

# 说明文档

作者：逗逗班学Python

面包多：<https://mbd.pub/o/deepcode>

CSDN：<https://deepcode.blog.csdn.net/>

BiliBili：<https://space.bilibili.com/582341464>

博客园：<https://www.cnblogs.com/deeppython>

知乎：<https://www.zhihu.com/people/deeppython>

原创代码，仅用于学习交流，  
受版权保护，禁止他人商用和盗卖，盗版追究法律责任！

.idea  
datasets  
icon  
runs  
tempDir  
test\_media  
ultralytics  
weights  
\_\_init\_\_.py  
LoggerRes.py  
Recognition\_UI.py  
requirements.txt  
run\_main\_web.py  
run\_test\_camera.py  
run\_test\_image.py  
run\_test\_video.py  
run\_train\_model.py  
style\_css.py  
utils\_web.py  
YOLOv8v5Model.py  
说明文档-网页版.pdf

◆ 功能演示

◆ 软件安装

◆ 环境配置

◆ 项目介绍

◆ 修改项目

# 1. 软件安装

## Pycharm和Anaconda安装教程

本项目运行会用到Pycharm和Anaconda两个软件，其安装教程在**对应资源博客下贴有教程的链接**！初学者可以按照上面的步骤顺利配置好。

也可见本人CSDN博客（[逗逗班学Python](#)）、B站视频（[逗逗班学Python](#)）或者到主页搜索“安装教程”

## 2. 环境配置

### 项目环境配置教程

除了Pycharm和Anaconda两个软件，**在运行之前需要按照作者在资源页面给定的Python版本，并按照项目文件夹中的requirements.txt中的依赖库版本配置**

Python环境！（这步很重要，请按照给定版本安装，以免出现版本不兼容问题）

这部分在对应资源博客下贴有安装教程的链接！初学者可以按照上面的步骤顺利配置好。

也见本人CSDN博客（[逗逗班学Python](#)）、B站视频（[逗逗班学Python](#)）

或者到主页搜索“环境配置”

