Fast R-CNN

Ross Girshick

微软研究院

rbg@microsoft.com

摘要

本文提出了一种快速的基于区域的卷积网络方法（fast R-CNN）用于目标检测。Fast R-CNN 建立在以前使用的深卷积网络有效地分类目标的成果上。相比于之前的研究工作，Fast R-CNN 采用了多项创新提高了训练和测试速度，同时也提高了检测准确度。Fast R-CNN 训练非常深的 VGG16 网络比 R-CNN 快 9 倍，测试时快 213 倍，并在 PASCAL VOC 上得到了更高的准确度。与 SPPnet 相比，Fast R-CNN 训练 VGG16 网络比他快 3 倍，测试速度快 10 倍，并且更准确。Fast R-CNN 的 Python 和 C++（使用 Caffe）实现以 MIT 开源许可证发布在：https://github.com/rbgirshick/fast-rcnn。

1. 引言

最近，深度卷积网络[14, 16]已经显著提高了图像分类[14]和目标检测[9, 19]的准确性。与图像分类相比，目标检测是一个更具挑战性的任务，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]）采用多级 pipelines 的方式训练模型，既慢且精度不高。

复杂性的产生是因为检测需要目标的精确定位，这就导致两个主要的难点。首先，必须处理大量候选目标位置（通常称为“proposals”）。

第二，这些候选框仅提供粗略定位，其必须被精细化以实现精确定位。这些问题的解决方案经常会影响速度、准确性或简洁性。

在本文中，我们简化了最先进的基于卷积网络的目标检测器的训练过程[9, 11]。我们提出一个单阶段训练算法，联合学习候选框分类和修正他们的空间位置。

结果方法能够训练非常深的检测网络（例如 VGG16），其网络比 R-CNN 快 9 倍，比 SPPnet 快 3 倍。在运行时，检测网络在 PASCAL VOC 2012 数据集上实现最高准确度，其中 mAP 为 66％（R-CNN 为 62％），每张图像处理时间为 0.3 秒，不包括候选框的生成（注：所有的时间都是使用一个超频 875MHz 的 Nvidia K40 GPU 测试的）。

1.1. R-CNN 与 SPPnet

基于区域的卷积网络方法（RCNN）[9]通过使用深度卷积网络来分类目标候选框，获得了很高的目标检测精度。然而，R-CNN 具有明显的缺点：

1. 训练过程是多级 pipeline。R-CNN 首先使用目标候选框对卷积神经网络使用 log 损失进行 fine-tunes。然后，它将卷积神经网络得到的特征送入 SVM。这些 SVM 作为目标检测器，替代通过 fine-tunes 学习的 softmax 分类器。在第三个训练阶段，学习 bounding-box 回归器。

2. 训练在时间和空间上是的开销很大。对于 SVM 和 bounding-box 回归训练，从每个图像中的每个目标候选框提取特征，并写入磁盘。对于 VOC07 trainval 上的 5k 个图像，使用如 VGG16 非常深的网

络时，这个过程在单个 GPU 上需要 2.5 天。这些特征需要数百 GB 的存储空间。

3. 目标检测速度很慢。在测试时，从每个测试图像中的每个目标候选框提取特征。用 VGG16 网络检测目标时，每个图像需要 47 秒（在 GPU 上）。

R-CNN 很慢是因为它为每个目标候选框进行卷积神经网络前向传递，而没有共享计算。SPPnet 网络[11]提出通过共享计算加速 R-CNN。SPPnet 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来对每个候选框进行分类。通过最大池化将候选框内的特征图转化为固定大小的输出（例如 6×6）来提取针对候选框的特征。多输出尺寸被池化，然后连接成空间金字塔池[15]。SPPnet 在测试时将 R-CNN 加速 10 到 100 倍。由于更快的候选框特征提取，训练时间也减少了 3 倍。

SPP 网络也有显著的缺点。像 R-CNN 一样，训练过程是一个多级 pipeline，涉及提取特征、使用 log 损失对网络进行 fine-tuning、训练 SVM 分类器以及最后拟合检测框回归。特征也要写入磁盘。但与 R-CNN 不同，在[11]中提出的 fine-tuning 算法不能更新在空间金字塔池之前的卷积层。不出所料，这种局限性（固定的卷积层）限制了深层网络的精度。

1.2. 贡献

我们提出一种新的训练算法，修正了 R-CNN 和 SPPnet 的缺点，同时提高了速度和准确性。因为它能比较快地进行训练和测试，我们

称之为 Fast R-CNN。Fast RCNN 方法有以下几个优点：

1. 比 R-CNN 和 SPPnet 具有更高的目标检测精度（mAP）。
2. 训练是使用多任务损失的单阶段训练。
3. 训练可以更新所有网络层参数。
4. 不需要磁盘空间缓存特征。

Fast R-CNN 使用 Python 和 C++(Caffe[13])编写，以 MIT 开源许可证发布在：https://github.com/rbgirshick/fast-rcnn。

