



**计算机视觉**

**上机实验报告**

|  |  |
| --- | --- |
| 专 业： | 人工智能 |
| 班 级： | 2204 |
| 学 号： | 19220432 |
| 姓 名： | 陆昊宇 |

2025年04月08日

## 一、实验目的

1. 熟悉 Faster R-CNN 的实现细节。

2. 熟悉使用 Faster R-CNN 完成目标检测任务。

## 二、实验内容要求

#### 1. 目标检测实验

（1）实现 Faster R-CNN（参照https://github.com/bubbliiiing/faster-rcnn-pytorch）；

（2）在VOC 2007数据集上测试Faster R-CNN，展示测试结果；

（3）分析并解释frcnn\_training.py中FasterRCNNTrainer()的代码；

（4）分析并解释 rpn.py 中RegionProposalNetwork()的代码；

（5）分析并解释frcnn.py中FasterRCNN()的代码。

（解释每行代码的作用，可参考代码内的注释）

**提交入口：**

https://send2me.cn/w89aK2N5/SyWvsinHvbDABg

## 三、实验报告评分标准

1. 完成目标检测实验（2）（20’）

2. 完成目标检测实验（3）（20’）

3. 完成目标检测实验（4）（20’）

4. 完成目标检测实验（5）（20’）

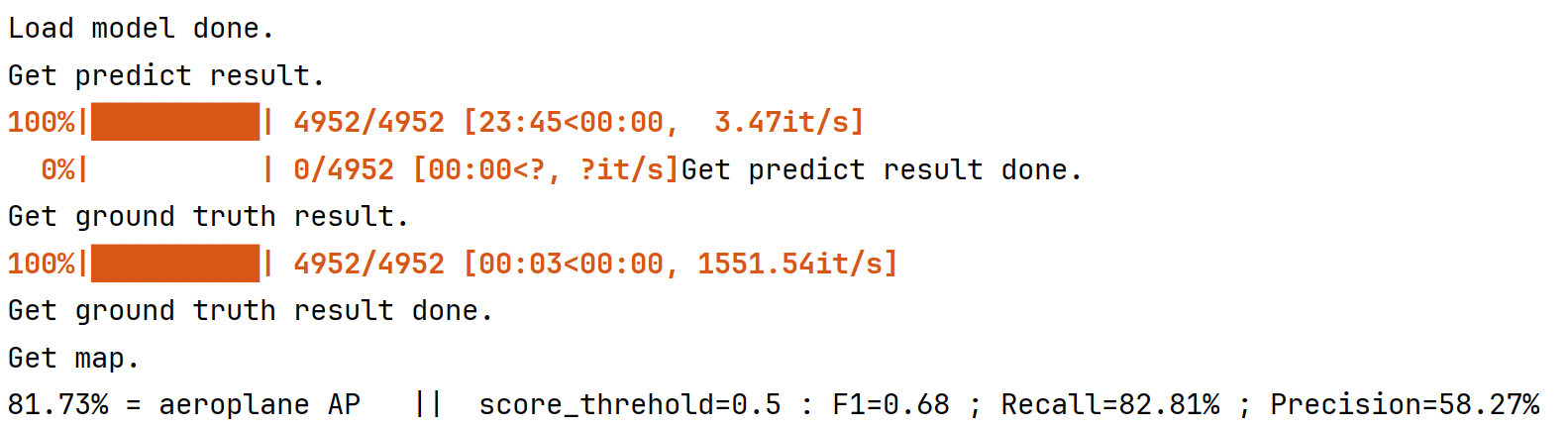
5. 实验过程记录完整，表述逻辑清晰（10’）

6. 排版工整自洽，图表说明完整（10’）

## 四、实验过程

运行结果图：





## 五、实验结论

代码分析与解释

## FasterRCNNTrainer

### 1. 类结构

class FasterRCNNTrainer(nn.Module):  
 def \_\_init\_\_(self, model\_train, optimizer):  
 def \_fast\_rcnn\_loc\_loss(self, pred\_loc, gt\_loc, gt\_label, sigma):  
 def forward(self, imgs, bboxes, labels, scale):  
 def train\_step(self, imgs, bboxes, labels, scale, fp16=False, scaler=None):

* 继承自nn.Module的完整训练模块
* 包含初始化、损失计算、前向传播和训练步骤

### 2. 核心组件

self.anchor\_target\_creator = AnchorTargetCreator() # 生成RPN训练目标  
self.proposal\_target\_creator = ProposalTargetCreator() # 生成ROI训练目标  
self.loc\_normalize\_std = [0.1, 0.1, 0.2, 0.2] # 坐标归一化参数

### 3. 回归损失计算

def \_fast\_rcnn\_loc\_loss(self, pred\_loc, gt\_loc, gt\_label, sigma):  
 # 使用平滑L1损失（Smooth L1 Loss）  
 regression\_loss = torch.where(  
 regression\_diff < (1. / sigma\_squared),  
 0.5 \* sigma\_squared \* regression\_diff \*\* 2, # L2损失部分  
 regression\_diff - 0.5 / sigma\_squared # L1损失部分  
 )

* 对正样本（gt\_label > 0）计算定位损失
* 采用平滑L1损失，结合L1/L2的优点

### 4. 前向传播流程

def forward(...):  
 # 特征提取  
 base\_feature = self.model\_train(imgs, mode='extractor')  
   
 # RPN网络输出  
 rpn\_locs, rpn\_scores, rois, ... = self.model\_train(... mode='rpn')  
   
 # RPN损失计算  
 for i in range(n):  
 gt\_rpn\_loc, gt\_rpn\_label = self.anchor\_target\_creator(...)  
 rpn\_loc\_loss = self.\_fast\_rcnn\_loc\_loss(...)  
 rpn\_cls\_loss = F.cross\_entropy(...)  
   
