# A1 yolov5face

## A1.1 用github下载代码，用venv结合vscode跑起来

### 小结：

A1.1.1 如何在有墙的情况下，安装并使用（拉仓库，代码同步与提交）github客户端

A1.1.2 略

A1.1.3第一次运行test.py需要的一些调整

A1.1.4数据集Dataset和.pt的路径调整，以及对应需要为test.py改哪些东西

### A1.1.1 墙的问题

以下操作，在有墙的时候，会非常不稳定：需要翻墙（当你可以翻墙时），或使用Watt Toolkit勾选github加速

1. 下载Github客户端安装软件GitHubDesktopSetup-x64.exe
2. Github客户端登陆
3. GIthub同步代码
4. 运行库代码是，自带的下载.pt和coco 数据集操作

### A1.1.2 venv结合vscode用法

请参考教程：[傻瓜级Venv+VsCode调试(Release和Debug两种模式)配置教程（含常见问题解决办法）\_vscode venv-CSDN博客](https://blog.csdn.net/weixin_42941859/article/details/140243028)

截止到2025年1月6日：按照以上教程有可能会出现，不能debug的问题（尤其迁移了环境之后）。解决的办法或许有使用vscode自带的venv，具体方法为ctrl+shift+p，选择Create Virtual Environment

### A1.1.3 仓库，requirements.txt和第一次运行

使用的仓库路径为 <https://github.com/deepcam-cn/yolov5-face>

问题：该仓库并没有包含requirements.txt以方便使用pip install直接配置环境。

解决：鉴于本仓库明显是基于yolov5的，所以从yolov5的官网（https://github.com/ultralytics/yolov5）下载了requirements.txt使用。

结果：基本是可以适配的，

问题a&解决：numpy.int需要全部替换成python自带的int

问题b&解决：common.py需要import warnings

问题c：

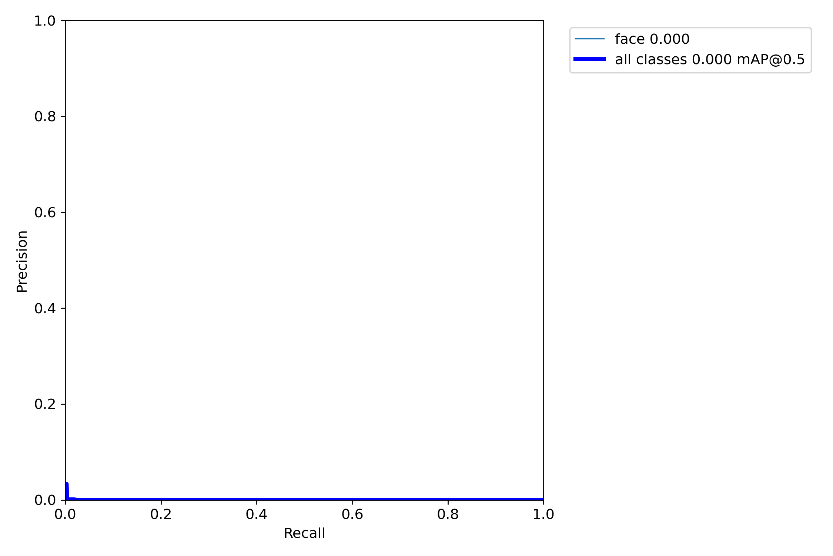
分析：根据仓库的issues 204（），此问题是由于test.py默认下载并运行的‘yolov5s.pt’

    parser.**add\_argument**('--weights', nargs='+', type=**str**, default='yolov5s.pt', help='model.pt path(s)')

解决：手动下载，在test.py将上面的default=‘yolov5s.py’换成default='yolov5n-face.pt'

Link: <https://pan.baidu.com/s/1UgiKwzFq5NXI2y-Zui1kiA> pwd: s5ow  
Link: <https://pan.baidu.com/s/1xsYns6cyB84aPDgXB7sNDQ> pwd: lw9j  
Link: <https://pan.baidu.com/s/1fyzLxZYx7Ja1_PCIWRhxbw> Link: eq0q  
Link: <https://pan.baidu.com/s/1oePvd2K6R4-gT0g7EERmdQ> pwd: jmtk  
Link: <https://pan.baidu.com/s/11l4qSEgA2-c7e8lpRt8iFw> pwd: 0mq7

问题d：运行出来的不正确PR数据集不太正确，如下图所示。



分析：这估计是因为coco数据集是多类型目标检测，以至于单目标网络的结果画不出来。

结果：暂时别管了

### A1.1.4 调整Dataset和.pt的目录

将Coco128数据集放到DataSet文件夹下。

则 data/coco128.yaml文件也需要跟着修改

同时，\DataSet\coco128\labels\train2017.cache也要删掉，触发重新生成。因为里面存的是绝对路径，随着Coco128数据集路径的移动，需要重新更新。

将pretrain的yolov5n-0.5.pt、yolov5n-face.pt等文件收拢到\weights\Pretrained。

则parser.**add\_argument**('--weights', nargs='+', type=**str**, default='weights/Pretrained/yolov5n-face.pt', help='model.pt path(s)')

