基于注意力机制的单阶段目标检测锚点框部件感知特征表达

唐乾坤[1,2], 胡瑜[1,2)*]

[1)] (中国科学院计算技术研究所智能计算机研究中心 北京 100190)
[2)] (中国科学院大学 北京 100049)
(huyu@ict.ac.cn)

摘 要: 针对现有单阶段目标检测算法锚点框特征表达不足影响检测精度的问题, 提出了一种增强锚点框特征表达的算法, 其包含注意力机制模块和部件感知模块. 首先, 注意力机制模块根据各个锚点框的不同属性自适应地提供不同的特征表达. 然后, 部件感知模块准确地提取各个锚点框内部的判别性部件特征以作为各个锚点框进行预测所需的特有特征. 将所提设计与现有 SSD 算法结合并在多个公开的目标检测数据集上进行实验, 结果表明, 所提算法能够显著提高单阶段目标检测算法的精度并维持实时运行速度(14 ms); 进一步地, 在扩展实验上的结果表明, 所提算法也能够改善生成的区域建议框的召回率及两阶段目标检测算法的精度.

关键词: 卷积神经网络; 单阶段目标检测; 区域建议框; 注意力模块; 部件感知模块
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2020.18046

Attention Based Part-Aware Features of Anchor Boxes for Single-Shot Object Detection

Tang Qiankun[1,2] and Hu Yu[1,2)*]

[1)] (*Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)
[2)] (*University of Chinese Academy of Sciences, Beijing* 100049)

Abstract: In this paper, we propose a lightweight but effective design to enhance the anchor representations for improving the performance of single-shot object detectors. This design consists of an attention module and a part-aware module. First, an attention module is applied for a location to adaptively express various representations based on the targets of the anchor boxes covering the location. The part-aware module further extracts the discriminative part features inside each anchor box as its individual features for robust prediction. Assembling the proposed modules to the SSD, our method is able to consistently boost the performance on several public benchmarks while maintaining real-time inference speed (14 ms). Extensive experiments on the region proposal generation indicate that the proposed method also promotes the recall of region proposals, so as the accuracy of two-stage object detection.

Key words: convolution neural networks; single-shot object detection; region proposal; attentive module; part-aware module

目标检测是计算机视觉的基础任务之一, 并被广泛地应用在视频监控、机器人、自动驾驶中.

收稿日期: 2019-08-14; 修回日期: 2019-10-18. **基金项目**: 国家重点研发计划科技创新 2030—"新一代人工智能"重大项目(2018AAA0102701); 空间智能控制技术实验室开放基金(HTKJ2019KL502003); 中国科学院计算技术研究所创新课题(20186090).
唐乾坤(1993—), 男, 博士研究生, 主要研究方向为计算机视觉、目标检测; **胡瑜**(1975—), 女, 博士, 研究员, 博士生导师, CCF 会员, 论文通讯作者, 主要研究方向为自主导航、自动驾驶感知与决策、深度学习算法加速.

目标检测的目的是准确地找出给定图片上物体的位置及其类别. 传统算法使用超像素[1]、边缘[2]等底层特征进行组合和筛选来检测目标, 但是这些底层特征往往不具备高级语义信息, 导致难以获得令人满意的检测结果.

近年来, 深度学习技术的发展为目标检测提供了新的解决思路. 目前, 基于深度学习的目标检测算法大致可分为2类: 两阶段目标检测和单阶段目标检测. 两阶段目标检测算法首先使用区域选择方法(如选择性搜索[1]、区域建议网络(region proposal networks, RPN)[3]等)生成可能存在物体的区域建议框; 然后提取这些区域建议框的特征输入到检测子网络, 用以判断该区域建议框所含物体的类别及边界框. R-CNN[4]首次将区域选择与深度学习相结合, 获得了比传统算法高约 20%的检测精度. 后续 SPPNet[5]和 Fast R-CNN[6]通过共享特征提取网络并采用空间金字塔池化算法提取区域建议框的特征, 从而加快检测速度和提高检测精度. Faster R-CNN[3]则使用RPN生成区域建议框, 并实现 RPN 和检测子网络的端到端训练; 而特征金字塔网络(feature pyramid networks, FPN)[7]、可变形网络(deformable convnet, DCN)[8]的提出, 使得两阶段检测算法获得了更高的检测精度. 然而这些检测算法通常采用复杂的网络结构(如ResNet-101[9]和 ResNeXt-101[10])提取特征, 以及大尺寸(如最短边至少 800 像素)图片作为输入, 导致推测速度往往低于 10 fps.

单阶段目标检测算法由于省略了区域建议框的生成阶段, 直接借助于预先定义的锚点框生成最终的物体边界框, 从而获得实时的运行速度, 但是检测精度往往只有上述两阶段检测算法的10%~20%. 例如, YOLO[11]将输入图片划分为 7×7 的网格, 每个网格用于预测 2 个边界框. YOLO v2[12]则使用诸如批归一化[13]、高分辨率的分类器和回归器等改进 YOLO 网络结构, 并在每个网格的位置放置多个以聚类方式获得长宽比而设置的锚点框. 单发多框检测(single-shot multibox detector, SSD)[14]则以 VGG-16[15]为特征提取网络, 采用分层的锚点框匹配方案检测多尺度的物体. 为了改进单阶段检测模型的精度, 文献[16]将SSD使用的VGG-16 替换为性能更好的 ResNet-101 作为特征提取网络. 而文献[17]提出了 Focal Loss 来改善单阶段检测网络训练时存在的类别不平衡问题, 并使用 ResNet-FPN[7]构建了 RetinaNet[17]网络. 这些算法虽然能获得与两阶段检测网络相当甚至更好的检测精度, 却牺牲了运行速度, 如 SSD 能够达到 40 fps, 而 RetinaNet 仅为 5 fps.

因此, 本文首先分析现有基于锚点框的单阶段检测算法存在的关键问题, 然后针对这些问题提出基于注意力机制的单阶段目标检测锚点框自适应特征表达算法; 并在公开的目标检测数据集(PASCAL VOC[18]和 MS COCO[19])上评估本文算法的检测精度和推测速度; 最后将其扩展到基于锚点框的 RPN, 进一步验证它在两阶段检测算法中的有效性.

1 基于锚点框的单阶段检测算法

本文以 SSD 检测框架为例, 简要介绍基于锚点框的单阶段检测算法原理. 如图 1 所示, 在特征图的每一个位置, 放置 k 个以该位置为中心(图 1 中白色点)的初始框, 即锚点框(图 1 中黑色实线、灰色实线和虚线方框). 为了检测不同大小的物体, 锚点框通常被设置为不同尺度和长宽比. 例如, SSD设置了1种尺度和5种长宽比(1, 2, 3, 1/2, 1/3), 因此 k=6. 然后使用一个 $n\times n$ (通常 n=3)的卷积窗口(图 1 中灰色阴影区域)在特征图上滑动, 以提取各个位置的特征用于该位置锚点框的预测. 分类层判断 k 个锚点框对应的图片区域是背景还是某一具体的类别(PASCAL VOC 有 20 类, MS COCO 有 80 类); 回归层则是回归设定的锚点框和真实物体边界框之间的偏移量(中心点坐标、长、宽).

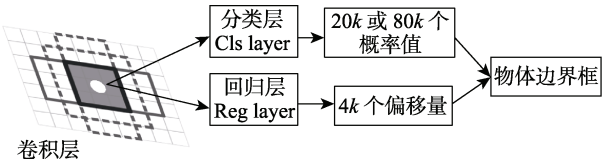


图 1 基于锚点框的单阶段检测算法原理

本文观察发现, 相比于两阶段检测网络, 基于锚点框的单阶段检测网络并未充分利用锚点框的特征表达, 该问题源于如下几个方面.

