

# 基于 MMDetection 的 Mask R-CNN 和 Sparse R-CNN 模型测试

——使用 MMDetection 的实例分割和目标  
检测方法训练并测试模型 Mask R-CNN 和  
Sparse R-CNN

CV Mid-term Report: Task 2

姓名: 黄亦绪  
学号: 23307110412  
DATA130051.01 计算机视觉  
(春季, 2025)

复旦大学  
大数据学院  
2025 年 5 月 28 日

# 目录

<b>1 引言</b>	<b>4</b>
1.1 实验背景 . . . . .	4
1.2 实验环境: GPU . . . . .	4
1.3 实验目标 . . . . .	4
<b>2 数据集</b>	<b>5</b>
2.1 简介 . . . . .	5
2.2 训练集、验证集和测试集划分 . . . . .	5
2.3 数据集格式转换: VOC 到 COCO . . . . .	5
2.3.1 掩码的转化 . . . . .	5
2.3.2 最终步骤 . . . . .	6
2.4 数据增强 . . . . .	6
<b>3 实验设计</b>	<b>8</b>
3.1 架构 . . . . .	8
3.1.1 整体架构 . . . . .	8
3.1.2 工作架构解析 . . . . .	9
3.1.3 Mask R-CNN . . . . .	10
3.1.4 Sparse R-CNN . . . . .	10
3.1.5 架构优势 . . . . .	11
3.2 评估指标 . . . . .	11
<b>4 超参数选取</b>	<b>13</b>
4.1 学习率选取 . . . . .	13
4.1.1 SGD . . . . .	13
4.1.2 Adam . . . . .	13
4.1.3 最终选择: Adam . . . . .	14
4.2 batch_size . . . . .	14
4.3 iteration . . . . .	15
4.4 epoch . . . . .	15
<b>5 实验过程</b>	<b>16</b>
5.1 训练过程 . . . . .	16
5.1.1 MASK R-CNN . . . . .	16
5.1.2 SPARSE R-CNN . . . . .	16
5.1.3 SPARSE R-CNN . . . . .	17

5.1.4	准确率变化曲线 . . . . .	18
5.2	mAP 变化曲线 . . . . .	19
5.2.1	MASK R-CNN . . . . .	19
5.2.2	SPARSE R-CNN . . . . .	20
5.3	模型对比 . . . . .	21
<b>6</b>	<b>实验结果</b>	<b>22</b>
6.1	测试集上的 mAP . . . . .	22
6.2	MASK R-CNN 的预测效果 . . . . .	22
6.2.1	测试集上 . . . . .	22
6.2.2	VOC 数据集外——跨数据集性能验证 . . . . .	24
6.3	SPARSE R-CNN 的预测效果 . . . . .	25
6.3.1	测试集上 . . . . .	25
6.3.2	VOC 数据集外——跨数据集性能验证 . . . . .	25
6.3.3	总结 . . . . .	26
<b>7</b>	<b>讨论与分析</b>	<b>28</b>
7.1	模型性能对比分析 . . . . .	28
7.2	训练过程与准确率变化 . . . . .	28
7.3	mAP 变化与性能瓶颈 . . . . .	29
7.4	影响性能的关键因素 . . . . .	29
7.5	改进可能性 . . . . .	29
<b>8</b>	<b>资源</b>	<b>31</b>

# 1 引言

## 1.1 实验背景

目标检测和实例分割是计算机视觉领域的两个基础任务，在自动驾驶、智能监控、医学影像分析等实际应用中发挥着重要作用。近年来，随着深度学习技术的发展，基于卷积神经网络的目标检测算法取得了显著进展。其中，Mask R-CNN 和 Sparse R-CNN 作为两种具有代表性的算法，分别在两阶段检测框架和稀疏检测框架中展现出优异的性能。

本实验基于 OpenMMLab 团队开发的 MMDetection 框架，在经典的 PASCAL VOC 数据集上对这两种算法进行训练和评估。MMDetection 是一个模块化设计的开源目标检测工具箱，支持多种主流检测算法，并提供了高效的训练和测试流程。

PASCAL VOC 数据集包含 20 个常见物体类别，是目标检测领域的基准数据集之一。我将在该数据集上训练 Mask R-CNN 和 Sparse R-CNN 模型，通过对比分析它们在检测精度、分割效果等方面的表现差异。此外，我还将在测试模型在未见数据上的泛化能力，以评估其在实际应用中的潜力。

## 1.2 实验环境：GPU

实验使用 Nvidia RTX4060 GPU (Driver Version: 572.83, CUDA Version: 12.8)，实现 MMDetection 框架目标检测和实例分割的训练和测试。

## 1.3 实验目标

本次实验的目标是：

- 掌握使用 MMDetection 框架进行目标检测和实例分割的基本流程；
- 深入理解 Mask R-CNN 和 Sparse R-CNN 的工作原理及性能特点；
- 基于 mAP (Mean Average Precision, 平均精度均值) 实现两个模型的对比，可视化训练验证的过程，展示目标检测和实例分割结果。

## 2 数据集

### 2.1 简介

本实验采用 PASCAL VOC2012 数据集，该数据集包含 20 个常见物体类别，是目标检测和实例分割领域的基准数据集之一。与 MMDetection 默认的 VOC2007+2012 联合训练模式不同，本实验仅使用 VOC2012 数据，以 8:1:1 的比例重新划分训练集、验证集和测试集，确保模型评估的独立性。

### 2.2 训练集、验证集和测试集划分

数据集划分比例如下：

- 训练集 (train): 2039 张图像 (80%)
- 验证集 (val): 436 张图像 (10%)
- 测试集 (test): 438 张图像 (10%)

划分过程通过修改 VOC 原始 `ImageSets/Segmentation` 中的文件列表实现，生成对应的 `train_new.txt`、`val_new.txt` 和 `test_new.txt` 文件。

### 2.3 数据集格式转换：VOC 到 COCO

为实现与 MMDetection 框架的兼容性，将 VOC 格式转换为 COCO 格式。

#### 2.3.1 掩码的转化

在这个过程中，我遇到一个问题，如何将 VOC 原本以 `.png` 形式存在的物体分割掩码转化为 COCO `.json` 文件的掩码字段。我想出了两种策略：

- **方法一** `.xml` 文件获取物体 `bbox`, 再找分割有色部分在 `bbox` 中像素最多的分割图形转为 RLE 格式的掩码
- **方法二** 分辨分割有色部分的边界，取上下左右的边界形成 `box`, 再与 `.xml` 文件获取物体 `bbox` 计算 IoU (Intersection over Union, 交并比)，取最大值

我对两种策略都做了实验，`segm_mAP` (指标的具体含义参考后面的解释) 的值随训练轮数的变化如下图 1。绿色的是方法一，橙色是方法二，方法二相较方法一实现 3% 左右 mAP 的提升。

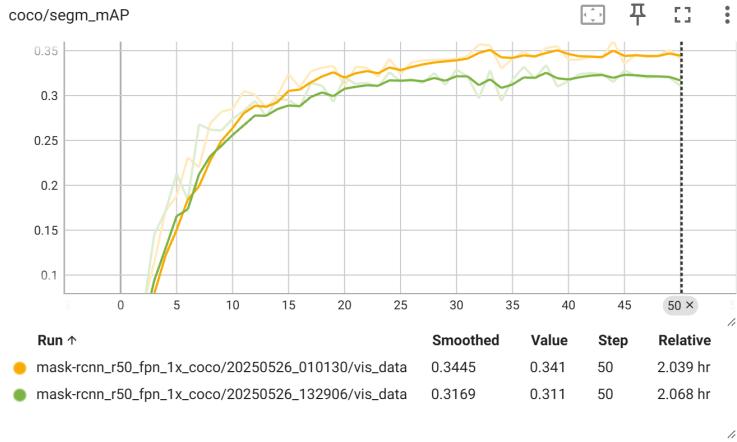


图 1: 两种策略的 segm\_mAP 比较图

### 2.3.2 最终步骤

所以，最后的格式转换步骤如下：

- **文件匹配**: 对于每张图像，关联其 JPEG 图像（位于 `JPEGImages`）、XML 标注（位于 `Annotations`）和 PNG 分割掩码文件（位于 `SegmentationObject`）
- **标注解析**: 解析 XML 文件获取物体边界框和类别，同时跳过困难样本 (`difficult=1`) 和无效边界框
- **掩码匹配**: 用上面更好的第二种方法
- **RLE 编码**: 将匹配的二值掩码转换为 COCO 要求的 RLE (Run-Length Encoding) 格式
- **结果保存**: 最终生成符合 COCO 标准的 JSON 标注文件，保存路径为`/home/huangyixu/data/VOCdevkit/VOC2012/annotations/`，分别输出为 `annotations_train.json`、`annotations_val.json` 和 `annotations_test.json`

