# 机器人视觉

技术报告

# 实验一：单线与双线巡线算法实现

## 实验目的

1. 掌握机器人巡线任务中的传感器数据处理方法；

2. 实现PID控制算法对小车方向的动态调整；

3. 完成单线巡线（含S型路径和直角转弯）与双线巡线（含M型路径和环岛处理）任务。

## 实验内容

1. 通过红外传感器或摄像头采集地面黑线位置信息；

2. 编写PID控制算法实现单线巡线；

3. 设计状态机逻辑完成双线巡线中的M型路径跟踪和环岛半圆行驶；

4. 在仿真环境（`task1.world`和`task2.world`）中验证算法有效性。

## 实验器材

- ROS智能车仿真平台（Gazebo环境）

- 红外传感器/摄像头

- Ubuntu 18.04系统（ROS melodic）

- PID控制器代码模块

## 实验原理

1. 车道线识别：

利用opencv对摄像头中的图像进行处理提取轮廓计算出要到达的黑色车道线的中心点位置

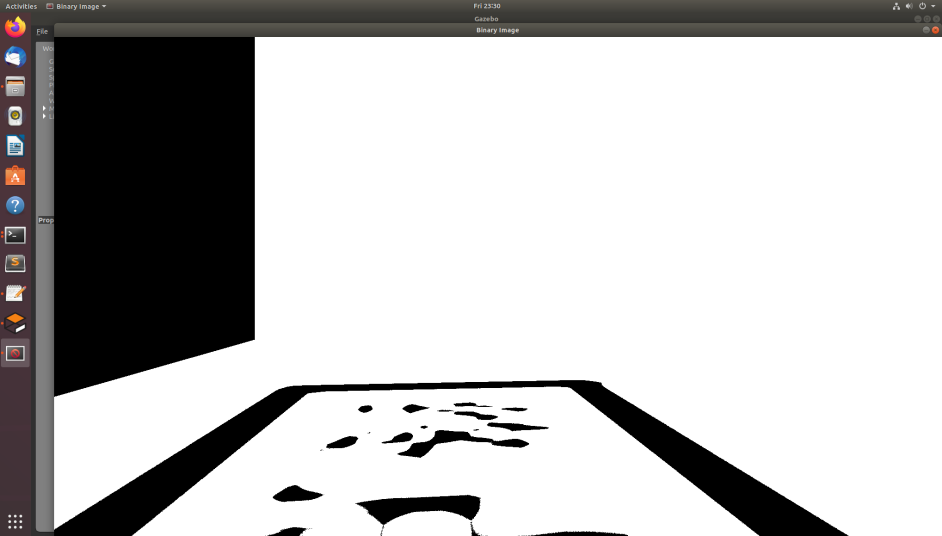
灰度化：将RGB图像转为灰度图（cv2.cvtColor），简化后续处理。

高斯模糊：用高斯滤波（cv2.GaussianBlur）消除噪声，平滑图像。

二值化：通过阈值分割（cv2.threshold）将黑色车道线与其他区域分离。

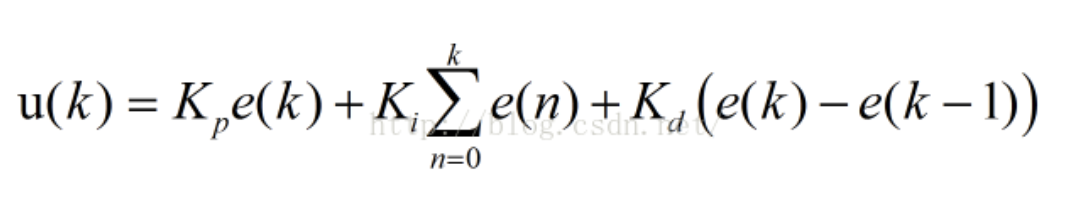
边缘检测：用Canny算法（cv2.Canny）检测车道线边缘。

ROI掩膜：只识别图像下半部分内容（近处），排除远方干扰。



1. 单线巡线：

- PID控制：通过比例（P）、积分（I）、微分（D）三环节动态调整小车转向。



其中e（k）表示小车当前时刻的位置与目标位置的误差。

第一项为比例项，Kp为比例系数，若小车偏离目标路径的横向距离为 e（k），比例项会按偏离距离的Kp倍调整转向角度或电机转速。Kp过大可能导致震荡或超调；过小则响应缓慢。

第二项是积分项，若小车因地面摩擦始终无法达到目标速度，积分项会随时间累积误差，逐步增大输出补偿。可以消除长期偏差，提升稳态精度。

第三项微分项，可以防止当前位置在接近目标位置时又过于调整。Kd越大越不容易矫枉过正。

2. 双线巡线：

- 中线跟踪：根据双线位置计算中线轨迹，控制小车沿中线行驶；

- 状态机设计：将M型路径分解为直行、转弯和环岛处理等状态，通过条件判断切换控制逻辑。

实验步骤

1. 仿真环境配置：

- 将`robot\_task`功能包复制至ROS工作空间`src`目录，执行`catkin\_make`编译；

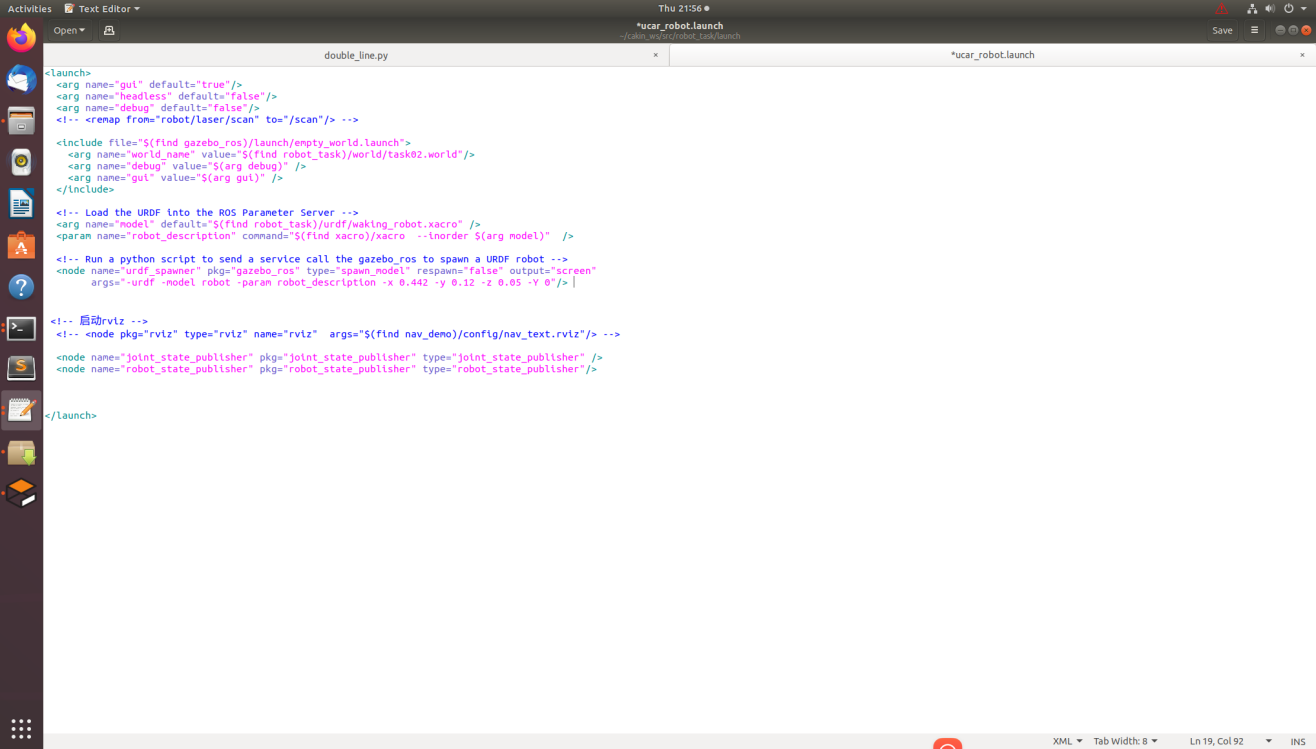
- 修改`ucar\_robot.launch`文件中的`world`参数，切换`task1.world`或`task2.world`。

