北京交通大学

《基于 yolov5 的图像识别训练实战》
综合报告

专　　业：计算机科学与技术

班　　级：计科 2101

学生姓名：郭翔裕　刘茹龙　依木兰

学　　号：21221168　19221193　19281114

北京交通大学计算机与信息技术学院

2023 年　8 月　23 日

目录

一、项目简要介绍

本项目是基于 Yolov5 项目实战项目，共完成口罩识别、车牌识别、安全帽识别、使用 kaggle 云平台的各类物体识别共四个模型的训练。

每一个模型都主要分为 4 步骤：**环境配置-数据集制作准备-训练模型-检验成果**

环境配置是任务的前提。主要使用的是 python 原生环境而非 ananconda 管理使用的环境，且其中也找到了配置技巧在报告内有阐述。

数据集制作准备是任务的基础。主要使用技术点是寻找爬取数据、Linux 下进行的数据图像处理、利用 labelimg 打标签、数据文件的组织；

训练模型是项目的保证。主要技术点是：预训练权重的准备、训练模型参数调整。

检验成果主要使用 detect.py 调参运行模型获得结果，除此之外使用 tensorboard 可观察训练过程的各项参数，进而尝试深入模型内部进行跟深入的训练参数调整。

二、成员工作及贡献比

组长：郭翔裕（21221168）

成员：刘茹龙（19221193） 依木兰（19281114）

郭翔裕工作内容：

1.配置可运行 yolov5 的编程环境

2.linux 系统使用以及图像爬取、处理

3.数据集的采集与制作、调参、运行四个模型的测试并完成模型预测结果导出

4.根据刘茹龙同学制作的 PPT，进行扩充、改正

5.小组展示视频录制

6.报告目录框架编写，报告扩充调整以及排版

7.编写 1-3 步骤《技术文档》（见附件）

8.编写 yolov5 各代码文件功能概述（见附件）

刘茹龙工作内容：

1.配置环境探索

2.学习 linux 系统使用以及图像爬取、处理

3.学习数据集的制作与划分、学习训练模型

4.PPT 制作

5.小组展示视频修改

6.报告编写

依木兰工作内容：

1.配置环境探索

2.学习 linux 系统使用以及图像处理

3.学习数据集的制作与划分、学习训练模型

4.数据搜集与处理

5.资源搜集

6.词汇释义查找与批注

7.报告编写

贡献比：

郭翔裕：刘茹龙：依木兰 = 5：3：2

三、为项目完成所做的工作

3.1Linux 命令学习

终端操作是跑深度学习的一项基本技能，龙老师在课堂上多次强调，因此需要完成以下常用命令的学习

vim ----------命令行编辑文件，修改保存文件的工具

cat ----------命令行打印文件内容

cd ----------抵达目录

mkdir----------创建文件夹（目录）

python3 ----------使用 python 作为.py 文件解释器

ls ----------进入目录自然的习惯，展示当前路径下内容

chmod----------更改文件的读写权限

ll -s----------更加详细地展示文件列表

pwd ----------展示当前路径

sh ----------运行.sh 的脚本文件，可以**传参数※**

当然，更多的操作命令如 pip install、mv 等等不一一列举，经过本次课程已经熟练掌握。《技术文档》内部有详细的内容可以参考。

3.2Linux 系统下进行 图像/视频 数据获取和数据增强

数据获取有以下操作：

1. 用 Image-Downloader/DownKyi 爬取百度图片或者哔哩哔哩数据
2. Kaggle 数据平台数据获取

数据增强有以下操作：

1. 改变图片大小
2. 图片裁剪
3. 转灰度图
4. 图像镜像、中心、轴对称变换

通过以上手段，郭翔裕复现对图像数据集的收集。此外有关视频的操作见《技术文档》。

3.3 配置 yolov5 的深度学习环境

配置深度学习环境的初始，我们对于虚拟环境是什么，为什么要使用虚拟环境所知甚少。于是做的事情是跟着网络教程进行执行。但遗憾的是网络教程上的执行步骤首先多而繁杂，python 的版本在官网上有数十个版本共我选择，令人眼花缭乱；anaconda python 版本各有不同，pytorch 中 torch 和 torchvision 的版本也有匹配问题，除此之外显卡的型号各有不同，加之仅仅支持英伟达的部分显卡，AMD 的显卡不支持等等问题，同时还有视频教程的时效性，各种版本的 yolov5 运行环境也有所差异...因此，我们的该项目的深度学习环境配置并不顺利，往往是跟着博主的操作，但是命令行显示却大相径庭，甚至各种 Warning 和 Error 发生.....

配置环境真实历程（郭翔裕）：

我曾经在 windows 上尝试安装失败后，此时我对环境的配置已经是一头雾水，甚至不知道我的 Windows 电脑上装载了到底多少个版本的 python，多少个版本的 Anaconda，于是我对本机的环境已经放弃，转到 Linux 虚拟机上进行包的安装调试。在 Linux 上我学习了许许多多命令后，我使用命令一条一条地安装软件包，我观看龙老师上课回放，一条命令一条命令地安装龙老师所讲的 python 解释器、opencv、pillow 等软件包，并使用 vim 进行复写所有图像处理的代码，这些都水到渠成。Linux 给我的感受就是命令层次分明，一条命令代表的含义也很清晰明确，可以查看自己操作流程。

在龙老师教学的基础上，我尝试跟着去配置 Linux 下的 yolov5 深度学习环境，我一步一步地跟着 pip install，但是仍然时有发生 Error。我发现是网络的问题。
于是我打算在虚拟机上安装一个 Clash for Linux 工具，很遗憾，我遇到的这个问题依然没有解决所谓的 Url 地址更改读取成功。最后我经过多次 pip install 安装尝试，长时间的等待，终于把符合要求的软件包安装完成了。随后，我开始在 Linux 下使用 pycharm，我尝试去打开 yolov5，并配置已经安装符合“官网”要求的环境，但是当我再次创建的环境，查看解释器后，我发现 pycharm 居然都无法识别我在终端中 pip install 的软件包！于是，我继续查阅问题解决方案，最后终于在一处博客的“回复”中看到这么一句话:“实在不行就创建环境选项勾选‘导入全局软件包’，解决这个破问题浪费了我一天”，终于成功导入所有软件包，我点开 detect.py 运行后，发现“downloading xxxxx”但是下载速度特别慢，多次报错失败，我一看问题，又是一个

不太明白模棱两可的报错，最后在我多次尝试运行 detect.py 成功了一次 downloading 后，发现其实仍然是网络问题。至此，我在 Linux 下调试环境的工作到此结束。

我把目光投向了可以挂 Vpn 的 Windows 主机上。然而由于我的不断安装各种版本的 anaconda、python 等，在这个杂乱的情况下我选择重装操作系统。

在这个期间，我将所有的材料制作成《技术文档》予以保留。

Windows 上与 Linux 最大的不同就是图形化界面与终端这一工具的使用。我想应当在什么系统下主要使用对应系统优势工具才是最最合适的。对于 Windows 应使用图形化界面才是最好的选择。于是我把目光聚焦在 pycharm 工具的使用上。一个偶然的学习机会，我遇到了一个 pip 换源的教程，可以自动将所需软件包以国内源来安装。于是我成功地应用了它。打开 Vpn，在 pycharm 上使用虚拟环境，一键安装 requirements.txt 内容，一键运行 detect.py，终于很快成功了！

