

搞定系列：yolox检测封装成类调用 (云未归来)

课程链接：

1. <https://download.csdn.net/course/detail/36773>

你将收获

- 1、学会Yolox封装基本技巧和大体思路
- 2、学会Yolox封装类的API调用技巧和自由扩展
- 3、学会使用Pycharm调试技巧和运行脚本技巧
- 4、学会目标检测代码识别基本流程

适用人群

研究Yolox学者，想对Yolox二次开发的工程师以及对Yolox感兴趣的人员

课程介绍

本课程主要核心是将Yolox框架封装成一个类，方便调用或者集成开发以及集成部署。通过本课程您将收获一下知识：

- (1) 了解Yolox框架检测基本流程；
- (2) 掌握目标检测封装类大体思路；
- (3) 学会Yolox封装函数以测试；
- (4) 学会利用封装类模块，进行API调用。

本课程优点：

- (1) 去掉课程无关的讲解，直击课程核心出发点，避免花费更多时间学习自己已经掌握知识，而着重掌握封装本质上来；
- (2) 提供代码讲解和API示例，让同学们更快获取本课程核心内容；
- (3) 课程简洁，重点课程重点讲解。

学习本课程之前您需要准备的是（本课程不提供讲解）：

- (1) 您需要提前搭建好自己Yolox环境，建议使用Anaconda3搭建；
- (2) 您需要保证自己demo.py可以正常运行
- (3) 您需要准备官方提供模型或者自己训练的模型

课程目录

- ▶ 课程导论 免费
- ▶ 认识yolox框架
- ▶ 认识demo.py检测流程
- ▶ 封装类的框架搭建
- ▶ 封装yolox核心检测代码
- ▶ 封装绘制图片方框函数
- ▶ 封装摄像头与视频检测函数
- ▶ 封装api调用示例

0、课程导论

(略)

1、认识 yolox 框架

yolox 与其他框架比较 (图片来源: <https://blog.csdn.net/amusi1994/article/details/118932891>)

