Fast R-CNN原文翻译

翻译 Alphapeople 2018-12-12 16:34:25 3558 收藏 14

分类专栏: 人工智能 深度学习 计算机视觉

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).FAST r-CNN建立在以前工作的基础上，利用深度卷积网络对目标进行有效分类。与以往的工作相比，FAST r-CNN在提高训练和测试速度的同时，也提高了检测精度。Fast r-cnn训练非常深的vg g16网络比rcnn快9倍，测试时快213倍，并在Pascal voc 2012上取得了更好的效果。与sppnet相比，Fast r-cnn训练速度为VGG16 的3倍快，测试速度为10倍快，且精度更高。FAST r-cnn是在python和c(使用caffe)中实现的，并且可以在开放源代码的MIT许可下获得：https: //github.com/rbgirshick/fast-rcnn.

一、引言

最近，深度网络[14，16]大大提高了图像分类[14]和目标检测[9，19]的准确性。与图像分类相比，目标检测是一项更具有挑战性的任务，需要更复杂的方法来解决。由于这一复杂性，目前的方法(例如，[9，11，19，25]在多级管道中训练模型是缓慢和不优雅的。

复杂性的产生是因为检测需要精确地定位对象，这就产生了两个主要的挑战。首先，必须处理许多候选对象位置(通常称为“建议框”)。第二，这些候选框只提供粗略的本地化，必须加以改进才能实现精确的本地化。这些问题的解决方案往往会影响速度、准确性或简单性。

在本文中，我们简化了基于ConvNet的目标检测器的训练过程[9，11].我们提出了一种单级训练算法，它联合学习对目标方案进行分类并细化它们的空间位置。

该方法可以训练出一个非常深的检测网络比VGG16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(不包括对象提议时间)，同时在Pascal voc 2012[7]上达到最高精度，为66%(r-cnn为62%)。

1.1、RCNN和SPP-Net

基于区域的卷积网络方法(Rcnn)[9]利用深度ConvNet对目标方案进行分类，获得了很好的目标检测精度。然而，R-CNN有明显的缺点：

1.训练是一个多阶段的过程。美国有线电视新闻网(CNN)第一次使用日志丢失完成了关于对象提案的ConvNet。然后，它适合svms的ConvNet特性。这些svms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，边框才被学习到。

2.训练在空间和时间上都是昂贵的。对于svm和bounding-box回归的训练，从每个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(如vgg16)，这个过程需要GPU用2.5天的时间来处理07年trainval集的5k图像。这些特性需要数百GB的存储空间。

3.目标检测速度慢。在测试时，从每个测试图像中的每个对象方案中提取特征.使用vgg16进行检测需要47s/图像(在GPU上)。

r-cnn速度慢，因为它对每个对象方案执行一个ConvNet向前传递，而不需要共享计算。空间金字塔池网络(Sppnet)[11]被提出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映射，然后使用从共享特征映射中提取的特征向量对每个对象方案进行分类。通过最大限度地将提案内的特征映射部分合并成固定大小的输出(例如，6×6)，为提案提取特征。多个输出大小被池化，然后像在空间金字塔

池[15]中那样连接。在测试时间，sppnet将r-cnn加速10至100倍。由于建议特征提取速度更快，训练时间也减少了3倍。

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，用日志丢失对网络进行微调，训练svm，并最终安装bounding-box回归器。特性也被写入磁盘。但与r-cnn不同，微调算法不能更新空间金字塔池之前的卷积层。毫不奇怪，这种限制(固定的卷积层)限制了非常深的网络的精度。

1.2. Contributions

我们提出了一种新的训练算法，在提高训练速度和精度的同时，弥补了R-CNN和sppnet的缺点。我们称这种方法为“Fast R-CNN”，因为它的训练和测试相对较快。fast-rcnn方法有几个优点：

1.检测质量(MAP)高于R-CNN，sppnet

2.训练是单一阶段，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。

FAST r-cnn是用python和c++(caffe[13])编写的，在开放源码的MIT许可下可以在 https://github.com/rbgirshick/ fast-rcnn.

