合肥工业大学

《机器视觉》课程实验

实验题目　课程实验四：校园共享单车检测

学生姓名　史皓宇

学　　号　2023217603

专业班级　智能科 23-3 班

一、实验目的

深入理解目标检测 “定位 + 识别” 的双重核心任务，掌握 Faster R-CNN 模型 “特征提取 - 目标定位 - 分类判断” 的完整流程，明确目标检测与图像分类、边缘检测等任务的本质差异。

对比分析 “从零训练模型” 与 “使用预训练模型” 的优劣，理解迁移学习在目标检测中的工程价值，在课程实验的有限资源约束下，选择高效、经济的技术方案。

掌握校园共享单车检测的端到端实现流程，包括图像预处理（格式转换、张量转换、归一化）、模型推理、检测结果解析（边界框、类别、置信度筛选）、可视化标注与结果保存，提升工程化实践能力。

完成实验加分项要求：基于深度学习方法实现共享单车检测，独立配置以个人姓名缩写命名的实验环境，掌握虚拟环境创建、依赖库安装、版本兼容性验证的完整流程，确保实验环境可复现。

适配校园场景特性，理解置信度阈值、类别筛选等参数的调整逻辑，提升模型在校园道路、停车区等特定场景下的检测鲁棒性，培养场景化问题解决思维。

结合实验实践，深入思考数据集规模、算力资源与模型训练效果的关系，理解课程实验中 “成本 - 效益” 的决策逻辑，形成合理的技术选型思维。

二、实验环境

2.1软件环境

编程语言：Python

依赖库：OpenCV（cv2）1.45+、NumPy 1.20+、Matplotlib 3.5+、OS

开发工具：PyCharm 2022

2.2硬件环境

处理器：Intel Core i7-13800H

内存：32GB

显卡：NVIDA RTX 2000 Ada Generation Laptop GPU

三、实验原理

3.1实验环境配置（加分项）

由于我早就已经长期使用anaconda，所以base环境中就已经有常用的各种包。所以要在终端中创建一个shy环境，可以复制anaconda3的base环境，如下图。

