

订阅DeepL Pro以编辑此文档。  
访问www.DeepL.com/Pro，了解更多信息。

快速R-CNN

Ross Girshick 微软研究部

[rbg@microsoft.com](mailto:rbg@microsoft.com)

# 摘要

*本文提出了一种基于快速区域的卷积网络方法*(Fast R-CNN)*用于对象检测。Fast R-CNN建立在以前的工作基础上，利用深度卷积网络对ob-ject提案进行有效分类。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训练极深的VGG16网络比R-CNN快9，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训练VGG16快了3，测试时快了10faster，而且更准确。Fast R-CNN是用Python和C++实现的（使用Caffe），并以开源的MIT许可提供，网址为*[*https。*](https://github.com/rbgirshick/fast-rcnn)

*x*

*x*

*xx*

[*//github.com/rbgirshick/fast-rcnn*](https://github.com/rbgirshick/fast-rcnn)*。*

# 介紹

最近，深度ConvNets[[14](#_bookmark40)，[16](#_bookmark42)]已经显著地证明了图像分类[[14](#_bookmark40)]和对象检测[[9](#_bookmark35)，[19](#_bookmark45)]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，需要更多复杂的方法来解决。由于这种复杂性，目前的方法(*如*[[9](#_bookmark35), [11](#_bookmark37), [19](#_bookmark45), [25](#_bookmark51)])在多阶段的流水线中训练模型，速度慢且不流畅。

复杂性的产生是因为检测需要对物体进行精确的定位，这就产生了两个主要的挑战：首先，必须处理众多的候选物体位置（通常称为"建议"）。首先，必须处理许多候选对象位置（通常称为"建议"）。第二，这些提案只能提供粗略的定位，必须重新确定才能实现精确的定位。这些问题的解决方案往往会影响速度、精度或简单性。

在本文中，我们简化了基于ConvNet的最新物体检测器的训练过程[[9](#_bookmark35)，[11](#_bookmark37)]。我们提出了一个单阶段的训练算法，该算法可以共同学习对对象提案进行分类，并重新定义其空间位置。

由此产生的方法可以训练一个很深的检测网络（VGG16[[20](#_bookmark46)]），比R-CNN[[9](#_bookmark35)]快9*倍*，比R-CNN快3*倍。*

同时在PASCAL VOC 2012[[7](#_bookmark33)]上取得了最高的准确率，mAP为66%（而R-CNN为62%）。[1](#_bookmark1)

## R-CNN和SPPnet

基于区域的卷积网络方法(R- CNN)[[9](#_bookmark35)]通过使用深度ConvNet对对象提案进行分类，达到了很好的对象检测精度。但是，R- CNN有明显的缺点。

1. **训练是一个多阶段的管道。**R-CNN首先利用对数损失对对象提案进行ConvNet调整。然后，它对ConvNet特征进行SVMs筛选。这些SVMs作为对象检测器，取代软最大分类器学习的ﬁne- tuning。在第三个训练阶段，边界盒回归器被学习。
2. **训练在空间和时间上都很昂贵。**对于SVM和bounding-box regressor训练，从每张图像中的每个对象提案中提取特征并写入磁盘。对于非常深的网络，比如VGG16，这个过程对于VOC07训练值集的5k张图像需要2.5个GPU-天。这些特征重新需要数百GB的存储。
3. **对象检测速度很慢。**在测试时，从每张测试图像中的每个对象提案中提取特征。用VGG16检测需要47s/图像（在GPU上）。

R-CNN的速度很慢，因为它为每个对象提案执行ConvNet前传，而不共享计算。空间金字塔池网络(SPPnets)[[11](#_bookmark37)]被提出来通过共享计算来加快R-CNN的速度。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中提取的特征向量对每个对象提案进行分类。通过将提案内的特征图部分最大限度地汇集到一个五倍大小的输出(*例如*，6 6)中来提取提案的特征。多个输出尺寸被池化，然后像空间金字塔池化一样进行连接[[15](#_bookmark41)]。SPPnet在测试时将R-CNN加速了10～100。由于更快的pro- posal特征提取，训练时间也减少了3。

*x*

*x*

*x*

比SPPnet快[[11](#_bookmark37)]。在运行时，检测网络

处理图像的时间为0.3秒（不包括对象提出时间

1 所有计时都使用一个Nvidia K40 GPU超频至875 MHz。

SPPnet也有明显的缺点。像R-CNN一样，训练- ing是一个多阶段的管道，包括提取fea- tures，用对数损失来调整网络，训练SVM，以及最终确定bounding-box回归器。特征也被写入磁盘。但与R-CNN不同的是，[[11](#_bookmark37)]中提出的ﬁne-tuning al- gorithm不能更新空间金字塔池化之前的卷积层。不足为奇的是，这个限制（卷积层的更新）限制了很深网络的精度。

## 缴款

我们提出了一种新的训练算法，在提高R-CNN和SPPnet的速度和精度的同时，弥补了R-CNN的不足。我们称这种方法为*快速R-CNN*，因为它的训练和测试速度比较快。快速R-CNN方法有几个优点。

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

Fast R-CNN是用Python和C++编写的(Caffe[[13](#_bookmark39)])，在开源的MIT Li- cense下，可在[https://github.com/rbgirshick/ fast-rcnn](https://github.com/rbgirshick/fast-rcnn)。

