Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Abstract

本文提出了一种用于物体检测的快速区域卷积网络方法（Fast R-CNN）。Fast R-CNN 基于之前的工作，利用深度卷积网络对物体提案进行高效分类。与之前的工作相比，Fast R-CNN 采用了多项创新技术来提高训练和测试速度，同时提高检测准确率。Fast R-CNN 训练深度 VGG16 网络的速度比 R-CNN 快 9 倍，测试速度快 213 倍，并在 PASCAL VOC 2012 上实现了更高的 mAP。与 SPPnet 相比，Fast R-CNN 训练 VGG16 的速度快 3 倍，测试速度快 10 倍，而且更准确。Fast R-CNN 是用 Python 和 C++（使用 Caffe）实现的，在开源的 MIT 许可下可在 https: //github.com/rbgirshick/fast-rcnn 上获取。

1. Introduction

最近，深度 ConvNets [14, 16] 显著提高了图像分类 [14] 和物体检测 [9, 19] 的准确性。与图像分类相比，物体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（如 [9, 11, 19, 25]）在多阶段管道中训练模型，速度慢且不优雅。

检测需要对物体进行精确定位，这就带来了两个主要挑战，从而导致了检测的复杂性。首先，必须处理众多候选物体位置（通常称为 "建议"）。其次，这些候选位置只能提供粗略的定位，必须加以改进才能实现精确定位。解决这些问题的方法往往在速度、准确性或简便性方面大打折扣。

在本文中，我们简化了最先进的基于 ConvNet 的物体检测器 [9, 11] 的训练过程。我们提出了一种单阶段训练算法，它能共同学习对物体提案进行分类，并完善其空间位置。

这种方法训练超深度检测网络（VGG16 [20]）的速度比 R-CNN [9]快 9 倍，比 SPPnet [11]快 3 倍。在运行时，该检测网络处理图像的时间仅为 0.3 秒（不包括对象提议时间），同时在 PASCAL VOC 2012 [7] 中达到了最高准确率，mAP 为 66%（R-CNN 为 62%）。

1.1. R-CNN and SPPnet

基于区域的卷积网络方法（RCNN）[9] 通过使用深度卷积网络对物体提案进行分类，实现了出色的物体检测精度。不过，R-CNN 也有明显的缺点：

1. 训练是一个多阶段的过程。R-CNN 首先使用对数损失对 ConvNet 的对象建议进行微调。然后，它根据 ConvNet 特征拟合 SVM。这些 SVM 充当物体检测器，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. 训练耗费大量空间和时间。对于 SVM 和边界框回归器的训练，需要从每幅图像中的每个对象提案中提取特征并写入磁盘。如果使用 VGG16 等深度网络，以 VOC07 训练值集的 5k 幅图像为例，这一过程需要 2.5 个 GPU 日。这些特征需要数百 GB 的存储空间。

3. 物体检测速度较慢。测试时，要从每张测试图像中的每个对象提案中提取特征。使用 VGG16 检测每幅图像需要 47 秒（在 GPU 上）。

R-CNN 的速度很慢，因为它对每个对象的提议都要执行 ConvNet 前向传递，而不共享计算。有人提出了空间金字塔池网络（SPPnet）[11]，通过共享计算来加快 R-CNN 的速度。SPPnet 方法为整个输入图像计算一个卷积特征图，然后使用从共享特征图中提取的特征向量对每个对象建议进行分类。特征提取的方法是，将特征图中提案内的部分放大到固定大小的输出中（如 6×6）。多个输出大小被汇集，然后像空间金字塔汇集一样进行连接[15]。在测试时，SPPnet 可将 R-CNN 的速度提高 10 到 100 倍。由于提案特征提取速度更快，训练时间也缩短了 3 倍。

SPPnet 也有明显的缺点。与 R-CNN 一样，训练也是一个多阶段的过程，包括提取特征、使用对数损失微调网络、训练 SVM 以及最后拟合边界框回归因子。特征也会写入磁盘。但与 R-CNN 不同的是，[11] 中提出的微调算法无法更新空间金字塔池化之前的卷积层。不难理解，这一限制（固定卷积层）限制了深度网络的准确性。

1.2. 贡献

我们提出了一种新的训练算法，它可以解决 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 RCNN，因为它的训练和测试速度相对较快。快速 RCNN 方法有几个优点：

