自定义模型训练

1.数据集标注

准备好数据集（建议每种目标有 100 张图片）、放置在 dataset/images 文件夹中，orin 主控镜像中已安装好 labelImg 标注工具，在终端中输入 labelImg

图 1 打开 labelImg

点击前请按照下图进行设置：切换标签格式为 YOLO、工具栏中选择查看并确保“自动保存模式”已勾选

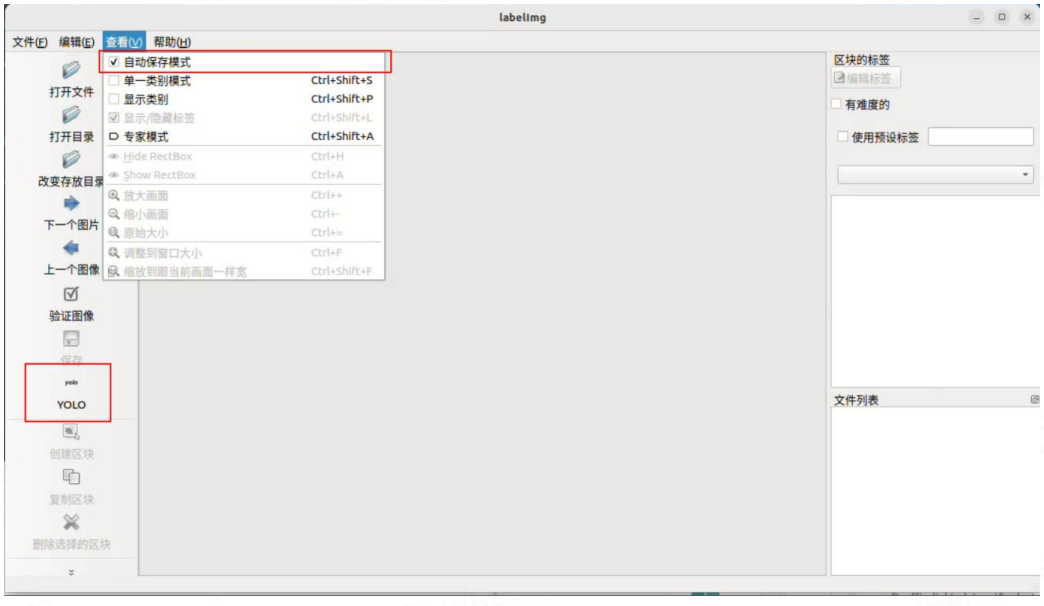

图 2 labelImg 设置示意

点击改变存放目录，选择~/dataset/labels，完成后点击打开目录，选择存放数据集的路径~/dataset/image，完成后在窗口中会显示数据集图片，按下 w 可以后可以进行标注，标注完成并输入目标名称后可以按“D”切换下一张图片，标注完成后~/dataset/labels 文件夹中，全部标注完成后 labels 文件夹中会新增和 images 文件夹中相同名称对应的.txt 标签文件