二、Fast-RCNN的架构和训练

Fig.1说明了Fastr-cnn的体系结构。一个快速的r-cnn网络接受一个完整的图像和一组对象建议作为输入。该网络首先用多个卷积(Conv)和最大池化层处理整个图像，生成一个Conv特征映射。然后，对于每个对象提案，感兴趣区域(ROI)池化层从特征映射中提取固定长度的特征向量。每个特征向量被输入到一个完全连接(FC)层的序列中，这些层最终被划分为两个兄弟级输出层：一个在k个对象类上生成Softmax概率估计，另一个层为每个k个对象类输出四个实值数字。每组4个值编码一个k类的bounding-box位置。

2.1 ROI池化层

ROI池层使用最大池化将感兴趣的任何有效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是与任何特定ROI无关的层超参数。在本文中，ROI是进入Conv特征映射的矩形窗口。每个roi由一个四元组(r，c，h，w)定义，该元组指定其左上角(r，c)及其高度和宽度(h，w)。

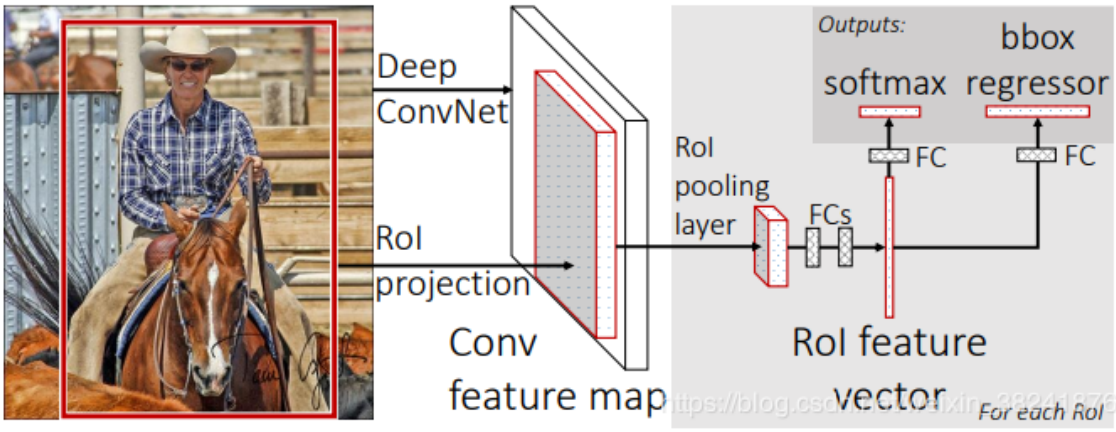

图1.快速r-CNN架构。输入图像和多个感兴趣区域(ROIS)被输入到全卷积网络中。每个ROI集合成一个固定大小的特征映射，然后通过完全连接层(FCS)映射到一个特征向量。该网络每个ROI有两个输出向量：Softmax概率和每类bounding-box回归偏移量。该体系结构是经过训练的端到端的多任务损失。

ROI最大池化层的工作原理是将h×w 的roi窗口划分为近似大小为h/H×w/W的子窗口的H×W的网格，然后将每个子窗口中的值合并到相应的输出网格单元中。池化独立地应用于每个特征映射通道，如标准的最大池化。ROI层只是sppnet[11]中使用的空间金字塔池层的特例，其中只有一个金字塔级别。我们使用[11]中给出的池子窗口计算。

2.1、从预先训练的网络初始化

我们用三个预先训练过的ImageNet[4]网络进行实验，每个网络都有5个最大池化层和5-13个Conv层(详见4.1节)。当预先训练的网络初始化一个快速的r-cnn网络时，它会经历三个转换.。

首先，将最后一个最大池化层替换为ROI池化层，通过设置h和w与网络的第一个全连接层(例如，h=w=7表示vgg16)来配置h和w。

其次，网络的最后一个全连接层和Softmax(为1000路ImageNet分类训练)被替换为前面描述的两个兄弟层(k1类别上的全连接层和Softmax以及特定类别的bounding-box回归器)。