1.比 R-CNN、SPPnet 更高的检测质量 (mAP)

2.使用多任务损耗进行单级训练

3.训练可更新所有网络层

4.无需磁盘存储进行特征缓存

Fast R-CNN 由 Python 和 C++（Caffe [13]）编写，采用开源 MIT 许可，网址为 https://github.com/rbgirshick/ fast-rcnn。

2.快速 R-CNN 架构和训练

图 1 展示了快速 R-CNN 架构。快速 R-CNN 网络将整幅图像和一组对象建议作为输入。该网络首先用几个卷积（conv）层和最大池化层处理整个图像，生成一个卷积特征图。然后，对于每个物体提案，兴趣区域（RoI）池层会从特征图中提取一个固定长度的特征向量。每个特征向量都被送入一系列全连接（fc）层，最后分支到两个同级输出层：一个层对 K 个对象类别加上一个总括的 "背景 "类别进行软最大概率估计，另一个层对 K 个对象类别中的每个类别输出 4 个实值数字。每组 4 个数值编码 K 个类别中一个类别的细化边界框位置。

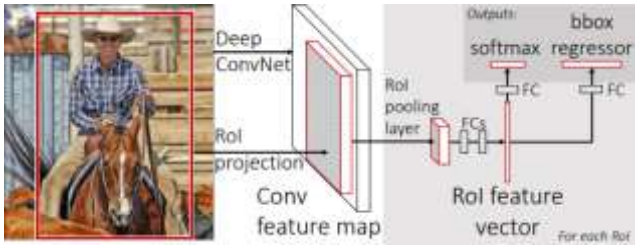

图 1.快速 R-CNN 架构。输入图像和多个感兴趣区域（RoIs）被输入一个全卷积网络。每个兴趣区域被汇集到一个固定大小的特征图中，然后通过全连接层（FC）映射到一个特征向量。该网络每个 RoI 有两个输出向量：软最大概率和每类边界框回归偏移。该架构采用多任务损失进行端到端训练。

2.1. The RoI pooling layer

RoI 池化层使用最大池化技术将任何有效感兴趣区域内的特征转换为一个空间范围固定为 H × W（如 7 × 7）的小特征图，其中 H 和 W 是与任何特定 RoI 无关的层超参数。在本文中，RoI 是进入 conv 特征图的一个矩形窗口。每个 RoI 由一个四元组（r,c,h,w）定义，该四元组指定了其左上角（r,c）及其高度和宽度（h,w）。

RoI 最大汇集法的工作原理是将 h × w RoI 窗口划分为 H × W 的子窗口网格，网格大小约为 h/H × w/W，然后将每个子窗口中的值最大汇集到相应的输出网格单元中。与标准最大汇集法一样，汇集法独立应用于每个特征图通道。RoI 层只是 SPPnets [11] 中使用的空间金字塔池化层的特例，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方法。

2.2. 从预训练网络初始化

我们使用三个预先训练好的 ImageNet [4] 网络进行实验，每个网络都有五个最大池化层和五到十三个 conv 层（网络详情见第 4.1 节）。当一个预训练网络初始化一个快速 R-CNN 网络时，它会经历三次转换。

首先，最后一个最大池化层被 RoI 池化层取代，RoI 池化层的 H 和 W 设置为与网络的第一个全连接层兼容（例如，VGG16 的 H = W = 7）。

其次，网络的最后一个全连接层和 softmax（针对 1000 路 ImageNet 分类进行了训练）被前面描述的两个同级层（一个全连接层和针对 K +1 类别的 softmax 以及特定类别的边界框回归因子）所取代。

第三，对网络进行修改，以接受两个数据输入：图像列表和这些图像中的 RoIs 列表。

2.3. Fine-tuning for detection

使用反向传播训练所有网络权重是快速 R-CNN 的一项重要功能。首先，我们来解释一下为什么 SPPnet 无法更新空间金字塔池层以下的权重。

根本原因在于，当每个训练样本（即 RoI）来自不同的图像时，通过 SPP 层进行反向传播的效率非常低，而这正是 R-CNN 和 SPPnet 网络的训练方式。效率低下的原因在于，每个 RoI 的感受野可能非常大，通常横跨整个输入图像。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）。