 # ROI处理  
 sample\_rois, ... = self.proposal\_target\_creator(...)  
   
 # 分类头输出  
 roi\_cls\_locs, roi\_scores = self.model\_train(... mode='head')  
   
 # ROI损失计算  
 for i in range(n):  
 roi\_loc\_loss = self.\_fast\_rcnn\_loc\_loss(...)  
 roi\_cls\_loss = nn.CrossEntropyLoss()(...)

### 5. 训练步骤

def train\_step(...):  
 # 标准精度训练  
 losses = self.forward(...)  
 losses[-1].backward()  
 self.optimizer.step()  
  
 # 混合精度训练（FP16）  
 with autocast():  
 losses = self.forward(...)  
 scaler.scale(losses[-1]).backward()  
 scaler.step(...)

### 6. 损失组成

losses = [  
 rpn\_loc\_loss/n, # RPN定位损失  
 rpn\_cls\_loss/n, # RPN分类损失  
 roi\_loc\_loss/n, # ROI定位损失  
 roi\_cls\_loss/n, # ROI分类损失  
 sum(losses) # 总损失  
]

### 关键设计点

1. **双阶段训练**：同时优化RPN（区域建议网络）和ROI（区域兴趣）分类器
2. **目标生成器**：
   * AnchorTargetCreator：为RPN生成正/负样本和回归目标
   * ProposalTargetCreator：为ROI分类生成训练样本
3. **多任务损失**：同时优化定位（回归）和分类任务
4. **混合精度支持**：通过FP16加速训练
5. **归一化处理**：使用loc\_normalize\_std对坐标偏移量进行标准化

## RegionProposalNetwork

### 1. 类初始化（**init**）

class RegionProposalNetwork(nn.Module):  
 def \_\_init\_\_(self, in\_channels=512, mid\_channels=512, ...):

* **功能**：初始化RPN网络结构
* **核心组件**：
  + anchor\_base：生成基础锚框（9种不同比例/尺度的初始框）
  + conv1：3x3卷积层，用于特征整合（通道数不变）
  + score：1x1卷积，输出锚框的前景/背景分类（2通道/锚框）
  + loc：1x1卷积，输出锚框坐标调整参数（4通道/锚框）
  + proposal\_layer：候选框生成器（包含解码和非极大抑制）

### 2. 前向传播（forward）

def forward(self, x, img\_size, scale=1.):

* **输入**：
  + x：来自骨干网络的特征图（如ResNet/VGG的输出）
  + img\_size：原始图像尺寸
  + scale：图像预处理时的缩放比例
* **处理流程**：
  1. **特征整合**：通过3x3卷积 + ReLU
  2. **预测输出**：
     + rpn\_locs：锚框坐标调整参数（dx, dy, dw, dh）
     + rpn\_scores：锚框前景/背景分类得分
  3. **生成候选框**：
     + 通过\_enumerate\_shifted\_anchor生成所有位置的锚框
     + 使用ProposalCreator解码预测参数，应用NMS生成最终候选框

### 3. 关键实现细节

* **锚框生成**：
  + 基础锚框通过ratios和anchor\_scales参数生成9种组合
  + 使用feat\_stride将锚框平铺到特征图每个空间位置
  + 例如600x600输入图像，在stride=16的特征图上生成约20000个锚框
* **预测输出处理**：
* rpn\_locs = rpn\_locs.permute(0, 2, 3, 1).contiguous().view(n, -1, 4)  
  rpn\_scores = rpn\_scores.permute(0, 2, 3, 1).contiguous().view(n, -1, 2)
* 将卷积输出转换为(batch\_size, num\_anchors, 4/2)的形状，便于后续处理
* **候选框生成**：
* roi = self.proposal\_layer(rpn\_locs[i], rpn\_fg\_scores[i], anchor, img\_size)
* 使用预测的调整参数修正锚框坐标，通过NMS筛选出高质量候选框（通常保留2000个）

### 4. 输出结果

* rpn\_locs：所有锚框的坐标调整参数（用于RPN损失计算）
* rpn\_scores：所有锚框的分类得分（用于RPN损失计算）
* rois：最终生成的候选区域坐标（格式：[x\_min, y\_min, x\_max, y\_max]）
* roi\_indices：对应候选区域的批次索引
* anchor：生成的所有原始锚框坐标

### 5. 设计特点

* **全卷积结构**：可处理任意尺寸输入
* **参数共享**：所有位置共享相同的锚框生成和预测参数
* **高效处理**：通过矩阵运算批量处理所有锚框
* **多尺度检测**：通过不同尺度的锚框实现多尺度目标检测

## FasterRCNN

1. **双主干支持**：
   * 支持VGG16和ResNet50两种主干网络
   * 不同主干的特征维度不同（VGG:512，ResNet50:1024）
   * 对应不同的ROI池化尺寸（VGG:7x7，ResNet50:14x14）
2. **模块化设计**：
   * 特征提取器（extractor）：负责图像特征提取
   * RPN网络（rpn）：生成候选区域建议
   * ROI头部（head）：实现分类和边界框回归
3. **多模式前向传播**：
   * forward: 完整推理流程
   * extractor: 仅特征提取（可用于特征可视化）
   * rpn: 单独运行区域建议网络（用于RPN训练）
   * head: 单独运行分类头部（用于微调）
4. **实现细节**：
   * 使用feat\_stride处理特征图与原始图像的尺度关系
   * 锚框设计支持多尺度和多种宽高比
   * 分类数包含背景类（num\_classes + 1）
5. **训练优化**：
   * freeze\_bn()方法用于冻结批量归一化层
   * 预训练权重支持（pretrained参数）