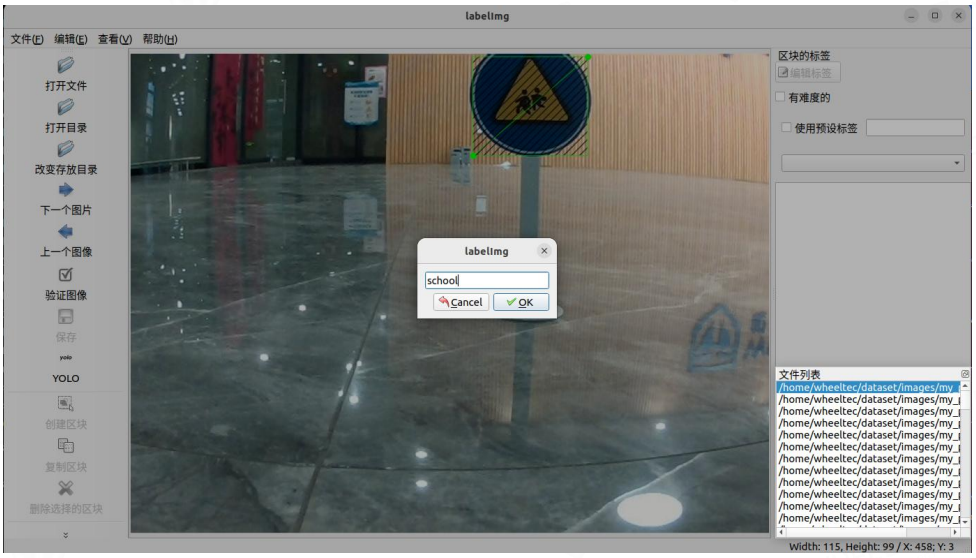

图 3 标注示意

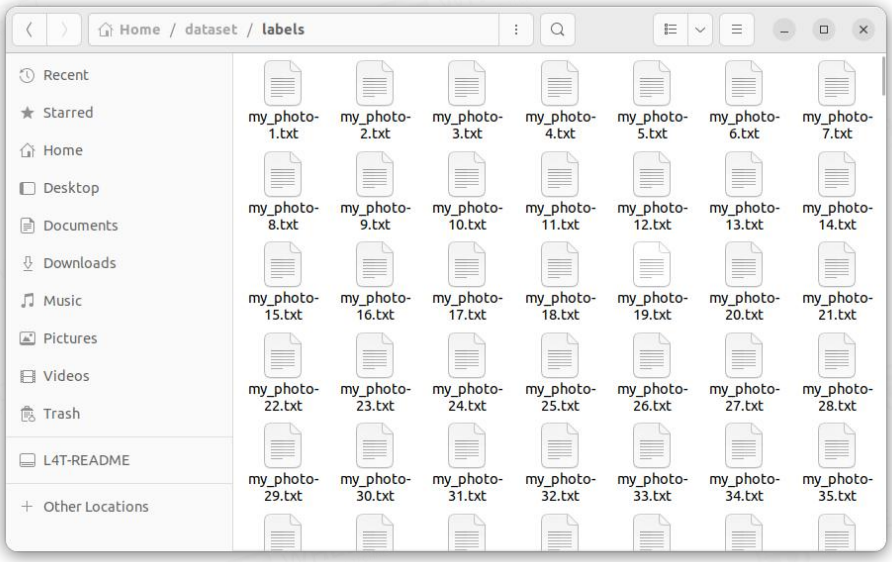

图 4 标签文件

2.训练前准备

接下来将资料包中的 classify.py、train.py、yolov8n.pt、data.yaml 文件放到~/dataset 目录下，如图所示

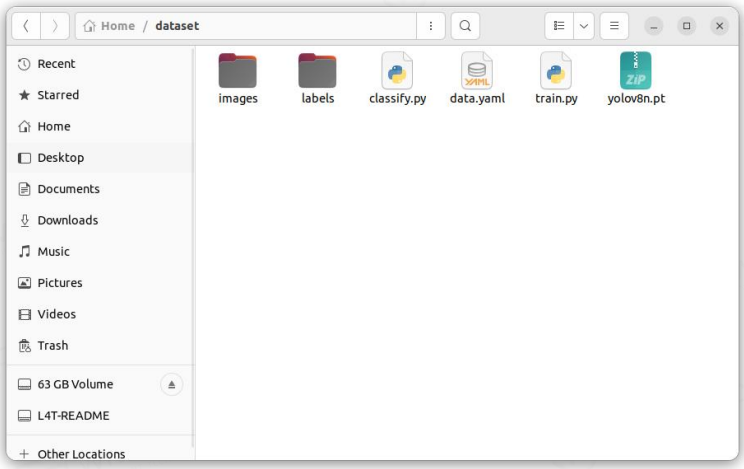

图 5 文件存放

接下来打开终端并进入到~/dataset，执行：

python classify.py

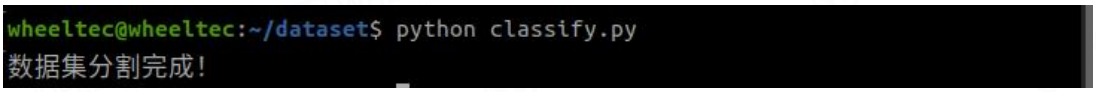

图 6 分割数据集

运行后终端打印“数据集分割完成”，~/dataset 路径下多出了两个文件夹：train 和 val，数据集按照 8：2 的比例分为了训练集和验证集，可修改源码对应位置对分割比例进行设置。