2.2、检测微调

利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。

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层的反向传播效率很低，这正是r-cnn和sppnet网络被训练的方式。低效率源于这样一个事实：每个ROI可能有一个非常大的接受场，通常跨越整个输入图像。由于前传必须处理整个接收场，训练输入量很大(通常是整个图像)。

提出了一种在训练过程中利用特征共享的高效训练方法。在快速rcnn训练中，随机梯度下降(SGD)小批被分层采样，首先对n个图像进行采样，然后从每幅图像中采样r/n ROI。关键的是，来自同一图像的ROIS在向前和向后传递中共享计算和内存。使n小了，减少了小批量计算.例如，当使用n=2和r=128时，所提出的训练方案比从128幅不同图像(即R-CNN和sppnet策略)中采样一次ROI快约64倍。

对这种策略的一个关注是，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。这似乎不是一个实际的问题，我们取得了较好的结果，n=2和r=128使用较少的SGD迭代比r-CNN。

除了分层抽样之外，FAST r-cnn还使用了一个简化的训练过程，通过一个微调阶段联合优化Softmax分类器和bounding-box回归器，而不是将Softmax分类器、svms和回归器训练在三个不同的阶段[9，11]。此过程的组成部分(损失、小批量抽样策略、通过ROI池层的反向传播和SGD超参数)如下所述。

多任务损失：一个Fastr-cnn网络有两个兄弟输出层。第一个输出离散概率分布(每roi)，$p = (p_0.....p_k)$，k+1以上类别。和往常一样，p是由全连接层的k1输出上的Softmax计算的。第二兄弟层输出bounding-box回归偏移量$t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$。对于每个k对象类，按k索引。我们使用了[9]中给出的t^k参数化，其中t^k指定了相对于对象方案的尺度不变平移和对数空间高度/宽度移动。

每个训练指标都有一个ground-trut类u和一个ground-truth bounding-box回归目标v。我们使用多任务损失L对每个标记的roi进行分类和bounding-box回归的联合训练：

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

其中$L_{cls}(p, u) == -logp_u$是u的log损失。

对于u而言，$u, v = (v_x, v_y, v_w, v_h)$和预测元组$t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$，定义了第二个任务损失$L_{loc}$。相反，当u≥1和0时，指示函数[u≥1]计算为1。按照惯例，捕获所有背景类标记为u=0。对于背景ROI，不存在

ground-truth bounding-box的概念，因此L_{loc}被忽略。对于bounding-box回归，我们使用损失：

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$$

这个 $smooth_{L_1}(x) = \begin{cases} 0.5x^2 if(|x| < 1) \\ |x| - 0.5 otherwise \end{cases}$

是一种鲁棒的L1损失，它对离群值的敏感性不如r-cnn和sppnet中使用的L2损失。当回归目标是无界的，L2损失的训练可能需要仔细调整学习率，以防止梯度暴涨。Eq3消除了这种敏感性 。

方程中的超参数λ1控制两项任务损失之间的平衡。我们将ground-truth回归指标vi规范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对象提取网络。与我们的AP-proach不同，[6]主张采用两种网络系统，即SEPA对本地化和分类进行分级。Overfeat[19]、r-cnn[9]和sppnet[11]也训练分类器和 bounding-box定位器，但是这些方法都是分阶段训练的，我们发现对于快速r-cnn来说，这些方法是次优的(5.1节)。

小批量抽样 ： 在微调过程中，每个SGD小批由n=2幅图像组成，随机一致地选择(通常的做法是，我们实际上迭代了数据集的排列)。我们使用r=128的小批次，从每幅图像中抽取64 rois.与[9]一样，我们从与联合(IOU)交叉的对象提案中获取25%的ROI，与至少0.5的ground-truth bounding-box重叠。这些ROI包括标记为前台对象类(即u≥1)的示例。其余的ROI是从目标提案中取样的，这些提案在[11]之后的[0.1，0.5]间隔内有maximum Iou。这些是背景示例，标记为u=0。较低的阈值0.1似乎作为一个启发式的硬例子挖掘[8]。在训练期间，图像水平翻转，概率为0.5。不使用其他数据增强。