也需要跟着修改

## A1.2 WiderFace数据集

链接：[WIDER FACE: A Face Detection Benchmark](http://shuoyang1213.me/WIDERFACE/)

截至到20250107，WiderFace的下载共有5项，分别是Training Images, Validation Images, Test Images，Face annotation和Example and formats of the submissions

### A1.2.1 Training Images, Validation Images, Test Images

三个zip中，分别有12880(train) + 16097(test) + 3226(val)张jpg图像

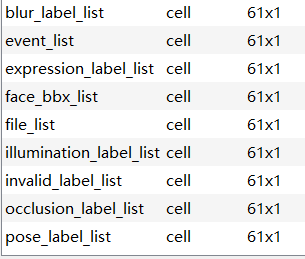
### A1.2.2 Face annotation: wider\_face\_val.mat/\_gt.txt

共有7个文件，除了readme.txt以外，以下先分析wider\_face\_val.mat

这个.mat文件得用matlab打开，没有matlab也可以用octave打开，指令如下：

load('C:\Users\xiaoyaopan\Downloads\wider\_face\_split\wider\_face\_val.mat')

使用此指令，得到两个变量：



这里的61对应的是WIDER\_val/images中的61个文件夹。

file\_list从这61个文件夹中，挑选对应的图像，例如file\_list[0][0]的值是0\_Parade\_marchingband\_1\_465，对应了WIDER\_val/images/0—Parade/0\_Parade\_marchingband\_1\_465.jpg。则blur\_label\_list[0][0]是这张图片中，各个脸的模糊程度，face\_bbx\_list[0][0]是各个脸的bbox。具体每个的含义，可参考wider\_face\_val.mat同路径下的readme.txt。

以这个顺序排列的ground truth, 则列在了wider\_face\_val\_bbx\_gt.txt中，也就是上面这个mat整理出了简约的形式。

### A1.2.3 Face annotation: wider\_face\_test\_filelist.txt

A1.2.2解释了，训练集train和验证集val的label们是如何排列在.mat和.txt文件中的。

而测试集test略有不同：概括地说，wider\_face\_test.mat相比wider\_face\_val.mat，以及wider\_face\_test\_filelist.txt相比wider\_face\_val.txt，删减只剩了文件路径和场景。

因为test集是用来做Benchmark的，不能用来训练，所以不提供ground truth.

### A1.2.4 retinaface\_gt\_v1.1.zip

在执行A1.2.5 Wider数据集转成Yolo5face格式之前，还需要下载retinaface\_gt\_v1.1.zip

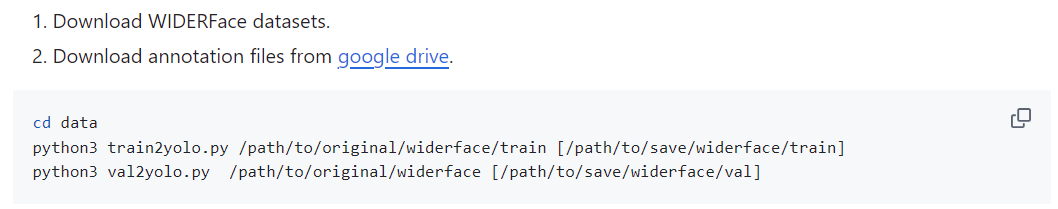
链接：

<https://drive.google.com/file/d/1tU_IjyOwGQfGNUvZGwWWM4SwxKp2PUQ8/view?usp=sharing>

因为，Yolo5face产生的结果是‘框+5个关键点’，所以要用retinaface的输出作为gt。

### A1.2.5 Wider数据集转成Yolo5face格式

Yolo5face的官网是这样描述的：



这段代码的作用，是把WiderFace的数据集，转换成yolo格式

也就是每张图配一个txt



每个txt中，每一行是一张脸，14个float，对应的意义是：

框的中心点(x,y,w,h) + 5个关键点(x,y)

### A1.2.6 CPU训练：指令

在运行这个指令之前，和A1.1.4一样，需要调整data/widerface.yam 测试集的路径，train该路径应该设置为0\_Parade\_marchingband\_1\_5.txt所在的路径（如果A1.2.5结果正确的话，应该存在这个.txt），val路径依次类推。

python train.py --data data/widerface.yaml --cfg models/yolov5s.yaml --weights weights/Pretrained/yolov5s-face.pt --device cpu --workers 1

相比github官网上的指令：--data data/widerface.yaml --cfg models/yolov5s.yaml --weights 'pretrained models'，有如下修改

1. --weights weights/Pretrained/yolov5s-face.pt，因为A1.1.4中，模型.pt的文件位置调整了；同时因为A1.1.3中的问题c，需要把yolov5s.pt换成yolov5s-face.pt。
2. --device cpu 要设置成cpu，否则默认会调用GPU。
3. --workers 1 这个代表并行训练数量，默认是4，可能会把电脑内存塞满，因为每个worker都要载入一个batch

