0.1到+0.8映射点。这种效果很小，但它表明，与之前的多阶段训练方法相比，"一次"微调就足够了。我们注意到，softmax与一对其余svm不同，在得分RoI时引入了类之间的竞争。

5.5. 提案越多就越好吗?

(广义上)有两种类型的目标检测器:使用稀疏目标建议集的检测器(*e.g.*，选择性搜索[21])和使用密集集的检测器(*e.g.*, DPM [8])。对稀疏建议集进行分类是一种级联[22]，其中建议机制首先拒绝大量候选集，留下一个小集合来评估分类器。当应用于DPM检测时，这种级联提高了检测精度[21]。有证据表明，建议分类器级联也提高了Fast R-CNN的精度。

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓				✓				✓	
test-time bbox reg?			✓	✓			✓	✓			✓	✓
VOC07 mAP	52.2	53.3	54.6	**57.1**	54.7	55.5	56.6	**59.2**	62.6	63.4	64.0	**66.9**

Table 6. 多任务训练(每组第四列)比分段训练(每组第三列)改进了mAP。

	SPPnet **ZF**		**S**		**M**		**L**
scales	1	5	1	5	1	5	1
test rate (s/im)	0.14	0.38	**0.10**	0.39	0.15	0.64	0.32
VOC07 mAP	58.0	59.2	57.1	58.4	59.2	60.7	**66.9**

Table 7. 多尺度vs.单尺度。SPPnet **ZF** (类似于模型**S**)结果来自[11]。具有单一尺度的大型网络提供了最佳的速度/精度权衡。(**L**由于GPU内存限制，无法在我们的实现中使用多尺度。)

method	classifier	**S**	**M**	**L**
R-CNN [9, 10]	SVM	**58.5**	**60.2**	66.0
FRCN [ours]	SVM	56.3	58.7	66.8
FRCN [ours]	softmax	57.1	59.2	**66.9**

Table 8. 基于softmax和SVM的快速R-CNN (VOC07 mAP)。

使用选择性搜索的质量模式，我们从每张图像的1k到10k个建议框，每次重新训练和重新测试模型**M**。如果建议纯粹用于计算，那么增加每个图像的建议数量应该不会损害mAP。

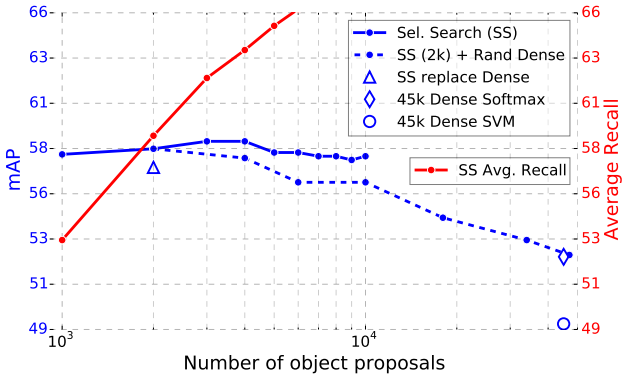

Figure 3. VOC07测试地图和AR为各种提案方案。

我们发现，随着建议数量的增加，mAP先上升后下降(Fig. 3，蓝色实线)。这个实验表明，用更多的建议框淹没深度分类器对准确率没有帮助，甚至略有伤害。

如果不实际运行实验，这个结果很难预测。测量对象建议框质量的最先进方法是平均召回率(AR)[12]。当每张图像使用固定数量的建议框时，对于使用*R-CNN*的几种建议框方法，AR与mAP的相关性很好。Fig. 3显示AR(实线红线)与mAP的相关性并不好，因为每个图像的建议数量是不同的。AR必须小心使用;提案越多AR越高并不意味着mAP会增加。幸运的是，使用**M**模型进行训练和测试只需要不到2.5小时。因此，Fast R-CNN能够高效、直接地评估目标建议图，这比代理度量更可取。

本文还研究了使用密集生成的框(超尺度、位置和宽高比)时的Fast R-CNN，速率约为45k框/图像。这个密集的集合是足够丰富的，当每个选择搜索框被替换为它最近的(IoU)密集框，mAP只下降1点(到57.7%，Fig. 3，蓝色三角形)。

密集搜索框的统计信息与选择性搜索框的统计信息不同。从2k个选择性搜索框开始，我们在添加1000 × $\{2, 4, 6, 8, 10, 32, 45\}$密集框的随机样本时测试地图。对于每个实验，我们重新训练和重新测试模型**M**。当添加这些密集的搜索框时，mAP的下降比添加更多选择性搜索框时更强烈，最终达到53.0%。

我们还仅使用密集框(45k /图像)训练和测试Fast R-CNN。这个设置产生52.9%的地图(蓝钻)。最后，我们检查是否需要具有硬负挖掘的svm来处理密集的盒分布。svm的表现更差:49.3%(蓝色圆圈)。

5.6. MS COCO初步结果

我们将Fast R-CNN(连同VGG16)应用于MS COCO数据集[18]以建立初步基线。我们在80k的图像训练集上进行了240k次迭代，并使用评估服务器在'test-dev '集上进行评估。pascal风格的mAP为35.9%;新的COCO-style AP平均也超过IoU阈值，为19.7%。

6. 结论

本文提出Fast R-CNN，对R-CNN和SPPnet进行干净快速的更新。除了报告最先进的检测结果外，本文提出了详细的实验，希望提供新的见解。特别值得注意的是，稀疏目标建议似乎可以提高检测器的质量。在过去，这个问题的成本(时间)太高，无法探测，但通过Fast R-CNN变得实用。当然，可能存在尚未发现的技术，可以使密集框的性能与稀疏建议一样好。这样的方法，如果开发出来，可能有助于进一步加速目标检测。

致谢　感谢He Kaiming、Larry Zitnick和Piotr Dollár的讨论和鼓励。

References

[1] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 4

[2] R. Caruana. Multitask learning. *Machine learning*, 28(1), 1997. 6

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 4

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[5] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 4

[6] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014. 3

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 1

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010. 3, 6

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 3, 4, 7

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *TPAMI*, 2015. 5, 6, 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2, 3, 4, 5, 6, 7

[12] J. H. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *arXiv preprint arXiv:1502.05082*, 2015. 7

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. of the ACM International Conf. on Multimedia*, 2014. 2

[14] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 4, 5

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1

[16] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comp.*, 1989. 1

[17] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 4, 5

[18] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *arXiv e-prints*, arXiv:1405.0312 [cs.CV], 2014. 7

[19] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *ICLR*, 2014. 1, 3

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 4

[21] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 6

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 6

[23] J. Xue, J. Li, and Y. Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, 2013. 4

[24] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In *BMVC*, 2012. 6

[25] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. In *CVPR*, 2015. 1, 4