快速的区域卷积网络方法（Fast R-CNN）

Ross Girshick
Microsoft Research
rbg@microsoft.com

摘 要

本文提出了一种基于快速的区域卷积网络方法（Fast R-CNN）用于物体检测。 快速 R-CNN 建立在先前的工作基础上，以使用深度卷积网络有效地分类对象提议。 与之前的工作相比，Fast R-CNN 采用了多种创新技术来提高训练和测试速度，同时提高了检测精度。 快速 R-CNN 训练非常深的 VGG16 网络比 R-CNN 快 9 倍，在测试时间上快 213 倍，并在 PASCAL VOC 2012 上实现更高的 mAP。与 SPPnet 相比，Fast R-CNN 在 VGG16 上训练速度提升至 3 倍，测试速度上快 10 倍，并且更准确。 FastR-CNN 在 Python 和 C ++中实现（使用 Caffe），可在开源 MIT License 下获得：
https: //github.com/rbgirshick/fast-rcnn.

1. 介绍

最近，深度 ConvNets [14,16]显著改善了图像分类[14]和物体检测[9,19]的准确性。与图像分类相比，对象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，当前的方法（例如，[9,11,19,25]）在多级管道中训练模型，这些方法是缓慢且不优雅的。

复杂性的产生是因为检测需要对象进行精确定位，从而产生两个主要的挑战。首先，必须处理许多候选对象位置（通常称为“区域建议”）。其次，这些可选区域只提供粗略的定位，必须对其进行细化以实现精确定位。解决这些问题往往会影响速度，准确性或简易性。

在本文中，我们简化了最先进的基于 ConvNet 的物体探测器的训练过程[9,11]。我们提出了一个单阶段训练算法，该算法共同学习对建议区域进行分类并改进其空间位置。

由此产生的方法可以比 R-CNN [9]更快地训练一个非常深的检测网络（VGG16 [20]），比 SPPnet [11]快 3 倍。在运行时，检测网络以 0.3s 处理图像（不包括生成建议区域时间）同时实现 PASCAL VOC 2012 [7]的最高精度，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 ConvNet 对建议区域进行分类，实现了出色的对象检测精度。 然而，R-CNN 有明显的缺点：

1. **训练是一个多阶段的管道。** R-CNN 首先使用对数损失对建议区域进行 ConvNet 微调。 然后，它应用 SVM 适应了 ConvNet 特征。 这些 SVM 充当对象检测器，取代了通过微调学习的 softmax 分类器。 在第三个训练阶段，学习边界框回归量。
2. **训练在空间和时间上都很昂贵。** 对于 SVM 和边界框回归训练，将从每个图像中的每个建议区域中提取特征并将其写入磁盘。 对于非常深的网络，如 VGG16，这个过程需要 2.5 个 GPU-day 才能获得 VOC07 trainval 集的 5k 图像。 这需要数百 GB 的存储空间。
3. **物体检测很慢。** 在测试时，从每个测试图像中的每个建议区域中提取特征。 使用 VGG16 进行检测需要 47 秒/图像（在 GPU 上）。

R-CNN 很慢，因为它为每个建议区域执行 ConvNet 前向传递，而不共享计算。 空间金字塔汇集网络（SPPnets）[11]被提议通过共享计算来加速 R-CNN。 SPPnet 方法计算整个输入图像的卷积特征图，然后使用从共享特征图提取的特征向量对每个对象特征进行分类。 通过将建议区域内的特征映射的部分最大化为固定大小的输出（例如，6×6）来提取特征以用于提议。 汇总多个输出大小，然后在空间金字塔池中连接[15]。 SPPnet 在测试时将 R-CNN 加速 10 到 100 倍。 由于更快的特征提取，训练时间也减少 3 倍。

[1]所有时序都使用一个超频到 875 MHz 的 Nvidia K40 GPU。

SPPnet 也有明显的缺点。 与 R-CNN 一样，训练是一个多阶段管道，包括提取特征，微调网络丢失，训练 SVM，最后拟合边界框回归。 特征也会被写入磁盘。 但与 R-CNN 不同，[11]中提出的微调算法无法更新空间金字塔汇集之前的卷积层。 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，它可以修复 R-CNN 和 SPPnet 的缺点，同时提高它们的速度和准确性。 我们称这种方法为快速 R-CNN，因为它训练和测试的速度相对较快。 快速 R-CNN 方法有几个优点：

