快速R-CNN

Ross Girshick
微软研究院rbg@mic
rosoft.com

摘要

本文提出了一种快速基于区域的卷积网络(Fast R-CNN)*的目标检测方法。快速R-CNN建立在以前的工作基础上,使用深度卷积网络有效地分类对象建议。与以前的工作相比,Fast R-CNN采用了几项创新来提高训练和测试速度,同时也提高了检测精度。快速R-CNN训练极深VGG16网络的速度比R-CNN快9倍,在测试时快213倍,并且在PASCAL VOC 2012上实现了更高的mAP。与SPPnet相比,Fast R-CNN训练VGG16速度快3倍,测试速度快10倍,并且更准确。快速R-CNN是用Python和c++实现的(使用Caffe),在开源的MIT许可证下可以在https: //github.com/rbgirshick/fast-rcnn上获得。*

1. 简介

最近，深度ConvNets[14,16]显著提高了图像分类[14]和目标检测[9,19]的准确率。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(例如[9,11,19,25])在多阶段管道中训练模型，速度缓慢且不美观。

复杂性的产生是因为检测需要对物体进行精确定位，这就产生了两个主要挑战。首先，必须处理大量候选物体位置(通常称为“提案”)。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往往会牺牲速度、准确性或简单性。

在本文中，我们简化了最先进的基于convnet的目标检测器的训练过程[9,11]。我们提出了一种单阶段训练算法，该算法联合学习对目标提案进行分类并细化其空间位置。

所得到的方法可以训练非常深的检测网络(VGG16[20])，比R-CNN[9]快9倍，比SPPnet[11]快3倍。在运行时，检测网络在0.3s内处理图像(不包括对象建议时间)，同时在PASCAL VOC 2012[7]上达到最高精度，mAP为66% (R-CNN为62%)

1.1. R-CNN和SPPnet

基于区域的卷积网络方法(R-CNN)[9]通过使用深度卷积神经网络对目标提案进行分类，实现了优异的目标检测精度。然而，R-CNN也有明显的缺点:

1. 培训是一个多阶段的流水线。R-CNN首先使用对数损失对目标提议的卷积神经网络进行微调。然后，将支持向量机拟合到卷积神经网络特征上。这些支持向量机充当对象检测器，取代通过微调学习的softmax分类器。在第三个训练阶段，学习边界盒回归器。
2. 培训在空间和时间上都很昂贵。对于SVM和有界盒回归器训练，从每张图像中的每个对象提议中提取特征并写入磁盘。对于非常深度的网络，如VGG16，这个过程需要2.5个gpu天来处理VOC07训练集的5k张图像。这些功能需要数百gb的存储空间。
3. 物体检测速度慢。在测试时，从每个测试图像中的每个对象提议中提取特征。使用VGG16进行检测需要47s /图像(在GPU上)。

R-CNN之所以慢，是因为它对每个对象提案执行卷积神经网络前向传递，而不共享计算。为了提高R-CNN的速度，提出了空间金字塔池网络(SPPnets)[11]。SPPnet方法对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提案进行分类。通过maxpooling将提案内部的特征映射部分提取为固定大小的输出(例如6 ×6)来提取提案的特征。将多个输出大小进行池化，然后像空间金字塔池化[15]一样进行连接。在测试时，SPPnet将R-CNN加速了10到100倍。由于更快的提案特征提取，训练时间也减少了3倍。

[1] All timings use one Nvidia K40 GPU overclocked to 875 MHz.