### A1.2.7 CPU训练：结果分析

问题a：文件夹exp/12下的results.txt是一组数据，但没有第一行数据是什么。

问题b：文件夹exp/12下的results.txt的第一列，从/249突变成/119，看不懂。

问题c：文件夹exp/12下的results.png，横轴是什么东西？纵轴是什么？为什么有阶跃？

TODO以上一系列问题，暂时列为TODO，与其调试yolo5face代码，不如调试yolo5代码。

# A2 小土堆B站Pytorch教程

https://www.bilibili.com/video/BV1hE411t7RN/?spm\_id\_from=333.337.search-card.all.click

## P1：Pytorch环境的配置及安装

0432：用anaconda做venv

0932：装什么样的Pytorch（尤其是和Cuda配合）

1026：Nvidia驱动查看与更新

1234：cudaToolKit和pytorch的bz2安装

1405：torch.cuda.is\_abailable()

## P2：IDE

略

## P2.5：Q&A torch.cuda.is\_available返回false

RT，需要时参考

后半截的内容是，conda安装慢怎么办，参考A1.1节的内容理应能够解决。

## P3 Python学习中的两大法宝函数dir和help

从0415开始看即可，是实际的例子

## P4 PyCharm以及Jupyter的使用及对比

暂略

## P5 PyTorch加载数据初认识

# B1 yolo5 quick start

源码官网 [ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite](https://github.com/ultralytics/yolov5)

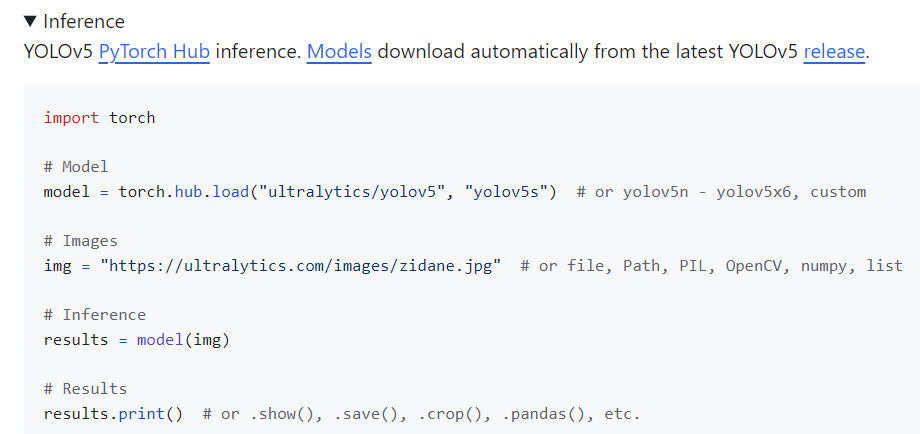
<https://github.com/ultralytics/yolov5> [B1A]

## B1.1 Install

根据A1.1.1~A1.1.3的方法，安装yolo5

## B1.2 First Try (建议放弃)

[B1A]中建议的使用Yolo5的First try是以下这个：



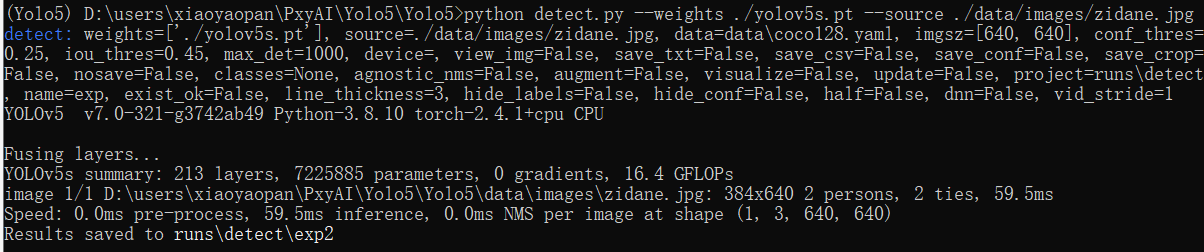
问题：在用torch.hub.load和img=时，会有各种网络的问题，建议直接使用下一节B1.3

