Fast R-CNN

Ross Girshick
Microsoft Research

摘要

这篇文章为物体检测提出了一个快速的基于区域的卷积网络方法（Fast R-CNN）。Fast R-CNN 以前工作上使用深度卷积网络来高效分类物体候选。与之前的工作相比，Fast R-CNN 应用了一些创新，在提高检测准确率的同时，提高了训练和测试速度。在训练非常深的 VGG16 时，Fast R-CNN 比 R-CNN 快 9 倍，在测试时快 213 倍，并在 PASCAL VOC 上达到了更高的 mAP。与 SPPnet 相比，Fast R-CNN 在训练时快 3 倍，在测试时快 10 倍，同时更加准确。我们使用 Python 和 C++（使用 Caffe）实现了 Fast R-CNN，同时代码在开源 MIT 执照下可用，见https://github.com/rbgirshick/fast-rcnn。

1 简介

近来，深度卷积网络显著提高了图片分类和物体检测的准确率。与图片分类相比，物体检测是一个更具挑战性的任务，为了求解它要求更加复杂的方法。由于它的复杂性，近来的方法训练多阶段流水线模型，这既缓慢又不优雅。

由于检测对于精确的物体定位的要求导致的复杂性，创造了两个首要的挑战。首先，必须处理大量候选物体位置（经常被称为“候选位置”）。其次，这些候选位置仅仅提供了粗略的位置，必须通过从新调整来得到精细的位置。解决这些问题的办法通常会在速度、准确率或简便性上折中。

在这片文章中，我们优化了基于卷积网络的 sota 物体检测器的训练过程。我们提出了一种可以同时学习分类物体候选并调整它们的空间位置的单一阶段训练算法。

最终的方法在训练很深的检测网络（VGG16[**vgg**]）时比 R-CNN[**rcnn**] 快 9 倍，比 SPPnet[**spp**] 快 3 倍。在运行时，检测器网络处理每张图片的

时间为 0.3 秒（不包括候选位置生成时间），同时在 PASCAL VOV 2012 上达到了 66% 的准确率（R-CNN 的准确率为 62%）。

1.1 R-CNN 和 SPPnet

基于区域的卷积网络方法（R-CNN）[**rcnn**] 通过使用深度卷积网络来为候选物体分类达到了很好的物体检测准确率。然而，R-CNN 有显著缺点：

1. **训练是一个多阶段流水线**。R-CNN 首先使用对数损失在候选物体上微调卷积网络。之后，它使用支持向量机（SVM）拟合卷积特征。哪些 SVM 作为物体检测器，取代了通过微调习得的 softmax 分类器。在第三个训练阶段，学习得到了边界框回归器。

2. **训练在空间和时间上十分昂贵**。为了训练 SVM 和边界框回归器，需要从每张图片的每个物体候选提取特征并写入磁盘。在很深的网络中，例如 VGG16，为了 VOC07 trainval 集合的 5k 张图片，这个过程需要 2.5 个 GPU 日。这些特征要求数百 G 来存储。

3. **物体检测很慢**。在测试时，需要从每张测试图片的每个候选物体提取特征。使用 VGG16 进行检测，在 GPU 上处理一张图片的时间为 47 秒。

R-CNN 之所以慢是因为它在没有共享计算的情况下，为每一个候选物体进行了一次卷积网络的前向传播。空间金字塔池化网络（SPPnets）[**spp**] 通过共享计算提高了 R-CNN 的速度。SPPnet 方法为整个输入图片计算了卷积特征图，之后通过从共享特征图中提取的特征向量来为每个物体候选分类。通过对特征图中候选位置内的部分进行最大池化并得到一个固定大小的输出（例如，6×6）来为候选区域提取特征。池化得到多个输出尺寸之后像空间金字塔池化 [**sp**] 一样将它们连接。SPPnet 在测试阶段为 R-CNN 加速了 10 到 100 倍。训练时间也由于更快的候选位置特征提取减少了 3 倍时间。

然而 SPPnet 也有显著的缺陷。类似 R-CNN，训练是一个多阶段流水面，包括特征提取、使用对数损失微调网络、训练 SVM 和最后的拟合边界框回归器。特征也被写入磁盘。但是不同于 R-CNN，[**spp**] 中提出的微调算法不能更新在空间金字塔池化前的卷积层。毫不意外地，这个局限（固定的卷积层）限制了很深的网络的准确率。