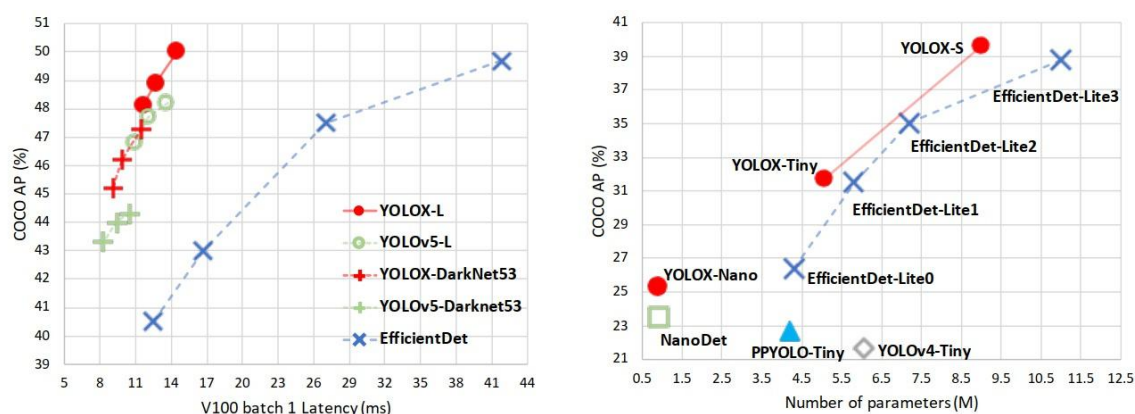
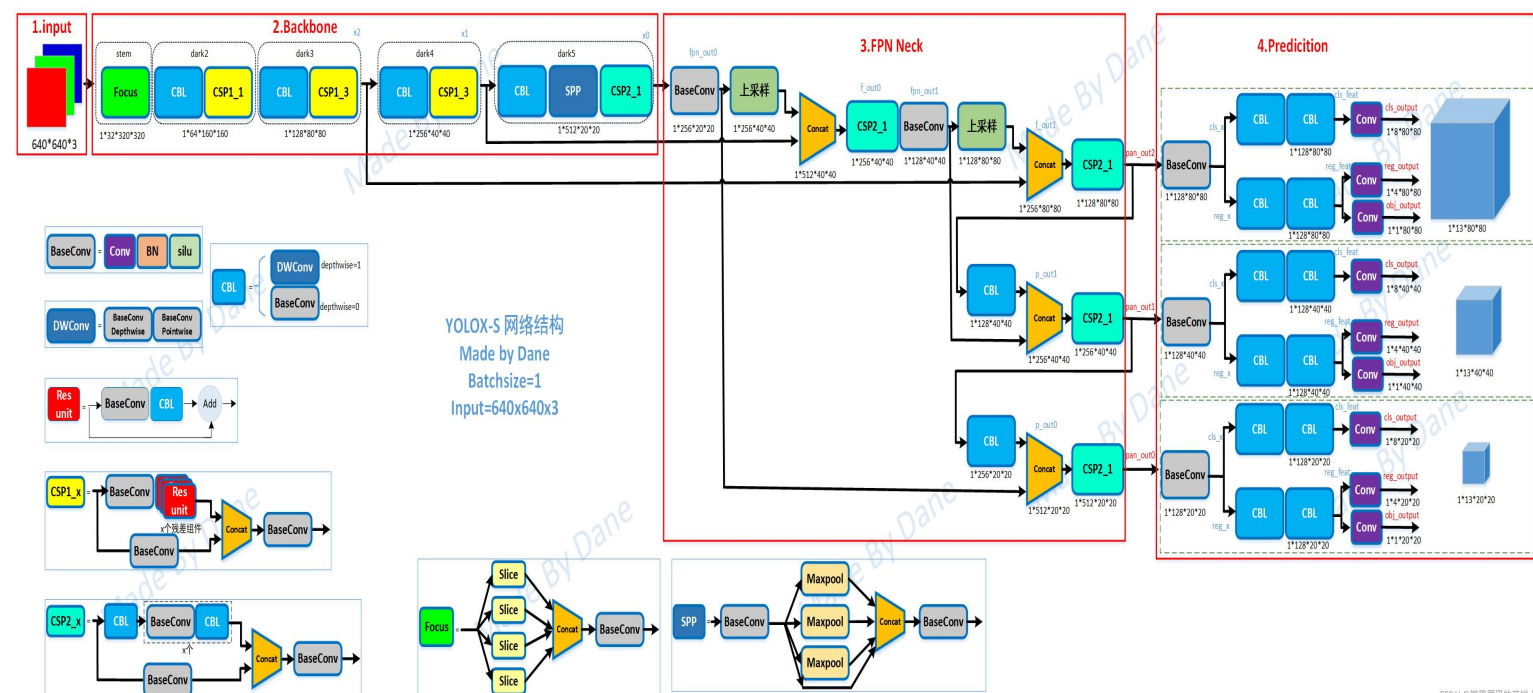


Figure 1: Speed-accuracy trade-off of accurate models (top) and Size-accuracy curve of lite models on mobile devices (bottom) for YOLOX and other state-of-the-art object detectors.

yolox 网络结构图



CSDN @ 聊天的墨迹

总结:

yolox 与 yolov5 比较

	yolov5	yolox-DarkNet53	yolox-SPP
输入端	1. Mosaic 2. 自适应锚框计算 3. 自适应图片放缩	1. Mosaic 2. MixUp (注: epoch=15时这两种 数据增强 方式关闭, 由于这种增强效果更好, ImageNet 的预训练无意义)	1. 增加了EMA权值更新 2. Cosine学习率机制 3. 增加了 RandomHorizontalFlip, ColorJitter, 多尺度数据增广, 马赛克数据增强, 不执行 RandomResizedCrop, 因为与马赛克重叠
backbone	1. Focus结构 2. CSP结构	1. DarkNet53	1. yolov3_SPP
Neck	FPN+PAN结构	1. FPN	1. FPN
Head	1. 训练时: GIOU_Loss 2. 预测时: DIOU_Loss	1. Decoupled Head 2. End-to-End yolo 3. Anchor-free 4. Multi positives	1. Strong augementation 2. Decoupled Head 3. anchor-free 4. multi positives 5. SimOTA

- 注:
- 在 Neck 部分: yolov4、yolov5、yolox-s、yolox-l 中采用的是 FPN+PAN
 - 在 yolox-SPP 中, 选用的是最基本的 yolov3-SPP, 如果采用 yolov3_ultralytics 的 yolov3-SPP + 四种 trick (除 strong augementation, 因为在此版本代码中已经改进)