SPPnet也有明显的缺点。像R-CNN一样，训练是一个多阶段的管道，包括提取特征，对网络进行对数损失微调，训练支持向量机，最后拟合边界盒回归器。特征也被写入磁盘。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之前的卷积层。不出所料，这种限制(固定卷积层)限制了非常深的网络的准确性。

1.2. 贡献

我们提出了一种新的训练算法，该算法修正了R-CNN和SPPnet的缺点，同时提高了它们的速度和准确性。我们称这种方法为*快速R-CNN*，因为它的训练和测试速度相对较快。快速R-CNN方法有几个优点:

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

快速R-CNN是用Python和c++ (Caffe[13])编写的，在开源的MIT许可下可以在https://github.com/rbgirshick/fast-rcnn上获得。

2. 快速R-CNN架构和培训

图1展示了快速R-CNN架构。快速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)和最大池化层来处理整个图像，以生成一个conv特征图。然后，针对每个目标提议，RoI池化层从特征映射中提取固定长度的特征向量。每个特征向量被馈送到一个完全连接(fc)层的序列中，最终分支成两个兄弟输出层:一个层产生K个对象类加上一个全面的“背景”类的softmax概率估计，另一个层为K个对象类中的每一个输出四个实值数字。每一组4个值都为K个类中的一个编码了精细的边界盒位置。

2.1. RoI池层

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固定空间范围H ×W(例如，7 ×7)的小特征映射，其中H和W是独立于任何特定RoI的层超参数。在本文中，RoI是一个矩形窗口到一个转换特征映射。每个RoI由一个四元组(r, c, h, w)定义，该组指定其左上角(r, c)及其高度和宽度(h, w)。

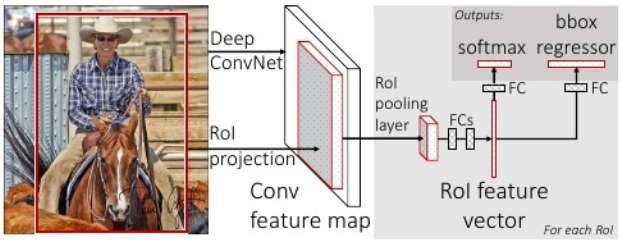

图1。快速R-CNN架构。一个输入图像和多个感兴趣区域(roi)被输入到一个全卷积网络中。每个RoI集合成一个固定大小的特征映射，然后通过全连接层(fc)映射到一个特征向量。该网络每个RoI有两个输出向量:softmax概率和每个类的边界盒回归偏移量。该体系结构在多任务损失下进行端到端训练。

RoI最大池化的工作原理是将h ×w RoI窗口划分为大小近似为h/ h ×w/W的子窗口组成的h ×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中。池化独立应用于每个特征映射通道，与标准max池化一样。RoI层只是SPPnets[11]中使用的空间金字塔池层的特殊情况，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. 从预训练的网络初始化

我们用三个预训练的ImageNet[4]网络进行实验，每个网络有5个最大池化层和5到13个conv层(网络细节见4.1节)。当一个预训练的网络初始化一个快速R-CNN网络时，它经历了三个转换。

首先，将最后一个最大池化层替换为一个RoI池化层，该层通过设置H和W与网络的第一个完全连接层兼容来配置(例如，对于VGG16, H = W = 7)。

其次，网络的最后一个完全连接层和softmax(经过1000路ImageNet分类训练)被替换为前面描述的两个兄弟层(K +1个类别和特定类别的边界盒回归器上的完全连接层和softmax)。

第三，将网络修改为接受两个数据输入:图像列表和这些图像中的roi列表。

2.3. 对检测进行微调

用反向传播训练所有网络权值是快速R-CNN的一项重要能力。首先，让我们阐明为什么SPPnet无法更新空间金字塔池层以下的权重。

根本原因是，当每个训练样本(即RoI)来自不同的图像时，通过SPP层的反向传播效率非常低，这正是R-CNN和SPPnet网络的训练方式。效率低下

源于这样一个事实，即每个RoI可能有一个非常大的接受域，通常跨越整个输入图像。由于前向传递必须处理整个接受场，因此训练输入很大(通常是整个图像)。

我们提出了一种更有效的训练方法，利用了训练过程中的特征共享。在快速R-CNN训练中，随机梯度下降(SGD)小批量分层采样，首先采样N个图像，然后从每个图像中采样R/N roi。关键的是，来自同一图像的roi在向前和向后传递中共享计算和内存。使N变小可以减少迷你批处理的计算。例如，当使用N = 2和R = 128时，所提出的训练方案大约比从128个不同的图像中采样一个RoI(即R-CNN和SPPnet策略)快64倍。