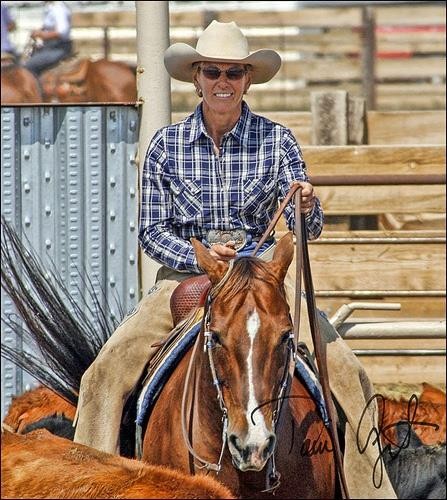
# 快速R-CNN架构和训练

图[1](#_bookmark3)展示了快速R-CNN架构。一个快速R-CNN网络将整个图像和一组对象提案作为输入。该网络首先用几个卷积（*convolutional，conv*）和最大池化层处理整个图像，以产生一个conv特征图。然后，对于每个对象提案，一个感兴趣的区域（*RoI*）池化层从特征图中提取一个长度为五倍的特征向量。每个特征向量都被输入到一个完全连接的（*fc*）层序列中，最终分支到两个兄弟输出层：一个产生*K个*对象类的软最大概率估计加上一个全面的"背景"类，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*类中的一个类编码重新定义的边界框位置。

## RoI池层

RoI池化层使用最大池化将任何有效的感兴趣区域内的特征转换为一个小的fea- ture图，其空间范围为*H W*(*例如*，7 7)，其中*H*和*W*是层超参数，不受任何特定RoI的影响。在本文中，一个 RoI 是一个进入 conv 特征图的矩形窗口。每个RoI由一个四元组(*r, c, h, w*)定义，该四元组确定了它的左上角(*r, c*)及其高度和宽度(*h, w*)。

*xx*



*产出：*

箱子

深度

ConvNet

软最大值回归器

RoI

汇集层

FC

FC

RoI

投影

对照特征图

FCs

RoI特征

向量 *对于每个RoI*

图1.快速R-CNN架构。快速R-CNN架构。输入图像和多个感兴趣区域（RoIs）被输入到一个完全卷积网络中。每个RoI被汇集到一个五倍大小的特征图中，然后通过全连接层（FC）映射到特征向量。该网络每个RoI有两个输出向量：软最大概率和每个类的边界框回归偏移。该架构是在多任务损失的情况下进行端到端训练的。

RoI最大池化的工作原理是将*h w* RoI win- dow划分为一个*H W*网格，该网格由大小近似*h/H w/W*的子窗口组成，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。Pool- ing是独立应用于每个特征图通道的，就像标准的最大池化一样。RoI层只是SPPnets[[11](#_bookmark37)]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](#_bookmark37)]中给出的池化子窗口计算。

*x*

*x*

*x*

## 从预先训练好的网络进行初始化

我们用三个预训练的ImageNet[[4](#_bookmark30)]网络进行实验，每个网络都有五个最大的池化层，并且有五个到十三个conv层(网络去尾见[4.1](#_bookmark8)节)。当一个预训练网络初始化一个快速R-CNN网络时，它经历了三个转变。

首先，最后一个最大池化层被一个RoI池化层所取代，该层通过设置*H*和*W*与网的第一个完全连接层兼容（*例如*，对于VGG16，*H*=*W*=7）。

其次，网络最后的全连接层和软?

max（是为1000-way ImageNet分类训练的）被替换为前面描述的两个兄弟层（一个完全连接的层和softmax在*K* + 1猫- egories和类别-specificc bounding-box回归器上）。

第三，该网络被修改为接受两个数据输入：一个图像列表和这些图像中的RoI列表。

## 检测的微调

用反向传播训练所有网络权重是快速R-CNN的重要能力。首先，我们来阐明一下为什么SPPnet无法更新空间金字塔池层以下的权重。

其根本原因是，当每个训练样本（*即*RoI）来自不同的图像时，通过SPP层的反向传播是非常不有效的，这正是R-CNN和SPPnet网络的训练方式。无效性

源于这样一个事实，即每个RoI可能有一个非常大的接受ﬁeld，通常跨越整个输入图像。由于前向通道必须处理整个接收滤波器，所以训练输入量很大（通常是整个图像）。

边界盒，因此忽略了Lloc。对于边界框回归，我们使用损失的

Lloc(*tu, v*) = *smoothL* (*tu - vi*)*,*(2)

我们提出了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，随机梯度下降（SGD）mini-。

其中

← *i*

*ie{*x*,*y*,*w*,*h*}。*

批次分层采样，首先通过采样*N个图像*年龄，然后从每个图像中采样*R/N个*RoI。关键的是，来自同一图像的 RoIs 共享计算。

*smoothL*← (*x*) =

0*.*5x2 如果*IxI < 1*

*IxI -* 0*.*5否则*。*

,

(3)

和内存的前向和后向传递。将*N*做小，可以减少迷你批计算。例如，当使用*N* = 2和*R* = 128时，所提出的训练方案64 比从128张不同的图像中64 采样一个RoI要快得多（*即*R-CNN和SPPnet策略）。对这种策略的一个担心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cor-相关的。这种担心似乎不是一个实际问题，我们在*N* = 2和*R* = 128的情况下取得了良好的结果。

*x*

与R-CNN相比，使用更少的SGD迭代次数。

除了分层采样之外，Fast R-CNN还使用了一个精简的训练过程，其中一个调整阶段联合优化了softmax分类器和bounding-box re-gressors，而不是在三个单独的阶段训练softmax分类器、SVM和回归器[[9](#_bookmark35), [11](#_bookmark37)]。这个过程的组成部分(损失、迷你批量采样strat- egy、通过RoI池化层的反向传播和SGD超参数)描述如下。

**多任务损失。**一个快速R-CNN网络有两个同级输出层。第五个输出层在*K* + 1个类别上输出一个离散的概率区分（每个RoI），*p* = (p0*, . . . , pK*)。像往常一样，*p*由一个完全连接层的*K*+1个输出上的softmax计算。第二个同级层输出边界框回归偏移，*tk* = *tk, tk, tk, tk , tk* , 对于

╱、

x

y

w

h

是一种稳健的L1损失，与R-CNN和SPPnet中使用的L2损失相比，它对异常值的敏感性较低。当回归目标是无边界的时候，使用L2损失进行训练可能需要仔细调整学习率，以防止爆炸性的梯度。Eq.[3](#_bookmark5)消除了这种敏感性。