1. 初始位置确定：

运行 ucar\_robot.launch文件后打开rviz对目标起点位置进行标定确定其出发位置坐标以及角度。

1. 单线巡线代码调用：

调用前需要调整小车的初始位置如下，使得小车摄像头可以首先看到要识别的单线轨迹



# 启动仿真环境

roslaunch robot\_task ucar\_robot.launch

# 运行单线巡线节点 进入到单线轨迹所在目录下运行

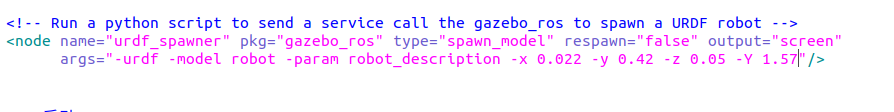
Python task01.py

- 关键代码逻辑：

4. 双线巡线代码调用：

# 切换至task2.world后运行双线巡线节点

双线循迹的初始位置如下：



运行指令如下

Python task02.py

实验反思

1. 小车运行轨迹不稳定，需要进一步进行实验验证调整参数，从而调整出最优路径。
2. 实验问题思考，小车如何将识别到的图像中的位置和地图中的目标位置进行转化
3. 注意在运行过程中，要退出conda环境

实验二：基于YOLOv5的目标检测模型训练与应用

## 实验目的

1. 掌握YOLOv5目标检测环境的配置方法；

2. 训练自定义数据集模型并实现实时检测；

3. 将检测结果通过终端输出。

## 实验内容

1. 在Ubuntu系统中配置YOLOv5依赖环境；

2. 使用标注工具对训练集进行标注，生成`dataset.yaml`；

3. 训练模型并测试摄像头实时检测效果。

## 实验器材

- Ubuntu 18.04系统

- USB摄像头/电脑自带摄像头

- YOLOv5开源代码库

- 标注数据集（COCO格式）

## 实验原理

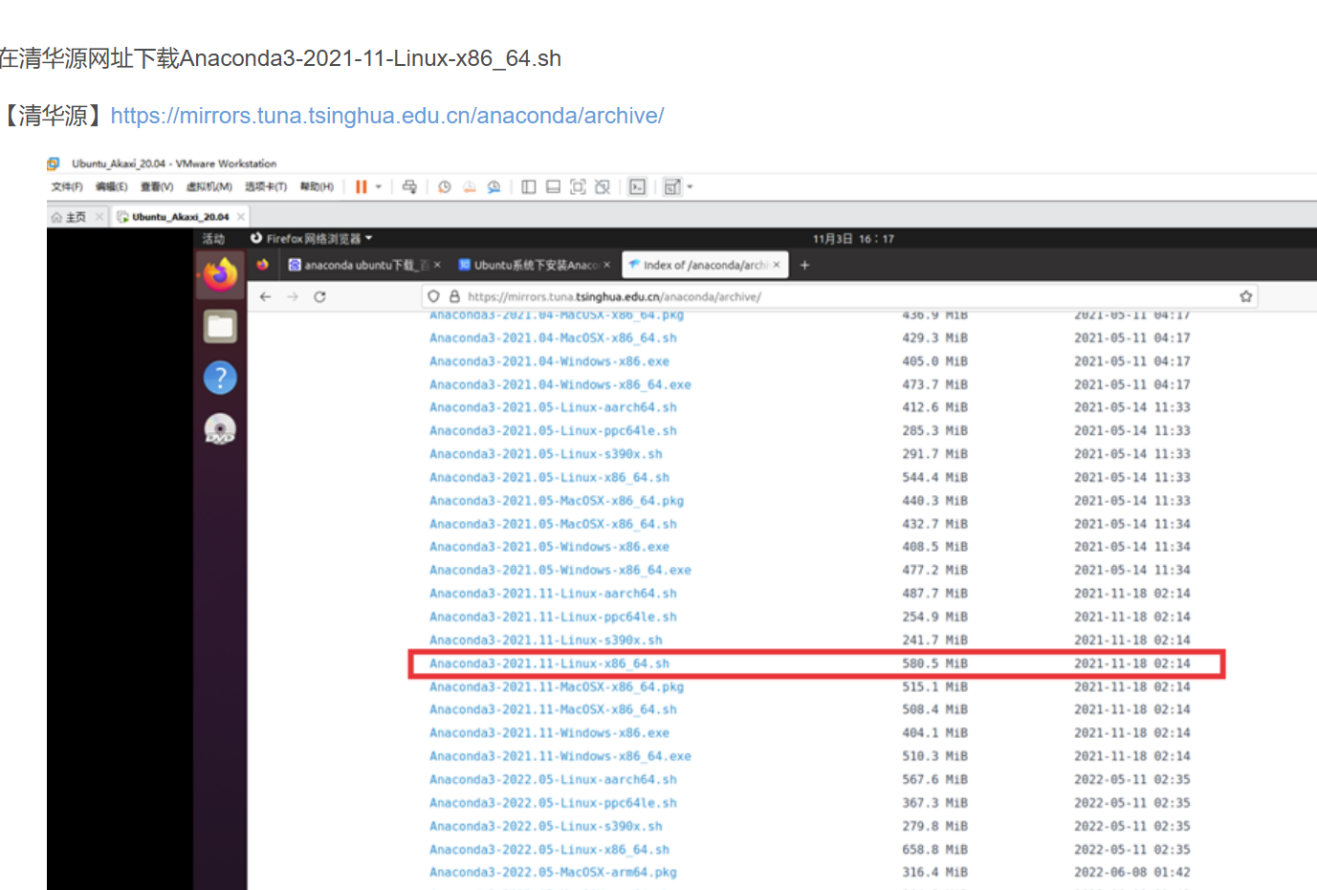
- YOLOv5算法：单阶段目标检测模型，通过锚框（Anchor）机制和特征金字塔网络（FPN）实现多尺度检测；