(1) 同一中心位置的锚点框共享相同特征. 虽然在特征图的每一个位置设定了 k 个不同尺度和长宽比的锚点框, 但是这些锚点框用于分类和回归的特征都是由同一个 $n\times n$ 的卷积核提取. 即使为每个锚点框学习到了不同的权重, 基于该共享特征的预测结果也可能不准确. 一方面是由于采用一个固定尺寸的卷积核提取特征, 未考虑锚点框的尺寸差异, 导致该卷积核所对应的有效感受

野[20]与这 k 个锚点框的尺寸不匹配. 如图 2 所示, 白色区域为有效感受野, 白色框为负例锚点框, 灰色框为正例锚点框, 该有效感受野只占所示锚点框的一小部分. 这就使得卷积核不能够提取足够的信息用于判别这些锚点框所属类别及准确位置. 另一方面, 共享特征降低了用于预测的特征的多样性, 导致分类器和回归器不能被有效地训练. 两阶段算法通过感兴趣区域池化(region of interest pooling, RoI Pooling)步骤提取每个区域建议框内部特征, 为检测子网络提供了多样性特征, 因而能够更鲁棒地预测并获得较高的检测精度.

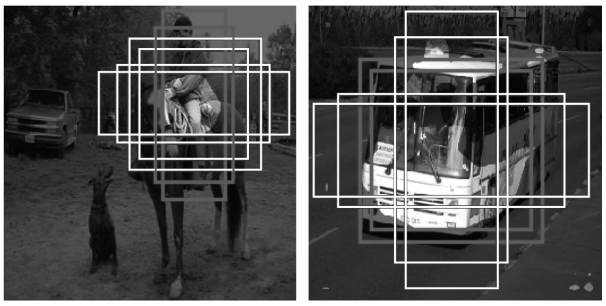


图 2　有效感受野与锚点框的不同尺寸不匹配

(2) 同一位置为不同预测目标的锚点框提供相同的特征表达. 由于设定的锚点框在相应的图片上密集分布, 图片上的某一位置(或区域)可能被多个锚点框包围, 该位置(或区域)应该根据锚点框的预测目标不同提供差异化的特征表达. 如图 3 所示, 某一位置(白色点)被 3 个不同的锚点框包围(白色框表示背景, 黑色框表示摩托车, 灰色框表示人), 对于黑色锚点框, 该位置应该提供更多关于摩托车的判别性信息; 对于灰色锚点框, 该位置应该提供一些上下文信息以便更好地识别人; 对于白色锚点框, 该位置则不应该提供信息. 而现有算法中, 尽管各锚点框的预测目标不同, 该位置通常给包围自身的各个锚点框提供相同的信息, 从而引入预测偏差.

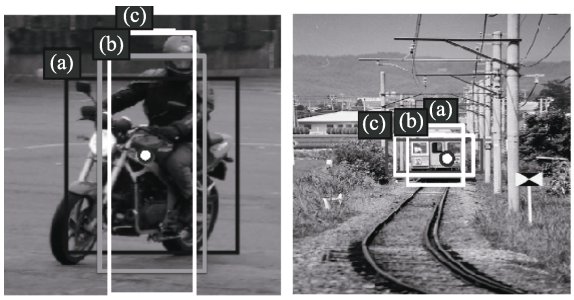
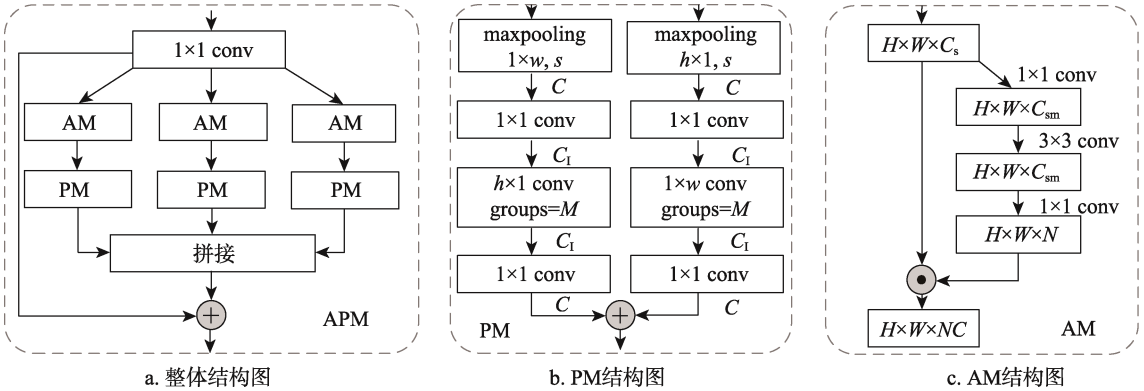


图 3　同一位置为不同锚点框提供相同的特征表达

(3) 当前一些针对单阶段目标检测算法的改进往往是以牺牲推测速度来改善检测精度. 如 RetinaNet 使用大尺寸的输入、复杂的主干网络以及分离的拥有多层结构的分类和回归子网络; 虽然其在检测精度上有所改善, 但失去了单阶段检测算法的速度优势.

2　本文算法

针对上文所述的现有基于锚点框的单阶段目标检测算法存在的问题, 本文提出一种基于注意力机制的部件感知模块(attentive part-aware module, APM), 结构如图 4a 所示. 该模块由部件感知模块 (part-aware module, PM) 和注意力模块 (attention module, AM)组成. PM 提取每个锚点框内具有判别性的部件特征作为每个锚点框独有的特征表达; 而 AM 则使得某一位置(或区域)根据锚点框预测目标的不同而提供不同的特征信息.



a. 整体结构图　　b. PM结构图　　c. AM结构图

图 4　本文所提设计的结构图

2.1 PM

为了使每个锚点框拥有单独的特征表达，同时所提取的特征能够考虑锚点框的尺寸，本文提出 PM. 给定一个以特征图 $\boldsymbol{X}\in\mathbb{R}^{H\times W\times C}$ 中任意位置(m, n)为中心的锚点框 $a^i_{m,n}\,(i\in[1, k])$，其设定的尺寸为 $h_i\times w_i$；将其划分为 $h_i\times w_i$ 的网格，划分的网格代表将该锚点框所包围物体划分为不同的部件，每个网格的尺寸为1×1，从而得到该锚点框所包围物体的部件特征图 $\boldsymbol{X}^i_{m,n}\in\mathbb{R}^{h_i\times w_i\times C}$. 为了减少无关信息的干扰，本文仅提取锚点框中包含物体判别性强的部件特征，即特征图 $\boldsymbol{X}^i_{m,n}$ 中每一行和每一列最大的特征值. 这样操作的原因在于：锚点框通常只能包含物体的一部分；所含物体至少有一个部件存在于部件特征图的每一行或者每一列中. 提取出的判别性部件特征经过加权融合，作为该锚点框的特有特征表达，用于后续分类和回归. 具体地，本文使用池化核为 $1\times w_i$ 和 $h_i\times 1$，步长为 s=1 的最大池化操作分别提取锚点框内每一行和每一列的判别性部件特征 $\boldsymbol{O}^i_{m,n}\in\mathbb{R}^{h_i\times 1\times C}$ 和 $\boldsymbol{P}^i_{m,n}\in\mathbb{R}^{1\times w_i\times C}$. 由于提取出的特征尺寸以及特征值的维度不同，不能直接进行融合；因此，本文使用一个“瓶颈层”分别对 $\boldsymbol{O}^i_{m,n}$ 和 $\boldsymbol{P}^i_{m,n}$ 的尺寸和维度进行归一化. 该瓶颈层首先使用卷积核为1×1的卷积操作将 $\boldsymbol{O}^i_{m,n}$ 和 $\boldsymbol{P}^i_{m,n}$ 的维度由 C 降低为 C_1，然后使用卷积核为 $h_i\times 1$ 和 $1\times w_i$ 的卷积归一化输出特征尺寸，并再次使用卷积核为1×1的卷积操作将特征维度由 C_1 升为 C 得到 $\hat{\boldsymbol{O}}^i_{m,n}\in\mathbb{R}^{1\times 1\times C}$ 和 $\hat{\boldsymbol{P}}^i_{m,n}\in\mathbb{R}^{1\times 1\times C}$. 该瓶颈层除了能够归一化特征的尺寸和维度，还能够进一步增强提取出的部件特征的表达能力. 经瓶颈层处理之后的特征相加，即得到锚点框 $a^i_{m,n}$ 的特征 $\boldsymbol{A}^i_{m,n}\in\mathbb{R}^{1\times 1\times C}$.