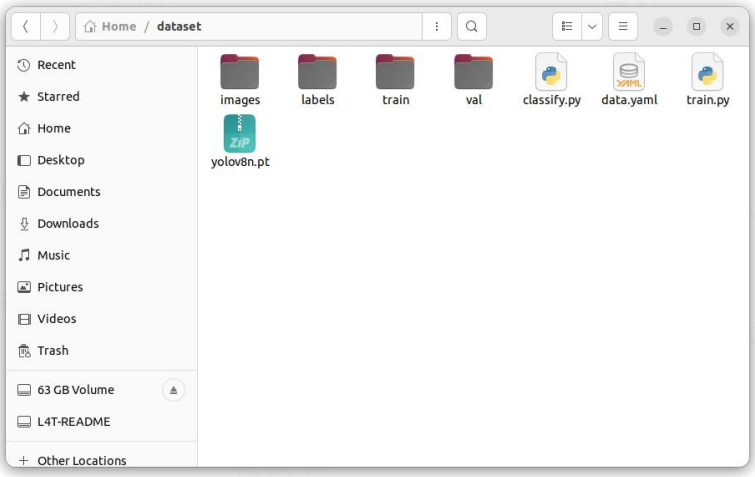

图 7 运行后文件夹架构

```
train_labels_dir = os.path.join(train_dir, 'labels')
val_images_dir = os.path.join(val_dir, 'images')
val_labels_dir = os.path.join(val_dir, 'labels')

os.makedirs(train_images_dir, exist_ok=True)
os.makedirs(train_labels_dir, exist_ok=True)
os.makedirs(val_images_dir, exist_ok=True)
os.makedirs(val_labels_dir, exist_ok=True)

# 获取所有图片文件
image_files = [f for f in os.listdir(images_dir) if f.endswith('.jpg')]
random.shuffle(image_files)  # 随机打乱

# 计算训练集和验证集的分割点
split_point = int(len(image_files) * 0.8)

# 分割数据集
train_files = image_files[:split_point]
val_files = image_files[split_point:]

```

图 8 分割比例设置

接下来修改 data.yaml 配置文件，train/val 后面填写分割后训练集/验证集图片的绝对路径，nc 为标注检测目标种类的数目，name 为标注的目标名称，注意和第一节中名称保持一致，同时注意 data.yaml 中标签名称的顺序也需和第一节中标注目标的先后顺序保持一致。

```
train: /home/wheeltec/traffic_sign_dataset/train/images
val: /home/wheeltec/traffic_sign_dataset/val/images


nc: 10
names: ['school','turn','slow','straight','stop','bus','parking','crossing_sign','construction','crossing']
```

图 9 配置文件

3.训练自定义模型

完成训练集/验证集分割之后可以开始训练自己的模型了，训练环境在 anaconda 虚拟环境中，所以需要先进入 anaconda 虚拟环境，执行下面的指令：

```
cd
sudo gedit .bashrc
```