- 数据增强：采用翻转、裁剪等技术提升模型泛化能力。

## 实验步骤

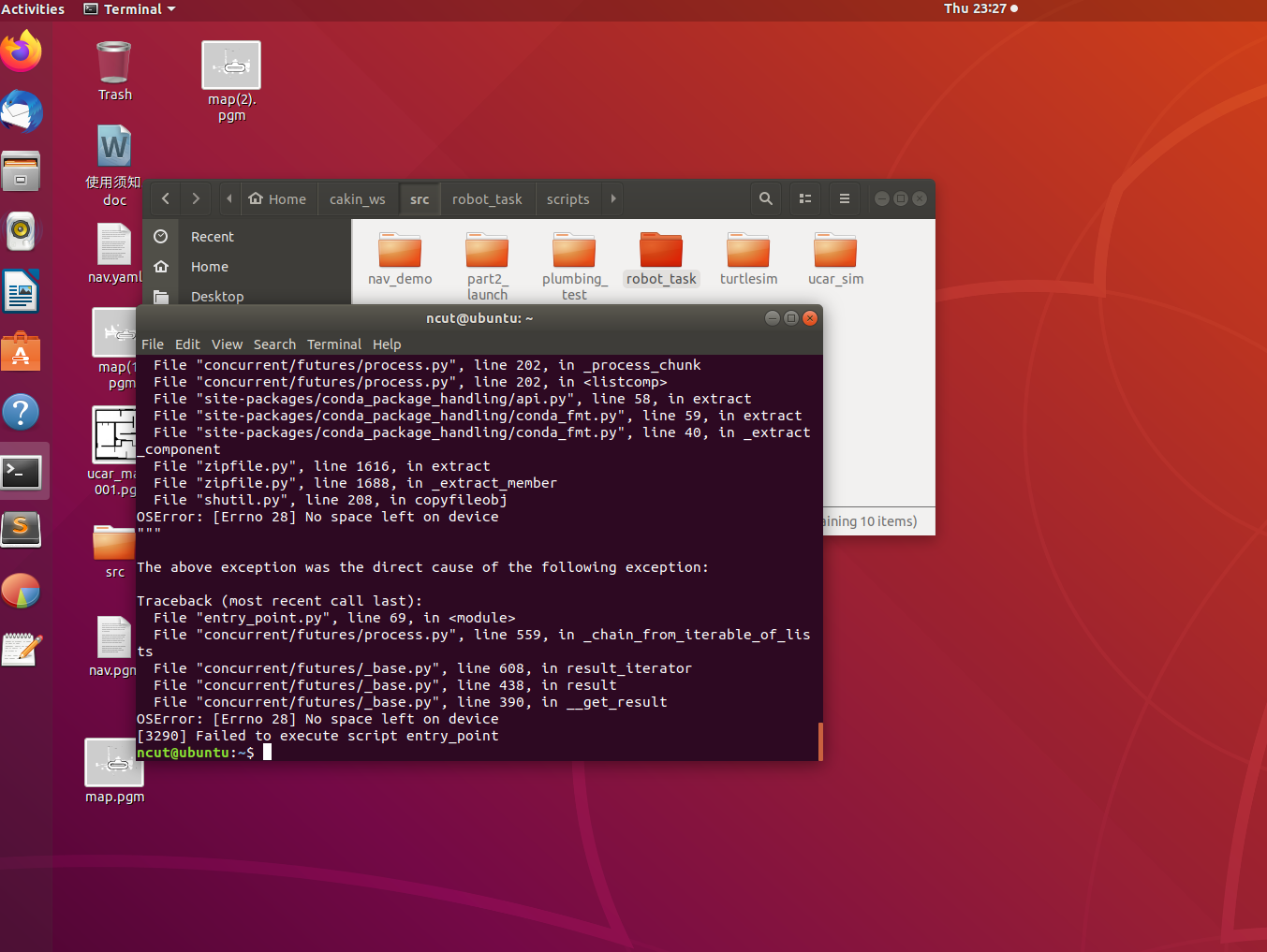
1. 环境配置：

* 首先下载anaconda

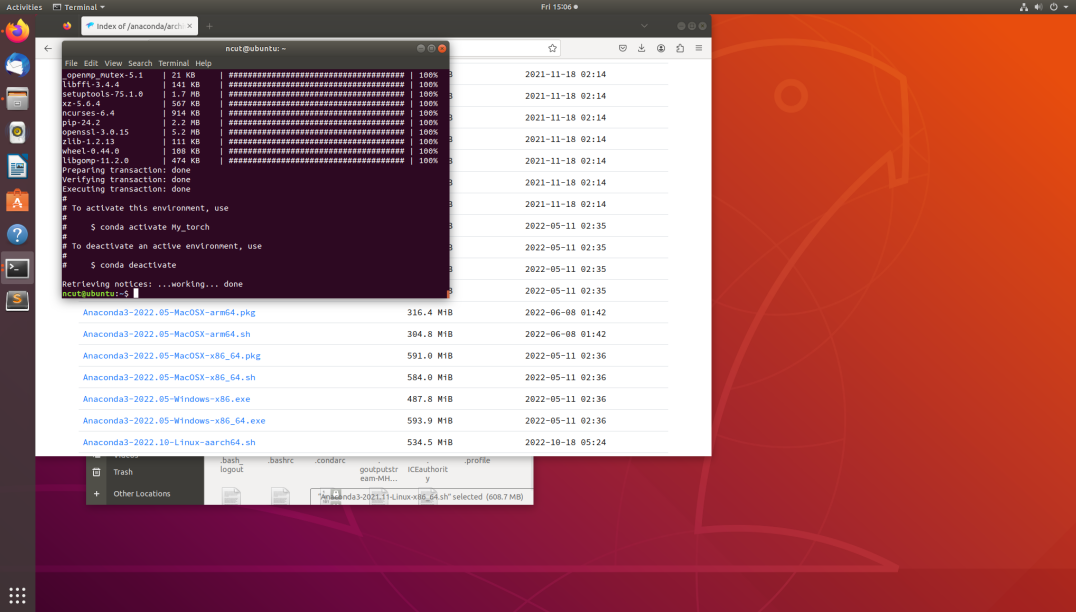


* 将下载安装好的包复制到主目录下，并打开终端输入如下命令进行安装

bash Anaconda3-2021.11-Linux-x86\_64.sh



此页面表示ubuntu内存不足，经过ubuntu内存扩容后，成功安装界面如下：



接下来进行环境配置：

vim ~/.bashrc

export PATH=/root/yes/condabin:$PATH（安装路径）

source ~/.bashrc

conda -V



添加清华源：

conda config --add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/'

conda config --add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/'

conda config --set show\_channel\_urls yes

配置conda环境：

conda clean -i

sudo gedit ~/.condarc



channels:

- defaults

show\_channel\_urls: true

default\_channels:

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2

custom\_channels:

conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

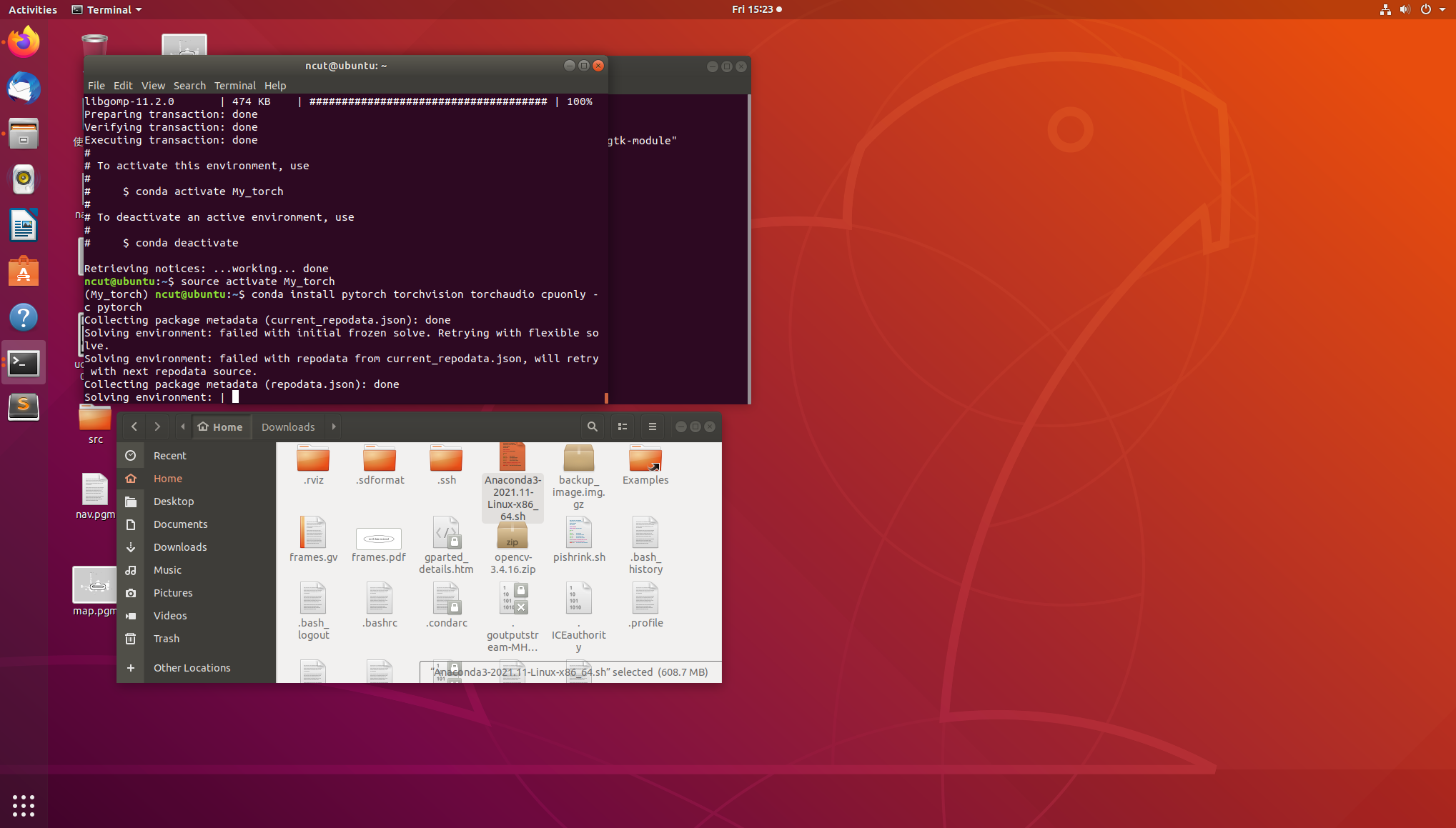
menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