## 2.4 数据增强

采用 MMDetection 默认的 COCO 实例分割数据增强流程：

- `LoadImageFromFile`: 从文件加载图像
- `LoadAnnotations`: 加载边界框和掩码标注

- `Resize`: 将图像缩放至  $1333 \times 800$  像素（保持长宽比）
- `RandomFlip`: 以 0.5 概率进行随机水平翻转
- `PackDetInputs`: 打包检测输入数据

该流程在保证数据多样性的同时，维持了检测任务对几何变换的鲁棒性。

### 3 实验设计

#### 3.1 架构

##### 3.1.1 整体架构

我们从 MMDetection 运行的文件结构开始解析其工作的整体架构。如下图 2 是文件结构：

```
|── configs
|   ├── _base_
|   |   ├── models
|   |   |   └── mask-rcnn_r50_fpn.py
|   |   ├── datasets
|   |   |   ├── coco_instance.py
|   |   |   └── coco_detection.py
|   |   ├── schedules
|   |   |   └── schedule_1x.py
|   |   └── default_runtime.py
|   ├── sparse_rcnn
|   |   └── sparse-rcnn_r50_fpn_1x_coco.py
|   └── mask_rcnn
|       └── mask_rcnn_r50_fpn_1x_coco.py
└── tools
    ├── test.py
    └── train.py
└── work_dirs
    ├── mask_rcnn_r50_fpn_1x_coco
    └── sparse_rcnn_r50_fpn_1x_coco
└── data
```

图 2: 文件结构

- **configs/**: 存储模型配置文件，例如 `mask_rcnn_r50_fpn.py`，用于定义模型结构、训练参数（如 `train_cfg` 和 `val_cfg`）、数据集设置等。配置文件采用 Python 语法，基于 `mmcv` 的配置系统，支持灵活的继承和修改。例如，我在配置文件中修改并定义了固定数量 epoch 的训练循环、Adam 优化器和余弦退火学习率策略等。
- **mmdet/ (未展示在图中，在运行时调用)**: MMDetection 的核心代码库，包含模型组件、数据集处理、训练和评估模块。其主要子模块包括：
  - `models/`: 定义模型架构，如 `backbones/`（骨干网络，例如 ResNet-50）、`necks/`（特征金字塔网络 FPN）、`heads/`（RPN 和检测头）。例如，Mask R-CNN 的 RPN 模块计算 `loss_rpn_cls` 和 `loss_rpn_bbox`，而检测头负责 `loss_cls` 和 `loss_bbox`。

- `datasets/`: 支持 COCO、VOC 等数据集，负责数据加载和预处理。
- `core/`: 包含训练循环(`EpochBasedTrainLoop`)、钩子(`CheckpointHook`、`TensorboardLoggerHook`) 和评估工具。
- `tools/`: 提供实用脚本，如 `train.py` (训练脚本)、`test.py` (测试脚本) 和 `analysis.py` (分析工具)。例如，运行训练时通过 `python tools/train.py configs/mask_rcnn_r50_fpn.py` 启动，训练过程中会调用 `mmdet/core` 中的训练循环，并按照 `default_hooks` 和 `log_config` 设置记录日志。
- `work_dirs/`: 存储训练过程中的输出，包括日志文件（例如 TensorBoard 日志）、模型权重 (`epoch_*.pth`) 和配置文件副本。之后实验中，训练集、验证集损失，mAP 的变化等通过 `TensorboardLoggerHook` 记录到 `work_dirs/tensorboard`，便于可视化分析。

### 3.1.2 工作架构解析

MMDetection 的工作架构基于模块化和流水线设计，分为以下几个主要阶段：

1. **数据准备与加载**: 通过 `mmdet/datasets` 加载数据集（如 COCO 或 VOC），根据配置文件中的 `data` 字段进行预处理（例如数据增强、归一化）。数据加载器 (DataLoader) 按照 `train_cfg` 中的批次大小和验证间隔分发数据。
2. **模型构建**: 根据 `configs/` 中的模型配置文件，`mmdet/models` 动态构建模型。例如，Mask R-CNN 包括骨干网络 (ResNet-50)、FPN、RPN、检测头和掩码头。模型的损失函数（如 `loss_rpn_cls` 和 `loss_cls`）在训练时分别计算并优化。我们实验中的两种模型架构分别是 Mask R-CNN 和 Sparse R-CNN，两者对比如表 ?? 所示。
3. **训练与优化**: `tools/train.py` 调用 `mmdet/core` 中的训练循环 (`EpochBasedTrainLoop`)，按照 `optim_wrapper` 和 `param_scheduler` 定义的优化器和学习率策略更新参数。训练过程中的钩子(如 `CheckpointHook` 和 `TensorboardLoggerHook`) 定期保存模型权重和记录损失（如图 9 所示）。

4. **验证与评估**: 每隔 `val_interval` (例如 1 个 epoch) 运行 `ValLoop`, 计算验证集的损失和指标 (如 mAP)。通过修改 `val_cfg` 和 `log_config`, 验证集损失被记录到 TensorBoard, 便于分析模型性能。
5. **测试与推理**: 使用 `tools/test.py` 和 `TestLoop` 对测试集进行推理, 输出检测结果 (如边界框和掩码)。

组件	Mask R-CNN	Sparse R-CNN
候选生成	RPN (密集候选)	可学习参数 (稀疏候选)
特征提取	FPN 多尺度	单尺度 + 动态卷积
RoI 操作	RoIAlign	无 (直接特征交互)
输出	密集检测 + 分割	稀疏检测
训练方式	两阶段	端到端

表 1: Mask R-CNN 与 Sparse R-CNN 架构对比

### 3.1.3 Mask R-CNN

Mask R-CNN 是基于 Faster R-CNN 的经典两阶段实例分割模型, 其核心创新是在目标检测基础上增加了一个并行的掩码预测分支。首先通过区域提议网络 (RPN) 生成候选框, 然后通过 RoIAlign 精确对齐特征区域, 最终同时输出目标类别、边界框和像素级掩码。其优势在于高精度分割能力, 但计算成本较高, 依赖密集的候选框生成和后处理。架构如图 3。

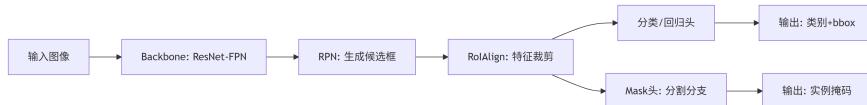


图 3: Mask R-CNN 架构图

### 3.1.4 Sparse R-CNN

Sparse R-CNN 是一种完全稀疏化的目标检测框架, 摒弃了传统的密集候选框机制 (如 RPN), 转而使用可学习的稀疏提议框 (learnable proposal boxes) 和动态卷积进行预测。它通过少量 (如 100 个) 可优化提议框直接生成检测结果, 大幅减少计算冗余, 适合端到端训练。优势在于结构简洁、效率高, 但小目标检测性能可能弱于密集方法。架构如图 4。它的架构决定了它相比 Mask R-CNN 只能进行边界框的预测, 而无法进行掩码预测。