上述操作的形式化表达为

$$\begin{aligned}&\boldsymbol{O}^i_{m,n}=\max\nolimits_O(\boldsymbol{X}^i_{m,n}),\\&\hat{\boldsymbol{O}}^i_{m,n}=\boldsymbol{W}^O_u*(\boldsymbol{W}^O_m*(\boldsymbol{W}^O_d*(\boldsymbol{O}^i_{m,n})))\end{aligned}\tag{1}$$

$$\begin{aligned}&\boldsymbol{P}^i_{m,n}=\max\nolimits_P(\boldsymbol{X}^i_{m,n}),\\&\hat{\boldsymbol{P}}^i_{m,n}=\boldsymbol{W}^P_u*(\boldsymbol{W}^P_m*(\boldsymbol{W}^P_d*(\boldsymbol{P}^i_{m,n})))\end{aligned}\tag{2}$$

$$\boldsymbol{A}^i_{m,n}=\hat{\boldsymbol{O}}^i_{m,n}+\hat{\boldsymbol{P}}^i_{m,n}.$$

其中，$\max_O$ 和 $\max_P$ 表示最大池化，其池化核分别为 $1\times w_i$ 和 $h_i\times 1$；$*$ 表示卷积操作；卷积核 $\boldsymbol{W}^O_{\bullet}$ 和 $\boldsymbol{W}^P_{\bullet}$ 分别为 $\boldsymbol{W}^O_d\in\mathbb{R}^{C_1\times C\times 1\times 1}$，$\boldsymbol{W}^O_m\in\mathbb{R}^{C_1\times C_1\times h_i\times 1}$，$\boldsymbol{W}^O_u\in\mathbb{R}^{C\times C_1\times 1\times 1}$，$\boldsymbol{W}^P_d\in\mathbb{R}^{C_1\times C\times 1\times 1}$，$\boldsymbol{W}^P_d\in\mathbb{R}^{C_1\times C\times 1\times 1}$，$\boldsymbol{W}^P_m\in\mathbb{R}^{C_1\times C_1\times 1\times w_i}$，$\boldsymbol{W}^P_u\in\mathbb{R}^{C\times C_1\times 1\times 1}$. 为了简化表述，上述公式消去了偏置项. 实际上锚点框以每个特征图的位置为中心密集铺设，因此上述操作可以在特征图上通过共享参数的方式来高效地实现.

由于特征图每个位置关联了 k 个锚点框，每个空间位置都需要 k 次上述操作，造成很大的时间和空间开销. 因此，本文采用一种折中的方法实现上述操作：将预设的锚点框按照其长宽比分成 M 组，相同长宽比的属于同一组，每组有 k/M 个锚点框. 锚点框的部件特征图 $\boldsymbol{X}^i_{m,n}$ 的空间尺寸 $h_i\times w_i$ 设定为每组最小尺度的锚点框所对应的尺寸. 在瓶颈层中的标准卷积操作替换为组卷积[10]来近似地保存每个锚点框的判别性部件信息，最后将每个分组提取的特征相加. 因此式(1)(2)相应地修改为

$$\begin{aligned}&\boldsymbol{D}^i_{m,n}=\boldsymbol{W}^O_d*(\boldsymbol{O}^i_{m,n}),\\&\boldsymbol{G}^O=\boldsymbol{W}^O_1*\boldsymbol{D}_1\,\|\,\boldsymbol{W}^O_2*\boldsymbol{D}_2\,\|\cdots\|\,\boldsymbol{W}^O_M*\boldsymbol{D}_M,\\&\hat{\boldsymbol{O}}^i_{m,n}=\boldsymbol{W}^O_u*\boldsymbol{G}^O,\\&\boldsymbol{F}^i_{m,n}=\boldsymbol{W}^P_d*(\boldsymbol{P}^i_{m,n}),\\&\boldsymbol{G}^P=\boldsymbol{W}^P_1*\boldsymbol{F}_1\,\|\,\boldsymbol{W}^P_2*\boldsymbol{F}_2\,\|\cdots\|\,\boldsymbol{W}^P_M*\boldsymbol{F}_M,\\&\hat{\boldsymbol{P}}^i_{m,n}=\boldsymbol{W}^P_u*\boldsymbol{G}^P.\end{aligned}$$

其中，$\boldsymbol{D}^i_{m,n}=\boldsymbol{D}_1\cup\boldsymbol{D}_2\cup\cdots\cup\boldsymbol{D}_M$，$\boldsymbol{D}_{\bullet}\in\mathbb{R}^{h_i\times 1\times C_1/M}$；$\boldsymbol{W}^O_{\bullet}\in\mathbb{R}^{(C_1/M)\times(C_1/M)\times h_i\times 1}$；$\boldsymbol{F}^i_{m,n}=\boldsymbol{F}_1\cup\boldsymbol{F}_2\cup\cdots\cup\boldsymbol{F}_M$；$\boldsymbol{W}^P_{\bullet}\in\mathbb{R}^{(C_1/M)\times(C_1/M)\times 1\times w_i}$；$\|$ 表示拼接操作. 所提 PM 的具体结构如图 4b 所示.

本文所提 PM 类似于 RoI Pooling 算法[3,5]，但存在如下区别：(1) RoI Pooling 算法的输出特征尺寸相同(如都是 7×7 或 14×14)，与输入的感兴趣区域的大小无关；而本文算法提取的锚点框内部特征与锚点框的尺度和长宽比有关；(2) RoI Pooling 算法提取的特征是每个网格内最大特征或者平均特征，本文算法是提取锚点框内部每一行和每一列的最大值来表示判别性强的部件特征，能够更好地避免无关信息的干扰；(3) 本文算法中模块所需的存储和时间少于 RoI Pooling.

