

Jetson NX 配置 GPU 版本 PyTorch 的 yolov8 教程

作者也是一个热爱学习的人，在深度学习模型部署这一部分在进行一定的研究。本博客的安装教程最原始的来源于链接：

<https://www.bilibili.com/opus/825590106993721440>

我在看了其的教程后，在多次实践、操作、练习后，终于是做通了。我在其基础上将自己遇到的新的问题进行了整理，同时一镜到底录制了刷机到安装配置环境的全部教程。

如果你是初学者，请务必保证所有版本号和本文的一致！本文写于 2025 年 6 月，各部分版本号如下：

Ubuntu20.04

Jetpack5.1.3

Torch2.0.0+nv23.05

Torchvision0.15.1a0+42759b1

1 前言

1.1 前言——来自之前的作者

前段时间做无人机集群目标跟踪的比赛，需要在 5 台英伟达的 jetson NX 上都配置 yolov8 环境。我们首先配好了 CPU 版本的 torch 环境，虽然配环境十分轻松，但 CPU 版本 torch 的 yolov8 运行帧率只有 1~2 帧。

而在配置 GPU 版本 torch 的 yolov8 环境的过程非常坎坷，但经过一番摸索，在 5 台 NX 上都非常顺畅地配置完成了 yolov8 环境的配置。故把踩过的坑在此记录，分享给大家以供交流。

另外，我之前对 jetpack, torchvision, pycuda, tensorRT 等理解不是很清晰，很多时候配好了都不知道这东西是用来干啥的。对此，本文每配一个东西我也会说明其概念和作用。

本篇教程不会把需要运行的命令一条一条全部列出来，取而代之的是更多的文字描述与原理解读。我自认为逻辑非常清晰，读者跟随逻辑完全可以完成安装，读者只需要带点脑子即可。也许很多读者配环境的时候不看任何文字说明，把作者给出的指令逐行复制了事（我之前就是这样）。这个习惯很不好。作者会在文字说明中陈述很多细节，请认真读。

注意：如果你需要在 NX 上安装 ros 或 ros2，那么不要使用 conda!!! ros 与 conda 的兼容性十分糟糕，亲测 ros2 与 conda 兼容性更差，甚至连编译都过不去。请在 NX 原生 python 环境下安装 yolov8 环境。

观看本教程之前，最好满足以下条件：

1. 你的 NX 已经安装好了 ubuntu20.04 系统
2. 熟悉一些基础的 linux 命令
3. 会 python 编程
4. 了解 torch 的基本概念，在笔记本电脑上能够完成 torch 的配环境工作（但对于如何在 NX 上完成环境配置比较头疼）
5. 有一个 python3.8 环境，最好是 python3.8.10（ubuntu20.04 默认自带 python3.8）
6. 安装好了 git, pip 等基础工具
7. 最好有一个梯子，可以连接到外网。（是当时可以大大缩减安装时间）

2 刷机——请看视频。

3 安装深度学习依赖库文件

3.1 安装 Jetpack

Jetpack 是专供英伟达的嵌入式计算平台使用的人工智能包。这个官方定义现在理解起来有点抽象，咱们安装好之后，大家就知道他是什么了。

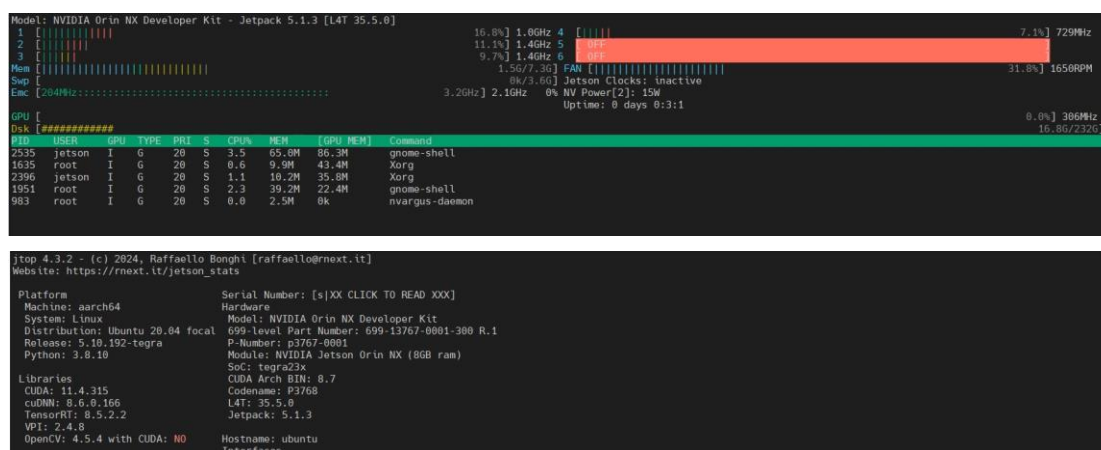
首先安装 `jtop`，这是一个监控 CPU，GPU 等使用情况的工具。

`sudo pip install jetson-stats`

然后安装 JetPack:

`sudo apt install nvidia-jetpack`

安装完 JetPack 后，命令行输入 `jtop` 并运行，即可看到当前电脑的 CPU 运行状态，按数字键可以切换页面，切换到 INFO 页面，可看到已经安装好的包：



可以发现，当前你的 NX 已经安装好了很多难装的底层库：Cuda，cuDNN，TensorRT，OpenCV。这下可能你大致明白了咱们的定义：

Jetpack 是英伟达提供的专门供他自己的嵌入式计算平台使用的人工智能包。Jetpack 把人工智能开发常用的底层驱动和库一股脑给你打包好，你安装了 Jetpack，就把这几样东西都安装上去了。

注意：上图 `opencv` 的版本后有“with CUDA NO”的字样，说明 `opencv` 也有支持 GPU 加速的版本，但是默认安装的 `opencv` 不支持 GPU 加速（`pip` 也只能安装 `cpu` 版本的 `opencv`）。由于作者的项目不涉及太多的 `opencv` 操作，配置 GPU 版本的 `opencv` 对整体性能影响不大，所以作者没有深入研究，如果需要安装支持 GPU 加速的 `opencv`，需要将原 `opencv` 卸载，并通过源码编译安装，在 `cmake` 阶段指定相应 `cuda` 配置，即可编译出支持 `cuda` 加速的 `opencv`。读者可自行百度解决。后文不会再提及 `cuda`，`cuDNN`，`opencv` 的安装。

3.2 安装 torch

torch 应该不需要介绍了,为了方便,一般大家都把 pytorch 直接叫做 torch,初学者看见不要觉得奇怪就行。安装 torch,参考官方链接,所有命令均参考官网的即可。

网址:<https://docs.nvidia.com/deeplearning/frameworks/install-pytorch-jetson-platform/index.html>

来到:“2.Prerequisites and Installation” 部分

第一步:

```
sudo apt-get -y update
```

```
sudo apt-get install -y python3-pip libopenblas-dev
```

第二步:

```
export TORCH_INSTALL=https://developer.download.nvidia.cn/compute/redist/jp/v511/pytorch/torch-2.0.0+nv23.05-cp38-cp38-linux_aarch64.whl
```

第三步:安装 PyTorch

```
python3 -m pip install --upgrade pip; python3 -m pip install numpy==1.26.1; python3 -m pip install --no-cache $TORCH_INSTALL
```

注意:(英文的引号!同时安装完成后可能会报:numpy 这个包版本兼容的问题,具体请看视频。答案先扔到这,按照上面的命令进行安装,等它报错了,再重新安装 python3 -m pip install numpy==1.24.4)。

等待 torch 安装完成——Torch2.0.0+nv23.05。

3.3 安装 torchvision

torchvision 是 torch 的一部分,可以理解为用 torch 实现了一份常用的基础的网络框架和工具类,你拿来就能用,不用自己写了。每个版本的 torch 都会有自己对应版本的 torchvision,一般装错版本就意味着不好使。torchvision 版本对应关系如下:

博客园: <https://www.cnblogs.com/phillee/p/18599125>

torch	torchvision	Python
main / nightly	main / nightly	>=3.9 , <=3.12
2.5	0.20	>=3.9 , <=3.12
2.4	0.19	>=3.8 , <=3.12
2.3	0.18	>=3.8 , <=3.12
2.2	0.17	>=3.8 , <=3.11
2.1	0.16	>=3.8 , <=3.11
2.0	0.15	>=3.8 , <=3.11

▼ older versions

torch	torchvision	Python
1.13	0.14	>=3.7.2 , <=3.10
1.12	0.13	>=3.7 , <=3.10
1.11	0.12	>=3.7 , <=3.10
1.10	0.11	>=3.6 , <=3.9
1.9	0.10	>=3.6 , <=3.9
1.8	0.9	>=3.6 , <=3.9
1.7	0.8	>=3.6 , <=3.9
1.6	0.7	>=3.6 , <=3.8
1.5	0.6	>=3.5 , <=3.8
1.4	0.5	==2.7 , >=3.5 , <=3.8
1.3	0.4.2 / 0.4.3	==2.7 , >=3.5 , <=3.7
1.2	0.4.1	==2.7 , >=3.5 , <=3.7
1.1	0.3	==2.7 , >=3.5 , <=3.7
<=1.0	0.2	==2.7 , >=3.5 , <=3.7

