华中科技大学

计算机视觉结课报告

姓　　名：

专　　业：

班　　级：

学　　号：

指导教师：

分数	
教师签名	

年月日

目录

1. 目标检测专题报告

1.1. 简介

目标检测是计算机视觉的基本任务之一，十分具有挑战性，旨在从图像中检测出物体实例，即用矩形框定位某些类别的物体。不同于最基本的图像分类任务，目标检测是一个多任务问题，包括定位和分类，其主要难点在于定位问题。

近 20 年来，目标检测领域收获越来越多的关注，并有了很多的突破和很大的发展。其大致的发展脉络如图 1 所示，该图包含了目标检测的经典、里程碑算法。

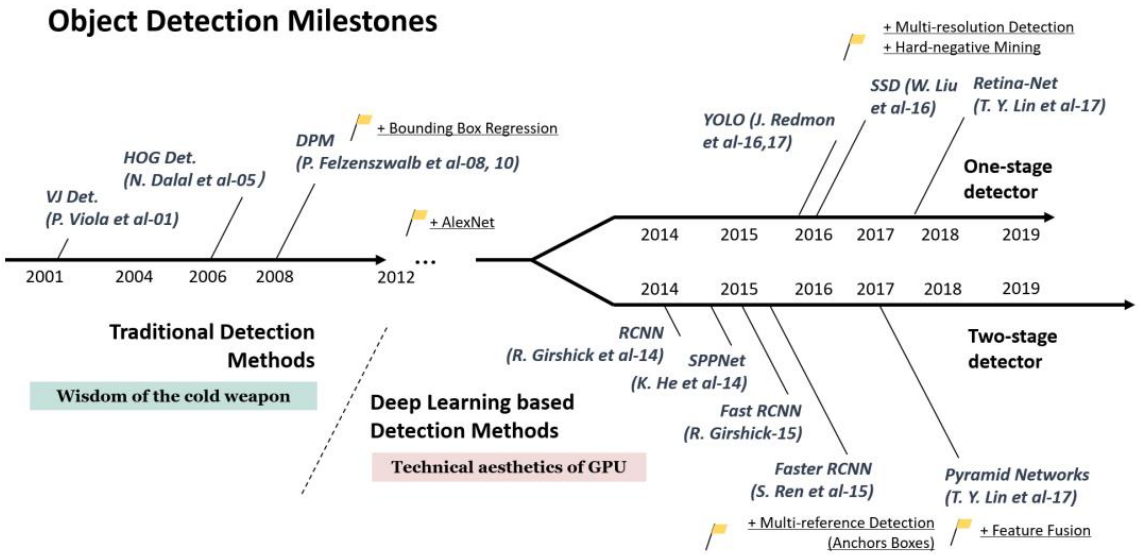


图 1：目标检测发展脉络（引用自[1]）

由上图可知：目标检测算法可分为传统检测算法和基于深度学习的检测算法，如今基于深度学习的检测算法已经成为主导。而基于深度学习的检测算法中的 Anchor-Based 方法又分为一阶段和两阶段。本次专题实验报告是基于 Anchor-Based 的两阶段目标检测算法，主要包含 R-CNN[2]、SPPNet[3]、Fast R-CNN[4]、Faster R-CNN[5]以及 Mask R-CNN[6]。

1.2. 两阶段目标检测经典算法

本小节主要介绍几个 Anchor-Based 的两阶段目标检测算法的基本原理，具体如下：

1.2.1. RCNN

1、基本原理：

首先，通过 selective search[7]对每张图片生成指定数量的 region proposals（候选区域）。然后，将每个 region proposal 缩放至固定大小的图片，并输入 CNN 网络提取特征。再用一系列二分类 SVM 进行分类，其中每个类别对应一个 SVM。最后，通过线性回归模型对边界框进行修正，其中对于每个类别都会产生 4 个修正参数。测试时，对同一张图片的 region proposals 进行非极大抑制 nms，得到最终的该图片的目标检测结果。网络结构如图 2 所示。