1. 比 R-CNN，SPPnet 更高的检测质量（mAP）
2. 训练是单阶段的，使用多任务损失
3. 训练可以更新所有网络层
4. 功能缓存不需要磁盘存储

快速 R-CNN 是用 Python 和 C ++编写的（Caffe [13]），可以在开源 MIT Li-cense 下获得：https://github.com/rbgirshick/ fast-rcnn。

2. Fast R-CNN 架构和训练

图 1 说明了快速 R-CNN 架构。快速 R-CNN 网络将整个图像和一组建议区域作为输入。网络首先使用几个卷积（conv）和最大池化层处理整个图像，以产生转换特征映射。然后，对于每个建议区域，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系列完全连接层，最终分支到两个兄弟输出层：一个层产生对 K 类对象类加上一个全局的“背景”类的 softmax 概率估计，然后另一个层输出对每个 K 对象类的四个实数值，这 4 个值是对 K 类之一的精细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 是独立的层超参数，不依赖任何特定的 RoI。在本文中，RoI 是一个转换为转换特征映射的矩形窗口。每个 RoI 由四元组（r,c,h,w）定义，指定其左上角坐标（r,c）及其高度和宽度（h,w）。

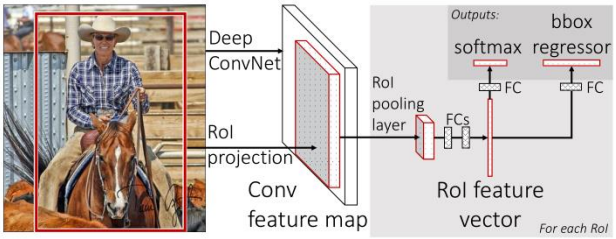

图 1.快速 R-CNN 架构。 输入图像和多个感兴趣区域（RoI）被输入到完全卷积网络中。 每个 RoI 汇集到固定大小的特征映射中，然后通过完全连接层（FC）映射到特征向量。 网络每个 RoI 有两个输出向量：softmax 概率和每类边界框回归偏移。 该架构通过多任务损失进行端到端的训练。

RoI max pooling 通过尺寸大约为 h/H × w/W 的子窗口将 h×w 的 RoI 窗口划分为 H×W 的网格，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来工作。池标准独立应用于每个要素图通道，如标准最大池中所示。 RoI 层只是 SPPnets [11]中使用的空间金字塔池层的特例，其只有一个金字塔层。我们使用[11]中给出的池窗口计算。

2.2. 从预训练的网络初始化

我们尝试了三个预先训练过的 ImageNet [4]网络，每个网络有五个最大池化层，五到十三个转换层（参见 4.1 节网络细节）。当预训练的网络初始化快速 R-CNN 网络时，它经历三次转换。

首先，最后一个最大池化层由 RoI 池化层替换，该池化层通过将 H 和 W 设置为与网络的第一个完全连接层相兼容来配置（例如，对于 VGG16，H = W = 7）。

其次，网络的最后一个完全连接层和 soft-max（经过 1000 路 ImageNet 分类训练）被前面描述的两个兄弟层替换（一个完全连接层和超过 K + 1 类别的 softmax 层和特定分类的边界框回归量）。

第三，修改网络以获取两个数据输入：图像列表和那些图像中的 RoI 列表。

2.3. 微调检测

使用反向传播训练所有网络权重是快速 R-CNN 的重要功能。首先，让我们阐明为什么 SPPnet 无法更新空间金字塔池化层下的权重。

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 层的反向传播非常低效，这正是 R-CNN 和 SPPnet 网络的训练方式。这种效率低下

源于每个 RoI 可能事实上具有非常大的感受野，通常跨越整个输入图像。由于前向传输必须处理整个感受野，因此训练需要的输入很大（通常是整个图像）。

我们提出了一种更有效的训练方法，利用训练期间的特征共享。在 Fast R-CNN 训练中，随机梯度下降（SGD）的小批量使用分层采样，首先采样 N 个图像，然后通过从每个图像采样 R/N 个 RoI。重要的是，来自相同图像的 RoI 在前向和后向传递中共享计算和存储器。这会使 N 减少 mini-batch 的计算。例如，当使用 N = 2 且 R = 128 时，所提出的训练方案比从 128 个不同图像采样一个 RoI（即，R-CNN 和 SPPnet 策略）快大约 64 倍。