这种策略的一个问题是它可能会导致缓慢的训练收敛，因为来自同一图像的roi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，在N = 2和R = 128的情况下获得了很好的结果。

除了分层采样之外，快速R-CNN使用了一个精简的训练过程，其中一个微调阶段共同优化softmax分类器和边界盒回归器，而不是在三个单独的阶段训练softmax分类器、svm和回归器[9,11]。该过程的组成部分(损失、小批量采样策略、通过RoI池层的反向传播和SGD超参数)如下所述。

多任务损失(multitask loss)。快速R-CNN网络有两个兄弟输出层。第一个输出离散概率分布(每个RoI)，$p = (p_0, \cdots, p_K)$，超过K + 1个类别。像往常一样，p是通过对全连接层的K+1个输出的softmax来计算的。第二个兄弟层为每k个对象类输出边界盒回归偏移量，$t^k = t^k_x, t^k_y, t^k_w, t^k_h$，以k为索引。我们使用[9]中给出的$t^k$的参数化，其中$t^k$指定了相对于对象建议的尺度不变平移和对数空间高度/宽度移位。

每个训练RoI被标记为一个真值类u和一个真值边界盒回归目标v，我们在每个标记的RoI上使用一个多任务损失L来联合训练分类和边界盒回归:

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

其中$L_{\text{cls}}(p, u) = -\log p_u$为真类u的对数损失。

第二个任务损失L_{loc}，定义在u类的真正的有界盒回归目标的元组上，v = (vx, vy, vw, vh)，和一个预测的元组tu = (tux, tuy, tuw, tuh)，同样适用于u类。当u≥1时，Iverson括号指标函数[u≥1]的值为1，否则为0。按照惯例，包罗万象的背景类被标记为u = 0。对于背景roi，没有ground-truth边界框的概念，因此忽略L_{loc}。对于边界盒回归，我们使用损失

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

其中

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

与R-CNN和SPPnet中使用的L_2损失相比，它对异常值的敏感性较低。当回归目标无界时，使用L_2loss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。我们将真值回归目标v_i归一化，使其均值和单位方差为零。所有实验都使用λ= 1。

我们注意到[6]使用相关损失来训练一个分类不可知论的对象提议网络。与我们的方法不同，[6]提倡将定位和分类分离的双网络系统。OverFeat[19]、R-CNN[9]和SPPnet[11]也训练分类器和定界盒定位器，但是这些方法使用分阶段训练，我们认为这对于快速R-CNN来说是次优的(第5.1节)。

Mini-batch采样。在微调期间，每个SGD小批都是由N = 2个随机选择的图像构建的(作为通常的做法，我们实际上迭代数据集的排列)。我们使用大小为R = 128的小批量，从每个图像中采样64个roi。与[9]中一样，我们从具有交集/联合(IoU)重叠且ground-truth边界框至少为0.5的对象建议中获取25%的roi。这些roi包括用前景对象类标记的示例，*即*u≥1。剩余的roi从具有最大IoU的目标提案中采样，其接地真值在区间[0.1,0.5)，[11]之后。这些是背景示例，并标记为u = 0。较低的阈值0.1似乎充当了硬示例挖掘[8]的启发式。在训练过程中，图像以0.5的概率水平翻转。不使用其他数据增强。

通过RoI池层进行反向传播。反向传播通过RoI池层路由衍生。为了清楚起见，我们假设每个mini-batch (N = 1)只有一张图像，尽管扩展到N > 1很简单，因为前向传递独立地处理所有图像。

设$x_i \in R$为RoI池化层的第i个激活输入，y_{rj}为该层从第R个RoI池化层的第j个输出。RoI池化层计算yrj = xi*(r, j)，其中$i^*(r,j) = \text{argmax}_{i_0 \in R(r,j)} x_{i_0}$。R(R, j)是索引