2.2 AM

本文的 AM 由 3 层简单的卷积网络组成. 给定一个特征图 $\boldsymbol{S}\in\mathbb{R}^{h_s\times w_s\times C_s}$，首先一个卷积核为1×1的卷积操作将 $\boldsymbol{S}$ 中的每个空间位置的特征值由 C_s 维嵌入到 C_{sm} 维，然后使用一个卷积核为3×3的卷积操作聚合每个空间位置周围的特征，以增强该位置特征的表达能力，最后使用一个1×1卷积操作生成每个空间位置对不同锚点框的特征表达的权

重 $\boldsymbol{\omega}\in\mathbb{R}^{h_s\times w_s\times C_t}$. C_t 表示包围某个位置(或者区域)的锚点框的数量. 在得到对每个锚点框的权重之后, 每个锚点框获得的差异化特征 $\boldsymbol{T}^i=\boldsymbol{S}\odot\boldsymbol{\omega}^i$, $i\in[1, C_t]$, $\boldsymbol{\omega}^i\in\mathbb{R}^{h_s\times w_s\times 1}$ 表示第 i 个锚点框的空间特征权重, $\boldsymbol{T}^i\in\mathbb{R}^{h_s\times w_s\times C_s}$ 为第 i 个锚点框获得的差异化特征图, $\odot$ 表示按元素相乘. 因此, 该模块输出的最终特征图为 $\boldsymbol{T}\in\mathbb{R}^{h_s\times w_s\times(C_s\times C_t)}$. 上述操作的形式化表达为

$$\boldsymbol{\omega}=\boldsymbol{W}^{\mathrm{t}}*(\boldsymbol{W}^{\mathrm{a}}*(\boldsymbol{W}^{\mathrm{s}}*\boldsymbol{S})),$$
$$\boldsymbol{T}=\boldsymbol{S}\odot\boldsymbol{\omega}.$$

其中, $\boldsymbol{W}^{\mathrm{s}}\in\mathbb{R}^{C_s\times C_{sm}\times 1\times 1}$; $\boldsymbol{W}^{\mathrm{a}}\in\mathbb{R}^{C_{sm}\times C_{sm}\times 3\times 3}$; $\boldsymbol{W}^{\mathrm{t}}\in\mathbb{R}^{C_t\times C_{sm}\times 1\times 1}$. 图 4c 所示为该模块的结构图.

由于每个空间位置对应锚点框的数量不同, 因此 C_t 会随着空间位置的不同而变化. 为了简化问题和操作, 本文设定每个空间位置的差异化特征表现只作用于以该位置为中心的锚点框, 即 $C_t=k$.

为了能得到更优的权重 $\boldsymbol{\omega}$, 本文采用初始化方法

$$\begin{cases}\boldsymbol{W}^{\mathrm{t}}(q)=\mu\\ \boldsymbol{b}^{\mathrm{t}}=\boldsymbol{1}\end{cases}.$$

其中, $\boldsymbol{W}^{\mathrm{t}}(q)$ 为 $\boldsymbol{W}^{\mathrm{t}}$ 中第 q 位置的权重值; $\mu\sim\mathcal{N}(0, \sigma^2)$, $\sigma\ll 1$, 本文中设定 σ = 0.001; $\boldsymbol{b}^{\mathrm{t}}\in\mathbb{R}^{C_t}$ 为偏置项. 根据上述初始化方法, 初始时某个空间位置的特征对于包围该位置的锚点框是相同的, 而随着训练的进行, 权重值根据锚点框的预测目标不同逐渐发生变化, 最终对不同锚点框表达出差异化特征.

2.3　网络结构

本文将 AM 与 PM 进行级联, 也即 AM 的输出作为 PM 的输入, 具体结构如图 4a 所示. 本文的模块具有如下的优点. (1) 简洁轻量. 上述模块在设计时采用降维、分组卷积等方式以减少总的参数和计算量. (2) 即插即用. 本文设计的模块可以用在不同的特征提取网络上. (3) 多样的特征表达. 相比于现有算法中锚点框共享同样的特征, 本文算法通过提取每个锚点框特有的特征以及每个空间位置对锚点框的差异化特征表现, 使得最终用于预测的特征更加多样性. 为了证明本文算法中模块的有效性, 将其部署在基于锚点框的单阶段目标检测算法 SSD 中.

2.3.1　SSD 算法

SSD 目标检测算法的网络结构如图 5 所示, 其以 VGG-16 网络作为特征提取网络. 使用 conv4_3, conv_7(FC7), conv8_2, conv9_2, conv10_2, conv11_2 等特征图构成的多层网络结构来检测多尺度的物体. 在 conv4_3~conv9_2 上分别设置了 6 种不同尺度和长宽比的锚点框, 在 conv10_2 和 conv11_2 分别设置了 4 种尺度和长宽比的锚点框. 训练时锚点框与真实物体边界框的交并比(intersection over union, IOU)大于 0.5 或是最大 IOU 将被标记为正例, 并赋予其相应真实物体边界框对应的物体类别及与真实物体边界框的位置和尺寸的偏差量; 然后采用损失函数

$$L(p,x,c,t)=\frac{1}{N}\left(\alpha_1 L_{\mathrm{cls}}(p,c)+\alpha_2 L_{\mathrm{r}}(x,t)\right)$$

进行训练. 其中, N 为样本数; c 代表锚点框匹配到的真实物体类别; t 为物体真实的边界框; p 为分类器预测的物体类别; x 为位置和尺寸的预测信息; L_{cls} 表示分类损失函数; L_{r} 表示预测框的位置和尺寸损失函数; α_1 和 α_2 分别为损失函数 L_{cls} 和 L_{r} 的权重.

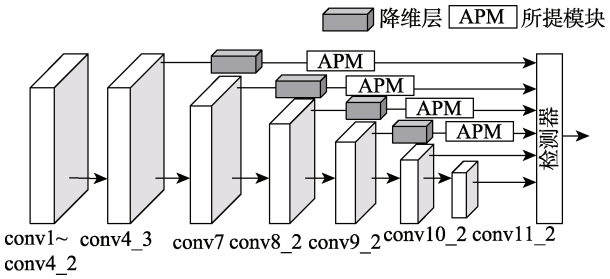


图 5　SSD 结构图(白色部分)及本文设计

2.3.2　与 SSD 算法的结合

本文将以代表性的单阶段检测算法 SSD 为例, 阐述如何将本文算法与单阶段检测算法进行结合. 本文的 APM 能够直接插入 SSD 用于预测的特征图, 如图 5 所示. 本文所采用的锚点框的尺寸设定规则与 SSD 相同, 因此可以将设定的锚点框按照其长宽比划分为 M=3 组, 每组锚点框的划分尺寸 $h_i\times w_i$ 相应地设置为 5×5, 3×7 和 7×3.

SSD 算法中用于预测的主干网络特征图维度很大, 例如, conv4_3 输出特征为 512 维, conv7 输出特征为 1 024 维, 直接将这些特征图输入到本文所提模块中会带来很大的时间和存储开销. 因此, 本文首先使用卷积核为1×1, 输出维度为 256 维的卷积操作进行降维(图 5 中“降维”部分). 主干特征提取网络往往能够提取到物体比较全局的特征, 而本文算法更偏重于每个锚点框局部的特征. 为

了使用于预测的特征的表达能力更强，在实现时将主干网络输出的信息与本文算法输出的特征相加构成一种残差结构，即每个锚点框的局部特征作为对全局特征的一种补充信息.

3 实验结果与分析

为验证本文算法的有效性，采用如表 1 所示的软硬件环境. 在训练时，使用在 ImageNet[21]上预训练好的 VGG-16 作为特征提取主干网络，使用动量大小为 0.9，权重衰减因子为 0.000 5 的随机梯度下降法(stochastic gradient descent, SGD)在目标检测数据集上精调. 同时也采用了难负样本挖掘以及数据增强策略(如左右翻转、输入图片放大缩小等). 一批训练数据包含 32 张图片，即批量大小设置为 32. 设定 $\alpha_1=\alpha_2=1$. 本文使用 2 种公开的目标检测数据集 PASCAL VOC 和 MS COCO；PASCAL VOC 有 20 个类别，MS COCO 包含 80 个类别.

