深度学习训练营

案例 5：滴滴出行-交通场景目标检测

相关知识点：目标检测、开源框架的配置和使用（mmdetection, mmcv）

1 任务和数据简介

本次案例将使用深度学习技术来完成城市交通场景下的目标检测任务，案例所使用的数据集由滴滴出行提供，基于 D^2-City 大规模行车记录视频数据集[1]，经过视频抽帧等预处理步骤得到。数据集共包含 12,000 张图像，每张图像的大小为 1080×1920 或 720×1280，已划分为训练集(10,000 张)、验证集(1,000 张)、测试集(1,000 张)，其中训练集和验证集提供了检测标签，测试集仅提供图像，用于提交后测试模型效果。本数据集共包含 12 个待检测的物体类别，包括小汽车、公交车、自行车、行人等，具体定义及示例如图 1 所示。本任务的目标是在给定的交通场景图像中，尽可能完整、准确地检测出所有要求的物体，检测结果示例如图 2 所示。关于数据的更多细节可参考 D^2-City 文献[1].

为了方便使用，数据集的标注信息已被预处理成 MS-COCO 格式，MS-COCO 是通用物体检测领域最常用的数据集，如需深入理解数据集格式，请自行学习：MS-COCO 数据集官网[2]、MS-COCO 数据集文献[3]、MS-COCO 标注格式[4]. 模型的评估指标也使用 MS-COCO 常用指标 mAP(mean average precision)，请自行学习其定义及计算方式(无需自己实现)：mAP 定义[5]，mAP 计算方式[6][7]。

2 参考程序及使用说明

本次案例提供了完整、可供运行的参考程序，选取了带 FPN[8]结构的 Faster R-CNN[9]模型，基于 MMDetection 物体检测框架[10]实现，各程序简介如下：

- faster_rcnn_r50_fpn_1x_didi.py 为模型配置文件，安装 MMDetection 后置于 mmdetection/configs/faster_rcnn 路径下；
- didi_detection.py 为数据集配置文件，置于 mmdetection/configs/_base_/datasets 路径下，并将 data_root 变量修改为数据集所在路径；
- didi_demo.ipynb 用于可视化模型的检测结果。

参考程序的使用步骤及说明：

- 自行安装 MMDetection 最新版(v2.1.0)及其全部依赖库，包括 PyTorch 等(MMDetection GitHub: [10]，安装指南: [11])；学习必要的使用说明: MMDetection 文档[12](请务必仔细阅读 Getting Started 章节);

Class Name	Class ID	Examples in D^2-City
car	1	
van	2	
bus	3	
truck	4	
person	5	
bicycle	6	
motorcycle	7	
open-tricycle	8	
closed-tricycle	9	
forklift	10	
large-block	11	
small-block	12	

图 1: 待检测的 12 个物体类别及其示例

图 2: 检测结果示例，不同颜色的框代表不同类别

- 下载案例数据集，配置上述文件并置于 MMDetection 目录下的指定位置；
- 以参考程序(Faster R-CNN with FPN)为例，训练模型:

  python tools/train.py configs/faster_rcnn/faster_rcnn_r50_fpn_1x_didi.py
- 根据训练得到的模型(.pth 文件，训练后自动保存于 work_dirs/路径下)，在测试集上进行推理并得到.json 格式的输出文件:

  python tools/test.py configs/faster_rcnn/faster_rcnn_r50_fpn_1x_didi.py \

  ./work_dirs/faster_rcnn_r50_fpn_1x_didi/latest.pth \

  --format-only --options "jsonfile_prefix=./test_results"

  该命令会生成 test_results.json 文件，用于最终提交和评分。
- 参考程序的超参数沿用了 MS-COCO 数据集上的配置，可能在本案例数据集上表现不够好，仅以此为例展示如何完成本案例；图 3 是参考程序训练完成后在验证集上的结果(每轮训练结束后自动输出)，图 4 是测试集上的结果(学生不可见，用于最后评分)。

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.270
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.453
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.260
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.107
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.287
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.445
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.259
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.413
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.417
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.228
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.436
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.611
```

图 3: 参考程序训练完成后在验证集上的 mAP 结果

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.224
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.391
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.222
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.116
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.236
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.309
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.237
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.376
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.378
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.240
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.422
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.469
```

图 4: 参考程序训练完成后在测试集上的 mAP 结果 (学生不可见)

3 要求和建议

在参考程序的基础上，综合使用深度学习各项技术，尝试提升该模型在城市交通场景目标检测任务上的效果，以最后提交的.json 输出结果对应的测试集 mAP 值为评价标准。

可从物体检测领域的各个方面着手对模型进行提升，如尝试其它的检测算法(替换 Faster R-CNN)，如 FCOS, Cascade R-CNN 等；或者尝试改进 Faster R-CNN 算法本身的部件，如尝试其它网络结构(替换 ResNet50)、其它更适合本案例数据集的超参数(训练图像分辨率、anchor 设置、训练策略等)。建议参考 MMDetection 已经实现好的若干模型[13]，以及通用物体检测领域的其它常用方法和技巧[14]。

4 注意事项

- 提交所有代码和一份案例报告，提交 test_results.json 文件；
- 案例报告应详细介绍所有改进尝试及对应的结果(包括验证集 mAP 值和若干检测结果示例图)，无论是否成功提升模型效果，并对结果作出分析；
- 禁止任何形式的抄袭，借鉴开源程序务必加以说明。

5 参考资料

[1] Che et al. D^2-City: A Large-Scale Dashcam Video Dataset of Diverse Traffic Scenarios. arXiv 2019.

[2] MS-COCO 数据集: https://cocodataset.org/

[3] Lin et al. Microsoft COCO: Common Objects in Context. ECCV 2014.

[4] MS-COCO 标注格式: https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch

[5] MS-COCO mAP 定义: https://cocodataset.org/#detection-eval

[6] mAP 计算方式: https://www.zhihu.com/question/53405779

[7] mAP 计算方式: https://github.com/rafaelpadilla/Object-Detection-Metrics/

[8] Lin et al. Feature pyramid networks for object detection. CVPR 2017.

[9] Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NIPS 2015.

[10] MMDetection: https://github.com/open-mmlab/mmdetection

[11] MMDetection 安装指南: https://github.com/open-mmlab/mmdetection/blob/master/docs/install.md

[12] MMDetection 文档: https://mmdetection.readthedocs.io/

[13] MMDetection Model Zoo: https://github.com/open-mmlab/mmdetection/blob/master/docs/model_zoo.md

[14] Liu et al. Deep Learning for Generic Object Detection: A Survey. IJCV 2020.