我们提出了一种更高效的训练方法，它能在训练过程中利用特征共享的优势。在快速 RCNN 训练中，随机梯度下降（SGD）小批量分层采样，首先采样 N 幅图像，然后从每幅图像中采样 R/N 个 RoI。重要的是，来自同一图像的 RoI 在前向和后向传递中共享计算和内存。N 越小，迷你批处理的计算量就越小。例如，当使用 N = 2 和 R = 128 时，建议的训练方案比从 128 幅不同图像中抽取一个 RoI（即 R-CNN 和 SPPnet 策略）快约 64 倍。

这种策略可能会导致训练收敛缓慢，因为来自同一图像的 RoIs 是相关的。但这一问题似乎并不实际，在 N = 2 和 R = 128 的情况下，我们使用比 R-CNN 更少的 SGD 迭代次数取得了良好的效果。

除了分层抽样外，快速 R-CNN 还采用了简化的训练流程，在一个微调阶段联合优化软最大分类器和边界框回归器，而不是分三个阶段分别训练软最大分类器、SVM 和回归器 [9，11]。该程序的各个组成部分（损失、迷你批量采样策略、通过 RoI 池层的反向传播以及 SGD 超参数）如下所述。

多任务损失。快速 R-CNN 网络有两个同级输出层。第一层输出 K + 1 个类别的离散概率分布（每个 RoI），p = (p0,...,pK)。按照惯例，p 是通过对完全连接层的 K+1 个输出进行软最大化计算得出的。第二个同级层输出边界框回归偏移量，$t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$，我们使用 [9] 中给出的 tk 参数，其中 tk 指定了相对于对象提案的尺度不变平移和对数空间高度/宽度移动。

每个训练 RoI 都标有一个地面实况类别 u 和一个地面实况边界框回归目标 v。我们在每个标注的 RoI 上使用多任务损失 L 来联合训练分类和边界框回归：

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

in which $L_{\text{cls}}(p,u) = -\log p_u$ is log loss for true class u.

第二个任务损失 Lloc 是根据类别 u 的真实边界框回归目标元组定义的。, $v =$ (v_x,v_y,v_w,v_h), and a 预测 tuple $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$, 再次对类 u 进行计算。当 u ≥ 1 时，艾弗森括号指示函数 [u ≥ 1] 的值为 1，否则为 0。对于背景 RoIs，不存在地面真实边界框的概念，因此 Lloc 将被忽略。对于边界框回归，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：

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

是一种稳健的 L1 损失，与 R-CNN 和 SPPnet 中使用的 L2 损失相比，它对异常值不那么敏感。当回归目标不受限制时，使用 L2 损失进行训练可能需要仔细调整学习率，以防止梯度爆炸。公式 3 则消除了这种敏感性。

公式 1 中的超参数 λ 控制着两个任务损失之间的平衡。我们对地面实况回归目标 vi 进行归一化处理，使其具有零均值和单位方差。所有实验均使用 λ = 1。

我们注意到，[6] 使用相关的损失来训练一个不分类的对象建议网络。与我们的方法不同，[6] 主张采用双网络系统，将定位和分类分开。OverFeat [19]、R-CNN [9] 和 SPPnet [11] 也训练分类器和边界框定位器，但这些方法都使用阶段性训练，而我们的研究表明，这对于快速 R-CNN 来说是次优的（第 5.1 节）。

迷你批次采样。在微调过程中，每个 SGD 迷你批次由 N = 2 张图像构建而成，这些图像都是随机均匀选择的（按照通常的做法，我们实际上是对数据集的排列进行迭代）。我们使用大小为 R = 128 的迷你批，从每幅图像中抽取 64 个 RoI。与文献[9]一样，我们从与地面实况边界框重叠度至少为 0.5 的对象提案中提取 25% 的 RoIs。这些 RoIs 包括标有前景对象类别（即 u ≥ 1）的示例。其余的 RoIs 是按照 [11] 的方法，从与地面实况的最大 IoU 在区间 [0.1,0.5) 内的对象建议中抽取的。这些是背景示例，标记为 u = 0。0.1 这个较低的阈值似乎可以作为硬示例挖掘的启发式方法 [8]。在训练过程中，图像水平翻转的概率为 0.5。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 池层传递导数。为清晰起见，我们假设每个迷你批次只有一个图像（N = 1），但由于前向传递会独立处理所有图像，因此扩展到 N > 1 也很简单。