## B1.3 更简单的First Try

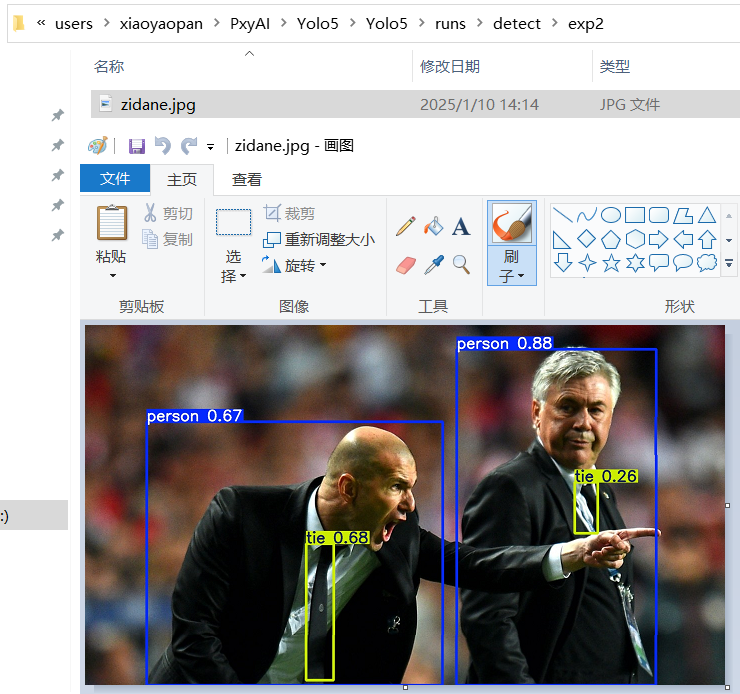
（在激活vevn之后）进入clone Yolo5的目录，使用以下指令：

python detect.py --weights./yolov5s.pt --source ./data/images/zidane.jpg

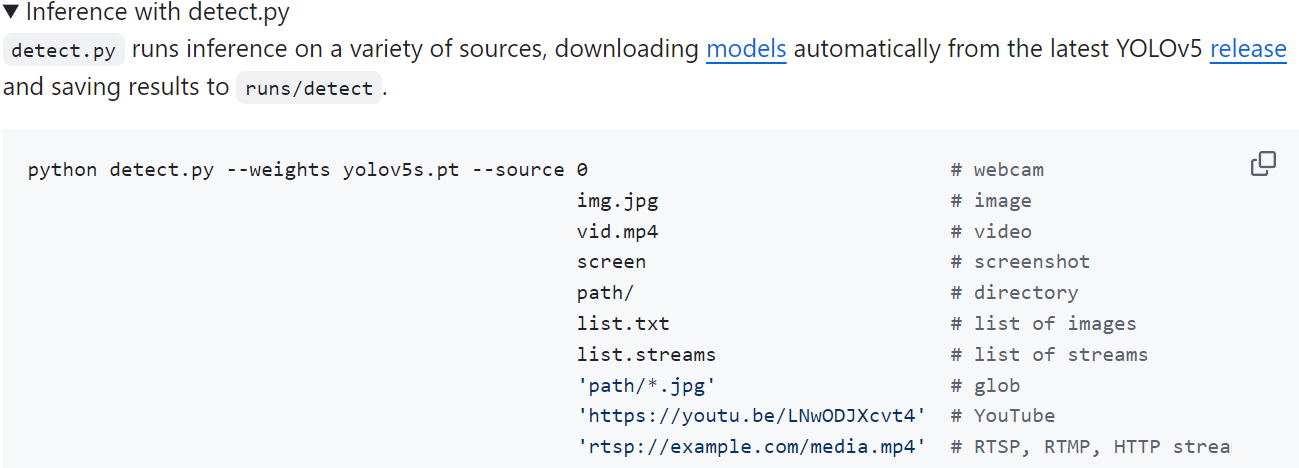
cmd成功界面如下：



出图界面如下：



以上测试，对应的是[B1A]的这一段：用到的detect.py文件的cmd 参数可以打开文件看，很容易理解。



## B1.4 用CPU跑Yolo5的coco128数据集

### 指令和指令解析：

（管理员）D:\users\xiaoyaopan\PxyAI\Yolo5\Yolo5>python train.py --data coco.yaml --epochs 1 --weights '' --cfg yolov5n.yaml --batch-size 32

指令解析：

基本等同于A1.2.6的作法，由于此时尚未安装Pytorch的Nvidia版本，所以不需要单独指定--device cpu

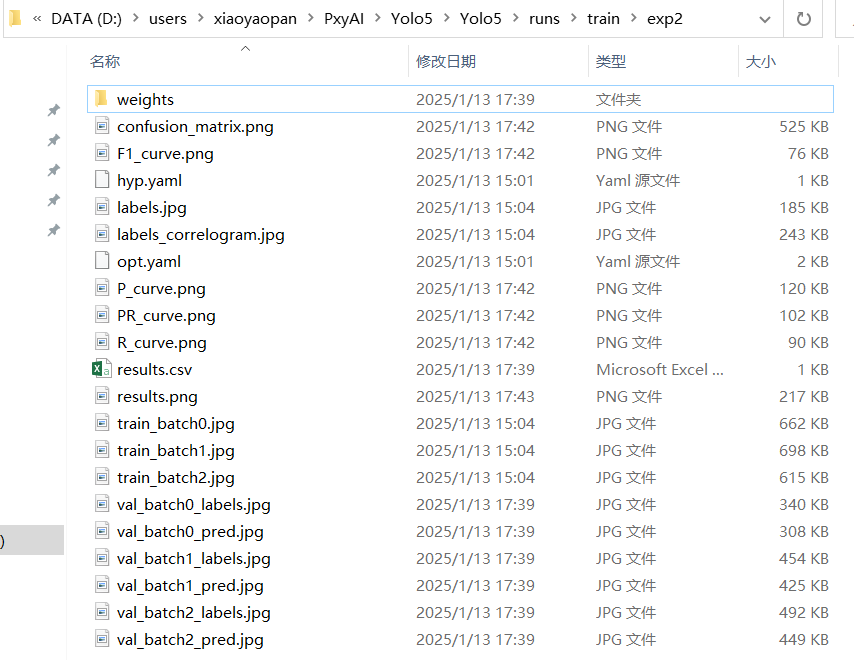
### 运行结果：

1. 下载coco128的数据集以及解压缩（尤其解压缩）非常慢，请耐心等待，并没有死机。

2. 运行完毕后cmd窗口如下：

TODO

3. 新产生了如下目录D:\users\xiaoyaopan\PxyAI\Yolo5\Yolo5\runs\train\exp2，也就是本次训练的产出，以下逐个分析：