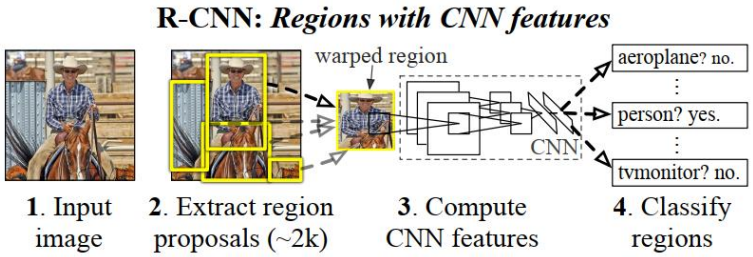


图 2：RCNN 网络结构（引用自[2]）

2、不足：

RCNN 虽然使用深度卷积网络在目标检测上取得了很大进展，但是有几点明显的缺点，如下：

（1）整个训练过程是多阶段的，十分繁琐，包括微调提取特征的卷积网络、训练多个 SVM 和修正边界框；

（2）由于是多阶段的，且前一阶段的输出结果需要作为后一阶段的输入，所以中间要将 region proposals 分为正负样本，联合提取的特征，作为下一阶段的数据集存入磁盘中，占用很大的空间。

（3）对来自同一张图片的上千个 region proposals，RCNN 重复通过深度卷积网络提取它们的特征，尽管许多 region proposals 有很多的重叠。因此导致检测速度很慢。

1.2.2. SPPNet

1、基本原理：

引入空间金字塔池化层 Spatial Pyramid Pooling (SPP) layer，从而使 CNN 的输入可以是任意尺寸的图像，而保证输出维度固定。其结构和原理如图所示。

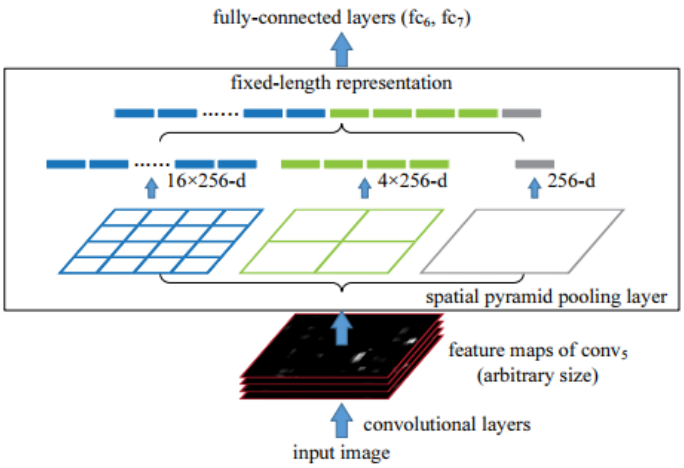


图 3：SPP layer 的结构（引用自[3]）

设输入 SPP layer 的特征图维度为(H, W, C)，其中 C 是常数，不随着输入图像大小变化而变化。以上图为例，SPP layer 的金字塔有 3 层，分别是 4x4、2x2、1x1。以最左边的 4x4 网格为例，先把特征图划分为 4x4 网格，每个网格的大小为(H/4, W/4, C)，称为 spatial bins，对每个 bin 逐通道做 max pooling，得到一个 C 维的向量，金字塔中的其他层做同样的操作，再将所有 bin 池化后的向量拼接在一起。最终，SPP layer 的输出为 CxM（M 为 bin 的个数，是预先定义好的）维向量，维数固定。

2、技术改进：

当 SPPNet 应用于目标检测时，对每张图片只提取一次特征，得到特征图。然后对于在原始图片上使用 selective search 生成的 region proposals，将其映射到该图片特征图的对应区域，得到该 region 的特征图。最后将这些不同大小的区域特征图通过 SPP layer，得到固定维度的特征向量。后续算法和 RCNN[2]基本一样。

由于同一张图片的所有 region proposals 共享只计算了一次的特征图，避免了像 RCNN 中那样重复计算特征，目标检测在不损失准确度的前提下，速度有

了巨大的提升。

3、不足：

（1）和 RCNN[2]一样，SPPNet 的训练过程依然是多阶段的，十分繁琐，包括微调提取特征的卷积网络、训练多个 SVM 和修正边界框。并且中间过程的特征图也需要存入磁盘中，占用很大的空间。

（2）SPPNet[3]中提出的微调算法无法微调 spp layer 之前的卷积网络，从而限制了网络的性能。

1.2.3 Fast RCNN

1、基本原理：

Fast RCNN[4]在 RCNN[2]和 SPPNet[3]的基础上进行改进，将一整张图片和生成的一系列 object proposals（也称作 RoI）作为输入，整个网络架构如图 4 所示。