2. Fast R-CNN 架构与训练

Fast R-CNN 的架构如图 1 所示。Fast R-CNN 网络将整个图像和一组候选框作为输入。网络首先使用几个卷积层（conv）和最大池化层来处理整个图像，以产生卷积特征图。然后，对于每个候选框，RoI 池化层从特征图中提取固定长度的特征向量。每个特征向量被送入一系列全连接（fc）层中，其最终分支成两个同级输出层 ：一个输出 K 个类别加上 1 个包含所有背景类别的 Softmax 概率估计，另一个层输出 K 个类别的每一个类别输出四个实数值。每组 4 个值表示 K 个类别中一个类别的修正后检测框位置。

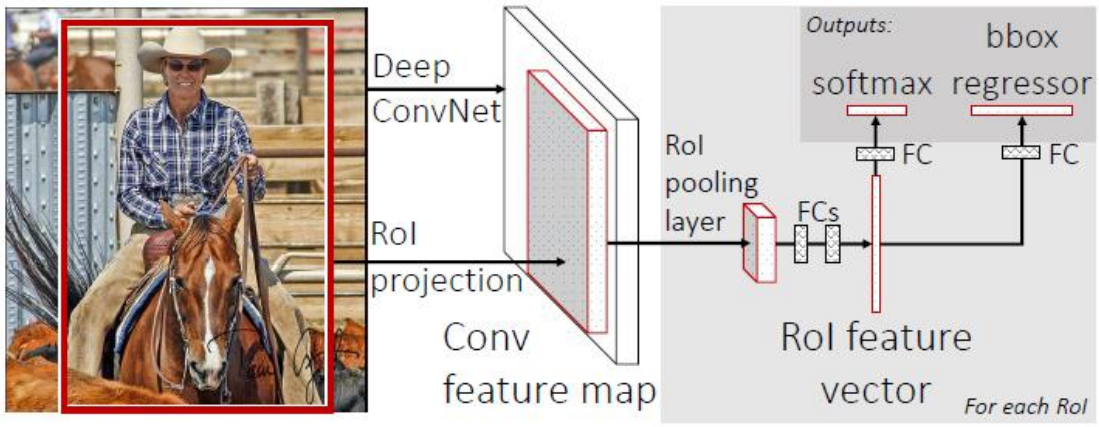

图 1. Fast R-CNN 架构。输入图像和多个感兴趣区域（RoI）被输入

到全卷积网络中。每个 RoI 被池化到固定大小的特征图中，然后通过全连接层（FC）映射到特征向量。网络对于每个 RoI 具有两个输出向量：Softmax 概率和每类 bounding-box 回归偏移量。该架构是使用多任务损失进行端到端训练的。

2.1. RoI 池化层

RoI 池化层使用最大池化将任何有效的 RoI 内的特征转换成具有 H×W（例如，7×7）的固定空间范围的小特征图，其中 H 和 W 是层的超参数，独立于任何特定的 RoI。在本文中，RoI 是卷积特征图中的一个矩形窗口。每个 RoI 由指定其左上角(r,c)及其高度和宽度(h,w)的四元组(r,c,h,w)定义。

RoI 最大池化通过将大小为 h×w 的 RoI 窗口分割成 H×W 个网格，子窗口大小约为 h/H×w/W，然后对每个子窗口执行最大池化，并将输出合并到相应的输出网格单元中。同标准的最大池化一样，池化操作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层的特例，其只有一个金字塔层。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

我们实验了三个预训练的 ImageNet [4]网络，每个网络有五个最大池化层和 5 至 13 个卷积层（网络详细信息见 4.1 节）。当预训练网络初始化 fast R-CNN 网络时，其经历三个变换。

首先，最后的最大池化层由 RoI 池层代替，其将 H 和 W 设置为与网络的第一个全连接层兼容的配置（例如，对于 VGG16，H=W=7）。

然后，网络的最后一格全连接层和 Softmax（其被训练用于 1000 类 ImageNet 分类）被替换为前面描述的两个同级层（全连接层和 K+1 个类别的 Softmax 以及特定类别的 bounding-box 回归）。

最后，网络被修改为采用两个数据输入：图像的列表和这些图像中的 RoI 的列表。

2.3 微调

用反向传播训练所有网络权重是 Fast R-CNN 的重要能力。首先，让我们阐明为什么 SPPnet 无法更新低于空间金字塔池化层的权重。

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 层的反向传播是非常低效的，这正是训练 R-CNN 和 SPPnet 网络的方法。低效是因为每个 RoI 可能具有非常大的感受野，通常跨越整个输入图像。由于正向传播必须处理整个感受野，训练输入很大（通常是整个图像）。

我们提出了一种更有效的训练方法，利用训练期间的特征共享。在 Fast RCNN 网络训练中，随机梯度下降（SGD）的小批量是被分层采样的，首先采样 N 个图像，然后从每个图像采样 R/N 个 RoI。关键的是，来自同一图像的 RoI 在前向和后向传播中共享计算和内存。减小 N，就减少了小批量的计算。例如，当 N=2 和 R=128 时，得到的训练方案比从 128 幅不同的图采样一个 RoI（即 R-CNN 和 SPPnet 的策略）快 64 倍。

