**数据集制作**

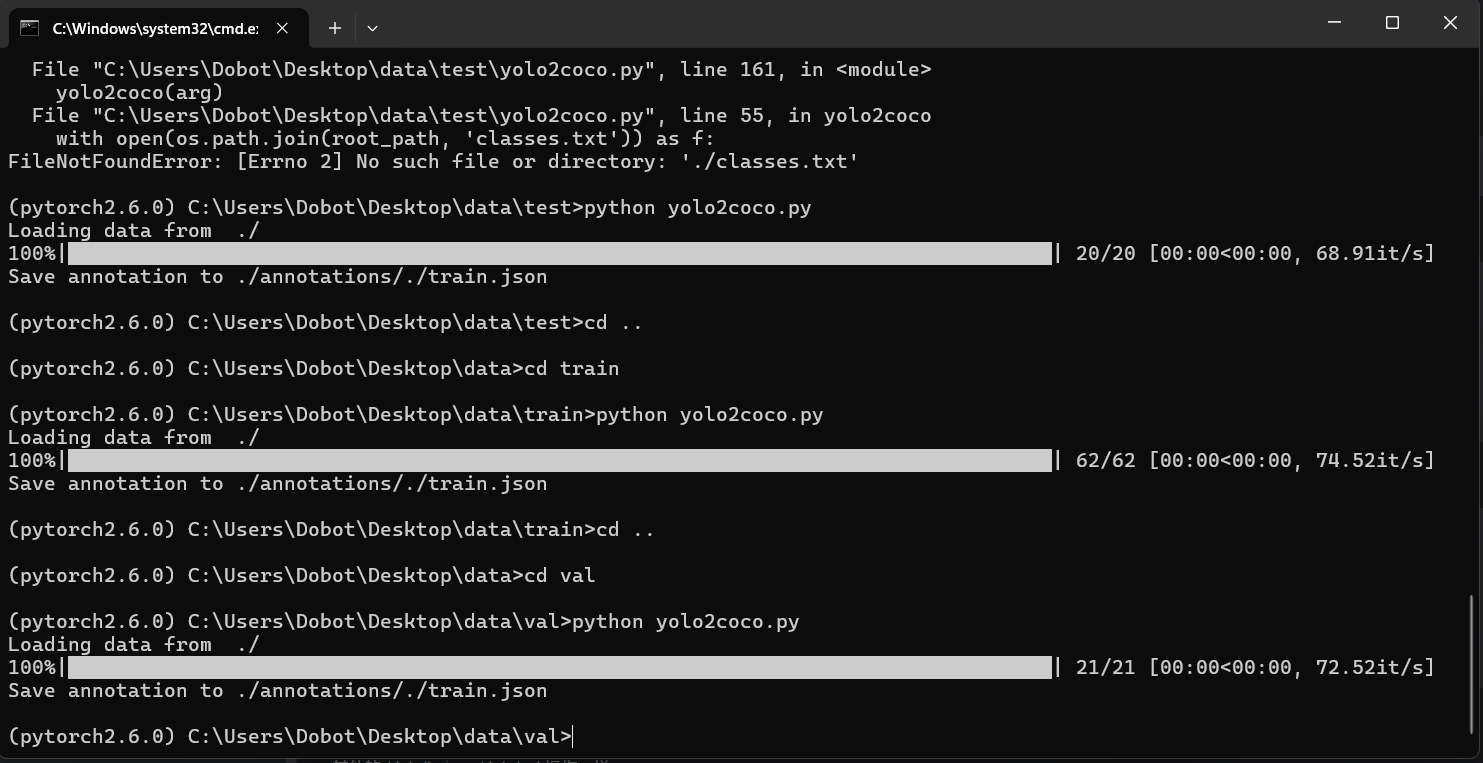
**拍摄含有三种类别物体的图片数据：apple、orange、metal rod,图片格式为png，图片尺寸均为1280\*720，共计103张图片，0-橙子，1-金属杆，2-苹果，坐标（xmin,ymin,xmax,ymax）**

**坐标的测算方法：使用labelImg软件进行手动标记，针对每一张图片的物体进行手动位置框取，选择YOLO或者PascalVOC格式导出**

**数据标记方法：使用labelImg软件进行手动标记**

**xml文件：按照顺序对应0、1、2编号的物体，标注其物体类型、物体编号、物体坐标**

**训练数据格式转换：YOLO-->COCO，使用代码：yolo2coco-master**



格式转换：YOLO-->COCO

**难点与解决方案**

最难的问题：自己去探寻如何将开源的模型变为己用，因为不知道别人是用什么数据进行训练的，别人只开源了模型

量化过程：将PC运行的模型运行到边缘端

解决方案：修改或删除一些参数，看原厂说明书，sdk（提供案例，会教如何做量化）

借鉴同行案例

**插曲**

PC端训练后，发现内存太小，需要转移到具有更大内存、更强显卡性能的Nvidia Agx Orin主机进行RT-DETRv2 Pytorch的训练与优化,但是其他同事的误操作导致该主机根文件系统损坏，需要进行Nvidia Agx Orin刷机操作，即系统重装。

