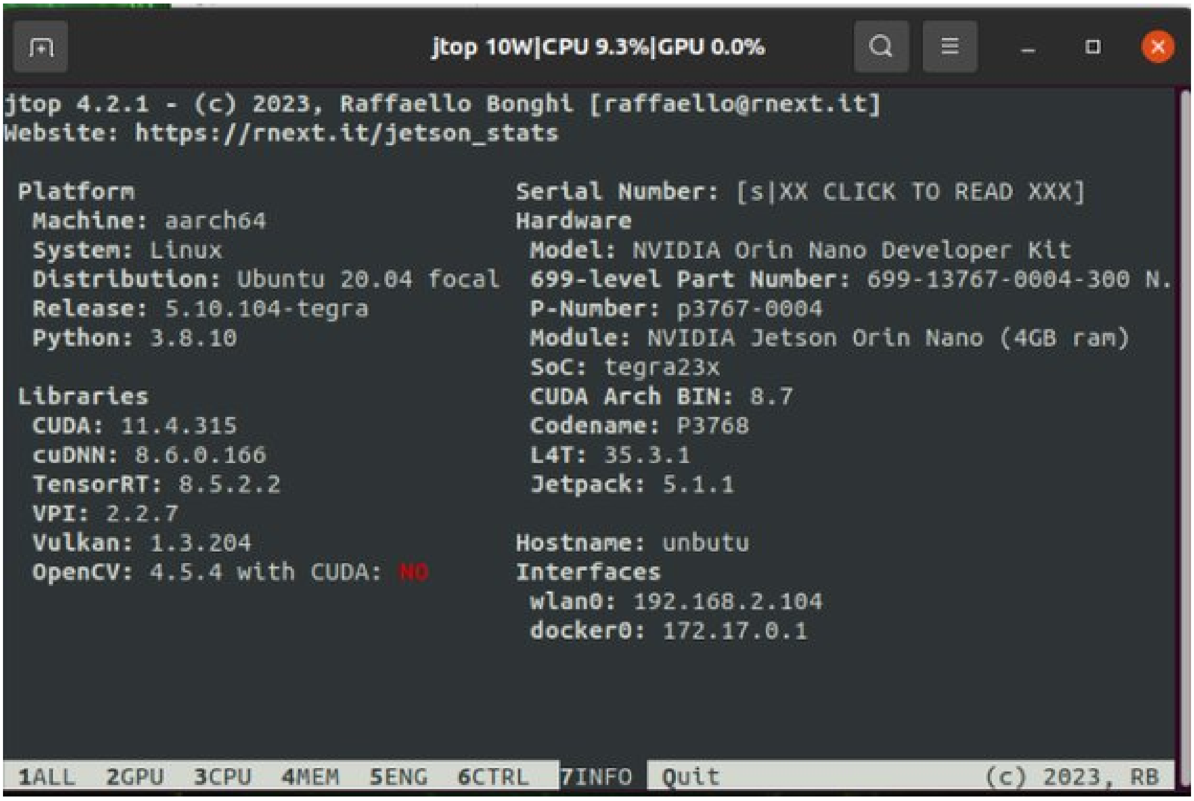
YOLO5 construction

This tutorial is suitable for building the official image of Jetson Orin nano yourself. If you are using the YAHBOOM version of the image, this tutorial can be ignored.

1.preparation

jetson orin nano *1

jetson orin nano configuration is as follows：



```
jtop 4.2.1 - (c) 2023, Raffaello Bonghi [raffaello@rnext.it]
Website: https://rnext.it/jetson_stats

 Platform                                  Serial Number: [s|XX CLICK TO READ XXX]
  Machine: aarch64                         Hardware
  System: Linux                             Model: NVIDIA Orin Nano Developer Kit
  Distribution: Ubuntu 20.04 focal          699-level Part Number: 699-13767-0004-300 N.
  Release: 5.10.104-tegra                   P-Number: p3767-0004
  Python: 3.8.10                            Module: NVIDIA Jetson Orin Nano (4GB ram)
                                            SoC: tegra23x
 Libraries                                  CUDA Arch BIN: 8.7
  CUDA: 11.4.315                            Codename: P3768
  cuDNN: 8.6.0.166                          L4T: 35.3.1
  TensorRT: 8.5.2.2                         Jetpack: 5.1.1
  VPI: 2.2.7
  Vulkan: 1.3.204                          Hostname: unbutu
  OpenCV: 4.5.4 with CUDA: NO              Interfaces
                                            wlan0: 192.168.2.104
                                            docker0: 172.17.0.1



 1ALL  2GPU  3CPU  4MEM  5ENG  6CTRL  7INFO  Quit                       (c) 2023, RB
```

If you have read the installation tutorials for torch and torch vision in the previous chapter, you can start directly from section 2.3

2.Environment construction of YOLO5（yolo5 v5.0)