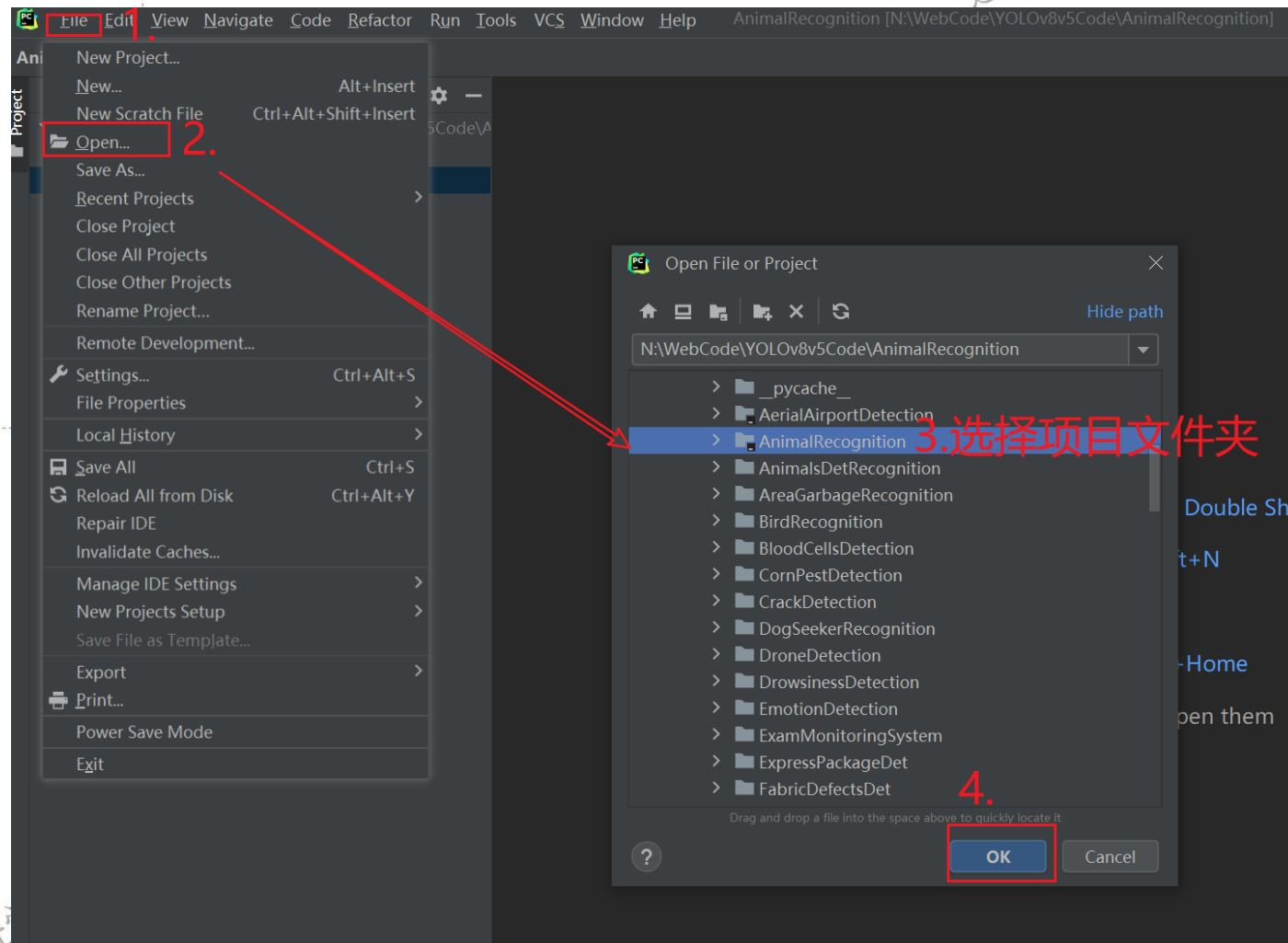
# 项目介绍

在前面的安装和配置工作后，  
使用pycharm后打开项目：

如右图，依次打开项目文件夹

(**注意**：解压出的项目文件夹有时可能  
嵌套了一个同名文件夹，打开项目时选择  
最里面一层文件夹)