pytorch-lts: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

simpleitk: <https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud>

然后创建虚拟环境conda create -n My\_torch python=3.8

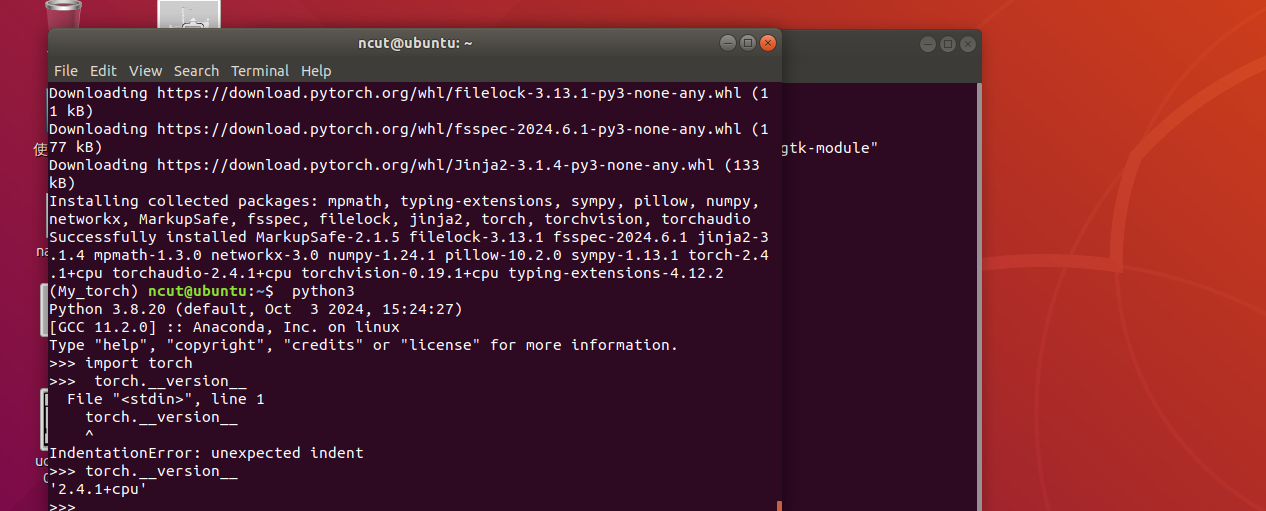


接着需要进入虚拟环境后安装pytorch

<https://pytorch.org/>

在官网选择合适的指令，复制到终端进行安装。

安装好的界面如下：

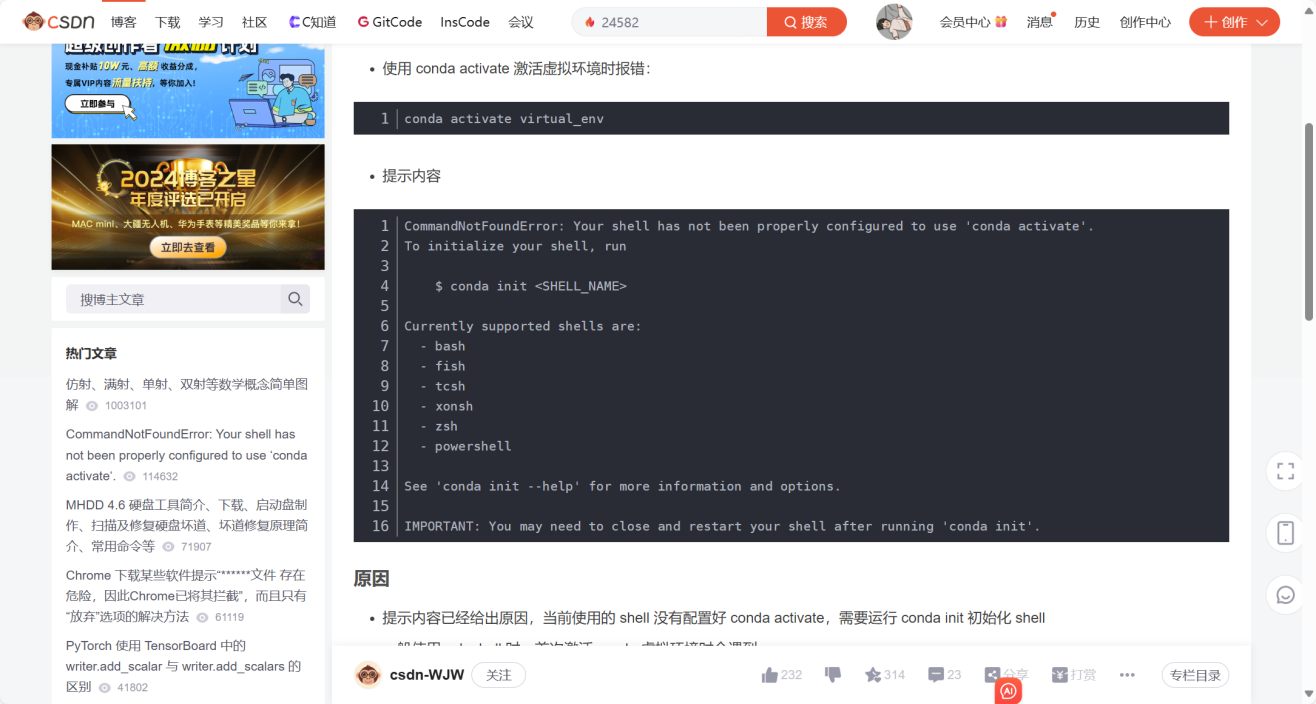


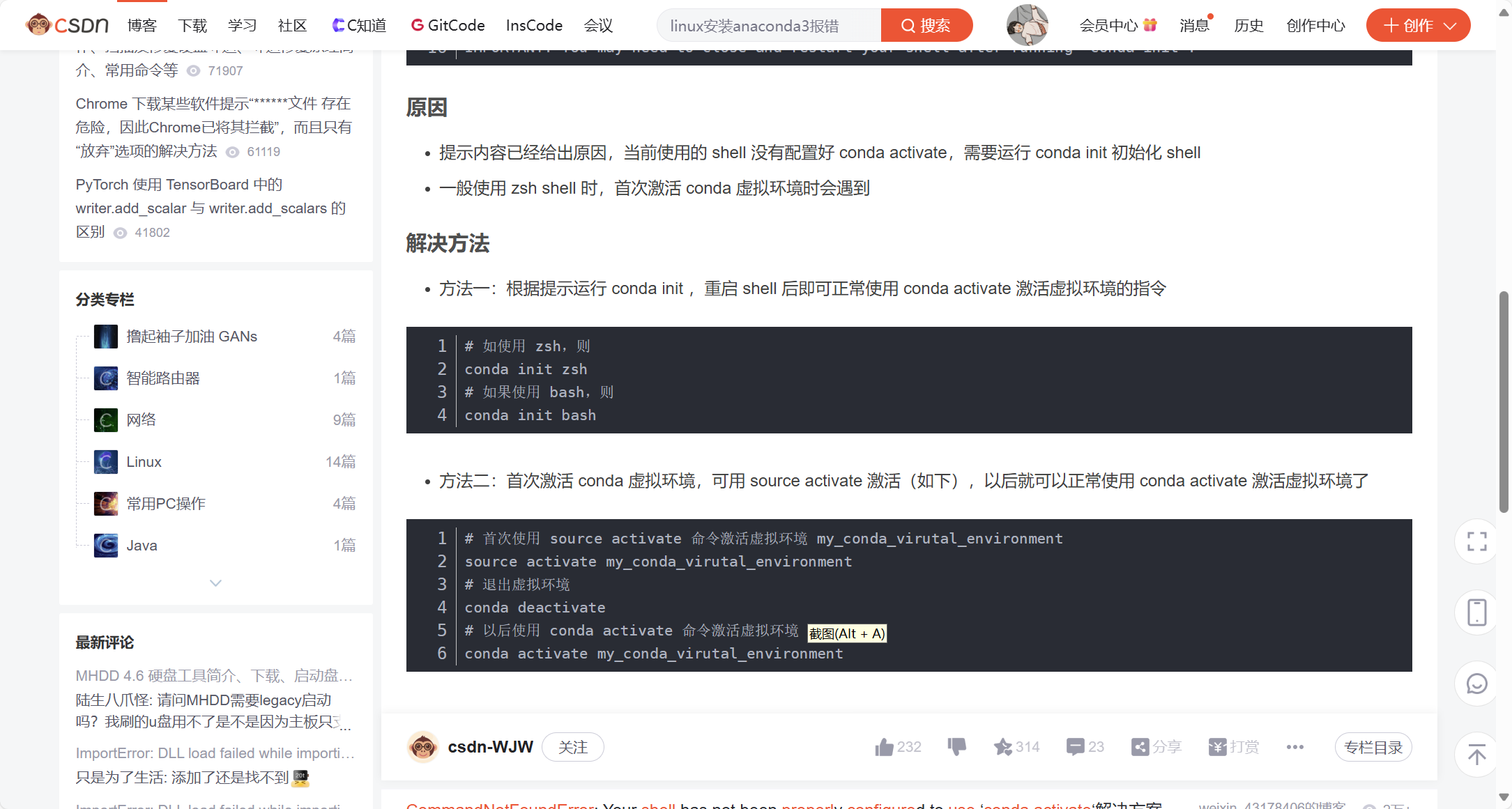
<https://search.gitee.com/?skin=rec&type=repository&q=yolov5>