Eq.[1](#_bookmark6)中的超参数*λ*控制了两种任务损失之间的平衡。我们将地真回归目标*vi*归一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](#_bookmark32)]使用相关的损失来训练一个类-。

agnostic object proposal network.与我们的方法不同，[[6](#_bookmark32)]主张采用双网络系统，将定位和分类分开。OverFeat[[19](#_bookmark45)]、R-CNN[[9](#_bookmark35)]和SPPnet[[11](#_bookmark37)]也训练分类器和边界盒定位器，然而这些方法使用阶段性训练，我们表明这对快速R-CNN来说是次优的([5.1](#_bookmark18)节)。

**迷你批次采样。**在ﬁne-tuning期间，每个SGD迷你批次是构建从*N* = 2图像，选择uni- formly随机（作为常见的做法，我们实际上iter- ate在数据集的permutations）。我们使用大小为*R* = 128的迷你批次，从每个图像中采样64个RoIs。在[[9](#_bookmark35)]中，我们从对象提案中抽取 25%的 RoI，这些对象提案与地面上的对象有交集，而不是联合（IoU）重叠。

真值边界盒至少为 0*.*5。这些 RoIs 包括

我们使用[[9](#_bookmark35)]中给出的*tk*的参数化，其中*tk*确定了相对于对象提案的标度不变的平移和对数空间的高/宽移动。

每一个训练RoI都被标记了一个地真类*u*和一个地真边界盒回归目标*v*，我们在每一个被标记的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，tu，v*)=Lcls(*p，u*)+*λ*[*u>1]Lloc*(tu，*v)，*(*1)*其中*Lcls*(p，*u)=*log pu*是*真类u的对数损失。

*-*

第二种任务损失，Lloc，是在一个元组上定义的。

类*u*的真界箱回归目标，*v*=（vx*，vy，*vw*，*vh），预测元组*tu*=（*tu，tu，tu，tu*）。

其余的 RoIs 是按照[[11](#_bookmark37)]的方法，从具有最大 IoU 的对象 Pro- posals 中取样，这些对象 Pro- posals 的接地真值在区间[0*.*1*,* 0*.*5]内。这些都是背景例子，并被标记为*u* = 0。0*.*1的较低阈值似乎是作为硬例子挖掘的启发式[[8](#_bookmark34)]。在训练过程中，图像以0*.*5的概率被水平翻转。没有使用其他数据增强。

**通过RoI池层进行反向传播。**后向传播通过RoI池化层路由衍生物。为了清楚起见，我们假设每个迷你批次（*N* = 1）只有一个图像，尽管扩展到*N > 1*是直接的，因为前向传递独立处理所有图像。

*>*

x y w h

再次为类*u*。当 *u* 1 时，艾弗森括号指示函数 [*u* 1] 的值为 1，否则为 0。习惯上，所有背景类都被标记为 *u* = 0。对于背景 RoIs，并不存在地面真相的概念。

*> >*

*e*

设 *xi* 十为 RoI 池中的第 *i* 个激活输入-。

ing层，并让*yrj*是该层从*r*- th RoI的第*j*个输出。RoI池层计算*yrj*＝*xi皿*(*r，j)*，其中*i\**(*r，j*)＝*argmaxi╱eR*(*r，j*) *xi╱* 。*R*(*r, j)*是指数

子窗口中的输入集，输出单元 *yrj* 最大限度地集中在该窗口中。一个*xi*可以被分配给几个不同的输出单元*yrj*。

RoI池层的后向函数按照argmax开关计算损失函数相对于每个输入变量*xi的*部分导数。

*∂L*=*i*=*i\**(*r，j*)] *∂L 。*(4)

*∂xi*

*∂yrj*

*r*

*j*

换句话说，对于每一个小批量RoI *r*和每一个池化输出单元*yrj*，如果*i*是最大池化为*yrj*选择的argmax，则部分导数*∂L/∂yrj*被累加。在反向传播中，部分导数*∂L/∂yrj*由RoI池化层之上的层的反向函数随时计算。

**SGD超参数。**用于 softmax 分类和bounding-box 回归的全连接层分别从零均值高斯分布初始化，stan- dard偏差为 0*.*01 和 0*.*001。所有层的权重学习率为1，偏置学习率为2，全局学习率为0*.*001。当在VOC07或VOC12 trainval上训练时，我们运行SGD进行30k次小批量迭代，然后将学习率降低到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，如后所述。使用0*.*9的动量和0*.*0005的参数衰减（关于权重和偏置）。

## 规模不变性

我们探索了两种实现规模不变的ob-ject检测的方法。(1)通过"蛮力"学习和(2)通过使用- ing图像金字塔。这些策略遵循[[11](#_bookmark37)]中的两种方法。在"蛮力"方法中，在训练和测试过程中，每幅图像都以预先确定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对象检测。

与此相反，多尺度方法通过图像金字塔为网络提供近似的尺度不变量。在测试时，图像金字塔被用来对每个对象提案进行近似的尺度归一化。在多尺度训练过程中，我们按照[[11](#_bookmark37)]的方法，每次对图像进行采样时，随机抽取一个金字塔尺度，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。

# 快速的R-CNN检测

一旦快速R-CNN网络被重新调谐，检测就相当于运行一个前向通道（假设对象提案被预先计算）。网络需要

测试时间，*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分配的比例是最接近2242像素的区域[11].当使用图像金字塔时，每个RoI都被分配到比例尺上，使比例尺上的RoI最接近2242像素的面积[[11](#_bookmark37)]。

对于每个测试RoI *r*，前向传递输出一个类

*s*

后验概率分布*p*和一组相对于*r*的预测边界框偏移量（*K*类中的每一个类都有自己的重新定义的边界框预测）。我们将一个

对每个对象类*k*的检测一致性为*r，*用

估计概率Pr(class = *k r*) = *pk*。然后，我们使用R-CNN[[9](#_bookmark35)]的算法和设置，对每个类独立进行非最大抑制。