点击File→Open→项目文件夹→OK打开  
项目

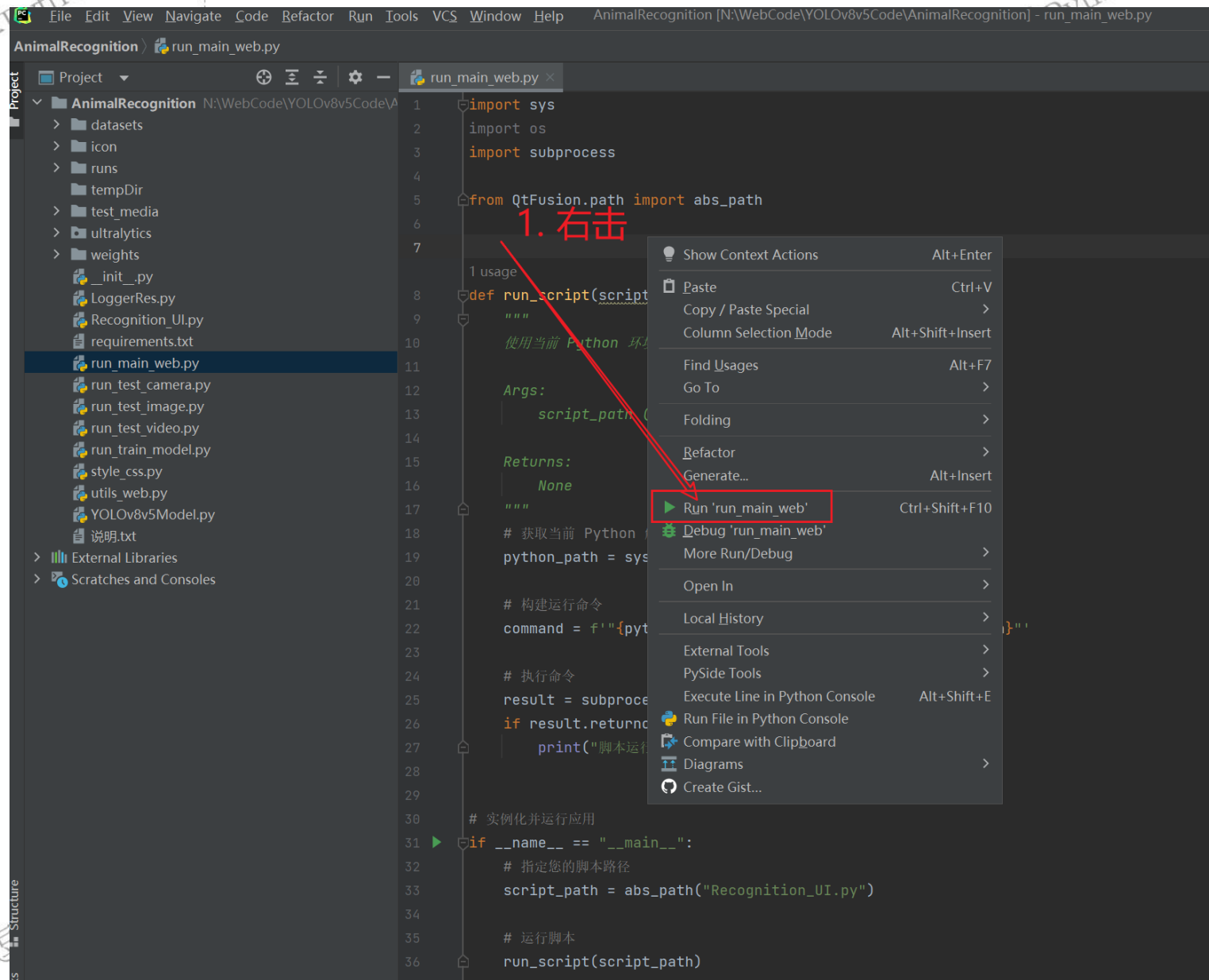


# 项目介绍

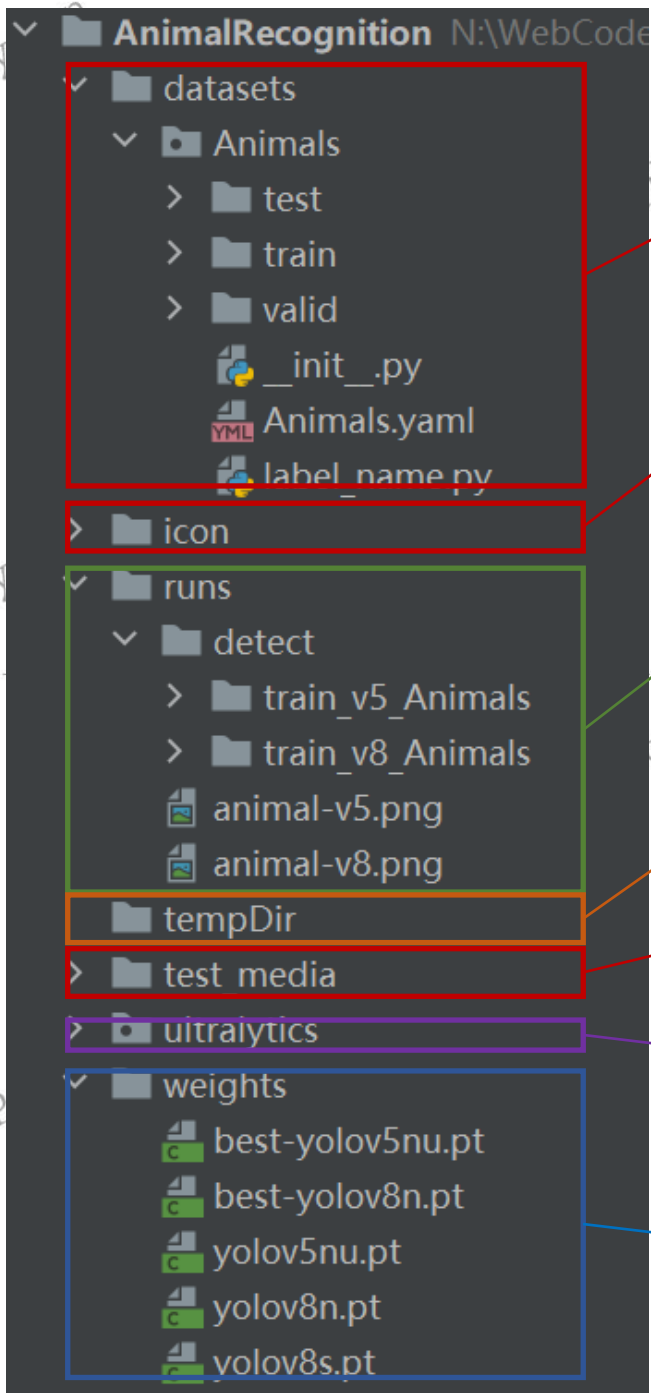
打开项目后的pycharm界面如右图

(需要在前面步骤中已经在  
Pycharm中选择了python环境)