通过roi池化层的反向传播： 反向传播将衍生物通过ROI池化层传播。为了清晰起见，我们假设每个小批只有一个图像(n=1)，但是对n>1的扩展很简单，因为向前传递处理所有图像都是独立的。

设$x_i \in R$是ROI池化层的第一个激活输入，而y_{rj}是该层从ROI输出的第j个激活输入。ROI池化层计算$y_{rj} = x_{i*(r,j)}$，其中$i*(r,j) = argmax_{i \in R(r,j)} x_i$。$R(r,j)$是输出单元$y_(rj)$的最大池化所经过的子窗口中输入的索引集。可以将单个xi分配给几个不同的输出y_{rj}。

ROI池化层的后向函数计算损失函数相对于每个输入变量xi的偏导数，方法是按照argmax转换：

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

也就是说，对于每个小型批处理ROI，r和每个池输出单元y_{rj}，如果我是通过最大池化为y_{rj}选择的argmax，则会累积偏导数$\frac{\partial L}{\partial y_{rj}}$。在反向传播中，偏导数$\frac{\partial L}{\partial y_{rj}}$已经由ROI池化层上的层的后向函数来计算。

SGD超参数 ： 利用标准差分别为0.01和0.001的零均值高斯分布，对用于softmax（软最大值分类）和bounding-box回归的全连接层进行了初始化。偏差被初始化为0。所有层都使用1的每层学习速率作为权重，2用于偏置，全局学习速率为0.001。在VOC07或VOC12的训练中，我们运行SGD进行30k的小批量迭代，然后将学习率降低到0.0001，再训练10K的迭代。当我们对较大的数据集进行训练时，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(关于权重和偏置)。

2.4.尺度不变性

我们探讨了两种实现尺度不变目标检测的方法：(1)通过“brute force”学习和(2)利用图像金字塔。这些战略遵循[11]中的两种方法。在“brute force”法中，在训练和测试过程中，每幅图像都以预先定义的像素大小进行处理。网络必须从训练数据中直接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了近似的尺度不变性.在测试时，图像金字塔被用来对每一个目标方案进行近似规模的规范化。在多尺度训练中，我们随机抽样一个金字塔尺度，每次一幅图像被采样，如下[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。

三、Fast R-CNN的检验

一旦fast r-cnn网络被微调，检测就相当于向前通过(假设目标提案是预先计算的)。该网络以图像(或图像金字塔，编码为图像列表)和r对象提案列表来得分作为输入。在测试时，r通常在2000左右，尽管我们将考虑它更大的情况(≈45k)。当使用图像金字塔时，每个ROI被分配到缩放，使得缩放后的ROI在区域[11]中最接近224x224像素。

对于每个测试的ROI，r正向传递输出一个类后验概率分布p和一组相对于r的预测bounding-box偏移量(每个k类都有自己的精化边界盒预测)。我们使用估计的概率$Pr(class = k|r) = p_k$为每个对象类k指定一个检测置信度为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极大抑制。

3.1.截断SvD以提高检测速度

对于全图像分类，计算全连接层的时间比Conv层少。相反，为了检测要处理的ROI数目很大，并且几乎一半的前向通过时间用于计算全连接层(参见图)。使用截断的SvD[5，23]压缩大的全连接的层很容易加速。

在这种技术中，含有权重为u×v的矩阵w参数的层被近似分解为：

$$W \approx U \sum_t V_T$$

使用SvD。在这个因式分解中，u是一个u×t矩阵，它包含w的第一个t左奇异向量，σt是包含w的t个奇异值的t×t对角矩阵，v是包含w的第一个t-右奇异向量的v×t矩阵。截断的SvD将参数从UV减少到t(Uv)，如果t比min(u，v)小得多，这是非常显著的。为了压缩网络，将对应于w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σtv t(和无偏差)，第二层使用u(具有与w相关的原始偏差)。这种简单的压缩方法提供了良好的加速比，当ROIs的数目很大时。