设 xi∈R 为 RoI 池化层的第 i 个激活输入，设 yrj 为该层来自第 r 个 RoI 的第 j 个输出。RoI 池化层计算 yrj = xi∗(r,j)，其中 i∗(r,j) = argmaxi0∈R(r,j) xi0。R(r,j) 是输出单元 yrj 最大池化的子窗口中输入的索引集。一个 xi 可以分配给多个不同的输出 yrj。

RoI 池层的后向函数通过 argmax 开关计算损失函数相对于每个输入变量 xi 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 yrj，如果 i 是通过最大池化为 yrj 选择的 argmax，则部分导数 ∂L/∂yrj 将被累积。在反向传播过程中，部分导数 ∂L/∂yrj 已由 RoI 池化层顶层的反向函数计算得出。

SGD 超参数。用于软最大分类和边界框回归的全连接层由零均值高斯分布初始化，标准偏差分别为 0.01 和 0.001。所有层的权重和偏置的每层学习率分别为 1 和 2，全局学习率为 0.001。在 VOC07 或 VOC12 trainval 上训练时，我们先运行 SGD 进行 30k 次迷你批次迭代，然后将学习率降至 0.0001，再进行 10k 次迭代训练。当我们在更大的数据集上进行训练时，我们会运行 SGD 进行更多的迭代，详见下文。我们使用 0.9 的动量和 0.0005 的参数衰减（权重和偏置）。

2.4. 规模不变性

我们探索了两种实现物体尺度不变检测的方法：(1) 通过 "蛮力 "学习；(2) 使用图像金字塔。这些策略沿用了 [11] 中的两种方法。在 "蛮力 "方法中，在训练和测试过程中，每幅图像都按照预先确定的像素尺寸进行处理。网络必须直接从训练数据中学习比例不变的物体检测。

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。测试时，图

像金字塔用于对每个对象建议进行近似比例归一化。在多尺度训练过程中，我们按照文献[11]的方法，每次对图像进行采样时，都会随机采样一个金字塔尺度，作为一种数据增强形式。由于 GPU 内存限制，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 网络进行了微调，检测只需运行前向传递即可（假定对象提议已预先计算）。该网络的输入是一幅图像（或图像金字塔，编码为图像列表）和需要评分的 R 个对象建议列表。在测试时，R 通常在 2000 左右，不过我们也会考虑 R 更大（≈ 45k）的情况。在使用图像金字塔时，每个 RoI 都会被按比例分配，这样按比例分配的 RoI 在面积上最接近 224 像素[11]。

对于每个测试 RoI r，前向传递输出一个类别后验概率分布 p 和一组相对于 r 的预测边界框偏移（K 个类别中的每个类别都有自己的细化边界框预测）。我们使用估计概率 Pr(class = k | r) =Δ pk 为每个对象类别 k 的 r 分配检测置信度。然后，我们使用 R-CNN [9] 的算法和设置，对每个类别独立执行非最大抑制。

3.1. 截断 SVD，加快检测速度

对于全图像分类，计算全连接层所需的时间比计算 conv 层所需的时间要少。相反，对于检测，需要处理的 RoIs 数量很大，前向传递时间的近一半用于计算全连接层（见图 2）。使用截断 SVD 压缩大型全连接层可以轻松加快计算速度[5, 23]。

在这种技术中，以 u × v 权重矩阵 W 为参数的层近似因式分解为

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

使用 SVD。截断 SVD 将参数数从 uv 减少到 t(u+v)，如果 t 远远小于 min(u,v)，参数数会非常可观。为了压缩网络，与 W 相对应的单个全连接层被两个全连接层取代，它们之间不存在非线性关系。其中第一层使用权重矩阵 ΣtV T（无偏置），第二层使用 U（与 W 相关的原始偏置）。当 RoIs 数量较多时，这种简单的压缩方法可以很好地提高速度。