这时打开run\_main\_web.py文件，  
右击选择“Run\_main\_web”即可  
运行主程序，会自动弹出网页







**数据集文件夹**，用于模型训练和评估的数据集，包含训练、测试、验证集图片和标注文件，可在此文件夹中添加自定义数据集进行训练（保持文件夹目录结构一致）

**图标文件夹**，存放用于项目界面中的图标文件，包含界面中所需的背景、图标文件，可自行替换修改

**训练结果文件夹**，用于存储模型训练过程中生成的日志、训练结果、评估图表，包含不同模型训练后的pt模型文件、训练结果、图表文件，可查看训练和评估的结果；

**暂存文件夹**，用于存放临时文件，在程序运行过程中保存的结果csv文件以及保存的标记视频结果文件会保存在这里；

**测试图片和视频文件夹**，用于测试模型性能的媒体文件，包含用于测试的图片、视频文件；

**YOLO的包**，官方代码包，Ultralytics YOLO模型有关的脚本或模块，包含模型定义、配置、工具等文件；

**模型文件夹**，存储训练好的模型权重文件，包含预训练模型pt文件，以及在本项目数据集训练好的模型pt文件；

- > datasets
- > icon
- > runs
- tempDir
- > test\_media
- > ultralytics
- > weights

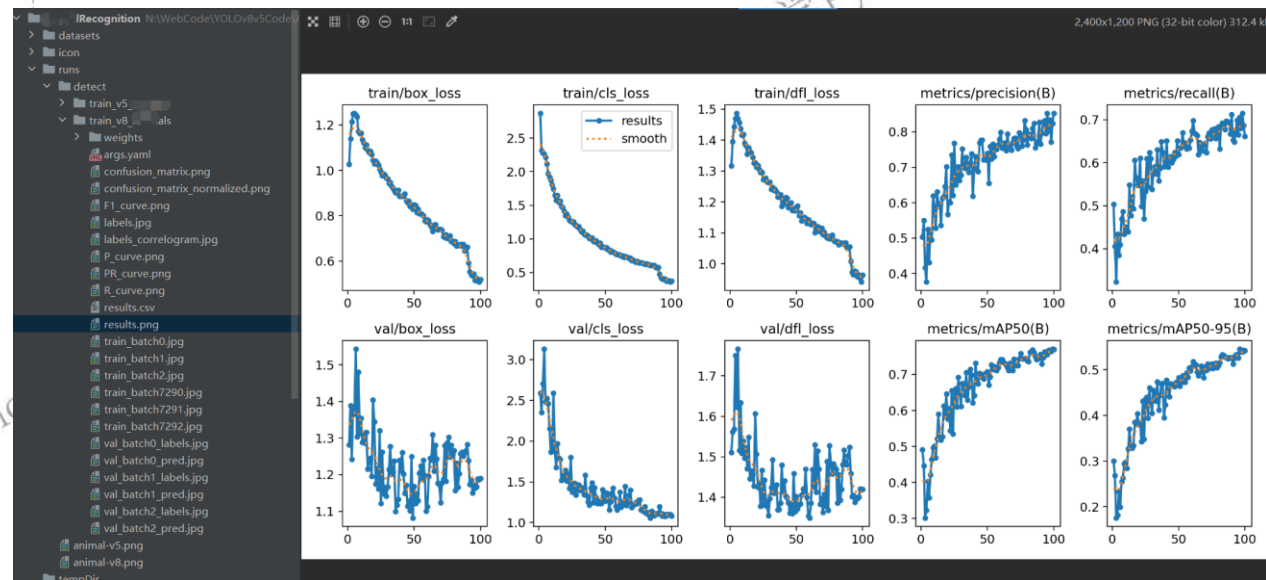
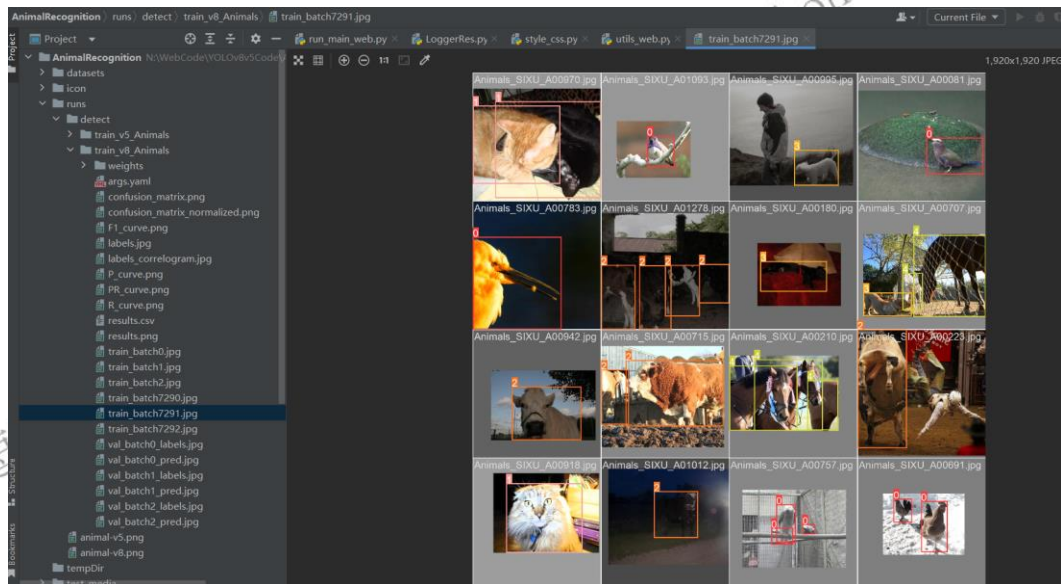
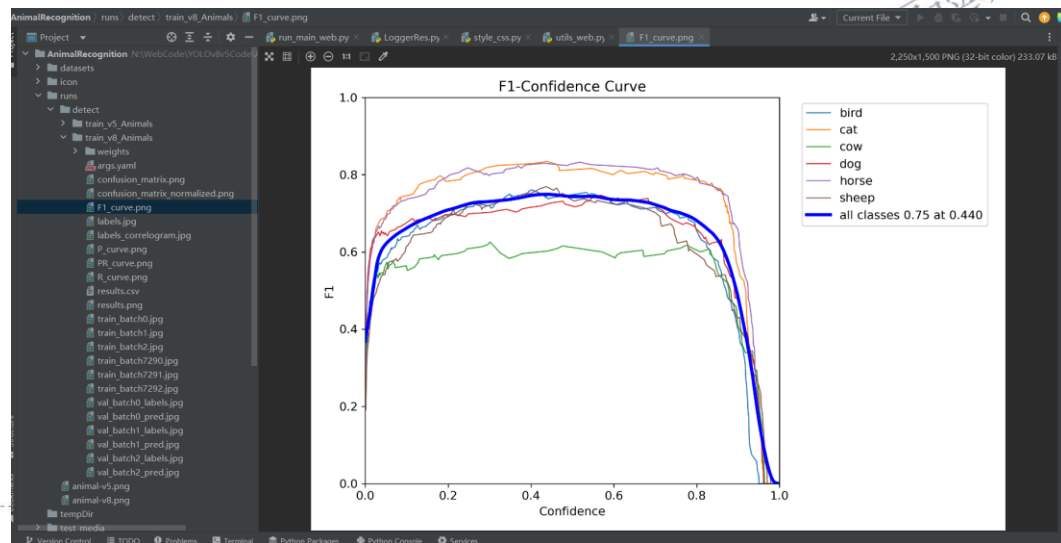
- \_init\_.py
- LoggerRes.py
- Recognition\_UI.py
- requirements.txt
- run\_main\_web.py
- run\_test\_camera.py
- run\_test\_image.py
- run\_test\_video.py
- run\_train\_model.py
- style\_css.py
- utils\_web.py
- YOLOv8v5Model.py
- 说明文档-网页版.pdf