输出单元y_{rj}Max池的子窗口的输入集合。一个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max池化为y_{rj}选择的argmax，则会累积偏导数$\partial L/\partial y_{rj}$。在反向传播中，$\partial L/\partial y_{rj}$的偏导数已经由RoI池化层之上的层的向后函数计算出来。

SGD超参数。用于softmax分类和定界盒回归的全连接层分别从标准差为0.01和0.001的零均值高斯分布初始化。偏置初始化为0。所有层的权重和偏差的每层学习率分别为1和2，全局学习率为0.001。当在VOC07或VOC12训练上训练时，我们运行SGD进行30k的小批迭代，然后将学习率降低到0.0001，并进行另外10k的迭代训练。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，如后面所述。动量为0.9，参数衰减为0.0005(在权重和偏差上)。

2.4. 尺度不变性

我们探索了实现尺度不变性目标检测的两种方法:(1)通过“蛮力”学习和(2)通过使用图像金字塔。这些策略遵循[11]中的两种方法。在蛮力方法中，在训练和测试期间，每个图像都以预定义的像素大小进行处理。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，使用图像金字塔对每个对象提案进行近似尺度归一化。在多尺度训练期间，我们在每次采样图像时随机采样一个金字塔尺度，遵循[11]，作为数据增强的一种形式。由于GPU内存的限制，我们只对较小的网络进行多尺度训练实验。

3. 快速R-CNN检测

一旦对快速R-CNN网络进行微调，检测就相当于向前传递(假设对象提议是预先计算的)。该网络将一张图像(或一个图像金字塔，编码为图像列表)和R个对象建议列表作为输入进行评分。在测试时，R通常在2000左右，尽管我们会考虑它更大的情况(≈45k)。当使用图像金字塔时，每个RoI被分配到比例，这样缩放后的RoI最接近[11]区域中的224^2像素。

对于每个测试RoI r，前向传递输出一个类后验概率分布p和一组相对于r的预测边界盒偏移量(K个类中的每一个都有自己的改进的边界盒预测)。我们使用估计概率$Pr(class = k \mid r) \triangleq p_k$为每个对象类别k分配一个检测置信度r。然后，我们使用R-CNN[9]中的算法和设置独立地对每个类执行非最大抑制。

3.1. 截断SVD更快的检测

对于整幅图像分类，与conv层相比，计算完全连接层所花费的时间更少。相反，对于检测，需要处理的roi数量很大，近一半的前向传递时间用于计算完全连接层(见图2)。通过使用截断的SVD压缩大型完全连接层，可以很容易地加速它们[5,23]。

在该技术中，由u ×v权矩阵W参数化的层近似因式分解为

$$W \approx U\Sigma_t V^T \quad (5)$$

使用SVD。在这个分解中，U是包含W的前t个左奇异向量的U ×t矩阵，Σ_t是包含W的前t个奇异值的t ×t对角矩阵，V是包含W的前t个右奇异向量的V ×t矩阵，截断SVD将参数计数从uv减少到t(U + V)。如果t远远小于min(u, v)，这可能是显著的。为了压缩网络，将W对应的单个全连接层替换为两个全连接层，它们之间没有非线性。这些层中的第一层使用权重矩阵$\Sigma_t V^T$(没有偏差)，第二层使用U(与W相关的原始偏差)。当roi数量很大时，这种简单的压缩方法可以提供很好的加速。

4. 主要结果

三个主要结果支持本文的贡献:

1. 最先进的VOC07、2010和2012地图
2. 与R-CNN, SPPnet相比，快速训练和测试
3. 在VGG16中微调转换层可以改善地图

4.1. 实验设置

我们的实验使用了三个在线可用的预训练ImageNet模型第一个是来自R-CNN[9]的CaffeNet(本质上是AlexNet[14])。我们交替引用

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

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**

表1。2007年VOC测试检测平均精度(%)。所有方法均使用VGG16。训练集键:07:VOC07训练，07 \ diff: 07无“难”样例，07+12:07与VOC12训练的并集。†SPPnet结果由[11]的作者准备。