至此，深度学习环境配置告一段落。

3.3.1 库/包的安装问题总结

安装一些软件包的时候如 pytorch、pillow、opencv-python 等等报错，大概率原因是网络问题，解决方案是 pip 源换为国内源，以下是《技术文档》中解决方案如下：

1. 使用 pycharm
2. Win+R 后，搜索 %appdata%进入配置文件夹
3. 创建 pip 文件夹，内部新建一记事本，写入

```
[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
```

修改后缀为 ini

4. 这样，在 pycharm 中安装软件包（torch、pillow...etc...）时候优先从清华源 https://pypi.tuna.tsinghua.edu.cn/simple 内下载，而不是国外的 pip 源，这样可以解决大部分问题。

3.3.2 使用 pycharm 自动安装所有需要的软件包并成功配置到环境中

在 pycharm 解释器设置处，一定要选择使用虚拟环境！虚拟环境就像一扇门，为我们不同的项目提供不一样的配置环境。因此使用虚拟环境是十分必要的。

在换源之后，本步骤可以一键安装要求即可。

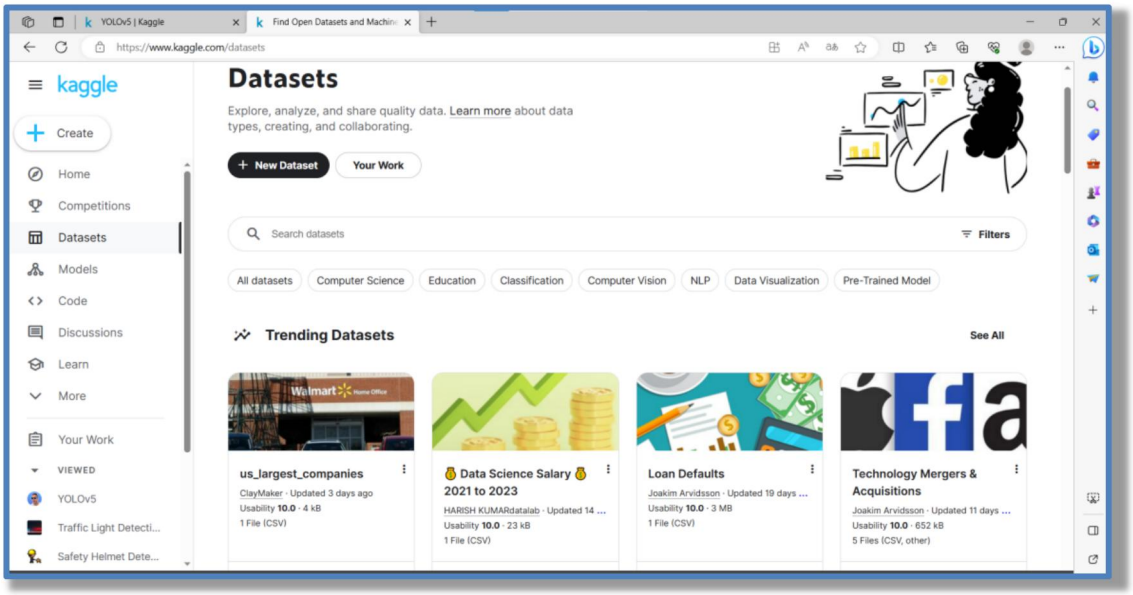
3.4 数据集准备

本项目中数据集收集采用下三种方式：

1. Windows 下使用 DownKyi 爬取哔哩哔哩数据

2. Image-Downloader 的使用：cd 到文件夹内，打开终端，输入 python3 image_downloader_gui.py 即可弹出 UI 界面进行自助爬取

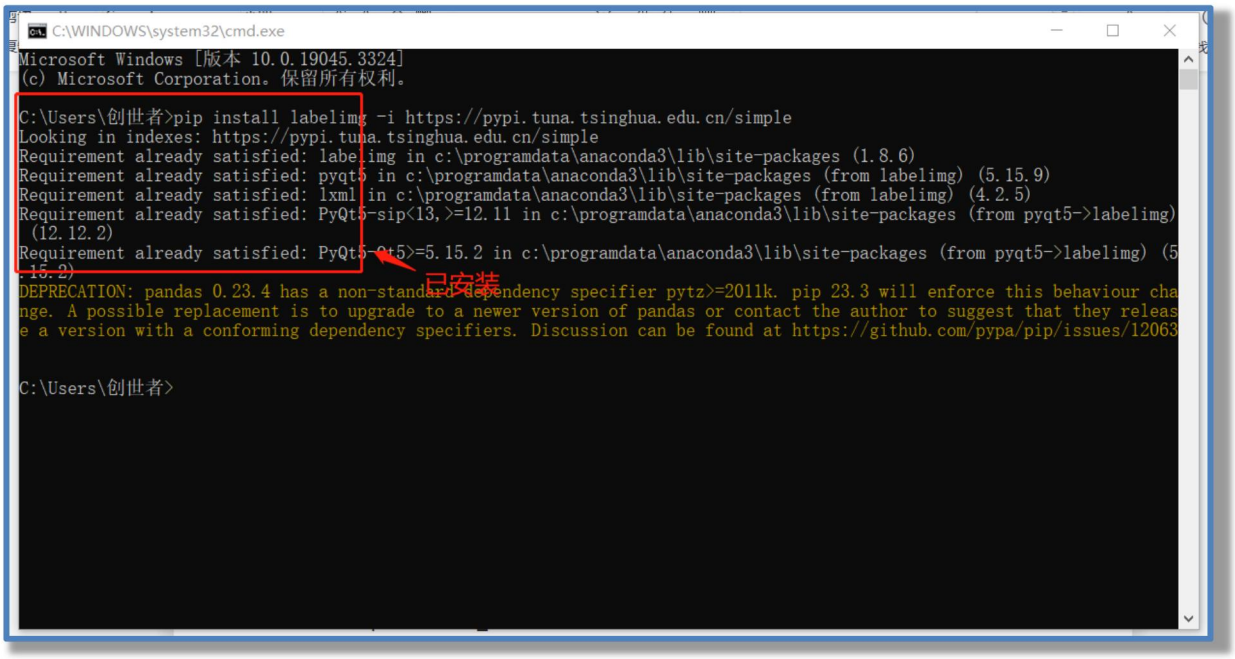
3. 也可使用 kaggle 平台获得图片数据或者是已标注好的数据，但是有的是 csv 格式，需要辨析格式采取使用。