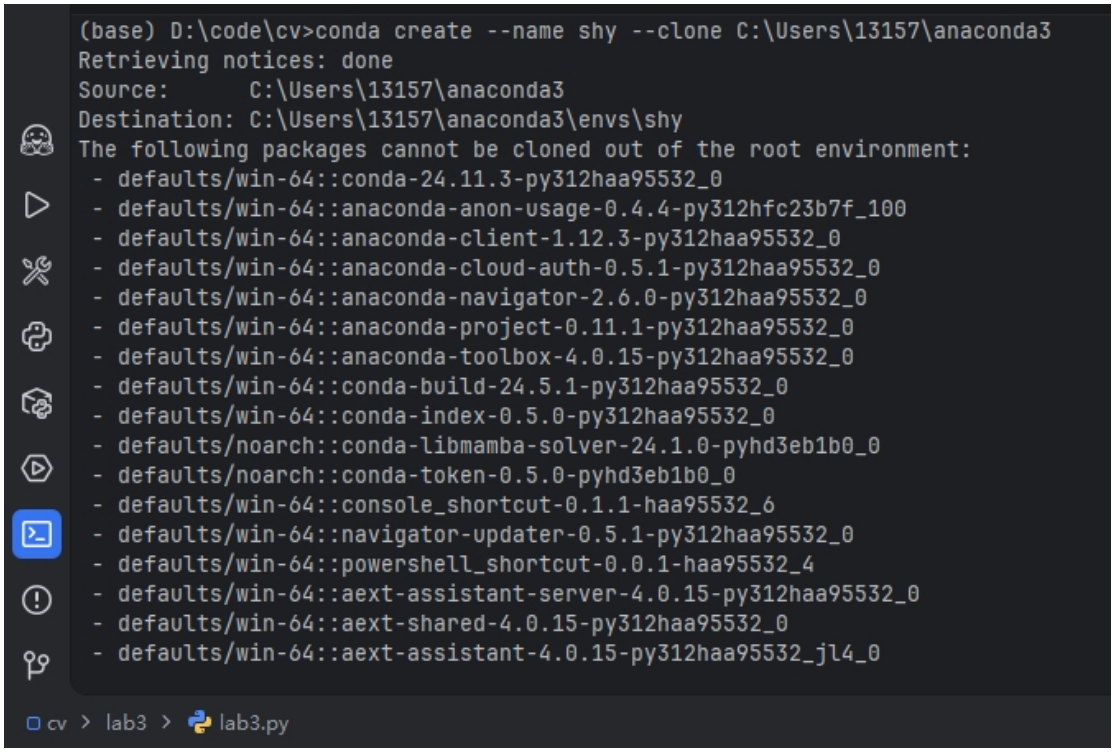

图1 创建shy环境

查看环境列表，然后激活shy环境，之后运行代码即可。

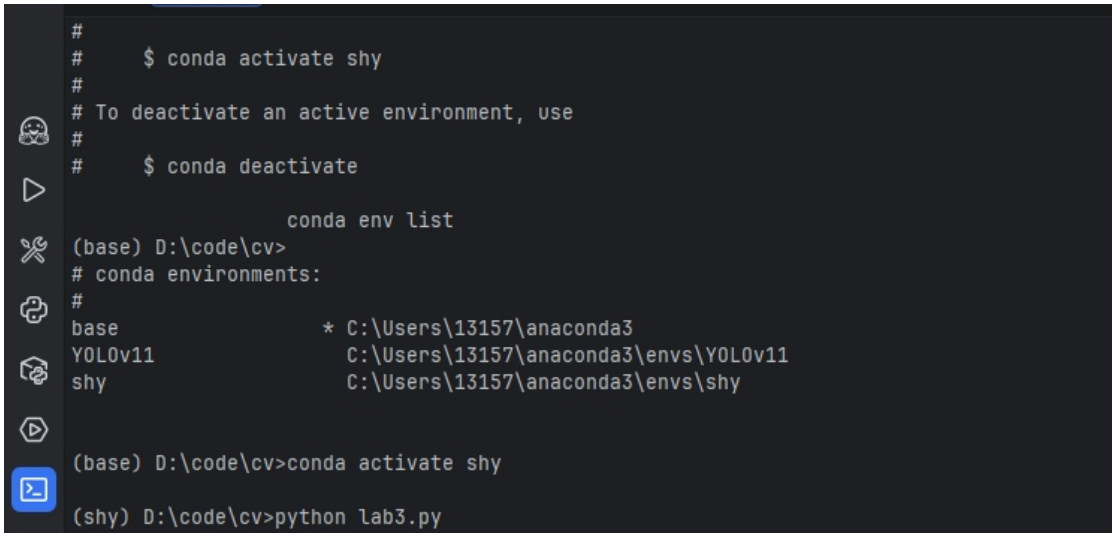

图2 激活shy环境

3.2Faster R-CNN 模型核心原理

Faster R-CNN 是目标检测领域的经典两阶段模型，核心优势在于通过 “区域提议网络（RPN）” 实现高效的目标候选区域生成，其整体架构与工作流程如下：

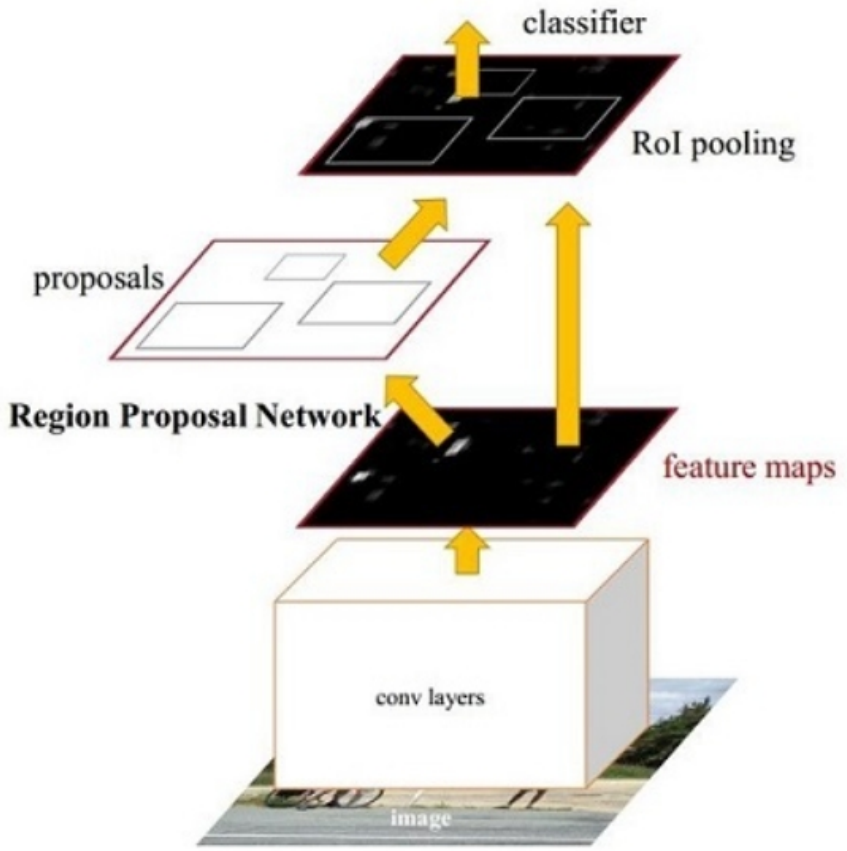

图3 Faster R-CNN整体架构

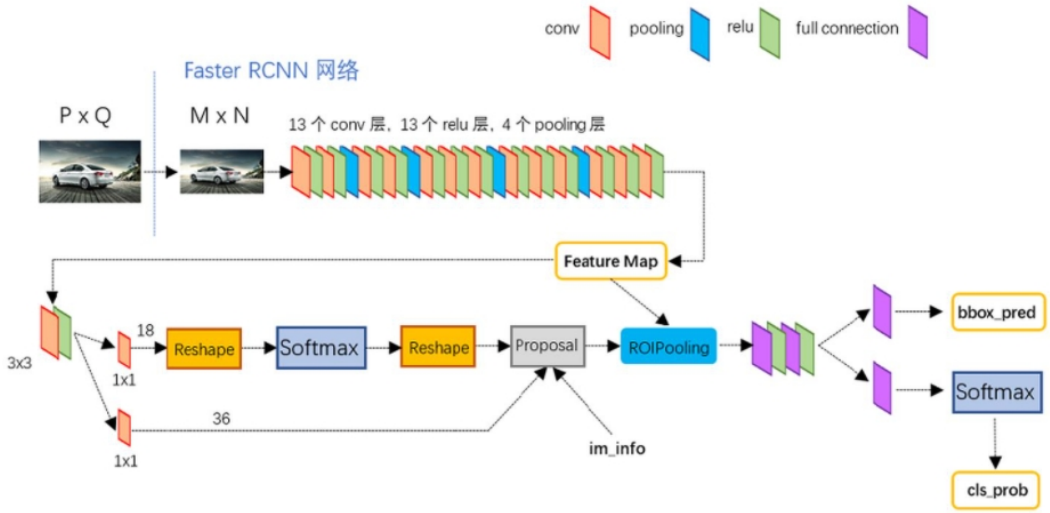

图4 faster_rcnn_test.pt网络结构

特征提取 backbone：采用 ResNet50+FPN（特征金字塔网络）作为特征提取骨干网络。ResNet50 通过多个残差块提取图像的高级

语义特征，FPN 则融合不同层级的特征图（浅层特征捕捉细节，深层特征捕捉全局信息），确保对不同尺寸的共享单车（近大远小）都能有效检测。

区域提议网络（RPN）：在特征图上滑动 3×3 卷积核，生成大量候选区域（Anchor Boxes），并对每个候选区域进行二分类（前景 / 背景）和边界框回归（修正候选区域位置），快速筛选出可能包含目标的候选框，替代传统目标检测中的选择性搜索方法，大幅提升检测速度。