首先，通过全卷积网络处理原始图片，产生特征图，和 SPPNet 一样，Fast RCNN 也只计算一次特征图。对于每个 RoI，将其投影到该图片的特征图上，提取出对应的部分，从而得到该 RoI 对应的特征图。然后，通过 RoI 池化层将所有 RoI 提取为固定维度的特征向量，RoI pooling layer 其实是 spatial pyramid pooling layer 的一种特殊情况。特征向量在经过几个全连接层后，进入 2 个不同的输出分支：1 个分支进行分类任务，通过 softmax 计算一个 RoI 属于每一类（K 个物体类别+背景类）的概率；另一个分支则进行回归任务，修正边界框。整个网络通过定义多任务损失，可以端到端的训练。

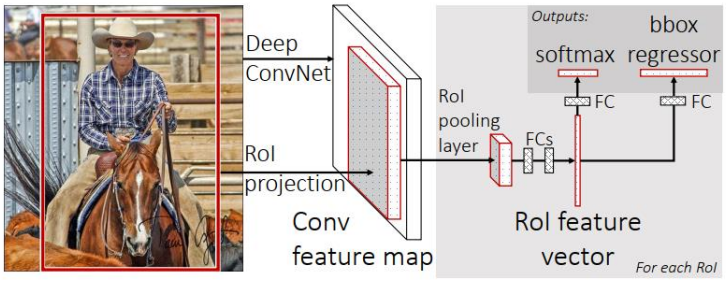


图 4：Fast RCNN 架构（引用自[4]）

2、技术改进：

相较于 RCNN 和 SPPNet，Fast RCNN 有如下技术改进：

（1）通过多任务损失函数，Fast RCNN 将训练过程综合为一个阶段，可以端到端的训练，因此也不用将特征图暂存入磁盘中；

（2）解决了 RoI pooling layer 的反向传播问题，所以训练时可以更新网络中所有层。没有了 SPPNet 中无法更新 spatial pyramid pooling layer 之前的卷积网络的限制，Fast RCNN 的目标检测精度有了很大的提升；

（3）和 SPPNet 一样，每张图片只计算一次特征，检测速度比 RCNN 快了 200 多倍。

3、不足：

虽然 Fast RCNN 成功地综合了 RCNN 和 SPPNet 并进行改进，但是其检测速度受到了候选区域生成算法 selective search[7]的限制。因为和高效的目标检测网络[4]相比，selective search 算法慢了一个数量级，所以候选区域的生成算法成为了提升检测速度的瓶颈。

1.2.4 Faster RCNN

1、基本原理：

Faster RCNN[5]提出了可以端到端训练的 Region Proposal Network (RPN)，用于生成高质量的 region proposals，其时间开销几乎可以忽略。RPN 结构如图 5 所示：

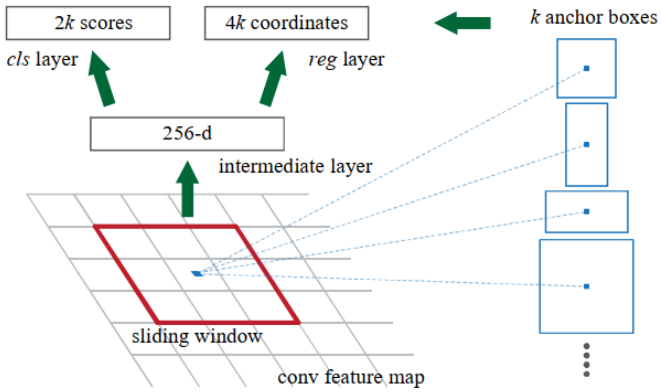


图 5：RPN 结构（引用自[5]）

RPN 共享目标检测模型（如 Fast RCNN[4]）中的卷积特征图。它以卷积特征图上的一个$n \times n$大小的窗口作为输入，也就是说以一个$n \times n$大小的滑动窗口在卷积特征图上滑动，每个窗口内的卷积特征作为 RPN 的输入。然后每个滑动窗口被映射为一个低维的特征向量，再进入两个不同的全连接层分支：一个

分支进行二分类（物体或背景类别），另一个分支进行边界框修正。所以 RPN 的实现也很简单，一个$n \times n$的卷积层，加上两个1×1的卷积层分支。

在 RPN 的每个滑动窗口位置，同时预测 k 个 region proposals，所以分类分支输出 2k 个分数，边界框修正分支输出 4k 个值。k 个 proposals 被参数化为 k 个参考框，叫作 anchor。anchor 的中心就是滑动窗口的中心，每个滑动窗口处的 k 个 anchors 是根据预先定义的 a 个不同的尺度大小和 b 个不同的长宽比生成的，$a \times b = k$。对于一个大小为$W \times H$的卷积特征图，共有$W \times H \times k$个 anchors。