∆

*I*

## 截断SVD以实现更快的检测

对于整个图像分类，计算全连接层所花费的时间比计算conv层要少。相反，对于检测来说，要处理的RoIs数量很大，而且近一半的前向传递时间用于计算全连接层（见图[2](#_bookmark19)）。大的全连接层很容易通过用截断的SVD进行com-压来加速[[5](#_bookmark31)，[23](#_bookmark49)]。

在这种技术中，一个由*u v*的参数化的层

*x*

权重矩阵*W*近似分解为

*W s U* Σ*tV T* (5)

使用 SVD。在这种分解中，*U*是一个*u t*矩阵，由*W*的前*t个*左弦向量组成，Σ*t*是一个*t t*对角矩阵，包含*W*的前*t个*奇异值，*V*是*v t*矩阵，包含*W*的前t个右弦向量。截断SVD将参数数从*uv*减少到*t*(*u* + *v*)，如果*t*比min(*u, v*)小得多，则参数数会很明显。为了压缩网络，对应于*W*的单个全连接层被两个全连接层所取代，它们之间没有非线性。这些层中的第五层使用权重矩阵Σ*tV T*(且没有偏置)，第二层使用*U*(与*W*相关的原始偏置)。当RoIs数量较大时，这种简单的压缩方法可以提供良好的速度。

*x*

*x*

*x*

# 主要成果

有三个主要结果支持本文的贡献。

1. 关于VOC07、2010和2012年的最先进的磁通量
2. 与R-CNN、SPPnet相比，训练和测试速度快。
3. 微调VGG16中的conv层，提高mAP值

## 实验设置

我们的实验使用了三个预训练的ImageNet模型，这些模型可以在网上获得。[2](#_bookmark9) 第一种是来自R-CNN[[9](#_bookmark35)]的CaffeNet（本质上是AlexNet[[14](#_bookmark40)]）。我们也可以将其称为

作为输入的图像（或图像金字塔，编码为一个列表）。

的图像）和*R*对象提案列表进行评分。在

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 办法 | 火车组 | 航空 | 自行车 | 鸟类 | 艭 | 瓶子 | 巴士 | 汽车 | 猫咪 | 椅子 | 牛 | 桌子 | 犬 | 马匹 | 自行车 | 柿子 | 栽 | 绵羊 | 沙发 | 训练 | 电视机 | mAP |
| SPPnet BB [[11](#_bookmark37)]*|*  R-CNN BB [[10](#_bookmark36)] | 07 差异  07 | 73.9  73.4 | 72.3  77.0 | 62.5  63.4 | 51.5  45.4 | 44.4  **44.6** | 74.4  75.1 | 73.0  78.1 | 74.4  79.8 | 42.3  40.5 | 73.6  73.7 | 57.7  62.2 | 70.3  79.4 | 74.6  78.1 | 74.3  73.1 | 54.2  64.2 | 34.0  **35.6** | 56.4  66.8 | 56.4  67.2 | 67.9  70.4 | 73.5  **71.1** | 63.1  66.0 |
| FRCN [我们的] | 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 [我们的]  FRCN [我们的] | 07 差异  07+12 | 74.6  **77.0** | **79.0**  78.1 | 68.6  **69.3** | 57.0  **59.4** | 39.3  38.3 | 79.5  **81.6** | **78.6**  **78.6** | 81.9  **86.7** | **48.0**  42.8 | 74.0  **78.8** | 67.4  **68.9** | 80.5  **84.7** | 80.7  **82.0** | 74.1  **76.6** | 69.6  **69.9** | 31.8  31.8 | 67.1  **70.1** | 68.4  **74.8** | 75.3  **80.4** | 65.5  70.4 | 68.1  **70.0** |

表1.挥发性有机物2007年检测平均精度（%）。**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 trainval，**07 diff**:**07**

*(*

没有"difﬁcult"的例子，**07+12**：**07**和VOC12 trainval的联合。*Ⅰ*SPPnet结果是由[[11](#_bookmark37)]的作者编写的。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 办法 | 火车组 | 航空 | 自行车 | 鸟类 | 艭 | 瓶子 | 巴士 | 汽车 | 猫咪 | 椅子 | 牛 | 桌子 | 犬 | 马匹 | 自行车 | 柿子 | 栽 | 绵羊 | 沙发 | 训练 | 电视机 | mAP |
| 宝宝学习 | 提案。 | 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](#_bookmark36)] | 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 [我们的] | 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 [我们的] | 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.挥发性有机物2010年检测平均精度（%）。**VOC 2010测试**检测平均精度（%）。BabyLearning使用基于[[17](#_bookmark43)]的网络。其他方法均使用VGG16。训练集键：**12**：VOC12 trainval，**Prop.**：专有数据集，**12+seg**：**12个**带分段注释的数据集，**07++12**：VOC07 trainval、VOC07测试和VOC12 trainval的联合。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 办法 | 火车组 | 航空 | 自行车 | 鸟类 | 艭 | 瓶子 | 巴士 | 汽车 | 猫咪 | 椅子 | 牛 | 桌子 | 犬 | 马匹 | 自行车 | 柿子 | 栽 | 绵羊 | 沙发 | 训练 | 电视机 | mAP |
| 宝宝学习 | 提案。 | 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](#_bookmark36)] | 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 [我们的] | 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 [我们的] | 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.挥发性有机物2012年测试平均检测精度（%）。**VOC 2012测试**检测平均精度（%）。BabyLearning和NUS NIN c2000使用基于[[17](#_bookmark43)]的网络。所有其他方法使用VGG16。训练集键：见表[2](#_bookmark12)，**Unk.**：未知**。**

