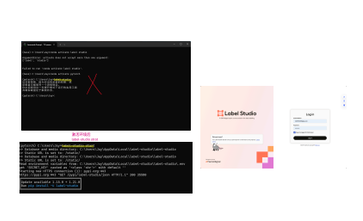
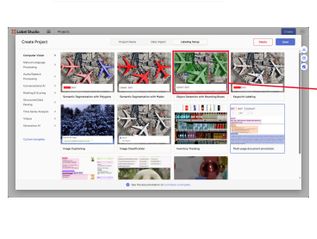
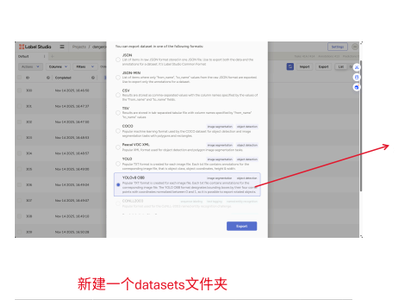
label studio

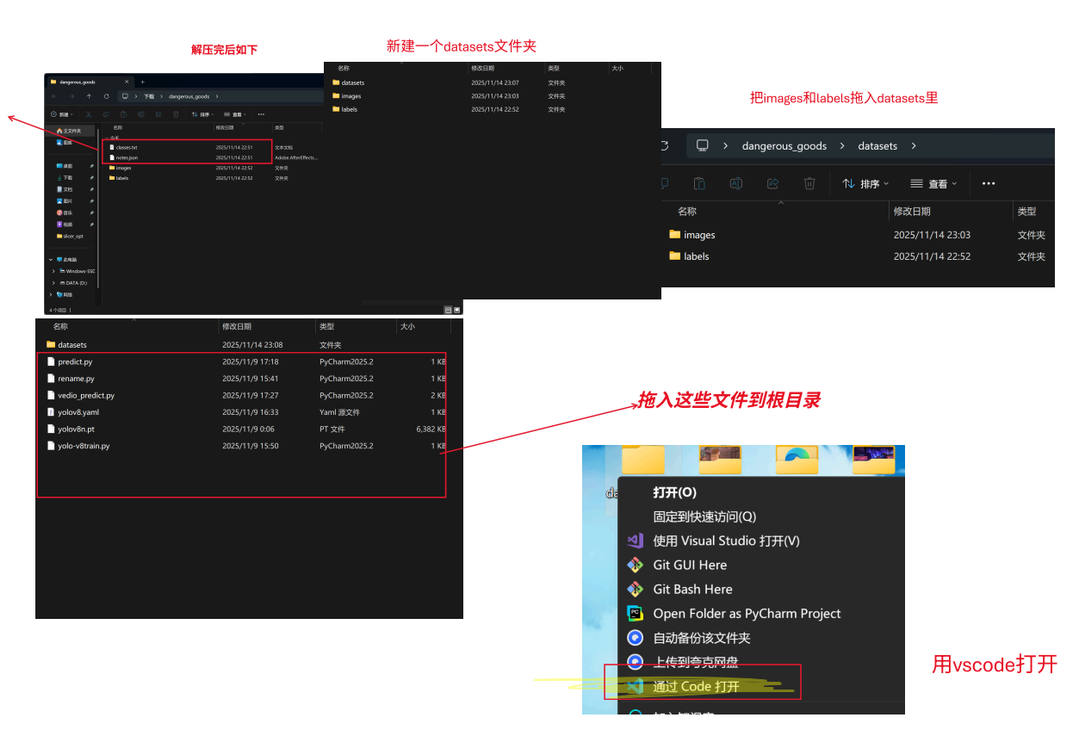




目标检测的话，记得选这个



全部标注完后 export导出yolov8格式



解压完后如下

新建一个datasets文件夹

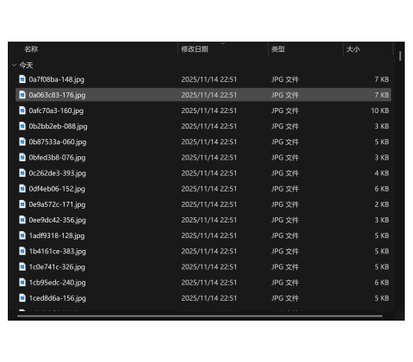
把images和labels拖入datasets里

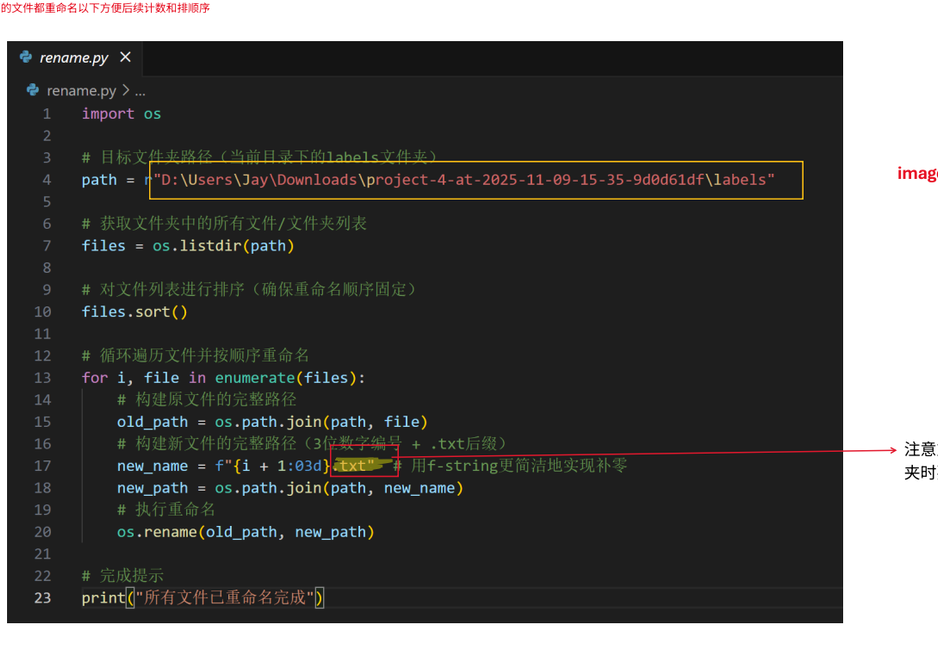
前两个可删除

拖入这些文件到根目录

用vscode打开

把images和labels里面的文件都重命名以下方便后续计数和排顺序

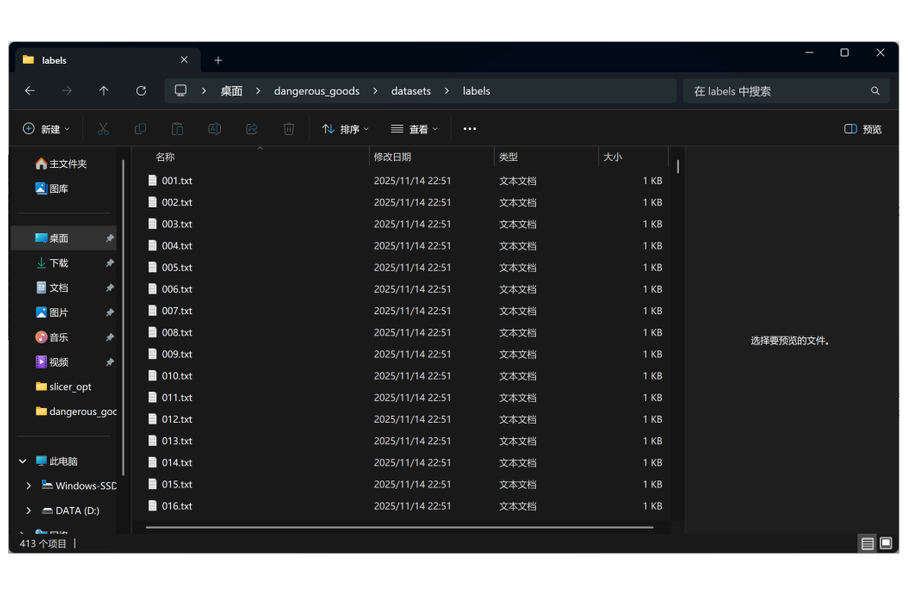




```python
import os

# 目标文件夹路径（当前目录下的labels文件夹）
path = r"D:\Users\Jay\Downloads\project-4-at-2025-11-09-15-35-9d0d61df\labels"

# 获取文件夹中的所有文件/文件夹列表
files = os.listdir(path)

# 对文件列表进行排序（确保重命名顺序固定）
files.sort()

# 循环遍历文件并按顺序重命名
for i, file in enumerate(files):
    # 构建原文件的完整路径
    old_path = os.path.join(path, file)
    # 构建新文件的完整路径（3位数字编号 + .txt后缀）
    new_name = f"{i + 1:03d}.txt"  # 用f-string更简洁地实现补零
    new_path = os.path.join(path, new_name)
    # 执行重命名
    os.rename(old_path, new_path)

# 完成提示
print("所有文件已重命名完成")
```

