

Yolov5环境配置教程

@ powered by Doctor-James

本文章记录本人从零配置yolov5环境，作为一个环境配置教程。

1. Anaconda安装

Python? Anaconda?

装anaconda，就不需要单独装python了

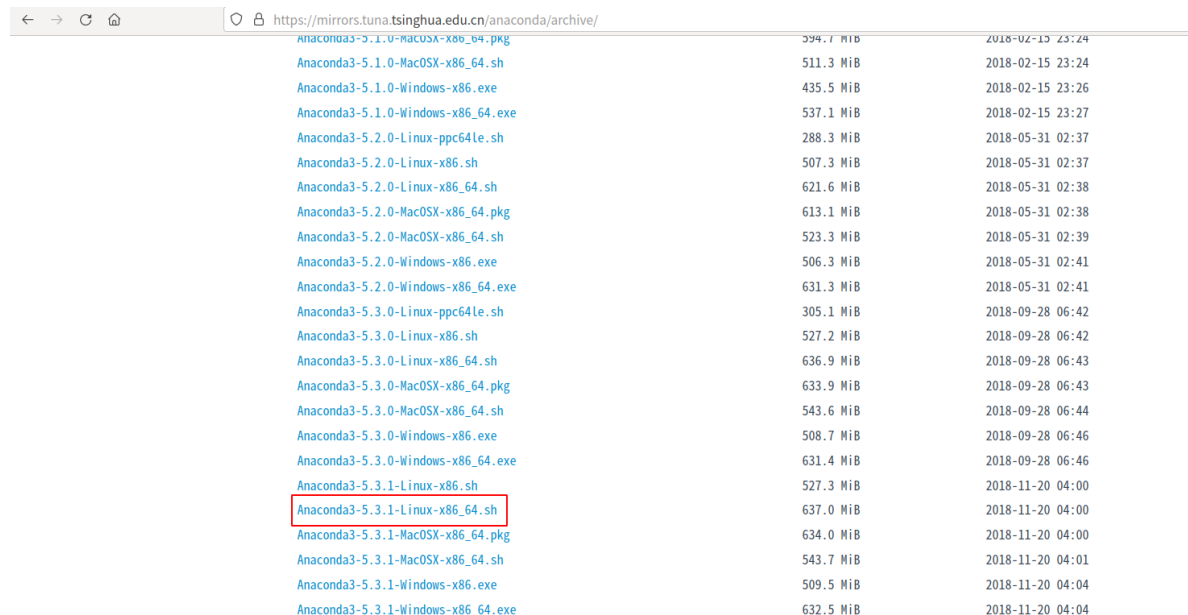
anaconda 是一个python的发行版，包括了python和很多常见的软件库，和一个包管理器conda，可以通过anaconda在电脑上配置多个python环境，方便不同需求

1、anaconda里面集成了很多关于python科学计算的第三方库，主要是安装方便，而python是一个编译器，如果不使用anaconda，那么安装起来会比较痛苦，各个库之间的依赖性就很难连接的很好。

2、常见的科学计算类的库都包含在里面了，使得安装比常规python安装要容易

安装anaconda建议使用清华镜像站，速度较快,但清华镜像站目前只更新到v5.3.1，一些新包用这个版本可能安装不上，所以我们选择先用清华镜像站下载老版本，后续再升级