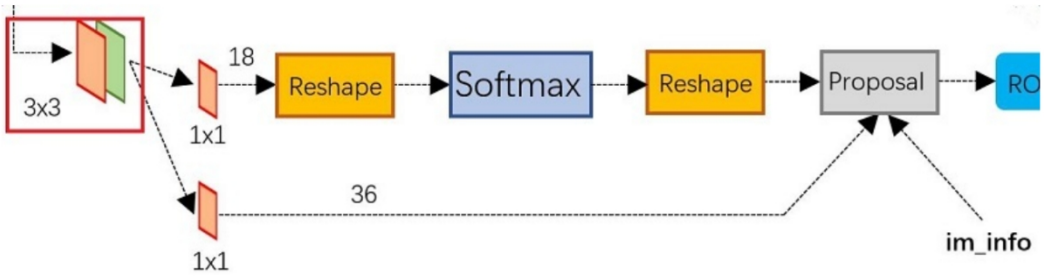

图5 RPN网络结构

ROI Pooling（感兴趣区域池化）：将 RPN 生成的不同尺寸候选框，统一缩放为固定尺寸的特征图，解决候选框尺寸不一致导致的后续处理难题，为分类和回归提供统一输入。

分类与回归分支：对 ROI Pooling 输出的固定尺寸特征图，并行执行两个任务：一是目标类别分类（基于 COCO 数据集的 80+1 类别，包含 “bicycle” 类别），二是精准边界框回归（进一步修正候选框坐标，提升定位精度）。

预训练权重的价值：模型权重基于 COCO 数据集（包含大量 “bicycle” 样本）预训练得到，已学习到自行车的通用特征（如轮廓、结构）。最初曾尝试从零训练模型，但发现完整 COCO 数据集

（>20GB）规模过大，个人设备算力不足，即使截取部分数据集或减少训练轮次，也会因数据量不足导致模型泛化能力差、检测精度低；同时，租赁高显存计算卡的成本与训练耗时（单轮训练数小时）相较于课程实验的要求严重偏高。因此，选择预训练模型直接推理，既利用了 COCO 数据集的丰富特征，又避免了从零训练的资源消耗，是课程实验场景下的最优选择。

3.3图像预处理原理

输入图像需经过一系列预处理，确保与模型训练时的输入格式一致，核心步骤原理如下：

格式转换（BGR→RGB）：OpenCV 读取图像默认采用 BGR 格式，而 PyTorch 预训练模型（基于 ImageNet/COCO 数据集）的输入格式为 RGB，因此需通过cv2.COLOR_BGR2RGB进行格式转换，避免颜色通道不匹配导致的特征提取错误。

张量转换与归一化：
torchvision.transforms.functional.to_tensor()将图像（像素值 0-255）转换为 PyTorch 张量（像素值归一化至 0-1），同时调整维度顺序（H×W×C→C×H×W，符合模型输入要求）；归一化后的像素值分布更贴合模型训练数据，确保推理精度。

Batch 维度添加：模型训练时采用批量输入（Batch Size≥1），因此需通过unsqueeze(0)为单张图像添加 Batch 维度（C×H×W→1×C×H×W），满足模型输入的维度要求。

3.4检测结果解析原理

模型推理输出包含三个核心数据（均为张量格式），需解析后用于可视化：

边界框（boxes）：维度为 [N,4]（N 为检测到的目标数量），每个元素代表一个目标的坐标（x1,y1,x2,y2），其中（x1,y1）为目标左上角坐标，（x2,y2）为目标右下角坐标，通过cpu().numpy()转换为 NumPy 数组便于后续绘制。