对此策略的一个担忧是它可能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，使用 N = 2 和 R = 128 获得了良好的结果。

除了分层采样之外，Fast R-CNN 使用简化的训练过程和一个微调阶段，共同优化 softmax 分类器和边界框重构器，而不是在三个单独的阶段训练 softmax 分类器，SVM 和回归器[9,11]。该程序的组成部分（损失，小批量抽样策略，通过 RoI 汇集层的反向传播和 SGD 超参数）如下所述。

多任务损失。快速 R-CNN 网络具有两个兄弟输出层。第一个输出离散概率分布（每 RoI），p =（p0, … , pK），超过 K + 1 个类别。通常，p 由完全连接层的 K + 1 个输出上的 softmax 计算。第二个兄弟层输出边界框回归偏移，$t^k = (t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}})$，对于每个 K 对象类，由 k 索引。 我们使用[9]中给出的t^k的参数化，其中t^k指定相对于建议区域的尺度不变的平移和对数空间高度/宽度偏移。

每个训练 RoI 都标有完全真实类 u 和完全真实边界框回归目标 v。我们在每个标记的 RoI 上使用多任务损失 L 来联合训练分类和边界框回归：

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

其中$L_{\mathrm{cls}}(p, u) = -\log p_u$是真实类 u 的对数损失。

第二个任务损失L_{loc}是针对类 u，v =$(v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{w}}, v_{\mathrm{h}})$的真实边界框回归目标的元组定义的，并且预测的元组$t^u = (t^u_{\mathrm{x}}, t^u_{\mathrm{y}}, t^u_{\mathrm{w}}, t^u_{\mathrm{h}})$，再次针对类 u。 当 u ≥ 1 时，Iverson 括号指示器函数[u ≥ 1]的计算结果为 1，否则为 0。 按照惯例，全局背景类被标记为 u = 0。对于背景 RoI，没有完全真实的边界框，因此L_{loc}被忽略。 对于边界框回归，我们使用损失

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

其中

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

是一种强大的 L1 损耗，对于异常值的敏感度低于 R-CNN 和 SPPnet 中使用的 L2 损耗。当回归目标无限制时，L2 损失训练可能需要仔细调整学习率以防止爆炸梯度。式 3 消除了这种敏感性。

在方程 1 中的超参数 λ 控制两个任务损失之间的平衡。我们将真实回归目标v_i标准化为零均值和单位方差。所有实验都使用λ = 1。

我们注意到[6]使用相关的损失来训练一个与类无关的区域建议网络。与我们的方法不同，[6]提出了一种双网络系统，它将本地化和分类分开。 OverFeat [19]，R-CNN [9]和 SPPnet [11]也训练分类器和边界框定位器，但是这些方法使用阶段式训练，我们表明它对于快速 R-CNN 来说是次优的（第 5.1 节）。

小批量抽样。在微调期间，每个 SGD 小批量由 N = 2 个图像构成，随机选择单一形式（通常的做法是，我们实际上只对数据集的排列进行了调整）。我们使用尺寸为 R = 128 的小批量，从每个图像中采样 64 个 RoI。与[9]中一样，我们从具有交叉联合（IoU）重叠的建议区域中选取 RoI 值为 25%并且与至少 0.5 的真实实例边界框重叠。这些 RoI 包括用前景对象类标记的示例，即 u ≥ 1。剩余的 RoI 是从区域事件中采样的，其具有完全真实的特性，且 IoU 在区间[0.1, 0.5)中，参阅[11]。这些是背景示例，并标记为 u = 0。0.1 的下限阈值似乎充当了困难实例挖掘的启发式算法[8]。在训练期间，图像以 0.5 的概率水平翻转，没有使用其他数据扩充。

通过 RoI 池化层反向传播。反向传播路径通过 RoI 池化层导出。为清楚起见，我们假设每个小批量只有一个图像（N = 1），但 N > 1 的扩展是直截了当的，因为前向传递独立地处理所有图像。