### 结果分析：weights

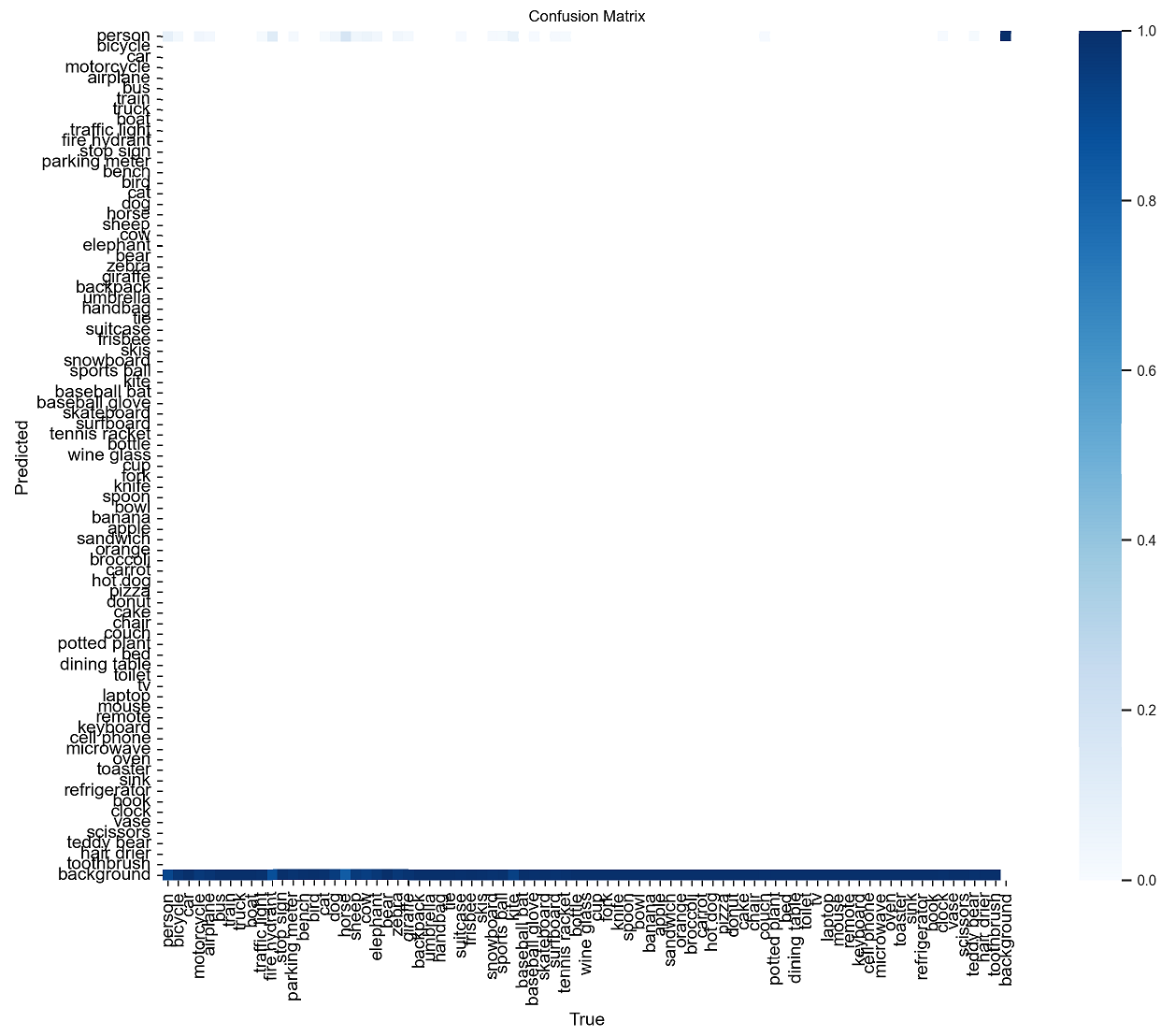
weights/best.pt 大小：4,021,032字节

weights/last.pt 大小：4,021,032字节

显然是同一个，毕竟只训练了一个epoch。

没有初始版本.pt可以比较，因为是从头开始训练的。

### 结果分析：confusion\_matrix.png



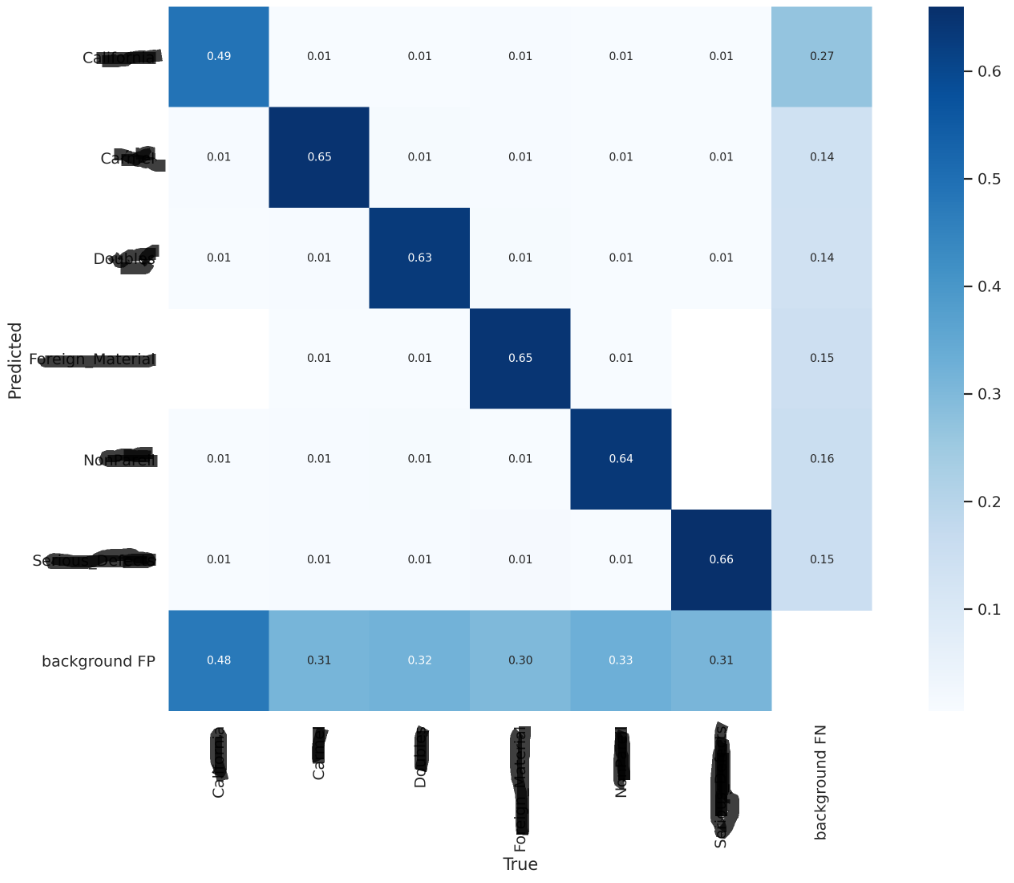
横坐标：True：指的是应该检测出，图像在这里有一个属于当前横坐标的框。

纵坐标：Predicted：指的是实际检测出，图像在这里有一个属于当前纵坐标的框。

