

一、创建虚拟环境

Q: 为什么要使用虚拟环境?

使用Anaconda创建yolo v8运行的虚拟环境

二、安装深度学习需要的库

1.安装除pytorch之外的库

报错一:

报错二:

报错三:

2.安装Pytorch

一、创建虚拟环境

Q: 为什么要使用虚拟环境?

A: 在Windows中, 如果不使用虚拟环境, 默认会将各种安装包安装到系统的环境变量中。

例: 如果你之前安装过Python, 并将Python的路径添加到了系统的环境变量, 那么你可以在win+R -> cmd 打开的命令行中输入python来启动python。如图:

```
管理员: C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 10.0.19044.1288]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>python
Python 3.8.0 (default, Nov 6 2019, 16:00:02) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

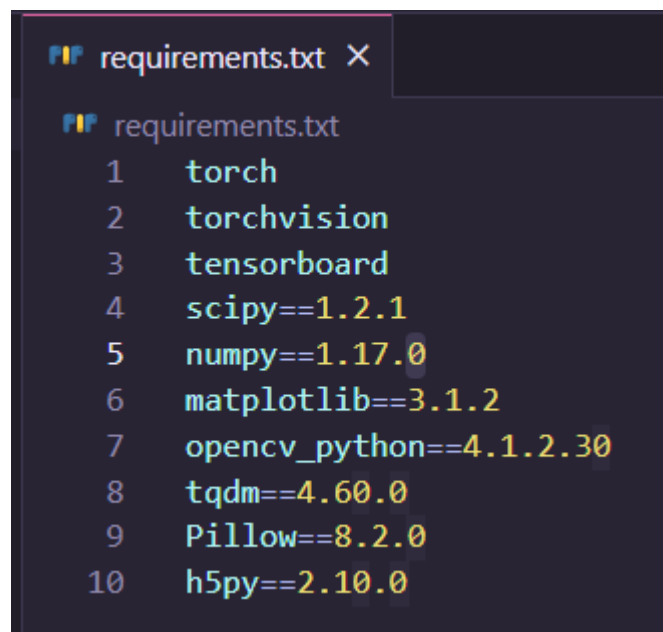
注意到该python的版本为3.8.0。如果你需要安装python需要用的其他的库, 如numpy, 也会安装到系统的环境变量中。

```
管理员: C:\Windows\system32\cmd.exe - python

C:\Users\Administrator>python
Python 3.8.0 (default, Nov 6 2019, 16:00:02) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> print(numpy.__version__)
1.24.4
>>>
```

上图中安装的numpy为1.24.4版本。

此时你打算跑一个深度学习的代码, 一般比较完善的代码中都会有requirements.txt这样一个环境要求文档, 你需要安装其中所有的库才能将代码跑起来, 如图为yolo v8的库版本要求:



```
requirements.txt X
requirements.txt
1 torch
2 torchvision
3 tensorboard
4 scipy==1.2.1
5 numpy==1.17.0
6 matplotlib==3.1.2
7 opencv_python==4.1.2.30
8 tqdm==4.60.0
9 Pillow==8.2.0
10 h5py==2.10.0
```

可以看到上图中要求的numpy版本为1.17.0，那么我之前安装的numpy可能就会让这个代码跑不起来。如果此时我需要更换numpy的版本，就需要将原来的版本卸载掉，根据要求来安装。而如果你要想再跑yolo v7，其库版本要求如下：



Code

Blame

39 lines (34 loc) · 958 Bytes



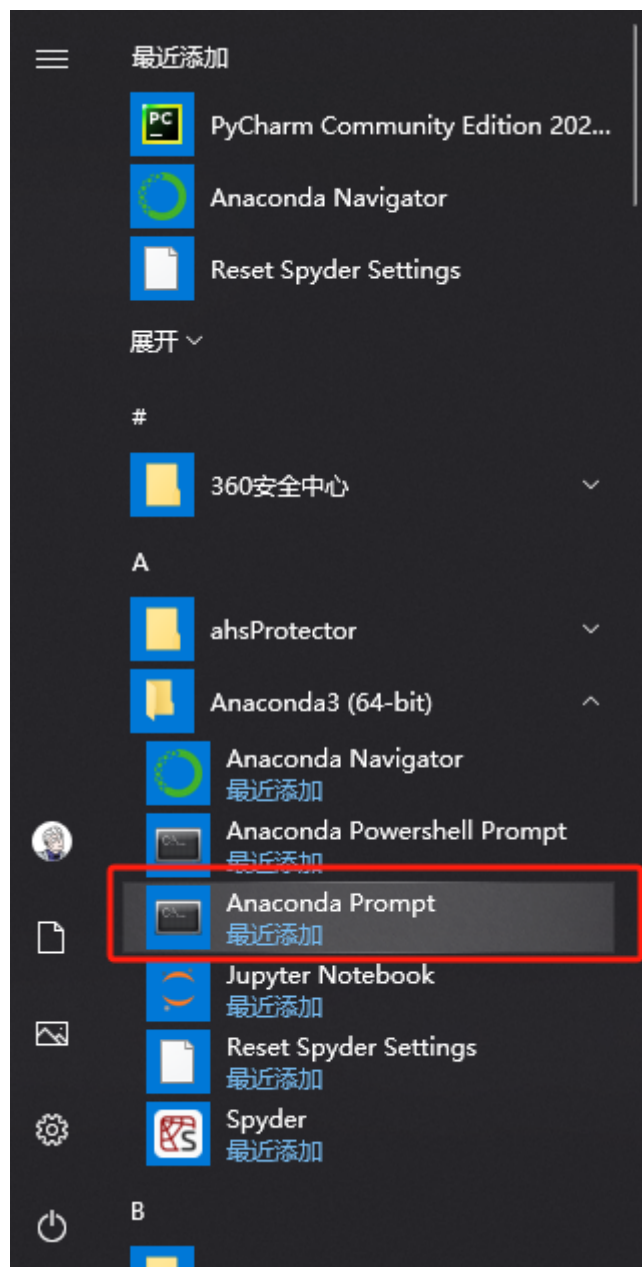
Code 55% faster with GitHub Copilot