这个策略的一个令人担心的问题是它可能导致训练收敛变慢，因为来自相同图像的 RoI 是相关的。这个问题似乎在实际情况下并不存

在，当 N=2 和 R=128 时，我们使用比 R-CNN 更少的 SGD 迭代就获得了良好的结果。

除了分层采样，Fast R-CNN 使用了一个精细的训练过程，在 fine-tuning 阶段联合优化 Softmax 分类器和 bounding-box 回归，而不是分别在三个独立的阶段训练 softmax 分类器、SVM 和回归器[9, 11]。下面将详细描述该过程（损失、小批量采样策略、通过 RoI 池化层的反向传播和 SGD 超参数）。

多任务损失。Fast R-CNN 网络具有两个同级输出层。第一个输出在 K+1 个类别上的离散概率分布（每个 RoI），$p=(p0,\ldots,pK)$。通常，基于全连接层的 K+1 个输出通过 Softmax 来计算 p。第二个输出层输出 bounding-box 回归偏移，即 $t^k=(t^k_x, t^k_y, t^k_w, t^k_h)$，k 表示 K 个类别的索引。我们使用[9]中给出方法对 t^k 进行参数化，其中 t^k 指定相对于候选框的尺度不变转换和对数空间高度/宽度移位。

每个训练的 RoI 用类真值 u 和 bounding-box 回归目标真值 v 打上标签。我们对每个标记的 RoI 使用多任务损失 L 以联合训练分类和 bounding-box 回归：

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

其中 $L_{cls}(p, u) = -\log p_u$，是类真值 u 的 log 损失。

对于类真值 u，第二个损失 L_{loc} 是定义在 bounding-box 回归目标真值元组 $u, v = (v_x, v_y, v_w, v_h)$和预测元组 tu=(tux,tuy,tuw,tuh)上的损失。Iverson 括号指示函数[u≥1]，当 u≥1 的时候值为 1，否则为 0。按照惯例，任何背景类标记为 u=0。对于背景 RoI，没有检测框真值的概念，

因此 L_{loc} 被忽略。对于检测框回归，我们使用损失：

$$L_{loc}(t^u, v) = \sum_{i\in\{x,y,w,h\}} smooth_{L1}(t_i^u - v_i) \tag{2}$$

其中：

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if \ \ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{3}$$

是鲁棒的 L_1 损失，对于异常值比在 R-CNN 和 SPPnet 中使用的 L_2 损失更不敏感。当回归目标无界时，具有 L_2 损失的训练可能需要仔细调整学习速率，以防止爆炸梯度。公式(3)消除了这种敏感性。

公式(1)中的超参数 λ 控制两个任务损失之间的平衡。我们将回归目标真值 v_i 归一化为具有零均值和方差为 1 的分布。所有实验都使用 λ=1。

我们注意到[6]使用相关损失来训练一个类别无关的目标候选网络。与我们的方法不同的是[6]倡导一个将定位和分类分离的双网络系统。OverFeat[19]，R-CNN[9]和 SPPnet[11]也训练分类器和检测框定位器，但是这些方法使用逐级训练，这对于 Fast R-CNN 来说不是最好的选择。

小批量采样。在微调期间，每个 SGD 的小批量由 N=2 个图像构成，均匀地随机选择（如通常的做法，我们实际上迭代数据集的排列）。我们使用大小为 R=128 的小批量，从每个图像采样 64 个 RoI。如在[9]中，我们从候选框中获取 25％的 RoI，这些候选框与检测框真值的交并比 IoU 至少为 0.5。这些 RoI 只包括用前景对象类标记的样本，即 $u\geq 1$。根据[11]，剩余的 RoI 从候选框中采样，该候选框与检测框

真值的最大 IoU 在区间[0.1, 0.5]。这些是背景样本，并用 u=0 标记。0.1 的阈值下限似乎充当困难样本重挖掘的启发式算法[8]。在训练期间，图像以概率 0.5 水平翻转。不使用其他数据增强。

通过 RoI 池化层的反向传播。反向传播通过 RoI 池化层。为了清楚起见，我们假设每个小批量(N=1)只有一个图像，扩展到 N>1 是显而易见的，因为前向传播独立地处理所有图像。

令 RoI 池化层的第 i 个激活输入 $x_i \in R$，令 y_{rj} 是第 r 个 RoI 层的第 j 个输出。RoI 池化层计算 $y_{rj} = x_{i^*(r,j)}$，其中 $i^*(r,j) = \text{argmax}\, x_{i'}$. $R(r;j)$是输出单元 y_{rj} 最大池化的子窗口中的输入的索引集合。一个 x_i 可以被分配给几个不同的输出 y_{rj}。

RoI 池化层反向传播函数通过遵循 argmax switches 来计算关于每个输入变量 x_i 的损失函数的偏导数：

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

换句话说，对于每个小批量 RoI r 和对于每个池化输出单元 y_{rj}，如果 i 是 y_{rj} 通过最大池化选择的 argmax，则将这个偏导数$\partial L/\partial y_{rj}$ 积累下来。在反向传播中，偏导数$\partial L/\partial y_{rj}$ 已经由 RoI 池化层顶部的层的反向传播函数计算。