在gitee上寻找一个yolov5包进行下载解压，要在进入刚刚创建的环境之后在进入解压后的YOLO文件夹目录，然后安装相关依赖

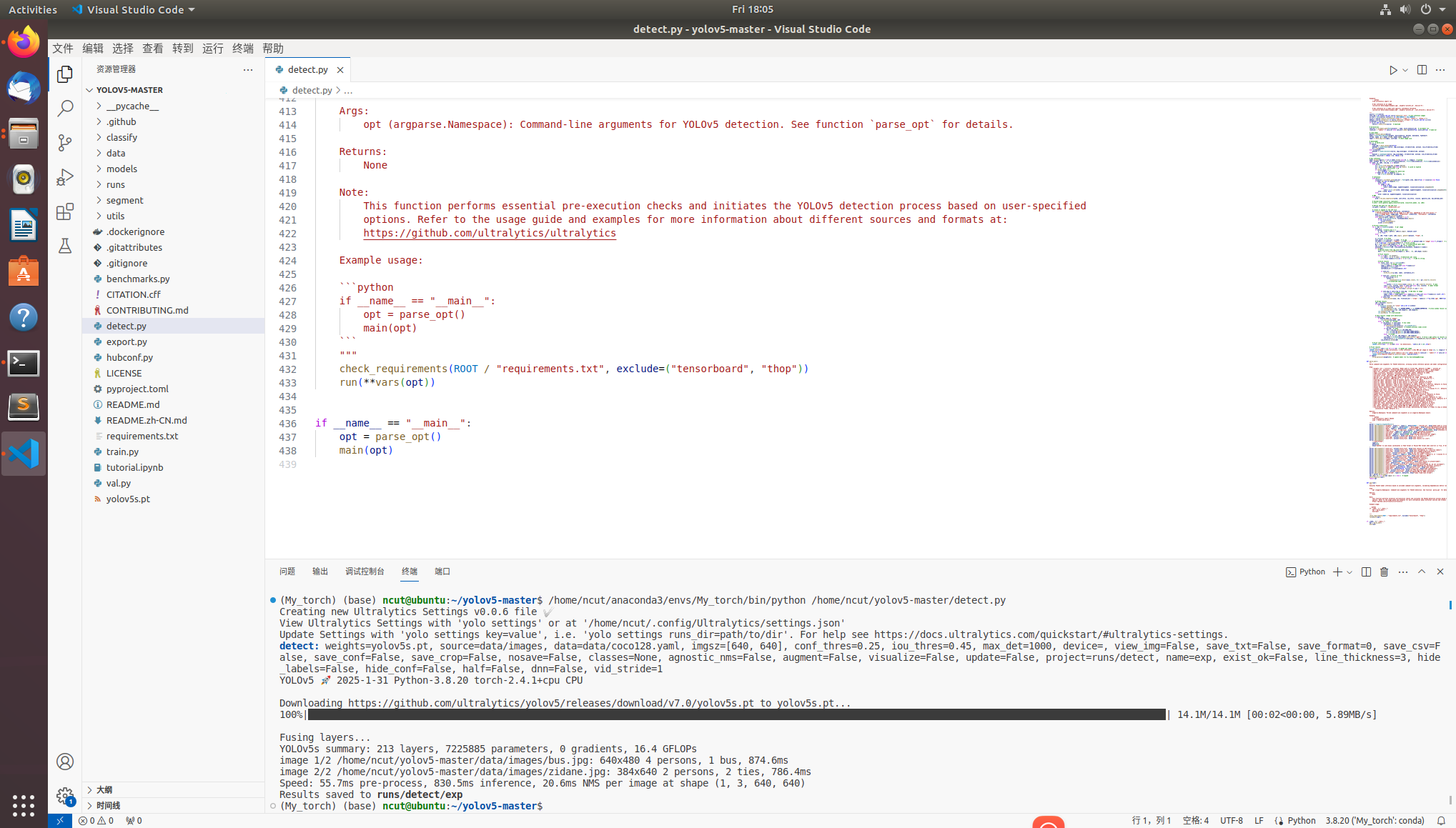
pip install -U -r requirements.txt -i <https://pypi.tuna.tsinghua.edu.cn/simple>

PS：如果出现进入虚拟环境失败的情况，出现如下报错解决情况如下:





最后进入pytorch所在环境中运行yolov5包中的detect.py文件得到如下运行结果表示安装成功：



下面是使用 YOLOv5 训练自定义数据集的完整制备流程，从数据收集到最终验证的详细步骤：

一、数据集目录结构

YOLOv5 要求数据集按以下结构组织：

mydataset/ # 数据集根目录

├── images/ # 存放所有图片

│ ├── train/ # 训练集图片（如：image1.jpg, image2.jpg...）

│ └── val/ # 验证集图片

└── labels/ # 存放所有标签

├── train/ # 训练集标签（与图片同名，扩展名为.txt）

└── val/ # 验证集标签

```

没有按照标签对训练集和测试集进行一一对应的划分是本次训练任务失败的主要原因，本人只的images和labels文件下没有train和val的文件夹

关键要求：

1. 文件名严格对应

- 图片 `images/train/image001.jpg` → 标签 `labels/train/image001.txt`

- 文件名需完全一致（包括大小写），扩展名无关（`.jpg`、`.png` 均可）。

2. 标签路径自动关联

YOLOv5 会根据 `images/train/` 路径自动查找 `labels/train/` 下的标签文件。

二、数据标注

1. 标注工具

- 使用labelimg进行标注，导出YOLO格式（需在保存时选择YOLO模式）。

pip install labelimg

labelimg # 启动后选择YOLO格式

2. 标签格式规范

每个 `.txt` 文件对应一张图片，格式如下：

<class\_id> <x\_center> <y\_center> <width> <height>

所有值需除以图片的宽度和高度，得到 `0~1` 之间的浮点数。

- \*\*示例\*\*：

若图片尺寸为 `640x480`，某物体边界框的左上角为 `(100,200)`，宽高为 `(320,240)`，则计算方式：