为什么纵坐标Predicted为background的那一行颜色很深（比较接近1）：因为有大量的该检测出框，没有被检测出来，从而被归于了backround。（原因是coco128数据集人的数量很多，在只有一次epoch的情况下还能有点拟合效果，其他的类别都欠拟合/欠训练了，详见“结果分析：labels.jpg”）。

为什么纵坐标为Predicted为person，横坐标True为background的那一格颜色很深（很接近1）：因为有大量的背景被检测成了人脸（或者说人类的框预测得大致对，但不准，所以和标准答案IoU之后，认为框出来的不能算人，还是得算background）

看一个理想中训练的不错的confusion\_matrix的样子：



### 结果分析：F1\_curve.png

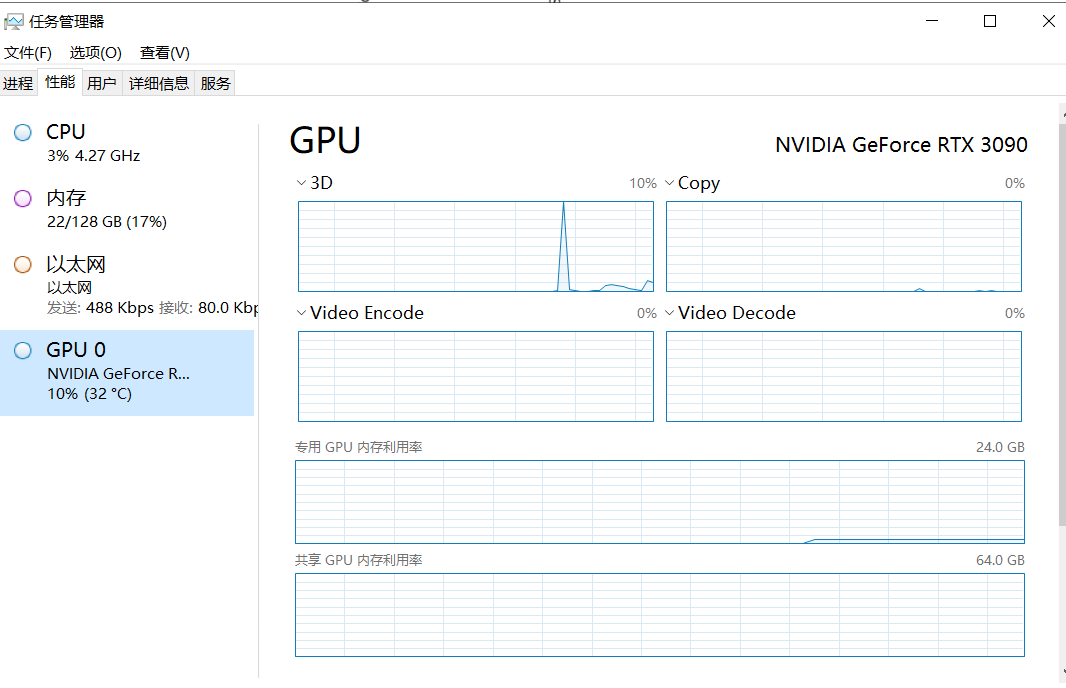
以下的暂略，先把GPU和“继续训练”