SGD 超参数。用于 Softmax 分类和检测框回归的全连接层的权重分别使用具有方差 0.01 和 0.001 的零均值高斯分布初始化。偏置初始化为 0。所有层的权重学习率为 1 倍的全局学习率，偏置为 2 倍的全局学习率，全局学习率为 0.001。当对 VOC07 或 VOC12 trainval 训练时，我们进行 30k 次小批量 SGD 迭代，然后将学习率降低到 0.0001，

再训练 10k 次迭代。当我们训练更大的数据集，我们运行 SGD 更多的迭代，如下文所述。使用 0.9 的动量和 0.0005 的参数衰减（权重和偏置）。

2.4. 尺度不变性

我们探索两种实现尺度不变目标检测的方法：（1）通过“brute force”学习和（2）通过使用图像金字塔。这些策略遵循[11]中的两种方法。在“brute force”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学习尺度不变性目标检测。

相反，多尺度方法通过图像金字塔向网络提供近似尺度不变性。在测试时，图像金字塔用于大致缩放-规范化每个候选框。按照[11]中的方法，作为数据增强的一种形式，在多尺度训练期间，我们在每次图像采样时随机采样金字塔尺度。由于 GPU 内存限制，我们只对较小的网络进行多尺度训练。

3. Fast R-CNN 检测

一旦 Fast R-CNN 网络被微调完毕，检测相当于运行前向传播（假设候选框是预先计算的）。网络将输入图像（或图像金字塔，编码为图像列表）和待计算得分的 R 个候选框的列表作为输入。在测试的时候，R 通常在 2000 左右，尽管我们需要考虑更大（约 45k）的情况。当使用图像金字塔时，每个 RoI 被缩放，使其最接近[11]中的 224^2 个像素。

对于每个测试的 RoI r，正向传播输出类别后验概率分布 p 和相对于 r 的预测检测框偏移集合（K 个类别中的每个类别获得其自己的

修正的检测框预测结果）。我们使用估计的概率 Pr(class=k|r)≙pk 为每个对象类别 k 分配 r 的检测置信度。然后，我们使用 R-CNN [9]算法的设置和算法对每个类别独立执行非极大值抑制。

3.1. 使用截断的 SVD 实现更快的检测

对于整个图像进行分类任务时，与卷积层相比，计算全连接层花费的时间较小。相反，在图像检测任务中，要处理大量的 RoI，并且接近一半的前向传播时间用于计算全连接层（参见图 2）。较大的全连接层可以轻松地通过用截短的 SVD[5, 23]压缩来提升速度。

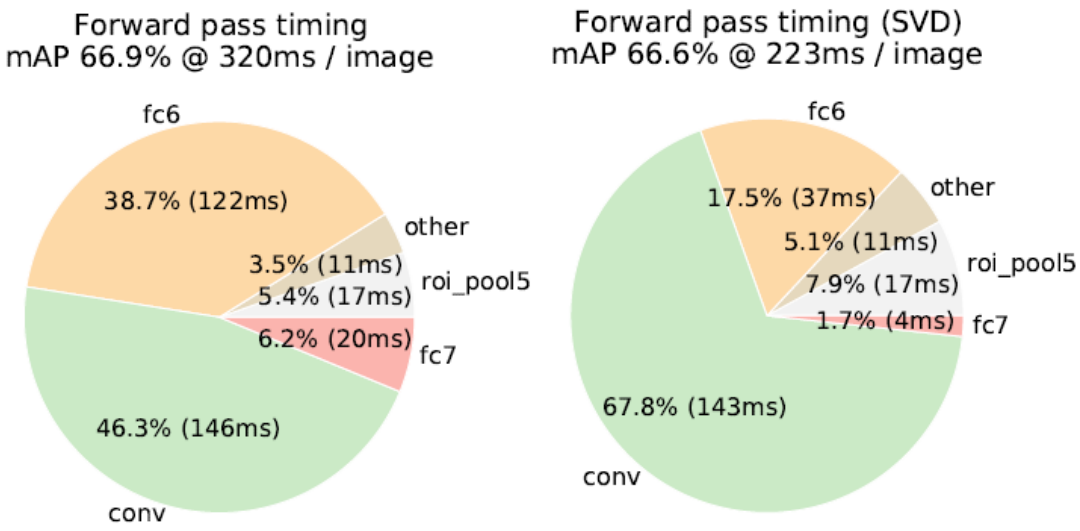

图 2. 截断 SVD 之前和之后 VGG16 的时间分布。在 SVD 之前，全连接层 fc6 和 fc7 消耗 45％的时间。

在这种技术中，层的 u × v 权重矩阵 W 通过 SVD 被近似分解为：

$$W \approx UE_tV^T \tag{5}$$

在这种分解中，U 是一个 u×t 的矩阵，包括 W 的前 t 个左奇异向量，E_t 是 t×t 对角矩阵，其包含 W 的前 t 个奇异值，并且 V 是 v×t 矩阵，