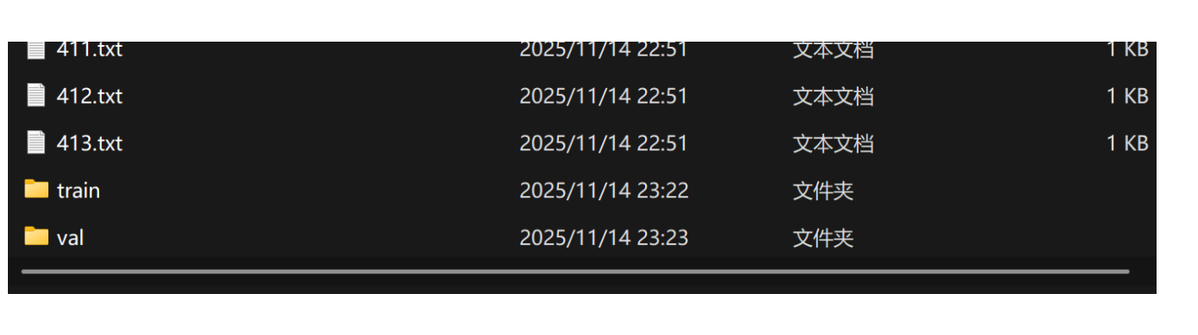
images和labels的文件夹地址

注意重命名images文件夹时要改为jpg或者png



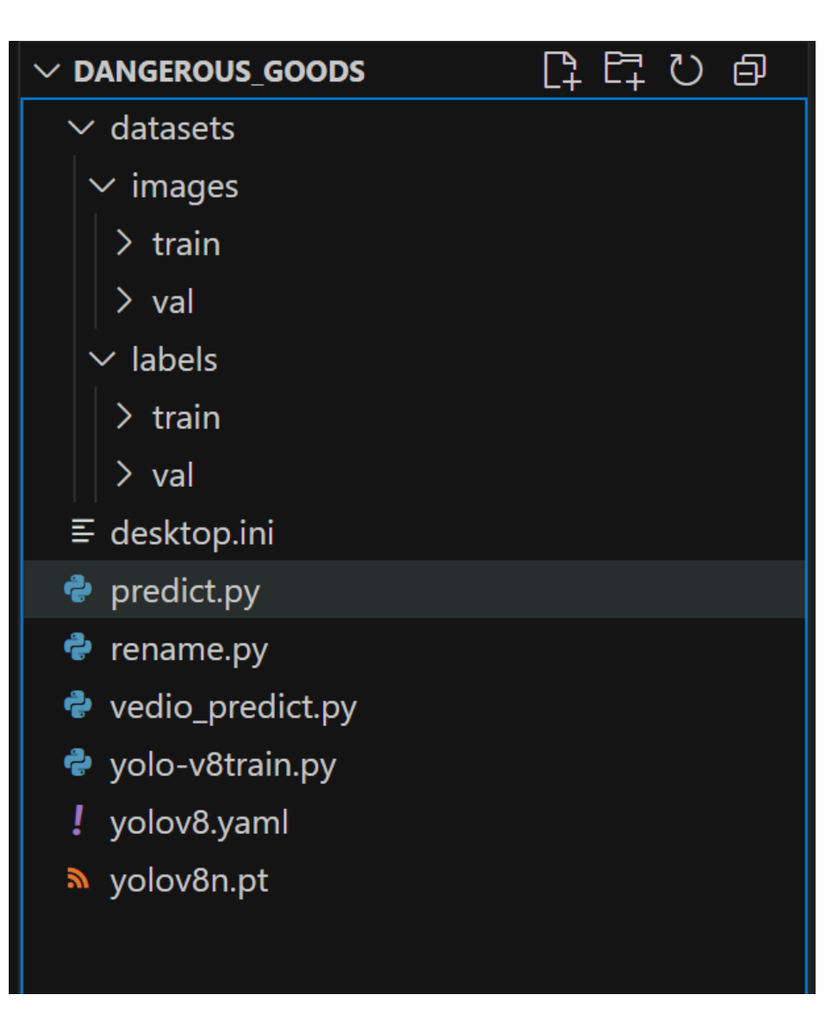
```
001.txt   2025/11/14 22:51   文本文档   1 KB
002.txt   2025/11/14 22:51   文本文档   1 KB
003.txt   2025/11/14 22:51   文本文档   1 KB
004.txt   2025/11/14 22:51   文本文档   1 KB
005.txt   2025/11/14 22:51   文本文档   1 KB
006.txt   2025/11/14 22:51   文本文档   1 KB
007.txt   2025/11/14 22:51   文本文档   1 KB
008.txt   2025/11/14 22:51   文本文档   1 KB
009.txt   2025/11/14 22:51   文本文档   1 KB
010.txt   2025/11/14 22:51   文本文档   1 KB
011.txt   2025/11/14 22:51   文本文档   1 KB
012.txt   2025/11/14 22:51   文本文档   1 KB
013.txt   2025/11/14 22:51   文本文档   1 KB
```

images和labels文件夹里都新建两个 文件夹



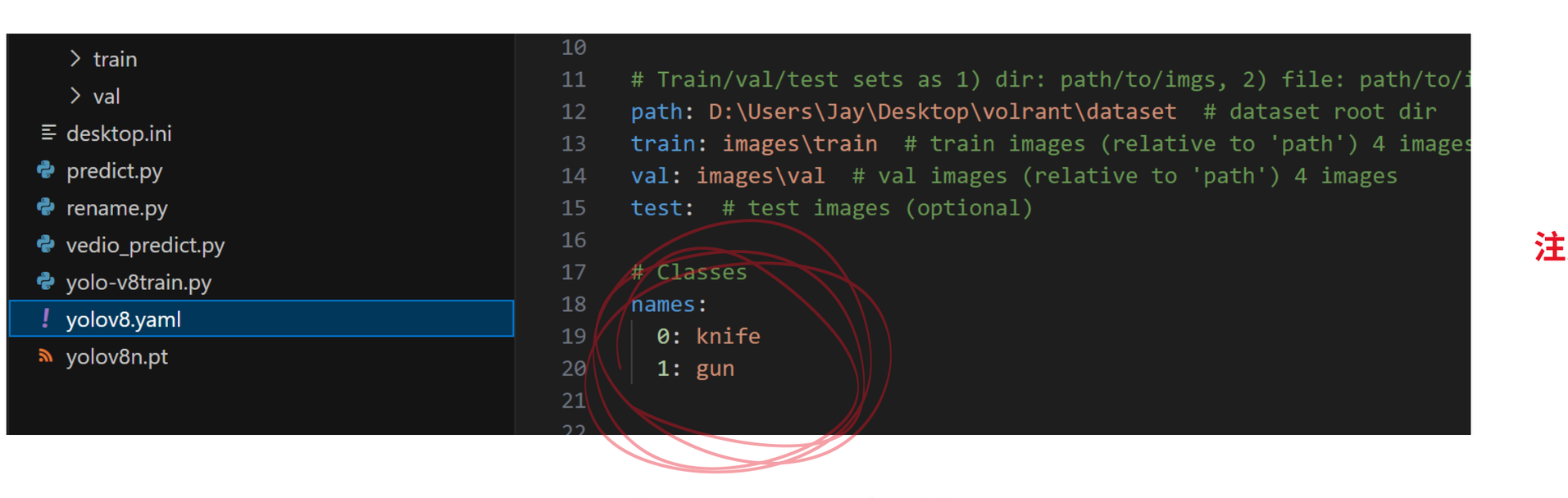
```
411.txt   2025/11/14 22:51   文本文档   1 KB
412.txt   2025/11/14 22:51   文本文档   1 KB
413.txt   2025/11/14 22:51   文本文档   1 KB
train     2025/11/14 23:22   文件夹
val       2025/11/14 23:23   文件夹
```