```
1  # Usage: pip install -r requirements.txt
2
3  # Base -----
4  matplotlib>=3.2.2
5  numpy>=1.18.5,<1.24.0
6  opencv-python>=4.1.1
7  Pillow>=7.1.2
8  PyYAML>=5.3.1
9  requests>=2.23.0
10 scipy>=1.4.1
11 torch>=1.7.0,!<1.12.0
12 torchvision>=0.8.1,!<0.13.0
13 tqdm>=4.41.0
14 protobuf<4.21.3
15
16 # Logging -----
17 tensorboard>=2.4.1
18 # wandb
19
20 # Plotting -----
21 pandas>=1.1.4
22 seaborn>=0.11.0
23
24 # Export -----
25 # coremltools>=4.1 # CoreML export
26 # onnx>=1.9.0 # ONNX export
27 # onnx-simplifier>=0.3.6 # ONNX simplifier
28 # scikit-learn==0.19.2 # CoreML quantization
29 # tensorflow>=2.4.1 # TFLite export
30 # tensorflowjs>=3.9.0 # TF.js export
31 # openvino-dev # OpenVINO export
32
33 # Extras -----
34 ipython # interactive notebook
35 psutil # system utilization
36 thop # FLOPs computation
37 # albumentations>=1.0.3
38 # pycocotools>=2.0 # COCO mAP
39 # roboflow
```

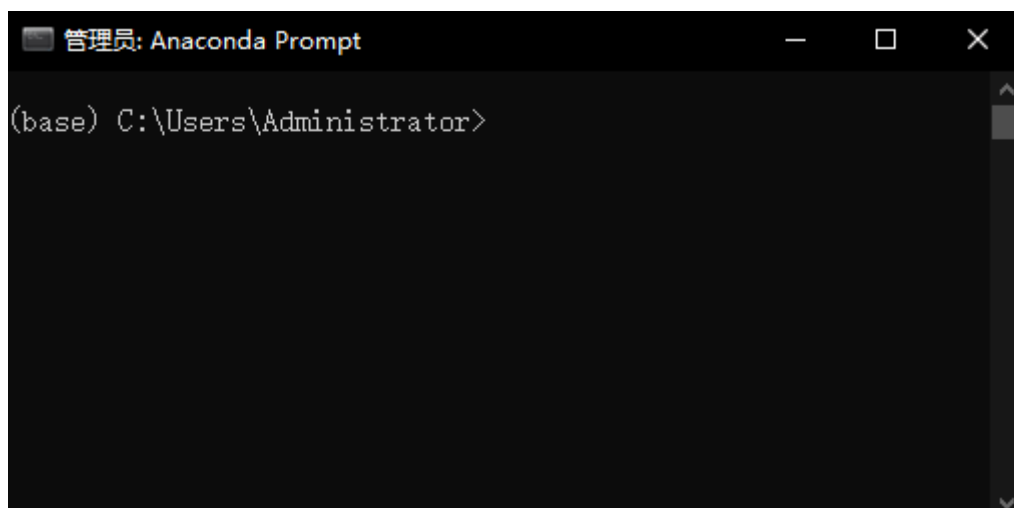
可以看到其需求的库非常多，这也是新手跑深度学习时最头疼的问题。当你按照yolo v7的要求对环境修改改，终于跑了起来，你会发现原来跑的好好的yolo v8再也跑不起来了。因此我们需要使用虚拟环境来为不同的项目做出不同的环境配置。

使用Anaconda创建yolo v8运行的虚拟环境

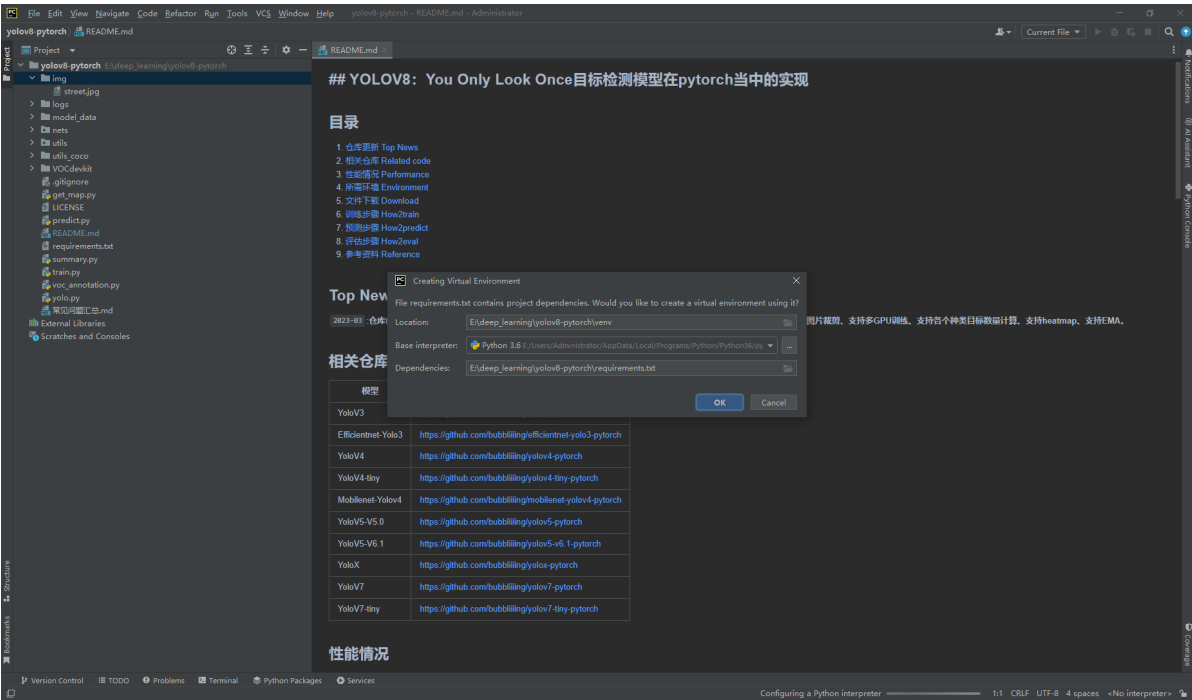
在开始界面中找到Anaconda Prompt并打开



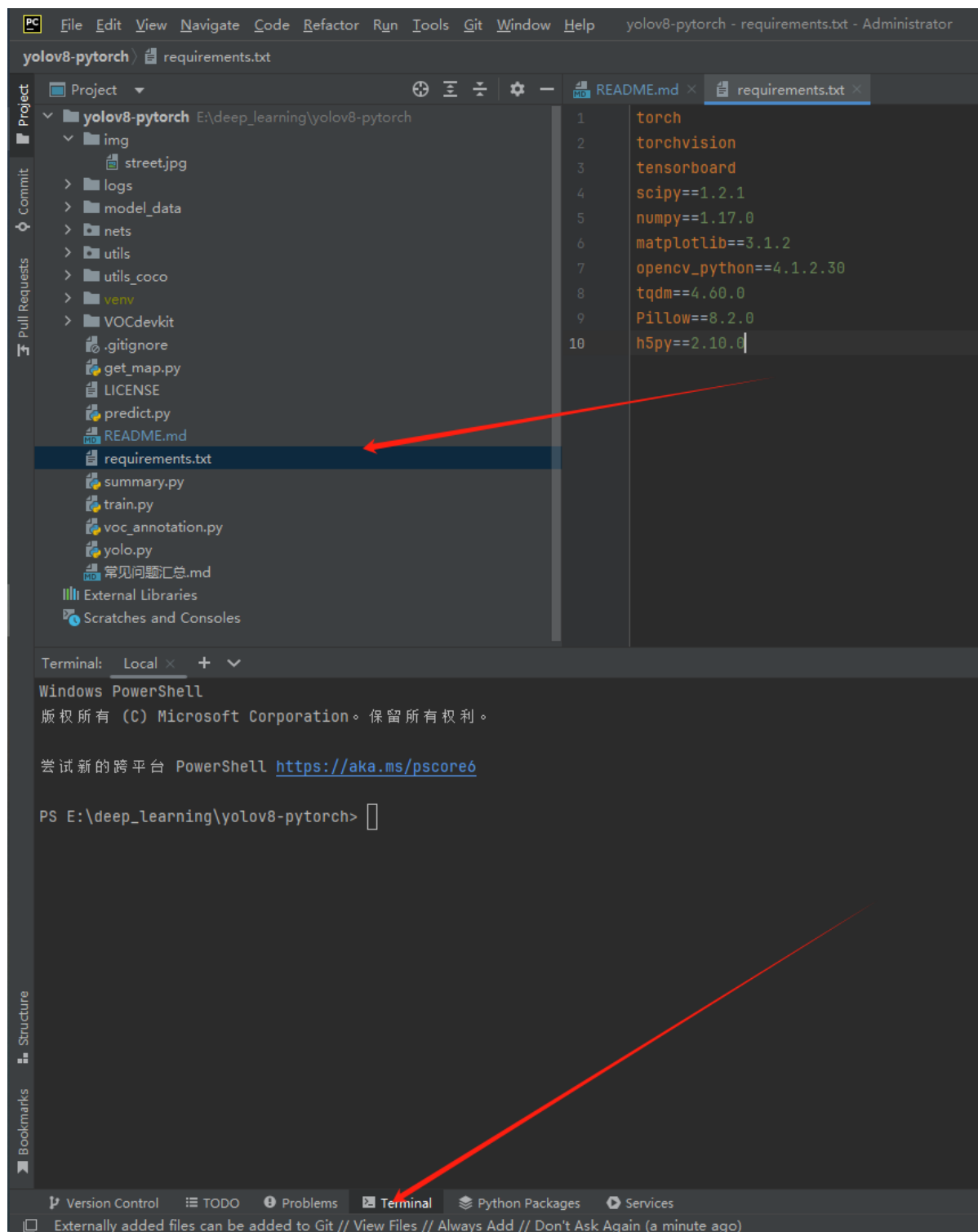
打开的命令行中会有(base)标识，表示当前处于conda默认的base环境下。base环境是安装完Anaconda时默认安装的一个虚拟环境



使用git或者直接下载yolo v8的代码，并解压缩，使用Pycharm打开该项目。此时会让你选择Python解释器，我们还没有配置好环境，所以暂时先不去管他：



双击打开左侧的requirements.txt来查看安装要求，并在下面单击打开一个终端：



输入