4. 主要成果

三项主要成果支持了本文的贡献：

1. VOC07、2010 和 2012 上最先进的 mAP
2. 与 R-CNN、SPPnet 相比，训练和测试速度更快
3. VGG16 中对 conv 层的微调改善了 mAP

4.1.实验设置

第一个模型是 R-CNN [9] 的 CaffeNet（本质上是 AlexNet [14]）。我们也将此 CaffeNet 称为模型 S，意为 "小型"。第二个网络是来自 [3] 的 VGG CNN M 1024，其深度与 S 相同，但更宽。我们称该网络为模型 M，表示 "中等"。最后一个网络是来自 [20] 的超深度 VGG16 模型。在本节中，所有实验都使用单尺度训练和测试（s = 600；详见第 5.2 节）。

4.2. VOC 2010 and 2012 结果

在这些数据集上，我们将快速 R-CNN（简称 FRCN）与公共排行榜中 comp4（外部数据）赛道的顶级方法进行了比较（表 2、表 3）。 对于 NUS NIN c2000 和 BabyLearning 方法，目前还没有相关的出版物，我们也无法找到所用 ConvNet 架构的确切信息；它们是 Network-in-Network 设计的变体[17]。所有其他方法都是从相同的预训练 VGG16 网络初始化的。

快速 R-CNN 在 VOC12 上取得了最佳结果，mAP 为 65.7%（使用额外数据时为 68.4%）。与其他基于 "慢速 "R-CNN 管道的方法相比，它的速度也快了两个数量级。在 VOC10 上，SegDeepM [25] 的 mAP 比快速 R-

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 检测平均精度（%）。所有方法均使用 VGG16。训练集关键字：07：VOC07 trainval，07 \ diff：07：没有 "困难 "示例的 07，07+12：07 和 VOC12 trainval 的结合。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.VOC 2010 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。训练集关键字：12：VOC12 训练值；Prop.：专有数据集；12+seg：12 个带分割注释的数据集；07++12：VOC07 训练值、VOC07 测试值和 VOC12 训练值的结合。

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 测试的平均检测精度（%）。BabyLearning 和 NUS NIN c2000 使用基于 [17] 的网络。其他方法均使用 VGG16。训练集关键字：见表 2，Unk：unknown

dCNN 高（67.2% 对 66.1%）。SegDeepM 是在 VOC12 trainval 和分割注释的基础上进行训练的；它旨在通过使用马尔可夫随机场对 R-CNN 检测和来自 O2P [1] 语义分割方法的分割进行推理，从而提高 R-CNN 的准确性。快速 R-CNN 可以换成 SegDeepM 来代替 R-CNN，这可能会带来更好的结果。在使用扩大的 07++12 训练集时（见表 2 标题），Fast R-CNN 的 mAP 增至 68.8%，超过了 SegDeepM。

4.3. VOC 2007 results

在 VOC07 上，我们将快速 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 结果由 [11] 的作者计算得出。SPPnet 在训练和测试过程中使用了五个标度。快速 R-CNN 对 SPPnet 的改进表明，即使快速 R-CNN 使用单尺度训练和测试，对 conv 层进行微调也能大幅提高 mAP（从 63.1%提高到 66.9%）。R-CNN 的 mAP 为 66.0%。还有一点需要说明的是，SPPnet 在训练时没有使用 PASCAL 中标记为 "困难 "的示例。 去掉这些示例后，快速 R-CNN 的 mAP 提高到了 68.1%。所有其他实验都使用了 "困难 "示例。

4.4. Training and testing time

训练和测试时间短是我们的第二个主要成果。表 4 比较了快速 RCNN、R-CNN 和 SPPnet 在 VOC07 上的训练时间（小时）、测试速度（每幅图像秒数）和 mAP。对于 VGG16，快速 R-CNN 处理图像的速度比不使用截断 SVD 的 R-CNN 快 146 倍，比使用截断 SVD 的 R-CNN 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 RCNN 训练 VGG16 的速度提高了 2.7 倍（9.5 小时对 25.5 小时），在不使用截断 SVD 的情况下测试速度提高了 7 倍，

在使用截断 SVD 的情况下测试速度提高了 10 倍。由于不缓存特征，Fast R-CNN 还节省了数百 GB 的磁盘存储空间。

	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	-	-	-	-

能够 4.相同模型在 Fast R CNN、R-CNN 和 SPPnet 中的运行时间比较。快速 R-CNN 使用单尺度模式。SPPnet 使用 [11] 中指定的五个尺度。时间由 [11] 作者提供。时间在 Nvidia K40 GPU 上测量。