- `__init__.py` - Python包初始化文件，允许Python知道该目录可以被视为一个包；
- `LoggerRes.py` - 处理页面结果记录和保存的类代码，记录检测结果到表格并保存为视频或csv文件等；
- `Recognition_UI.py` - 主界面的布局py代码，包含本项目主界面的创建、生成、布局逻辑；
- `requirements.txt` - 包含项目依赖项及其版本的文本文件，用于设置开发环境。
- `run_main_web.py` - 启动网页功能的主运行脚本，点击运行后进入检测主页面。
- `run_test_camera.py` - 用于测试摄像头输入的脚本，直接画面标记检测结果显示。
- `run_test_image.py` - 用于测试图像识别功能的脚本，直接画面标记检测结果显示。
- `run_test_video.py` - 用于测试视频流识别功能的脚本，直接画面标记检测结果显示。
- `run_train_model.py` - 启动模型训练过程的脚本，可重新训练。
- `style_css.py` - 包含页面的css样式表，美化界面效果，设置页面显示样式的代码。
- `utils_web.py` - 项目工具代码，包括保存上传文件、显示检测结果、加载图片等函数。
- `YOLOv8v5Model.py` - 包含YOLO模型相关代码的Python脚本。
- `环境配置.txt` - 包含环境配置说明的文本文件。
- `说明文档-网页版.pdf` - 项目的详细说明文档。