再把images和labels里的文件按照一定的比例分配到train和val文件夹里



```
DANGEROUS_GOODS
  datasets
    images
      train
      val
    labels
      train
      val
  desktop.ini
  predict.py
  rename.py
  vedio_predict.py
  yolo-v8train.py
  yolov8.yaml
  yolov8n.pt
```

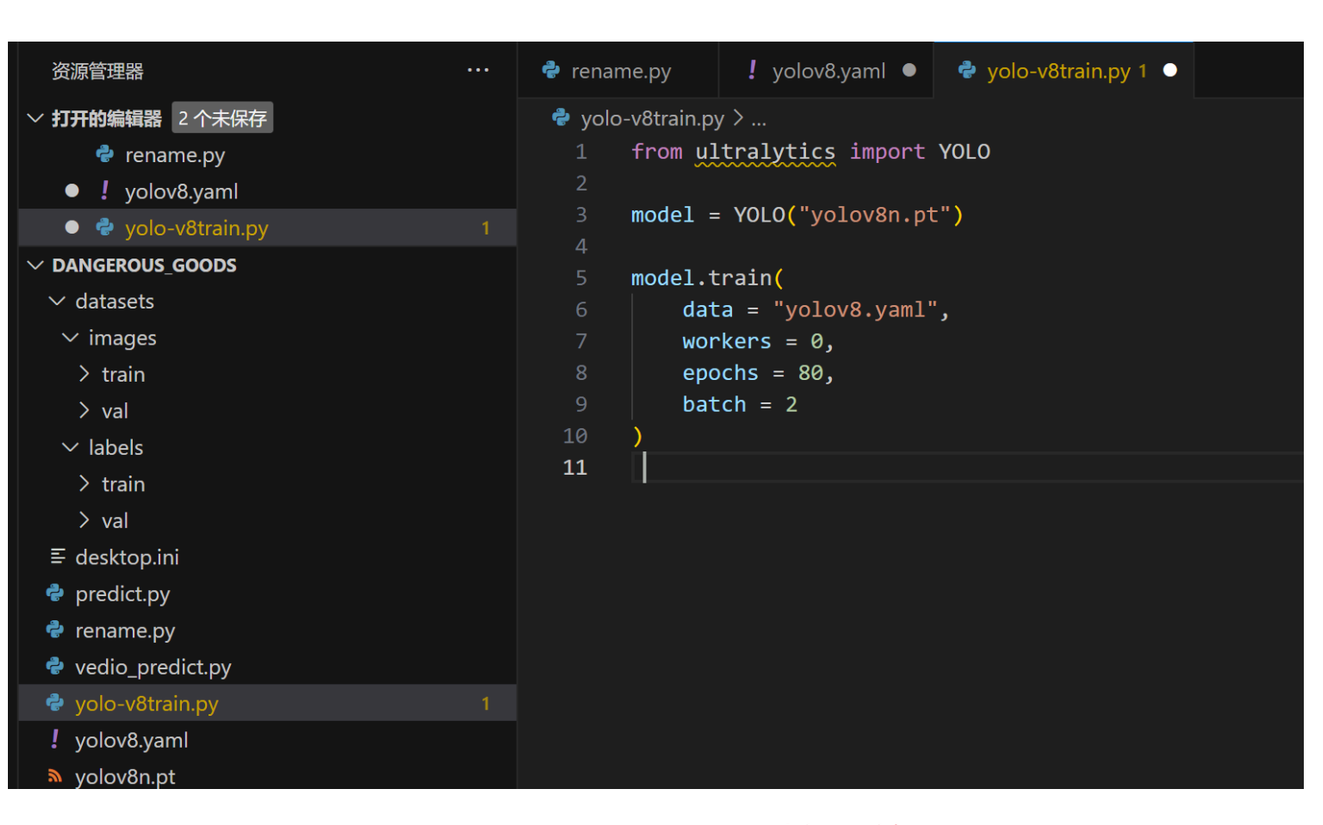
改写yaml文件



```yaml
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/i
path: D:\Users\Jay\Desktop\volrant\dataset  # dataset root dir
train: images\train  # train images (relative to 'path') 4 images
val: images\val  # val images (relative to 'path') 4 images
test:  # test images (optional)

# Classes
names:
  0: knife
  1: gun
```

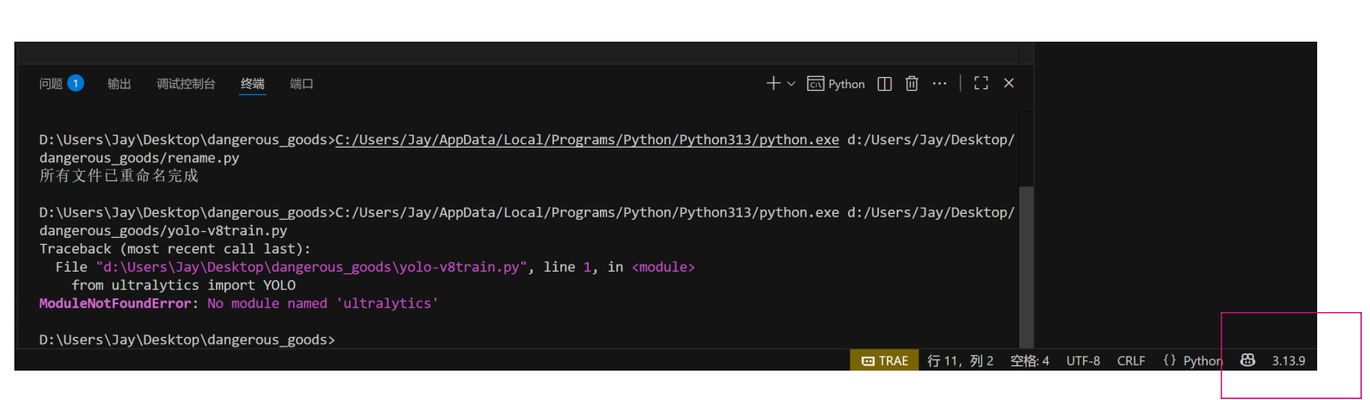
注意这里别标反了

改写train文件



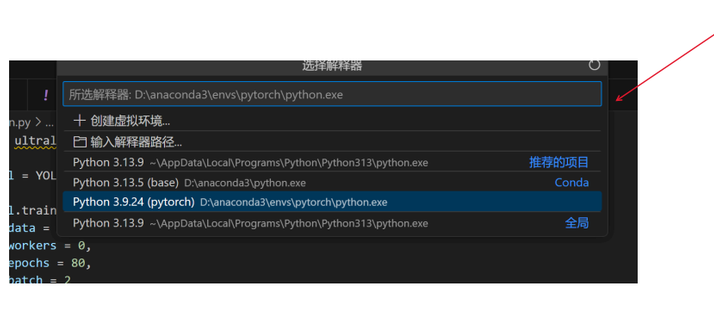
```python
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
    data = "yolov8.yaml",
    workers = 0,
    epochs = 80,
    batch = 2
)
```