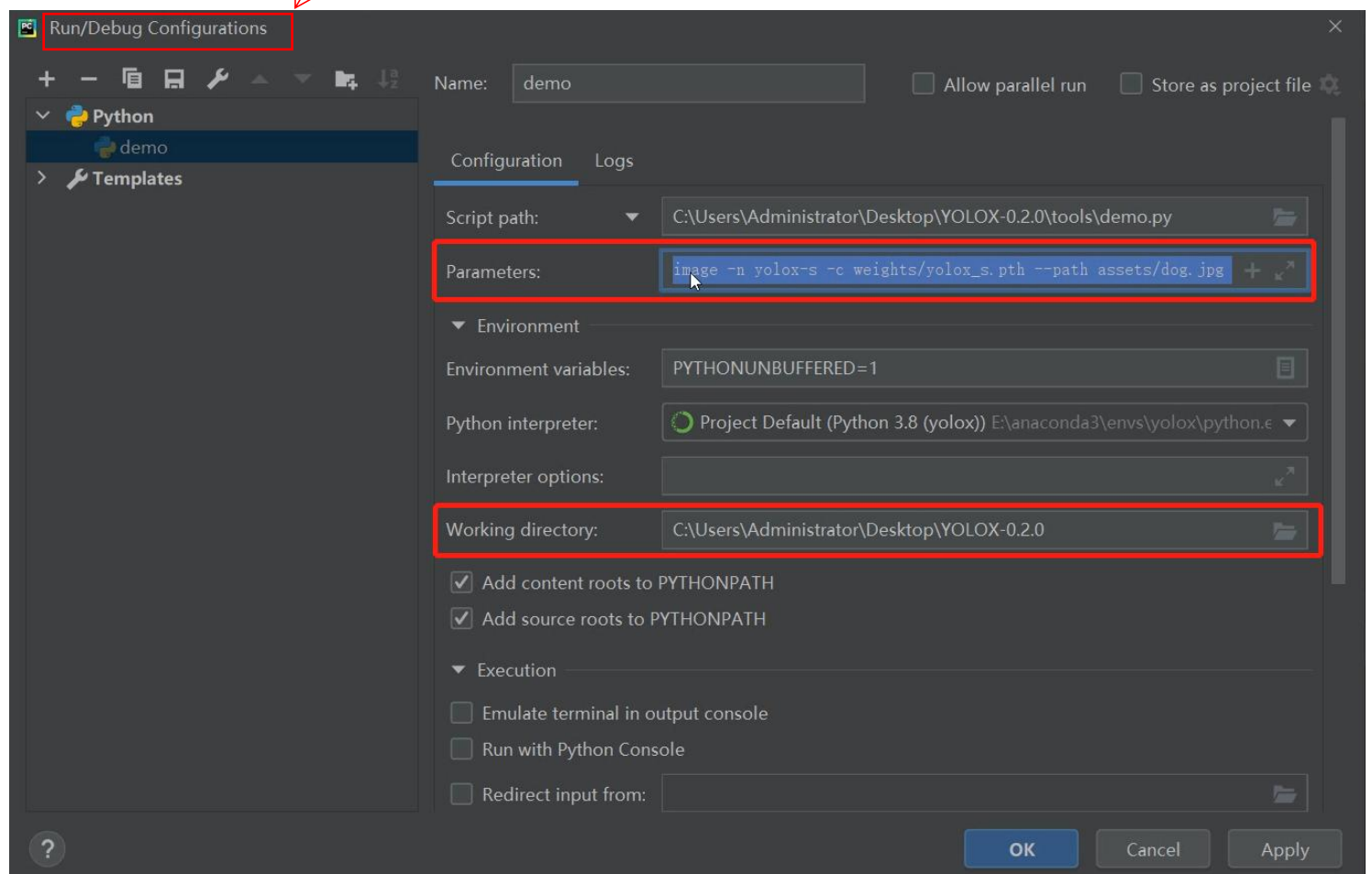
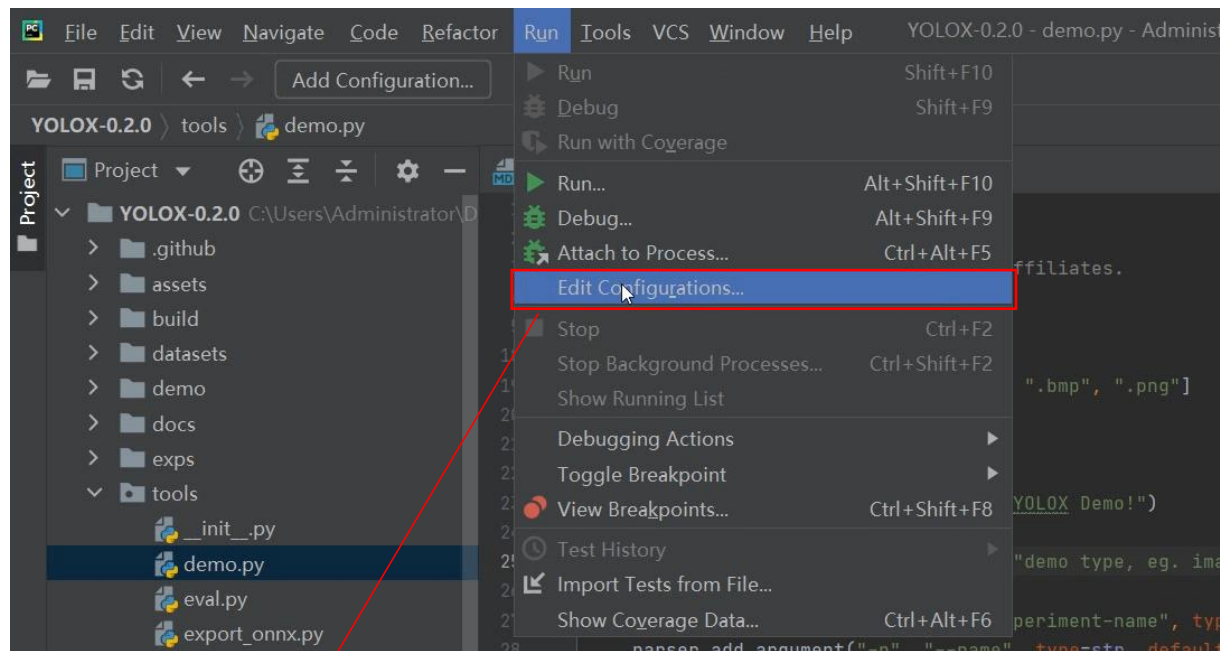
参考一篇比较好的: https://xw.qq.com/cmsid/20210823A056WE00?pgv_ref=baidutw