截断 SVD。截断 SVD 能将检测时间缩短 30% 以上，而 mAP 只下降很小（0.3 个百分点），并且在模型压缩后无需执行额外的微调。图 2 展示了在 VGG16 的 fc6 层中使用 25088×4096 矩阵的前 1024 个奇异值和在 4096×4096 fc7 层中使用前 256 个奇异值如何缩短运行时间，而 mAP 损失很小。如果在压缩后再次进行微调，还可能进一步提高速度，但 mAP 的下降幅度会更小。

4.5. Which layers to fine-tune?

对于 SPPnet 论文[11]中考虑的深度较低的网络，仅对全连接层进行微调似乎就足以获得良好的精度。我们假设这一结果对于深度网络来说并不成立。为了验证微调卷积层对 VGG16 的重要性，我们使用快速 R-CNN

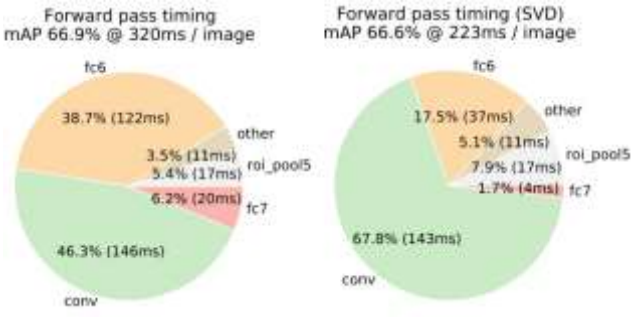

图 2.截断 SVD 前后 VGG16 的时序。在 SVD 之前，全连接层 fc6 和 fc7 占用 45% 的时间。

进行微调，但冻结了 13 个卷积层，因此只有全连接层在学习。这种消减模拟了单规模 SPPnet 训练，将 mAP 从 66.9% 降至 61.4%（表 5）。这一实验验证了我们的假设：通过 RoI 池层进行训练对于深度网络非常重要。

	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 VGG16 微调层数的效果。微调≥ fc6 模拟了 SPPnet 训练算法[11]，但使用的是单一尺度。SPPnet L 的结果是使用五个尺度获得的，但在速度上付出了巨大代价（7 倍）。

这是否意味着所有信念层都应进行微调？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，与任务无关（这是众所周知的事实 [14]）。无论是否允许 conv1 学习，都不会对 mAP 产生有意义的影响。对于 VGG16，我们发现只需更新 conv3 1 及以上的层即可（13 个 conv 层中的 9 个）。这一观察结果很实用：(1) 与从 conv3 1 开始学习相比，从 conv2 1 开始更新会使训练速度减慢 1.3 倍（12.5 小时对 9.5 小时）；(2) 从 conv1 1 开始更新会使 GPU 内存超载。从 conv2 1 开始学习时，mAP 的差异仅为 +0.3（表 5，最后一列）。本文所有使用 VGG16 的快速 R-CNN 结果都对 conv3 1 及以上层进行了微调；所有使用 S 和 M 模型的实验都对 conv2 及以上层进行了微调。

5. Design evaluation

我们进行了实验，以了解快速 RCNN 与 R-CNN 和 SPPnet 的比较情况，并对设计决策进行评估。按照最佳实践，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流水线。但它也有可能改善结果，因为各任务通过共享表征（ConvNet）

	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）比零散训练（每组第三列）提高了 mAP。

相互影响[2]。多任务训练是否能提高快速 R-CNN 的物体检测精度？

为了检验这个问题，我们训练了只使用公式 1 中的分类损失 Lcls 的基线网络（即设置 $\lambda = 0$）。表 6 中每组第一列分别列出了模型 S、M 和 L 的基线。请注意，这些模型没有边框回归因子。接下来（每组第二列），我们选取使用多任务损失（公式 1，$\lambda = 1$）训练的网络，但在测试时禁用边界框回归。这样就将网络的分类准确性分离出来，并与基线网络进行对比。

在所有三个网络中，我们观察到多任务训练相对于单纯的分类训练提高了纯分类准确率。提高幅度从 +0.8 到 +1.1 mAP 点不等，表明多任务学习具有持续的积极效果。