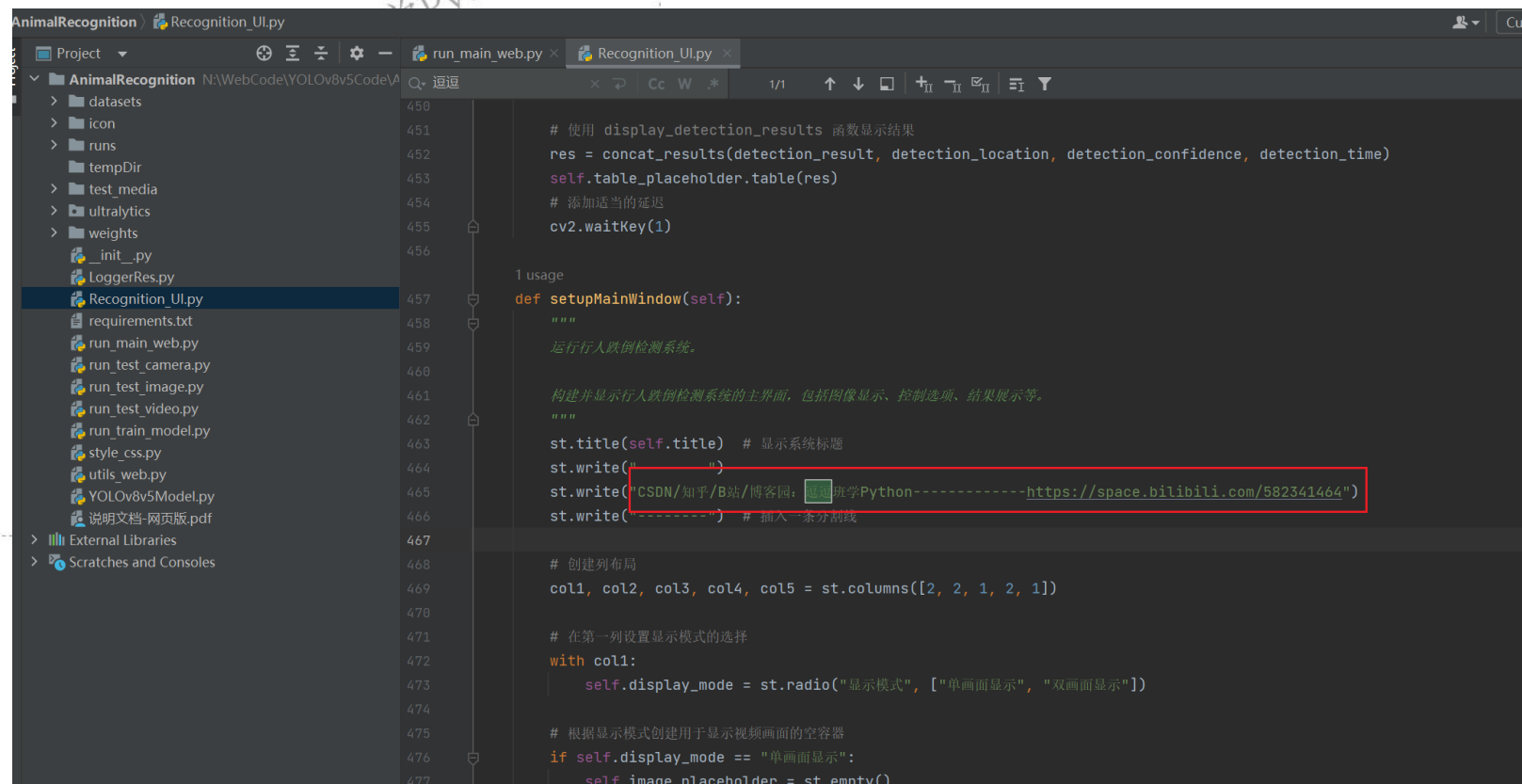


# 项目介绍

在项目文件夹下的runs文件夹中，可以找到训练和评估的结果图表



## 修改项目



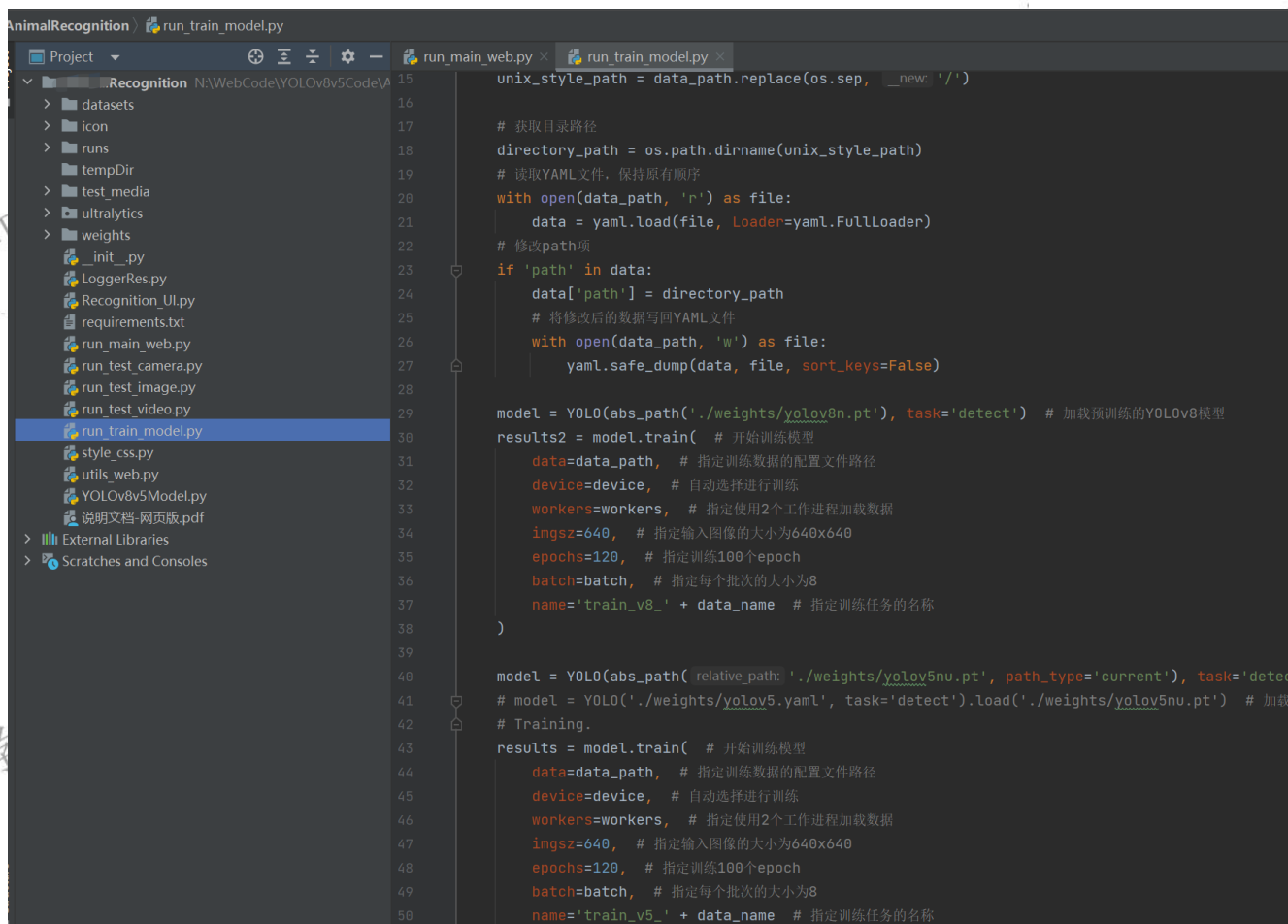
```
450
451 # 使用 display_detection_results 函数显示结果
452 res = concat_results(detection_result, detection_location, detection_confidence, detection_time)
453 self.table_placeholder.table(res)
454 # 添加适当的延迟
455 cv2.waitKey(1)
456
457 1 usage
458 def setupMainWindow(self):
459     """
460     运行行人跌倒检测系统。
461
462     构建并显示行人跌倒检测系统的主界面，包括图像显示、控制选项、结果展示等。
463     """
464     st.title(self.title) # 显示系统标题
465     st.write(" ")
466     st.write("CSDN/知乎/B站/博客园: 逗逗班学Python-----https://space.bilibili.com/582341464")
467     st.write("-----") # 插入一条分割线
468
469 # 创建列布局
470 col1, col2, col3, col4, col5 = st.columns([2, 2, 1, 2, 1])
471
472 # 在第一列设置显示模式的选择
473 with col1:
474     self.display_mode = st.radio("显示模式", ["单画面显示", "双画面显示"])
475
476 # 根据显示模式创建用于显示视频画面的空容器
477 if self.display_mode == "单画面显示":
478     self.image_placeholder = st.empty()
```