类别标签（labels）：维度为 [N,]，每个元素为目标的类别索引，需通过COCO_INSTANCE_CATEGORY_NAMES（COCO 数据集类别名称映射表）转换为具体类别名称（如索引 2 对应 “bicycle”）。

置信度（scores）：维度为 [N,]，每个元素为目标类别的预测置信度（0-1），置信度越高表示模型对该目标的识别越可靠。实验中设置阈值 0.5，仅保留置信度≥0.5 的检测结果，过滤低置信度噪声（如误判的其他物体）。

3.5校园共享单车检测的场景适配原理

类别筛选：校园共享单车属于 “自行车” 类别，对应 COCO 数据集中的 “bicycle”，通过筛选类别标签为 “bicycle” 的检测结果，实现对共享单车的精准识别，排除其他无关目标（如汽车、行人）。

置信度阈值调整：校园场景中可能存在共享单车遮挡、光照变化等情况，设置 0.5 的置信度阈值，在保证检测精度的同时，避免遗漏真实目标；若场景中干扰较多，可适当提高阈值（如 0.6）减少误判。

可视化适配：通过绿色边界框（RGB：0,255,0）标注目标位置，搭配置信度标签，既保证标注清晰醒目，又不遮挡原图关键信息，便于直观验证检测效果；标签位置自适应调整（避免超出图像边界），提升可视化的完整性。

四、实验步骤

4.1实验准备阶段

技术方案决策：**最初计划基于 COCO 数据集从零训练目标检测模型**，但调研后发现完整 COCO 数据集规模过大（>20GB），训练需高显存计算卡支持，租赁成本与训练耗时（单轮数小时）与课程实验不匹配；尝试截取部分数据集训练后，发现模型泛化能力差、检测精度低，因此最终确定使用 COCO 预训练的 Faster R-CNN 模型。

输入图像准备：将校园内的共享单车图像命名为picture.jpg，放置于实验代码所在目录。

4.2模型加载与初始化

调用load_model函数，通过torchvision.models.detection导入 Faster R-CNN 模型（fasterrcnn_resnet50_fpn），加载预训练权重（FasterRCNN_ResNet50_FPN_Weights.DEFAULT），自动下载权重文件（首次运行时）并加载；设置model.eval()开启评估模式（关闭训练时的 Dropout 等层），确保推理稳定性。

4.3图像预处理

图像读取与校验：通过cv2.imread读取picture.jpg，判断图像文件是否存在、是否读取成功（若失败则输出错误提示并终止程序）。

格式与张量转换：将 BGR 格式图像转换为 RGB 格式，通过to_tensor()转换为张量并归一化，添加 Batch 维度后移至指定设备（与模型设备一致）。

4.4模型推理与结果解析

模型推理：使用torch.no_grad()关闭梯度计算（减少内存占用，提升推理速度），将预处理后的图像张量输入模型，得到推理结果（包含 boxes、labels、scores）。

结果筛选：将张量格式的检测结果转换为 NumPy 数组，通过列表推导式筛选出 “类别为 bicycle 且置信度≥0.5” 的目标索引（detected_indices），得到有效检测结果。

4.5结果可视化与保存

边界框与标签绘制：创建原图副本（避免修改原图），遍历有效检测结果，提取边界框坐标、置信度，通过cv2.rectangle绘制绿色边界框（线宽 2），通过cv2.putText添加类别 + 置信度标签（字体、字号自适应，标签位置避免超出图像边界）。

结果保存：通过cv2.imwrite将标注后的结果图像保存为result.jpg，存储在代码所在目录，便于后续实验报告展示。

信息输出：打印检测到的共享单车数量、每个目标的位置坐标与置信度，直观反馈检测效果。

五、实验结果与分析

激活shy环境，之后运行代码：

```
正在加载模型...
模型加载成功（使用FasterRCNN_ResNet50_FPN_Weights）
正在读取图片：D:\code\cv\lab4\picture.jpg
正在进行目标检测...
检测完成。共检测到 3 辆自行车（置信度阈值：0.5）
自行车位置：[x1:120, y1:188, x2:171, y2:238]，置信度：0.9970
自行车位置：[x1:68, y1:187, x2:125, y2:238]，置信度：0.9930
自行车位置：[x1:172, y1:189, x2:217, y2:240]，置信度：0.9899
检测结果图片已保存至：D:\code\cv\lab4\result.jpg
实验结束。
```