包括 W 的前 t 个右奇异向量。截断 SVD 将参数计数从 uv 减少到 t(u+v)个，如果 t 远小于 min(u,v)，则是非常有意义的。为了压缩网络，对应于 W 的单个全连接层由两个全连接层替代，在它们之间没有非线性。这些层中的第一层使用权重矩阵 E_tV^T（没有偏置），并且第二层使用 U（其中原始偏差与 W 相关联）。当 RoI 的数量较大时，这种简单的压缩方法能实现很好的加速。

4. 主要结果

三个主要结果支持本文的贡献：

1. VOC07，2010 和 2012 的最高的 mAP。

2. 相比 R-CNN，SPPnet，训练和测试的速度更快。

3. 对 VGG16 卷积层 Fine-tuning 后提升了 mAP。

4.1. 实验设置

我们的实验使用了三个经过预训练的 ImageNet 网络模型，这些模型可以在线获得(脚注：https://github.com/BVLC/caffe/wiki/Model-Zoo)。第一个是来自 R-CNN [9]的 CaffeNet（实质上是 AlexNet[14]）。我们将这个 CaffeNet 称为模型 S，即小模型。第二网络是来自[3]的 VGG_CNN_M_1024，其具有与 S 相同的深度，但是更宽。我们把这个网络模型称为 M，即中等模型。最后一个网络是来自[20]的非常深的 VGG16 模型。由于这个模型是最大的，我们称之为 L。在本节中，所有实验都使用单尺度训练和测试（s=600，详见 5.2 节）。

4.2. VOC 2010 和 2012 数据集上的结果

（如上面表 2，表 3 所示）在这些数据集上，我们比较 Fast R-

CNN（简称 FRCN）和公共排行榜中 comp4（外部数据）上的主流方法（脚注：http://host.robots.ox.ac.uk:8080/leaderboard）。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使用的 ConvNet 体系结构的确切信息；它们是 Network-in-Network 的变体 16。所有其他方法都通过相同的预训练 VGG16 网络进行了初始化。

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。所有其他方法使用 VGG16。训练集关键字：12 代表 VOC12 trainval，Prop.代表专有数据集，12+seg 代表具有分割注释的 VOC2012，07++12 代表 VOC2007 trainval、VOC2007 test 和 VOC2012 trainval 的组合。

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。所有其他方法使用 VGG16。训练设置：见表 2，Unk.代表未知。

Fast R-CNN 在 VOC12 上获得最高结果，mAP 为 65.7%（加上额外数据为 68.4%）。它也比其他方法快两个数量级，这些方法都基于比较“慢”的 R-CNN 网络。在 VOC10 上，SegDeepM [25]获得了比 Fast R-CNN 更高的 mAP（67.2%对比 66.1%）。SegDeepM 使用 VOC12

trainval 训练集及分割标注进行了训练，它被设计为通过使用马尔可夫随机场推理 R-CNN 检测和来自 O2P [1]的语义分割方法的分割来提高 R-CNN 精度。Fast R-CNN 可以替换 SegDeepM 中使用的 R-CNN，这可以获得更好的结果。当使用扩大的 07++12 训练集（见表 2 标题）时，Fast R-CNN 的 mAP 增加到 68.8%，超过 SegDeepM。

4.3. VOC 2007 数据集上的结果

在 VOC07 数据集上，我们比较 Fast R-CNN 与 R-CNN 和 SPPnet 的 mAP。所有方法从相同的预训练 VGG16 网络开始，并使用 bounding-box 回归。VGG16 SPPnet 结果由论文[11]的作者提供。SPPnet 在训练和测试期间使用五个尺度。Fast R-CNN 对 SPPnet 的改进说明，即使 Fast R-CNN 使用单个尺度训练和测试，卷积层 fine-tuning 在 mAP 中贡献了很大的改进（从 63.1%到 66.9%）。R-CNN 的 mAP 为 66.0%。其次，SPPnet 是在 PASCAL 中没有被标记为“困难”的样本上进行了训练。除去这些样本，Fast R-CNN 的 mAP 达到 68.1%。所有其他实验都使用被标记为“困难”的样本。

4.4. 训练和测试时间

快速的训练和测试是我们的第二个主要成果。表 4 比较了 Fast RCNN，R-CNN 和 SPPnet 之间的训练时间（单位小时），测试速率（每秒图像数）和 VOC07 上的 mAP。对于 VGG16，没有截断 SVD 的 Fast R-CNN 处理图像比 R-CNN 快 146 倍，有截断 SVD 的 R-CNN 快 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，没有截断 SVD 的 Fast RCNN 训练 VGG16 网络比 SPPnet 快 2.7 倍

（9.5 小时相比于 25.5 小时），测试时间快 7 倍，有截断 SVD 的 Fast RCNN 比的 SPPnet 快 10 倍。Fast R-CNN 还不需要数百 GB 的磁盘存储，因为它不缓存特征。

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

表 4. Fast RCNN，R-CNN 和 SPPnet 中相同模型之间的运行时间比较。Fast R-CNN 使用单尺度模式。SPPnet 使用[11]中指定的五个尺度。†的时间由[11]的作者提供。在 Nvidia K40 GPU 上的进行了时间测量。