3.4.1 利用 labelimg 打标签

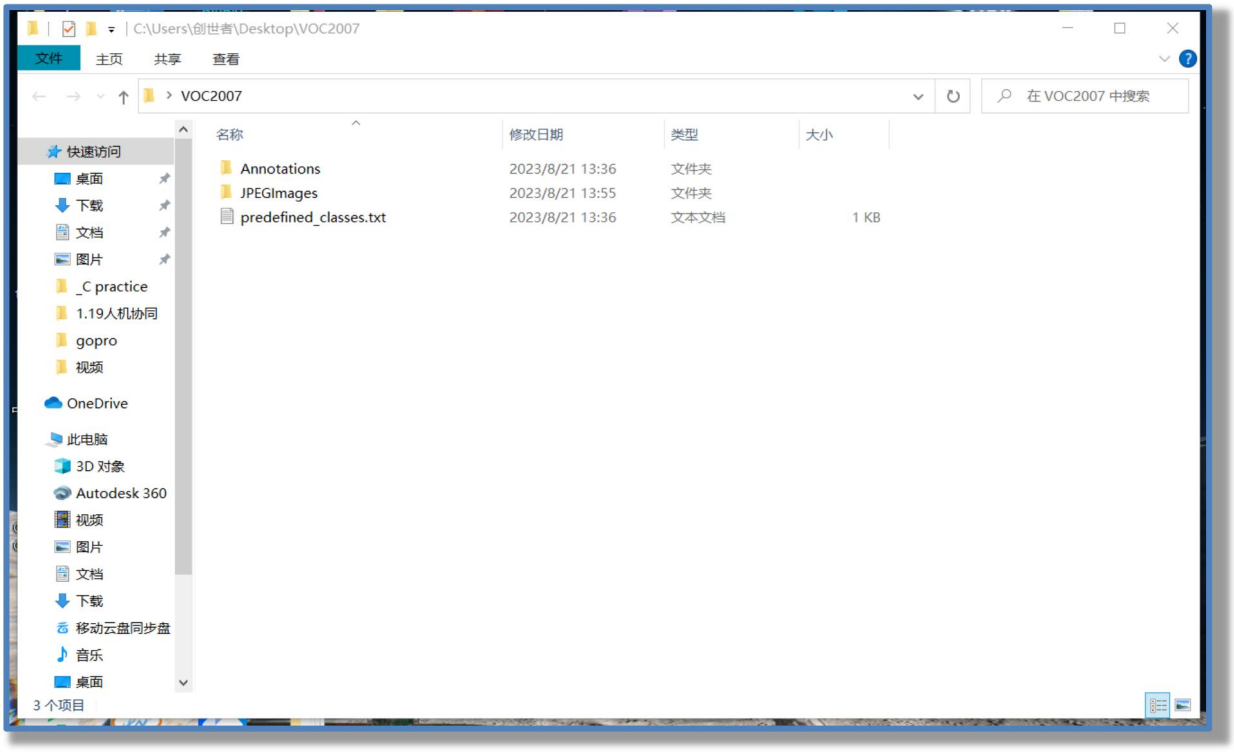
（1）安装

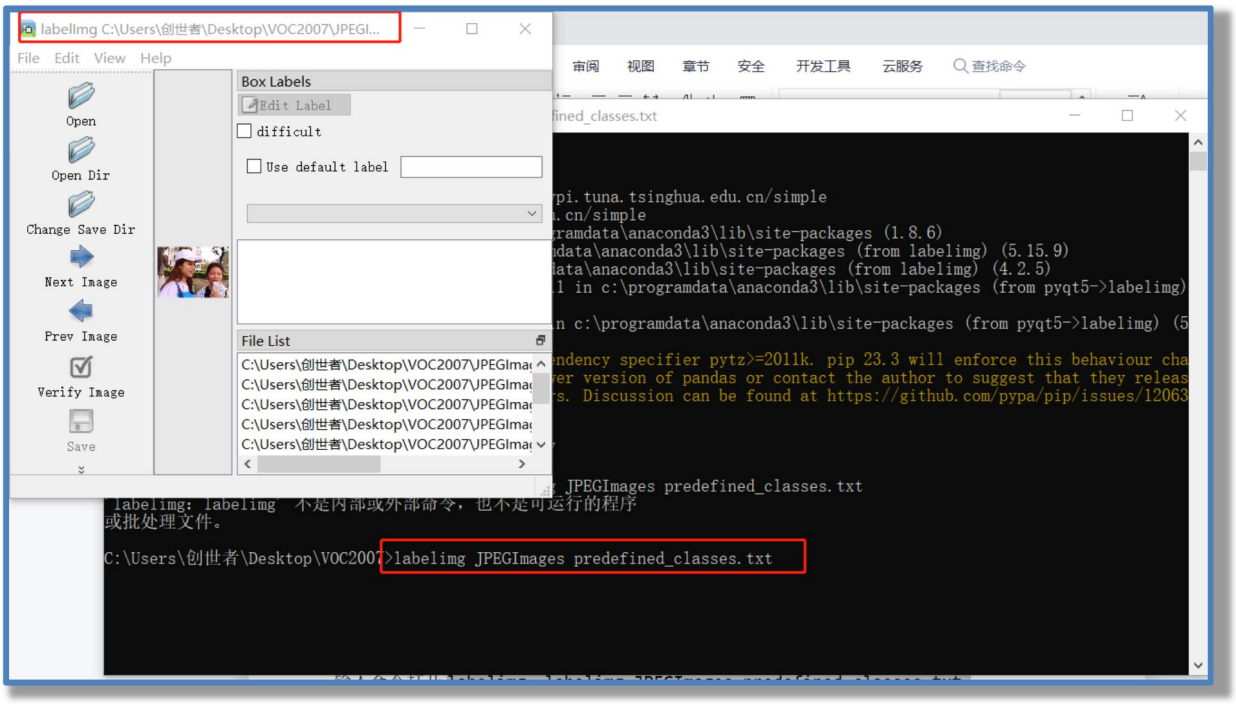
打开 cmd 命令行（快捷键：win+R）。进入 cmd 命令行控制台。输入命令：pip install labelimg -i https://pypi.tuna.tsinghua.edu.cn/simple



（2）数据准备

新建一个名为 VOC2007 的文件夹，里面创建一个名为 JPEGImages 的文件夹存放需要打标签的图片文件；再创建一个名为 Annotations 存放标注的标签文件；最后创建一个名为 predefined_classes.txt 的 txt 文件来存放所要标注的类别名称