2.1 Download the required module for YOLO5- torch

（If you follow the tutorial to install the Jetson-reference environment, this section can be ignored)

```
sudo apt-get install python3-pip libopenblas-base libopenmpi-dev
pip3 install Cython
pip3 install numpy torch-1.12.0a0+2c916ef.nv22.3-cp38-cp38-linux_aarch64
sudo apt-get install libjpeg-dev zlib1g-dev libpython3-dev libavcodec-dev libavformat-dev libswscale-dev
```

torch-1.12.0a0+2c916ef.nv22.3-cp38-cp38-linux_aarch64 This file is obtained from the attachment built in the environment and transferred to Jetson through WinSCP

2.2 Install the corresponding version of torchvision

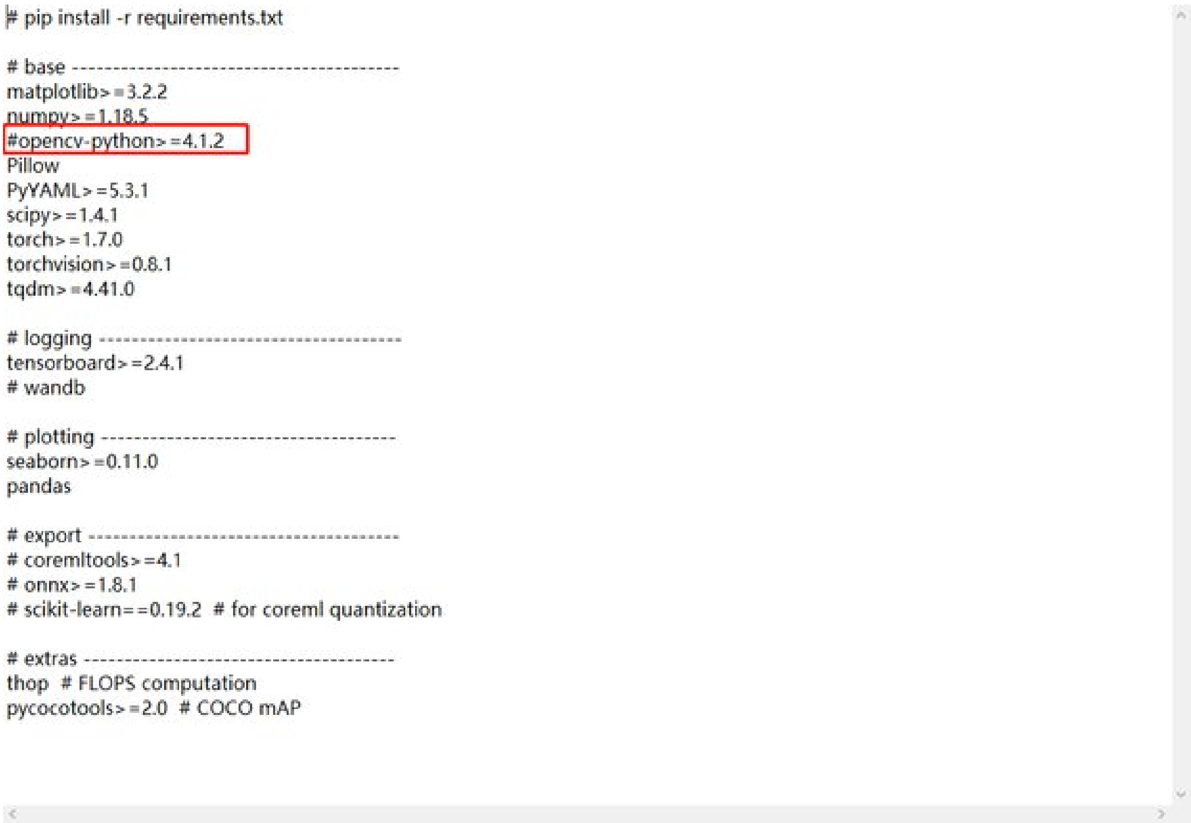
```
git clone --branch v0.13.0 https://github.com/pytorch/vision torchvision
cd torchvision
export BUILD_VERSION=0.13.0
python3 setup.py install --user
```

If git clone reports an error, please check the network and run again

2.3 Download the source code of YOLO5

```
cd ~
git clone https://github.com/marcoslucianops/DeepStream-Yolo
python3 -m pip install --upgrade pip
cd yolov5
```

Because Jetson orin nano already comes with opencv4.5.4, there is no need to install Python's opencv. You can verify it through import cv2
Therefore, we need to open the requirements file in the yolo5 directory and add a # sign before this line



```
# pip install -r requirements.txt

# base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
#opencv-python>=4.1.2
Pillow
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 ------------------------------------
seaborn>=0.11.0
pandas

# export --------------------------------------
# coremltools>=4.1
# onnx>=1.8.1
# scikit-learn==0.19.2  # for coreml quantization

# extras --------------------------------------
thop  # FLOPS computation
pycocotools>=2.0  # COCO mAP
```

Run after modification

```
pip3 install -r requirements.txt -i https://mirror.baidu.com/pypi/sample
```