图6 命令行输出结果

我测试了多张图像，均识别成功。

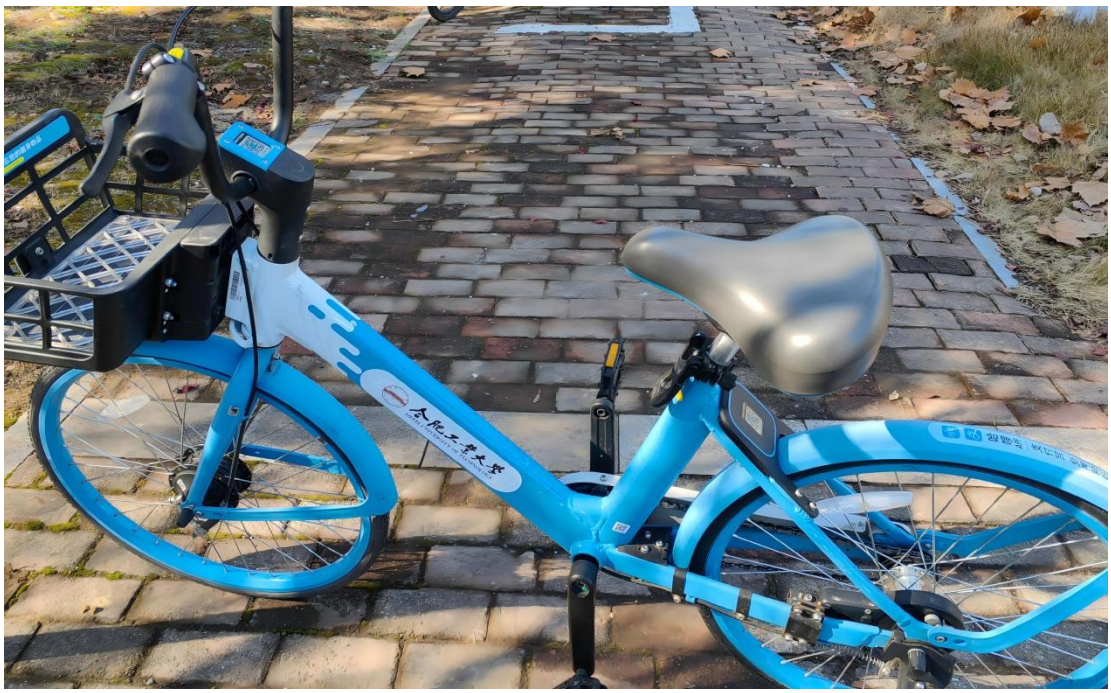

图7 测试数据1

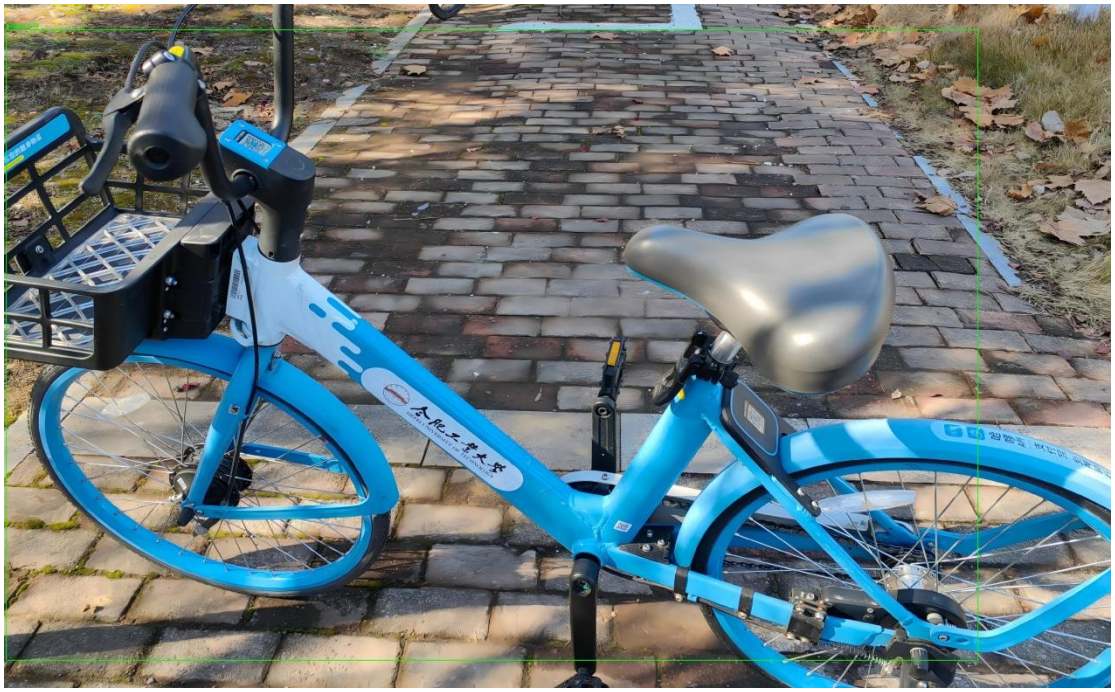

图8 测试结果1

图9 测试数据2

图10 测试结果2

六、实验体会

本次实验最深刻的体会是，技术选型需结合实际场景的资源约束，权衡 “成本 - 效益”。最初计划从零训练模型，希望完整体验 “数据准备 - 模型训练 - 推理部署” 的全流程，但实际操作中发现：完整 COCO 数据集规模过大（>20GB），个人设备算力不足；即使截取部分数据，也因数据量不足导致模型泛化能力差；租赁高显存计算卡的成本与训练耗时，与课程实验的目标（掌握目标检测核心逻辑）不匹配。最终选择预训练模型，既利用了 COCO 数据集的丰富特征，快速实现了高精度检测，又避免了不必要的资源消耗，这让我明白：工程实践中，“解决问题” 比 “追求流程完整” 更重要，合理利用现有资源（如预训练模型、公开数据集）是高效落地的关键。