令$x_i \in \mathbb{R}$为进入 RoI 池层的第 i 个激活输入，并让y_{rj}为 r 层 RoI 的第 j 个输出。RoI 池化层计算$y_{rj} = x_{i^*(r,j)}$，其中$i^*(r, j) = \mathrm{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$。$\mathcal{R}(r, j)$是

是子窗口中输入单元 y_{rj} 最大池化层的输入的索引集。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：

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

换言之，对于每个小批量 RoI r 和每个合并输出单元 y_{rj} ，如果 i 是通过最大合并为 y_{rj} 选择的 argmax，则偏导数 $\partial L/\partial y_{rj}$ 就会累积。在反向传播中，偏导数 $\partial L/\partial y_{rj}$ 已经由顶部的 ROI 池化层的反向函数计算。

SGD 超参数。用于 softmax 分类和边界框回归的完全连接层分别从具有标准偏差 0.01 和 0.001 的零均值高斯分布初始化。偏差初始化为 0。所有层使用权重的每层学习率为 1，偏差为 2，全局学习率为 0.001。在训练集 VOC07 或 VOC12 trainval 上时，我们运行 SGD 进行 30k 小批量迭代，然后将学习率降低到 0.0001 并进行另外 10k 次迭代训练。当我们在更大的数据集上训练时，我们运行 SGD 以进行更多迭代，如稍后所述。使用 0.9 的动量和 0.0005 的参数衰减（关于权重和偏差）。

2.4. 尺度不变性

我们探索了实现尺度不变对象检测的两种方法：（1）通过“强力”学习和（2）通过使用图像金字塔。这些策略遵循[11]中的两个方法。在蛮力方法中，在训练和测试期间以预定义的像素大小处理每个图像。网络必须直接从训练数据中学习尺度不变的物体检测。

相反，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，图像金字塔用于近似地缩放规范化每个建议区域。在多尺度训练期间，我们在每次采样图像时随机采样金字塔尺度[11]，作为数据增强的一种形式。由于 GPU 内存限制，我们仅针对较小的网络进行多尺度训练。

3. 快速 R-CNN 检测

一旦对快速 R-CNN 网络进行微调，检测就会比运行正向传递更多（预先计算出假设建议区域）。网络将图像（或图像金字塔，编码为图像列表）和 R 对象提议列表作为输入。在测试时间上，R 通常在 2000 左右，尽管我们会考虑它更大（≈45k）的情况。当使用图像金字塔时，每个 RoI 按一定比例，使得缩放的 RoI 最接近区域中的 224^2 像素[11]。

对于每个测试 RoI r，前向传递输出类后验概率分布 p 和一组相对于 r 的预测边界框偏移（每个 K 类获得其自己的精细边界框预测）。我们对每个对象类 k 使用估计概率 $\Pr(\text{class} = k \mid r) \triangleq p_k$ 去分配 r 的检测置信度。然后，我们使用来自 R-CNN [9]的算法和设置，为每个类独立地执行非最大值抑制。

3.1. 截断 SVD 以加快检测速度

对于全图像分类，与 conv 层相比，计算完全连接层所花费的时间很少。相反，为了检测，要处理的 RoI 的数量很大，并且将近一半的正向通过时间用于计算完全连接层（参见图 2）。通过用截断的 SVD 压缩它们可以很容易地加速大的完全连接层[5,23]。

该技术中，由 u×v 权重矩阵 W 参数化的层近似地分解为

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

使用的是 SVD。在这个因式分解中，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，并且 V 是包含 W 的前 t 个右奇异向量的 v×t 矩阵。截断的 SVD 将参数计数从 uv 减少到 t（u + v），如果 t 远小于 min（u, v），这将会很重要。为了压缩网络，对应于 W 的单个完全连接层被两个完全连接的层代替，它们之间没有非线性。这些层中的第一层使用权重矩阵 $\Sigma_t V^T$ （并且没有偏差），第二层使用 U（原始的与 W 相关联的 bias）。当 RoI 数量很大时，这种简单的压缩方法可以提供良好的加速。