通过 pip 安装的 torchvision 一般都是 CPU 版本的，版本只显示 0.15.1。这个是无法调用 jetson 底层的 cuda 加速的。GPU 版本的 torchvision 只能通过编译进

行安装。编译安装的 torch 版本号显示为“0.15.1a0+42759b1”（“版本号+一串数字英文”），说明你安装的是支持 jetson 上的 cuda 加速的。

下载 torchvision 源码并构建安装的方法如下：

```
git clone --branch v0.15.1 https://github.com/pytorch/vision torchvision
cd torchvision
python3 setup.py install --user
```

Nvidia Jetson Orin NX 很可能编译了一半就内存爆了然后卡死，可重复运行安装脚本，卡死之前的安装进度会被保留，多运行几次就可以安装成功了。

安装成功后输入：

```
python3
进入编译器；
import torch
import torchvision
torch.__version__
torchvision.__version__
torch.cuda.is_available()
```

torch.__version__ 和 torchvision.__version__ 均会输出“版本号+一串数字”，如下图所示，这就说明你安装 gpu 版本的 torch 和 torchvision 成功了。如果你是初学者，请务必保证所有版本号和本文的一致。

```
jetson@ubuntu:~/torchvision$ python3
Python 3.8.10 (default, Mar 18 2025, 20:04:55)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
  File "<stdin>", line 1
    import torch
    ^
SyntaxError: invalid syntax
>>> torch.__version__
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'torch' is not defined
>>> import torch
>>> torch.__version__
'2.0.0+nv23.05'
>>> import torchvision
/home/jetson/torchvision/torchvision/io/image.py:13: UserWarning: Failed
to load C++ extension. Please ignore this warning. Otherwise, there might be something wrong with your
installation.
  warn(
/home/jetson/torchvision/torchvision/__init__.py:25: UserWarning: You are
using torchvision from the source directory. This may give errors. Please exit the torchvision project source and relaunc
h the installation.
  warnings.warn(message.format(os.getcwd()))
>>> torchvision.__version__
'0.15.1a0+42759b1'
>>> torch.cuda.is_available()
True
```

3.4 安装 pyCUDA

pyCUDA 是一个 python 库，让访问 NVIDIA 的 CUDA 并行计算 API 更容易。我的理解是：如果你是初学者，只是使用 yolov8 做基础的训练和预测，是不需要管 python 如何访问 cuda 的，这些东西 torch 都帮你做好了。而如果你使用 torch 实现自己的网络架构，或者需要使用到 tensorRT 做加速，可能需要用到 pyCUDA 库做一些底层的操作。总之，对于初学者，不太需要关心这个库。本文一并安装，有备无患。

pip3 install Cython

pip3 install pycuda -user

如果下载速度极慢，可以选择添加清华源进行加速。

说到 TensorRT 加速，其运行速度比 GPU 版本的 torch 又快了一个数量级，但是水很深，建议新手不要尝试。CSDN 上的文章写的乱七八糟的，我研究很长时间连个 demo 都没跑通，对于新手来说，GPU 版本的 torch 帧率一般已经够高了。本视频是在研究很久的情况下才给出的相关教程——先从 pt 文件到 onnxruntime 进行加速推理。等时机成熟了再向 Tensor RT 进军。

3.5 安装 yolov8

onnx 也是 python 的一个库，可以将 torch 的网络模型打包成一种通用的网络模型格式，方便其他神经网络框架直接调用网络模型和网络参数进行 forward 前向预测。ultralytics 库就是 yolov8 的库了，ultralytics 是发布 yolov8 的公司的名字。