在此期间，我学习并在短时间内实现了对Nvidia Agx Orin的刷机（系统重装）操作

**显存超载问题解决**

这个错误表明您的GPU显存不足（CUDA error: out of memory）。以下是完整的解决方案：

### ****1. 立即解决方案****

#### **(1) 减少批次大小（最有效）**

修改训练命令，降低batch\_size：

Cmd

python train.py --batch\_size 4 --device cuda:0 # 原可能是8或16

#### **(2) 启用梯度检查点（节省显存）**

在配置文件中添加：

Yaml

model: use\_checkpoint: True

#### **(3) 使用混合精度训练**

Cmd

python train.py --use\_amp True --device cuda:0

### ****2. 进阶优化方案****

#### **(1) 监控显存使用**

Cmd

nvidia-smi -l 1 *# 每秒刷新显存使用*

#### **(2) 优化数据加载**

Yaml

*# 修改config.yml*train\_dataloader: num\_workers: 4 *# 根据CPU核心数调整* pin\_memory: True *# 加速数据传输*

#### **(3) 模型裁剪**

Python

*# 在模型定义中减少层数*RTDETR:

decoder: num\_layers: 4 *# 原可能是6*

### ****3. 永久解决方案****

| **方法** | **操作** | **预期显存下降** |
| --- | --- | --- |
| 升级GPU | 更换显存≥12GB的显卡 | 100% |
| 使用云服务 | 租用A100/A10G实例 | 按需扩展 |
| 模型量化 | 添加torch.quantization | ~40% |
| 梯度累积 | --accumulate\_steps 2 | ~50% |

### ****4. 错误分析****

Python

RuntimeError: CUDA error: out of memory

* ****根本原因****：模型+数据所需显存 > GPU物理显存
* ****典型阈值****：
  + RTX 3060 (12GB)：最大支持batch\_size=8
  + RTX 3090 (24GB)：最大支持batch\_size=16

### ****5. 验证修复效果****

Cmd

*# 测试最小可行配置*python train.py \ --batch\_size 2 \ --image\_size 512 \ *# 降低输入分辨率* --use\_amp True \ --device cuda:0