最后，我们采用基线模型（仅使用分类损失进行训练），附加边界框回归层，并使用 Lloc 对其进行训练，同时冻结所有其他网络参数。每组的第三列显示了这种分阶段训练方案的结果：mAP 比第一列有所改进，但分阶段训练的效果不如多任务训练（每组第四列）。

5.2. 规模不变性：蛮干还是巧干？

我们比较了实现尺度不变物体检测的两种策略：暴力学习（单尺度）和图像金字塔（多尺度）。无论哪种方法，我们都将图像的尺度 s 定义为其最短边的长度。

所有单比例实验都使用 s = 600 像素；某些图像的 s 可能小于 600，因为我们将图像最长边的上限设定为 1000 像素，并保持图像的纵横比。选择这些值是为了让 VGG16 在微调过程中适合 GPU 内存。较小的模型不受内存限制，可以从较大的 s 值中获益；不过，为每个模型优化 s 值并不是我们的主要关注点。我们注意到 PASCAL 图像的平均像素为 384 × 473，因此单比例设置通常会将图像的采样率提高 1.6 倍。因此，RoI 汇集层的平均有效跨距≈ 10 像素。

	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.多尺度与单尺度。SPPnet ZF（类似于模型 S）的结果来自 [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，在我们的实现中无法使用多尺度)。

在多尺度设置中，我们使用了 [11] 中指定的相同五个尺度（$s \in \{480,576,688,864,1200\}$），以方便与 SPPnet 进行比较。不过，为了避免超出 GPU 内存，我们将最长边的上限设定为 2000 像素。

表 7 显示了使用一个或五个尺度进行训练和测试时的模型 S 和 M。也许 [11] 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 ConvNets 擅长直接学习尺度不变性。多尺度方法只提高了少量的 mAP，却耗费了大量的计算时间（表 7）。就 VGG16（模型 L）而言，由于实现细节的限制，我们只能使用单一尺度。然而，尽管 R-CNN 使用的是 "无限 "尺度，即每个提案都被扭曲为一个典型大小，但它仍实现了 66.9% 的 mAP，略高于 R-CNN [10]报告的 66.0%。

由于单尺度处理可在速度和准确性之间实现最佳平衡，特别是对于深度模型而言，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. 我们是否需要更多的训练数据?

一个好的物体检测器应该在获得更多训练数据后有所改进。Zhu 等人[24]发现，

DPM[8] mAP 只需几百到几千个训练实例就会达到饱和。在这里，我们用 VOC12 的训练集来增强 VOC07 的训练集，大约将图像数量增加了两倍，达到 16.5k，以评估快速 R-CNN 的效果。扩大训练集后，VOC07 测试的 mAP 从 66.9% 提高到 70.0%（表 1）。在该数据集上进行训练时，我们使用了 60k 迷你批次迭代，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 的组合中构建了一个包含 21.5k 张图像的数据集。在该数据集上进行训练时，我们使用 100k SGD 迭代，每 40k 次迭代（而不是每 30k 次）降低 0.1 倍学习率。对于 VOC10 和 2012，mAP 分别从 66.1% 提高到 68.8% 和从 65.7% 提高到 68.4%。

5.4. SVM 是否优于 softmax?

Fast R-CNN 使用在微调过程中学习到的 softmax 分类器，而不是像在 R-CNN 和 SPPnet 中那样，在事后训练 one-vs-rest 线性 SVM。为了解这一选择的影响，我们在 Fast R-CNN 中使用硬负挖掘实现了事后 SVM 训练。我们使用了与 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.带有 softmax 的快速 R-CNN 与 SVM 的对比（VOC07 mAP）。

表 8 显示，在所有三个网络中，softmax 的性能略优于 SVM，差距在 +0.1 到 +0.8 mAP 点之间。这种影响很小，但它表明与以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 "一比一 "SVM 不同，在对 RoI 进行评分时会引入类之间的竞争。