四、主要结论

三个主要结果支持了本文的贡献：

1、最新VOC07，2010和2012

2、与r-cnn，sppnet 相比，快速训练和测试。

3、微调vgg16中的conv层改进map

4.1.实验装置

我们的实验使用了三个预先训练过的ImageNet模型，它们都可以在线上使用。第一个是来自r-cnn[9]的caffenet(本质上是alexnet[14])。我们也可以将此caffenet称为模型s，表示“小”。第二个网络是[3]的vgg CNN m 1024，其深度与s相同，但更宽。我们把这个网络模型叫做m，意思是“中等”。最后的网络是[20]中非常深的vgg16模型。因为这个模型是最大的，所以我们称它为模型L。在本节中，所有的实验都使用单尺度的训练和测试(s=600；详见5.2节)。

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

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

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

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

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

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

表3.2012年VOC测试平均检测精度(%)。在C 2000中，BabyLearning和NUS nus使用基于[17]的网络。所有其他方法都使用vgg16。训练设置键：见表2，Unk。：未知。

4.2.2010年和2012年成果

在这些数据集上，我们比较了FAST r-CNN(简称FRCN)与Comp 4(外部数据)轨道上来自公共领导板的顶级方法(表2，表3)。3对于NUS nin C 2000和婴儿学习方法，目前还没有相关的出版物，我们无法找到使用的ConvNet体系结构的确切信息；它们是网络中网络设计的变体[17]。所有其他方法都是从同一个预先训练的vgg16网络中初始化的。

FAST r-CNN以65.7%的特征(和68.4%的额外数据)获得了VOC12的最高结果。这也是两个数量级的速度比其他方法，这都是基于“慢”的r-cnn管道。在VOC10上，SegDeepM[25]实现了一个更高的mAP比快速的r-cnn(67.2%比66.1%)。SegDeepM是关于VOC12加分割注释的训练；它的设计是为了提高r-cnn的准确性，使用马尔可夫随机场对r-cnn检测和从O2P[1]语义分割方法中的分段进行推理。快速r-cnn可以用SegDeepM代替r-cnn，这样可以得到更好的效果。当使用扩大的07 12训练集(见表2标题)时，快速r-cnn的地图增加到68.8%，超过了SegDeepM。

4.3. VOC 2007 结果

在VOC2007，我们比较了快速的r-cnn，r-cnn和sppnet.所有方法都从相同的预先训练的vgg16网络开始，并使用bounding-box回归。vgg16sppnet结果由[11]作者计算。sppnet在训练和测试中使用五种量表。快速r-cnn在sppnet上的改进说明，尽管FAST r-cnn使用单尺度的训练和测试，但对conv层的微调使MAP得到了很大的改进(从63.1%提高到66.9%)。R-CNN实现了66.0%的map。作为一个小问题，sppnet在Pascal没有被标记为“困难”的例子进行了训练。删除这些例子提高了快速r-cnn map到68.1%.所有其他实验都使用“困难”的例子。

4.4.训练和测试时间

快速的训练和测试时间是我们的第二大成果。表4比较了快速rcnn、r-cnn和sppnet之间的训练时间(小时)、测试速率(每张图像秒数)，以及voE 07上的地图。对于vg 16，快速r-cnn处理图像的速度比没有截断SvD的r-cnn快146×，用它处理图像的速度为213×。训练时间从84小时减少到9.5小时，缩短了9×10。与sppnet相比，快速rcnn列车vg 16 2.7×捷(9.5比25.5小时)，测试速度7×快而不截断SvD或10×快。快速r-cnn也消除了数百GB的磁盘存储，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	-	-	-	-

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

Table 5. Effect of restricting which layers are fine-tuned fo VGG16. Fine-tuning ≥ fc6 emulates the SPPnet training algo rithm [11], but using a single scale. SPPnet **L** results were ob tained using five scales, at a significant (7×) speed cost.