1.2 贡献

我们提出了一个新的弥补了 R-CNN 和 SPPnet 缺陷的新训练算法，同时提高了它们的速度和准确率。由于相比起来更快的训练和测试，我们将这种方法成为 *Fast R-CNN*。Fast R-CNN 方法有如下优势：

1. 比 R-CNN 和 SPPnet 更高的检测质量（mAP）
2. 单阶段，使用多任务损失的训练
3. 可以更新网络所有层的训练
4. 特征缓存不需要磁盘存储

Fast R-CNN 使用 Python 和 C++（Caffe）编写，并在 MIT 执照下开源，见https://github.com/rbgirshick/fast-rcnn。

2 Fast R-CNN 的架构和训练

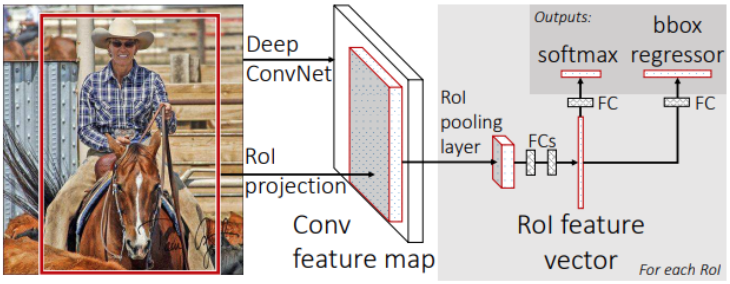

图 1: Fast R-CNN 结构。一张输入图片和多个感兴趣位置（RoIs）被输入一个全卷积网络。每个 RoI 被池化为一个固定大小的特征图之后通过被全连接层映射为一个特征向量。网络对每个 ROI 有两个输出向量：softmax 概率和每个类别的边界框回归偏移。通过多任务损失端到端的训练这个结构。

图1阐述了 Fast R-CNN 的架构。Fast R-CNN 网络将一张完整图片和一系列物体候选作为输入。网络首先对图片进行多次卷积和最大池化来产生一个卷积特征图。之后，感兴趣区域（region of interest, *RoI*）池化层为每一个物体候选从特征图提取一个固定长度的特征向量。每个特征向量被送入一个最终分为两个兄弟输出层的全连接层序列，一个产生在 K 个物体类

别加上一个“背景”类的估计的 softmax 概率，另一个则为 K 个物体类别输出四个实数值。这四个实数为其中一个类别编码了调整后的边界框位置。

2.1 RoI 池化层

RoI 池化层使用最大池化来将在任意合法感兴趣区域内的特征转化为一个有着固定空间大小 $H \times W$（例如，7×7）的小特征图，其中 H 和 W 是与任意 RoI 无关的层超参数。在这篇文章中，RoI 是卷积特征图中的矩形窗口。每一个 RoI 由一个指定它的左上角 (r, c) 和它的高宽的 (h, w) 的四元组 (r, c, h, w) 定义。

RoI 最大池化通过将 $h \times w$ 的 RoI 窗口划分为 $H \times W$ 的子窗口网格，其中每个子窗口的大小近似为 $h/H \times w/W$，之后对每个子窗口中的值进行最大池化来得到对应网格的输出。正如标准最大池化，每个特征图通道的池化是独立进行的。RoI 层是 SPPnets[**spp**] 中使用的空间金字塔池化层的特殊情况。我们使用 [**spp**] 中给出的池化子窗口计算方式。

2.2 从预训练的网络初始化

我们对三种在 ImageNet 山预训练的网络进行了实验，每一个有五个最大池化层以及五到十三个卷积层（网络细节见 4.1 节）。当一个预训练网络初始化 Fast R-CNN 网络时，会进行三种转变。

第一，最后一个最大池化层被替换为通过设置 H 和 W 来与网络的第一个全连接层兼容的 RoI 池化层（例如，在 VGG16 中，$H = W = 7$）。

第二，网络的最后一个全连接层和 softmax（它们为了 1000 路 ImageNet 分类而训练）被替换为早些描述的两个兄弟网络（一个全连接网络和 $K + 1$ 类的 softmax 以及类别特定的边界框回归器）。