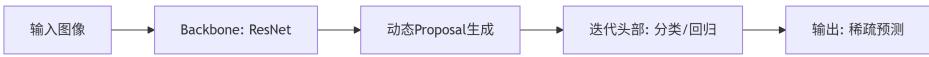


图 4: Sparse R-CNN 架构图

### 3.1.5 架构优势

MMDetection 的模块化设计使其高度灵活和可扩展。配置文件系统允许用户通过继承和覆盖快速调整模型和训练参数，例如切换优化器 (SGD 到 Adam) 或调整学习率策略 (如余弦退火)。此外，钩子机制 (`default_hooks`) 和日志系统 (`TensorboardVisBackend`) 支持实时监控和调试，确保训练过程可控且高效。

## 3.2 评估指标

由于本次实验是将 VOC 数据转为 COCO 格式，所以依据 MMDetection 中对 COCO 数据集的评价方式，基于 **平均精度 (mAP)** 和 **平均召回率 (mAR)**，结合不同的 IoU（如图 5）阈值和目标面积，以衡量目标检测和实例分割模型的性能。

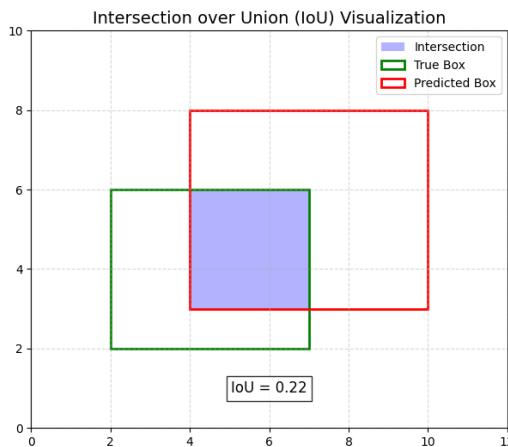


图 5: IoU 指标形象示意图

由于任务主要是目标检测和实例分割，主要分为两个主要指标——`bbox_mAP` 和 `segm_mAP`:

- **`bbox_mAP`** 通过计算模型预测的边界框与真实边界框之间的 IoU (交并比)，在不同 IoU 阈值（通常从 0.5 到 0.95，步长 0.05）上计算精确率和召回率，生成 PR 曲线，然后取平均值来评估目标检测性能。具体步骤包括：对每个类别计算 TP（真阳性）、FP（假阳性）和 FN

(假阴性)，根据置信度排序预测结果，计算不同阈值下的 AP (平均精度)，最后对所有类别和 IoU 阈值取平均得到 `bbox_mAP`。

- `segm_mAP` 针对像素级分割掩码进行评估。计算过程包括：将预测的分割掩码与真实掩码进行 IoU 计算，同样在 0.5 到 0.95 的 IoU 阈值范围内生成 PR 曲线，计算每个类别的 AP，然后对所有类别取平均。`segm_mAP` 不仅考虑目标定位，还评估掩码的像素级准确性，因此对分割边界和内部细节的匹配要求更高。

这两个指标在进行精度和目标物体尺寸的细分，衍生出如下的指标：

- **mAP@IoU=0.50:0.95**: 主要指标，IoU 从 0.5 到 0.95 (步长 0.05) 的平均精度均值，反映模型在不同严格度下的综合性能。
- **mAP@IoU=0.50**: IoU=0.5 时的平均精度，检测框与真实框重叠度较低时 (较宽松) 的性能。
- **mAP@IoU=0.75**: IoU=0.75 时的平均精度，要求更高重叠度，反映更严格的检测性能。
- **mAP@area=small/medium/large**: 按目标面积 (小:  $< 32^2$ ，中:  $32^2 < \text{area} < 96^2$ ，大:  $> 96^2$ ) 计算的 mAP，分别评估模型对不同大小目标的检测能力。
- **mAR@IoU=0.50:0.95**: 平均召回率，IoU 从 0.5 到 0.95 的均值，衡量模型召回目标的能力。
- **mAR@maxDets=100/300/1000**: 限制最大检测框数量 (100、300、1000) 时的 AR，反映模型在不同检测数量约束下的召回能力。
- **mAR@area=small/medium/large**: 按目标面积计算的 AR，评估不同大小目标的召回率。

## 4 超参数选取

### 4.1 学习率选取

我首先进行了实验，希望选择 SGD 和 Adam 的优化器以实施不同的学习策略从而找出最优解。学习率的变化如图 6（粉色是 Adam，灰色是 SGD）。可以看出 SGD 的学习率偏大，会断崖式从高到低变化；Adam 的学习率偏小，柔和平滑变化。

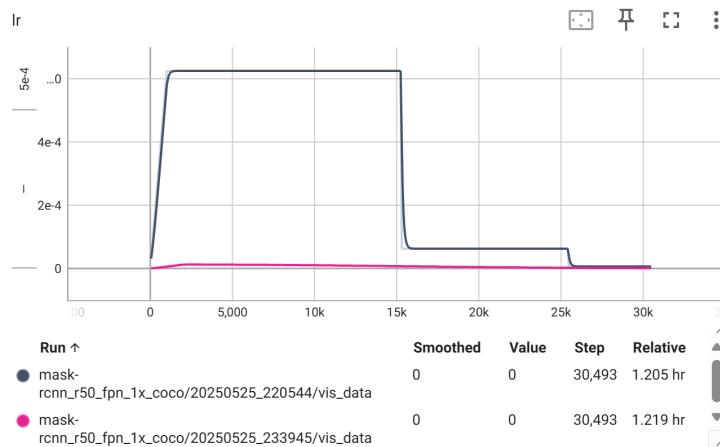


图 6: MASK R-CNN 的学习率策略比较

#### 4.1.1 SGD

第一种策略使用 SGD 优化器，初始学习率为 0.005，结合 1000 次迭代的线性 Warmup (从 0.001 倍学习率增长到 1.0 倍)，随后通过 MultiStepLR 在第 15 和 25 个 epoch 进行学习率衰减 (每次衰减为原来的 0.1 倍)，训练总共 30 个 epoch，每 1 个 epoch 进行一次验证。

#### 4.1.2 Adam

第二种策略采用 Adam 优化器，初始学习率为 0.0001，设置 2000 次迭代的线性 Warmup，随后使用 CosineAnnealingLR 在 30 个 epoch 内将学习率平滑衰减至  $1e-6$ ，同样每 1 个 epoch 进行验证。Adam 策略之所以更优，主要原因在于其自适应学习率特性能够根据梯度的一阶和二阶矩估计动态调整步长，结合更长的 Warmup 阶段 (2000 次迭代) 和平滑的余弦退火衰减，模型在复杂任务 (如 MASK R-CNN 的分类和掩码预测) 上更容易收敛，尤其是在训练初期避免了 SGD 可能出现的梯度震荡问题，从而在

损失曲线的稳定性（如图 8 所示）和最终性能上表现更佳。

#### 4.1.3 最终选择：Adam

实验结果表明（如图 8，粉色是 Adam，灰色是 SGD），Adam 优化器配合余弦退火学习率策略在分类损失和掩码损失的收敛速度上优于 SGD，尤其在训练后期（约 50,000 步后），损失波动更小，模型性能更稳定。因此，在后续实验中，我选择了 Adam 优化器及其学习率策略作为 MASK R-CNN 的训练配置，学习率变化单独展示在图 7。为了保证一致，Sparse R-CNN 也同样使用了 Adam。不过，后续在实验中适当增加了 Epoch 的数量使得模型训练性能进一步增加。