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**

表2。2010年VOC测试检测平均精度(%)。BabyLearning使用基于[17]的网络。其他方法均使用VGG16。培训设置键:12:VOC12培训，道具。:专有数据集，12+seg: 12带有分割注释，07++12:VOC07训练、VOC07测试和VOC12训练的联合。

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**

表3。VOC 2012测试检测平均精度(%)。BabyLearning和NUS NIN c2000使用基于b[17]的网络。其他方法均使用VGG16。训练集键:见表2,Unk。:未知。

把这个CaffeNet称为S型，代表“小”。第二个网络是来自[3]的VGG CNN M 1024，它的深度和S一样，但是更宽。我们称这个网络模型为M，代表“medium”。最后的网络是[20]的非常深的VGG16模型。由于这个模型是最大的，我们称之为模型L.在本节中，所有的实验都使用单尺度训练和测试(s = 600;详见5.2节)。

4.2. VOC 2010年和2012年的结果

在这些数据集上，我们将Fast R-CNN(简称FRCN)与来自公共排行榜的comp4(外部数据)轨道上的顶级方法进行了比较(表2、表3)对于NUS NIN c2000和BabyLearning方法，目前没有相关的出版物，我们无法找到所使用的ConvNet架构的确切信息;它们是Network-in-Network设计[17]的变体。所有其他方法都是从相同的预训练VGG16网络初始化的。

Fast R-CNN以65.7%的mAP在VOC12上获得了最高的结果(额外数据为68.4%)。它也比其他方法快两个数量级，这些方法都是基于“慢”的R-CNN管道。在VOC10上，SegDeepM[25]实现了比Fast R-CNN更高的mAP(67.2%比66.1%)。SegDeepM是在VOC12训练和分割注释上进行训练的;它的目的是提高R-CNN的准确性，通过使用马尔科夫随机场推理R-CNN检测和从O_2P[1]语义分割方法分割。Fast R-CNN可以换成SegDeepM代替R-CNN，这样可能会得到更好的结果。当使用扩大的07++12训练集(见表2标题)时，Fast R-CNN的mAP增加到68.8%，超过SegDeepM。

4.3. 2007年VOC结果

在VOC07上，我们比较了Fast R-CNN与R-CNN和SPPnet。所有方法都从相同的预训练VGG16网络开始，并使用有界盒回归。VGG16 SPPnet结果由[11]的作者计算。SPPnet在训练和测试期间使用五个量表。Fast R-CNN对SPPnet的改进表明，尽管Fast R-CNN使用单尺度训练和测试，微调转换层提供了mAP的巨大改进(从63.1%提高到66.9%)。R-CNN的mAP达到66.0%。作为一个小问题，SPPnet没有在PASCAL中标记为“困难”的示例进行训练。删除这些例子将快速R-CNN mAP提高到68.1%。其他所有实验都使用“困难”的例子。

3 http://host.robots.ox.ac.uk:8080/leaderboard(accessed April 18, 2015)

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结果。表4比较了Fast R-CNN、R-CNN和SPPnet对VOC07的训练时间(小时)、测试速率(每幅图像秒)和mAP。对于VGG16, Fast R-CNN处理图像的速度比不截断SVD的R-CNN快146倍，截断SVD的处理速度快213倍。训练时间减少了9倍，从84小时减少到9.5小时。与SPPnet相比，Fast R-CNN训练VGG16的速度提高了2.7倍(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倍，有了截断SVD的测试速度提高了10倍。快速R-CNN还消除了数百gb的磁盘存储空间，因为它没有缓存功能。

	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	-	-	-	-

表4.。Fast R-CNN、R-CNN和SPPnet中相同模型的运行时比较。快速R-CNN使用单尺度模式。SPPnet使用[11]中指定的五个尺度。†Timing由[11]的作者提供。在Nvidia K40 GPU上测量时间。