第三，网络被修改为接受两个数据输入：一个图片列表和一个这些图片的 RoI。

2.3 为了检测微调

使用反向传播训练网络所有的参数是 Fast R-CNN 的一个重要特性。首先，我们来解释为什么 SPPnet 不能更新空间金字塔池化层下的参数。

根源是当每一个训练样本（也就是 RoI）来自不同的图片时，SPP 层的反向传播将变得十分低效，而这正是 R-CNN 和 SPPnet 是如何训练的。低

效源于每个 RoI 可能有一个非常大的接收域，通常跨越整个输入图像。由于前向传播必须处理全部接收域，所以训练输入是很大的（通常是整个图片）。

我们通过在训练时使用特征共享的优势提出了一个更加高效的训练方法。在 Fast R-CNN 的训练中，随机梯度下降（SGD）的迷你批是层次化采样得到的，首先采样 N 张图片紧接着从每张图片采样 R/N 个 RoI。十分重要的是，来自同一张图片的 RoI 可以在前向和反向传播中共享计算和内存。将 N 变小可以减少迷你批的计算量。例如，当使用 $N = 2$ 以及 $R = 128$ 时，我们提出的训练方案大约比从 128 张不同的图片中采样 RoI（也就是，R-CNN 和 SPPnet 的策略）快 64 倍。

对这个策略的一个担忧在于由于来自同一张图片的 RoI 是相互关联的，这可能导致训练收敛变慢。然而这个担忧并没有成为实际问题，我们在 $N = 2$ 同时 $R = 128$ 的情况下获得了很好的记过，并与 R-CNN 相比使用了更少的 SGD 迭代次数。

除了层次化采样，Fast R-CNN 使用了一种流水线化的训练过程，它使用一个微调阶段同时优化 softmax 分类器和边界框回归器，而不是在三个分离阶段训练 softmax 分类器、SVM 和回归器。这个过程的组件（损失、迷你批采样策略、通过 RoI 池化层反向传播和 SGD 超参数）将会在下面描述。

多任务损失。 Fast R-CNN 网络有两个兄弟输出层。第一个输出一个 $K+1$ 个类别上的离散概率分布（为每一个 RoI），$p = (p_0, \ddots, p_K)$。照常，通过在全连接层的 $K + 1$ 个输出上进行 softmax 得到 p。第二个兄弟层为每一个类别输出边界框回归偏移，$t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$，其中 k 为类别索引。我们使用 [**rcnn**] 中给出的 t^k 参数化方法，其中 t^k 指明了一个与一个物体候选相关的尺度不变转化和对数空间的高/宽转化。

我们为每个训练的 RoI 标记一个 ground-truth 类别 u 和一个 ground-truth 边界框回归目标 v。为了同时训练分类和边界框回归，我们为每个标记的 RoI 使用一个多任务损失 L：

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

其中 $L_{cls} = -\log p_u$ 是对于真实类别 u 的对数损失。

第二个任务损失，L_{loc}，定义在类别为 u 的真实边界框回归目标元组，$v = (v_x, v_y, v_w, v_h)$ 和对于类别 u 的预测元组 $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$。Iverson 括号表达式函数 $u \geq 1$ 当 $u \geq 1$ 是为 1 否则为 0。根据传统，包含全部的背

景类别被标记为 $u = 0$。对于背景 RoI，并不存在 ground-truth 边界框的表示，所以 L_{loc} 忽略它。对于边界框回归，我们使用损失

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u, v_i), \tag{2}$$

其中，

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

是一个鲁棒的 L_1 损失，相比于 R-CNN 和 SPPnet 中使用的 L_2 损失，它对于突出点更加不敏感。当回归目标是无界的时，使用 L_2 损失训练需要小心地依次调整学习率来避免梯度爆炸。公式 3 消除了这个敏感性。

公式 1 中的超参数 λ 控制两个任务的损失间的平衡。我们正则化 ground-truth 回归目标 v_i 的均值为 0 并为单位方差。所有的实验使用 $\lambda = 1$。