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



← → ↺ ↻	https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/		
Anaconda3-5.1.0-MacOSX-x86_64.pkg	594.7 MiB	2018-02-15 23:24	
Anaconda3-5.1.0-MacOSX-x86_64.sh	511.3 MiB	2018-02-15 23:24	
Anaconda3-5.1.0-Windows-x86.exe	435.5 MiB	2018-02-15 23:26	
Anaconda3-5.1.0-Windows-x86_64.exe	537.1 MiB	2018-02-15 23:27	
Anaconda3-5.2.0-Linux-ppc64le.sh	288.3 MiB	2018-05-31 02:37	
Anaconda3-5.2.0-Linux-x86.sh	507.3 MiB	2018-05-31 02:37	
Anaconda3-5.2.0-Linux-x86_64.sh	621.6 MiB	2018-05-31 02:38	
Anaconda3-5.2.0-MacOSX-x86_64.pkg	613.1 MiB	2018-05-31 02:38	
Anaconda3-5.2.0-MacOSX-x86_64.sh	523.3 MiB	2018-05-31 02:39	
Anaconda3-5.2.0-Windows-x86.exe	506.3 MiB	2018-05-31 02:41	
Anaconda3-5.2.0-Windows-x86_64.exe	631.3 MiB	2018-05-31 02:41	
Anaconda3-5.3.0-Linux-ppc64le.sh	305.1 MiB	2018-09-28 06:42	
Anaconda3-5.3.0-Linux-x86.sh	527.2 MiB	2018-09-28 06:42	
Anaconda3-5.3.0-Linux-x86_64.sh	636.9 MiB	2018-09-28 06:43	
Anaconda3-5.3.0-MacOSX-x86_64.pkg	633.9 MiB	2018-09-28 06:43	
Anaconda3-5.3.0-MacOSX-x86_64.sh	543.6 MiB	2018-09-28 06:44	
Anaconda3-5.3.0-Windows-x86.exe	508.7 MiB	2018-09-28 06:46	
Anaconda3-5.3.0-Windows-x86_64.exe	631.4 MiB	2018-09-28 06:46	
Anaconda3-5.3.1-Linux-x86.sh	527.3 MiB	2018-11-20 04:00	
Anaconda3-5.3.1-Linux-x86_64.sh	637.0 MiB	2018-11-20 04:00	
Anaconda3-5.3.1-MacOSX-x86_64.pkg	634.0 MiB	2018-11-20 04:00	
Anaconda3-5.3.1-MacOSX-x86_64.sh	543.7 MiB	2018-11-20 04:01	
Anaconda3-5.3.1-Windows-x86.exe	509.5 MiB	2018-11-20 04:04	
Anaconda3-5.3.1-Windows-x86_64.exe	632.5 MiB	2018-11-20 04:04	

下载完成后，运行安装脚本

```
bash Anaconda3-5.2.0-Linux-x86_64.sh
```

安装过程中除了问是否安装VSCode选no，其他均选yes或按Enter即可

安装过程中会自动配置环境变量，所以安装完成后只需执行以下命令即可

```
source ~/.bashrc
```

最后检验是否安装好

```
conda --version  
conda list
```

```
zjl@zjl-NH5x-7xEDx-RCx-RDx:~$ conda --version  
conda 4.5.11  
zjl@zjl-NH5x-7xEDx-RCx-RDx:~$ conda list  
# packages in environment at /home/zjl/anaconda3:  
#  
# Name                                Version                                Build      Channel  
_ipyw_jlab_nb_ext_conf                0.1.0                                py37_0  
alabaster                              0.7.11                              py37_0  
anaconda                              5.3.1                                py37_0  
anaconda-client                       1.7.2                                py37_0  
anaconda-navigator                    1.9.2                                py37_0  
anaconda-project                      0.8.2                                py37_0  
appdirs                               1.4.3                                py37h28b3542_0  
asn1crypto                            0.24.0                              py37_0  
astroid                               2.0.4                                py37_0  
astropy                               3.0.4                                py37h14c3975_0  
atomicwrites                          1.2.1                                py37_0  
attrs                                  18.2.0                              py37h28b3542_0  
automat                               0.7.0                                py37_0  
babel                                  2.6.0                                py37_0  
backcall                              0.1.0                                py37_0  
backports                             1.0                                  py37_1  
backports.shutil_get_terminal_size    1.0.0                                py37_2  
beautifulsoup4                        4.6.3                                py37_0
```

检测python环境，发现Anaconda已经为我们安装好了

```
python
```

```
zjl@zjl-NH5x-7xEDx-RCx-RDx:~$ python  
Python 3.7.0 (default, Jun 28 2018, 13:15:42)  
[GCC 7.2.0] :: Anaconda, Inc. on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> □
```

最后进行Anaconda的升级,会自动升级到最新版本

```
conda upgrade -n base -c defaults --override-channels conda
```