```bash

x\_center = (100 + 320/2) / 640 = 0.40625

y\_center = (200 + 240/2) / 480 = 0.5833

width = 320 / 640 = 0.5

height = 240 / 480 = 0.5 最终标签行：`0 0.40625 0.5833 0.5 0.5`

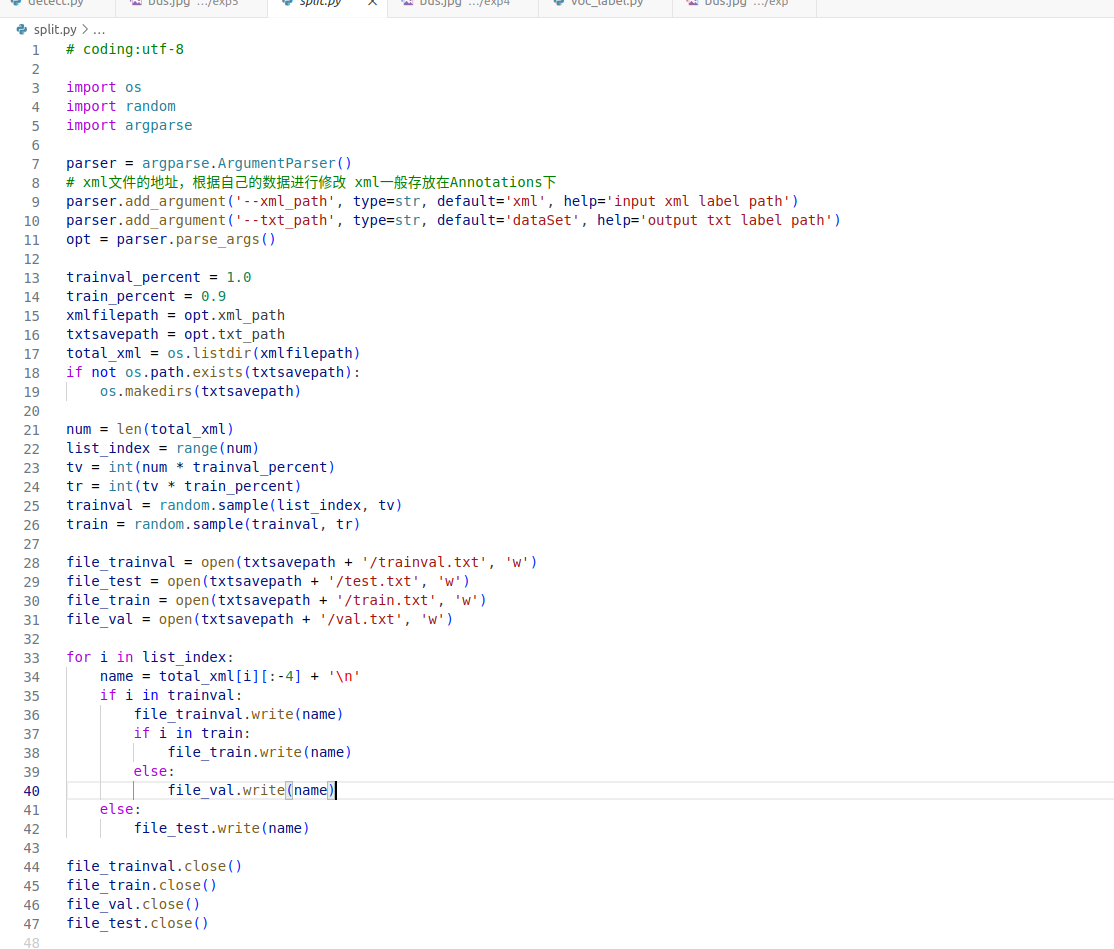
3. 标签文件示例

0 0.35 0.48 0.12 0.23

1 0.62 0.71 0.08 0.15

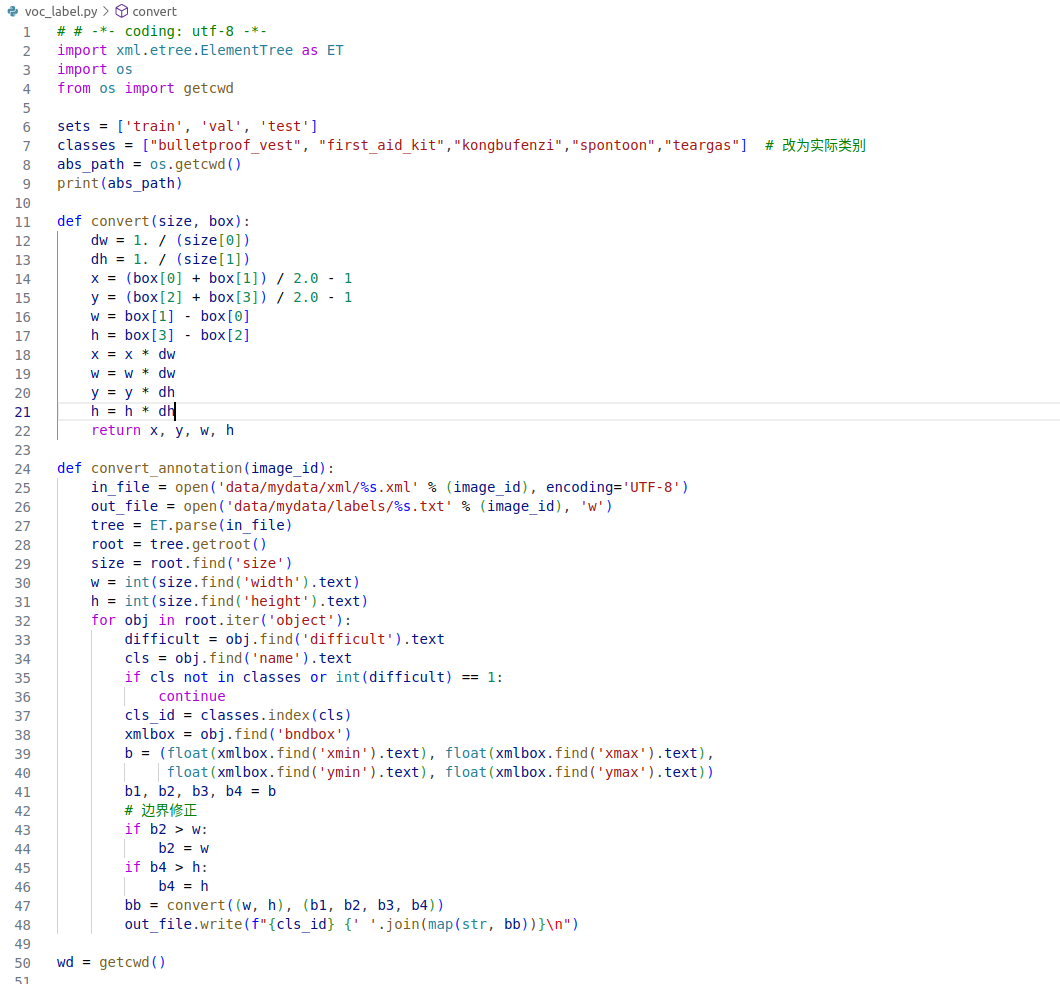
三、数据集划分

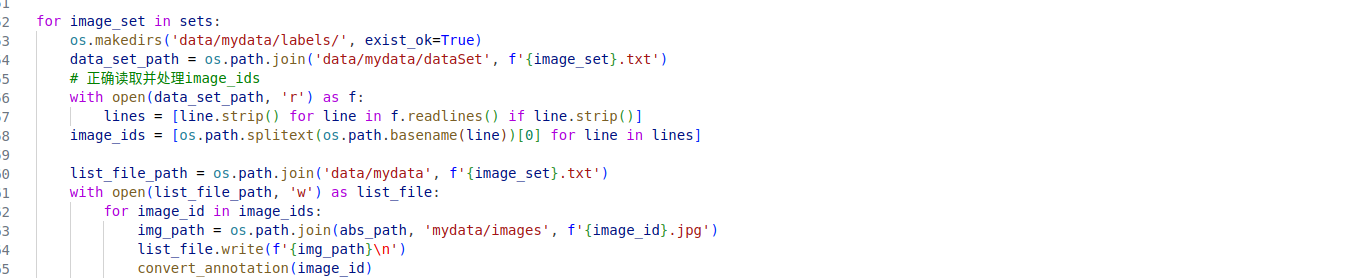
利用split.py对数据集进行划分



1. 创建数据集配置文件

创建voc\_label.py文件，将训练集、验证集、测试集生成****label标签（训练中要用到）****，同时将数据集路径导入txt文件中，代码内容如下：





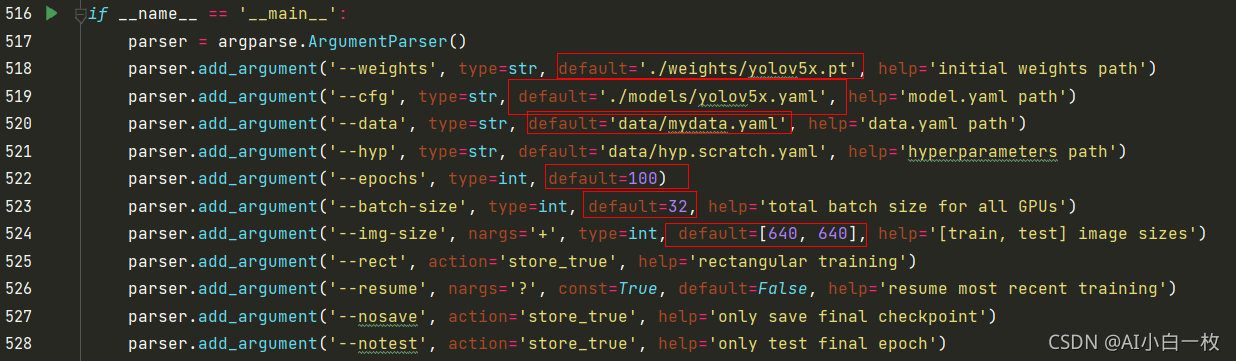