开始训练



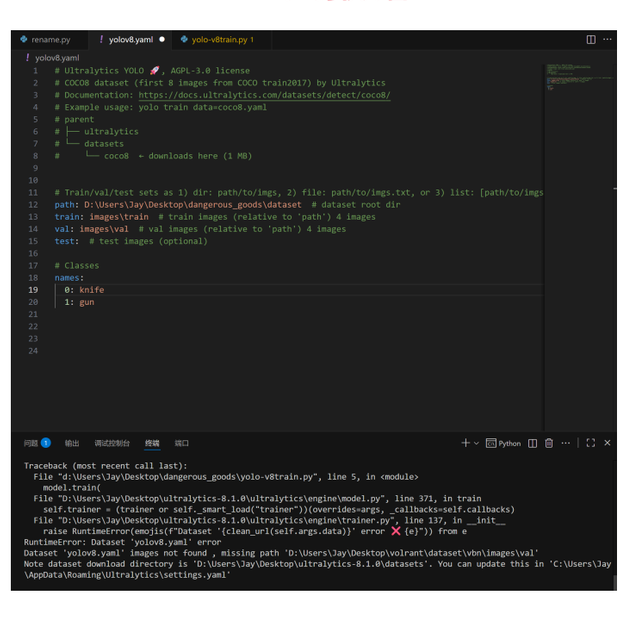
```
D:\Users\Jay\Desktop\dangerous_goods>C:/Users/Jay/AppData/Local/Programs/Python/Python313/python.exe d:/Users/Jay/Desktop/
dangerous_goods/rename.py
所有文件已重命名完成

D:\Users\Jay\Desktop\dangerous_goods>C:/Users/Jay/AppData/Local/Programs/Python/Python313/python.exe d:/Users/Jay/Desktop/
dangerous_goods/yolo-v8train.py
Traceback (most recent call last):
  File "d:\Users\Jay\Desktop\dangerous_goods\yolo-v8train.py", line 1, in <module>
    from ultralytics import YOLO
ModuleNotFoundError: No module named 'ultralytics'

D:\Users\Jay\Desktop\dangerous_goods>
```

这里报错是因为没有在pytorch环境里



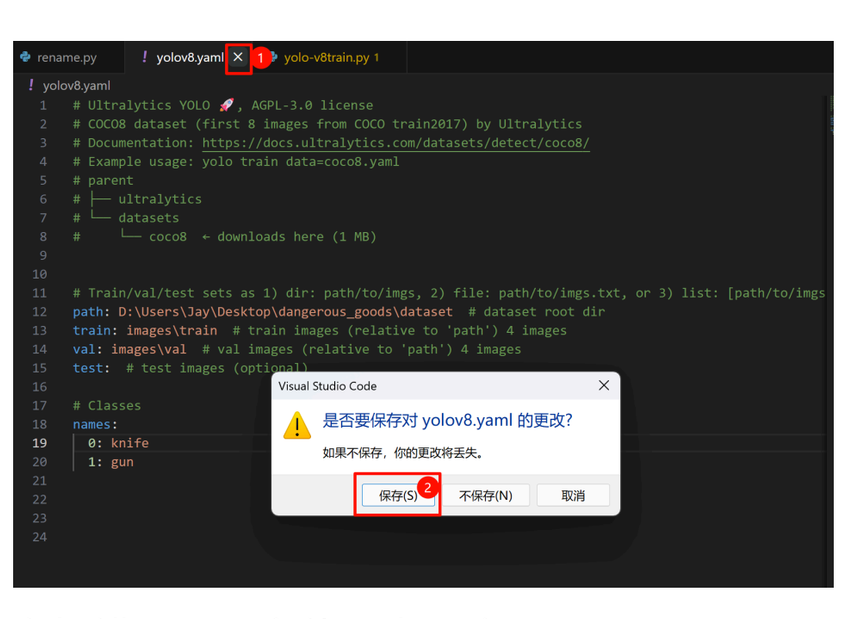
选择解释器
所选解释器: D:\anaconda3\envs\pytorch\python.exe
+ 创建虚拟环境...
输入解释器路径...
Python 3.13.9 ~\AppData\Local\Programs\Python\Python313\python.exe 推荐的项目
Python 3.13.5 (base) D:\anaconda3\python.exe Conda
Python 3.9.24 (pytorch) D:\anaconda3\envs\pytorch\python.exe
Python 3.13.9 ~\AppData\Local\Programs\Python\Python313\python.exe 全局

又报错



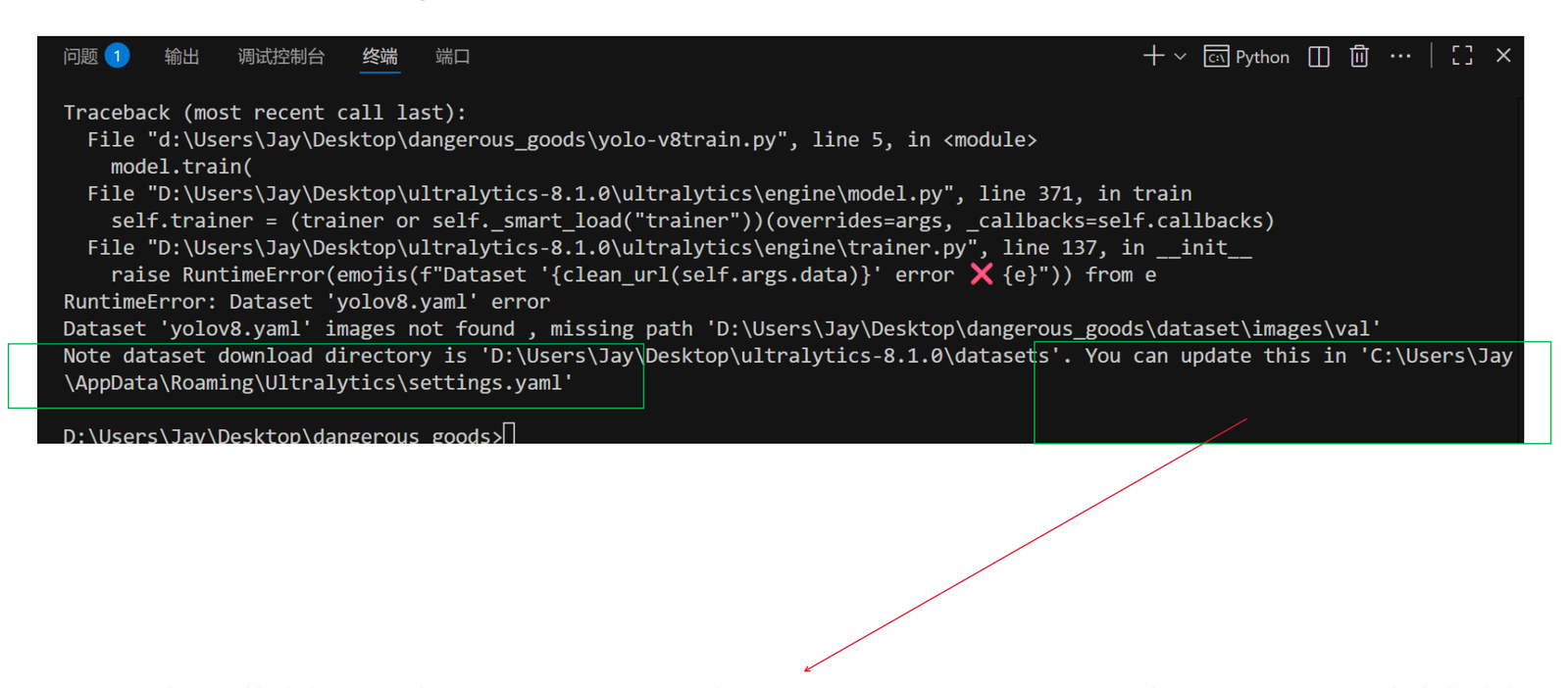
```
Traceback (most recent call last):
  File "d:\Users\Jay\Desktop\dangerous_goods\yolo-v8train.py", line 5, in <module>
    model.train(
  File "D:\Users\Jay\Desktop\ultralytics-8.1.0\ultralytics\engine\model.py", line 371, in train
    self.trainer = (trainer or self._smart_load("trainer"))(overrides=args, _callbacks=self.callbacks)
  File "D:\Users\Jay\Desktop\ultralytics-8.1.0\ultralytics\engine\trainer.py", line 137, in __init__
    raise RuntimeError(emojis(f"Dataset '{clean_url(self.args.data)}' error ❌ {e}")) from e
RuntimeError: Dataset 'yolov8.yaml' error
```