Does this mean that *all* conv layers should be fine-tuned' In short, *no*. In the smaller networks (**S** and **M**) we find

表4.快速rcnn、r-cnn和sppnet中相同模型的运行时比较。快速r-cnn采用单尺度模式。sppnet使用[11]中规定的五个比例。†时间由[11]的作者提供。在NVIDIA K40 GPU上测量时间。

截段的SvD：截断的SvD可以减少30%以上的检测时间，只需在特征图上下降一点点(0.3个百分点)，并且无需在模型压缩后执行额外的微调。Fig2说明利用vgg16层fc6层中25088×4096矩阵的前1024个奇异值和4096×4096fc7层的前256个奇异值如何在不损失MAP的情况下减少运行时。如果压缩后再进行一次细调的话，map上会有更小的下降，从而可以进一步加快速度.。

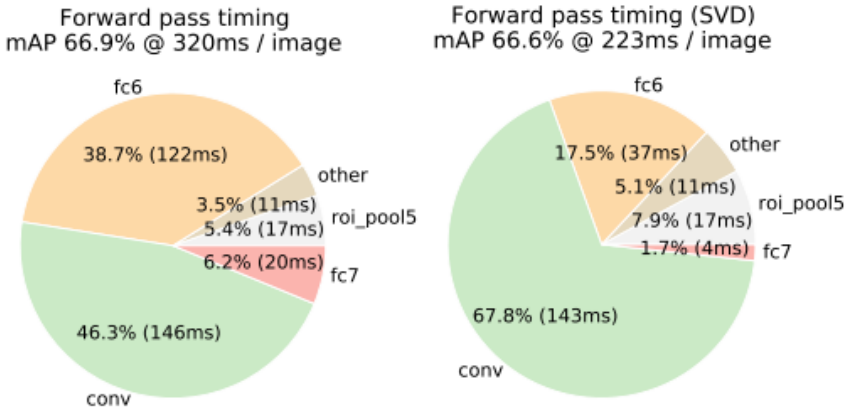

Figure 2. Timing for VGG16 before and after truncated SVD. Before SVD, fully connected layers fc6 and fc7 take 45% of the time.

图2.在截断SvD之前和之后进行vgg16的计时。在SvD之前，全连接层fc6和fc7占45%的时间。

4.5.哪一层要细调?

对于sppnet论文[11]中考虑的深度较小的网络，仅对全连接层进行微调似乎就足以获得很好的精度。我们假设这一结果不会适用于非常深的网络。为了验证微调conv层对于vgg16很重要，我们使用快速r-cnn来微调，但是冻结13个conv层，以便只有全连接层才能学习。此消融模拟单尺度sppnet训练，将MAP从66.9%降至61.4%(表5)。本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非常重要。

这是否意味着所有的conv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是泛型的和任务无关的(众所周知的事实[14])。是否允许conv1学习，对map没有任何意义。对于vgg16，我们发现只需要从conv3_1和以上更新层(13个conv层中的9个)。这是一种实用的观察方法：(1)用conv2_1进行更新，使训练速度比从conv3_1学习慢1.3倍(12.5比9.5小时)；(2)从conv1溢出GPU内存中进行更新。从conv2_1向上学习时，map的差异仅为0.3点(表5，最后一栏)。所有快速r-cnn的结果在本文中使用vgg16微调层conv3_1及以上；所有的实验模型s和m微调层conv2以上。

5.设计评价

我们进行了实验，以了解rcnn与r-cnn和sppnet相比有多快，以及评估设计决策。按照最佳实践，我们在Pascal语音07数据集上进行了这些实验。

5.1.多任务训练有用吗?

多任务训练是很方便的，因为它避免了管理一系列经过连续训练的任务。但是它也有改善结果的潜力，因为任务通过共享表示(ConvNet)相互影响[2]。多任务训练能提高快速r-cnn的目标检测精度吗?

为了测试这个问题，我们在eq中训练只使用分类损失LCLS的基线网络。1(即设置λ=0)。这些基线是在表6每组的第一栏中为模型s、m和l打印的。请注意，这些模型没有bounding-box回归器。接下来(每组第二列)，我们采用多任务损失(Eq)训练的网络。1，λ=1)，但我们在测试时禁用了bounding-box回归。这隔离了网络的分类精度，并允许将苹果与基线网络进行比较。

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

Table 6. Multi-task training (forth column per group) improves mAP over piecewise training (third column per group).