截断的SVD。截断的SVD可以将检测时间减少30%以上，而mAP仅下降很小(0.3个百分点)，并且无需在模型压缩后执行额外的微调。图2说明了如何在VGG16的fc6层中使用来自25088 ×4096矩阵的前1024个奇异值和来自4096×4096 fc7层的前256个奇异值在mAP中损失很小的情况下减少运行时间。如果在压缩后再次微调，则可以使用更小的mAP下降来进一步加速。

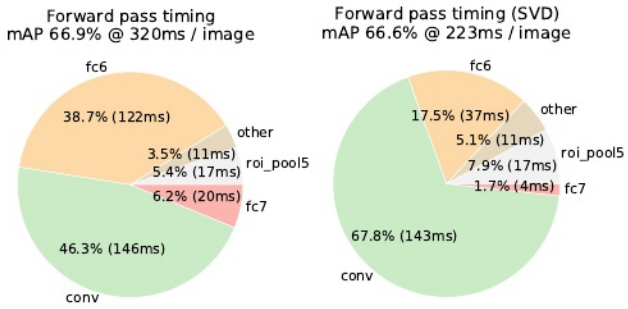

图2。截断SVD前后VGG16的时序。在SVD之前，完全连接层fc6和fc7占用45%的时间。

4.5. 哪些层需要微调?

对于SPPnet论文[11]中考虑的深度较低的网络，仅对完全连接的层进行微调似乎足以获得良好的精度。我们假设这一结果不适用于非常深的网络。为了验证微调卷积层对VGG16很重要，我们使用Fast R-CNN进行微调，但*冻结*13个卷积层，以便只有完全连接的层学习。这种烧蚀模拟了单尺度SPPnet训练，将*mAP从66.9%降低到61.4%*(表5)。该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

表5。限制VGG16微调层的影响。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个量表，以显著的(7倍)速度成本获得。

这是否意味着*所有*的转换层都应该微调?简而言之，*不是*。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(一个众所周知的事实[14])。允许或不允许conv1学习对mAP没有有意义的影响。对于VGG16，我们发现只需要从conv31更新层(13个conv层中的9个)。这一观察结果是实用的:(1)与从conv31学习相比，从conv21更新会使训练速度减慢1.3倍(12.5小时vs. 9.5小时);(2)从conv11更新会占用GPU内存。从conv21向上学习时，mAP的差异仅为+0.3点(表5，最后一列)。本文中所有快速R-CNN结果均使用VGG16微调层conv31及以上;所有使用模型S和M微调层conv2及以上的实验。

5. 设计评价

我们进行了实验，以了解R-CNN与R-CNN和SPPnet相比有多快，并评估设计决策。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，因为它避免了管理一系列按顺序训练的任务。但它也有改善结果的潜力，因为任务通过共享表示(ConvNet)[2]相互影响。多任务训练有提高了Fast R-CNN的目标检测精度?

为了测试这个问题，我们在Eq. 1(即设置)中训练仅使用分类损失L_{cls}的基线网络

	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**

表6。多任务训练(每组第四列)比分段训练(每组第三列)提高了mAP。

λ= 0)，这些基线在表6中每组的第一列中打印为模型S、M和L。请注意，这些模型*没有*边界盒回归量。接下来(每组第二列)，我们采用用多任务损失(Eq. 1， λ= 1)训练的网络，但我们在测试时禁*用*边界盒回归。这隔离了网络的分类准确性，并允许与基线网络进行苹果对苹果的比较。

在所有三个网络中，我们观察到，相对于单独的分类训练，多任务训练提高了纯分类准确率。改善范围从+0.8到+1.1 mAP点，显示出多任务学习的一致积极效果。

最后，我们采用基线模型(仅使用分类损失进行训练)，附加上边界盒回归层，并使用L_{loc}训练它们，同时保持所有其他网络参数冻结。每组中的第三列显示了这种*分阶段*训练方案的结果:mAP比第一列有所改进，但分阶段训练不如多任务训练(每组第四列)。