2、认识 demo.py 检测流程

yolox demo.py 检测流程

1. 加载模型结构和权重参数文件
2. 获取图像数据
3. 图像预处理
4. 送入模型
5. 推理数据后处理
6. 分析或者显示

请用 pycharm 调试看看代码走向，方便理解检测整个流程。

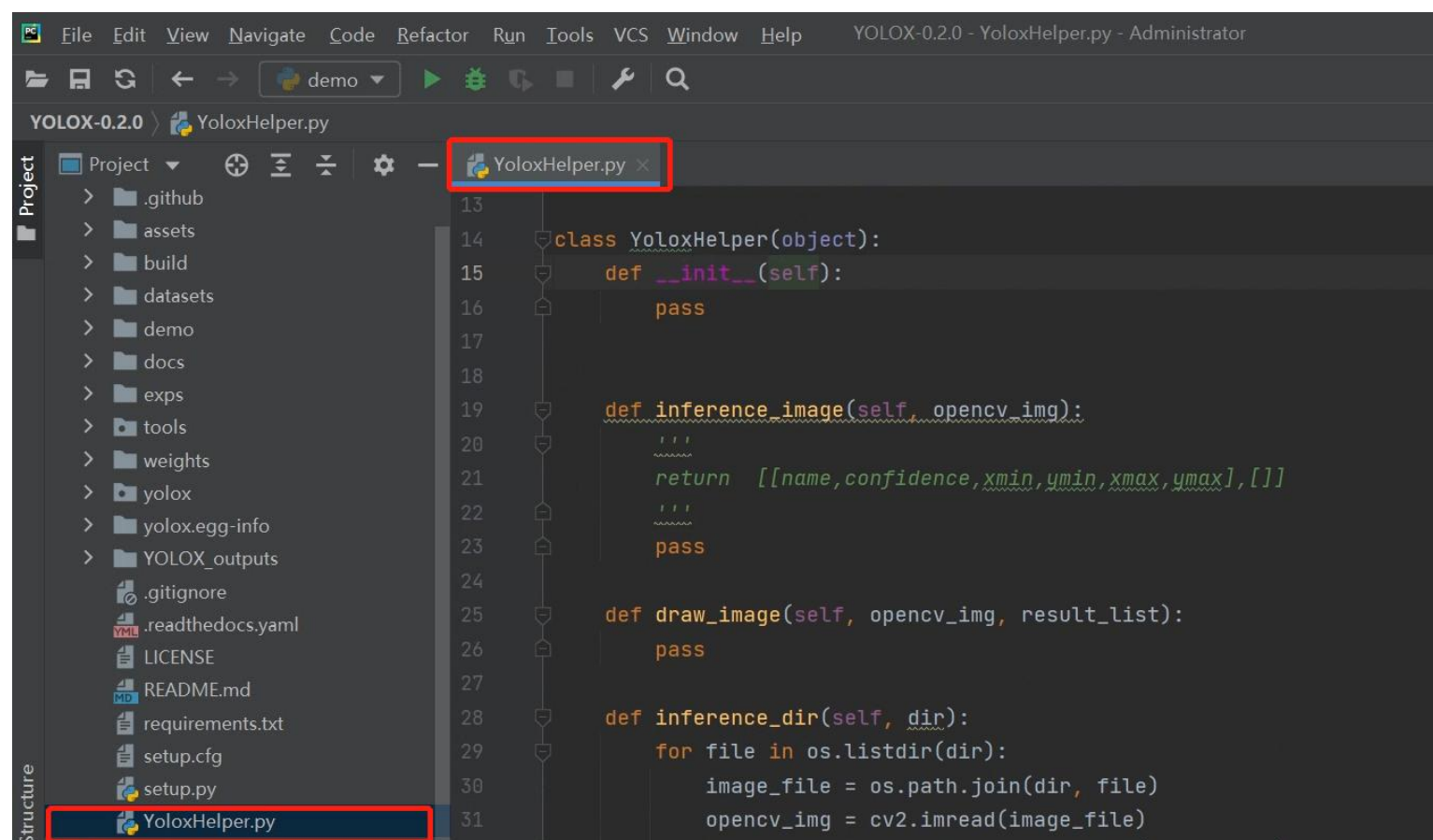


总结：

3、封装类的框架搭建

yolo封装类框架思路

1. 直接加载模型和权重参数，避免反复加载模型可以在构造函数完成加载模型任务。
2. 将图像预处理+推理+后处理放一个函数直接出结果，类似 [[name, confidence, xmin, ymin, xmax, ymax], ...]
3. 封装对一个图像处理函数则就可以拓展其他检测函数，比如视频、摄像头。
4. 在第3步基础上编写文件夹所有图片推理，视频，摄像头推理。
5. 为了能够自由开发，可以只要推理结果，然后二次开发自由发挥。为了更傻瓜式调用，所以加了个显示函数。