在这三个网络中，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的准确性。改进范围为0.8~1.1个地图点，多任务学习显示出一致的积极效果。

最后，我们采用基线模型(只对分类损失进行训练)，在bounding-box回归层上定位，并使用L_{loc}对其进行训练，同时保持所有其他网络参数的冻结。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，但分阶段训练的效果不如多任务训练(每组第四列)。

5.2.尺度不变性：蛮力还是技巧?

我们比较了两种实现尺度不变目标检测的策略：蛮力学习(单尺度)和图像金字塔(多尺度).在任何一种情况下，我们定义图像的尺度s为其最短边的长度。

对于某些图像，所有使用s=600像素的单尺度实验都可能小于600，因为我们将最长的图像侧限制在1000个像素，并保持图像的高宽比。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约束，可以从更大的s值中受益；但是，为每个模型优化s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提升1.6倍。因此，ROI池化层的平均有效步长为≈10像素。

在多尺度设置中，我们使用[11](s∈{480，576，688，864，1200})中指定的五个标度，以便于与sppnet进行比较。然而，我们限制最长的一面在2000像素，以避免超过GPU内存。

表7显示了用一种或五种尺度训练和测试的模型s和m。也许在[11]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。我们的发现证实了他们的结果：深度网络擅长直接学习尺度不变性。多尺度方法在计算时间上只提供了少量的地图增长，但成本却很高(表7)。在vgg16(模型l)的情况下，我们仅限于通过实现细节使用单个比例。然而，它实现了66.9%的map，略高于为r-cnn[10]报道的66.0%，尽管r-cnn使用“无限”尺度，即每个提案都被扭曲成标准大小。

由于单尺度处理提供了速度和精度之间的最佳折衷，特别是对于非常深的模型，在这个分区之外的所有实验都使用单尺度的训练和测试，s=600像素。

5.3.我们需要更多的训练数据吗?

当提供更多的训练数据时，一个好的目标检测器应该得到改进。朱等人[24]发现DPM[8]地图仅在几百到1000个训练示例之后就饱和了。在这里，我们用VOC12斜交集增加了VOC07的斜交集，大致将图像的数量增加到16.5k，以评估快速的r-cnn。训练集的扩大提高了VOC07测试的MAP，从66.9%提高到70.0%(表1)。在此数据集上进行训练时，我们使用60k的小批处理迭代，而不是40k。

我们对VOC10和2012进行了类似的实验，我们为其构建了一个21.5k图像的数据集，该数据集来自VOC07trainval、test和voc12trainval。在此数据集上进行训练时，我们使用100 k SGD迭代，将学习速度降低0.1×每40k迭代(而不是每30k)。10和2012年，MAP分别从66.1%提高到68.8%，从65.7%提高到68.4%。

5.4.svms的表现优于Softmax吗?

快速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中所做的那样训练1-VS-REST线性svms。为了了解这一选择的影响，我们在快速r-cnn中实施了带有硬负挖掘的特设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**

Table 8. Fast R-CNN with softmax vs. SVM (VOC07 mAP).

Table 8 shows softmax slightly outperforming SVM for all three networks, by +0.1 to +0.8 mAP points. This effect is small, but it demonstrates that “one-shot” fine-tuning is sufficient compared to previous multi-stage training approaches. We note that softmax, unlike one-vs-rest SVMs, introduces competition between classes when scoring a RoI.

表8显示，在所有三个网络中，Softmax略优于svm，比svm高0.1至0.8个百分点。这一效果很小，但它表明，“一次”微调比以往的多阶段训练方法是足够的。我们注意到，与1-VS-REST svms不同的是，Softmax在得分ROI时引入了类间的竞争。

5.5.更多的建议总是更好吗?