```
conda create -n v8 python=3.9
```

其中v8为虚拟环境的名称，表示创建的环境供yolo v8使用，你可以随便起一个。python=3.9表示在创建环境时安装一个3.9版本的python，具体要安装哪个版本的要看项目要求，本文所用的代码要求大于3.7，随便安装一个符合要求的就行。系统提示以下包将被安装，输入y并按回车继续

```
The following NEW packages will be INSTALLED:
```

bzip2	pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
ca-certificates	pkgs/main/win-64::ca-certificates-2024.3.11-haa95532_0
libffi	pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
openssl	pkgs/main/win-64::openssl-3.0.13-h2bbff1b_1
pip	pkgs/main/win-64::pip-24.0-py310haa95532_0
python	pkgs/main/win-64::python-3.10.14-he1021f5_1
setuptools	pkgs/main/win-64::setuptools-69.5.1-py310haa95532_0
sqlite	pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
tk	pkgs/main/win-64::tk-8.6.14-h0416ee5_0
tzdata	pkgs/main/noarch::tzdata-2024a-h04d1e81_0
vc	pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime	pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel	pkgs/main/win-64::wheel-0.43.0-py310haa95532_0
xz	pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
zlib	pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1

```
Proceed ([y]/n)?
```

出现以下提示就说明一个新的虚拟环境就创建好了：

```
Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate v8
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

输入conda env list来查看当前已有环境

```
PS E:\deep_learning\yolov8-pytorch> conda env list
# conda environments:
#
base                        D:\ProgramData\anaconda3
v8                          D:\ProgramData\anaconda3\envs\v8
d:\ProgramData\anaconda3
```

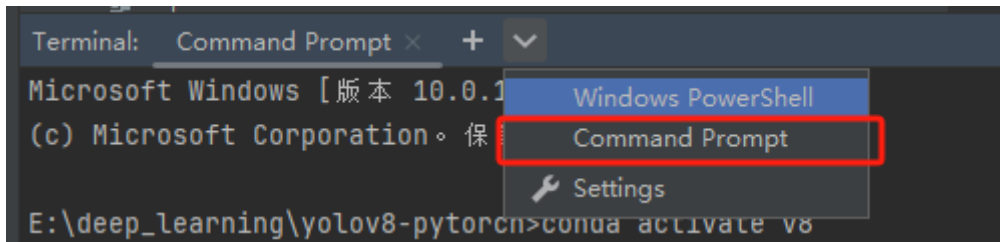
看到当前除了base环境，还有一个名为v8的环境是我们刚才安装的。接下来需要在我们的虚拟环境里安装深度学习所需的库，即requirements.txt里提到的那些。

二、安装深度学习需要的库

在命令行中输入conda activate v8，此时在路径的前面会有一个（v8）表示已经激活了该环境

```
E:\deep_learning\yolov8-pytorch>conda activate v8  
(v8) E:\deep_learning\yolov8-pytorch>
```

如果输入完没有反应，检查一下终端的模式，要切换成第二个：



此时可以开始在当前环境下安装所需的库了。

1.安装除pytorch之外的库

注意到requirements.txt里第一个是torch，我们先将他屏蔽掉，在前两行的前面加上#来注释掉(torch这个库比较特殊，后面单独讲怎么安装)，安装其他的库。

```
# torch  
# torchvision  
tensorboard  
scipy==1.2.1  
numpy==1.17.0  
matplotlib==3.1.2  
opencv_python==4.1.2.30  
tqdm==4.60.0  
Pillow==8.2.0  
h5py==2.10.0
```

在命令行中运行

```
pip install -r ./requirements.txt
```

如果下载过慢，或者连接超时，可以尝试换源，即在命令后面加上-i <https://pypi.mirrors.ustc.edu.cn/simple/>，如

```
pip install -r ./requirements.txt -i https://pypi.mirrors.ustc.edu.cn/simple/
```