2. 确定需求

首先在github上面git clone Yolov5源码，打开其中requirements.txt文件，里面详细写了需要配置的环境版本

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

```
# Base -----  
matplotlib>=3.2.2  
numpy>=1.18.5  
opencv-python>=4.1.2  
Pillow>=7.1.2
```

```
PyYAML>=5.3.1
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0

# Logging -----
tensorboard>=2.4.1
# wandb

# Plotting -----
pandas
seaborn>=0.11.0

# Export -----
# coremltools>=4.1 # CoreML export
# onnx>=1.9.0 # ONNX export
# onnx-simplifier>=0.3.6 # ONNX simplifier
# scikit-learn==0.19.2 # CoreML quantization
# tensorflow>=2.4.1 # TFLite export
# tensorflowjs>=3.9.0 # TF.js export

# Extras -----
# albumentations>=1.0.3
# Cython # for pycocotools https://github.com/cocodataset/cocoapi/issues/172
# pycocotools>=2.0 # COCO mAP
# roboflow
thop # FLOPs computation
```

其中大部分环境都可以用 `pip install -r requirements.txt`

其中 `torch` 和 `torchvision` 建议自己手动安装（将此文件中 `torch` 和 `torchvision` 部分删掉）

为什么我们需要手动安装 `torch` 和 `torchvision` 呢，是因为实际上这里要求的 `torch` 是大于 1.7.0，而直接执行这个安装命令会安装为 1.7.0，导致程序不能运行，所以需要手动安装更高版本