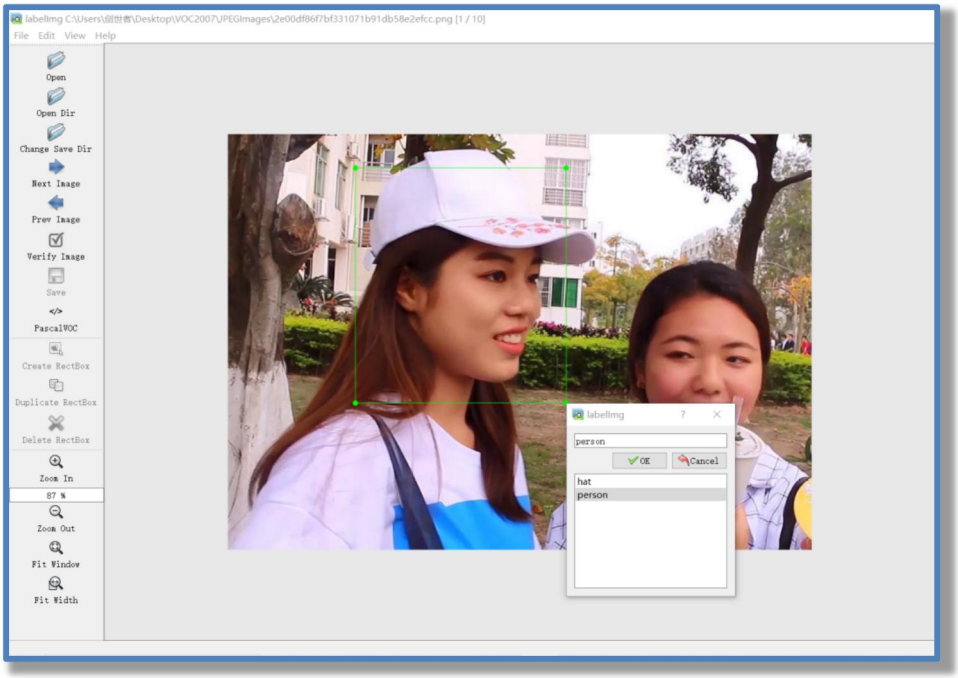


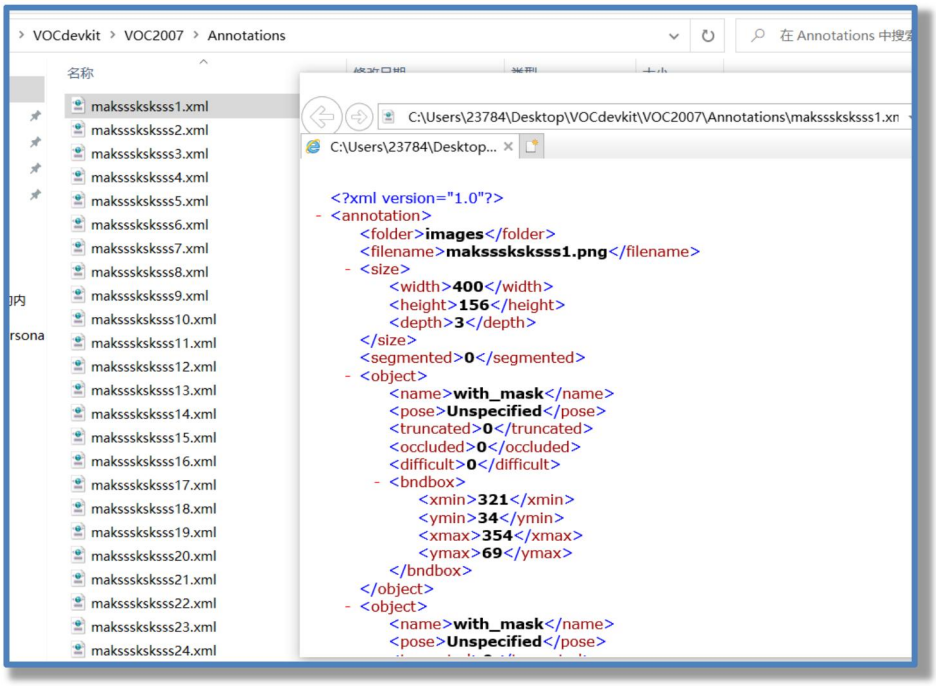
（3）前期准备

打开 cmd 命令终端（快捷键：win+R）。进入到刚刚创建的这个 VOC2007 路径，输入命令打开 labelimg：labelimg JPEGImages predefined_classes.txt