我们注意到 [**6**] 中使用了一个相关的损失来训练一个不知类别的物体候选网络。与我们的方法不同的是，[**6**] 使用两个网络来分离定位和分类。OverFeat[**overfeat**]，R-CNN[**rcnn**] 和 SPPnet[**spp**] 也训练了分类器和边界框定位器，然而这些方法使用分阶段训练，我们在 5.1 节中表明这对于 Fast R-CNN 是次优的。

迷你批采样。 在微调过程中，每个 SGD 迷你批都由随机均匀选择（正如常用的方法，我们实际上遍历数据集的排列）的 $N = 2$ 张图片构造。我们使用迷你批的大小为 $R = 128$，从每张图片采样 64 个 RoI。正如 [**rcnn**] 中所说的，我们在与 ground-truth 的 IoU 重叠至少为 0.5 的物体候选挑选 25% 的 RoI。这些 RoI 构成了标记为前景类别（也就是，$u \geq 1$）的样例。遵循 [**spp**]，剩余的 RoI 从与 ground-truth 的最大 IoU 在 $[0.1, 0.5)$ 区间中的物体候选中采样得到。这些是背景样例冰杯标注为 $u = 0$。低阈值 0.1 发挥难例挖掘的启发作用。在训练中，图片以 0.5 的概率水平翻转。其余的数据增强并未使用。

通过 RoI 池化层反向传播。 反向传播指引导数通过 RoI 池化层。虽然由于前向传播过程独立处理所有图片，所以扩展到 $N > 1$ 是十分直接的，但是为了清晰，我们假设每个迷你批只有一张图片（$N = 1$）。

假设 $x_i \in \mathbb{R}$ 为 RoI 池化层的第 i 个激活输入，y_{rj} 是这一层的第 r 个 RoI 的第 j 个输出。RoI 池化层会计算 $y_{rj} = x_{i^*(r,j)}$，其中 $i^*(r, j) =$

$\text{argmax}_{i' \in R(r,j)} x_{i'}$。$R(r,j)$ 是输入的输出单位 y_{rj} 最大池的子窗口的索引集合。单独的 x_i 可能会为多个不同的输出 y_{rj} 赋值。

RoI 池化层的backwards函数通过遵循 argmax 转化计算每个输入变量 x_i 对于损失函数的偏导数：

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

用语言描述来说，对于每一个迷你批的 RoIr 以及对于每一个池化输出单元 y_{rj}，如果 i 是被 argmax 通过最大池化为 y_{rj} 选择的，那么偏导数 $\partial L/\partial y_{rj}$ 将会被累加。在反向传播中，偏导数 $\partial L/\partial y_{rj}$ 已经被 RoI 池化层之上的层的backwards函数计算得到了。

SGD 超参数。 为了 softmax 分类和边界框回归的全连接层分别通过 0 均值 0.01 和 0.001 标准差的高斯分布初始化。偏差被设置为 0。所有层使用对于权重为 1 对偏差为 2 的每一层学习率一集一个 0.001 的全局学习率。当在 VOC07 或 VOC12 上运行时，我们运行迭代 SGD 迷你批 30k 次，之后降低学习率至 0.0001 并再训练迭代 10k 次。正如后面将要描述的，当在更大的数据集上训练时，我们进行更多次 SGD 迭代。我们使用了 0.9 的动量以及 0.0005 的参数（在权重和偏差上）衰减。

2.4 尺度不变性

为了实现尺度不变的物体检测，我们探索了两种方式：(1) 通过 “蛮力” 学习以及 (2) 通过使用图片金字塔。这些策略遵循了 [**spp**] 中的两种方法。在蛮力法中，每张图片在训练和测试中在一个预先定义的像素尺寸上被处理。网络必须直接从训练数据学习尺度不变的物体检测。

多尺度方法，与之形成对比，通过图片金字塔为网络提供了近似的尺度不变性。在测试时，使用图片金字塔来近似尺度正则化每个物体候选。遵循 [**spp**], 在多尺度训练中每次图片被采样时，我们随机采样一个金字塔尺寸，作为一种数据增强的形式。由于 GPU 内存的限制，我们仅在较小的网络上对多尺度训练进行了实验。