5.5.提案越多越好吗？

物体检测器（大致）有两种类型：一种是使用稀疏的物体提议集（如选择性搜索 [21]），另一种是使用密集的提议集（如 DPM [8]）。对稀疏提议进行分类是一种级联[22]，其中提议机制首先会拒绝大量的候选提议，只留下一小部分提议供分类器评估。这种级联方法应用于 DPM 检测时可提高检测精度 [21]。我们发现有证据表明，建议分类器级联也能提高快速 R-CNN 的准确性。

利用选择性搜索的质量模式，我们对每幅图像进行了 1k 到 10k 次搜索，每次都对模型 M 进行了重新训练和测试。

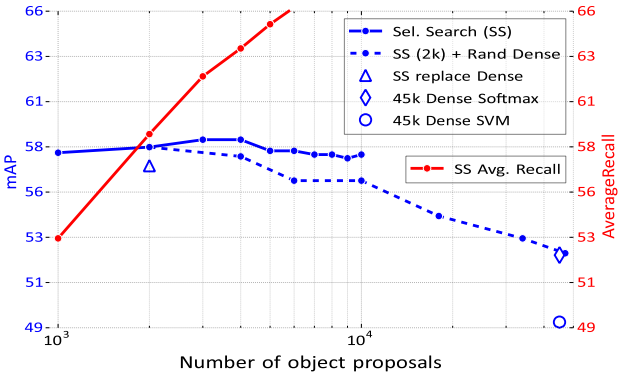

图 3.各种建议方案的 VOC07 测试 mAP 和 AR。

我们发现，随着建议数的增加，mAP 先是上升，然后略有下降（图 3，蓝色实线）。这一实验表明，用更多的建议淹没深度分类器无助于提高准确率，甚至会略微降低准确率。

在没有实际操作实验的情况下，很难预测这一结果。衡量物体建议质量的最先进方法是平均召回率 (AR) [12]。对于使用 R-CNN 的几种建议方法，当每幅图像使用固定数量的建议时，AR 与 mAP 的相关性很好。图 3 显示，当每幅图像的建议数变化时，AR（红色实线）与 mAP 的相关性并不好。必须谨慎使用 AR；提议数越多，AR 越高，并不意味着 mAP 也会增加。幸运的是，模型 M 的训练和测试时间不到 2.5 小时。因此，快速 R-

CNN 可以高效、直接地评估对象提议的 mAP，这比替代指标更为可取。

我们还研究了快速 R-CNN 在使用密集生成的方框时的效果（在比例、位置和长宽比上），每幅图像大约有 45k 个方框。这个密集集足够丰富，当每个选择性搜索框被最接近（在 IoU 中）的密集框替换时，mAP 只下降了 1 个点（降至 57.7%，图 3，蓝色三角形）。

密集方框的统计数据与选择性搜索方框的统计数据不同。从 2k 个选择性搜索框开始，我们测试了随机添加 1000 × {2,4,6,8,10,32,45} 个密集搜索框时的 mAP。在每次实验中，我们都会重新训练和测试模型 M。当添加这些密集搜索框时，mAP 的下降幅度比添加更多选择性搜索框时更大，最终达到 53.0%。

我们还只使用密集盒（45k/图像）对快速 R-CNN 进行了训练和测试。在这种情况下，mAP 为 52.9%（蓝钻）。最后，我们检查是否需要使用硬负挖掘的 SVM 来处理密集方框分布。SVM 的结果更糟：49.3%（蓝色圆圈）。

5.6. MS COCO 初步结果

我们在 MS COCO 数据集[18]中应用了快速 R-CNN（含 VGG16），以建立初步基准。我们在 8 万张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 样式的 mAP 为 35.9%；新 COCO 样式的 AP（也是 IoU 阈值的平均值）为 19.7%。

6. 结论

本文提出了快速 R-CNN，它是对 R-CNN 和 SPPnet 的简洁而快速的更新。除了报告最新的检测结果外，我们还进行了详细的实验，希望能提供新的见解。特别值得注意的是，稀疏对象建议似乎提高了检测器的质量。这个问题在过去的探测中代价太高（时间上），但在快速 R-CNN 中变得切实可行。当然，也可能存在尚未被发现的技术，能让密集方框的性能与稀疏提案一样好。如果开发出这种方法，将有助于进一步加快物体检测速度。

致谢。感谢何开明、Larry Zitnick 和 Piotr Dollar 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. 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. FeiFei. 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. Regionbased 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. 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, 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, 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