修改位置如下图

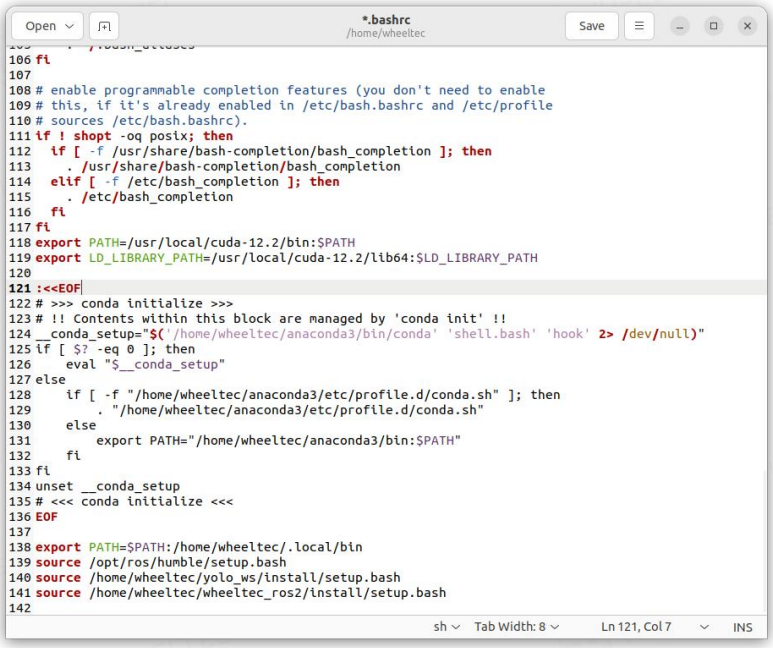

图 10 修改前示意

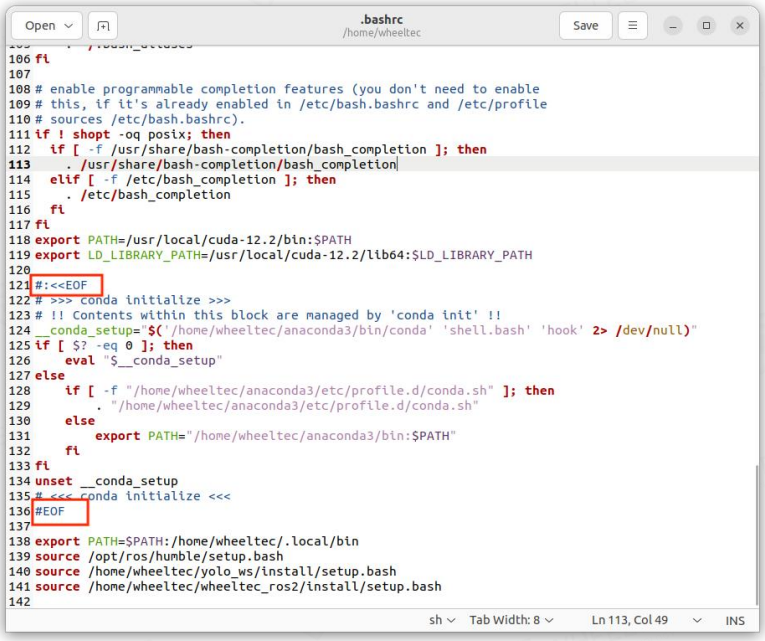

图 11 修改后示意

修改后保存重新打开终端，终端中输入

```
conda activate wheeltec
```

图 12 进入虚拟环境

即可进入配置好的训练环境。

进入 dataset 目录并在 dataset 目录下执行

```
python train.py
```

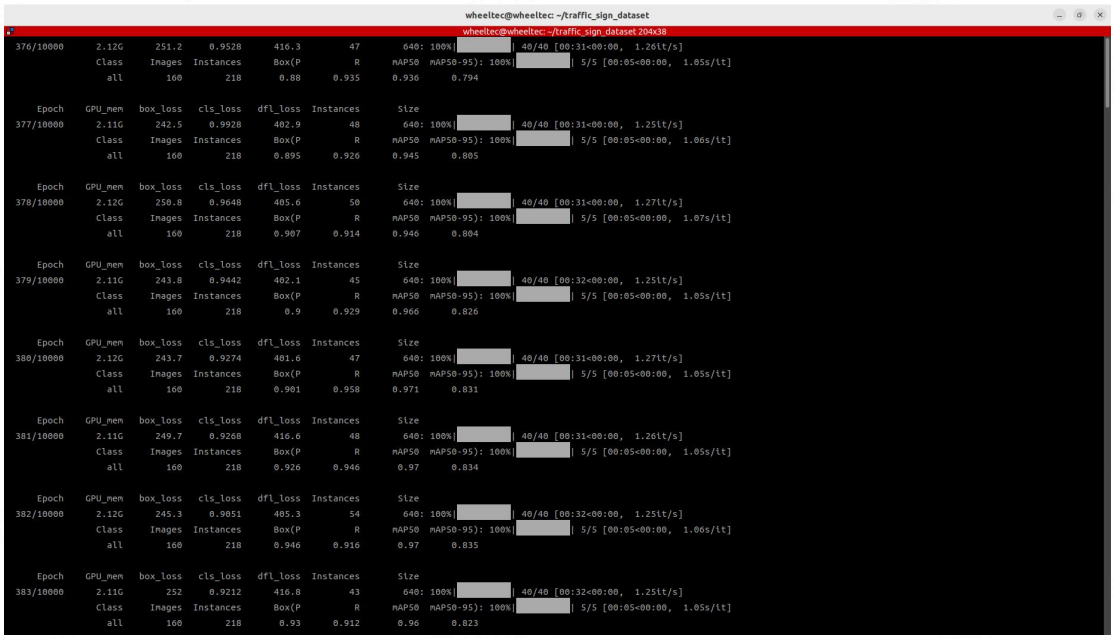

图 12 训练示意

等待训练完成后会在~/dataset 路径下生成 runs 文件夹，训练好的.pt 模型文件就在~/dataset/runs/detect/train/weights 路径中，last.pt 就是训练得到的模型。

4.设置训练超参数

利用 ultralytics 库训练非常简单，只需三行代码即可开始训练，可以根据需求修改默认超参数。

在 train.py 对应位置设置所需的超参数

```
dataset > train.py
1  from ultralytics import YOLO
2
3  model = YOLO("yolov8n.pt")
4
5  results = model.train(data="data.yaml",epochs=2000,imgsz=640,save_period=500)
6
```

图 12 训练超参设置

model 为预训练权重文件，默认提供 yolov8n.pt，官方还提供了其他模型可供使用

https://docs.ultralytics.com/models/yolov8/#supported-tasks-and-modes

Performance Metrics

Performance

Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) | Pose (CO >

See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP val 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

图 13 模型下载

```
results = model.train()
```

括号中可以根据实际使用需求添加超参数，常用的超参数有

```
'epochs': 100,          # 训练轮数
'batch': 16,            # 批量大小