截断的 SVD。截断的 SVD 可以将检测时间减少 30%以上，同时能保持 mAP 只有很小（0.3 个百分点）的下降，并且无需在模型压缩后执行额外的 fine-tuning。图 2 显示了如何使用来自 VGG16 的 fc6 层中的 25088×4096 矩阵的顶部 1024 个奇异值和来自 fc7 层的 4096×4096 矩阵的顶部 256 个奇异值减少运行时间，而 mAP 几乎没有损失。如果在压缩之后再次微调，则可以在 mAP 更小下降的情况下进一步提升速度。

4.5. fine-tune 哪些层？

对于 SPPnet 论文[11]中提到的不太深的网络，仅 fine-tuning 全连

接层似乎足以获得良好的准确度。我们假设这个结果不适用于非常深的网络。为了验证 fine-tuning 卷积层对于 VGG16 的重要性，我们使用 Fast R-CNN 进行 fine-tuning，但冻结十三个卷积层，以便只有全连接层学习。这种消融模拟了单尺度 SPPnet 训练，将 mAP 从 66.9%降低到 61.4%（如表 5 所示）。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重要的。

	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. 对 VGG16 fine-tune 的层进行限制产生的影响。fine-tune fc6 以上的层模拟了单尺度 SPPnet 训练算法[11]。SPPnet L 是使用五个尺度，以显著（7 倍）的速度成本获得的结果。

这是否意味着所有卷积层应该进行 fine-tune？简而言之，不是的。在较小的网络（S 和 M）中，我们发现 conv1（译者注：第一个卷积层）是通用的、不依赖于特定任务的（一个众所周知的事实[14]）。允许 conv1 学习或不学习，对 mAP 没有很关键的影响。对于 VGG16，我们发现只需要更新 conv3_1 及以上（13 个卷积层中的 9 个）的层。这个观察结果是实用的：（1）与从 conv3_1 更新相比，从 conv2_1 更新使训练变慢 1.3 倍（12.5 小时对比 9.5 小时），（2）从 conv1_1 更新时 GPU 内存不够用。从 conv2_1 学习时 mAP 仅增加 0.3 个点（如表 5 最后一列所示）。本文中所有 Fast R-CNN 的结果都 fine-tune VGG16 conv3_1 及以上的层，所有用模型 S 和 M 的实验 fine-tune conv2 及以

上的层。

5. 设计评估

我们通过实验来了解 Fast RCNN 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。

5.1. 多任务训练有用吗？

多任务训练是方便的，因为它避免管理顺序训练任务的 pipeline。但它也有可能改善结果，因为任务通过共享的表示（ConvNet）[2]相互影响。多任务训练能提高 Fast R-CNN 中的目标检测精度吗？

为了测试这个问题，我们训练仅使用公式(1)中的分类损失 L_{cls}（即设置 λ=0）的基准网络。这些 baselines 是表 6 中每组的第一列。请注意，这些模型没有 bounding-box 回归器。接下来（每组的第二列），是我们采用多任务损失（公式(1)，λ=1）训练的网络，但是我们在测试时禁用 bounding-box 回归。这隔离了网络的分类准确性，并允许与基准网络的相似类别之类的比较（译者注：apples-to-apples comparision 意思是比较两个相同类别的事或物）。

	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（每组第三列）的 mAP。

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准确度。改进范围从+0.8 到+1.1 个 mAP 点，显示了

多任务学习的一致的积极效果。

最后，我们采用 baseline 模型（仅使用分类损失进行训练），加上 bounding-box 回归层，并使用 L_{loc} 训练它们，同时保持所有其他网络参数冻结。每组中的第三列显示了这种逐级训练方案的结果：mAP 相对于第一列有改进，但逐级训练表现不如多任务训练（每组第四列）。

5.2. 尺度不变性：暴力或精细？

我们比较两个策略实现尺度不变物体检测：暴力学习（单尺度）和图像金字塔（多尺度）。在任一情况下，我们将尺度 s 定义为图像短边的长度。

所有单尺度实验使用 s=600 像素，对于一些图像，s 可以小于 600，因为我们保持横纵比缩放图像，并限制其最长边为 1000 像素。选择这些值使得 VGG16 在 fine-tune 期间不至于 GPU 内存不足。较小的模型占用显存更少，所以可受益于较大的 s 值。然而，每个模型的优化不是我们的主要的关注点。我们注意到 PASCAL 图像平均大小是 384×473 像素的，因此单尺度设置通常以 1.6 的倍数对图像进行上采样。因此，RoI 池化层的平均有效步长约为 10 像素。

在多尺度模型配置中，我们使用[11]中指定的相同的五个尺度（$s \in \{480,576,688,864,1200\}$），以方便与 SPPnet 进行比较。但是，我们限制长边最大为 2000 像素，以避免 GPU 内存不足。

表 7 显示了当使用一个或五个尺度进行训练和测试时的模型 S 和 M 的结果。也许在[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。我们的研究结果能证明他们的结果：深度卷积网络