4.主要结果

三个主要结果支持本文的贡献：

1.关于 VOC07,2010 和 2012 的最新 mAP

2.与 R-CNN，SPPnet 相比，快速训练和测试

3.VGG16 中的精细调整转换层提高了 mAP

4.1. 实验装置

我们的实验使用了三种可在线获得的预先训练的 ImageNet 模型。第一种是来自 R-CNN 的 CaffeNet（基本上是 AlexNet [14]）[9]。我们另外参考

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. **VOC 2007** 测试检测平均精度（%）。 所有方法都使用 VGG16。 训练集关键：**07**：VOC07 trainval; **07 \ diff**：没有“困难”例子的 **07**; **07 + 12**：联合 07 和 VOC12 trainval。 SPPnet 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**

表 2. **VOC 2010** 测试检测平均精度（%）。 BabyLearning 使用基于[17]的网络。 所有其他方法都使用 VGG16。 训练集关键：**12**：VOC12 trainval；**Prop.**：专有数据集，**12+seg**：带分段注释的 **12**，**07++12**：VOC07 trainval，VOC07 测试和 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**

表 3. **VOC 2012** 测试检测平均精度（%）。 BabyLearning 和 NUS_NIN_c2000 使用基于[17]的网络。 所有其他方法都使用 VGG16。 训练集密钥：见表 2，**Unk.**：未知。

这个 CaffeNet 作为模型 S，用于“small”。第二个网络是来自[3]的 VGG_CNN_M_1024，其深度与 S 相同，但更宽。我们将此网络模型称为 M，称为“中等”。最终网络是来自[20]的非常深的 VGG16 模型。由于这个模型是最大的，我们称之为模型 L。在本节中，所有实验都使用单一规模的训练和测试（s = 600;详见 5.2 节）。

4.2. VOC 2010 和 2012 年结果

在这些数据集上，我们将 Fast R-CNN（简称 FRCN）与公共排行榜的 comp4（外部数据）轨道上的顶级方法进行比较（表 2，表 3）。对于 NUS_NIN_c2000 和 BabyLearning 方法，目前没有相关的出版物，我们无法找到有关所用 ConvNet 架构的确切信息;它们是网络设计的变种[17]。所有其他方法都是从相同的预先训练的 VGG16 网络初始化的。

快速 R-CNN 在 VOC12 上取得了最高成果，mAP 为 65.7%（额外数据为 68.4%）。它比其他方法快两个数量级，这些方法都基于“慢速”R-CNN 流水线。关于 VOC10，SegDeepM [25] 实现了比快速 R-CNN 更高的 mAP（67.2%对 66.1%）。 SegDeepM 接受了 VOC12 trainval 加分段注释的训练;它被设计用于通过使用马尔可夫随机场来推理 R-CNN 的准确度来推理来自 O2P [1]语义分割方法的 R-CNN 检测和分割。可以将快速 R-CNN 换成 SegDeepM 来代替 R-CNN，这可以产生更好的结果。当使用扩大的 07++12 训练集（见表 2 标题）时，快速 R-CNN 的 mAP 增加到 68.8%，超过 SegDeepM。

4.3. VOC 2007 结果

在 VOC07 上，我们将 Fast R-CNN 与 R-CNN 和 SPPnet 进行比较。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 结果由[11]的作者计算。 SPPnet 在训练和测试期间使用五个量表。快速 R-CNN 相对于 SPPnet 的改进说明即使快速 R-CNN 使用单一规模的训练和测试，对转换层进行微调也可以使 mAP 有很大的改进（从 63.1%到 66.9%）。 R-CNN 的 mAP 达到 66.0%。作为一个小问题，SPPnet 在没有 PASCAL 标记为“困难”的例子的情况下接受了训练。删除这些示例可将快速 R-CNN 的 mAP 提高到 68.1%。所有其他实验都使用“困难”的例子。

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原因。 表 4 比较了快速 R-CNN，R-CNN 和 SPPnet 之间的训练时间（小时），测试速率（每个图像的秒数）和 VOC07 上的 mAP。 对于 VGG16，快速 R-CNN 处理图像比没有截断 SVD 的 R-CNN 快 146 倍，并且使用它处理速度提升 213 倍。 训练时间减少 9 倍，从 84 小时减少到 9.5 小时。 与 SPPnet 相比，Fast R-CNN 更快地训练 VGG16 2.7 倍（9.5 对 25.5 小时），并且在没有截断 SVD 的情况下更快地测试 7 次，或者使用它更快地测试 10 次。 快速 R-CNN 还消除了数百 GB 的磁盘存储空间，因为它不用缓存功能。