通过训练，RPN 可以更快速地生成高质量的 region proposals，因为它可以告诉模型应该关注图片中的哪些部分，就像注意力机制。

把 RPN 和 Fast RCNN 的检测器结合成一个统一的网络，就形成了 Faster RCNN，如图 6 所示。

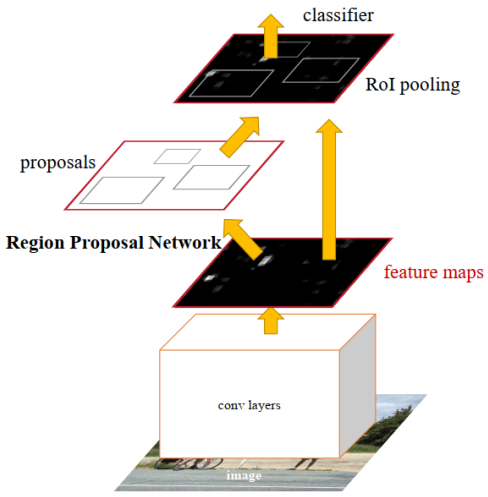


图 6：Faster RCNN 架构（引用自[5]）

2、技术改进：

在 Fast RCNN[4]的基础上，提出了新的 region proposals 生成算法——RPN，而不再使用流行的 selective search[7]算法。

由于 RPN 是全卷积网络，且可以通过学习生成高质量的 region proposals，所以在测试时，后续网络只需检测其生成的 300 个最好的 RoIs。因此，Faster RCNN 再一次大大提高了检测速度和精度，是第一个最接近实时的目标检测算法。

3、不足：

（1）Faster RCNN 在后续的检测阶段中仍存在一些计算冗余；

（2）对 anchors 进行正负样本标记分类时，如果 IOU 阈值设置得低，可能会引起噪声检测的问题；如果 IOU 设置得高，可能会引起过拟合。

（3）为了训练方便，当前的主流训练方法采用的是 Faster RCNN[5]中的提出的近似联合训练。但是这种方法忽略了 RoI pooling layer 对于 RPN 生成的边界框坐标的梯度，也就是说反向传播并不会从 RoI pooling layer 到 RPN。

1.2.5 Mask RCNN

1、基本原理：

Mask RCNN[6]主要在 Faster RCNN[5]的基础上进行拓展，使得目标检测和实例分割可以在同一个网络中同时进行。

在检测器头部增加一个新的分支，用于预测 RoIs 的 segmentation masks，与原本的分类分支和边界框修正回归分支平行，如图 7 所示。

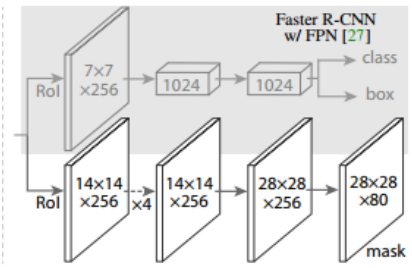


图 7：Mask RCNN 检测器头部结构（引用自[6]）

mask 分支通过一系列卷积和反卷积层，将 RoI 特征图转换为$K \times m \times m$维度的输出，即 K 个$m \times m$的二值 mask，对应 K 个物体类别。由于新增分支，多任务损失函数也需增加 mask 分支的损失函数L_{mask}，根据分类分支预测的类别 k，L_{mask}定义在第 k 个 mask 上，先逐像素 sigmoid，然后计算平均交叉熵损失。

因为分割任务是逐像素分类，所以需要 RoI 特征和 RPN 输出的 region proposals 之间的像素能够很好的对应匹配上。但是原先的 RoI Pool 操作在由原图映射到特征图、池化 RoI 特征时均进行了量化，这些量化操作带来了很大的像素误差，导致池化后的 RoI 特征和原始 RoI 不匹配。为了解决上述问题，Mask RCNN 提出了 RoIAlign layer，替换原先的 RoI Pooling layer，具体见图 8。