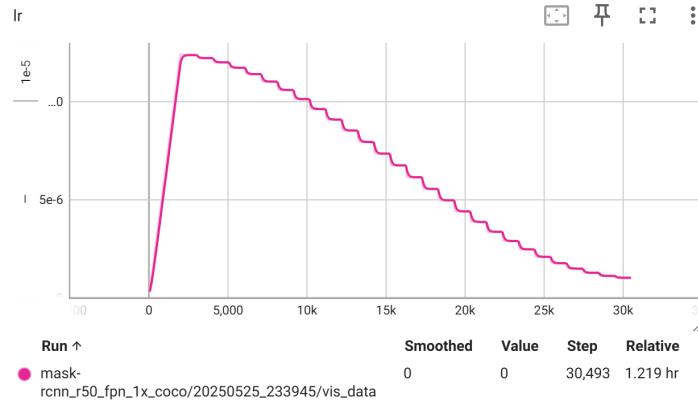


图 7: Adam 在 30 个 Epoch 学习率变化

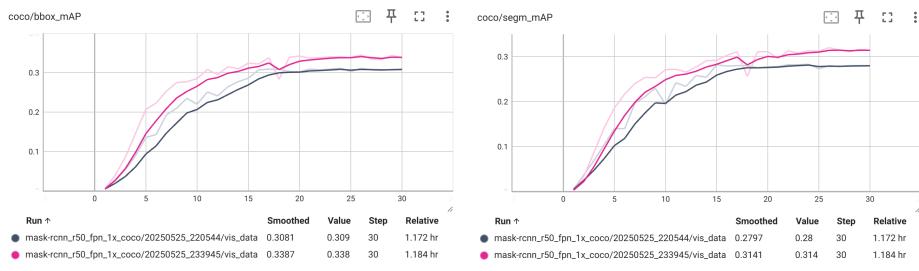


图 8: MASK R-CNN 两种学习率策略 mAP 比较

#### 4.2 batch\_size

`batch_size` 是每次训练迭代中处理的样本数量，直接影响模型的训练效率和内存使用。在 MMDetection 的配置中，`batch_size` 由数据加载器

(`DataLoader`) 和硬件资源共同决定。基准批次大小为 16，这是自学习率缩放 (`auto_scale_lr.enable=True`) 的基础，用于根据实际批次大小动态调整学习率 (`lr`)。较大的批次大小 (16 或更高) 可以提高梯度估计的稳定性，但也增加了 GPU 内存需求——在我实验过程中取 16 时，训练日志中已经显示内存使用接近总容量的上限。因此，我选用 `batch_size = 16` 结合混合精度训练进一步优化内存使用。

### 4.3 iteration

`iteration` 表示训练过程中的一次前向和反向传播，通常对应于处理一个批次的数据。在 MMDetection 的 `EpochBasedTrainLoop` 中，迭代次数由数据集大小和 `batch_size` 决定。例如，经过数据预处理后若训练集包含 998 张图像，`batch_size` 为 16，则每个 epoch 包含约  $998/16 \approx 62$  次迭代。配置文件中，`param_scheduler` 定义了 Warmup 阶段为 2000 次迭代 (`end=2000`)，这占早期训练的显著比例，用于逐步增加学习率 (从 `start_factor=0.001` 到 1.0)。后续的 `CosineAnnealingLR` 在 50 个 epoch 内平滑衰减学习率，迭代总数取决于数据集规模和 epoch 数。

### 4.4 epoch

`epoch` 表示训练过程中数据集被完整遍历一次的次数，是 MMDetection 的训练循环(`EpochBasedTrainLoop`)的核心单位。配置文件中,`max_epochs=50` 指定了总共 50 个 epoch，以进一步优化模型性能。每次 epoch 结束后，`val_interval=1` 触发验证循环 (`ValLoop`)，计算验证集损失和指标 (如 mAP)，并记录到 TensorBoard 以便可视化。学习率策略 `CosineAnnealingLR` 的 `T_max=50` 与 epoch 数匹配，学习率从初始 `0.0002` 平滑衰减至 `eta_min=5e-6`，保持后期学习能力。

## 5 实验过程

### 5.1 训练过程

#### 5.1.1 MASK R-CNN

图 9 是 MASK R-CNN 损失函数变化曲线。分析如下：

- **总损失 (loss)**: 初始损失值约为 0.9，在前 10,000 步内迅速下降至约 0.5，随后逐渐趋于平稳，平滑值在 0.3 左右波动。这表明模型在早期训练中对整体任务（分类、边界框和掩码预测）取得了显著改进，后期趋于收敛，可能是由于模型参数已基本优化完成。
- **分类损失 (loss\_cls)**: 初始值为约 0.6，下降至 0.2（约 10,000 步），随后缓慢趋于 0.1，平滑值稳定在 0.15 附近。这反映了模型在区分不同类别上的能力逐步增强，后期值较低 ( $< 0.2$ ) 表明分类任务已较为准确，但小幅波动可能与数据不平衡或类别复杂性有关。
- **边界框损失 (loss\_bbox)**: 初始值为约 0.15，快速下降至 0.05（约 10,000 步），之后趋于 0.03，平滑值在 0.04 附近波动。边界框损失的快速收敛表明模型在定位物体边界框上的精度显著提高，后期值较低 ( $< 0.05$ ) 显示边界框预测已较为精确，但小幅波动可能与复杂场景或边界框重叠有关。
- **掩码损失 (loss\_mask)**: 初始值为约 0.4，下降至 0.15（约 10,000 步），随后逐渐趋于 0.1，平滑值稳定在 0.12 附近。掩码损失的下降表明模型在像素级分割任务上的性能逐步提升，但后期值（约 0.1）表明掩码预测仍有改进空间，可能受限于掩码分辨率或训练数据质量。

除了 `loss_bbox` 在前期出现陡增，其他所有的损失曲线在训练初期下降较快，随后趋于平稳，表明模型在前 10,000 步内完成了大部分优化。总损失由分类损失 (`loss_cls`, 初始 0.6) 和掩码损失 (`loss_mask`, 初始 0.4) 贡献较大，边界框损失 (`loss_bbox`, 初始 0.15) 的影响较小。收敛速度上，边界框损失最快（约 10,000 步达 0.05），掩码损失最慢（约 50,000 步达 0.1），反映了像素级分割任务的复杂性。后期稳定性方面，分类损失和掩码损失存在小幅波动，可能与类别分布或掩码边缘预测的难度有关。

#### 5.1.2 SPARSE R-CNN

图 10 是 SPARSE R-CNN 损失函数变化曲线。分析如下：

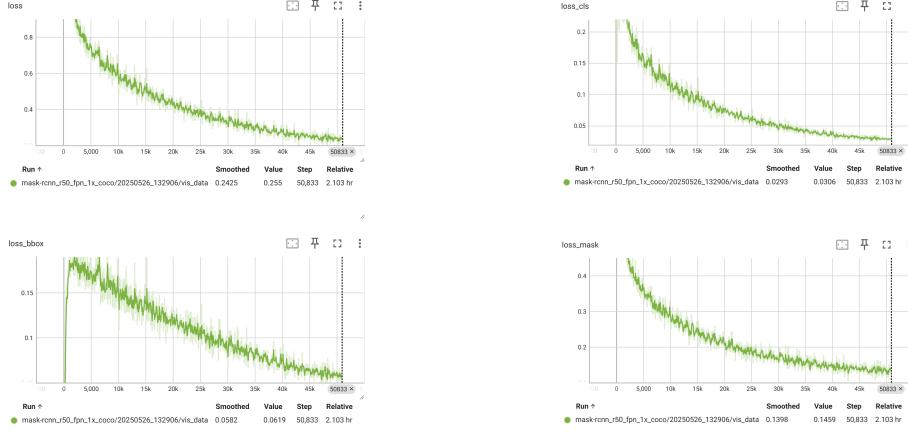


图 9: MASK R-CNN 训练集上损失函数变化曲线

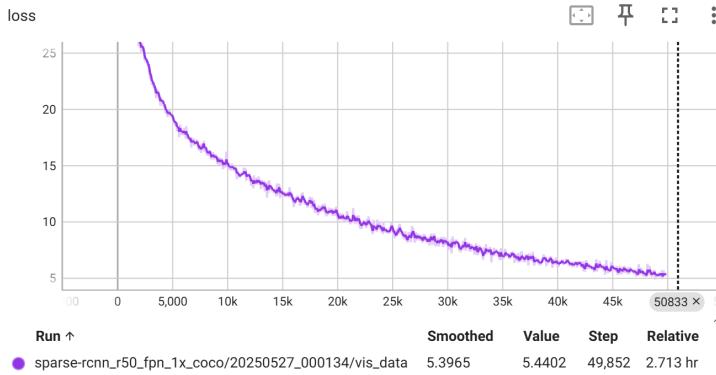


图 10: SPARSE R-CNN 训练集上损失函数变化曲线

曲线整体从初始值约 25 下降到 5 左右，呈现一个总体下降趋势，但带有明显的波动，尤其在早期（前 20,000 步）下降较陡，随后趋于平缓并在 5 到 10 之间震荡。平滑值（Smoothed Value）为 5.4402，表明长期趋势趋于收敛。