	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.快速 R-CNN，R-CNN 和 SPPnet 中相同模型之间的运行时比较。快速 R-CNN 使用单比例模式。SPPnet 使用[11]中指定的五个标度。时间由[11]的作者提供。时间是在 Nvidia K40 GPU 上测量的。

截断 SVD。 截断的 SVD 可以将检测时间减少 30%以上，mAP 只有很小的（0.3%年龄点）下降，并且在模型压缩后无需执行额外的微调。 图 2 示出了如何使用来自 VGG16 的 fc6 层中的 25088×4096 矩阵的前 1024 个奇异值，以及来自 4096×4096 大小的 fc7 层的前 256 个奇异值如何减少运行时间而让 mAP 损失很小。 如果在压缩后再次进行微调，则可以进一步加速 mAP 中的小幅下降。

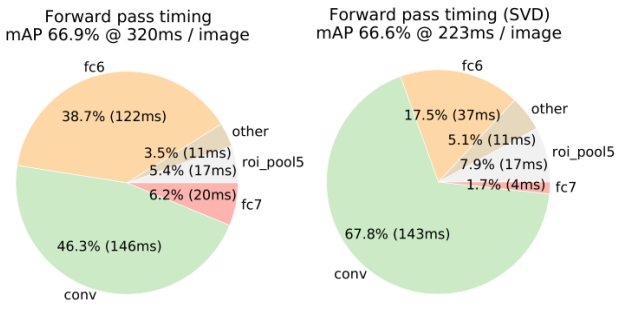

图 2.截断 SVD 之前和之后 VGG16 的时序。 在没有截断 SVD 时，完全连接层 fc6 和 fc7 占用了 45%的时间。

4.5. 哪些层要微调？

对于 SPPnet 论文 [11]中考虑的深度较低的网络，仅对完全连接层进行微调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于 VGG16 非常重要，我们使用快速 R-CNN 进行微调，但冻结了 13 个转换层，以便只有完全连接层才能学习。 该消融模拟单级 SPPnet 训练并将 mAP 从 66.9%降低至 61.4%（表 5）。 该实验验证了我们的假设：通过 RoI 汇集层进行训练对于非常深的网络非常重要。

	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.限制哪些层针对 VGG16 微调的效果。 微调 fc6 模拟 SPPnet 训练算法[11]，但使用单一尺度。 使用五个量表获得 SPPnet **L** 结果，成本显着（7）。

这是否意味着所有转换层都应该进行微调？总之，没有。在较小的网络（S 和 M）中，我们发现 conv1 是通用的并且与任务无关（众所周知的事实[14]）。允许 conv1 学习或不学习对 mAP 没有任何有意义的影响。对于 VGG16，我们发现只需更新 conv3_1 及更高层（13 个转换层中的 9 个）的层。这种观察是务实的：（1）与从 conv3_1 学习相比，从 conv2_1 更新使训练减慢 1.3 倍（12.5 小时对 9.5 小时）；（2）从 conv1_1 更新超出 GPU 内存。从 conv2_1 向上学习时，mAP 的差异仅为+0.3 分（表 5，最后一栏）。本文使用 VGG16 微调层 conv3_1 及以上的所有快速 R-CNN 结果;模型 S 和 M 的所有实验都对 conv2 及以上层进行微调。

5. 设计评估

我们进行了实验，以了解 Fast R-CNN 与 R-CNN 和 SPPnet 的比较，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. 多任务培训有帮助吗？

多任务培训很方便，因为它可以避免管理顺序训练的任务管道。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]。多任务训练是否可以提高快速 R-CNN 中的对象检测精度？

为了测试这个问题，我们训练了仅使用等式 Eq1.中的分类损失 L_{cls}的基线网络（即设定

	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）改进了分段训练的 mAP（每组第三列）。

λ = 0)。这些基线打印在表 6 中每组第一列中的模型 S，M 和 L。请注意，这些模型没有边界框回归量。接下来（每组第二列），我们采用经过多任务丢失训练的网络（公式 1，λ = 1），但我们在测试时禁用了边界框回归。这隔离了网络的分类准确性，并允许与基线网络进行一对一的比较。

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粹的分类准确性。改进范围从+0.8 到+1.1 mAP 点，显示了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），在边界框回归层上进行处理，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，但是阶段式训练不足以执行多任务训练（每组第四列）。

5.2. 尺度不变性：蛮力或技巧?

