项目需求理解文档

一、项目背景

为提升交通图像处理效率，本项目基于 Agentic Object Detection API（参考吴恩达目标检测技术框架），开发一套数据管道（Data Pipeline）。该管道需实现指定流程和目标，解决传统单线程处理速度慢、结果分散的问题。

二、业务目标

1. 用 Python 开发一个数据处理管道，实现交通图片的批量读取、并行处理、预测生成、和结果分析。
2. 提升图像识别的效率，满足实际应用场景的需求。

三、功能需求

1. **图像读取模块**
    - 支持从指定目录或数据源批量读取 JPEG/PNG 格式图片（适配不同分辨率与光照条件）
2. **多线程 API 调用模块**
    - 并行调用目标检测 API，处理图像数据
    - 准确检测交通场景核心目标（车辆、行人、交通标志、信号灯）
3. **生成预测结果模块**
    - 收集并格式化 API 返回的预测结果
4. **整理分析模块**
    - 对预测结果进行统计分析和可视化展示

四、团队分工

- 优先级 1：API 调用与多线程实现
- 优先级 2：数据读取与结果存储
- 优先级 3：结果分析与可视化

环境配置说明

操作系统：Windows 11

Python: 版本 3.13

IDE: PyCharm Community 2025.1.1

项目环境：使用 PyCharm 创建的独立虚拟环境（venv）

已安装的依赖库：

- requests：用于发送 HTTP 请求，调用目标检测 API
- opencv-python：用于读取和预处理图像数据
- pandas：用于数据整理与分析

Agentic Object Detection API 调研报告

一、API 简介

Agentic Object Detection 是由 LandingAI 推出的基于推理驱动的目标检测技术。用户通过提供文本提示，AI 代理能够在图像中识别并定位目标对象，无需传统的标注数据和模型训练过程。

二、API 调用方式

1. 认证方式

- API 使用 Bearer Token 进行认证：*Authorization: Bearer API_KEY*
- 可用的 API KEY: MGZ1bDNxMDZ6dHc2N2p5ZmR0NDBlOnFUSnFYQ2V6d3dsQUdiRFdWSjh4VzljWWpqcEFleFBC

2. HTTP 请求格式

- **方法**：POST
- **地址**：https://api.va.landing.ai/v1/tools/agentic-object-detection
- **请求体**：图像的 Base64 编码和文本提示
- **Cost:** $0.03 per image

3. 输入参数

- **image**：JPEG、PNG 图像（或 Base64 编码），建议分辨率小于等于 1920×1080，不低于 224x224 像素，以确保检测精度。
- **prompt**：描述目标对象的文本提示。一次只检测一种物体类型，使用单数词会更准确 。

4. 输出结果

返回 JSON 格式，包含检测到的对象列表，每个对象包括:

- **label**：标签名称。
- **score**：置信度评分，范围为 0 ~ 1。
- **bounding_box**：位置坐标，格式为 [x_min, y_min, x_max, y_max]

5. 速率限制

API 对请求频率有限制，默认限制每秒 10 次请求。超过限制时，API 将返回 429 Too Many Requests 错误。

三、API 调用示例代码

https://github.com/ygao2002/internship2025/blob/main/week_1/API_example.py

输入：

输出：

```
C:\Users\0916g\OneDrive\Desktop\siemens_intern\pycharm_traffic_detection\.venv\Scripts\python.exe C
{'data': [[{'label': 'a yellow car', 'score': 1.0, 'bounding_box': [216.0, 66.0, 387.0, 152.0]}]]}
```

数据清洗报告

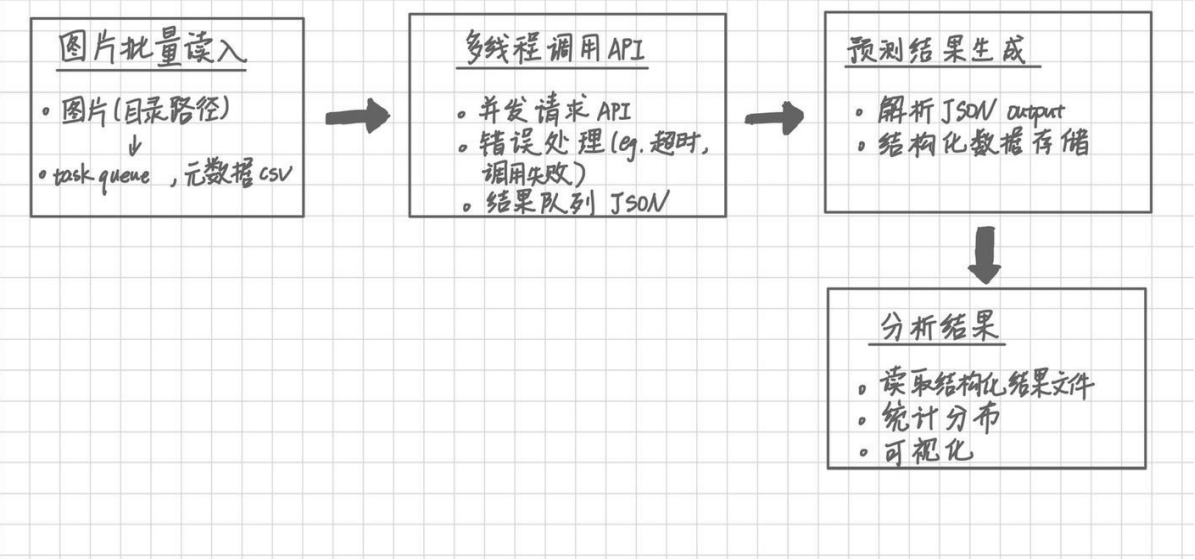
一、图像数据概况

- **Source:** COCO
- **总图像数量**：约 2500 张
- **主要文件格式**：JPEG（.jpg）
- **颜色模式**：RGB
- **分辨率范围**：多样，需统一处理

二、数据清洗步骤

1. **筛选交通相关图像**：
    - 使用图像分类模型筛选包含交通场景的图像，如道路、车辆、行人、交通标志等。（目前为手动筛选，后续可连接 AOD API）
2. **记录图像元数据**：
    - 提取并记录每张图像的文件路径、拍摄时间（本次收集的照片都无法提取此信息）、场景描述等信息，保存为 CSV 文件。
3. **分析图像数据格式**：
    - 使用 OpenCV 获取图像的分辨率、颜色模式和文件类型。

数据管道架构图和模块功能说明



图片批量读入模块

- 从指定目录批量读取图片文件。
- OpenCV 验证是否可读
- 元数据（路径、分辨率、格式）至 CSV 文件。
- 将图片数据（Base64 编码或文件路径）推送至 task queue，用于存储待处理的图像路径
- **输入**：图片路径
- **输出**：任务队列、元数据文件（.csv）

多线程 API 调用模块

- **输入**：任务队列
- 获取图片数据
- Threading 模块并发调用 Agentic Object Detection API。
- 处理 API 错误（如超时、速率限制）
- **输出：**将结果（JSON）推送至结果队列

预测结果生成模块

- **输入：**结果队列
- 解析 JSON 数据，提取关键字段（类别、置信度、边界框坐标）。
- **输出：**结构化数据存储至 CSV 文件或数据库（.csv）。

分析模块

- **输入：**结构化结果文件
- 统计目标类别分布、平均置信度。
- 生成图表（柱状图、热力图）。

Questions:

- 要用到的 python 版本
- 数据集用 coco 的 API 还是批量下载图片
- 后续是否需要统一图像尺寸或颜色模式以便批量处理