擅长直接学习到尺度的不变性。多尺度方法消耗大量的计算时间仅带来了很小的 mAP 提升（表 7）。在 VGG16（模型 L）的情况下，我们实现细节限制而只能使用单个尺度。然而，它得到了 66.9％的 mAP，略高于 R-CNN[10]的 66.0％，尽管 R-CNN 在某种意义上使用了“无限”尺度，但每个候选区域还是被缩放为规范大小。

	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. 多尺度对比单尺度。SPPnet ZF（类似于模型 S）的结果来自[11]。具有单尺度的较大网络具有最佳的速度/精度平衡。（L 在我们的实现中不能使用多尺度，因为 GPU 内存限制。）

由于单尺度处理能够权衡好速度和精度之间的关系，特别是对于非常深的模型，本小节以外的所有实验使用单尺度 s=600 像素的尺度进行训练和测试。

5.3. 我们需要更过训练数据吗？

当提供更多的训练数据时，好的目标检测器应该会进一步提升性能。Zhu 等人[24]发现 DPM [8]模型的 mAP 在只有几百到几千个训练样本的时候就达到饱和了。实验中我们增加 VOC07 trainval 训练集与 VOC12 trainval 训练集，大约增加到三倍的图像使其数量达到 16.5k，以评估 Fast R-CNN。扩大训练集将 VOC07 测试集的 mAP 从 66.9％提高到 70.0％（表 1）。使用这个数据集进行训练时，我们使用 60k 次小批量迭代而不是 40k。

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。训练集关键字：07 代表 VOC07 trainval，07\diff 代表 07 没有“困难”的样本，07 + 12 表示 VOC07 和 VOC12 trainval 的组合。SPPnet 结果由[11]的作者提供。

我们对 VOC2010 和 2012 进行类似的实验，我们用 VOC07 trainval、test 和 VOC12 trainval 数据集构造了 21.5k 个图像的数据集。当用这个数据集训练时，我们使用 100k 次 SGD 迭代，并且每 40k 次迭代（而不是每 30k 次）降低学习率 10 倍。对于 VOC2010 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 分类是否优于 Softmax？

Fast R-CNN 在 fine-tune 期间使用 softmax 分类器学习，而不是像 R-CNN 和 SPPnet 在最后训练一对多线性 SVM。为了理解这种选择的影响，我们在 Fast R-CNN 中进行了具有难负采样的事后 SVM 训练。我们使用与 R-CNN 中相同的训练算法和超参数。

如表 8 所示，对于所有三个网络，Softmax 略优于 SVM，mAP 分别提高了 0.1 和 0.8 个点。这个提升效果很小，但是它表明与先前的多级训练方法相比，“一次性”fine-tune 是足够的。我们注意到，不像一对多的 SVM 那样，Softmax 会在计算 RoI 得分时引入类别之间的竞争。

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. 用 Softmax 的 Fast R-CNN 对比用 SVM 的 Fast RCNN（VOC07 mAP）。

5.5. 更多的候选区域更好吗？

（广义上）存在两种类型的目标检测器：一类是使用候选区域稀疏集合检测器（例如，selective search [21]）和另一类使用密集集合（例如 DPM [8]）。分类稀疏候选区域通过一种级联方式[22]的，其中候选机制首先舍弃大量候选区域，留下较小的集合让分类器来评估。当应用于 DPM 检测时，这种级联的方式提高了检测精度[21]。我们发现 proposal-classifier 级联方式也提高了 Fast R-CNN 的精度。

使用 selective search 的质量模式，我们对每个图像扫描 1k 到 10k 个候选框，每次重新训练和重新测试模型 M。如果候选框纯粹扮演计算的角色，增加每个图像的候选框数量不会影响 mAP。

我们发现随着候选区域数量的增加，mAP 先上升然后略微下降（如图 3 蓝色实线所示）。这个实验表明，深度神经网络分类器使用更多的候选区域没有帮助，甚至稍微有点影响准确性。

如果不实际进行实验，这个结果很难预测。用于评估候选区域质量的最流行的技术是平均召回率(Average Recall, AR) [12]。当对每个图像使用固定数量的候选区域时，AR 与使用 R-CNN 的几种候选区域方法时的 mAP 具有良好的相关性。图 3 表明 AR（红色实线）与

mAP 不相关，因为每个图像的候选区域数量是变化的。AR 必须谨慎使用，由于更多的候选区域会得到更高的 AR，然而这并不意味着 mAP 也会增加。幸运的是，使用模型 M 的训练和测试需要不到 2.5 小时。因此，Fast R-CNN 能够高效地、直接地评估目标候选区域的 mAP，是很合适的代理指标。