确定了 `torch` 版本，还需要确定对应的 **显卡驱动**，**CUDA**，**CUDNN** 版本，可在 PyTorch 官网查询对应的版本

<https://pytorch.org/get-started/previous-versions/> //PyTorch 官网

此处我选择了安装 **pytorch v1.8.0**，对应 **CUDA 11.1**，**Linux x86_64 显卡驱动版本 >=455.23**

3. 配置环境

3.1 安装显卡驱动

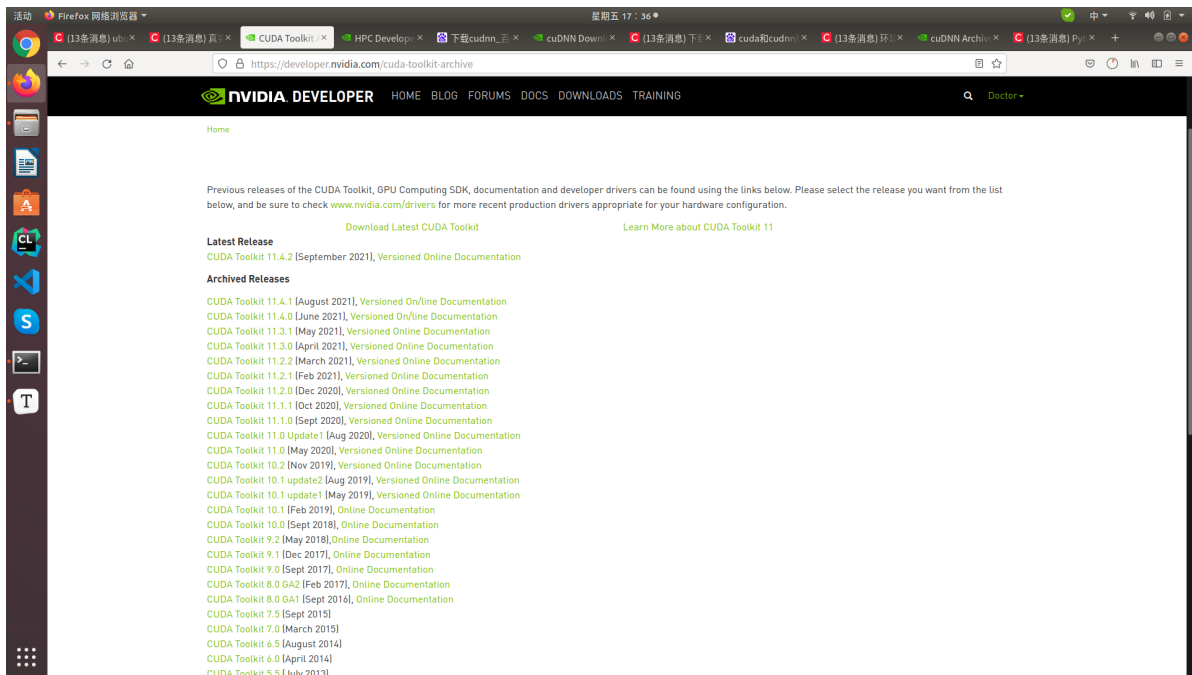
详细教程见此篇知乎

<https://zhuanlan.zhihu.com/p/59618999>

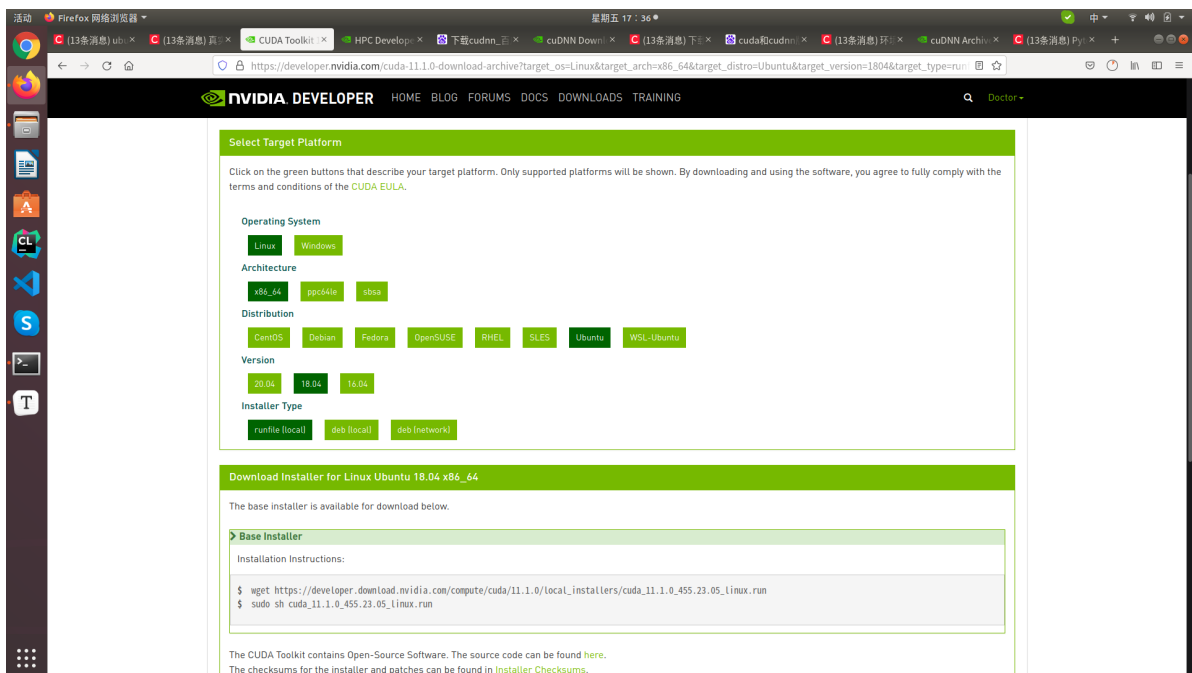
3.2 安装CUDA&cuDNN

- 进入 NVIDIA 官网下载 CUDA

<https://developer.nvidia.com/cuda-toolkit-archive>



选择合适的版本之后（我这里选择的CUDA 11.1），选择如下图



下载完成后运行脚本

```
sudo sh cuda_11.1.0_455.23.05_linux.run
```

安装完毕之后，将以下两条加入 `.bashrc` 文件中

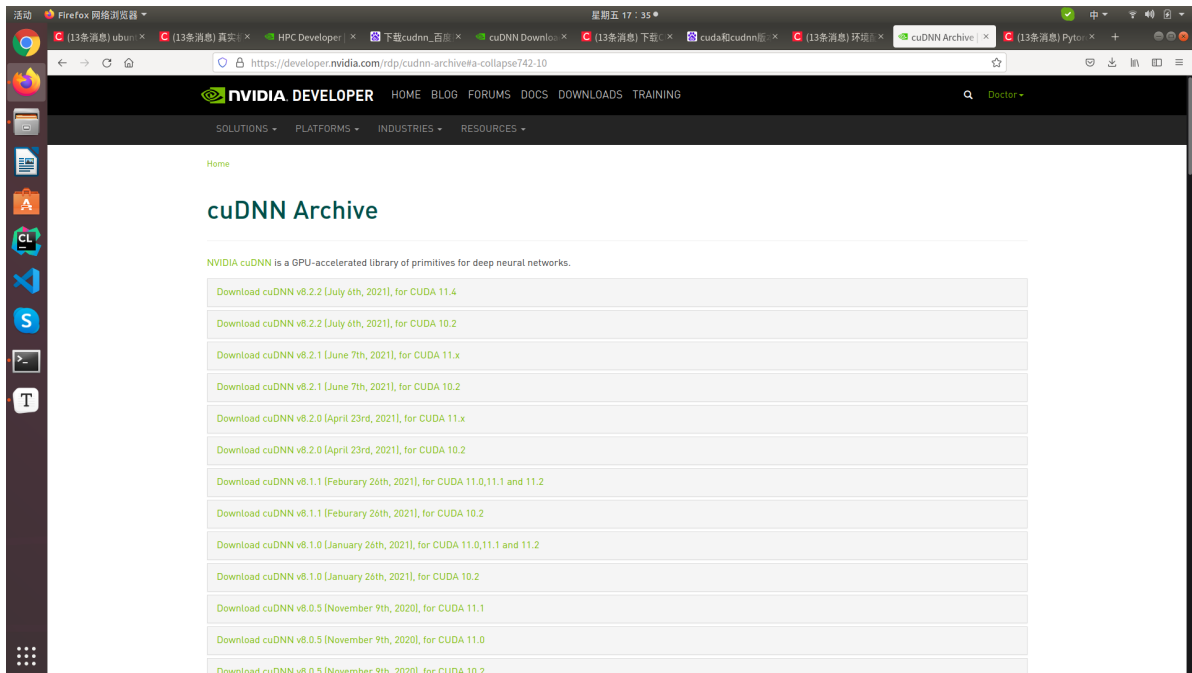
```
sudo vim ~/.bashrc
```

```
export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}} #注意，根据自己的版本，修改cuda-11.1...
export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}} #注意，根据自己的版本，修改cuda-11.1
```

到此为止cuda就安装好了

- 进入NVIDIA官网下载cuDNN（需要注册登陆）

<https://developer.nvidia.com/rdp/cudnn-archive#a-collapse742-10>



选择适配CUDA版本的cuDNN，我这里选择的是cuDNN v8.0.5

下载下来之后解压

接着复制cuDNN内容到cuda相关文件夹内

```
sudo cp cuda/include/cudnn.h /usr/local/cuda/include #注意，解压后的文件夹名称为cuda,将对应文件复制到 /usr/local中的cuda内
sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*
```

由于libcudnn*较大，笔者这里遇到了根目录内存不足的问题，遂使用软连接的方法，避免内存重复占用命令为

```
ln -s source/* target/
```

如我把/home/zjl/Downloads/cudnn-11.1-linux-x64-v8.0.5.39/cuda/lib64下所有文件都连接到/usr/local/cuda/lib64/中：

```
sudo ln -s /home/zjl/Downloads/cudnn-11.1-linux-x64-v8.0.5.39/cuda/lib64/* /usr/local/cuda/lib64/
```

到此处，CUDA和cuDNN的安装就完成了。

可运行NVIDIA_CUDA-11.1_Samples里面的demo检验一下安装

```
cd /usr/local/cuda/samples/1_Uutilities/deviceQuery #由自己电脑目录决定
make
sudo ./deviceQuery
```

```
活动 终端
zjl@zjl-NH5x-7xEDx-RCx-RDx: ~/NVIDIA_CUDA-11.1_Samples/1_Utils/DeviceQuery$ ./deviceQuery
CUDA Device Query (Runtime API) version (CUDA static linking)
Detected 1 CUDA Capable device(s)
Device 0: "GeForce GTX 1660 Ti"