### 结果分析：hyp.yaml（炼丹超参数，暂略）

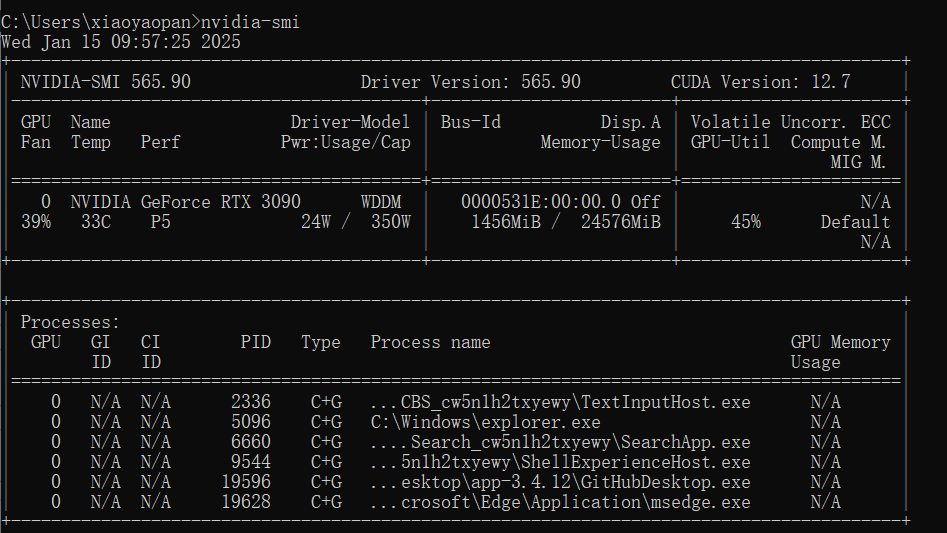
## B1.5 用GPU跑Yolo5的coco128

### Nvidia版本Pytorch的安装

当前显卡参数：



当前驱动版本：cmd窗口输入nvidia-msi



指令：

pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url https://download.pytorch.org/whl/cu124

指令哪里来：<https://pytorch.org/get-started/previous-versions/>

为什么要用previous-version，因为本章中安装的python3.8不支持最新版的pytorch

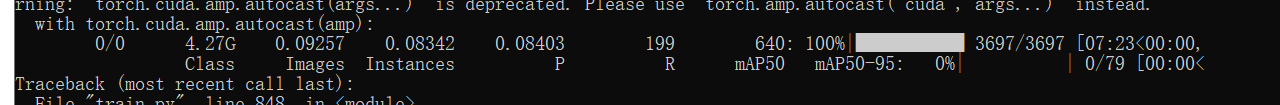
指令的效果：

使用这个指令，会自动让pip卸载掉cpu版本的pytorch，然后安装Nvidia版本的pytorch。

### 训练指令：

（管理员）D:\users\xiaoyaopan\PxyAI\Yolo5\Yolo5>python train.py --data coco.yaml --epochs 1 --weights '' --cfg yolov5n.yaml --batch-size 32