表 1 实验所用环境

环境	参数
操作系统	Ubuntu 16.04 LTS
CPU	Intel i7-6850k
GPU	NVIDIA GTX 1 080 Ti(11G 显存)
CUDA&CuDNN	CUDA 9.0 & CuDNN 7.0

本文首先在 PASCAL VOC 上使用输入尺寸为 300×300 的图片进行实验，分析所使用的超参数以及每个子模块对检测精度的影响. 使用 PASCAL VOC 2007 与 PASCAL VOC 2012 的训练验证图片的合集作为训练集，在 PASCAL VOC 2007 测试集上进行测试. 初始学习率设为 4×10^{-3}，学习率在 150 和 200 个训练周期后依次降为 4×10^{-4} 和 4×10^{-5}，总的训练周期为 250 个. 测试时每张图片中预测的锚点框，其类别分数若低于 0.01 则会被过滤掉，然后对余下的锚点框使用阈值为 0.45 的非极大值抑制(non-maximum suppression, NMS)，过滤掉重复的检测结果，最终保留前 300 个检测框进行评估.

3.1 超参数的作用

在本文算法中使用了多个与降低特征维度有关的超参数，因此将首先评估这些超参数的变化对最终检测精度的影响. 表 2 所示为设置的 PM 中瓶颈层的维度 C_{I} 分别为 128, 64, 32, 16 时的检测结果，实验结果表明，设置 C_{I}=16 时，能取得最好的检测精度(mean average precision, mAP)，为 80.1%. 该设置下的参数量和计算量相比于其他设置都要小. 表 2 显示了 AM 中设置 C_{sm} 为 32 和 16 时的检测精度，可见，C_{sm}=16 比 C_{sm}=32 时精度高 0.5%. 为了减少注意力模块所输出的特征图的维度和该模块的计算复杂度，本文使用一个 1×1 的卷积操作将其输入的特征维度(256 维)降维到 C_{s}. 表 2 显示了在设置 C_{s} 为 64 维时，能够达到检测精度与参数量的均衡. 本文也评估了在 PM 中分组提取锚点框判别性的部件特征之后的融合操作(按元素加或者以通道为次序拼接)的影响. 如表 2 所示，按照通道拼接不仅带来很大的参数量，而且其性能也不及按元素加操作；按元素加操作可以使得不同锚点框提取的特征进行相互补充，进而带来更好的检测精度. 由表 2 也可以看出，得到最优检测精度时本文算法的参数量仅为 0.15 MB，只占主干网络(VGG-16, 138 MB)参数量的 0.11%.

表 2 超参数对检测性能和模块参数量的影响

参数	取值	检测精度/%	参数量/MB
C_{I}	128	79.3	0.40
	64	79.7	0.26
	32	79.9	0.18
	16	80.1	0.15
C_{sm}	32	79.6	0.17
	16	80.1	0.15
C_{s}	128	79.8	0.23
	64	80.1	0.15
融合方式	拼接	79.7	0.19
	累加	80.1	0.15

3.2 所提模块的有效性

本文针对基于锚点框的单阶段目标检测算法存在的问题，相应地设计了 APM，在此分析每个模块对检测精度的影响. 文献[14]所述 SSD 是由 Caffe[22]框架实现(表 3 中编号 a 项)，本文使用 Pytorch 框架重新实现 SSD，所得检测精度如表 3 中编号 b 项所示为 77.9%，比文献[14]高 0.7%，而时间却只有 12 ms，运行速度的改善可能是由于 Pytorch 库中相关操作(如卷积、池化等)的优化要优于 Caffe.

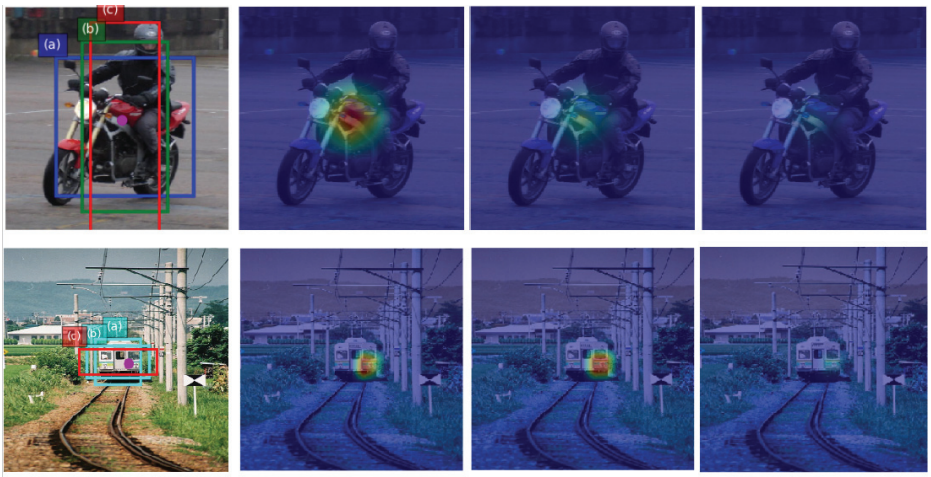
当只添加 PM 时，检测精度相对于基准提高了 1.6%，达到了 79.5% mAP(表 3 中编号 d 项). 说明在主干网络所提取锚点框全局特征的基础上，进一步补全每个锚点框的特有特征，能够在很大程度上丰富每个锚点框用于分类和回归的特征，从而产生更好的预测结果；而由于轻量化的设计，该模块仅带来约 1 ms 的时间开销. 在此基础上，级联的 AM，能够使最终的检测精度提高到 80.1%(表 3

中编号 e 项)，而整个模块只增加了 2 ms 左右的时间开销. AM 使得同一位置为不同目标的锚点框提供不同的特征表达，进一步增加了锚点框用于预测的特征的多样性.

表 3　本文算法的有效性验证结果

编号	网络结构	运行时间/ms	检测精度/%
a	SSD[14]	22	77.2
b	SSD(Pytorch)	12	77.9
c	SSD(更多层)	13	78.5
d	SSD(+PM)	13	79.5
e	SSD(+APM)	14	80.1
f	SSD(RoI Pooling)	18	79.1

图 6 显示了使用注意力机制时，某个空间位置(图中紫色的点)对不同锚点框特征表达的权重热点图，颜色的深浅代表权重值的大小. 从图中可以看出，空间中的点根据其所处位置，对有不同预测目标的锚点框所表现出的权重不同. 如图 6 上部紫色点处于摩托车的位置，它对类别为同一类摩托车的锚点框(a)的权重最大，为其提供更丰富的判别特征，对类别为人的锚点框(b)也能提供一定的上下文的信息；而对类别为背景的锚点框(c)则表现较低的权重即不提供信息. 即使是锚点框为相同的预测类别，由于锚点框预测所需的特征信息不同，也会有不同的权重，也即特征表现的不同，如图 6 第 2 行所示.



a. 输入　　b. 锚点框(a)权重图　　c. 锚点框(b)权重图　　d. 锚点框(c)权重图

图 6　同一位置对不同锚点框的特征表达的影响

本文算法中的模块相比于原有的结构[14]增加了网络的深度和参数量，因此需要检验检测精度的改善是否由深度或者参数量的增加带来的. 为了验证该猜想，本文在原来 SSD 结构中简单地增加与所提模块相同的层数及参数量，如表 3 中编号 c 项所示，这种设置相比于原有的 SSD 只增加了 0.6%检测精度，远远小于所提模块带来的性能改善(2.2%)，说明简单地增加深度或者参数量并不能够带来很大的性能改善. 表 3 中编号 f 项显示了直接使用 RoI Pooling 提取每个锚点框的特征时的运行时间和检测精度，与之相比，本文所提 PM **拥有更少的运行时间和更高的检测精度**.