pip install onnx==1.4.1

pip install ultralytics

如果下载速度极慢，可以选择添加清华源进行加速。

注意：原作者在这里遇到了很多坑，但是本人刷机三次，没有遇到过一次下述情况，因此我也将这部分情况放出来，仅供大家参考。

安装完 ultralytics 之后，pip 会报错：pandas 和 matplotlib 要求的 numpy 版本过高，如下图：

```
Stored in directory: /home/zerone/.cache/pip/wheels/11/92/62/05b14ac00212161d31afed082245377b2ce000c20307c0002
Successfully built psutil
ERROR: pandas 2.0.3 has requirement numpy>=1.20.3; python_version < "3.10", but you'll have numpy 1.19.4 which is incompatible.
ERROR: matplotlib 3.7.2 has requirement numpy>=1.20, but you'll have numpy 1.19.4 which is incompatible.
Installing collected packages: tzdata, python-dateutil, pytz, pandas, contourpy, pillow, fonttools, zipp, importlib-resources, packaging, matplotlib, seaborn, tqdm, opencv-python, py-cpuinfo, psutil, requests, ultralytics
Successfully installed contourpy-1.1.0 fonttools-4.41.1 importlib-resources-6.0.0 matplotlib-3.7.2 opencv-python-4.8.0.74 packaging-23.1 pandas-2.0.3 pillow-10.0.0 psutil-5.9.5 py-cpuinfo-9.0.0 python-dateutil-2.8.2 pytz-2023.3 requests-2.31.0 seaborn-0.12.2 tqdm-4.65.0 tzdata-2023.3 ultralytics-8.0.143 zipp-3.16.2
zerone@zerone-nx-004:~/Developer/yolo_test$
```

注意，这是个大坑，如果不处理好 numpy 版本问题，可能即使 torch 和 torchvision 的版本完全正确，yolo 运行还是会报错。而解决这个问题之后，一般

就好使了。此时把他们都卸了即可：

pip uninstall pandas

pip uninstall matplotlib

当然，如果你有别的需求真的需要用到这两个库，你可以重新装一下这两个库的较低版本，具体版本要求可参考 [github](#) 上 **ultralytics** 官方仓库的 **pip** 依赖文件。由于我没有遇到过，这里也没有能够分享出来的。

3.6 安装 onnxruntime-gpu 版本

从网址：

https://elinux.org/Jetson_Zoo#ONNX_Runtime

可以知道，本版本的 jetpack 的版本是 5.1.3；python 版本是 3.8.10，同时可以检索到对应的 onnxruntime-gpu 版本是 1.17.0，同时 1.18.0 也可以，但这里我选择的是 onnxruntime 1.17.0。

通过点击 “[pip wheel](#)” 就可以直接在 windows 上进行下载，下载完成之后可以直接通过 U 盘拷贝到 NX 的系统中。

cd 到 onnxruntime 1.17.0 的文件夹中，打开命令行终端输入：

`python3 -m pip install onnxruntime_gpu-1.17.0-cp38-cp38-linux_aarch64.whl`

然后安装成功。不出意外的话会出意外的。

由于 jetpack5.1.3 版本系统自带的一个名叫：“GLIBCXX” 的版本最高为：“GLIBCXX_3.4.28”。

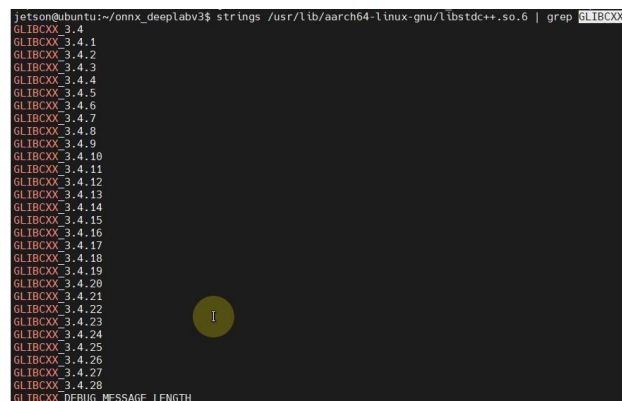
而 onnxruntime_gpu-1.17.0-cp38-cp38-linux_aarch64.whl 需要的“GLIBCXX”最低为 “GLIBCXX_3.4.29”。这也是属于包之间的冲突。

要解决这个问题。

第一步：查看自己的“GLIBCXX” 包含的版本。

`strings /usr/lib/aarch64-linux-gnu/libstdc++.so.6 | grep GLIBCXX`

我的结果如下：



```
jetson@ubuntu:~/onnx_deeplabv3$ strings /usr/lib/aarch64-linux-gnu/libstdc++.so.6 | grep GLIBCXX
GLIBCXX_3.4
GLIBCXX_3.4.1
GLIBCXX_3.4.2
GLIBCXX_3.4.3
GLIBCXX_3.4.4
GLIBCXX_3.4.5
GLIBCXX_3.4.6
GLIBCXX_3.4.7
GLIBCXX_3.4.8
GLIBCXX_3.4.9
GLIBCXX_3.4.10
GLIBCXX_3.4.11
GLIBCXX_3.4.12
GLIBCXX_3.4.13
GLIBCXX_3.4.14
GLIBCXX_3.4.15
GLIBCXX_3.4.16
GLIBCXX_3.4.17
GLIBCXX_3.4.18
GLIBCXX_3.4.19
GLIBCXX_3.4.20
GLIBCXX_3.4.21
GLIBCXX_3.4.22
GLIBCXX_3.4.23
GLIBCXX_3.4.24
GLIBCXX_3.4.25
GLIBCXX_3.4.26
GLIBCXX_3.4.27
GLIBCXX_3.4.28
GLIBCXX_DEBUG_MESSAGE_LENGTH
```