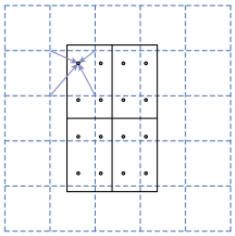


图 8：RoIAlign layer 示意图

如图 8 所示：RoIAlign 不再对 RoI 边界框和 bin 做量化取整操作，而是保持浮点数。图中蓝色虚线是卷积特征图，黑色实线是 RoI，被划分为2×2的 bins，每个 bin 有 4 个采样点，每个采样点位置的像素值通过双线性插值计算得到。然后对每个 bin 进行池化操作。

综上所述，Mask RCNN 整体框架如所示。

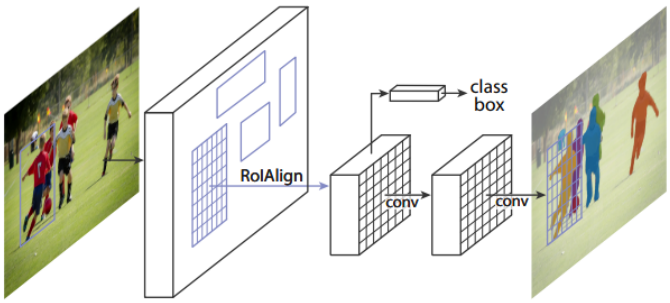


图 9：Mask RCNN 整体框架（引用自[6]）

2、技术改进：

相比于 Faster RCNN[5]，Mask RCNN 主要有以下改进：

（1）增加与分类分支和回归分支相平行的 mask 分支，使得网络可以同时进行目标检测和实例分割。并且由于 mask 分支是全卷积的，计算效率很高，所以检测和分割速度仍然很快，训练过程也很简便；

（2）用 RoIAlign layer 替换 RoI Pooling layer，解决了大感受野带来的像素误差的问题，减少了输入和输出的误差，从而提高了目标检测和实例分割的精度。

3、不足：

Mask RCNN [6]中的 mask 分支主要是通过一系列非常简单的卷积和反卷积操作进行分割，没有过多的投入和设计以提高分割精度，当然这也不是原论文的重点。

我认为可以参考一些图像分割网络，如 FCN[8]、U-Net[9]等，设计更复杂、效果更好的 mask 分支，提高分割精度，如在上采样时融合之前下采样时的低维、高像素的特征图等等

1.3. 代码运行

本专题实验中，我利用 MMDetection 工具箱，下载并运行了 Faster RCNN[5]和 Mask RCNN[6]的源代码，在 coco 测试集上进行测试和可视化。由于 MMDetection 项目中对于 Faster RCNN 和 Mask RCNN 的实现做了改进，所以性能要比原论文中的数据更好。