3.3　与其他算法检测结果对比

本文进一步与当前具有代表性的性能优异的目标检测算法进行比较，如表 4 所示. 在 PASCAL VOC 2007 测试集上，使用图片尺寸为 300×300 作为输入时，本文算法以更快的速度获得了比其他单阶段目标检测算法更优的性能，其他单阶段检测更多地关注提高主干网络提取特征的丰富性，而忽略了锚点框在使用这些丰富特征的不足之处，虽然它们也能获得相对较好的精度，但是精度要弱于本文算法. RefineDet320[23]通过两步位置精调和分类获得与本文相当的精度，但是多步调整也降低了网络的运行速度. 本文算法虽然使用的输入尺寸很小，但是也超过了某些以大尺寸作为输入的两阶段目标检测算法，并且速度也更快.

当使用较大尺寸 384×384 为输入时，本文算法能够获得 mAP 为 81.0%，且速度依然能够保持实时的 62.5 fps. 本文算法能够保持如此快速度的原因，除了深度学习库 Pytorch 中相关操作的优化外，还在于如下几个方面. (1) 输入尺寸较小. 小尺寸输入可以使得网络的计算量更少. (2) 主干网络较浅. ResNet-101 和 ResNet-FPN 的网络结构都很深，相应地增加了网络计算时间. (3) 所提模块

表 4 与现有先进算法在 PASCAL VOC 上比较

算法	主干网络	输入尺寸	速度/ fps	VOC 2007	VOC 2012
Fast[6]	VGG-16	1 000×600	0.5	70.0	68.4
Faster[3]	VGG-16	1 000×600	7.0	73.2	70.4
Faster[9]	ResNet-101	1 000×600	2.0	76.4	73.8
R-FCN[24]	ResNet-101	1 000×600	9.0	80.5	77.6
MR-CNN[25]	VGG-16	1 000×600	0.1	78.2	73.9
YOLOv2[12]	DarkNet-19	544×544	40.0	78.6	73.4
RON384[26]	VGG-16	384×384	15.0	75.4	73.0
SSD300[14]	VGG-16	300×300	46.0	77.2	75.8
SSD512[14]	VGG-16	512×512	19.0	79.8	78.5
DSSD321[16]	ResNet-101	321×321	10.0	78.6	76.3
RefineDet320[23]	VGG-16	320×320	40.3	80.0	78.1
PFPNet-S300[27]	VGG-16	300×300	39.0	79.9	76.8
DFPR300[28]	VGG-16	300×300	39.5	79.6	77.5
本文 300	VGG-16	300×300	71.4	80.1	77.9
本文 384	VGG-16	384×384	62.5	81.0	79.1

结构简洁轻量. RefineDet320 使用了多步精调和较复杂的特征转移结构, PFPNet[27]构建了多层空间金字塔和层与层之间的特征融合, DFPR[28]使用多层特征融合和特征重分配到多层金字塔. 而 RetinaNet400 使用特征金字塔并拥有较重的且分离的回归和分类头部. 与上述结构相比, 本文算法涉及的参数量和计算量都比较少.

除了在 PASCAL VOC 2007 测试集上的结果, 表 4 也展示了在 PASCAL VOC 2012 测试集上的检测结果. 此结果是使用 PASCAL VOC 2007 和 2012 的训练验证集以及 2007 的测试集进行训练, 使用 2012 测试服务器得到的. 从表 4 可以看出, 本文算法仍然能获得优于现有先进的单阶段目标检测算法的检测精度.

3.4 MS COCO 实验结果分析

进一步地, 本文也在 MS COCO 上进行相应的实验. 在训练时使用常用的共有 11 万张图片的 trainval35k 作为训练集, 然后使用测试集 test-dev 进行测试并将预测结果提交至在线评估服务器得到检测精度. 训练时设置初始学习率为 2×10^{-3}, 在 90 和 120 个训练周期之后将学习率下降为 2×10^{-4} 和 2×10^{-5}, 总的训练周期为 140 个.

最终的检测结果如表 5 所示. 在使用输入图片尺寸为 300×300 时本文算法获得 mAP 为 30.0%, 超

表 5 与现有先进算法在 MS COCO 上的结果对比

算法	主干网络	输入尺寸	AP	AP50	AP75
Fast[6]	VGG-16	1000×600	19.7	35.9	
Faster[3]	VGG-16	1000×600	21.9	42.7	
R-FCN[24]	ResNet-101	1000×600	29.9	51.9	
FPN[7]	ResNet-101	1000×600	36.2	59.1	39.0
YOLOv2[12]	DarkNet-19	544×544	21.6	44.0	19.2
SSD300[14]	VGG-16	300×300	25.1	43.1	25.8
RON384[26]	VGG-16	384×384	27.4	49.5	27.1
DSSD321[16]	ResNet-101	321×321	28.0	46.1	29.2
DFPR300[28]	VGG-16	300×300	28.4	48.2	29.1
SSD512[14]	VGG-16	512×512	28.8	48.5	30.3
RefineDet320[23]	VGG-16	320×320	29.4	49.2	31.3
PFPNet-S300[27]	VGG-16	300×300	29.6	49.6	31.1
YOLOv3[29]	DarkNet-53	416×416	31.0	55.3	
DFPR512[28]	VGG-16	512×512	31.5	50.9	32.2
PFPNet-R320[27]	VGG-16	320×320	31.8	52.9	33.6
RetinaNet400[16]	ResNet-FPN	400×400	31.9	49.5	34.1
本文 300	VGG-16	300×300	30.0	49.2	31.7
本文 384	VGG-16	384×384	32.1	52.4	34.1

过了一些尺寸更大和使用性能更好的 ResNet-101 作为主干网络的检测算法; 在输入尺寸为 384×384 时也超过了当前性能最好的相似输入尺寸的 RetinaNet.

本文也统计了不同的目标检测算法在 MS COCO 测试集 test-dev 上的检测精度和测试时间的分布曲线. 从图 7 可以看出, 相比于已有的一些代表性检测算法, 在相似检测性能的情况下, 本文算法拥有更快的测试速度; 在相似测试速度的情况下, 本文算法能够获得较高的检测精度.

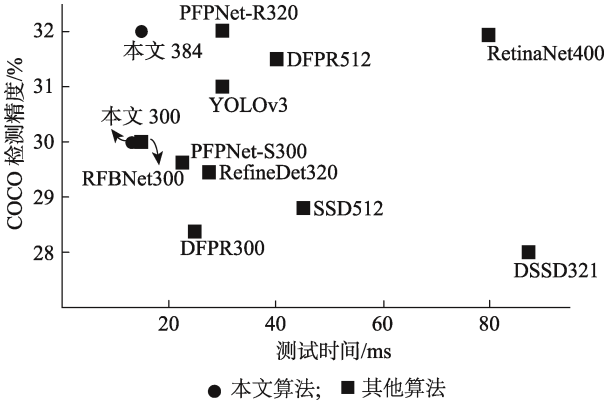


图 7　与现有算法的测试速度与检测精度对比

4　扩展实验

为了更好地验证本文算法的有效性, 除了上述基于锚点框的单阶段目标检测实验外, 还将其扩展到基于锚点框的 RPN. RPN[3]结构类似于图 1, 分类的输出不再是具体的物体类别, 而是二分类, 即锚点框所包含的是物体还是背景. 该实验在 PASCAL VOC 上使用以 ResNet-101 作为主干网络, conv4_23 的输出作为 RPN 子网络的输入. 与文献[7]类似, 在 MS COCO 上使用以 ResNet-50 构成的 FPN 作为 RPN 子网络的输入.

4.1　PASCAL VOC 性能对比

与上述单阶段目标检测算法相似, 仍然以 PASCAL VOC 2007 和 2012 的训练验证集作为训练集, 在 PASCAL VOC 2007 测试集上测试性能. 训练时保持输入图片的最小尺寸为 600 像素, 数据增强策略只采用左右翻转. 初始学习率设置为 0.001, 使用 SGD 训练 60 000 次迭代之后降为 0.000 1, 总迭代次数为 80 000 次. 测试时使用阈值为 0.7 的 NMS 过滤掉重复的区域建议框.