在项目文件夹中的Recognition\_UI.py文件中可以修改网面上的文字等

比如，将上图中的作者信息的text替换成“**你的信息**”，再次运行可在页面修改过来；

# 重新训练

运行run\_train\_model.py文件可以重新进行训练，其中如果安装的是gpu版本的pytorch会自动启动gpu训练，如果是cpu版本的pytorch则使用cpu运行；本项目中已经保存了训练好的模型和训练结果，可以不用再次训练而直接使用；



```
15 unix_style_path = data_path.replace(os.sep, '/')
16
17 # 获取目录路径
18 directory_path = os.path.dirname(unix_style_path)
19 # 读取YAML文件，保持原有顺序
20 with open(data_path, 'r') as file:
21     data = yaml.load(file, Loader=yaml.FullLoader)
22 # 修改path项
23 if 'path' in data:
24     data['path'] = directory_path
25 # 将修改后的数据写回YAML文件
26 with open(data_path, 'w') as file:
27     yaml.safe_dump(data, file, sort_keys=False)
28
29 model = YOLO(abs_path('./weights/yolov8n.pt'), task='detect') # 加载预训练的YOLOv8模型
30 results2 = model.train( # 开始训练模型
31     data=data_path, # 指定训练数据的配置文件路径
32     device=device, # 自动选择进行训练
33     workers=workers, # 指定使用2个工作进程加载数据
34     imgsz=640, # 指定输入图像的大小为640x640
35     epochs=120, # 指定训练100个epoch
36     batch=batch, # 指定每个批次的大小为8
37     name='train_v8_' + data_name # 指定训练任务的名称
38 )
39
40 model = YOLO(abs_path(relative_path='./weights/yolov5nu.pt', path_type='current'), task='detect')
41 # model = YOLO('./weights/yolov5.yaml', task='detect').load('./weights/yolov5nu.pt') # 加载
42 # Training.
43 results = model.train( # 开始训练模型
44     data=data_path, # 指定训练数据的配置文件路径
45     device=device, # 自动选择进行训练
46     workers=workers, # 指定使用2个工作进程加载数据
47     imgsz=640, # 指定输入图像的大小为640x640
48     epochs=120, # 指定训练100个epoch
49     batch=batch, # 指定每个批次的大小为8
50     name='train_v5_' + data_name # 指定训练任务的名称
```

1. 内存较小的电脑可以将batch的数值设置更小一些防止内存不够报错；
2. 训练采用的数据集存放在datasets文件夹下，训练过程中的结果保存在runs文件夹下的子文件夹中；
3. 每次运行训练会自动添加一个文件夹，最终训练得到的模型pt文件也会在该文件夹中；