（4）标注

手动进行标注，标签打完以后可以去 Annotations 文件下看到标签文件已经保存在这个目录下

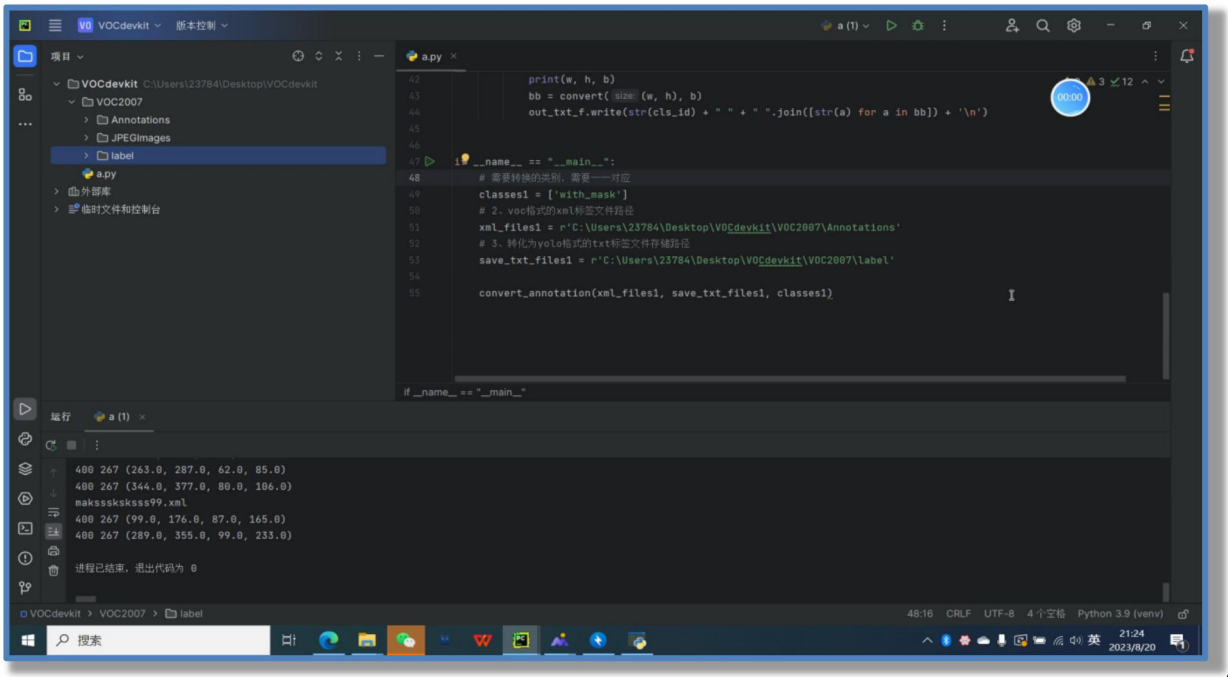




3.4.2 制作自己的可用于 yolov5 的目标检测数据集

3.4.2.1 将 xml 文件按照一定的文件组织格式转 yolo 的 txt 格式

使用 yolo 官方提供的脚本，将 xml 文件按照一定的文件组织格式转 yolo 的 txt 格式。



3.5 训练准备

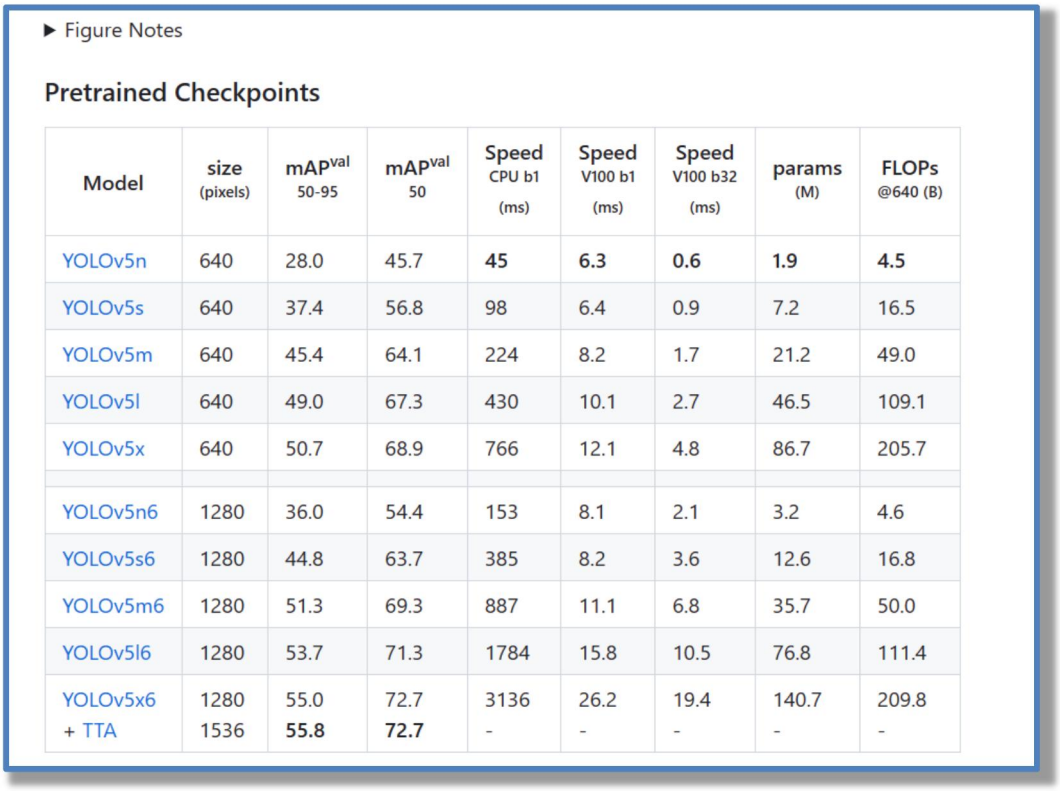
3.5.1 划分数据图片、对应的标注文件为 train（训练集）和 val（验证集）

可以将干刚刚转 yolo 格式的标注文件按照 80：20 的比例分为训练集和验证集，手动拖动相应数量的文件放入新建的目录中即可完成。
Yolov5 训练所用的数据集目录安排如下：

```
└──helmet
    ├──images
    │   ├──train
    │   └──val
    └──labels
        ├──train
        └──val
```

3.5.2 下载 yolov5 的预训练权重文件（选择 xxxxn.pt）

n、s、m、l 相当于尺码，根据实际数据量大小进行调整，数据量小就用相对较小的预选连权重，这样可以在一定训练 epochs 范围内提高训练效率。



▶ Figure Notes

Pretrained Checkpoints

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

3.5.3 训练模型参数调整

1.配置文件（yaml）修改：

训练目标检测模型需要修改两个个 yaml 文件中的参数。一个是 data 目录下的相应的 yaml 文件，一个是 model 目录文件下的相应的 yaml 文件。