在 PASCAL VOC 上采用 2 种策略评估生成的区域建议框的性能, 并与相同实验设置和软硬件环境下的 RPN[3]的性能进行比对. 首先评估生成固定数量的区域建议框与真实边界框不同 IOU 时的召回率, 如表 6 所示. 表 6 中列出了生成 50, 100 和 300 个区域建议框时整体召回率和在 IOU=0.5, 0.6, 0.7, 0.8, 0.9 时的召回率(分别表示为表 6 中的 0.5, 0.6, 0.7, 0.8, 0.9). 从表 6 中可以得出以下结论: (1) 从整体上看, 本文算法生成的区域建议框要比 RPN 的召回率高, 即使是在生成的区域建议框数量很少的情况下, 如同样生成 50 个区域建议框, 本文算法比 RPN 召回率高 7.5%. (2) 在生成相同数目的区域建议框时, 本文算法相比于 RPN 的召回率改进随着 IOU 的增加而逐渐增加, 在 IOU=0.8 时召回率的提高最明显, 即使是 IOU=0.9 时也能提高大约 7.5%, 说明本文算法生成的区域建议框相比于 RPN 定位性能要好. (3) 不管生成区域建议框的数目为多少, 算法的召回率随着 IOU 的增加而逐渐下降, 在 IOU=0.5 时的召回率最高. 造成这种现象的一种可能原因是, 在训练采样正样本时采用的标准较低, 即锚点框与真实边界框的 IOU≥0.5 就标记其为正样本, 这使得最终训练的网络所生成的区域建议框更偏向于该 IOU.

表 6　生成固定数量的区域建议框时在不同 IOU 阈值下的召回率

策略	区域建议框数量为 50						区域建议框数量为 100						区域建议框数量为 300					
	平均	0.5	0.6	0.7	0.8	0.9	平均	0.5	0.6	0.7	0.8	0.9	平均	0.5	0.6	0.7	0.8	0.9
RPN	54.1	88.6	83.7	72.5	42.5	5.8	57.4	92.6	88.3	77.8	45.9	6.0	60.9	96.5	93.2	83.7	49.1	6.3
本文	61.6	92.4	88.5	80.3	57.4	13.4	63.8	94.9	91.5	84.0	60.0	13.6	65.9	97.1	94.3	87.4	62.6	13.8
Δ	+7.5	+3.8	+4.8	+7.8	+14.9	+7.6	+6.4	+2.3	+3.2	+6.2	+14.1	+7.6	+5.0	+0.6	+1.1	+3.7	+13.5	+7.5

其次, 评估在固定 IOU 值时不同数量的区域建议框的召回率, 如表 7 所示, 本文主要评估在 IOU 为 0.5, 0.7 和 0.8 时分别生成 10, 100, 300 及 1000(1k)个区域建议框时的召回率(average recall, AR)(分别表示为 AR_10, AR_100, AR_300, AR_1k). 从表 7 中可以看出: (1) 同一 IOU 阈值下, 随着生成区域建议框的增加召回率逐渐增加. (2) 同一 IOU 阈值下, 本文算法随着生成区域建议框数量

表 7 在固定 IOU 阈值条件下生成不同数量的区域建议框的召回率

策略	IOU=0.5				IOU=0.7				IOU=0.8			
	AR_10	AR_100	AR_300	AR_1k	AR_10	AR_100	AR_300	AR_1k	AR_10	AR_100	AR_300	AR_1k
RPN	71.8	92.6	96.5	97.9	55.5	77.8	83.7	86.0	31.5	45.9	49.1	50.4
本文	80.3	94.9	97.1	97.9	66.4	84.0	87.4	88.8	47.1	60.0	62.6	63.3
Δ	+8.5	+2.3	+0.6	+0.0	+10.9	+6.2	+3.7	+2.8	+15.6	+14.1	+13.5	+12.9

的增加, 相比于 RPN 的改进逐渐减少. 例如, 在 IOU=0.5 时, RPN 的召回率在区域建议框数量为 300 和 1k 时与本文算法相似. (3) 在高 IOU 阈值下, 随着区域建议框数量的增加本文算法仍然比 RPN 获得更高的召回率, 尤其是 IOU=0.8 时, 说明本文算法比 RPN 生成的区域建议框质量高.

4.2 MS COCO 检测性能对比

在该数据集上仍然采用 trainval35k 作为训练集, 由于 test-dev 无公开的真实边界框标记, 因此使用 minival 作为测试集来评估区域建议框的性能. 实验中使用 ResNet-50 为主干网络的 FPN 结构构建 RPN, 并采用 MS COCO 的评估标准, 即在生成数量为 10, 100, 1 000(1k)时的召回率(分别表示为 AR^{10}, AR^{100}, AR^{1k})以及在生成数量为 1k 时小尺寸(AR_s^{1k}), 中等尺寸(AR_m^{1k})和大尺寸(AR_l^{1k})的召回率. 以目前性能较优的文献[7]作为对比算法, 训练时输入图片的最短边为 800 像素, 每张图片有 256 个锚点框参与训练, 初始学习率为 0.006 5, 在 320 000 次迭代之后学习率降为 0.000 65, 最终训练 480 000 次迭代, 批量大小设置为 6. 实验结果如表 8 所示. 从表 8 可以看出, 相比于文献[7]算法(表示为 RPN-FPN), 本文算法在不同指标下获得了更好的召回率. RPN-FPN* 是使用开源的代码库 Detectron[30]和训练好的模型在本文的实验环境下测试得到的, 该结果比文献[7]有所改善. 但是在区域建议框数量仅为 10 个时, 本文算法提高了 13.6%的召回率; 而且在生成 1k 个区域框时, 大尺寸物体召回率也获得了很显著的提高. 这是由于本文算法提取了更多与大尺寸物体相匹配的特征. MS COCO 中的小物体尺寸较小, 在提取锚点框特有的特征时可能也会引入一些无关的信息, 因此本文算法在小尺寸物体上的召回率改进仅有 3.7%.

表 8 在 MS COCO 上的区域建议框召回率对比

算法	AR^{10}	AR^{100}	AR^{1k}	AR_s^{1k}	AR_m^{1k}	AR_l^{1k}
RPN-FPN[7]		44.0	56.3	44.9	63.4	66.2
RPN-FPN*[30]	19.6	43.5	57.3	46.5	64.6	65.8
本文	33.2	53.4	62.4	50.2	69.0	74.5

相应地, 还测试了使用整合本文所提模块生成的区域建议框的两阶段目标检测算法与相同条件下使用 RPN[3]生成的区域建议框的两阶段目标检测算法的检测精度, 结果如表 9 所示. 其中算法的主干网络为 ResNet-101, -DCN 表示在主干网络中的 stage5 加了 DCN 并使用单层的检测子网络, -FPN 表示使用 FPN 的结构. 从表 9 可以看出, 使用本文模块生成的区域建议框也有助于两阶段检测算法的精度改善. 检测精度的改善是因为本文算法生成的区域建议框的质量比 RPN 生成的区域建议框高, 从而在提取建议框内部的特征时能获得更加对齐、高质量的特征, 分类器和回归器能够被更好地训练, 进而获得更高的检测精度. 这也证明本文算法的有效性.