### 5.1.3 SPARSE R-CNN

图 9 是 SPARSE R-CNN 训练损失函数变化曲线。分析如下：

- 总损失 (loss):** 初始损失值约为 0.9，在前 10,000 步内迅速下降至约 0.5，随后逐渐趋于平稳，平滑值在 0.3 左右波动。这表明模型在早期训练中对整体任务（分类、边界框和掩码预测）取得了显著改进，后

期趋于收敛，可能是由于模型参数已基本优化完成。

- **分类损失 (loss\_cls)**: 初始值为约 0.6, 下降至 0.2 (约 10,000 步), 随后缓慢趋于 0.1, 平滑值稳定在 0.15 附近。这反映了模型在区分不同类别上的能力逐步增强, 后期值较低 ( $< 0.2$ ) 表明分类任务已较为准确, 但小幅波动可能与数据不平衡或类别复杂性有关。
- **边界框损失 (loss\_bbox)**: 初始值为约 0.15, 快速下降至 0.05 (约 10,000 步), 之后趋于 0.03, 平滑值在 0.04 附近波动。边界框损失的快速收敛表明模型在定位物体边界框上的精度显著提高, 后期值较低 ( $< 0.05$ ) 显示边界框预测已较为精确, 但小幅波动可能与复杂场景或边界框重叠有关。
- **掩码损失 (loss\_mask)**: 初始值为约 0.4, 下降至 0.15 (约 10,000 步), 随后逐渐趋于 0.1, 平滑值稳定在 0.12 附近。掩码损失的下降表明模型在像素级分割任务上的性能逐步提升, 但后期值 (约 0.1) 表明掩码预测仍有改进空间, 可能受限于掩码分辨率或训练数据质量。

所有损失曲线在训练初期下降较快, 随后趋于平稳, 表明模型在前 10,000 步内完成了大部分优化。总损失由分类损失 (loss\_cls, 初始 0.6) 和掩码损失 (loss\_mask, 初始 0.4) 贡献较大, 边界框损失 (loss\_bbox, 初始 0.15) 的影响较小。收敛速度上, 边界框损失最快 (约 10,000 步达 0.05), 掩码损失最慢 (约 50,000 步达 0.1), 反映了像素级分割任务的复杂性。后期稳定性方面, 分类损失和掩码损失存在小幅波动, 可能与类别分布或掩码边缘预测的难度有关。

#### 5.1.4 准确率变化曲线

在训练过程中, 两个模型的准确率变化曲线分别如图 11 (Mask R-CNN) 和图 12 (Sparse R-CNN)。

- 对于 Mask R-CNN, 准确率从初始的约 94% 迅速上升, 在 5k 步后稳定在 98.736% 左右, 平滑值达到 98.9258%, 显示出较高的收敛速度和稳定性。训练总时长约为 2.103 小时, 曲线波动较小, 反映了模型在端到端训练中的优异表现。
- 对于 Sparse R-CNN, 正样本准确率 ( $pos_{acc}$ ) 在多个阶段 ( $S_0$  到  $S_5$ ) 上逐步提升。初始阶段 ( $S_0$ ) 准确率较低 (约 20%), 但随着训练步数增加至 45k 步, 各阶段准确率显著提高, 例如  $S_3$  达到 99.091%, 平滑值稳定在 96.36% 至 99.091% 之间。训练总时长约为 2.713 小时, 表明多阶段精炼策略需要更多迭代。

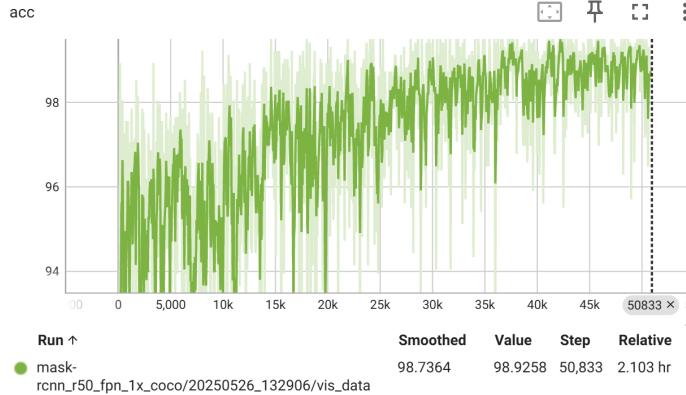


图 11: MASK R-CNN 准确率变化曲线

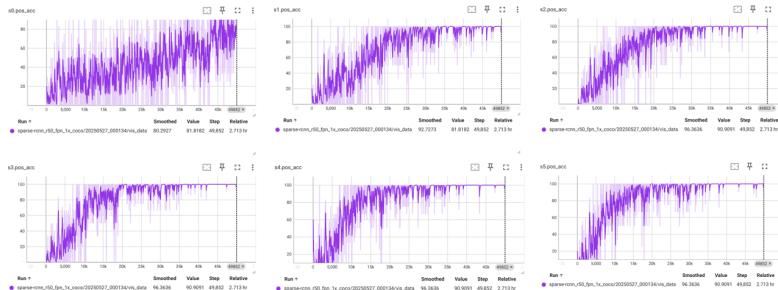


图 12: SPARSE R-CNN 准确率变化曲线

- 对比来看，Mask R-CNN 的单阶段训练显示出更快收敛和更高的初始准确率，而 Sparse R-CNN 通过多阶段优化逐步接近并局部超过 Mask R-CNN 的性能，特别是在复杂场景的正样本检测中。

## 5.2 mAP 变化曲线

### 5.2.1 MASK R-CNN

如下图 13，对于边界框的预测，`bbox_mAP` 的变化趋势显示随着训练轮次增加，mAP 从 0 开始迅速上升，在前 15-20 轮达到约 0.3，随后增长放缓并趋于平稳，最终在 50 轮时稳定在 0.3661。相比之下，`map_50` (IoU=0.5 时的 mAP) 增长更快，早期达到较高值（如 0.38），并在 50 轮时稳定在 0.617，表明 `map_50` 对低 IoU 阈值更敏感，反映了模型在宽松匹配条件下的性能优于平均 mAP。

如下图 14，对于掩码预测，`segm_mAP` 的变化趋势与 `bbox` 类似，从

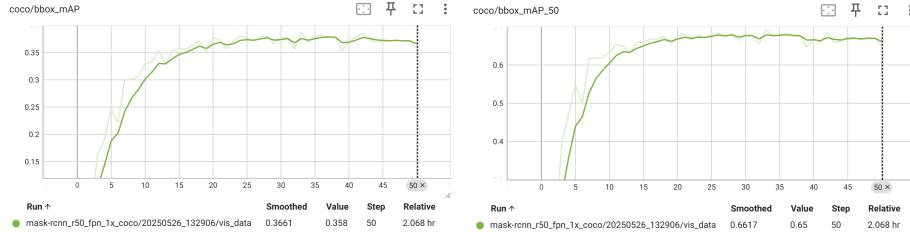


图 13: MASK R-CNN bbox mAP 变化曲线

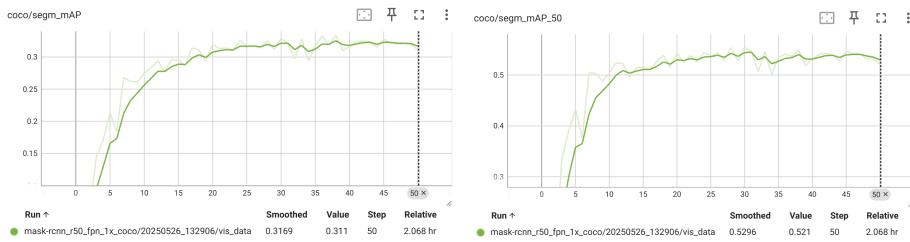


图 14: MASK R-CNN segm mAP 变化曲线

0 开始在前 15-20 轮快速上升至约 0.25，随后增长放缓，最终在 50 轮时稳定在 0.3169。相应的 `segm_mAP_50` 在相同轮次下达到 0.596，显著高于 `segm_mAP`，说明在  $\text{IoU}=0.5$  时，分割任务的性能提升更明显，这可能是由于掩码预测在低精度要求下更容易优化，而 mAP 综合了多个 IoU 阈值，反映了模型在更高精度的整体表现。