5.2. Scale invariance: to brute force还是finesse?

我们比较了实现尺度不变对象检测的两种策略:蛮力学习(单尺度)和图像金字塔(多尺度)。在这两种情况下，我们都将图像的尺度s定义为其*最短*边的长度。

所有的单尺度实验都使用s = 600像素;对于某些图像，S可能小于600，因为我们将最长的图像边限制在1000像素，并保持图像的宽高比。选择这些值是为了使VGG16在微调期间适合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并不是我们主要关心的问题。我们注意到PASCAL图像平均为384 ×473像素，因此单尺度设置通常会将图像样本增加1.6倍。因此，RoI池化层的平均有效步幅为≈10个像素。

在多尺度设置中，我们使用[11](s∈{480,576,688,864,1200})中指定的相同的五个尺度，以便与SPPnet进行比较。然而，我们将最长的边限制在2000像素，以避免超出GPU内存。

表7显示了模型S和M在用一个或五个尺度进行训练和测试时的情况。也许[11]中最令人惊讶的结果是，单尺度检测的表现几乎和多尺度检测一样好。我们的发现证明了

	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**

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

巩固他们的结果:深度卷积神经网络擅长于直接学习尺度不变性。多尺度方法只提供了少量的mAP增加，但花费了大量的计算时间(表7)。在VGG16(模型L)的情况下，由于实现细节的限制，我们只能使用单一尺度。然而，它达到了66.9%的mAP，略高于R-CNN b[10]报道的66.0%，尽管R-CNN使用“无限”的尺度，即每个提案都被扭曲成一个标准的大小。

由于单尺度处理在速度和精度之间提供了最好的权衡，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会有所提高。Zhu等人[24]发现DPM [8] mAP在仅仅几百到几千个训练样例之后就饱和了。在这里，我们用VOC12训练集增强VOC07训练集，大约将图像数量增加三倍，达到16.5k，以评估Fast R-CNN。扩大训练集将VOC07测试的mAP从66.9%提高到70.0%(表1)。当在该数据集上训练时，我们使用60k次小批迭代而不是40k次。

我们对VOC10和2012进行了类似的实验，为此，我们从VOC07训练、测试和VOC12训练的联合中构建了一个21.5k图像的数据集。在此数据集上进行训练时，我们使用100k SGD迭代，并且每40k迭代(而不是每30k迭代)将学习率降低0.1倍。对于VOC10和2012年，mAP分别从66.1%和65.7%提高到68.8%和68.4%。

5.4. 支持向量机优于softmax吗?

快速R-CNN使用在微调期间学习的softmax分类器，而不是训练一对一的线性支持向量机

就像在R-CNN和SPPnet中做的那样。为了理解这种选择的影响，我们在快速R-CNN中实现了带有硬负挖掘的事后支持向量机训练。我们使用与R-CNN相同的训练算法和超参数。

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**

表8.快速R-CNN与softmax与支持向量机(VOC07 mAP)。

表8显示，softmax在所有三种网络上的表现都略优于支持向量机，高出+0.1到+0.8个mAP点。这种影响很小，但它表明，与之前的多阶段训练方法相比，“一次”微调是足够的。我们注意到softmax，不像one-vs-rest支持向量机，在对RoI进行评分时引入了类之间的竞争。

5.5. 提案越多越好吗?

有(大致上)两种类型的对象检测器:那些使用*稀疏*的对象建议集(例如，选择性搜索[21])和那些使用*密集*集(例如，DPM[8])。对稀疏提议进行分类是一种*级联*[22]，其中提议机制首先拒绝大量的候选对象，留给分类器一个小的集来评估。当应用于DPM检测[21]时，该级联提高了检测精度。我们发现证据表明，提议分类器级联也提高了快速R-CNN的准确性。

使用选择性搜索的*质量模式*，我们从每张图像扫描1k到10k个提案，*每次重新训练*和*重新测试*模型m。如果提案服务于纯粹的计算作用，那么增加每张图像的提案数量应该不会损害mAP。