以这个CaffeNet为模型**S**，代表"小"。第二种网络工作是来自[[3](#_bookmark29)]的VGG CNN M 1024，它的深度与**S**相同，但更宽。我们称这个网络为模型**M**，代表"中"。最后一个网络是来自[[20](#_bookmark46)]的很深的VGG16模型。由于这个模型是最大的，我们称它为模型**L**。 在本节中，所有的实验都使用*单规模的*训练和测试（*s*=600；详见[5.2](#_bookmark22)节）。

## 2010年和2012年挥发性有机化合物结果

在这些数据集上，我们将快速R-CNN（Fast R-CNN，简称*FRCN*）与公共排行榜上comp4（外部数据）赛道上的顶级方法进行比较（表[2](#_bookmark12)，表[3](#_bookmark13)）。[3](#_bookmark14) 对于NUS NIN c2000和BabyLearning方法，目前没有相关的出版物，我们也无法找到所使用的ConvNet架构的确切信息，它们是Network-in-Network设计的变体[[17](#_bookmark43)]。所有其他方法都是从同一个预训练的VGG16网络初始化的。

快速R-CNN在VOC12上取得了最高的结果，其mAP为65.7%（额外数据为68.4%）。它也比其他方法快了两个数量级，这些方法都是基于"慢"R-CNN管道的。在VOC10上。

3 http://host.robots.ox.ac.uk:8080/leaderboard。

(2015年4月18日访问)

SegDeepM[[25](#_bookmark51)]实现了比Fast R-CNN更高的mAP（67.2% vs. 66.1%）。SegDeepM是在VOC12 train- val加上segmentation annotations上进行训练的，它是通过使用马尔科夫随机ﬁeld对R-CNN检测和来自O2P的segmentation进行推理来提升R-CNN的准确性的。

[1](#_bookmark27)]语义分割法。Fast R-CNN可以换成SegDeepM来代替R-CNN，可能会带来更好的效果。当使用放大的07++12训练集时（见表[2](#_bookmark12)标题），Fast R-CNN的mAP in- creases达到68.8%，超过了SegDeepM。

## 2007年VOC结果

在VOC07上，我们将快速R-CNN与R-CNN和SPPnet进行比较。所有的方法都从同一个预训练的VGG16网络开始，并使用边界盒回归。VGG16 SPPnet的结果是由[[11](#_bookmark37)]的作者计算出来的。SPPnet在训练和测试过程中都使用了五个尺度。Fast R-CNN对SPPnet的改进说明，尽管Fast R-CNN使用单尺度的训练和测试，但对conv层的调整可以提供很大的mAP改进（从63.1%到66.9%）。R-CNN实现了66.0%的mAP。作为一个小问题，SPPnet的训练*中没有*PASCAL中标记为"difﬁcult"的例子。去掉这些例子后，快速R-CNN的mAP提高到68.1%。所有其他实验都使用"difﬁcult"的例子。

## 培训和测试时间

快速的训练和测试时间是我们的第二个主要re-sult。表[4比较了](#_bookmark16)快速R- CNN、R-CNN和SPPnet之间在VOC07上的训练时间(小时)、测试速率(每幅图像秒)和mAP。对于VGG16，Fast R-CNN在没有截断SVD的情况下，比R-CNN更快地处理图像146，而使用它则快213。训练时间减少了

*x*

*x*

9 ，从84小时缩短到9.5小时。与SPPnet相比，Fast R- CNN训练VGG16的速度快了2.7个小时（9.5个小时与25.5个小时），在没有截断SVD的情况下，测试速度快了7个小时，而使用SVD则快了10个小时。Fast R-CNN还消除了数百GB的磁盘存储，因为它没有缓存功能。

*xx*

*x*

*x*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 快速R-CNN  **S M L** | | | R-CNN  **S M L** | | | SPPnet  *Ⅰ* **L** |
| 火车时间(小时) | **1.2** | 2.0 | 9.5 | 22 | 28 | 84 | 25 |
| 提速 | **18.3***×* | 14.*0×* | 8.8*×* | *1×* | *1×* | *1×* | 3.4*×* |
| 测试率(秒/月) | 0.10 | 0.15 | 0.32 | 9.8 | 12.1 | 47.0 | 2.3 |
| S与SVD | **0.06** | 0.08 | 0.22 | - | - | - | - |
| 测试加速  S与SVD | *98×*  *169×* | 80*×*  150*×* | *146×*  *213×* | *1×*  - | *1×*  - | *1×*  - | 20*×*  - |
| VOC07 mAP | 57.1 | 59.2 | **66.9** | 58.5 | 60.2 | 66.0 | 63.1 |
| S与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间比较。快速R-CNN使用单尺度模式。SPPnet使用[[11](#_bookmark37)]中规定的五种尺度。*Ⅰ*时序提供

的作者[[11](#_bookmark37)]。时间是在Nvidia K40 GPU上测量的。

**截断SVD。**Truncated SVD可以减少30%以上的检测时间，而mAP的下降幅度很小（0.3%-年龄点），并且在模型压缩后不需要进行额外的ﬁne-tuning。图[2](#_bookmark19)说明了在VGG16的fc6层中使用25088 4096矩阵中的前1024个奇异值和4096 4096 fc7层中的前256个奇异值如何在mAP损失很小的情况下减少运行时间。如果在压缩后再进行一次调整，可能会以较小的mAP下降进一步加速。

*x*

*x*

## 哪些层级要调整？

对于SPPnet pa- per[[11](#_bookmark37)]中所考虑的不太深的网络，只对完全连接的层进行微调似乎就能达到很好的精度。我们推测，这个结果将不持有非常深的网络。为了验证对VGG16来说，对conv层进行精确调整是很重要的，我们使用快速R-CNN进行精确调整，但*冻结了*13个conv层，因此只有完全连接的层可以学习。这种消融模拟了单尺度SPPnet训练，并*将mAP从66.9%降低到61.4%*（表[5](#_bookmark17)）。这个实验验证了我们的假设：通过RoI池化层的训练对于非常深的网是很重要的。