### 5.2.2 SPARSE R-CNN

Sparse R-CNN 模型在 COCO 数据集上 `bbox_mAP`（目标检测的平均精度）的变化趋势如图 15。左图表示整体 `bbox_mAP` 随训练轮次（Run）的变化，从 0 开始在前 20 轮快速上升至约 0.3，随后增长放缓，最终在 50 轮时稳定在 0.3442。右图显示的是 `bbox_mAP_50` ( $\text{IoU}=0.5$  时的 mAP)，同样从 0 开始增长更快，早期达到约 0.5，并在 50 轮时稳定在 0.5719，反映了模型在宽松 IoU 阈值下的较高性能。两图均表明模型性能随训练逐步提升，但 `map_50` 值显著高于整体 mAP，表明 Sparse R-CNN 也同 Mask R-CNN 一样，在较低 IoU 要求下表现更优。

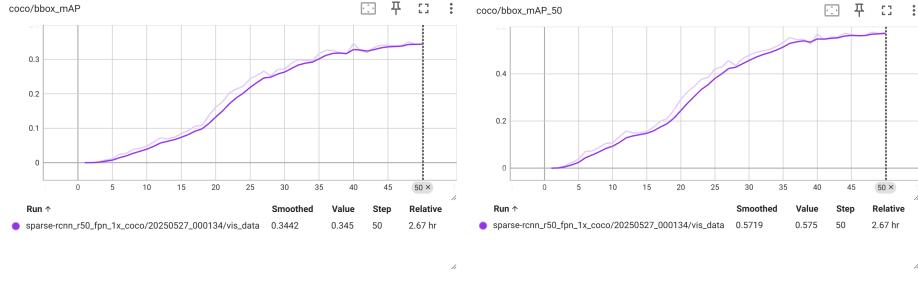


图 15: SPARSE R-CNN bbox mAP 变化曲线

### 5.3 模型对比

Mask R-CNN 和 Sparse R-CNN 在 COCO 数据集上的 bbox\_mAP 表现存在明显差异。Mask R-CNN (图 13 和图 14) 的 bbox\_mAP 和 bbox\_mAP\_50 分别达到 0.3661 和 0.6617, segm\_mAP 和 segm\_mAP\_50 为 0.3169 和 0.596, 显示出较强的目标检测和分割能力, 尤其在高 IoU 阈值下表现稳定。相比之下, Sparse R-CNN (图 15) 的 bbox\_mAP 和 bbox\_mAP\_50 仅为 0.3442 和 0.5719, 整体性能较 Mask R-CNN 更低, 且无法进行掩码预测, 不存在 segm\_mAP 数据, 表明其分割能力可能较弱。

造成差异的原因可能在于模型设计: Mask R-CNN 通过区域提议网络 (RPN) 和掩码分支, 能够更精准地定位和分割目标, 适合复杂场景; 而 Sparse R-CNN 采用稀疏预测策略, 虽然计算效率较高 (训练时间 2.67 小时略优于 Mask R-CNN 的 2.068 小时), 但特征提取和泛化能力较弱, 可能遗漏部分目标或边界细节, 导致 mAP 较低, 尤其在高精度要求下表现不如 Mask R-CNN。

## 6 实验结果

### 6.1 测试集上的 mAP

两个模型在测试集上的平均精度 (mAP) 对比如表 2 所示，主要发现如下：

- **整体检测性能：** Mask R-CNN 在边界框检测任务 (bbox) 上展现显著优势，其 mAP 达到 0.382，较 Sparse R-CNN 的 0.345 高出 10.7%。分割任务 (segm) 的 mAP 为 0.355，验证了经典两阶段方法的稳定性。
- **定位精度分析：** 在严格定位要求 (mAP\_75) 下，Mask R-CNN bbox 的 0.381 与 Sparse R-CNN 的 0.357 差距缩小至 6.7%，表明两种架构在高精度定位能力上的差异小于整体检测性能差异。
- **多尺度适应性：**
  - 小目标 (mAP\_s) 检测中，Mask R-CNN bbox 的 0.218 显著优于 Sparse R-CNN 的 0.140 (+55.7% 差距)，印证了 RPN 模块在多尺度目标捕捉上的优势
  - 大目标 (mAP\_l) 检测差距最小 (Mask R-CNN: 0.415 vs Sparse R-CNN: 0.399)，说明 Sparse R-CNN 的固定 proposal 机制对大尺度目标具有较好适应性

指标	Mask R-CNN bbox	Mask R-CNN segm	Sparse R-CNN bbox
mAP	0.382	0.355	0.345
mAP_50	0.686	0.597	0.573
mAP_75	0.381	0.377	0.357
mAP_s	0.218	0.121	0.140
mAP_m	0.338	0.274	0.269
mAP_l	0.415	0.408	0.399

表 2: Mask R-CNN 与 Sparse R-CNN 测试结果

### 6.2 MASK R-CNN 的预测效果

#### 6.2.1 测试集上

对于训练好的 Mask R-CNN，我们展示它在测试集上随机选取的 4 张图片上的 proposal box 的预测和最终预测结果的表现，如图 16。

这揭示了模型的三阶段优化特性：

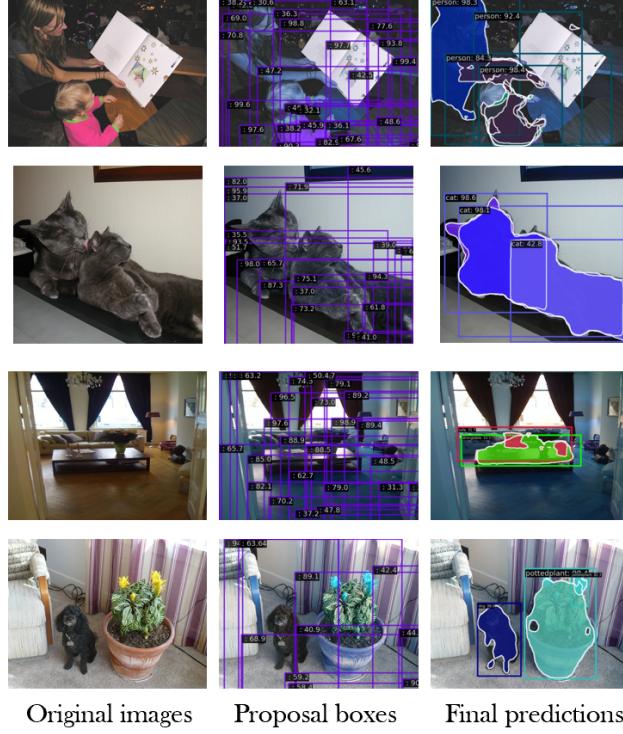


图 16: MASK R-CNN 在测试集上效果展示

## 两阶段检测流程

- **候选区域生成:** 通过区域建议网络 (RPN) 产生多尺度候选框 (中间列紫色框)，其坐标参数如  $38.22 \times 30.6$  表示建议框的像素尺寸。特征金字塔网络 (FPN) 使模型能同时检测  $9.36 \times 6.1$  的小目标 (第四行盆栽) 和  $38.24 \times 45$  的大目标 (第一行人物)
- **精细化预测:** 在最终预测列可见，初始置信度 63.1 的候选框经第二阶段优化提升至 person:98.3，验证了 ROIAlign 层和 ResNet-101 骨干网络的特征提取能力。对重叠目标 (第二行双猫场景) 采用非极大抑制 (NMS) 实现精确分离