'imgsz': 640,           # 输入图像大小
'device': '0',          # 使用的设备，'0' 表示 GPU，'cpu' 表示 CPU
'workers': 8,           # 数据加载的工作线程数
'lr0': 0.01,            # 初始学习率
'momentum': 0.937,      # 动量
'weight_decay': 0.0005, # 权重衰减
'warmup_epochs': 3.0,   # 预热轮数
'warmup_momentum': 0.8, # 预热动量
'warmup_bias_lr': 0.1,  # 预热偏置学习率
'box': 7.5,             # 框损失权重
'cls': 0.5,             # 分类损失权重
'dfl': 1.5,             # 分布焦点损失权重
'fl_gamma': 0.0,        # 焦点损失 gamma
'label_smoothing': 0.0, # 标签平滑
'nbs': 64,              # 名义批量大小
'save_period': -1,      # 保存模型的周期，-1 表示不保存中间模型
'project': 'runs/train', # 项目保存路径
'name': 'exp',          # 实验名称
'exist_ok': False,      # 是否允许覆盖现有实验
'pretrained': True,     # 是否使用预训练权重
'optimizer': 'auto',    # 优化器，'auto', 'SGD', 'Adam', 'AdamW'
'verbose': True,        # 是否打印详细信息
'seed': 0,              # 随机种子
'deterministic': True,  # 是否确定性训练
'single_cls': False,    # 是否单类别训练
'rect': False,          # 是否矩形训练
'cos_lr': False,        # 是否使用余弦学习率调度
'close_mosaic': 10,     # 最后多少轮关闭 mosaic 数据增强
'resume': False,        # 是否从上次的训练结果恢复
'amp': True,            # 是否使用混合精度训练
'fraction': 1.0,        # 数据集使用比例
'profile': False,       # 是否进行性能分析
```

图 14 常用超参