表 9 不同算法生成的区域建议框在 MS COCO 上的目标检测精度对比

算法	AP	AP_50	AP_75	AP_s	AP_m	AP_l
R-FCN[24]	29.2	51.5		10.3	32.4	43.3
FPN[7]	36.2	59.1	39.0	18.2	39.0	48.2
FPN*[30]	39.4	61.5	42.8	22.7	42.1	49.9
Faster-DCN[8]	33.1	50.3		11.6	34.9	51.2
RFCN-DCN[8]	34.5	55.0		14.0	37.7	50.3
Mask R-CNN[31]	39.8	62.3	43.4	22.1	43.2	51.2
本文-DCN	37.4	57.7	41.1	15.5	41.4	54.1
本文-FPN	41.5	62.2	45.4	26.2	44.2	51.8

Light-Head R-CNN[32]通过采用轻量级的主干网络 xception 和精简的 R-CNN 获得了较快的检测速度, 大尺寸的输入使其获得较高的检测精度, 如表 10 所示. 虽然本文仅使用小尺寸的输入获得了比其更高的检测精度, 但是采用的主干网络 VGG-16 拥有较大的计算量, 导致检测速度有所降低. 为了更好地验证本文算法的泛化能力, 将 Light-Head R-CNN 采用的 RPN 替换为本文所设计的 RPN, 其他设置保持不变, 最终使得 Light-Head R-CNN 保持速度不变的情况下获得了更高的检测精度(如表 10 最后 2 行 Light Head R-CNN* 所示).

表 10　与 Light-Head R-CNN 的检测速度和精度对比

算法	主干网络	输入尺寸	速度/fps	检测精度/%
Light-Head R-CNN	xception	1 200×800	95.0	31.5
Light-Head R-CNN	xception	1100×700	102.0	30.7
本文	VGG-16	300×300	71.0	30.0
本文	VGG-16	384×384	62.5	32.1
Light-Head R-CNN*	xception	1 200×800	95.0	33.0
Light-Head R-CNN*	xception	1100×700	102.0	31.2

5　结　论

深度学习技术的发展极大地促进了目标检测的进步. 本文发现影响单阶段目标检测精度的原因主要在于同一中心位置的锚点框共享同样的特征，而且同一位置对不同预测目标的锚点框提供相同的特征表达. 因此，本文提出了 APM 以解决上述问题，进而提高单阶段目标检测的精度. 本文设计了 PM，为不同的锚点框提供与其尺寸相关的特征，以增强锚点框特征的多样性；同时还提出了 AM，使得特征图上的某位置为包围其的锚点框提供自适应的特征表达. 相比于现有算法在改进单阶段目标检测精度时往往牺牲检测速度，在公开的目标检测数据集上的实验结果表明，本文算法在显著地改进 SSD 算法的检测精度同时，能够维持其实时的运行速度. 扩展实验进一步展示了本文所提模块能够改进基于锚点框的区域建议框的召回率及相应的两阶段目标检测算法的检测精度. 在未来工作中，将考虑在计算资源有限的嵌入式和移动设备上应用该单阶段目标检测算法.

参考文献(References):

[1] Uijlings J R R, van de Sande K E A, Gevers T, *et al*. Selective search for object recognition[J]. International Journal of Computer Vision, 2013, 104(2): 154-171

[2] Lawrence Zitnick C, Dollár P. Edge boxes: locating object proposals from edges[C] //Proceedings of the European Conference on Computer Vision. Heidelberg: Springer, 2014: 391-405

[3] Ren S Q, He K M, Girshick R, *et al*. Faster R-CNN: towards real-time object detection with region proposal networks[C] // Proceedings of the Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2015: 91-99

[4] Girshick R, Donahue J, Darrell T, *et al*. Rich feature hierarchies for accurate object detection and semantic segmentation[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2014: 580-587

[5] He K M, Zhang X Y, Ren S Q, *et al*. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(9): 1904-1916

[6] Girshick R. Fast R-CNN[C] //Proceedings of the IEEE International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2015: 1440-1448

[7] Lin T Y, Dollár P, Girshick R, *et al*. Feature pyramid networks for object detection[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2017: 2117-2125

[8] Dai J F, Qi H Z, Xiong Y W, *et al*. Deformable convolutional networks[C] //Proceedings of the IEEE International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2017: 764-773

[9] He K M, Zhang X Y, Ren S Q, *et al*. Deep residual learning for image recognition[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2016: 770-778

[10] Xie S N, Girshick R, Dollár P, *et al*. Aggregated residual transformations for deep neural networks[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2017: 1492-1500

[11] Redmon J, Divvala S, Girshick R, *et al*. You only look once: unified, real-time object detection[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2016: 779-788

[12] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2017: 7263-7271

[13] Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift[OL]. [2019-08-14]. https://arxiv.org/abs/1502.03167

[14] Liu W, Anguelov D, Erhan D, *et al*. SSD: single shot multibox detector[C] //Proceedings of the European Conference on Computer Vision. Heidelberg: Springer, 2016: 21-37

[15] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[OL]. [2019-08-14] https://arxiv.org/abs/1409.1556

[16] Fu C Y, Liu W, Ranga A, *et al*. DSSD: deconvolutional single shot detector[OL]. [2019-08-14]. https://arxiv.org/abs/1701.06659

[17] Lin T Y, Goyal P, Girshick R, *et al*. Focal loss for dense object detection[C] //Proceedings of the IEEE International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2017: 2980-2988

[18] Everingham M, van Gool L, Williams C K I, *et al*. The pascal visual object classes (VOC) challenge[J]. International Journal of Computer Vision, 2010, 88(2): 303-338

[19] Lin T Y, Maire M, Belongie S, *et al*. Microsoft COCO: common objects in context[C] //Proceedings of the European Conference on Computer Vision. Aire-la-Ville: Eurographics Association Press Heidelberg: Springer, 2014: 740-755

[20] Zhou B L, Khosla A, Lapedriza A, *et al*. Object detectors emerge in deep scene CNNs[OL]. [2019-08-14]. https://arxiv.org/abs/1412.6856

[21] Russakovsky O, Deng J, Su H, *et al*. ImageNet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252

[22] Jia Y Q, Shelhamer E, Donahue J, *et al*. Caffe: convolutional architecture for fast feature embedding[C] //Proceedings of the 22nd ACM International Conference on Multimedia. New York: ACM Press, 2014: 675-678

[23] Zhang S F, Wen L Y, Bian X, *et al*. Single-shot refinement neural network for object detection[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2018: 4203-4212

[24] Dai J F, Li Y, He K M, *et al*. R-FCN: object detection via region-based fully convolutional networks[C] //Proceedings of the Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2016: 379-387

[25] Gidaris S, Komodakis N. Object detection via a multi-region and semantic segmentation-aware CNN model[C] //Proceedings of the IEEE International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2015: 1134-1142

[26] Kong T, Sun F C, Yao A B, *et al*. RON: reverse connection with objectness prior networks for object detection[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2017: 5936-5944

[27] Kim S W, Kook H K, Sun J Y, *et al*. Parallel feature pyramid network for object detection[C] //Proceedings of the European Conference on Computer Vision. Heidelberg: Springer, 2018: 234-250

[28] Kong T, Sun F C, Huang W B, *et al*. Deep feature pyramid reconfiguration for object detection[C] //Proceedings of the European Conference on Computer Vision. Heidelberg: Springer, 2018: 169-185

[29] Redmon J, Farhadi A. YOLOv3: an incremental improvement[OL]. [2019-08-14]. https://arxiv.org/abs/1804.02767

[30] Girshick R, Radosavovic I, Gkioxari G, *et al*. Detectron[OL]. [2019-08-14]. https://github.com/facebookresearch/Detectron

[31] He K M, Gkioxari G, Dollár P, *et al*. Mask R-CNN[C] // Proceedings of the IEEE International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2017: 2961-2969

[32] Li Z M, Peng C, Yu G, *et al*. Light-Head R-CNN: in defense of two-stage object detector[OL]. [2019-08-14]. https://arxiv.org/abs/1711.07264