两文件内容如图所示：

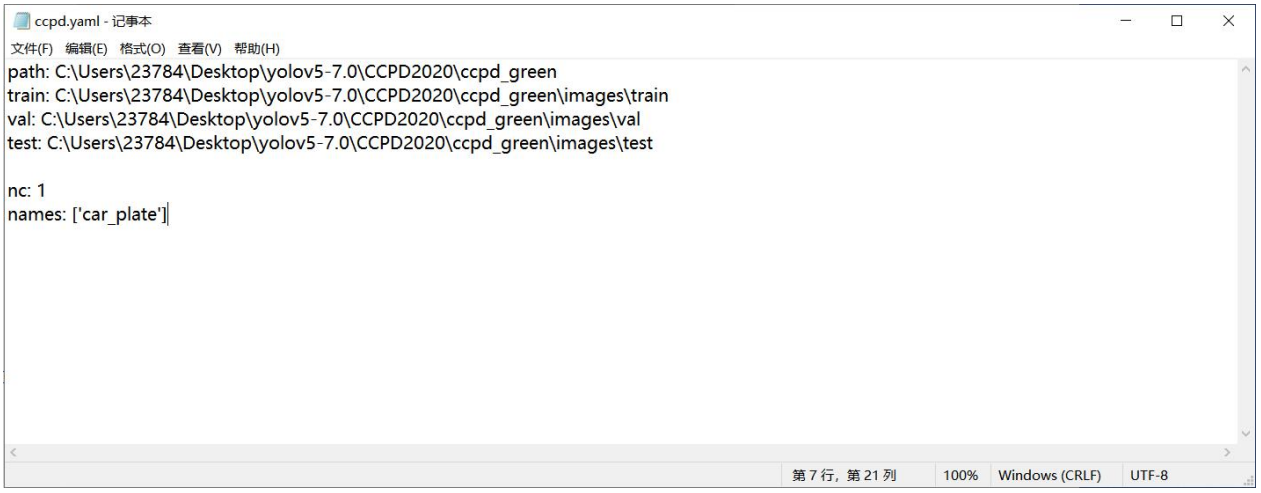


图 车牌识别项目 data/ccpd.yaml

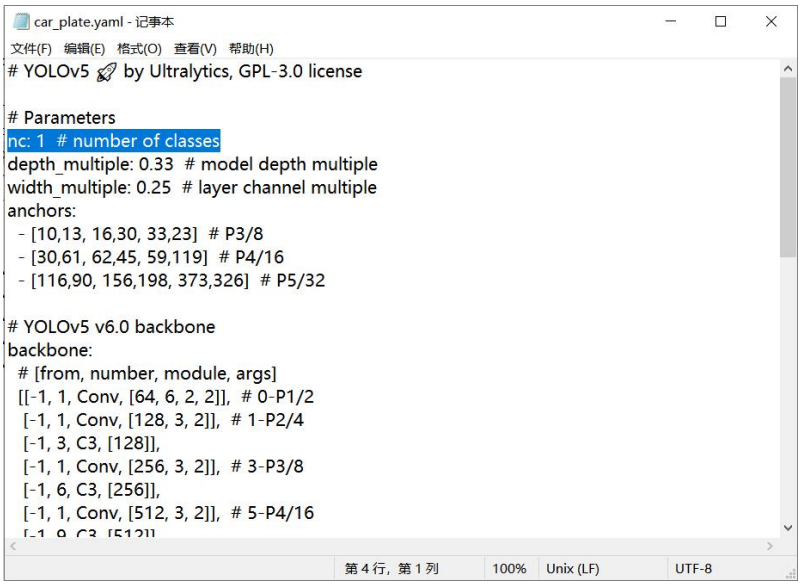


图 车牌识别项目 modles/car_plate.yaml（只需要将 yolov5n.yaml 的 nc 修改为项目识别的类别数 1）

2. train.py 文件修改：

主要修改 5 个参数：weights cfg data epochs batch-size

```python
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument( *name_or_flags: '--weights', type=str, default=ROOT / 'C:/Users/23784/Desktop/yolov5-7.0/weights/yolov5n .pt', help='initial weights path')
    parser.add_argument( *name_or_flags: '--cfg', type=str, default='', help='C:/Users/23784/Desktop/yolov5-7.0/models/car_plate.yaml')
    parser.add_argument( *name_or_flags: '--data', type=str, default=ROOT / 'C:/Users/23784/Desktop/yolov5-7.0/data/ccpd.yaml', help='dataset.yaml path')
    parser.add_argument( *name_or_flags: '--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
    parser.add_argument( *name_or_flags: '--epochs', type=int, default=16, help='total training epochs')
    parser.add_argument( *name_or_flags: '--batch-size', type=int, default=8, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument( *name_or_flags: '--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument( *name_or_flags: '--rect', action='store_true', help='rectangular training')
    parser.add_argument( *name_or_flags: '--resume', nargs='?', const=True, default=False, help='resume most recent trining')
    parser.add_argument( *name_or_flags: '--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument( *name_or_flags: '--noval', action='store_true', help='only validate final epoch')
    parser.add_argument( *name_or_flags: '--noautoanchor', action='store_true', help='disable AutoAnchor')
    parser.add_argument( *name_or_flags: '--noplots', action='store_true', help='save no plot files')
    parser.add_argument( *name_or_flags: '--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
    parser.add_argument( *name_or_flags: '--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument( *name_or_flags: '--cache', type=str, nargs='?', const='ram', help='image --cache ram/disk')
    parser.add_argument( *name_or_flags: '--image-weights', action='store_true', help='use weighted image selection for training')
    parser.add_argument( *name_or_flags: '--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument( *name_or_flags: '--multi-scale', action='store_true', help='vary img-size +/- 50%%')
    parser.add_argument( *name_or_flags: '--single-cls', action='store_true', help='train multi-class data as single-class')
    parser.add_argument( *name_or_flags: '--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')
    parser.add_argument( *name_or_flags: '--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
```

--weights: 这是一个命令行参数，它用于指定模型的初始权重文件的路径。参数类型是字符串 (str)。如果没有提供这个参数，将使用默认值，即路径'C:/Users/23784/Desktop/yolov5-7.0/weights/yolov5n .pt'。

--cfg: 这个参数也是一个路径，用于指定模型配置文件的路径。同样是字符串类型。如果没有提供这个参数，将使用默认值 ''。

--data: 这个参数用于指定数据集的配置文件路径。同样是一个字符串，表示数据集的配置。如果没有提供这个参数，将使用默认值，即路径'C:/Users/23784/Desktop/yolov5-7.0/data/ccpd.yaml'。

--hyp: 这是一个路径，用于指定超参数配置文件的路径。同样是一个字符串，用来配置训练过程中的超参数。如果没有提供这个参数，将使用默认值，即路径'data/hyps/hyp.scratch-low.yaml'。

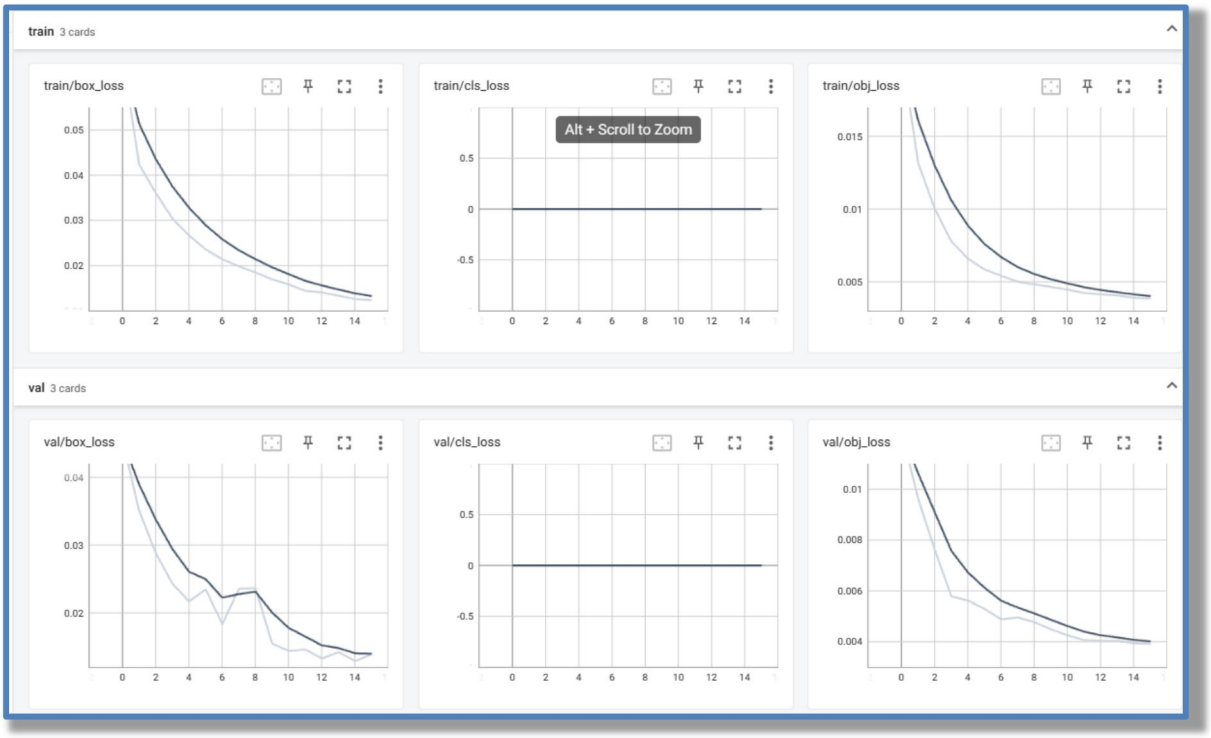
--epochs: 这是一个整数参数，用于指定训练的总轮数（迭代次数）。如果没有提供这个参数，将使用默认值 16。