我们比较了实现尺度不变物体检测的两种策略：蛮力学习（单一尺度）和图像金字塔（多尺度）。在任何一种情况下，我们将图像的尺度 s 定义为其最短边的长度。

所有单尺度实验都使用 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，以便在微调期间 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;但是，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上采样 1.6 倍。因此，RoI 汇集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 }）中指定的相同五个尺度来促进与 SPPnet 的比较。但是，我们将最长边设为 2000 像素，以避免超出 GPU 内存。

表 7 显示了使用一个或五个刻度进行训练和测试时的模型 S 和 M。也许[11]中最令人惊讶的结果是单尺度检测几乎与多尺度检测一样好。我们的发现确认了他们的结果：深度 ConvNets 擅长直接学习规模不变性。多尺度方法在计算时间内以较高的成本提供了 mAP 的小幅增加（表 7）。在 VGG16（模型 L）的情况下，我们通过扩展细节实现单尺度是受到限制的。然而，它实现了 66.9% 的 mAP，略高于 R-CNN 报告的 66.0%[10]，尽管 R-CNN 使用“无限”尺度，因为每个提案都被扭曲成规范尺寸。

	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.多尺度与单一尺度。 SPPnet ZF（类似于模型 S）的结果来自[11]。 具有单一规模的较大网络提供最佳速度/准确度权衡。 （由于 GPU 内存限制，L 在我们的实现中不能使用多尺度。）

由于单一尺度处理在速度和精度之间提供了最佳折衷，特别是对于非常深的模型，本小节之外的所有实验都使用单尺度训练和 s = 600 像素的测试。

5.3. 我们需要更多的训练数据吗?

当提供更多训练数据时，良好的物体探测器应该得到改善。朱等人[24]发现 DPM [8] 的 mAP 仅在数百至数千个训练实例之后饱和。在这里，我们使用 VOC12 trainval set 加 VOC07 trainval set，将图像数量大约增加三倍至 16.5k，以评估 Fast R-CNN。扩大训练集使 VOC07 测试的 mAP 从 66.9%提高到 70.0%（表 1）。在对此数据集进行训练时，我们使用 60k 小批量迭代而不是 40k。

我们对 VOC10 和 2012 进行了类似的实验，为此我们从 VOC07 trainval，test 和 VOC12 trainval 的联合数据集构建了 21.5k 图像的数据集。在对该数据集进行训练时，我们使用 100k SGD 迭代，并且每 40k 次迭代（而不是每次 30k）将学习率降低 0.1。对于 VOC10 和 2012，mAP 分别从 66.1%提高到 68.8%，从 65.7%提高到 68.4%。

5.4. SVM 的表现是否优于 softmax?

快速 R-CNN 使用在微调期间学习的 softmax 分类器，而不是在事后才训练”一对二”静态线性 SVM，

如在 R-CNN 和 SPPnet 中所做的那样。 为了理解这种选择的影响，我们在 Fast R-CNN 中使用困难负例挖掘实现了事后 SVM 训练。 我们使用与 R-CNN 相同的训练算法和超参数。

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.具有 softmax 与 SVM 的快速 R-CNN（VOC07 mAP）。

表 8 显示，对于所有三个网络，softmax 略微优于 SVM，+0.1 至+0.8 mAP 点。这种效果很小，但它表明，与以前的多阶段训练方法相比， “一次性”微调就足够了。我们注意到 softmax 与”一对二“静态 SVM 不同，在对 RoI 进行评分时引入了类之间的竞争。

5.5. 更多提议总是更好吗？

存在（广泛地）两种类型的对象检测器：使用稀疏对象建议集（例如，选择性搜索[21]）和使用密集对象集（例如，DPM [8]）的对象检测器。对稀疏建议进行分类是一种级联[22]，其中提议机制首先拒绝大量候选区域离开分类器并用一小组进行评估。当应用于 DPM 检测时，这种级联提高了检测精度[21]。我们发现提案分类器级联也提高了快速 R-CNN 精度的证据。