这里是因为改过yaml文件过后没有再保存一下，导致找不到文件



```yaml
# Ultralytics YOLO 🚀, AGPL-3.0 license
# COCO8 dataset (first 8 images from COCO train2017) by Ultralytics
# Documentation: https://docs.ultralytics.com/datasets/detect/coco8/
# Example usage: yolo train data=coco8.yaml
# parent
# ├── ultralytics
# └── datasets
#     └── coco8  ← downloads here (1 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs
path: D:\Users\Jay\Desktop\dangerous_goods\dataset  # dataset root dir
train: images\train  # train images (relative to 'path') 4 images
val: images\val  # val images (relative to 'path') 4 images
test:  # test images (optional)

# Classes
names:
  0: knife
  1: gun
```

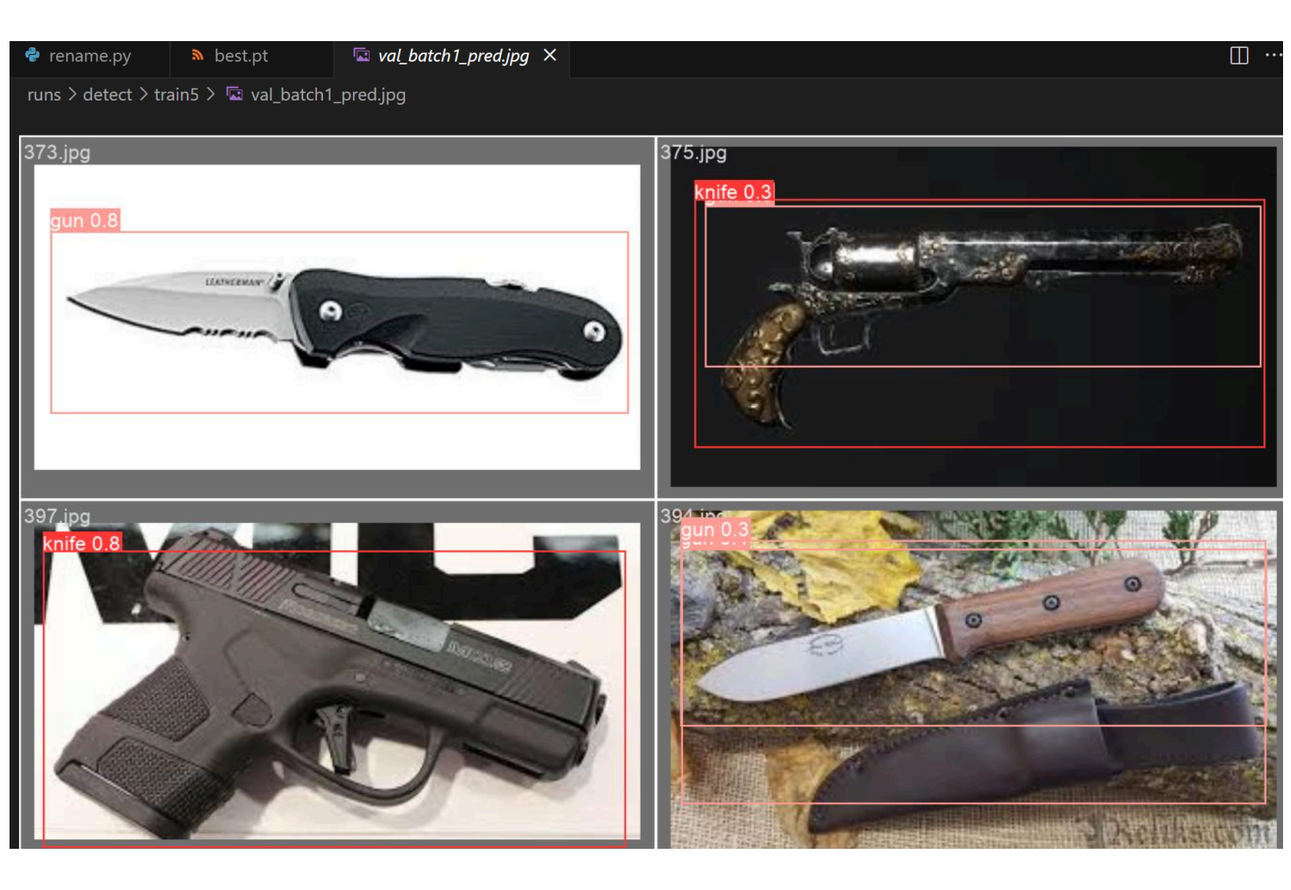
又一次报错，yaml文件里少了个s

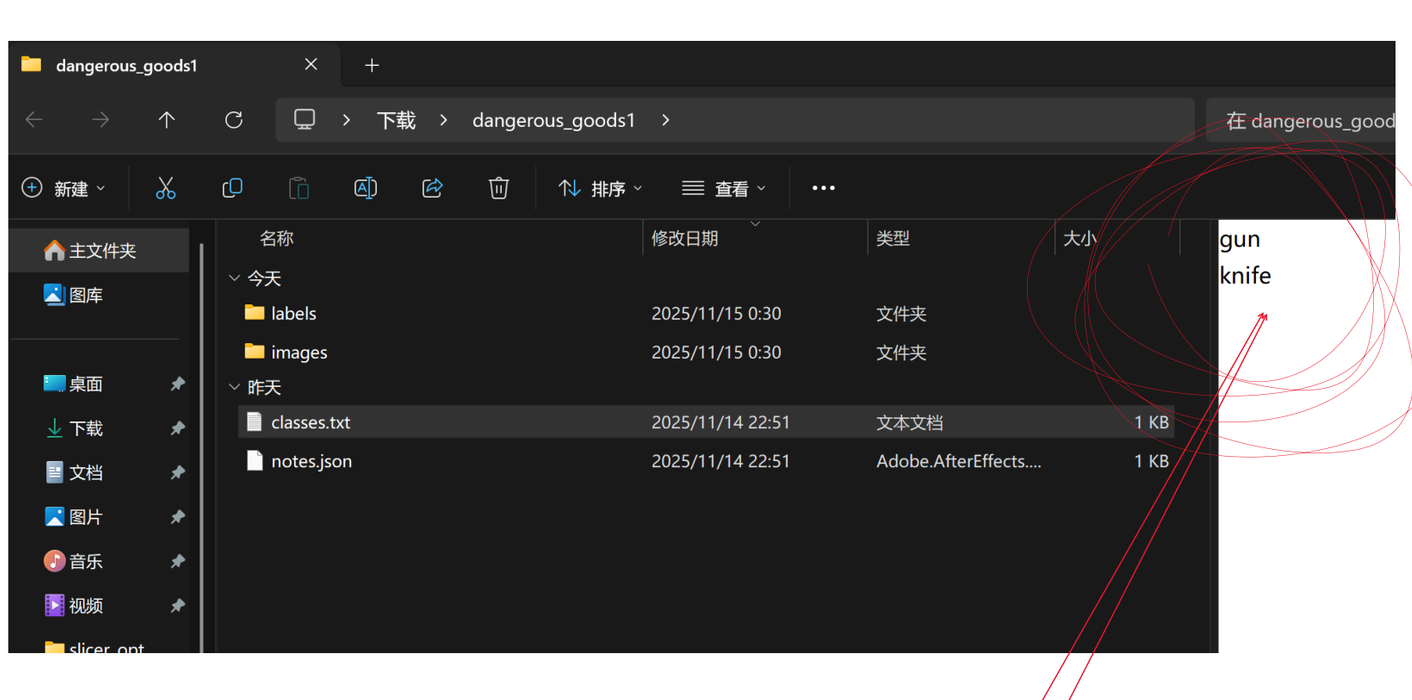


```
Traceback (most recent call last):
  File "d:\Users\Jay\Desktop\dangerous_goods\yolo-v8train.py", line 5, in <module>
    model.train(
  File "D:\Users\Jay\Desktop\ultralytics-8.1.0\ultralytics\engine\model.py", line 371, in train
    self.trainer = (trainer or self._smart_load("trainer"))(overrides=args, _callbacks=self.callbacks)
  File "D:\Users\Jay\Desktop\ultralytics-8.1.0\ultralytics\engine\trainer.py", line 137, in __init__
    raise RuntimeError(emojis(f"Dataset '{clean_url(self.args.data)}' error ❌ {e}")) from e
RuntimeError: Dataset 'yolov8.yaml' error
Dataset 'yolov8.yaml' images not found , missing path 'D:\Users\Jay\Desktop\dangerous_goods\dataset\images\val'
Note dataset download directory is 'D:\Users\Jay\Desktop\ultralytics-8.1.0\datasets'. You can update this in 'C:\Users\Jay
\AppData\Roaming\Ultralytics\settings.yaml'

D:\Users\Jay\Desktop\dangerous_goods>
```