具体见此 github 仓库，包括环境配置、数据集和预训练文件下载、脚本和部分代码编写。

代码运行环境如下：

1、操作系统：Ubuntu20.04

2、CUDA 版本：11.6

3、GPU 型号：1 个 TITAN X

1.3.1. Faster RCNN

我使用的 Faster RCNN 的配置为：pytorch 实现、backbone 为 ResNet-50[10]、Neck 模型为 FPN[11]。

1、性能测试结果如图 10、图 11 所示：

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.374
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.581
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.404
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.212
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.410
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.481
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.517
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.517
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.517
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.326
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.557
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.648
```

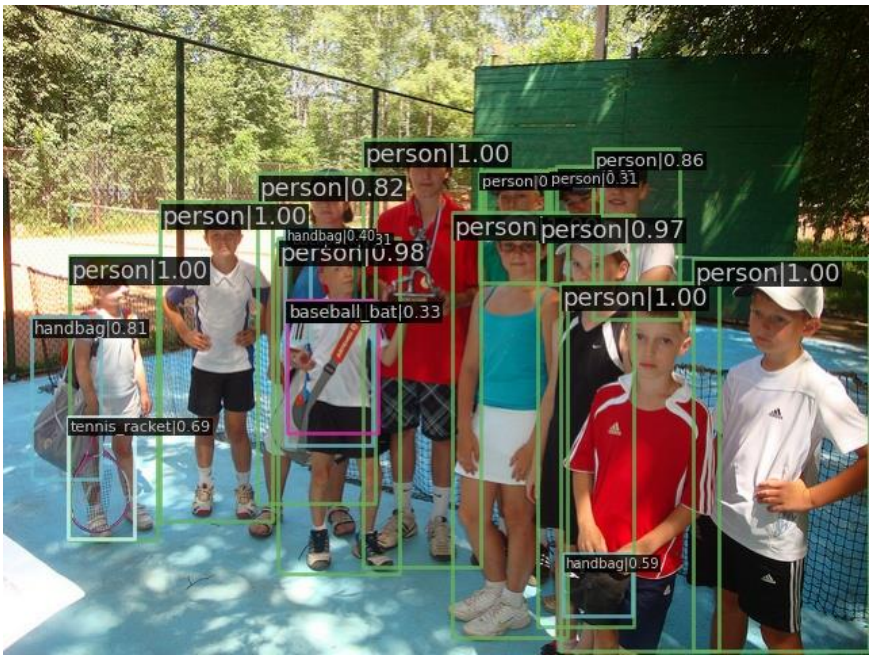
图 10：Faster RCNN bbox 测试结果

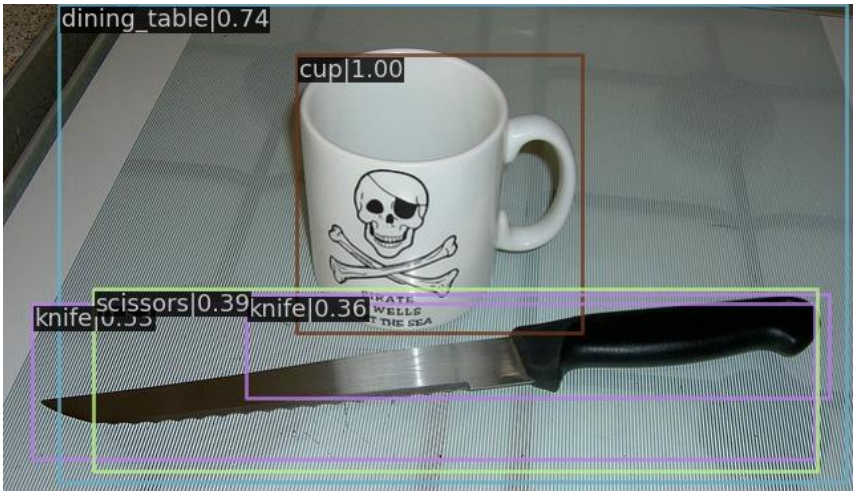
```
Overall fps: 15.1 img / s, times per image: 66.3 ms / img
```

图 11：Faster RCNN 推理速度测试结果

由图可知：Faster RCNN 的 bbox mAP 为 0.374，推理速度为 15.1fps。

2、在 coco 测试集上随机选取一些图片，可视化结果如图 12 所示：


person|1.00
person|0.86
person|0.82
person|1.00
person|0.97
person|0.98
person|1.00
person|1.00
baseball_bat|0.33
handbag|0.81
tennis_racket|0.69
handbag|0.59


dining_table|0.74
cup|1.00
scissors|0.39
knife|0.36


chair|0.47
suitcase|0.98
suitcase|0.39
teddy_bear|0.36
suitcase|0.46
suitcase|0.94
suitcase|0.97
suitcase|0.96
suitcase|0.40
suitcase|0.95

图 12：Faster RCNN 目标检测结果可视化

1.3.2. Mask RCNN

我使用的 Mask RCNN 的配置为：pytorch 实现、backbone 为 ResNet-50[10]、Neck 模型为 FPN[11]。

1、性能测试结果如图 13、图 14、图 15 所示：

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.382
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.588
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.414
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.219
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.409
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.524
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.524
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.524
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.329
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.557
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.662
```