该命令将下载地址从官方源更换为中科大源，也可换成其他源，如清华源/豆瓣源，只需要更换后缀即可。具体可自行搜索。

报错一：

```
ERROR: Ignored the following yanked versions: 3.4.11.39
ERROR: Could not find a version that satisfies the requirement
opencv_python==4.1.2.30 (from versions: 3.4.0.14, 3.4.10.37, 3.4.11.41,
3.4.11.43, 3.4.11.45, 3.4.13.47, 3.4.14.51, 3.4.14.53, 3.4.15.55, 3.4.16.57,
3.4.16.59, 3.
4.17.61, 3.4.17.63, 3.4.18.65, 4.3.0.38, 4.4.0.40, 4.4.0.42, 4.4.0.44, 4.4.0.46,
4.5.1.48, 4.5.2.52, 4.5.2.54, 4.5.3.56, 4.5.4.58, 4.5.4.60, 4.5.5.62, 4.5.5.64,
4.6.0.66, 4.7.0.68, 4.7.0.72, 4.8.0.74, 4.8.0.76, 4.8.1.78, 4.9.0.80)
ERROR: No matching distribution found for opencv_python==4.1.2.30
```

表明在系统中找不到opencv_python==4.1.2.30这个库，但是有以下版本可以选择。估计是该版本已经被淘汰了，我们修改requirements.txt中的opencv_python==4.1.2.30 改为 opencv_python==4.3.0.38，重新执行上面的命令，可以看到这次可以正常下载了：

```
Collecting opencv_python==4.3.0.38 (from -r ./requirements.txt (line 7))
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/a1/d6/8422797e35f8814b1d9842530566a949
d9b5850a466321a6c1d5a99055ee/opencv-python-4.3.0.38.tar.gz (88.0 MB)
_____ 17.8/88.0 MB 232.1 kB/s eta 0:05:03
```

报错二：

```
ERROR: Cannot install -r ./requirements.txt (line 4), -r ./requirements.txt (line
6), -r ./requirements.txt (line 7) and numpy==1.17.0 because these package
versions have conflicting dependencies.
```

The conflict is caused by:

```
The user requested numpy==1.17.0
  scipy 1.2.1 depends on numpy>=1.8.2
  matplotlib 3.1.2 depends on numpy>=1.11
  opencv-python 4.3.0.38 depends on numpy>=1.17.3
```

To fix this you could try to:

1. loosen the range of package versions you've specified
2. remove package versions to allow pip attempt to solve the dependency conflict

```
ERROR: ResolutionImpossible: for help visit
https://pip.pypa.io/en/latest/topics/dependency-resolution/#dealing-with-
dependency-conflicts
```

原因是正在安装的numpy版本为1.17.0，但是我们刚才安装的opencv-python 4.3.0.38 要求 numpy>=1.17.3，将requirements.txt中的numpy这一行改为numpy==1.17.3再次尝试

报错三：

还是遇到了错误：

note: This error originates from a subprocess, and is likely not a problem with pip.

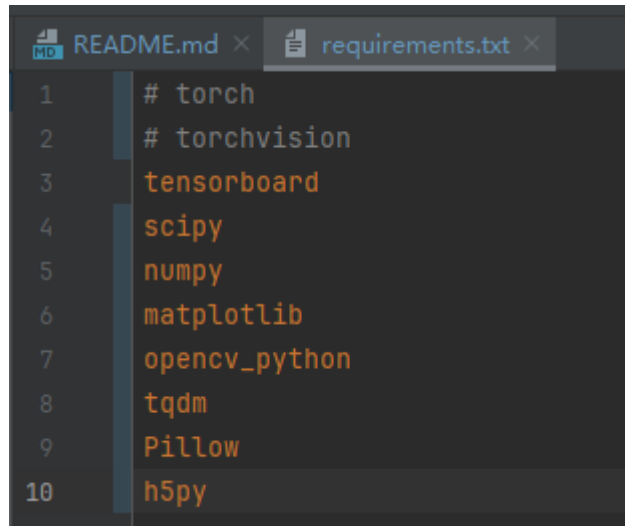
ERROR: Failed building wheel for h5py

Running setup.py clean for h5py

Failed to build scipy matplotlib opencv_python h5py

ERROR: Could not build wheels for scipy, matplotlib, opencv_python, h5py, which is required to install pyproject.toml-based projects

这次只好使出终极杀招，将所有的版本全部删除，让系统自动安装最新的：



```
1 # torch
2 # torchvision
3 tensorboard
4 scipy
5 numpy
6 matplotlib
7 opencv_python
8 tqdm
9 Pillow
10 h5py
```