有(广义)两种类型的对象检测器：使用稀疏的对象建议集(例如选择性搜索[21])和使用密集的检测器(例如DPM[8])。分类稀疏提案是一种级联式[22]，在这种类型中，提案机制首先拒绝了大量的候选方案，使

分类器有一小部分待评估的集合。当应用于DPM检测时，这种级联提高了检测精度[21]。我们发现，提出的分类器级联也提高了快速的r-CNN的准确性。

采用选择性搜索的质量模式，对每幅图像从1k到10k的方案进行扫描，每次对模型m进行再训练和再测试。如果提案是一个纯粹的计算角色，增加每个图像的建议数量不应该损害mAp。

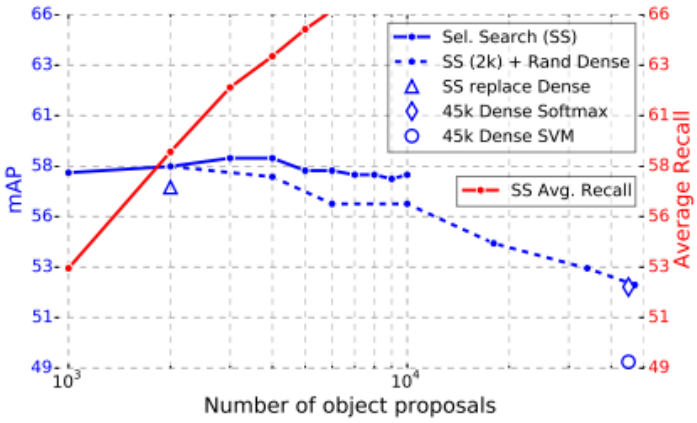

Figure 3. VOC07 test mAP and AR for various proposal schemes.

我们发现，随着建议数的增加，map会上升，然后略有下降(图)。3、实心蓝线)。这个实验表明，用更多的建议淹没深层分类器并没有帮助，甚至轻微地降低了准确性。

如果没有实际的实验，这个结果是很难预测的。测量对象提案质量的最新技术是平均召回(Ar)[12]。当每幅图像使用固定数目的建议时，AR与map有很好的相关性，使用r-cnn的几种建议方法。Fig3表明AR(实心红线)与map的相关性不是很好，因为每幅图像的建议数是不同的。必须谨慎使用AR；更多提案所导致的更高的AR并不意味着MAP将增加。幸运的是，使用m模型进行训练和测试所需时间不到2.5个小时。因此，快速r-cnn能够有效、直接地评估对象提案图，这比代理度量更可取。

我们还研究了快速r-cnn时，使用密集产生的盒子(超过规模，位置，和纵横比)，在大约45K盒/图像。这个密集足够丰富，当每个选择性搜索框被其最近的(IOU)密集框所取代时，MAP只下降1个点(到57.7%，图)。3，蓝色三角形)。

密集框的统计量与选择性搜索框的统计值不同。从2k选择性搜索盒开始，加入1000×{2，4，6，8，10，32，45}密集盒的随机样本，对map进行测试。对于每个实验，我们重新训练和再测试模型m。当添加这些密集框时，MAP下降的幅度比添加更多选择性搜索框时更大，最终达到53.0%。

我们还训练和测试快速r-cnn只使用密集盒(45K/图像)。此设置产生的map为52.9%(蓝色钻石)。最后，我们检查是否需要硬负挖掘的svms来处理密集盒分布。svms做得更糟：49.3%(蓝色圆圈)。

5.6.初步的Ms coco结果

我们将快速r-cnn(Vg 16)应用于Ms coco数据集[18]，以建立一个初步的基线。我们在80k图像训练集上训练240 k迭代，并使用评估服务器在“test-dev”集上进行评估。帕斯卡式地图为35.9%；新的coco式AP(平均超过IOU阈值)为19.7%。

六、结论

本文提出了一种对R-CNN和sppnet进行快速更新的方法.除了报告最新的检测结果外，我们还提供了详细的实验，希望能提供新的见解。特别值得注意的是，稀疏的目标建议似乎提高了检测器的质量。在过去，这个问题花费太大(时间太长)，但随着r-cnn的快速增长，这个问题变得实际起来。当然，可能还存在

一些尚未被发现的技术，这些技术允许密集的盒子执行以及稀疏的建议。这些方法，如果得到发展，可能有助于进一步加快目标检测。

致谢。我感谢凯明，他，拉里齐尼克，和皮奥特美元提供了有益的讨论和鼓励。