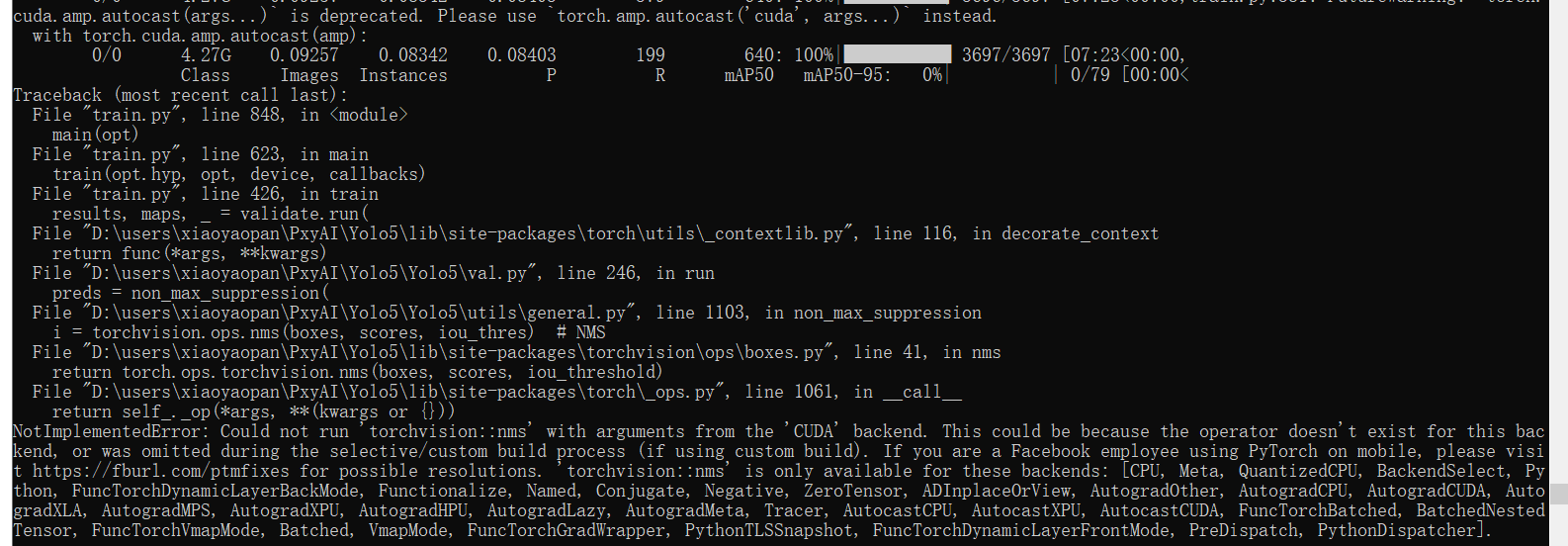
### 时间节省效果：



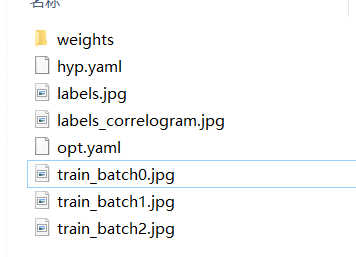
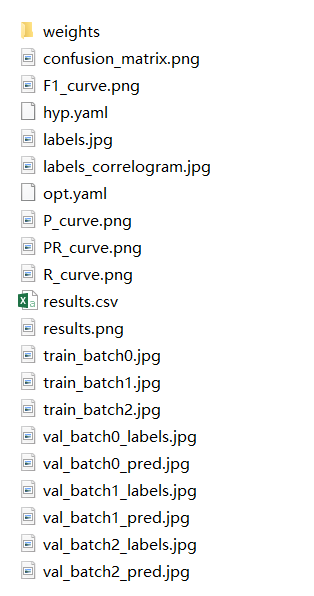
速度确实快多了，只用了7分23秒；相比之下，CPU版本的需要使用2~3个小时。

### 问题&解决：

问题：



Torchvision.ops.nms调用有问题，导致val集没能跑，看不到验证集的验证结果

本次正常训练完，但val没能跑到的，exp文件夹下的内容（左图）

正常测试完成后的exp文件夹下内容（右图）

原因：

torch从CPU版本换成Nvidia版本后，torchvision没有跟着换（本应是跟着torch自动换的）

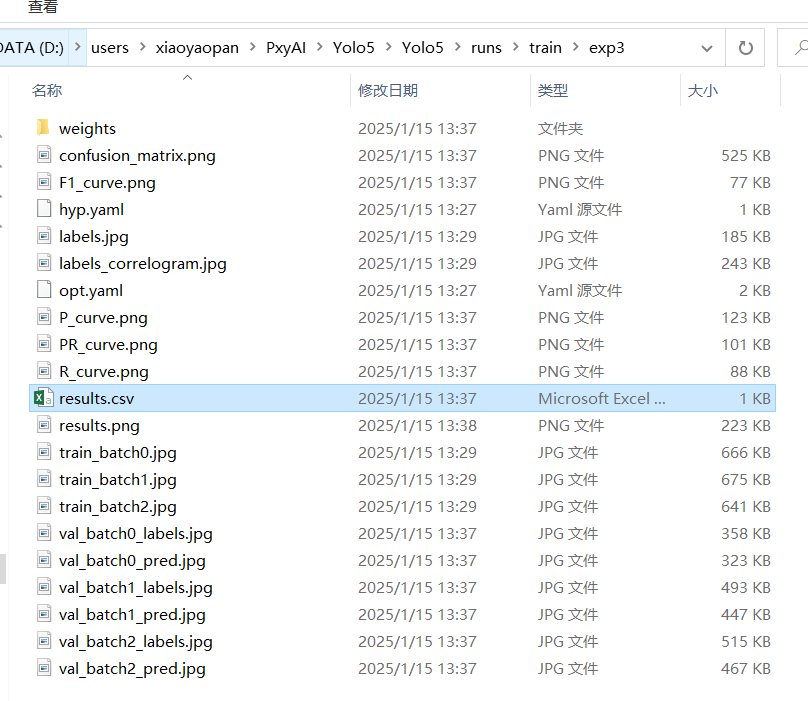
解决：

先卸载掉cpu版本的torchvision，在安装用上面的pip install torch指令，把合适版本的torchvision装回来

pip uninstall torchvision

pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url <https://download.pytorch.org/whl/cu124>

### 训练结果分析：



训练成功

## B1.6 用GPU分开跑5次Yolo5的coco128，前后继承