--batch-size: 这也是一个整数参数，用于指定训练时的批大小。如果设置为 -1，则表示自动调整批大小。如果没有提供这个参数，将使用默认值 8。

这些命令行参数将允许你在运行脚本时指定模型权重文件、模型配置文件、数据集配置文件、超参数配置文件、训练轮数以及批大小等参数，而无需直接修改脚本代码。

3.5.4 使用 tensorboard 查看训练过程细节

具体使用方式见《技术文档》



3.6 学习使用 kaggle 平台获得免费的 GPU 资源训练

在 NoteBook 上熟悉命令行代码指令进行训练，需要把自己的训练数据集上传到平台，然后再使用。

好处：可以分段运行，在每一段代码下可以看到运行的记录，方便查看整个流程

坏处：不方便即时对代码进行更改，需要切换到本机进行修改代码，然后调试。

总结：如果只是因为无 GPU 问题，但是已经调试好代码的情况下可以进行使用。

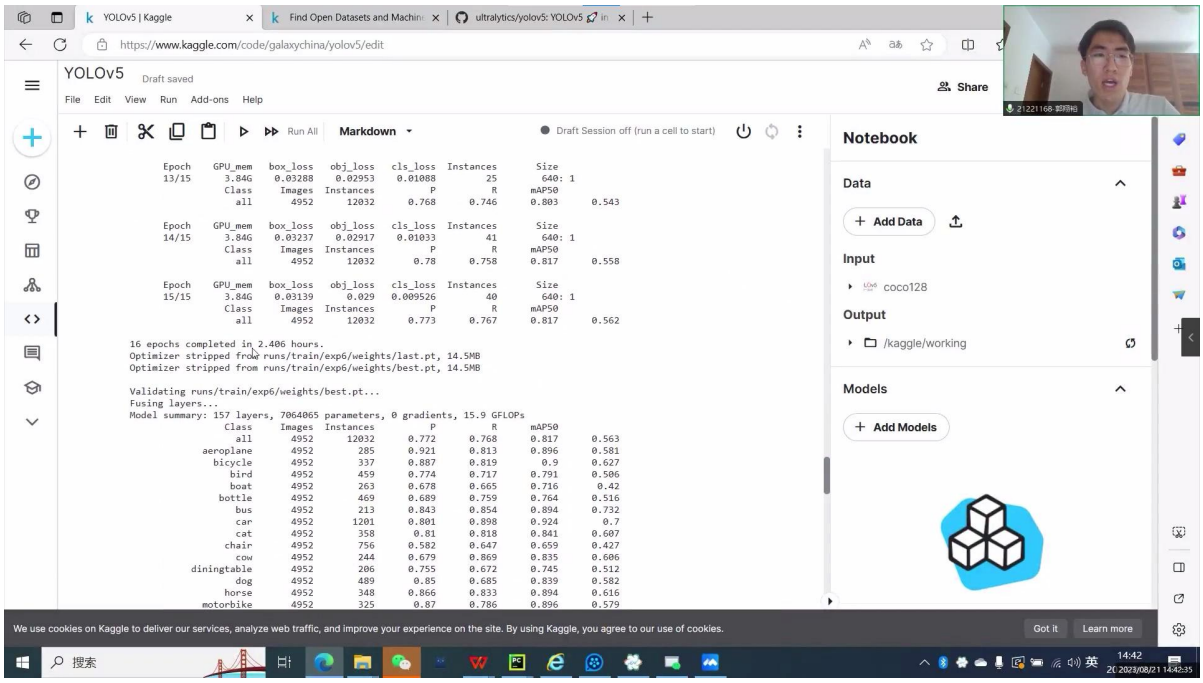


图　kaggle 上的训练情况

3.6.1 熟悉学习使用 Jupeter Notebook 的各项命令

当在 Kaggle 上运行代码时，你可能需要使用以下基础指令：

查看当前目录内容：使用 ls 命令来列出当前目录下的文件和文件夹。

切换目录：使用 cd 命令来切换当前工作目录。

查看文件内容：使用 cat 命令来查看文本文件的内容。

复制文件：使用 cp 命令来复制文件。

移动文件或重命名：使用 mv 命令来移动文件到另一个位置或者重命名文件。

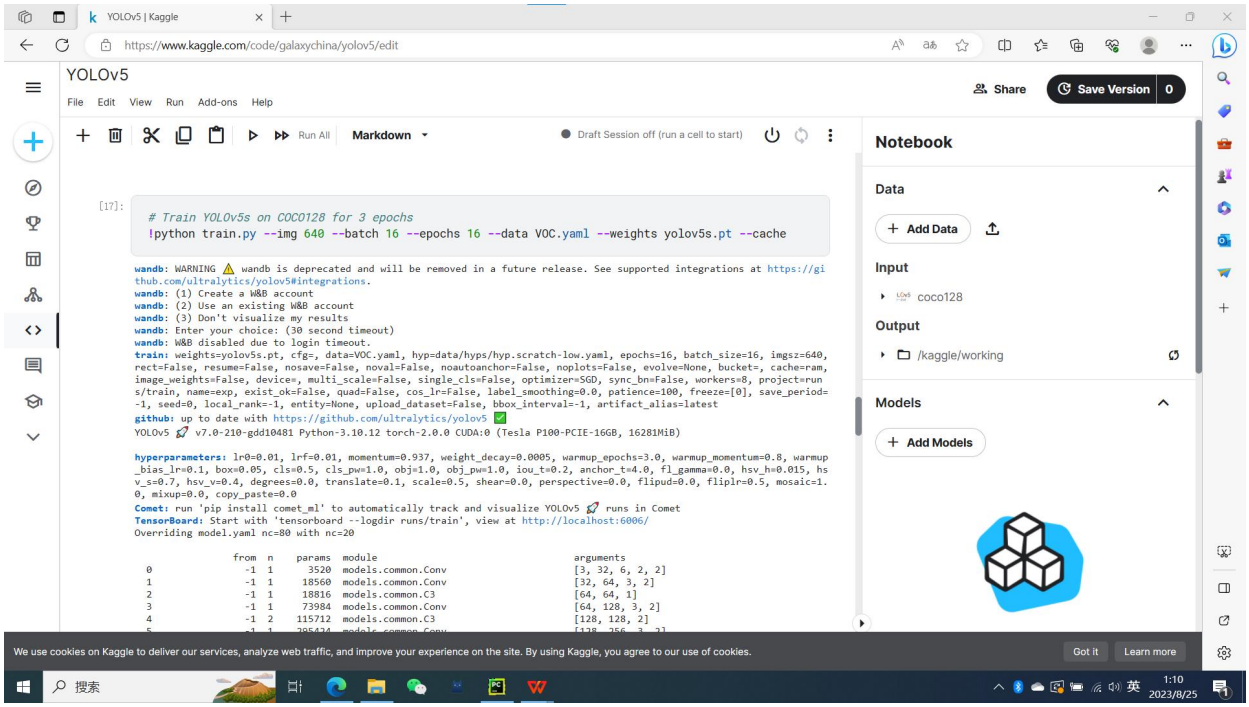
创建文件夹：使用 mkdir 命令来创建新的文件夹。

安装依赖：使用 !pip install package_name 命令来安装 Python 库或其他依赖。

运行 Python 脚本：使用 !python your_script.py 命令在 Notebook 中运行 Python 脚本。

下载文件：使用 !wget URL_of_the_file 命令从 URL 下载文件。

上传文件：在 Kaggle Notebook 中，使用 Kaggle 的文件上传功能。如果在命令行中上传文件，可以考虑使用 kaggle datasets upload 命令。



3.6.2 学习使用平台路径进行调整项目代码参数

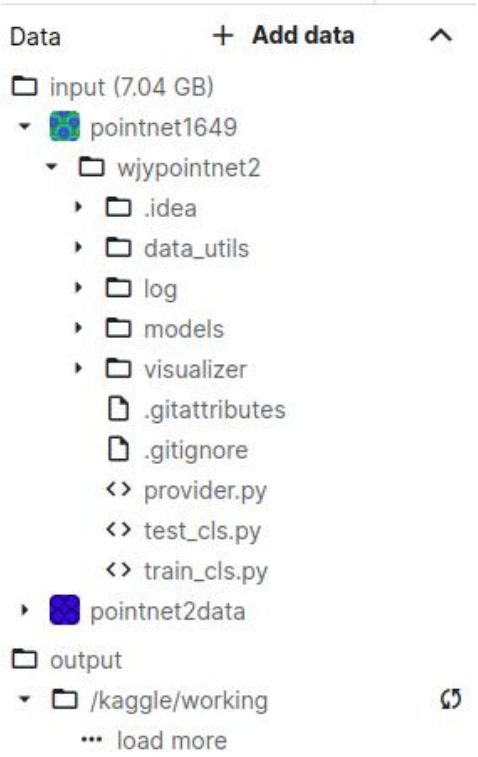
熟悉了 Notebook 的使用，便可以对平台运行代码游刃有余。在 kaggle 平台上，主要问题是代码路径的编写问题，需要多次调试。因此项目最好是可以直接从 GitHub 上 git clone 下来，这样比较方便更新代码，因此需要 3.6.3 使用 Git 工具上传代码到 Github。