4、封装 yolo 核心检测代码

5、封装绘制图片方框函数

6、封装摄像头与视频检测函数

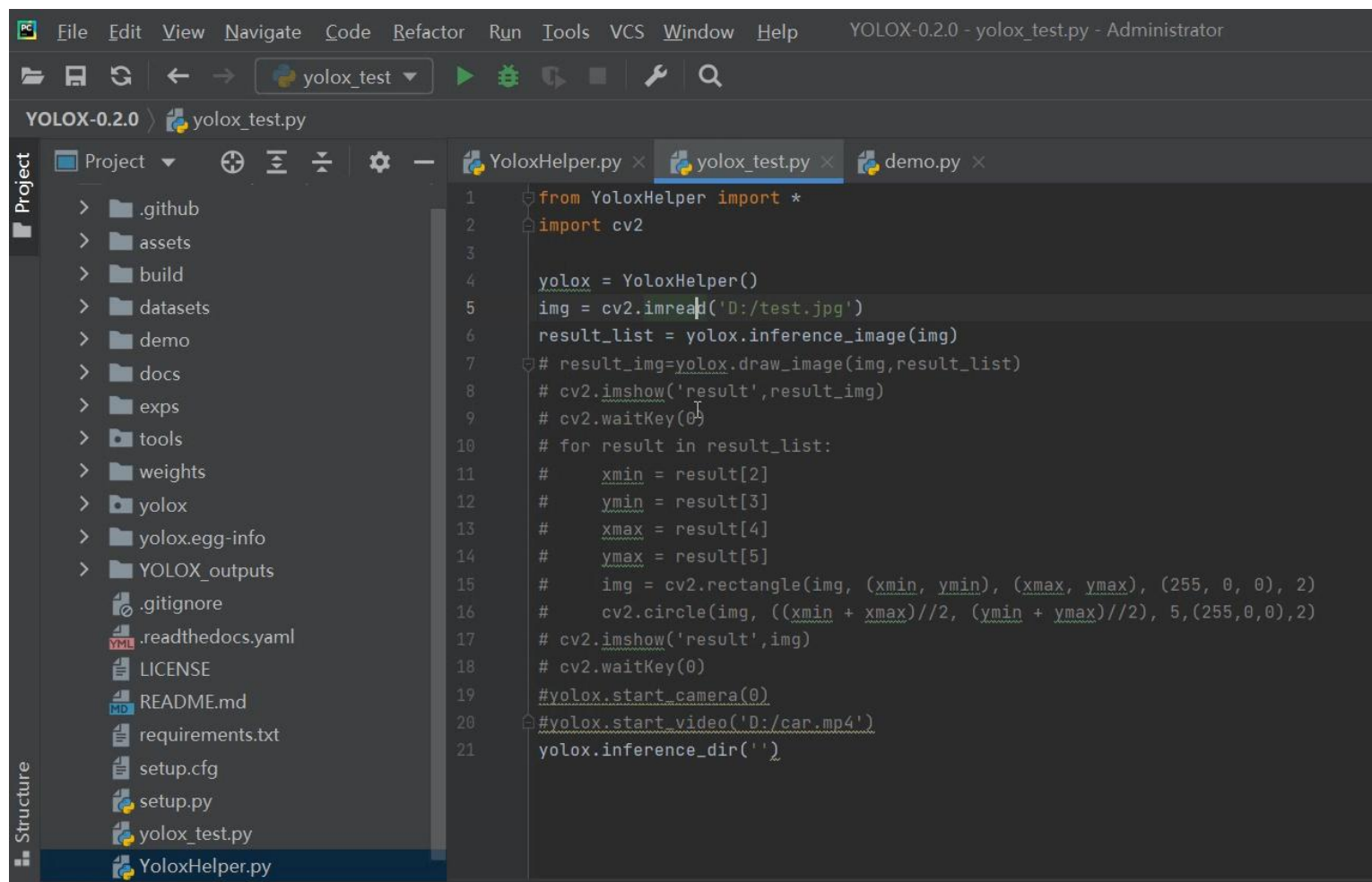
其实就是对上一节创建的封装类 YoloHelper 进行完善，往该类的框架内添加细节代码。添加的代码大部分是 copy 自 demo.py 文件，也有在网上找的代码，另外一部分是自己写的。

讲师自己写的 opencv 读取摄像头或者视频模板代码的文章：

<https://blog.csdn.net/FL1623863129/article/details/109180061>

总结：

7、封装 api 调用示例



The screenshot shows an IDE window titled "YOLOX-0.2.0 - yolo_test.py - Administrator". The left sidebar displays the project structure for "YOLOX-0.2.0", including folders like .github, assets, build, datasets, demo, docs, exps, tools, weights, yolo, yolo.egg-info, and YOLOX_outputs, as well as files like .gitignore, .readthedocs.yaml, LICENSE, README.md, requirements.txt, setup.cfg, setup.py, yolo_test.py, and YoloHelper.py. The main editor area shows the code in "yolo_test.py":

```
1 from YoloHelper import *
2 import cv2
3
4 yolo = YoloHelper()
5 img = cv2.imread('D:/test.jpg')
6 result_list = yolo.inference_image(img)
7 # result_img=yolo.draw_image(img,result_list)
8 # cv2.imshow('result',result_img)
9 # cv2.waitKey(0)
10 # for result in result_list:
11 #     xmin = result[2]
12 #     ymin = result[3]
13 #     xmax = result[4]
14 #     ymax = result[5]
15 #     img = cv2.rectangle(img, (xmin, ymin), (xmax, ymax), (255, 0, 0), 2)
16 #     cv2.circle(img, ((xmin + xmax)//2, (ymin + ymax)//2), 5, (255,0,0),2)
17 # cv2.imshow('result',img)
18 # cv2.waitKey(0)
19 #yolo.start_camera(0)
20 #yolo.start_video('D:/car.mp4')
21 yolo.inference_dir('')
```

所有代码：D:\KeepStudy\0_Projects\CSDN项目\yolox检测封装成类调用.py