终于安装成功：

```
Successfully installed MarkupSafe-2.1.5 Pillow-10.3.0 absl-py-2.1.0 colorama-0.4.6 contourpy-1.2.1 cycler-0.12.1 fonttools-4.51.0 grpcio-1.63.0 h5py-3.11.0 importlib-metadata-7.1.0 importlib-resources-6.4.0 kiwisolver-1.4.5 m-arkdown-3.6 matplotlib-3.8.4 numpy-1.26.4 opencv_python-4.9.0.80 packaging-24.0 protobuf-5.26.1 pyparsing-3.1.2 python-dateutil-2.9.0.post0 scipy-1.13.0 six-1.16.0 tensorboard-2.16.2 tensorboard-data-server-0.7.2 tqdm-4.66.4 werkzeug-3.0.3 zipp-3.18.1
```

但是这个方法有一个弊端，原作者使用的包比较老，可能某些函数在新的包里被删除或者修改了，那么在运行代码时需要去修改相应的接口。

2.安装Pytorch

为什么pytorch需要单独安装呢？因为如果我们用上述方法使用清华源或者中科大源去安装pytorch，会默认安装CPU版本的，无法使用显卡进行加速。所以我们要从[官网](#)来安装pytorch。

首先根据cuda版本来选择pytorch，我们之前安装的cuda版本为11.3，因此这里选择v1.10.0，高一点没关系，低了可能会不兼容。

```
v1.10.0

Conda

OSX

# conda
conda install pytorch==1.10.0 torchvision==0.11.0 torchaudio==0.10.0 -c pytorch

Linux and Windows

# CUDA 10.2
conda install pytorch==1.10.0 torchvision==0.11.0 torchaudio==0.10.0 cudatoolkit=10.2 -c pytorch

# CUDA 11.3
conda install pytorch==1.10.0 torchvision==0.11.0 torchaudio==0.10.0 cudatoolkit=11.3 -c pytorch -c conda-forge

# CPU Only
conda install pytorch==1.10.0 torchvision==0.11.0 torchaudio==0.10.0 cpuonly -c pytorch
```

将命令复制到终端运行，并选择Y，开始下载pytorch以及相关的依赖，如果安装比较慢，则请换源或者使用vpn进行加速

```
Downloading and Extracting Packages:
pytorch-1.10.0 | 1.45 GB | #####5 | 65%
cudatoolkit-11.3.1 | 610.3 MB | ##### | 100%
mkl-2024.1.0 | 104.4 MB | ##### | 100%
python-3.9.7 | 20.1 MB | ##### | 100%
torchvision-0.11.0 | 8.8 MB | ##### | 100%
openmpi-3.3.0 | 8.0 MB | ##### | 100%
numpy-1.26.4 | 5.0 MB | ##### | 100%
mkl-devel-2024.1.0 | 5.0 MB | ##### | 100%
libblas-3.9.0 | 5.0 MB | ##### | 100%
libblas-3.9.0 | 4.9 MB | ##### | 100%
liblapack-3.9.0 | 4.9 MB | ##### | 100%
liblapack-3.9.0 | 4.9 MB | ##### | 100%
libmkl-2.9.1 | 2.4 MB | ##### | 100%
torchaudio-0.10.0 | 2.1 MB | ##### | 100%
intel-openmp-2024.1. | 1.5 MB | ##### | 100%
vcrt-10.0.22421.0 | 1.2 MB | ##### | 100%
libtiff-4.3.0 | 1.1 MB | ##### | 100%
zstd-1.5.0 | 1004 KB | ##### | 100%
lcms2-2.12 | 882 KB | ##### | 100%
pillow-8.3.2 | 780 KB | ##### | 100%
mkl-include-2024.1.0 | 768 KB | ##### | 100%
vc14_runtime-14.38.3 | 732 KB | ##### | 100%
... (more hidden) ...
```

最后显示done则安装完成。

检验是否成功：

```
(v8) E:\deep_learning\yolov8-pytorch>
(v8) E:\deep_learning\yolov8-pytorch>
(v8) E:\deep_learning\yolov8-pytorch>python
Python 3.9.7 | packaged by conda-forge | (default, Sep 29 2021, 19:20:16) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
>>>
```

最终输出为True则表明安装成功并且可以调用CUDA。

大功告成，可以开始调试代码了。