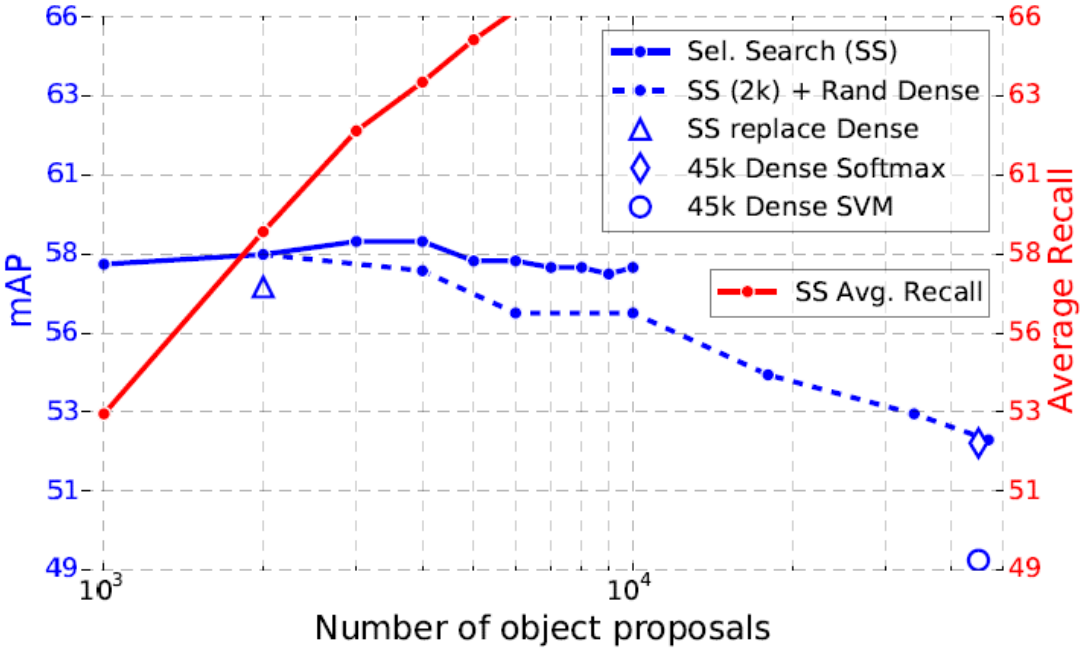

图 3. 各种候选区域方案下 VOC07 测试的 mAP 和 AR。

我们还研究了当使用密集生成框（在不同缩放尺度、位置和宽高比上）大约 45k 个框/图像比例时的 Fast R-CNN 网络模型。这个密集集足够大，当每个 selective search 框被其最近（IoU）密集框替换时，mAP 只降低 1 个点（到 57.7%，如图 3 蓝色三角形所示）。

密集框的统计信息与 selective search 框的统计信息不同。从 2k 个 selective search 框开始，我们再从 $1000\times\{2,4,6,8,10,32,45\}$ 中随机添加密集框，并测试 mAP。对于每个实验，我们重新训练和重新测试模型 M。当添加这些密集框时，mAP 比添加更多选择性搜索框时下降

得更强，最终达到 53.0%。

我们还训练和测试了 Fast R-CNN 只使用密集框（45k/图像）。此设置的 mAP 为 52.9%（蓝色菱形）。最后，我们检查是否需要使用难样本重训练的 SVM 来处理密集框分布。SVM 结果更糟糕：49.3%（蓝色圆圈）。

5.6. MS COCO 初步结果

我们将 Fast R-CNN（使用 VGG16）应用于 MS COCO 数据集[18]，以建立初始 baseline。我们在 80k 图像训练集上进行了 240k 次迭代训练，并使用评估服务器对“test-dev”数据集进行评估。PASCAL 形式的 mAP 为 35.9%；新的 COCO 标准下的 AP 为 19.7%，即超过 IoU 阈值的平均值。

6. 结论

本文提出 Fast R-CNN，一个对 R-CNN 和 SPPnet 更新的简洁、快速版本。除了报告目前最先进的检测结果之外，我们还提供了详细的实验，希望提供新的思路。特别值得注意的是，稀疏目标候选区域似乎提高了检测器的质量。过去这个问题代价太大（在时间上）而一直无法深入探索，但 Fast R-CNN 使其变得可能。当然，可能存在未发现的技术，使得密集框能够达到与稀疏候选框类似的效果。如果这样的方法被开发出来，则可以帮助进一步加速目标检测。

致谢：感谢 Kaiming He，Larry Zitnick 和 Piotr Dollár 的有益的讨论和鼓励。

参考文献

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

[2] R. Caruana. Multitask learning. Machine learning, 28(1), 1997. 6

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In BMVC, 2014. 5

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009. 2

[5] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In NIPS, 2014. 4

[6] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014. 3

[7] M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. IJCV, 2010. 1

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 2010. 3, 7, 8

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

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. 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. Doll´ar, and B. Schiele. What makes for effective detection proposals? arXiv preprint arXiv:1502.05082, 2015. 8

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. 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. Doll´ar, and C. L. Zitnick. Microsoft COCO: common objects in context. arXiv e-prints, arXiv:1405.0312 [cs.CV], 2014. 8

[19] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In ICLR, 2014. 1, 3

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015. 1, 5

[21] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013. 8
[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In CVPR, 2001. 8
[23] J. Xue, J. Li, and Y. Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In Interspeech, 2013. 4
[24] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In BMVC, 2012. 7
[25] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. In CVPR, 2015. 1, 5