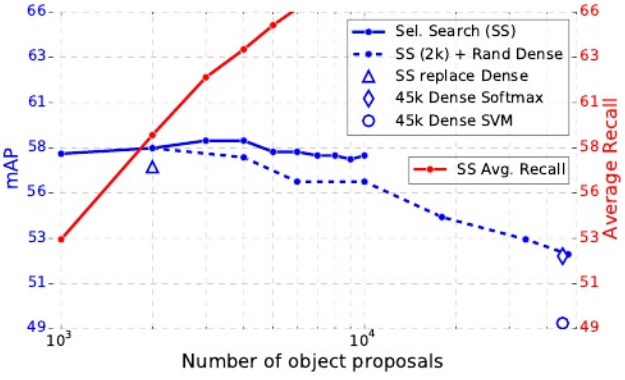

图3。各种提案方案的VOC07 mAP和AR测试。

我们发现mAP随着提案数的增加而上升，然后略有下降(图3，蓝色实线)。这个实验表明，用更多的提议淹没深度分类器对准确率没有帮助，甚至会有轻微的伤害。

如果不实际运行实验，这个结果很难预测。测量对象提议质量的最先进技术是平均召回率(AR)[12]。对于使用R-CNN的几种提议方法，*当每张图像使用固定数量的提议时*，AR与mAP具有良好的相关性。图3显示，由于每张图像的建议数量不同，AR(红色实线)与mAP的相关性并不好。AR必须谨慎使用;提案越多，AR值越高，并不意味着mAP会增加。幸运的是，模型M的训练和测试只需要不到2.5小时。因此，快速R-CNN能够有效、直接地评估对象提议mAP，这比代理度量更可取。

我们还研究了快速R-CNN在使用*密集*生成的盒子(超过比例，位置和宽高比)时，以大约45k个盒子/图像的速率。这个密集集足够丰富，当每个选择性搜索框被其最近的(IoU)密集框替换时，mAP仅下降1点(为57.7%，图3，蓝色三角形)。

密集框的统计数据与选择性搜索框的统计数据不同。从2k个选择性搜索框开始，我们在*添加*1000 ×{2,4,6,8,10,32,45}个密集框的随机样本时测试mAP。对于每个实验，我们重新训练和重新测试模型m。当这些密集的搜索框被添加时，mAP比添加更多选择性搜索框时下降得更强烈，最终达到53.0%。

我们还只使用密集盒(45k /图像)训练和测试快速R-CNN。这个设置产生52.9%的mAP(蓝色钻石)。最后，我们检查是否需要使用硬负挖掘的支持向量机来处理密集的箱形分布。支持向量机的表现更差:49.3%(蓝色圆圈)。

5.6. MS COCO初步结果

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

6. 结论

本文提出了快速R-CNN，它是对R-CNN和SPPnet的一种干净、快速的更新。除了报告最先进的检测结果外，我们还提出了详细的实验，我们希望提供新的见解。特别值得注意的是，稀疏对象建议似乎可以提高检测器的质量。这个问题在过去太昂贵(时间上)而无法探究，但在快速R-CNN中变得实用。当然，可能存在尚未被发现的技术，可以让密集的盒子表现得和稀疏的提议一样好。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。我感谢kaim He、Larry Zitnick和Piotr Doll´ar的有益讨论和鼓励。

参考文献

[1] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 5

[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. 5

[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, 7, 8

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 3, 4, 8

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *TPAMI*, 2015. 5, 7, 8

[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. Doll´ar, and B. Schiele. What makes for effective detection proposals? *arXiv preprint arXiv:1502.05082*, 2015. 8

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Gir-shick, 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, 6

[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. 5

[18] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick. Microsoft COCO: common objects in context. *arXiv e-prints*, arXiv:1405.0312 [cs.CV], 2014. 8

[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, 5

[21] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 8

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 8

[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. 7

[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, 5