  CUDA Driver Version / Runtime Version      11.2 / 11.1
  CUDA Capability Major/Minor version number:  7.5
  Total amount of global memory:              5945 Mbytes (6233391104 bytes)
  (24) Multiprocessors, ( 64) CUDA Cores/MP:  1536 CUDA Cores
  GPU Max Clock rate:                        1590 MHz (1.59 GHz)
  Memory Clock rate:                         6001 MHz
  Memory Bus Width:                           192-bit
  L2 Cache Size:                             1572864 bytes
  Maximum Texture Dimension Size (x,y,z)      1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:             65536 bytes
  Total amount of shared memory per block:    49152 bytes
  Total shared memory per multiprocessor:     65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                   32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:         1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z):   (2147483647, 65535, 65535)
  Texture alignment:                           512 bytes
  Concurrent copy and kernel execution:        Yes with 3 copy engine(s)
  Run time limit on kernels:                   Yes
  Integrated GPU sharing Host Memory:           No
  Support host page-locked memory mapping:      Yes
  Alignment requirement for Surfaces:           Yes
  Device has ECC support:                       Disabled
  Device supports Unified Addressing (UVA):     Yes
  Device supports Managed Memory:              Yes
  Device supports Compute Preemption:          Yes
  Supports Cooperative Kernel Launch:          Yes
  Supports MultiDevice Co-op Kernel Launch:    Yes
  Device PCI Domain ID / Bus ID / Location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple threads can use ::cudaSetDevice()) with device simultaneously >
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.2, CUDA Runtime Version = 11.1, NumDevs = 1
Result = PASS
zjl@zjl-NH5x-7xEDx-RCx-RDx:~/NVIDIA_CUDA-11.1_Samples/1_Utils/DeviceQuery$
```