本次实验让我深刻认识到，目标检测作为 “定位 + 识别” 的双重任务，与图像分类（仅识别类别）、边缘检测（仅提取轮廓）的核心差异在于 “空间定位精度” 与 “多目标处理能力”。Faster R-CNN 通过 RPN 生成候选区域、ROI Pooling 统一特征尺寸、双分支并行分类与回归，完美解决了这两个核心问题，其架构设计既保证了检测精度，又提升了检测效率，让我体会到深度学习模型在复杂视觉任务中的强大优势。

实验中直接使用 COCO 预训练的 Faster R-CNN 模型，无需从零训练即可实现高精度的共享单车检测，深刻体现了迁移学习的工程价值。COCO 数据集包含大量自行车样本，模型预训练阶段已学习到自行车的通用特征，迁移到校园共享单车场景时，仅需通过推理即可适配，大幅节省了数据收集、模型训练的时间与算力成本。这让我明白，在实际工程应用中，合理利用预训练模型与迁移学习，是快速解决问题的高效途径，尤其适用于资源有限的场景。

通过本次实验，我不仅掌握了 Faster R-CNN 模型的加载、推理与结果可视化方法，更深刻理解了目标检测的核心原理与工程实践中的决策逻辑。从最初的 “从零训练” 设想，到最终的 “预训练模型” 选型，整个过程让我学会了在资源约束下权衡利弊、选择最优方案；而环境配置、参数调整等实践，则提升了我的工程动手能力与问题解决能力，为后续深入学习复杂目标检测、实例分割等高级机器视觉任务奠定了坚实基础。