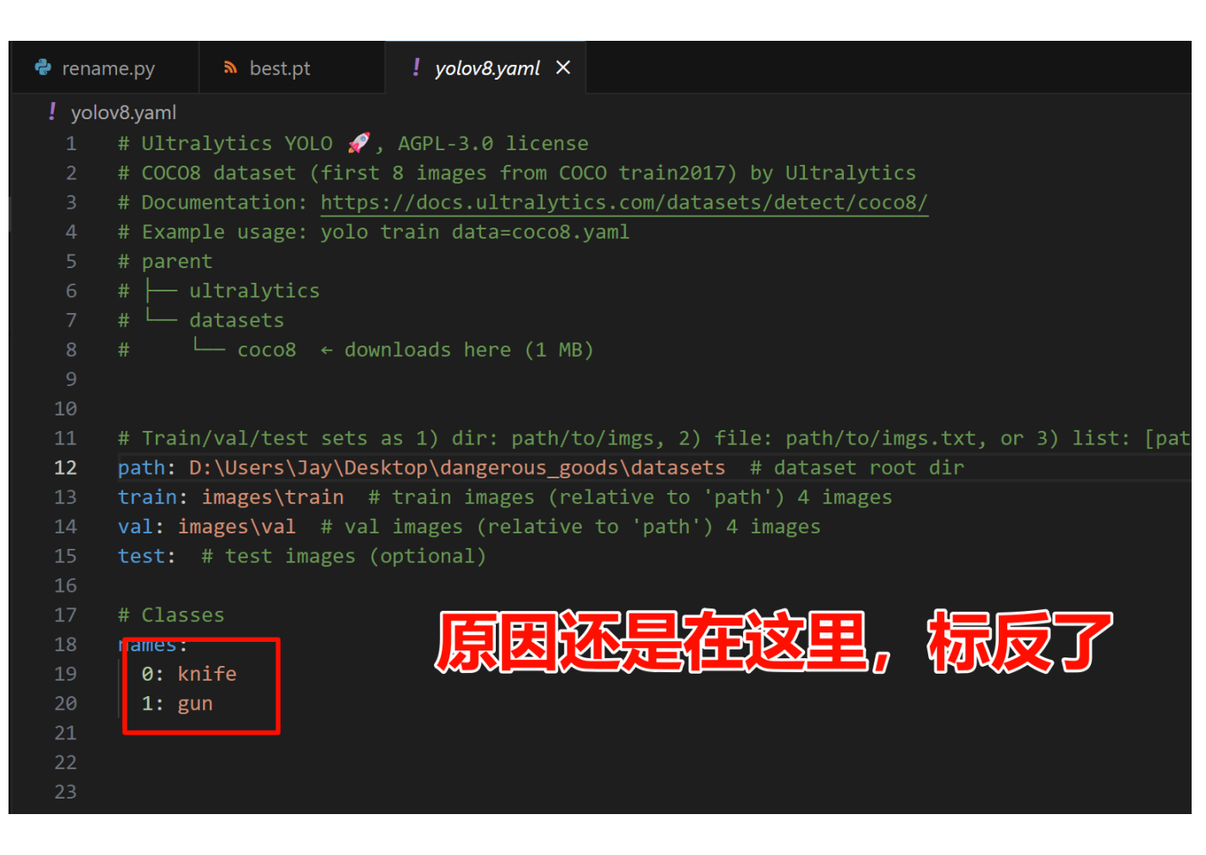
如果还是报错就到这个n路径下把settings.yaml五文件给删除掉就行了

又出现了这滑稽的一幕， 他又给我标反了





```
下载 > dangerous_goods1
今天
labels        2025/11/15 0:30    文件夹
images        2025/11/15 0:30    文件夹
昨天
classes.txt   2025/11/14 22:51   文本文档                 1 KB
notes.json    2025/11/14 22:51   Adobe.AfterEffects...   1 KB
```