使用选择性搜索的质量模式，我们每个图像扫描 1k 到 10k 个提议，每次重新训练和重新测试模型 M。如果提议服务于纯计算角色，增加每个图像的提议数量不应该损害 mAP。

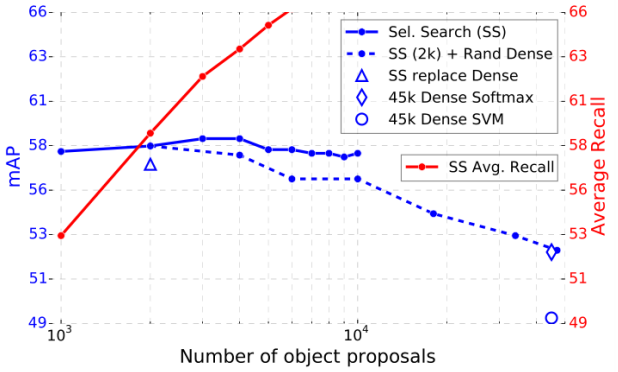

图 3.针对各种提案方案的 VOC07 测试 mAP 和 AR。

我们发现 mAP 随着前期计数的增加而上升然后略有下降（图 3，实线蓝线）。 这个实验表明，使用更多提案淹没深度分类器并没有帮助，甚至有点伤害准确性。

如果没有实际运行实验，这个结果很难预测。测量对象建议质量的最新技术是平均召回（AR）[12]。当使用每个图像的固定数量的提议时，AR 与使用 R-CNN 的若干提议方法的 mAP 很好地相关。图 3 示出了 AR（实线红线）与 mAP 不完全相关，因为每个图像的提议数量是变化的。 AR 必须小心使用;更多的 AR 由于更多的提案并不意味着 mAP 会增加。幸运的是，使用 M 型进行培训和测试的时间不到 2.5 小时。因此，快速 R-CNN 能够有效，直接地评估建议区域 mAP，这比代理度量更可取。

当使用密集生成的 box（超过比例，位置和纵横比）时，我们还研究了快速 R-CNN，速率约为 45k boxes/图像。这个密集的集合足够丰富，当每个选择性搜索框被其最接近（在 IoU）密集框中替换时，mAP 仅下降 1 个点（到 57.7%，图 3，蓝色三角形）。

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

我们还使用密集盒（45k /图像）训练和测试快速 R-CNN。此设置产生的 mAP 为 52.9%（蓝色菱形）。最后，我们检查是否需要具有困难负例挖掘的 SVM 来应对密集的盒子分布。 SVM 更糟糕：49.3%（蓝色圆圈）。

5.6. 初步的 MS COCO 结果

我们将快速 R-CNN（带有 VGG16）应用于 MS COCO 数据集[18]以建立初步基线。我们对 240k 迭代的 80k 图像训练集进行了训练，并使用评估服务器在“test-dev”集上进行了评估。 PASCAL 式 mAP 为 35.9%;新的 COCO 式 AP，平均超过 IoU 阈值，为 19.7%。

六. 结论

本文提出了快速 R-CNN，一种干净，快速的 R-CNN 和 SPPnet 更新。除了报告最先进的检测结果外，我们还提供了详细的实验，希望能够提供新的见解。特别值得注意的是，稀疏对象提议似乎可以提高检测器质量。这个问题在过去进行探测时成本太高（实时），但是对于快速 R-CNN 而言也很实用。当然，可能还有一些未被发现的技术可以让密集的盒子和稀疏的提议一样好。如果开发这样的方法，可以帮助进一步加速物体检测。

致谢。我感谢 Kaiming He，Larry Zitnick 和 Piotr Dollar' 的有益讨论和鼓励。

参考文献

[1] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Se-mantic 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 convo-lutional 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. Ra-manan. Object detection with discriminatively trained part based models. TPAMI, 2010. 3, 7, 8

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich fea-ture 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. Dollar,´ 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 clas-sification 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. Dollar,´ and C. L. Zit-nick. 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, Localiza-tion 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 detec-tion? 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