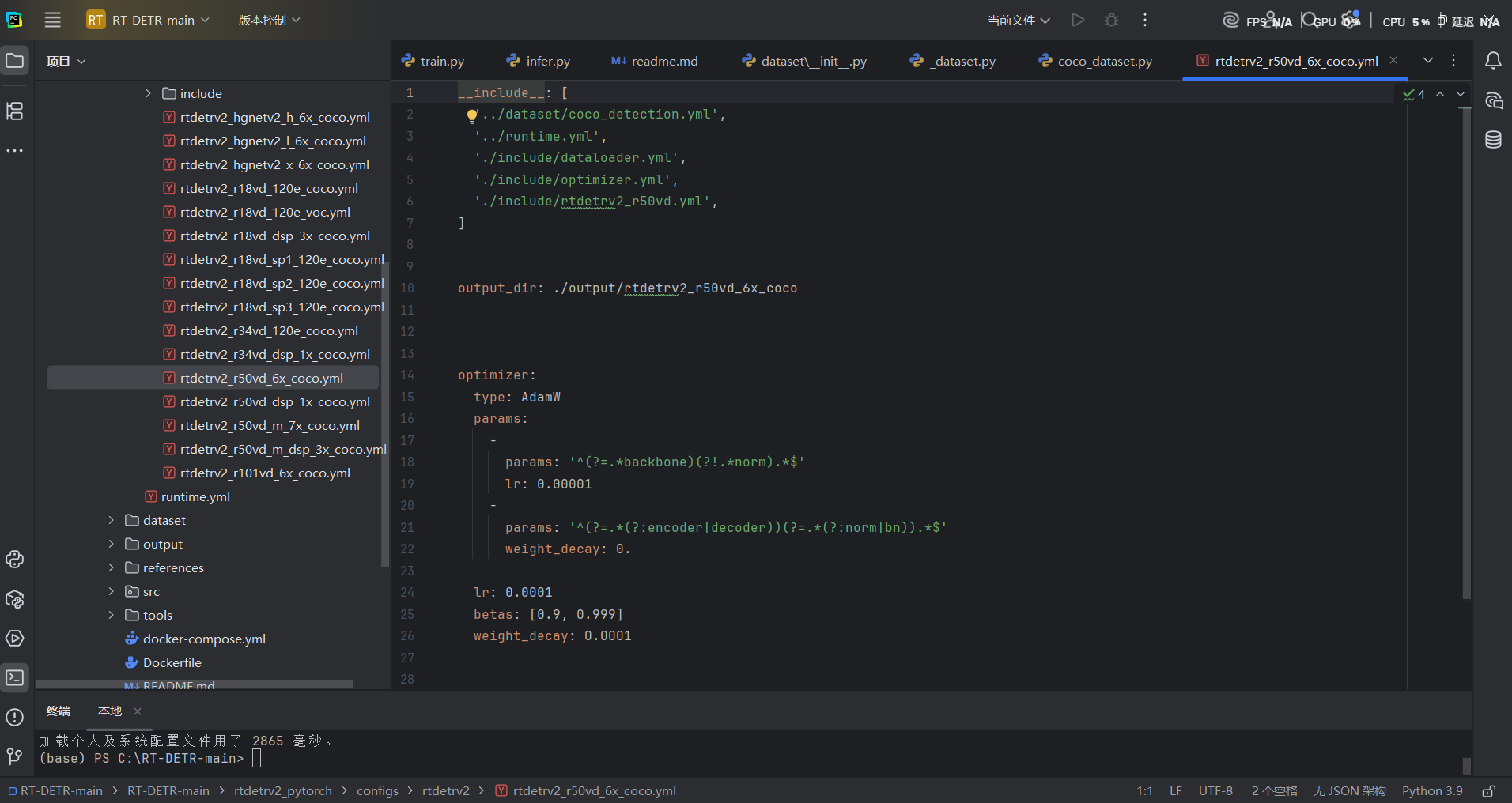
正常启动时应看到类似日志：

[Memory] Max allocated: 3.21/12.00 GB

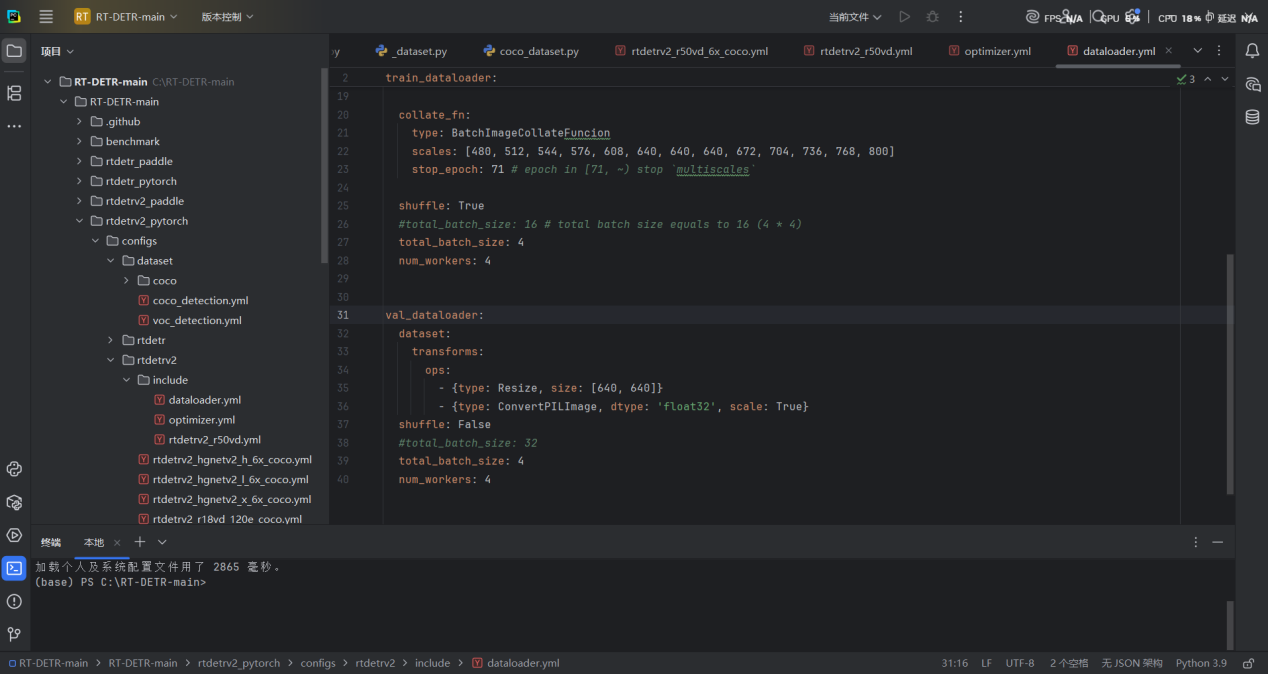
### ****关键参数对照表****

| **参数** | **安全值** | **风险值** |
| --- | --- | --- |
| batch\_size | 2-4 | ≥8 |
| image\_size | 512-640 | ≥800 |
| num\_layers | ≤4 | ≥6 |
| num\_workers | ≤4 | ≥8 |