在解决项目运行时，需要调整的目录如下规律所示：

```
! python '../input/pointnet1649/wjypointnet2/train_cls.py'
```

‘’ 里面是训练文件的路径。在右侧找到文件，可以直接复制文件路径。一定要注意更改代码里调用的模型路径，否则运行会报错。

下图是我的项目目录。



下图是我训练文件中更改的模型路径。

```
    '''MODEL LOADING'''
    num_class = 40
    MODEL = importlib.import_module(args.model)
    shutil.copy('../input/pointnet1649/wjypointnet2/models/pointnet2_cls_msg.py',
str(experiment_dir))
    shutil.copy('../input/pointnet1649/wjypointnet2/models/pointnet_util.py',
str(experiment_dir))
```

3.6.3 学习如何上传工程文件到 github，并在 kaggle 上使用

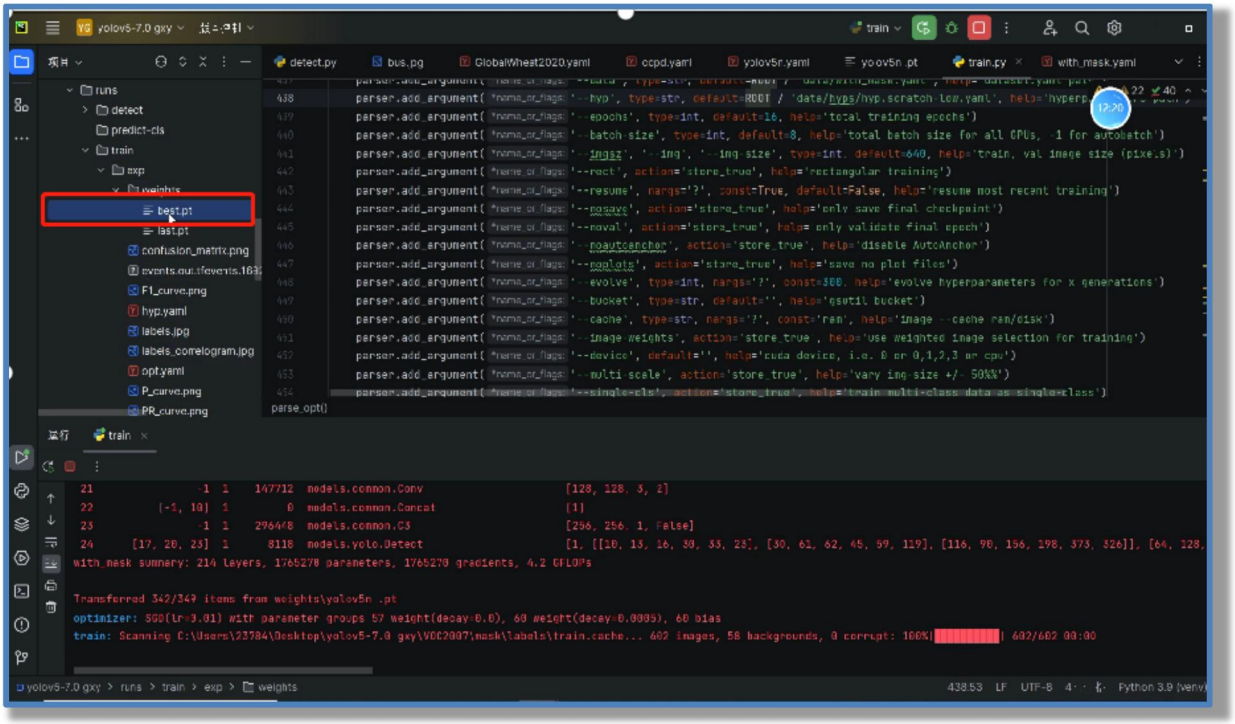
1.下载 git，右键 Open Git Bash
2.git init
3.git add *
4.更改.git/config
[user]
email=chinagalaxy39@gmail.com
name=chinagalaxy2002
5. git commit -m “first commit” (此时显示所有要提交的文件列表)
6. git remote add origin https:xxxxx.git (下划线部分为你的 github 项目地址)
7. git push -u origin master
8. 在项目网页点击 main，选择 master

※项目克隆

如果需要 git clone https:xxxxx.git,结果下载的只有 readme.md，需要 git clone -b master http://gitslab.yiqing.com/declare/about.git，来需要下载 master 这个分支，才是刚才上传的东西。
详细见博客：[git 克隆指定分支代码 git clone -b wudinaniya 的博客-CSDN 博客]

四、检验训练成果

4.1 寻找训练结果的权重文件目录（runs/exp/weight/best.pt）



4.2 调整 detect.py 参数，检验获得目标检测结果

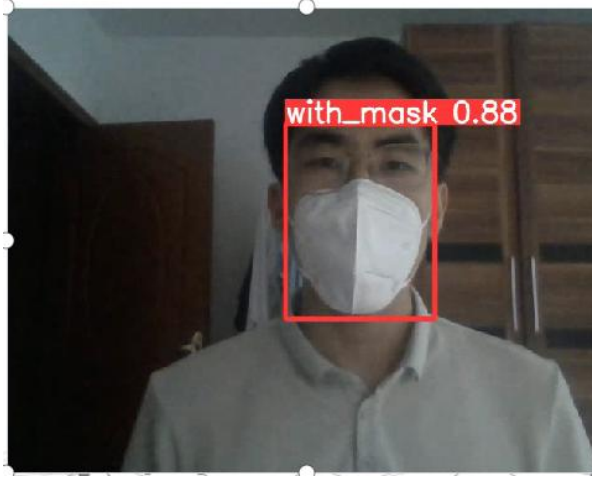
训练得到的模型文件是 best.pt 或者 last.pt，存储在 runs/train/weights 中，我们找到他们，选择 best.pt，查到其路径复制到下图函数中第一行的位置；将第二行的路径可以改为你想要传入模型的照片图片或者本机摄像头。
含义见《yolov5》，如：‘0’（摄像头），’其他任意视频、图片路径即可进行运行’

```
def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'C:/Users/23784/Desktop/yolov5-7.0/runs/train/ex
    parser.add_argument('--source', type=str, default=ROOT / 'C:/Users/23784/Desktop/yolov5-7.0/1.jpg', help='file/dir/UR
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')
```

将四个训练好的模型的 best.pt 的路径分别复制下来，更改 detect.py 的 weights

参数，再将其他参数修改完毕后，运行 detect.py，得到识别结果如下：

1.使用口罩模型处理的视频（郭翔裕）



2.原视频（使用DownKyi爬取哔哩哔哩的数据 ）

2.经过模型处理后的视频（使用DownKyi爬取哔哩哔哩的数据

3.使用kaggle在云端使用COCO数据集训练16epochs模型处理的照片（拍摄郭翔裕）

4.使用CCPD2020数据集，龙老师给的脚本得到yolo数据得到的车牌检测模型得到的车牌检测视频（郭翔裕）

五、项目心得

郭翔裕：《技术文档》 https://kdocs.cn/l/clHohiBTbsLS

郭翔裕：《yolov5》 各代码文件功能概述 https://kdocs.cn/l/cduyuQyP989n