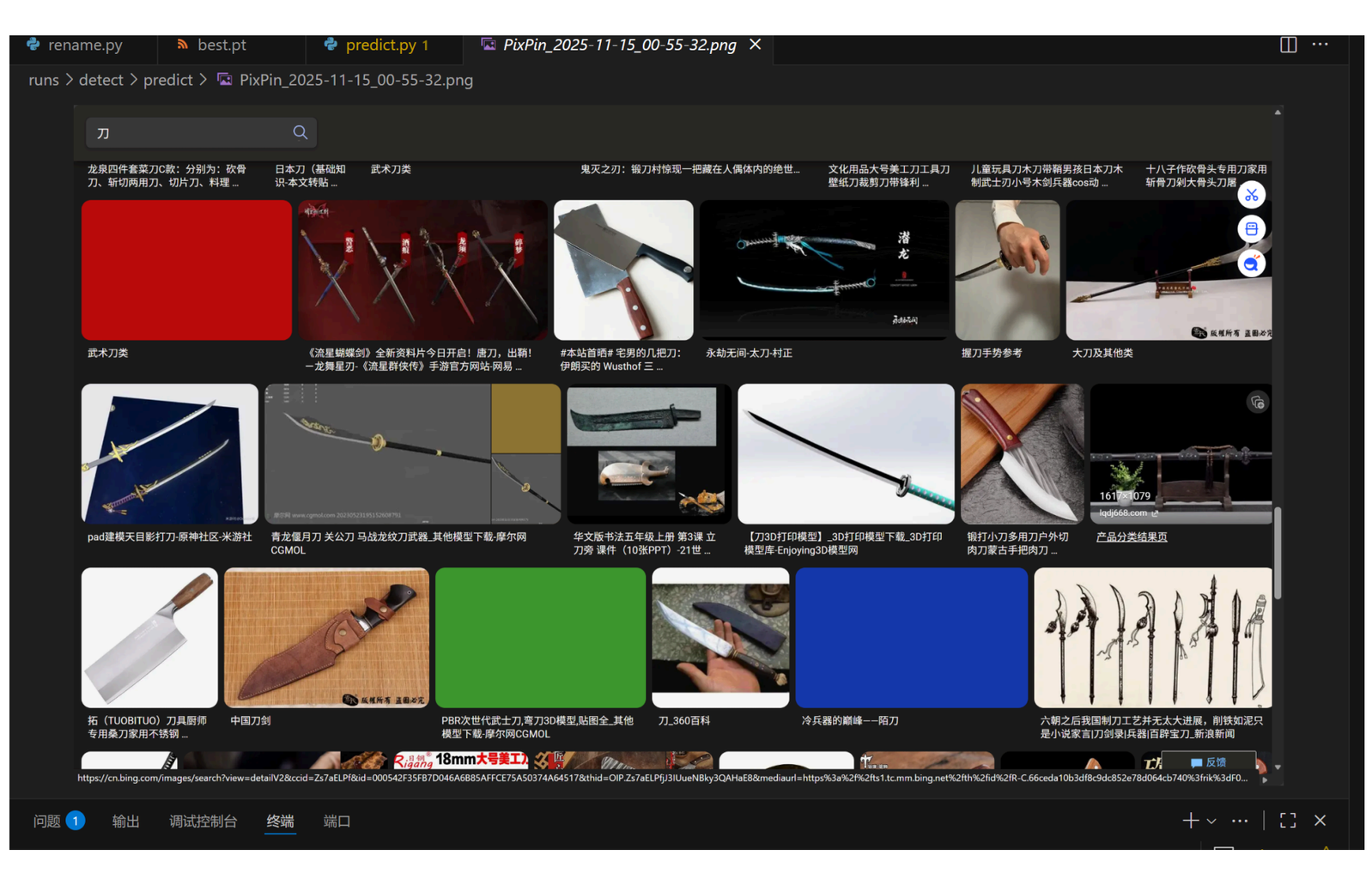


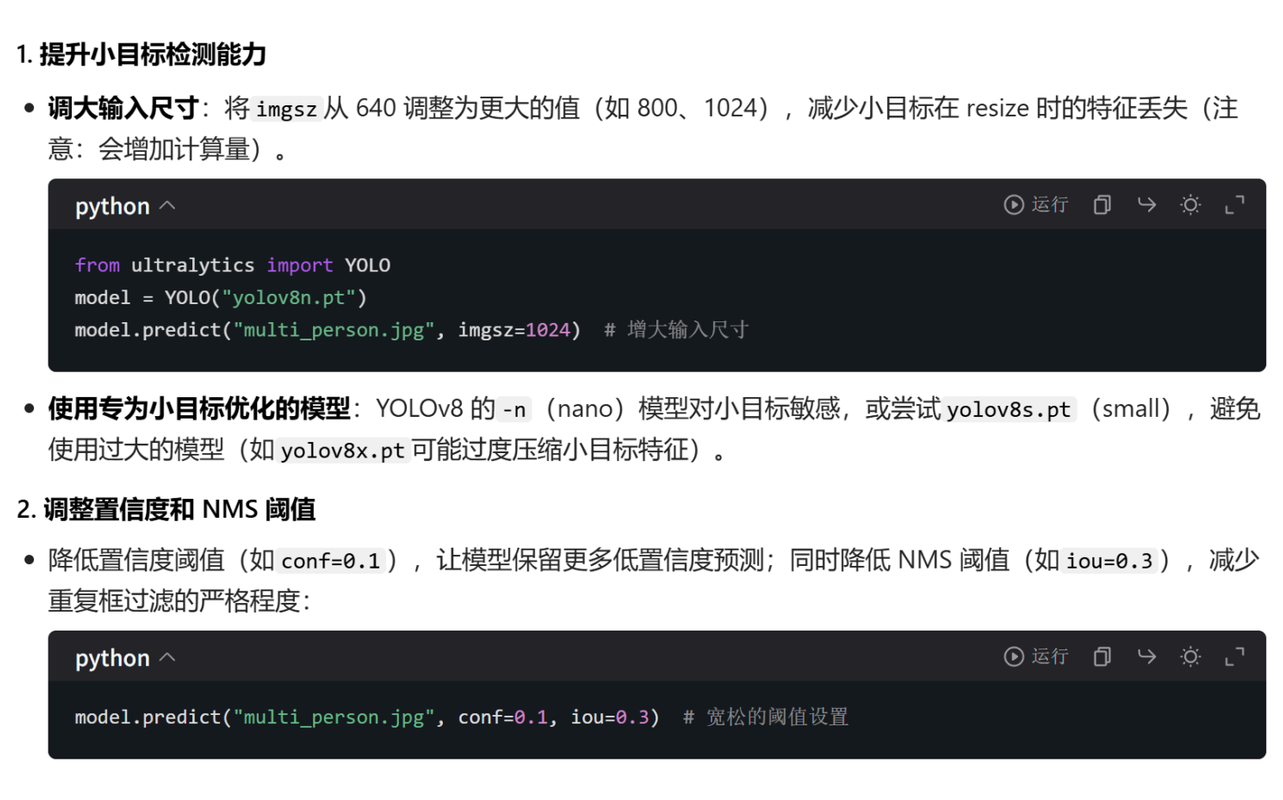
```yaml
# Ultralytics YOLO 🚀, AGPL-3.0 license
# COCO8 dataset (first 8 images from COCO train2017) by Ultralytics
# Documentation: https://docs.ultralytics.com/datasets/detect/coco8/
# Example usage: yolo train data=coco8.yaml
# parent
# ├── ultralytics
# └── datasets
#     └── coco8  ← downloads here (1 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [pat
path: D:\Users\Jay\Desktop\dangerous_goods\datasets  # dataset root dir
train: images\train  # train images (relative to 'path') 4 images
val: images\val  # val images (relative to 'path') 4 images
test:  # test images (optional)

# Classes
names:
  0: knife
  1: gun
```

原因还是在这里，标反了

标注数据的时候，
knife是第一个，
gun是第二个，
最终导出来的是这样的

传入一张大图片，啥也检测不到





1. 提升小目标检测能力

- **调大输入尺寸**：将 `imgsz` 从 640 调整为更大的值（如 800、1024），减少小目标在 resize 时的特征丢失（注意：会增加计算量）。

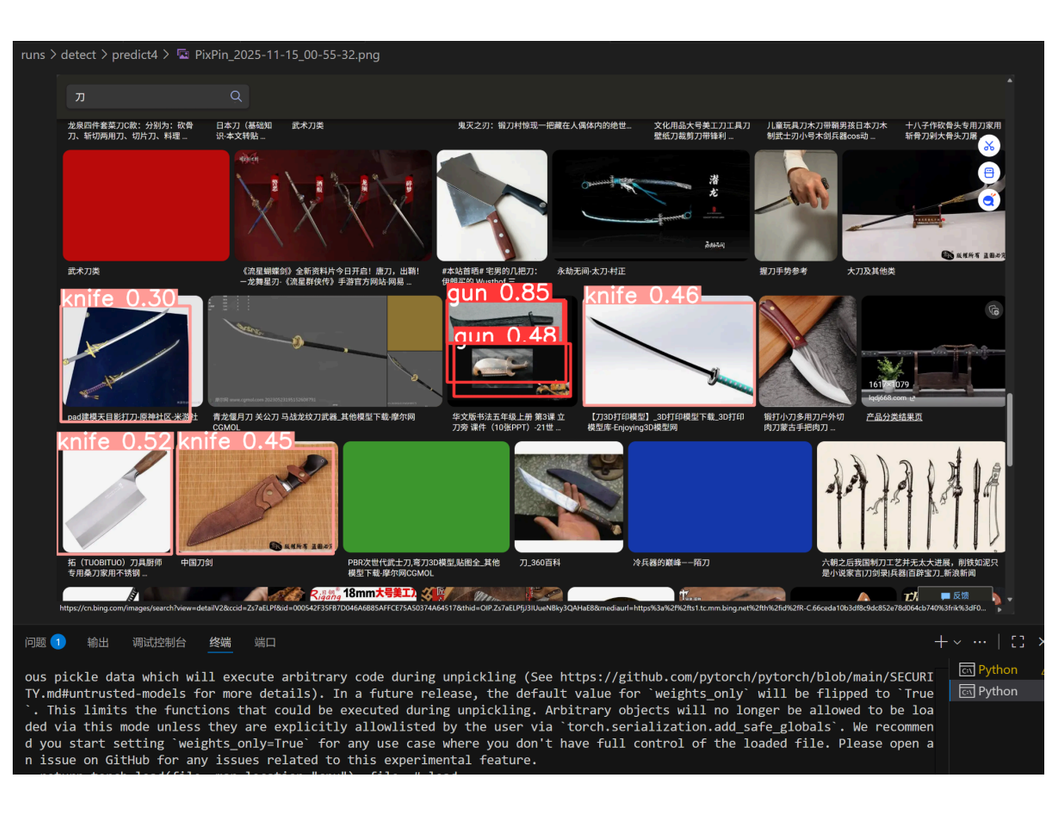
```python
from ultralytics import YOLO
model = YOLO("yolov8n.pt")
model.predict("multi_person.jpg", imgsz=1024)  # 增大输入尺寸
```

- **使用专为小目标优化的模型**：YOLOv8 的 `-n`（nano）模型对小目标敏感，或尝试 `yolov8s.pt`（small），避免使用过大的模型（如 `yolov8x.pt` 可能过度压缩小目标特征）。