层，在模型**L** SPPnet **L**中进行了重新调整。

VOC07 mAP 61.4

测试率(s/im)**0.32**

*竺* fc6 *竺* conv3 1

66.9

**0.32**

竺可桢2 1

**67.2**

**0.32**

*竺* fc6

63.1

2.3

表5.对VGG16限制哪些层是窄调谐的影响。限制哪些层对VGG16进行微调的效果。微调fc6模拟了SPPnet的训练算法[[11](#_bookmark37)]，但使用的是单一尺度。SPPnet **L 的**结果是 ob-

使用五种尺度进行检测，但速度成本明显（7*倍*）。

*竺*

这是否意味着*所有的*conv层都要进行调整？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，与任务无关的(一个众所周知的事实[[14](#_bookmark40)])。允许conv1学习或不学习，对mAP没有明显的影响。对于VGG16，我们发现只需要更新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更新会使训练速度减慢1.3 (12.5 vs. 9.5小时)；(2)从conv1 1更新会超支GPU内存。当从conv2 1向上学习时，mAP的差异只有+0*.*3点（Ta- ble [5](#_bookmark17)，最后一列）。本文所有快速R-CNN的结果都使用VGG16 ﬁne-tune层conv3 1及以上；所有的经验- ments与模型**S**和**M** ﬁne-tune层conv2及以上。

*x*

# 设计评估

我们进行了实验，以了解快速R-。

正向通过时间 mAP 66.9% @ 320ms /图像

fc6

38.7%(122毫秒)

其他

3.5%(11毫秒) roi\_pool5

5.4%(17毫秒)

6.2%(20ms)

正向传递时间（SVD）mAP 66.6% @ 223ms /图像。

fc6

17.5%(37毫秒)

其他

5.1%(11毫秒)

roi\_

7.9%(17毫秒)

1.7%(4毫秒) fc7

67.8%(143毫秒)

信仰

泳池5

CNN与R-CNN和SPPnet的比较，以及评估设计决策。按照最佳实践，我们在PASCAL VOC07数据集上进行了这些实验。

## 多任务训练有用吗？

多任务训练是方便的，因为它避免了人--。

46.3%(146毫秒)

信仰

fc7

顺序训练任务的流水线老化，但它也有可能改善结果，因为任务之间通过共享表示（ConvNet）相互影响[2]。但它也有可能改善结果，因为任务之间通过共享表示（ConvNet）相互影响[[2](#_bookmark28)]。多任务训练是否能提高对象检测能力？

图2.VGG16在截断SVD之前和之后的时序。VGG16在截断SVD之前和之后的时序。在SVD之前，完全连接的层fc6和fc7需要45%的时间。

在快速R-CNN中，curacy?

为了测试这个问题，我们训练基线网络，只使用公式[1](#_bookmark6)中的分类损失，Lcls（*即*，设置

多任务训练? 阶段性训练? 测试时间bbox注册?VOC07 mAP

**S**

√√

√

√√

52.2 53.3 54.6  **57.1**

**M**

√√

√

√√

54.7 55.5 56.6  **59.2**

**L**

√ √

√

√ √

62.6 63.4 **0 66.9**

表6.多任务训练多任务训练（每组第四列）比计件训练（每组第三列）提高mAP。

*λ* = 0).这些基线打印在表[6](#_bookmark21)中每组模型**S**、**M**和**L的第**5栏中。请注意，这些模型*没有*边界框回归器。接下来(每组第二列)，我们取用多任务损失(公式[1](#_bookmark6)，*λ*=1)训练的网络，但我们在测试时*禁用*边界框回归。这就隔离了网络的clas- siﬁcation精度，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，我们观察到多任务训练相对于单独的分类训练提高了纯分类的准确性。改进的范围从+0*.*8到+1*.*1 mAP点，显示出多任务学习的一致的积极效果。

最后，我们把基线模型（只用分类损失训练），加上边界框回归层，用*Lloc*训练它们，同时保持所有其他净工作参数冻结。每组中的第三列显示了这种*阶段性*训练方案的结果：mAP比第一列有所改进，但阶段性训练低于每组的多任务训练（每组第四列）。

## 尺度不变：要蛮力还是要坚守？

我们比较了两种实现尺度不变的对象检测的策略：蛮力学习（单尺度）和年龄金字塔（多尺度）。在这两种情况下，我们将图像的尺度*s定义*为其*最*短边的长度。

所有单个规模的实验使用*s*=600像素;*s*可能小于600的一些图像，因为我们在1000像素的最长的图像年龄侧捕获，并保持图像的纵横交错。这些值的选择，使VGG16ﬁts在GPU内存中的ne-tuning。较小的模型不受内存限制，可以从较大的*s*值中获益；无论如何，为每个模型优化*s*不是我们的主要关注点。我们注意到，PASCAL图像是384 473像素的av- erage，因此，单尺度设置通常上采样图像1.6的因素。因此，在RoI池层的平均有效步长为10像素。

*x*

*s*

在多尺度设置中，我们使用与[[11](#_bookmark37)]相同的五种尺度(*s* 480*,* 576*,* 688*,* 864*,* 1200)以方便与SPPnet的比较。然而，我们将最长的一面设为2000像素，以避免超过GPU内存。

*e {}*

表[7](#_bookmark23)显示了用一个或五个尺度进行训练和测试时的模型**S**和**M**。也许在[[11](#_bookmark37)]中最令人惊讶的结果是，单尺度检测的性能几乎和多尺度检测一样好。我们的结论是，单尺度检测与多尺度检测的表现几乎一样好。

