YOLOv8_X-ray危险品检测系统详解

一、项目架构全景图

```
YOLOv8_X-ray危险品检测系统
├── 数据准备
│   ├── data_processing.py (数据格式转换)
│   └── yolo_dataset/    (标准化数据集)
│       ├── images/        (训练/测试图像)
│       └── labels/        (标注文件)
├── 模型训练
│   ├── train.py           (训练脚本)
│   ├── pre_train_models/ (预训练权重)
│   └── trained_model/     (训练输出)
├── 推理部署
│   ├── x_ray_windows.py  (PyQt5图形界面)
│   └── yolo11n.pt         (最终部署模型)
└── 环境依赖
    └── requirements.txt  (依赖库清单)
```

二、核心文件详解

数据处理模块 data_processing.py

```
# 数据集路径配置
data_path = r"原始数据集路径"
yolo_dataset_path = r"YOLO标准数据集路径"

# 类别定义（8类危险品）
classes = [
    'Portable_Charger_1', 'Portable_Charger_2', # 充电宝
    'Mobile_Phone',      # 手机
    'Cosmetic',          # 化妆品
    'Nonmetallic_Lighter', # 非金属打火机
    'Water',             # 液体
    'Tablet',            # 平板电脑
    'Laptop'             # 笔记本电脑
]

# 标签格式转换（坐标归一化）
def convert_label_format(label_path, image_width, image_height):
    # 实现坐标转换逻辑：像素坐标→YOLOv8归一化坐标
    # 返回标准化后的标签数据
```

模型训练模块 train.py

```
# 训练参数配置
train_params = {
```

```
    'data': 'yolo_dataset/xray_data.yaml', # 数据集配置
    'epochs': 100,          # 训练轮次
    'batch': 4,             # 批量大小（根据显存调整）
    'imgsz': 640,           # 输入尺寸
    'device': 'cuda',       # 设备选择
    'optimizer': 'AdamW',   # 优化器类型
    'lr0': 0.001,           # 初始学习率
    'augment': True,        # 数据增强开关
    'mosaic': 0.75,         # 马赛克增强概率
    'project': 'trained_models', # 输出目录
    'name': 'xray_detection_v1' # 实验名称
}

# 训练流程
model.train(**train_params) # 启动训练
model.val()                 # 验证集评估
model.predict()             # 推理示例
model.export()              # 模型导出（ONNX/TensorRT）
```

推理部署模块 x_ray_windows.py

```
class DetectionUI(QWidget):
    def __init__(self):
        # 加载模型
        self.model = YOLO('trained_model/weights/best.pt')
        self.initUI()  # 初始化界面

    def open_image(self):  # 图像加载
        # 实现图像选择对话框

    def process_images(self):  # 图像处理
        results = self.model(img)  # 执行检测
        # 处理检测结果
        for box in results[0].boxes:
            cls = results[0].names[int(box.cls)]  # 获取类别
            conf = float(box.conf)                # 获取置信度
            xmin, ymin, xmax, ymax = box.xyxy[0]  # 获取坐标

    def save_results(self):  # 结果保存
        # 将检测结果保存为Excel文件
```

三、YOLOv8工作原理详解

网络架构特性：

单阶段检测器（Single-stage）

主干网络：CSPDarknet8

检测头：Decoupled Head（解耦检测头）
特征金字塔：PANet（Path Aggregation Network）

数据处理流程：

原始图像 → 填充/缩放 → 增强处理 → 输入网络
→ 输出 [x_center, y_center, width, height, conf, class_probs]
→ NMS后处理 → 可视化结果

关键技术点：

- Anchor-free机制：直接预测边界框坐标
- 动态标签分配：根据IoU动态匹配正负样本
- 任务解耦：分类和检测分支分离
- 自适应训练策略：根据设备自动调整参数

四、开发环境配置指南

环境依赖安装

```
# 创建虚拟环境
conda create -n x-ray python==3.10
# 激活环境
conda activate x-ray
# 安装依赖
pip install -r requirements.txt
```

数据集准备流程：

```
原始数据集
├── train/
│   ├── train_image/      (训练图像)
│   └── train_annotation/ (训练标注)
└── test/
    ├── test_image/       (测试图像)
    └── test_annotation/  (测试标注)
```

执行数据转换
python data_processing.py

模型训练流程：

```
# 启动训练
python train.py
# 训练输出结构
trained_models/
└── xray_detection_v1/
    ├── weights/
    │   ├── best.pt      (最佳模型)
    │   └── last.pt      (最终模型)
    └── results.csv      (训练结果)
```

应用部署流程：

```
# 启动检测应用
python x_ray_windows.py
# 模型导出
model.export(format='onnx')  # 导出ONNX模型
```

五、常见问题解决方案

CUDA内存不足：

- 调整train.py中的batch_size参数
- 减小输入尺寸imgsz至320-416
- 启用混合精度训练--amp True

检测精度低：

- 调整x_ray_windows.py中的conf_threshold置信度阈值
- 增加数据增强种类（旋转、透视变换）
- 使用更复杂的模型变体（yolov8s.pt/yolov8m.pt）

路径配置错误：

- 检查data_processing.py中的路径配置
- 验证train.py的data_yaml_path是否正确
- 确保x_ray_windows.py模型路径正确

六、项目扩展建议

性能优化方向：

- 添加TensorRT加速支持
- 实现多线程数据预处理
- 集成OpenVINO加速推理

功能扩展建议：

- 添加视频流检测功能
- 实现检测结果数据库存储
- 增加异常模式分析模块

工程改进方向：

- 添加训练参数可视化面板
- 实现自动超参数优化
- 构建持续集成（CI）流程

本教程完整覆盖了从环境搭建、数据预处理、模型训练到最终部署的完整流程，每个核心模块都有对应的代码实现和原理说明。对于技术小白，建议按照以下顺序实践：

1. 先运行已提供的预训练模型（yolo11n.pt）
2. 理解数据预处理流程（data_processing.py）
3. 尝试微调训练（train.py）
4. 最后进行界面应用开发（x_ray_windows.py）