2. 调整置信度和 NMS 阈值

- 降低置信度阈值（如 `conf=0.1`），让模型保留更多低置信度预测；同时降低 NMS 阈值（如 `iou=0.3`），减少重复框过滤的严格程度：

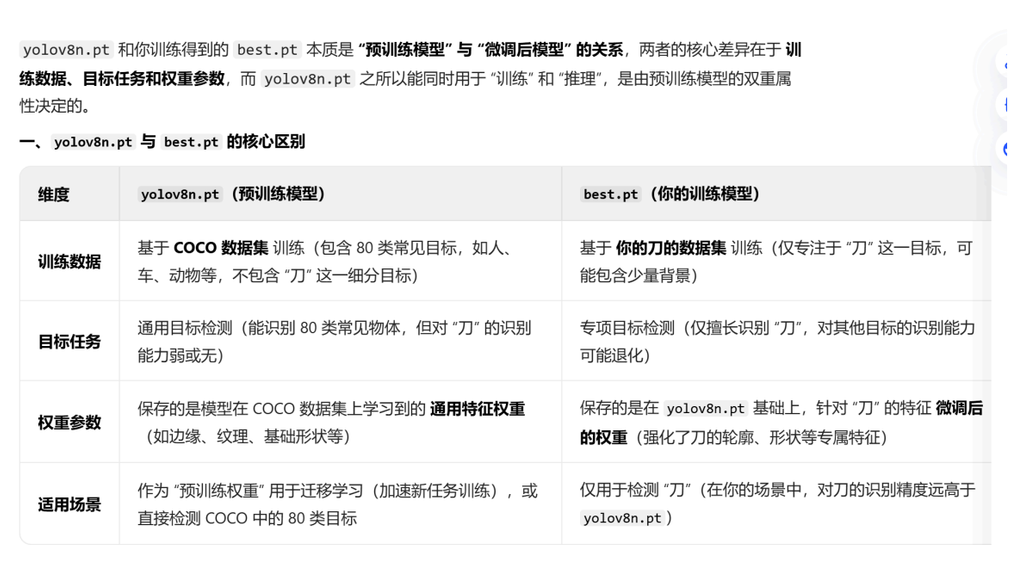
```python
model.predict("multi_person.jpg", conf=0.1, iou=0.3)  # 宽松的阈值设置
```



```
ous pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURI
TY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True
`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loa
ded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommen
d you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open a
n issue on GitHub for any issues related to this experimental feature.
```

解决

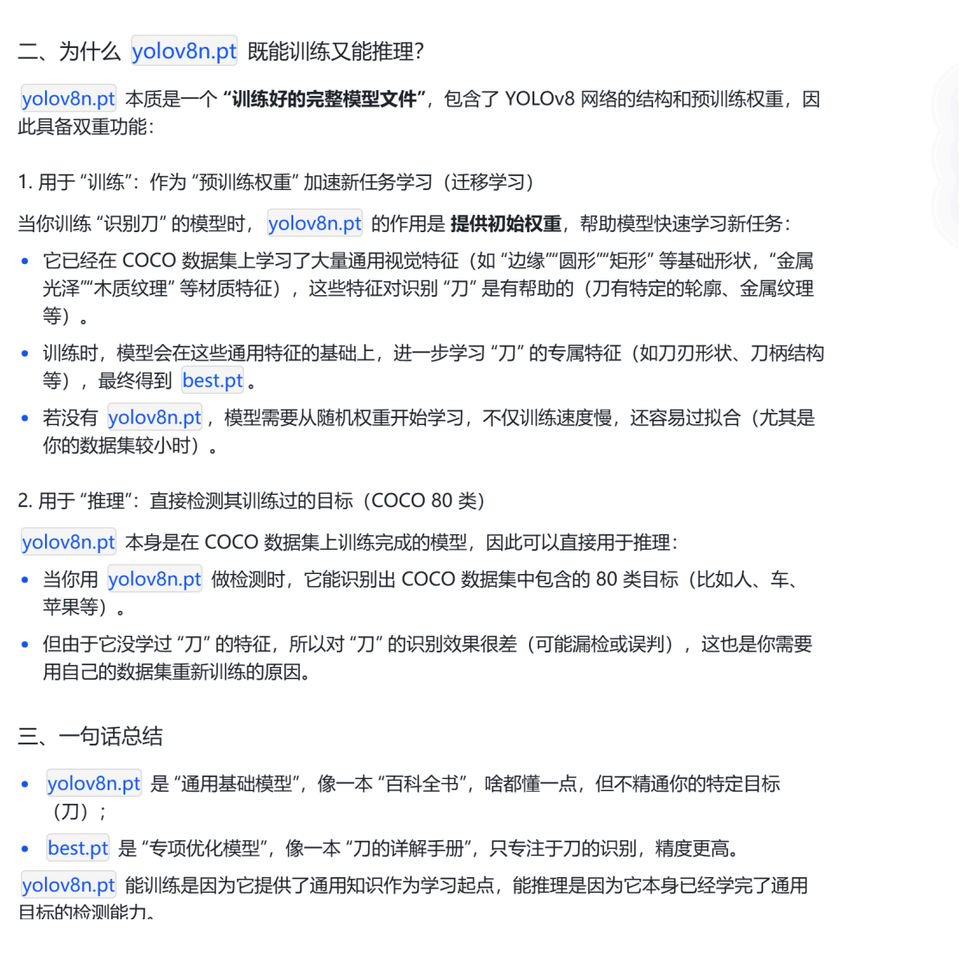
一些疑问



`yolov8n.pt` 和你训练得到的 `best.pt` 本质是 **"预训练模型" 与 "微调后模型" 的关系**，两者的核心差异在于 **训练数据、目标任务和权重参数**，而 `yolov8n.pt` 之所以能同时用于 "训练" 和 "推理"，是由预训练模型的双重属性决定的。

一、`yolov8n.pt` 与 `best.pt` 的核心区别

维度	`yolov8n.pt`（预训练模型）	`best.pt`（你的训练模型）
训练数据	基于 **COCO 数据集** 训练（包含 80 类常见目标，如人、车、动物等，不包含 "刀" 这一细分目标）	基于 **你的刀的数据集** 训练（仅专注于 "刀" 这一目标，可能包含少量背景）
目标任务	通用目标检测（能识别 80 类常见物体，但对 "刀" 的识别能力弱或无）	专项目标检测（仅擅长识别 "刀"，对其他目标的识别能力可能退化）
权重参数	保存的是模型在 COCO 数据集上学习到的 **通用特征权重**（如边缘、纹理、基础形状等）	保存的是在 `yolov8n.pt` 基础上，针对 "刀" 的特征 **微调后的权重**（强化了刀的轮廓、形状等专属特征）
适用场景	作为 "预训练权重" 用于迁移学习（加速新任务训练），或直接检测 COCO 中的 80 类目标	仅用于检测 "刀"（在你的场景中，对刀的识别精度远高于 `yolov8n.pt`）



二、为什么 yolov8n.pt 既能训练又能推理？

yolov8n.pt 本质是一个 **"训练好的完整模型文件"**，包含了 YOLOv8 网络的结构和预训练权重，因此具备双重功能：

1. 用于 "训练"：作为 "预训练权重" 加速新任务学习（迁移学习）

当你训练 "识别刀" 的模型时，yolov8n.pt 的作用是 **提供初始权重**，帮助模型快速学习新任务：

- 它已经在 COCO 数据集上学习了大量通用视觉特征（如 "边缘""圆形""矩形" 等基础形状，"金属光泽""木质纹理" 等材质特征），这些特征对识别 "刀" 是有帮助的（刀有特定的轮廓、金属纹理等）。
- 训练时，模型会在这些通用特征的基础上，进一步学习 "刀" 的专属特征（如刀刃形状、刀柄结构等），最终得到 best.pt。
- 若没有 yolov8n.pt，模型需要从随机权重开始学习，不仅训练速度慢，还容易过拟合（尤其是你的数据集较小时）。

2. 用于 "推理"：直接检测其训练过的目标（COCO 80 类）

yolov8n.pt 本身是在 COCO 数据集上训练完成的模型，因此可以直接用于推理：

- 当你用 yolov8n.pt 做检测时，它能识别出 COCO 数据集中包含的 80 类目标（比如人、车、苹果等）。
- 但由于它没学过 "刀" 的特征，所以对 "刀" 的识别效果很差（可能漏检或误判），这也是你需要用自己的数据集重新训练的原因。

三、一句话总结

- yolov8n.pt 是 "通用基础模型"，像一本 "百科全书"，啥都懂一点，但不精通你的特定目标（刀）；
- best.pt 是 "专项优化模型"，像一本 "刀的详解手册"，只专注于刀的识别，精度更高。

yolov8n.pt 能训练是因为它提供了通用知识作为学习起点，能推理是因为它本身已经学完了通用目标的检测能力。