图 13：Mask RCNN bbox 测试结果

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.347
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.557
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.372
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.158
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.369
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.511
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.478
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.478
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.478
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.280
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.514
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.626
```

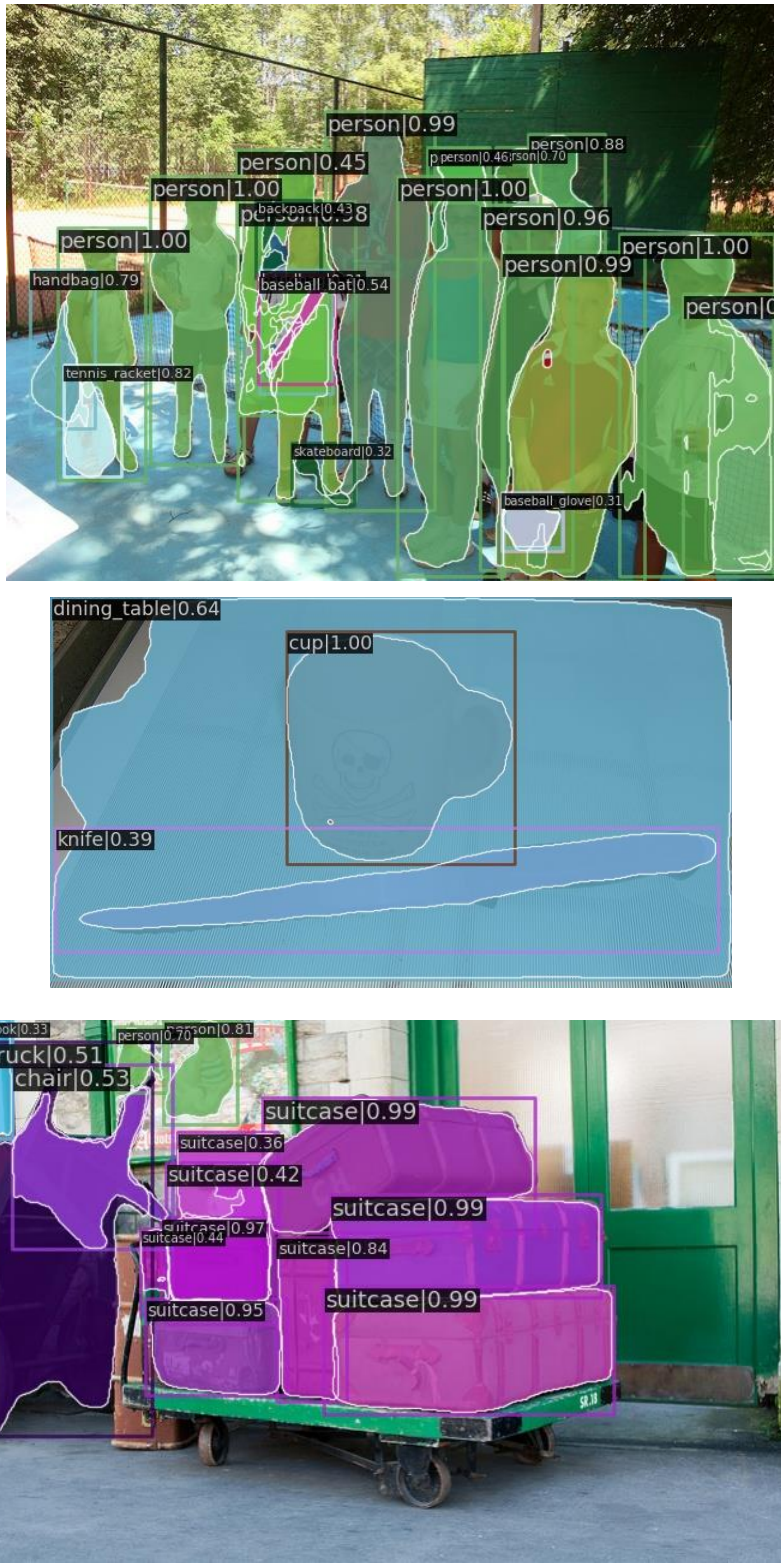
图 14：Mask RCNN 分割测试结果