建议从最小配置开始逐步调大参数，直到找到显存上限。如果仍有问题，可以尝试--device cpu先验证是否是代码逻辑问题。

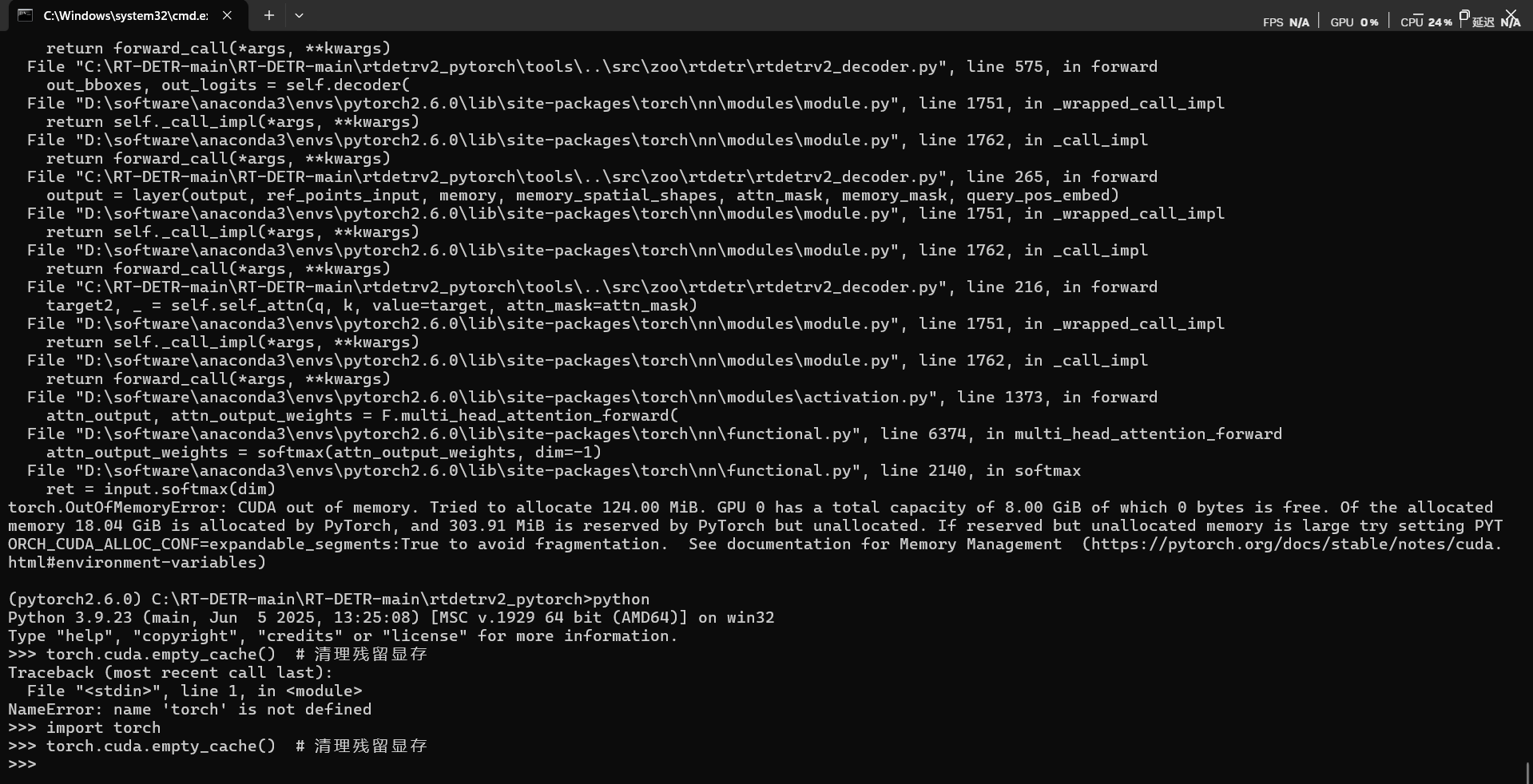


通过调整dataloader.yml实现对batch\_size的调整：



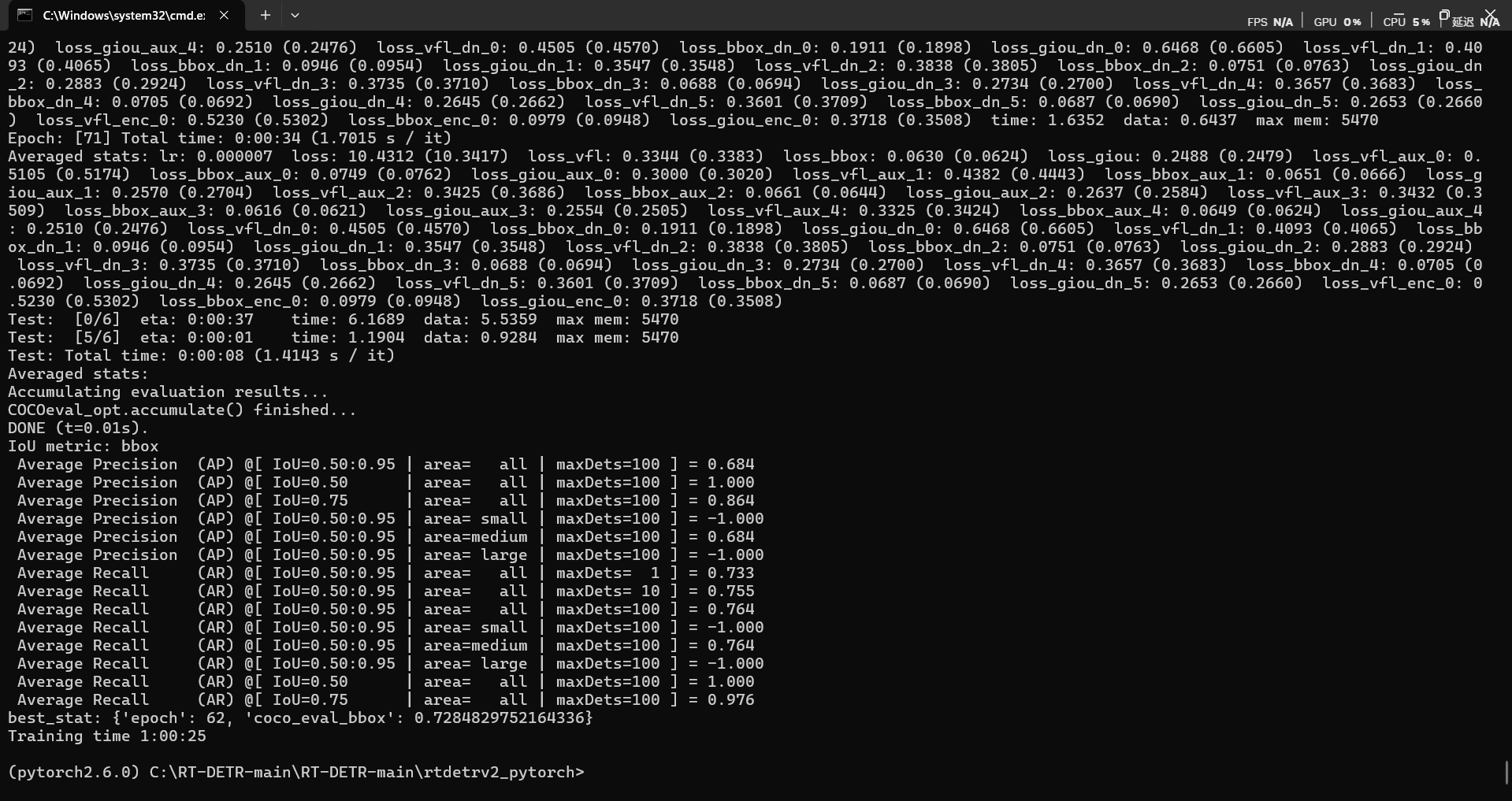
total\_batch\_size从16-->4,从32-->4

清理残留显存：



**RT-DETRv2模型训练**

RT-DETRv2 NVidia 4060 实验室数据集 epoch=71 训练结果：



**模型结构可视化分析**

将.pth或.onnx模型文件放入“netron.app”网站中，即可得到模型结构图，如上图所训练出的最优模型的结构图如下所示：