在 YO在yolov5目录下的data文件夹下新建一个mydata.yaml文件（可以自定义命名），用来存放训练集和验证集的划分文件（train.txt和val.txt），这两个文件是通过运行voc\_label.py代码生成的，然后是目标的类别数目和具体类别列表，mydata.yaml内容如下：

- 使用绝对路径避免歧义（如 `/home/user/mydataset/...`）。

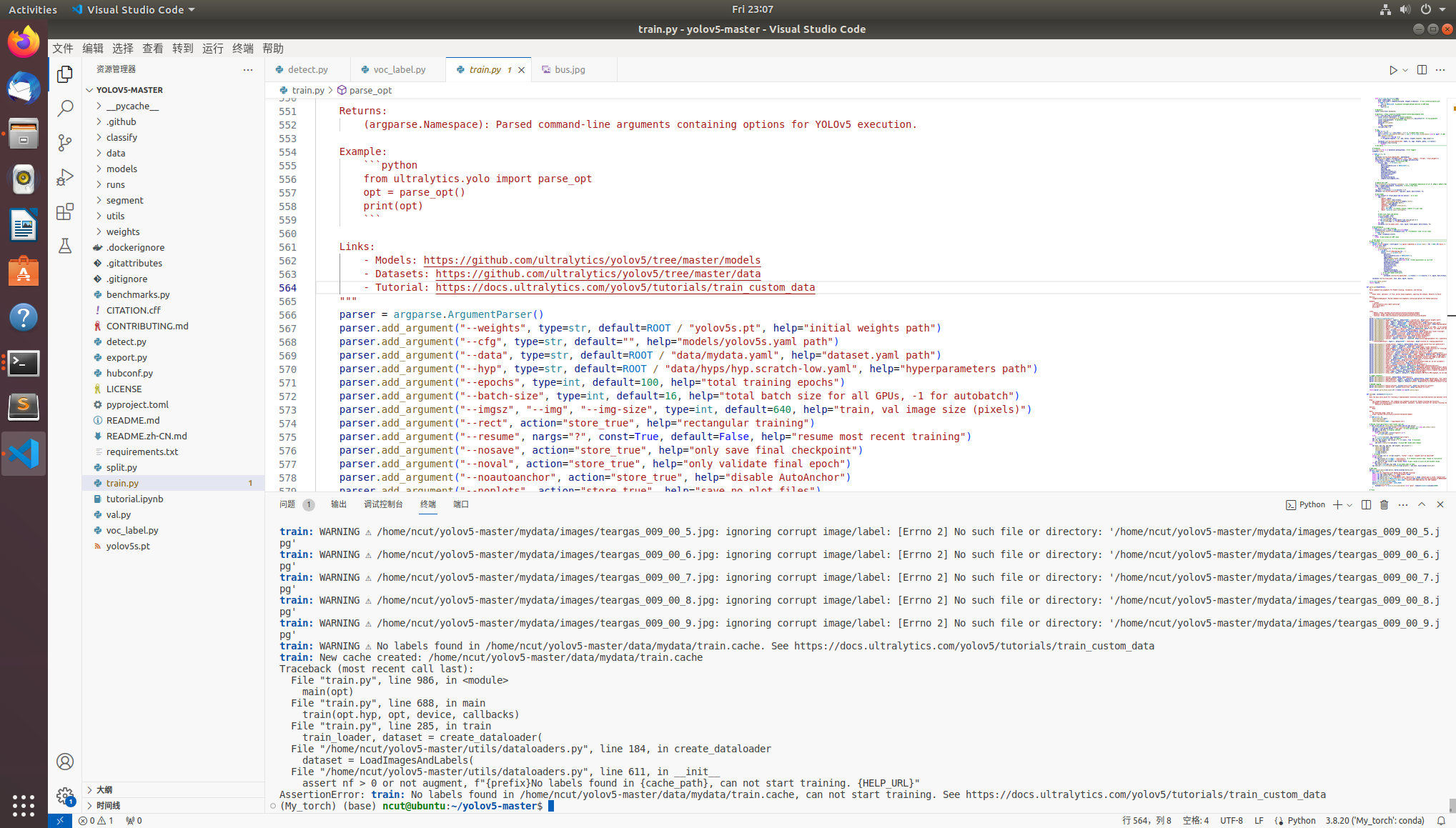
- 类别名称顺序必须与标注时的 `class\_id` 一致。

1. 开始训练

将下载的yolo模型中的，train.py的参数进行修改之后开始训练



python train.py --img 640 --batch 32 --epoch 300 --data data/mydata.yaml --cfg models/yolov5x.yaml --weights weights/yolov5x.pt --device 'cpu' （根据自己的设备选择是cpu还是gpu’0,1’）



由于前面对于训练集数据集没有进行正确划分，所以在训练时出现如上报错。