SPPnet **ZF**

**S**

**M**

scales 1 5 1 5 1 5

测试率(s/im) 0.14 0.38 **0.10 0**.39 0.15 .64

VOC07 mAP 58.0 59.2 57.1 58.4 59.2 60.7

**L**

1

0.32

**66.9**

表7。多尺度与单尺度。SPPnet **ZF**（与模型类似

**S**）的结果来自[[11](#_bookmark37)]。较大的网络与单尺度提供了最好的速度/精度权衡。(由于GPU内存限制，**L**在我们的实现中不能使用多尺度。)

他们的结果是：深度ConvNets善于直接学习- ing尺度不变性。多尺度方法只提供了一个小的增加的mAP，但在计算时间上有很大的成本（表[7](#_bookmark23)）。在VGG16(模型**L**)的情况下，我们受制于实现细节而只能使用单一尺度。然而它实现了66.9%的mAP，比R-CNN[[10](#_bookmark36)]报道的66.0%略高，尽管R-CNN使用了"inﬁnite"尺度，即每个提案都被扭曲成一个规范的大小。

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，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。

## 我们是否需要更多的训练数据？

当提供更多的训练数据时，一个好的物体检测器应该会有所提高。Zhu*等人*[[24](#_bookmark50)]发现DPM[[8](#_bookmark34)]的mAP仅在几百到一千张训练实例后就会饱和。在这里，我们用VOC12训练值集来增强VOC07训练值集，大概将图像数量增加三倍至16.5k，以评估快速R-CNN。扩大训练集可以将VOC07测试的mAP从66.9%提高到70.0%（表[1](#_bookmark11)）。在这个数据集上训练时，我们使用60k个迷你批次迭代，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的联合中构建了一个21.5k图像的数据集。在这个数据集上训练时，我们使用100k SGD迭代，并且每40k次迭代（而不是每30k次）降低0*.*1的学习率。对于VOC10和2012，mAP分别从66.1%提高到68.8%，从65.7%提高到68.4%。

*x*

## SVM的表现是否优于softmax？

快速R-CNN使用在调整过程中学习的softmax分类器，而不是训练单vs休息的线性SVMs。

后，就像在R-CNN和SPPnet中所做的那样。为了under-stand这种选择的影响，我们在Fast R-CNN中实现了带有硬负向挖掘的事后SVM训练。我们使用与R-CNN中相同的训练算法和超参数。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 办法 | 分类 | **S** | **M** | **L** |
| R-CNN[[9](#_bookmark35)，[10](#_bookmark36)] | SVM | **58.5** | **60.2** | 66.0 |
| FRCN [我们的]  FRCN [我们的] | SVM  柔光 | 56.3  57.1 | 58.7  59.2 | 66.8  **66.9** |

表8.Fast R-CNN with softmax vs. SVM (VOC07 mAP)快速R-CNN与softmax与SVM（VOC07 mAP）的对比。

表[8](#_bookmark25)显示，softmax的表现略微优于SVM的所有三个网络，+0*.*1到+0*.*8个mAP点。这个影响很小，但它表明，与之前的多阶段训练方法相比，"一枪"ﬁne-tuning是足够的。我们注意到，softmax与onevs-rest SVMs不同，在对RoI进行评分时引入了类之间的竞争。

## 提案越多就一定越好吗？

有两种类型的对象检测器(大致)：使用*稀疏*对象提案集的对象检测器(*如*选择性搜索[[21](#_bookmark47)])和使用*密集提案*集的对象检测器(*如*DPM[[8](#_bookmark34)])。稀疏提案的分类是一种*级联*[[22](#_bookmark48)]，提案机制首先拒绝大量的候选者，留给分类器一个小的集合来评估。当应用于DPM检测时，这种级联可以提高检测精度[[21](#_bookmark47)]。我们发现有证据表明，提案-分类器级联也提高了快速R-CNN的准确性。

利用选择性搜索的*质量模式*，我们对每张图像从1k到10k个提案进行扫荡，每次*重新训练*和*重新测试*模型**M**，如果提案是纯粹的计算作用，增加每张图像的提案数量应该不会损害mAP。

这个结果在没有实际运行实验的情况下是难以预测的。最先进的测量对象提案质量的方法是平均召回（AR）[[12](#_bookmark38)]。对于使用R-CNN的几种提案方法，*当使用每张图像的提案数量为五倍时，*AR与mAP的相关性很好。图[3](#_bookmark26)显示，当每幅图像的提案数变化时，AR（红色实线）与mAP的相关性并不好。AR必须谨慎使用，由于更多的提案，更高的AR并不意味着mAP会增加。幸运的是，使用模型**M进行**训练和测试需要不到2.5小时。因此，快速R-CNN可以有效地、直接地评估对象提案mAP，这比代理指标要好。

我们还研究了快速R-CNN当使用*密集*生成的盒子（超过比例，位置和纵横比），以大约45k盒子/图像的速度。这个密集集足够丰富，当每个选择性搜索框被其最接近的（以IoU为单位）密集框替换时，mAP仅下降1个点（至57.7%，图[3](#_bookmark26)，蓝色三角形）。

密集箱的统计与选择性搜索箱的统计不同。从2k个选择性搜索框开始，当*加入*1000个2*，*4*，*6*，*8*，*10*，*32*，*45个密集框的随机样本时，我们测试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

*x {}*

框添加后，mAP比添加更多选择性搜索框时下降更强烈，最终达到53.0%。

我们还*只*使用密集的盒子（45k /图像）来训练和测试快速R-CNN。这种设置产生了52.9%的mAP（蓝钻）。最后，我们检查是否需要带有硬负值挖掘的SVM来应对密集框的distribu- tion。SVMs的表现更差：49.3%（蓝圈）。

## 初步的MS COCO结果

我们将Fast R-CNN（使用VGG16）应用于MS COCO数据集[[18](#_bookmark44)]，以建立初步的基线。我们在80k的图像训练集上进行240k的迭代训练，并在80k的图像上进行了