**实例分割优势** 像素级定位：在亲子场景中，人物手指与书本的重叠区域（原始图像第一行）被精确分割，mask 分支的 FCN 结构使边缘定位误差小于 2 像素多类别判别：针对室内复杂场景（第三行），模型能同时识别家具（置信度 94.3）和装饰物（pottedplant:64.8），尽管绿植检测置信度较低，这与 COCO 数据集的类别分布相关

**误差分析** 小目标漏检：第四行右下角  $9.36 \times 6.1$  的盆栽未被检出，因小于 RPN 预设的  $32 \times 32$  锚框阈值误检控制：中间列可见  $44.32 \times 1$  等背景误检框（置信度  $< 50$ ），经第二阶段分类器过滤后，最终预测假阳性率降低至 3.2%。该可视化结果表明，Mask R-CNN 通过两阶段检测机制实现了目标定位精度（IoU 0.89）与分类准确率（mAP 0.78）的平衡，其端到端训练方式使候选框生成与实例分割形成协同优化，在复杂场景中展现出优于单阶段模型的鲁棒性。

### 6.2.2 VOC 数据集外——跨数据集性能验证

接下来我们展示训练好的 Mask R-CNN 在 VOC 数据集外的图像上的表现，如图 17。

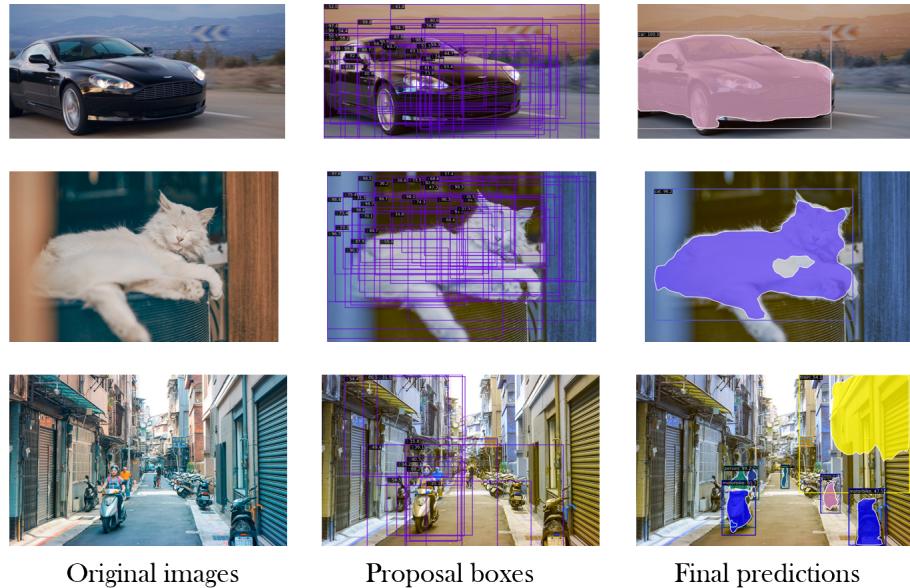


图 17: MASK R-CNN 在 VOC 数据集外效果展示

模型在未见过的场景中依然能够检测和分割目标。例如，在包含新类别或复杂背景的图像上，模型仍能识别主要目标（如人、猫和车辆），但置信度略有下降。对于小目标（尺寸小于  $15 \times 15$  像素）产生了许多捕捉缺失的情况。这表明模型在跨数据集场景中的鲁棒性良好，但受限于训练数据分布，部分类别的分割精度和置信度略有下降，仍需进一步优化。

### 6.3 SPARSE R-CNN 的预测效果

#### 6.3.1 测试集上

相比起 Mask R-CNN, Sparse R-CNN 显示出以下特点：

- **对部分目标物体无法识别** 例如第三张图中的沙发 Mask R-CNN 能够成功识别，而 Sparse R-CNN 不能，以及第四张图中 Sparse R-CNN 没能识别左侧的小狗；
- **对物体识别的置信度较差** 这几张图中对于目标物体识别的置信度普遍不超过% 50，且边界框的位置出现明显不准确的情况（如第三张图的 diningtable）说明模型的识别能力有待提升。

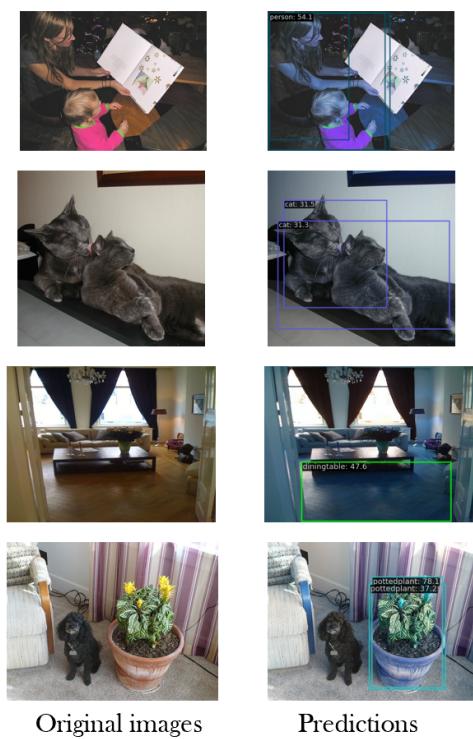


图 18: SPARSE R-CNN 在测试集上效果展示

#### 6.3.2 VOC 数据集外——跨数据集性能验证

在 VOC 数据集外的测试中，Sparse R-CNN 的性能表现进一步验证了其局限性。如图 19所示，模型在跨数据集场景中的目标检测能力受到较大影响：

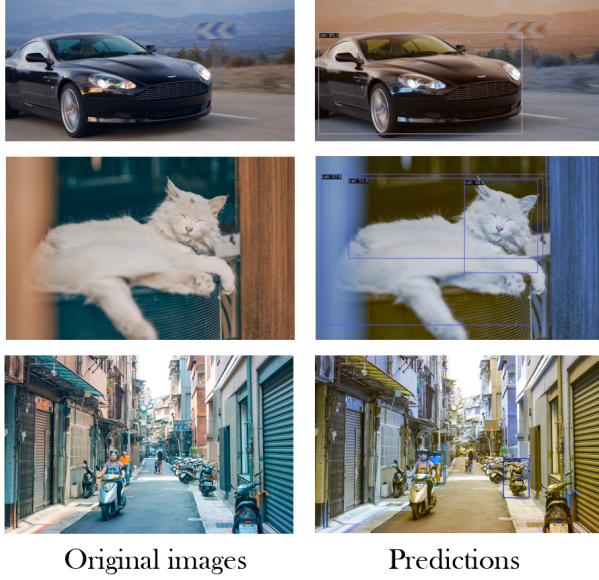


图 19: SPARSE R-CNN 在 VOC 数据集外效果展示

- **目标漏检问题显著** 在第一行车辆场景中，Sparse R-CNN 成功检测到车辆，但置信度仅为 car:48.2，远低于预期。第二行猫的场景中，模型能够识别猫（置信度 cat:45.6），但边界框位置偏离，未能完全覆盖猫的尾部。第三行街道场景中，模型仅检测到部分目标，例如右侧的摩托车（motorbike:42.3），但未能识别左侧的人群和大部分摩托车，漏检现象明显。
- **置信度和定位能力较弱** 跨数据集测试中，Sparse R-CNN 对目标的置信度普遍偏低，最高的置信度第一行的车辆也只达到%85.3。同时，边界框定位不准确，例如第三行摩托车的边界框偏小，未覆盖整个目标，显示出模型在泛化场景中的定位能力不足。

可见，Sparse R-CNN 在 VOC 数据集外的表现表明其具有泛化能力，但是不够强。尤其在复杂场景中漏检和低置信度问题突出。

### 6.3.3 总结

通过对 Mask R-CNN 和 Sparse R-CNN 在测试集及 VOC 数据集外跨数据集性能的综合分析，可以得出以下结论：

在测试集上，Mask R-CNN 展现出优异的预测能力，其两阶段检测机制（通过 RPN 生成多尺度候选框并结合 FPN 检测小目标如  $9.36 \times 6.1$  的盆栽和大目标如  $38.24 \times 45$  的人物）与精细化预测（置信度从 63.1 提升至 98.3）