出现上述信息，说明cuda配置正确

3.3 PyTorch环境

首先conda添加清华源，下载速度会比较快

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
```

创建一个新的虚拟环境,并命名为yolov5（命名随意）

```
conda create -n yolov5 python==3.7
source activate yolov5
```

安装PyTorch，torchvision

```
conda install pytorch==1.8.0 torchvision==0.9.0
```

最后验证pytorch和torchvision是否安装好

```
python
import torch
torch.__version__
import torchvision
torchvision.__version__
```

```
(yolov5) zjl@zjl-NH5x-7xEDx-RCx-RDx:~$ python
Python 3.7.0 (default, Oct 9 2018, 10:31:47)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.__version__
'1.8.0'
>>> import torchvision
>>> torchvision.__version__
'0.9.0'
```

3.4 安装其他包

在自己下载的yolov5源码目录下（我的是/home/zjl/code/DNN/yolov5）打开终端，注意，此时打开终端系统默认的环境是base环境，base环境是安装anaconda时候conda自动配置的

```
(base) zjl@zjl-NH5x-7xEDx-RCx-RDx:~/code/DNN/yolov5$
```

而之前我们安装的python, pytorch, torchvision都是在自创的虚拟环境yolov5中的，所以我们首先需要的是切换到我们自创的虚拟环境中，接着执行安装命令

注意：由于我们之前自己手动安装了pytorch和torchvision，所以执行安装命令之前要将requirements.txt中的**torch>=1.7.0**，**torchvision>=0.8.1**删除掉

```
source activate yolov5
pip install -r requirements.txt
```

到此为之我们所有的环境都已经安装好了

4. 运行demo

在yolov5文件夹下执行以下命令测试是否安装完毕