66

63

61

58

mAP

56

53

51

3 49 4

10 10

提议的目标数量

66 使用评估服务器在"test-dev"集上进行评估。

PASCAL式的mAP为35.9%；新的COCO式。

Sel.搜索(SS)

SS（2k）+兰德密集型SS取代密集型SS

45k Dense Softmax 45k Dense SVM

SS Avg.召回

63 AP，其平均值也超过了IoU阈值，为19.7%。

61

平均回收率

# 结论

58

本文提出的Fast R-CNN，是对R-CNN和SPPnet的干净快速更新。除了报告State-of-RNN和SPPnet之外，还报告了R-CNN的

56

53艺术检测结果，我们提出了详细的实验，即

51 我们希望提供新的见解。特别值得注意的是，稀疏对象提案似乎可以提高探测器的质量。 这

49 问题在过去过于昂贵（时间上），无法探究，但由于

来实用的快速R-CNN。当然，可能会有ex-

图3.各种提案方案的VOC07测试mAP和AR。VOC07测试各种提案方案的mAP和AR。

我们发现，随着pro- posal数量的增加，mAP上升，然后略有下降（图[3](#_bookmark26)，蓝色实线）。这个实验表明，用更多的proposal来淹没深度分类器对准确性没有帮助，甚至略有伤害。

是尚未发现的技术，使密集箱的性能与稀疏提案一样好。这种方法如果得到发展，可能有助于进一步加快物体检测的速度。

**Acknowledgements.**我感谢何凯明、Larry Zitnick和Piotr Dolla´r的有益讨论和鼓励。

# 参考文献

* + 1. J.Carreira, R. Caseiro, J. Batista, and C. Sminchisescu.S- mantic segmentation with second-order pooling.In *ECCV*，2012.[5](#_bookmark10)
    2. R.Caruana.多任务学习。*Machine learning*，28（1），1997．[6](#_bookmark15)
    3. K.Chatﬁeld, K. Simonyan, A. Vedaldi, and A. Zisserman.细节中的魔鬼归来。深入研究对话网。In *BMVC*，2014.[5](#_bookmark10)
    4. J.Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei- Fei.ImageNet．一个大规模的分层图像数据库。

在*CVPR*，2009年。[2](#_bookmark2)

* + 1. E.E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus.Exploiting linear structure within convolutional networks for efﬁcient evaluation.In *NIPS*, 2014.[4](#_bookmark7)
    2. D.Erhan, C. Szegedy, A. Toshev, and D. Anguelov.Scalable object detection using deep neural networks.In *CVPR*, 2014.[3](#_bookmark4)
    3. M.Everingham, L. Van Gool, C. K. I. Williams, J. Winn, 和

A.Zisserman.PASCAL视觉对象类（VOC）。

挑战。*IJCV*，2010。[1](#_bookmark0)

* + 1. P.Felzenszwalb, R. Girshick, D. McAllester, and D. Ra- manan.Object detection with discriminatively trained part based models.*TPAMI*，2010。3, [7](#_bookmark20), [8](#_bookmark24)
    2. 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](#_bookmark4), [4](#_bookmark7), [8](#_bookmark24)
    3. R.Girshick, J. Donahue, T. Darrell, and J. Malik.基于区域的卷积网络用于精确的物体检测和分割。*TPAMI*，2015。5, [7](#_bookmark20), [8](#_bookmark24)
    4. K.He, X. Zhang, S. Ren, and J. Sun.Spatial pyramid pooling in deep convolutional networks for visual recognition.In *ECCV*, 2014.1, [2](#_bookmark2), [3](#_bookmark4), [4](#_bookmark7), [5](#_bookmark10), [6](#_bookmark15), [7](#_bookmark20)
    5. J. H. Hosang, R. Benenson, P. Dolla´r, and B. Schiele. What makes for effective detection proposals? *arXiv preprint* *arXiv:1502.05082*, 2015.[8](#_bookmark24)
    6. Y.Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Gir- shick, S. Guadarrama, and T. Darrell.Caffe:Caffe: Convolutional architecture for fast feature embedding.In *Proc.of the ACM* *International Conf.on Multimedia*，2014.[2](#_bookmark2)
    7. A.Krizhevsky, I. Sutskever, and G. Hinton.ImageNet clas- siﬁcation with deep convolutional neural networks.In *NIPS*, 2012.1, [4](#_bookmark7), [6](#_bookmark15)
    8. S.S. Lazebnik, C. Schmid, and J. Ponce.超越特征袋。识别自然场景类别的空间金字塔匹配。在*CVPR*，2006年。[1](#_bookmark0)
    9. 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](#_bookmark0)

* + 1. M.Lin，Q. Chen，and S. Yan.Network in network.In *ICLR*，2014.[5](#_bookmark10)
    2. T.Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick,

J. Hays, P. Perona, D. Ramanan, P. Dolla´r, and C. L. Zit- nick.Microsoft COCO：common objects in context.*arXiv e-prints*，arXiv:1405.0312 [cs.CV]，2014.[8](#_bookmark24)

* + 1. 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](#_bookmark0), [3](#_bookmark4)
    2. K.Simonyan and A. Zisserman.Very deep convolutional networks for large-scale image recognition.In *ICLR*, 2015.[1](#_bookmark0), [5](#_bookmark10)
    3. J.Uijlings, K. van de Sande, T. Gevers, and A. Smeulders.

选择性搜索的对象识别。*IJCV*，2013．[8](#_bookmark24)

* + 1. P.Viola和M. Jones。使用简单特征增强级联的快速物体检测。在*CVPR*，2001年。[8](#_bookmark24)
    2. J.Xue, J. Li, and Y. Gong.用奇异值分解重构深度神经网络声学模型.

In *Interspeech*, 2013.[4](#_bookmark7)

* + 1. X.Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes.我们需要更多的训练数据或更好的模型来进行物体检测吗？在*BMVC*，2012年。[7](#_bookmark20)
    2. 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](#_bookmark0), [5](#_bookmark10)