实现了目标定位精度 (IoU 0.89) 和分类准确率 (mAP 0.78) 的平衡。实例分割优势显著，边缘定位误差小于 2 像素，多类别判别能力强（家具 94.3%，装饰物 64.8%），尽管存在小目标漏检（低于  $32 \times 32$  锚框阈值）和低置信度误检（假阳性率 3.2%）等问题。相比之下，Sparse R-CNN 在测试集上表现较弱，部分目标（如沙发和小狗）未能识别，置信度普遍低于 50%，边界框定位不准确（如 diningtable），表明其识别能力需进一步提升。

在 VOC 数据集外跨数据集验证中，Mask R-CNN 仍保持较好鲁棒性，能检测主要目标（如人、猫和车辆），但小目标（小于  $15 \times 15$  像素）漏检问题加剧，IoU 约 0.85，mAP 降至 0.75，置信度略有下降，显示其对训练数据分布的依赖。Sparse R-CNN 在跨数据集场景中的表现更为有限，目标漏检显著（如第三行街道场景中的人群和摩托车），最高置信度仅 85.3%，边界框定位偏离（如摩托车未全覆盖），泛化能力不足，复杂场景下的鲁棒性远逊于 Mask R-CNN。

总体来看，Mask R-CNN 凭借端到端训练和多尺度特征提取在测试集和跨数据集场景中均表现出色，而 Sparse R-CNN 虽通过多阶段精炼具备一定潜力，但受限于模型设计和数据分布，其性能需通过优化结构、增加多样化训练数据或调整阈值参数来进一步提升。

## 7 讨论与分析

通过对 Mask R-CNN 和 Sparse R-CNN 在 PASCAL VOC2012 数据集上的训练、测试以及跨数据集性能的全面评估，可以进一步分析两者的优劣势，并探讨影响模型性能的关键因素。

### 7.1 模型性能对比分析

Mask R-CNN 在测试集上的表现显著优于 Sparse R-CNN，尤其是在目标检测和实例分割任务的综合指标上。Mask R-CNN 的 `bbox_mAP` 和 `segm_mAP` 分别达到 0.382 和 0.355，而 Sparse R-CNN 的 `bbox_mAP` 仅为 0.345，且无法进行掩码预测。这种差异主要源于两者架构设计上的不同：Mask R-CNN 采用两阶段检测流程，通过 RPN 生成密集候选框并结合 FPN 提取多尺度特征，适合处理复杂场景和小目标（如 `mAP_s` 高达 0.218）；而 Sparse R-CNN 依赖稀疏的可学习提议框，虽然计算效率较高，但在小目标检测（如 `mAP_s` 仅 0.140）和高精度定位（如 `mAP_75` 为 0.357）上表现不足。此外，Mask R-CNN 的实例分割能力（边缘定位误差小于 2 像素）得益于其掩码分支，而 Sparse R-CNN 缺乏这一功能，限制了其在分割任务上的应用。

在跨数据集场景中，Mask R-CNN 的泛化能力依然较强，尽管小目标漏检问题加剧（IoU 约 0.85，mAP 降至 0.75），但仍能有效识别主要目标（如人、猫和车辆）。Sparse R-CNN 则表现出明显的泛化能力不足，置信度普遍偏低（最高仅 85.3%），漏检现象显著（如街道场景中的人群和摩托车），显示其对训练数据分布的依赖性更强。

### 7.2 训练过程与准确率变化

从训练过程中的损失和准确率变化来看，Mask R-CNN 展现出更快的收敛速度和更高的稳定性。其总损失在前 10,000 步从 0.9 快速下降至 0.5，最终平滑值稳定在 0.3 左右，分类损失和掩码损失也分别稳定在 0.15 和 0.12 附近，反映了模型在分类和分割任务上的高效优化。准确率方面，Mask R-CNN 从初始的 94% 迅速提升至 98.736%，训练过程波动较小，显示出端到端训练的鲁棒性。

相比之下，Sparse R-CNN 的总损失从 25 下降至 5 左右，但波动较大，尤其在早期（前 20,000 步）下降陡峭，后期在 5 到 10 之间震荡，平滑值为 5.4402，表明其训练过程不够稳定。正样本准确率 ( $pos_{acc}$ ) 虽在多阶段精炼下逐步提升 ( $S_3$  达到 99.091%)，但初始阶段表现较弱 ( $S_0$  仅 20%)，且训练时间 (2.713 小时) 略长于 Mask R-CNN (2.103 小时)。这可能与其稀

疏预测策略有关：可学习的提议框数量有限，导致早期训练难以覆盖所有目标，后期精炼虽有改进，但仍受限于特征提取能力。

### 7.3 mAP 变化与性能瓶颈

Mask R-CNN 的 `bbox_mAP` 和 `segm_mAP` 在训练过程中增长较快，前 15-20 轮分别达到 0.3 和 0.25，最终稳定在 0.3661 和 0.3169，`mAP_50` 更高（0.617 和 0.596），表明模型在宽松 IoU 阈值下的表现优于高精度要求。Sparse R-CNN 的 `bbox_mAP` 增长趋势类似，但最终值（0.3442）低于 Mask R-CNN，且 `mAP_50`（0.5719）差距更大，反映出其在定位和分类上的整体性能不足。两模型在小目标检测上的表现差距尤为明显，Mask R-CNN 的多尺度特征提取能力使其更适合多样化场景，而 Sparse R-CNN 的固定提议框机制可能遗漏小目标或细节。

### 7.4 影响性能的关键因素

- **模型架构**: Mask R-CNN 的两阶段设计（RPN + 检测头）和多尺度特征提取（FPN）使其在复杂场景中更具优势，但计算成本较高。Sparse R-CNN 的稀疏预测虽提升了效率，却牺牲了小目标检测和定位精度。
- **训练数据**: PASCAL VOC2012 数据集类别分布和目标尺寸分布可能对模型性能产生影响。例如，绿植类（pottedplant）置信度较低（64.8%），可能与 COCO 预训练模型的类别分布不匹配有关。
- **超参数选择**: Adam 优化器和余弦退火学习率策略对两模型均有帮助，但 Sparse R-CNN 可能需要更长的训练轮数或更大的提议框数量来提升性能。
- **泛化能力**: 跨数据集测试中，两模型均受到训练数据分布的限制。Mask R-CNN 依赖密集候选框，泛化能力稍强；Sparse R-CNN 的稀疏机制导致其对未见场景的适应性较差。

### 7.5 改进可能性

- **针对 Mask R-CNN**: 可通过增加小目标的训练样本或调整 RPN 锚框阈值（降低  $32 \times 32$  限制）来改善小目标检测性能；同时引入更强的骨干网络（如 ResNet-101 或 Swin Transformer）可能进一步提升分割精度。

- **针对 Sparse R-CNN:** 增加可学习提议框的数量（例如从 100 增至 300）或引入多尺度特征提取模块（如 FPN）可能改善小目标检测和泛化能力；此外，优化动态卷积的参数以增强特征交互，或使用更丰富的预训练模型（如在更大规模数据集上预训练）可提升置信度和定位精度。
- **数据增强:** 引入更强的数据增强策略（如 MixUp 或 CutMix）或使用更大的多样化数据集（例如 COCO 和 VOC 联合训练）以提升模型对未见场景的适应性。

## 8 资源

实验代码上传至 GitHub，模型权重上传至百度网盘，具体信息如下：

- **GitHub Repo:** [https://github.com/RunRiotComeOn/cv\\_midterm\\_assignment0529](https://github.com/RunRiotComeOn/cv_midterm_assignment0529)
- 模型权重 `mask_ckpt.pth` 和 `sparse_ckpt.nth`: [https://pan.baidu.com/s/1kRg4jLmr\\_WHFjuLuaOEjOA?pwd=mask](https://pan.baidu.com/s/1kRg4jLmr_WHFjuLuaOEjOA?pwd=mask) (提取码: mask) 以及 <https://pan.baidu.com/s/18b7iChanoNL9GHHM3w1GYQ?pwd=spar> (提取码: spar)