需要下载乃至更高版本的“GLIBCXX”来适配 onnxruntime_gpu-1.17.0。

第二步：尝试通过 APT 安装最新的 libstdc++6：

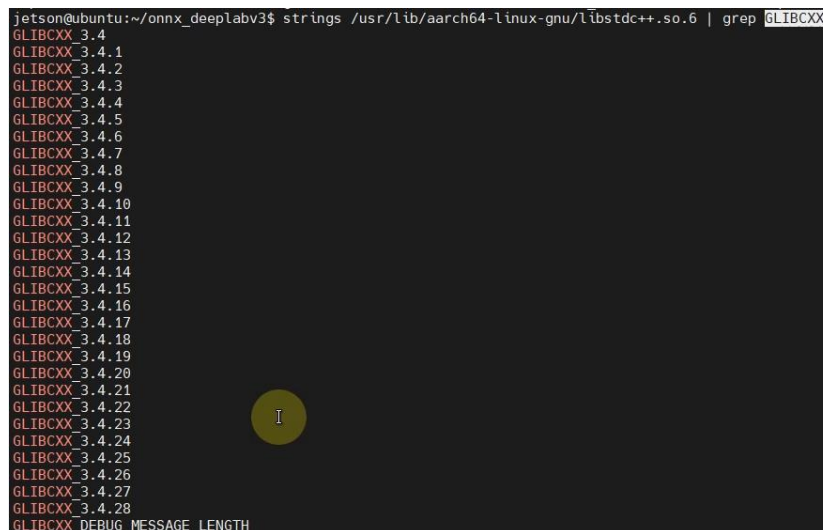
sudo apt update

sudo apt install libstdc++6

然后再输入命令：

strings /usr/lib/aarch64-linux-gnu/libstdc++.so.6 | grep GLIBCXX

应该还是没有变：



```
jetson@ubuntu:~/onnx_deeplabv3$ strings /usr/lib/aarch64-linux-gnu/libstdc++.so.6 | grep GLIBCXX
GLIBCXX_3.4
GLIBCXX_3.4.1
GLIBCXX_3.4.2
GLIBCXX_3.4.3
GLIBCXX_3.4.4
GLIBCXX_3.4.5
GLIBCXX_3.4.6
GLIBCXX_3.4.7
GLIBCXX_3.4.8
GLIBCXX_3.4.9
GLIBCXX_3.4.10
GLIBCXX_3.4.11
GLIBCXX_3.4.12
GLIBCXX_3.4.13
GLIBCXX_3.4.14
GLIBCXX_3.4.15
GLIBCXX_3.4.16
GLIBCXX_3.4.17
GLIBCXX_3.4.18
GLIBCXX_3.4.19
GLIBCXX_3.4.20
GLIBCXX_3.4.21
GLIBCXX_3.4.22
GLIBCXX_3.4.23
GLIBCXX_3.4.24
GLIBCXX_3.4.25
GLIBCXX_3.4.26
GLIBCXX_3.4.27
GLIBCXX_3.4.28
GLIBCXX_DEBUG_MESSAGE_LENGTH
```

再然后只有直接进行有风险的操作：通过“Ubuntu Toolchain”PPA 拉取更高版本的 libstdc++6：

Jetson Orin NX 虽然运行的是 Ubuntu 20.04，但默认源里没有 GCC 11 的库。我们可以临时加一个 Ubuntu Toolchain 测试源，让它带来新版本：

sudo apt install software-properties-common

sudo add-apt-repository ppa:ubuntu-toolchain-r/test

sudo apt update

sudo apt install libstdc++6

安装完成后，再次验证：

strings /usr/lib/aarch64-linux-gnu/libstdc++.so.6 | grep GLIBCXX

如果能够看到“GLIBCXX_3.4.29”，则说明系统库已经升级到了至少 GCC 11 的 libstdc++。此时直接运行用.onnx 模型进行推理。

4 总结

任重道远。不喜勿喷。

欢迎交流，共同进步。

制作不易，感谢支持。