```
Overall fps: 13.1 img / s, times per image: 76.4 ms / img
```

图 15：Mask RCNN 推理速度测试结果

由图可知：Mask RCNN 的 bbox mAP 为 0.382，segm mAP 为 0.347，推理速度为 13.1fps。

2、在 coco 测试集上随机选取一些图片（和 Faster RCNN 一样），可视化结果如图 16 所示：



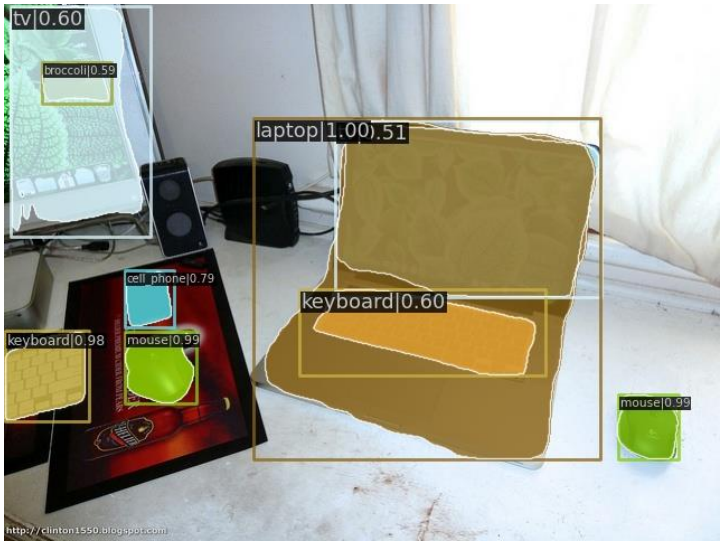


图 16：Mask RCNN 目标检测和实例分割结果可视化

1.3.3. 分析比较

Faster RCNN 和 Mask RCNN 的测试数据如表 1 所示：

表 1：Faster RCNN 和 Mask RCNN 的测试数据

模型	推理速度（fps）	bbox mAP	mask mAP
Faster RCNN	15.1	0.374	
Mask RCNN	13.1	0.382	0.347

由上表分析可知：

1、由于 Mask RCNN 在 Faster RCNN 的基础上新增了一个全卷积的 mask 分支，所以需要额外的计算开销，推理速度也因此稍慢一些。

2、因为 Mask RCNN 取消了量化操作，并将 RoI pooling layer 替换为 RoIAlign layer，减小了 RoI 的误差，所以 Mask RCNN 的 bbox mAP 比 Faster RCNN 略高。

1.4. 深入理解

在阅读这些经典算法的原论文时，有些地方不是很理解，于是我通过查阅资料，进行进一步的、更深入的学习和理解。主要问题如下：

1、边界框回归时，为什么回归目标要设计那样的形式？

在[2]的附录中，作者详细介绍了边界框回归的过程和方法，后续算法也一直沿用该边界框回归算法。

对于一个边框，我们一般使用四个参数来表示(x, y, w, h)，分别表示矩形框的中心坐标和宽高。我们主要通过平移和尺度缩放来对生成的边框进行修正，使其更接近于 ground-truth bounding box，因此边框回归的目的就是学习该平移多少、缩放多少。

而[2]中定义的回归目标如图 17 所示：

$$
\begin{aligned}
t_x &= (G_x - P_x)/P_w \\
t_y &= (G_y - P_y)/P_h \\
t_w &= \log(G_w/P_w) \\
t_h &= \log(G_h/P_h).
\end{aligned}
$$

图 17：边框回归目标

为什么回归目标会定义这种形式，而不是中心坐标和宽高的差值呢？

首先，对于中心坐标 x、y，理想情况下，不同尺度的同一物体得到的特征应该是相同的，即$\emptyset_1 = \emptyset_2$。设这两个不同尺度的同一物体的中心 x 坐标分别为x_1、x_2，预测的边框中心 x 坐标分别为p_1、p_2，如果直接学习坐标差值，以 x 坐标为例，则学习到的映射应为$f(\emptyset_1) = x_1 - p_1$，$f(\emptyset_2) = x_2 - p_2$。但是$x_1 - p_1 \neq x_2 - p_2$，而$\emptyset_1 = \emptyset_2$，这不满足函数的定义，而边框回归学习的回归函数，因此需要将坐标差值除以宽高，得到相对平移距离。

对于宽高的 log 形式，是因为缩放尺度必须大于 0，最直观的、很容易想到的就是 e 的幂次方函数，反过来推导就变成了 log 形式。

参考文献

[1]Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, “Object Detection in 20 Years: A Survey.” arXiv, Jan. 18, 2023. doi: 10.48550/arXiv.1905.05055.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” presented at the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” 2014, pp. 346–361. doi: 10.1007/978-3-319-10578-9_23.

[4] R. Girshick, “Fast R-CNN,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” presented at the Advances in Neural Information Processing Systems, 2015, pp. 91–99.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN.” arXiv, Jan. 24, 2018. doi: 10.48550/arXiv.1703.06870.

[7] K. E. A. Van De Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, “Segmentation as selective search for object recognition,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 1879–1886. doi: 10.1109/ICCV.2011.6126456.

[8] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” presented at the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 431–440. doi: 10.1109/CVPR.2015.7298965.

[9] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 9351, pp. 234–241, 2015, doi: 10.1007/978-3-319-24574-4_28.

[10]K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

[11]T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection.” arXiv, Apr. 19, 2017. doi: 10.48550/arXiv.1612.03144.