```
python detect.py --source data/images/ --weights yolov5s.pt --conf 0.4
```

执行这个命令会自动在官网下载yolov5s.pt文件，如果执行命令下载失败，可以手动到官网下载

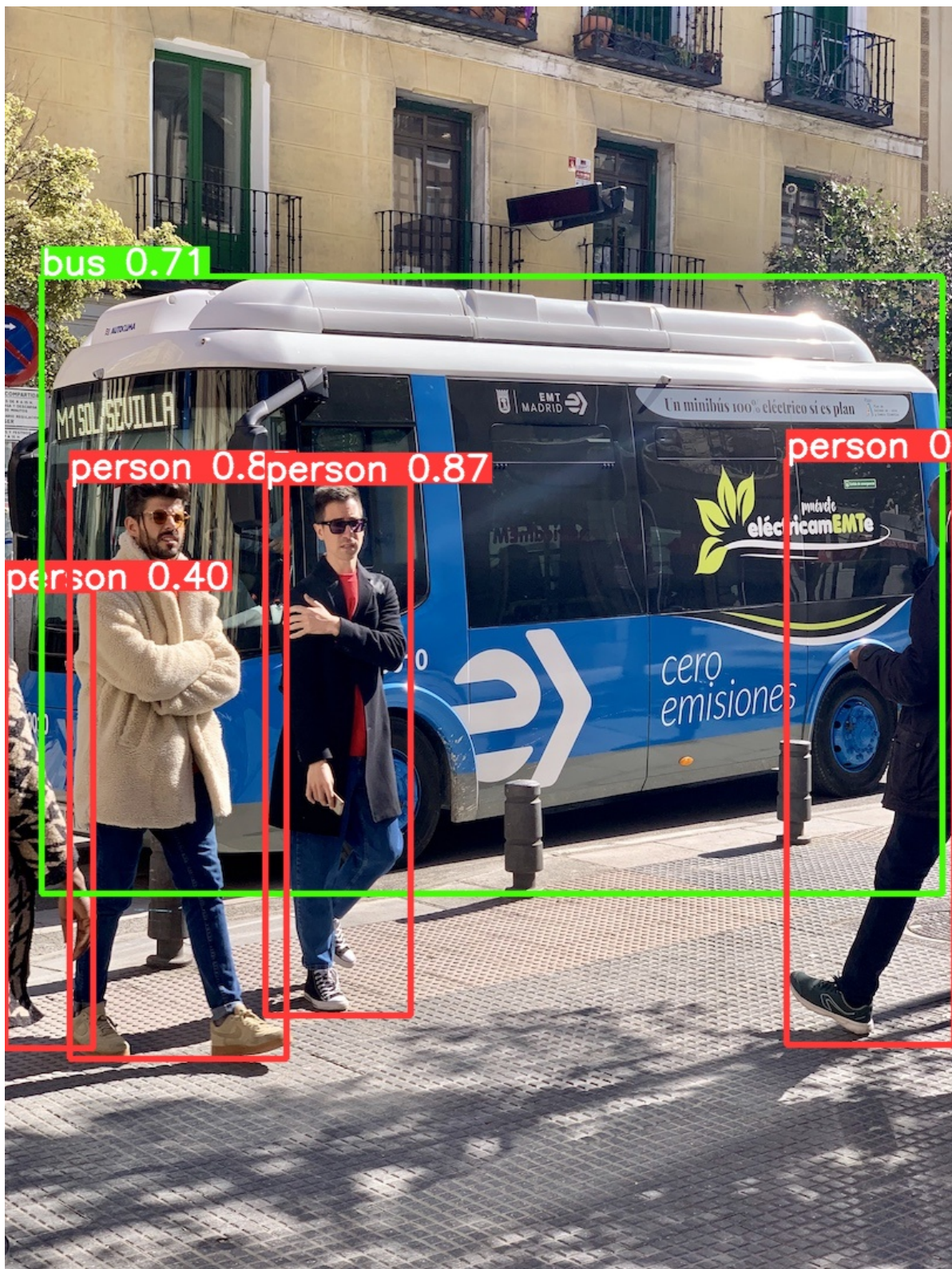
```
https://github.com/ultralytics/yolov5/releases/tag/v4.0
```

最后如果显示如下，那么恭喜你yolov5安装成功啦！！（也是恭喜我自己。。。）

```
Results saved to runs/detect/exp4
(yolov5) zjl@zjl-NH5x-7xEDx-RCx-RDx:~/code/DNN/yolov5$ python detect.py --source
data/images/ --weights yolov5s.pt --conf 0.4
detect: weights=['yolov5s.pt'], source=data/images/, imgsz=[640, 640], conf_thre
s=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, sa
ve_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False,
augment=False, visualize=False, update=False, project=runs/detect, name=exp, exi
st_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False
YOLOv5 🚀 2021-10-1 torch 1.8.0 CUDA:0 (GeForce GTX 1660 Ti, 5944.625MB)

Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 0 gradients
image 1/2 /home/zjl/code/DNN/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bu
s, Done. (0.009s)
image 2/2 /home/zjl/code/DNN/yolov5/data/images/zidane.jpg: 384x640 2 persons, 1
tie, Done. (0.008s)
Speed: 0.4ms pre-process, 8.5ms inference, 0.8ms NMS per image at shape (1, 3, 6
40, 640)
Results saved to runs/detect/exp4
```

demo执行结果如图



5. 结语

众所周知，深度学习配置环境一直是一个很搞心态的事情，但其实只要你心平气和，首先确定自己需要什么，再一步一步的配置，整个配置流程逻辑是很清晰的。

这也是笔者第一次配置这个环境，整个流程用时仅2个多小时。

希望大家以后做事也可以这样首先明确自己的目标，再落到实地，心平气和的一步一步的探索，配置过程中也许会遇到本教程没有出现的问题，希望大家可以静下心来排查问题，这样才可以起到事半功倍的效果。