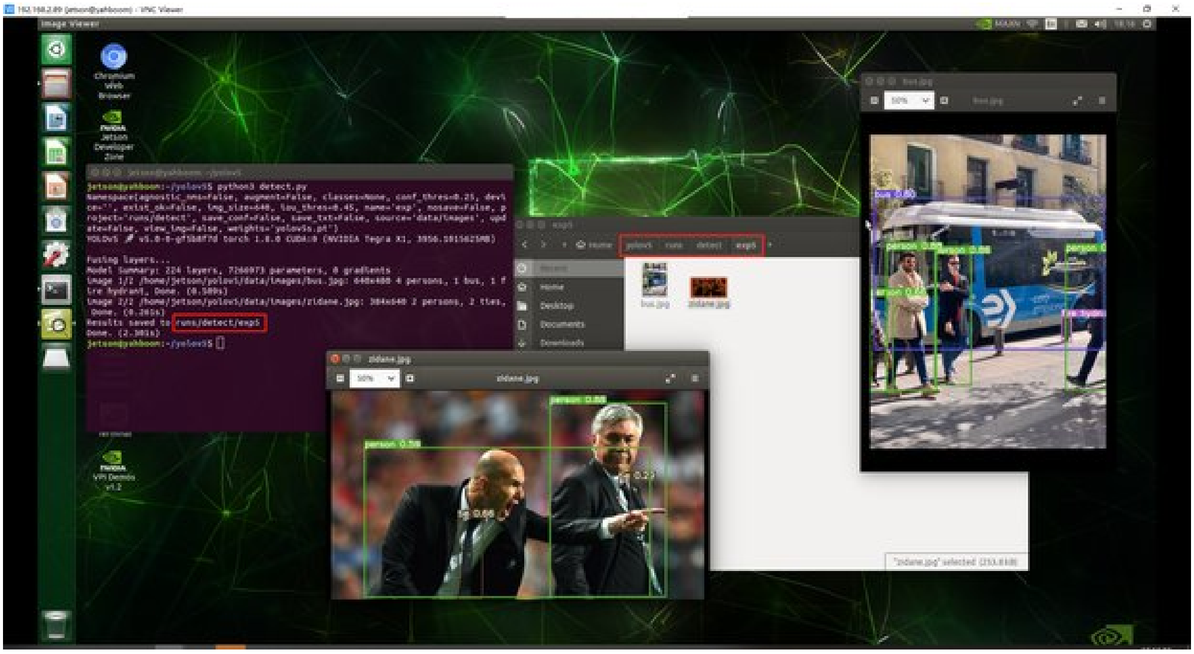
Wait until the download is complete

3.Verify if yolo5 has been successfully built

```
cd ~/yolov5
python3 detect.py
```

Wait for him to automatically download the weight file. If the network is not working, please obtain the yolov5s.pt file from the attachment provided by us and place it under the yolov5 folder If there are no errors reported, it indicates that YOLO5 has been successfully built and the identified results will be stored in the YOLOv5/runs/detect/exp path

The following is a picture of successful operation



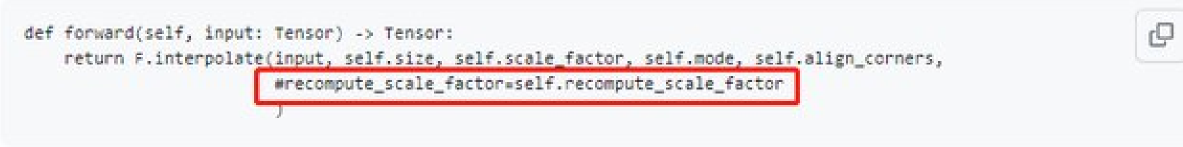
```
jetson@yahboom:~/yolov5$ python3 detect.py
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.25, devi
ce='', exist_ok=False, img_size=640, iou_thres=0.45, name='exp', nosave=False, p
roject='runs/detect', save_conf=False, save_txt=False, source='data/images', upd
ate=False, view_img=False, weights='yolov5s.pt')
YOLOv5 🚀 v5.0-5-gf5b8f7d torch 1.8.0 CUDA:0 (NVIDIA Tegra X1, 3956.1015625MB)

Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 0 gradients
image 1/2 /home/jetson/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, 1 f
ire hydrant, Done. (0.509s)
image 2/2 /home/jetson/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties,
 Done. (0.285s)
Results saved to runs/detect/exp5
Done. (2.381s)
jetson@yahboom:~/yolov5$
```

Exp5: Because the command python3 detect.py was run for the fifth time, the results were stored in the directory of exp5

If you run the third step, there may be an error due to the high version of the torch, so we need to make some modifications

```
sudo gedit /usr/local/lib/python3.8/dist-packages/torch/nn/modules/upsampling.py
```

Make a modification to line 153, recompute_scale_factor=self.recompute_scale_factor Just add '#' in front of the factor line.
The modified results are shown in the following figure：



```
def forward(self, input: Tensor) -> Tensor:
    return F.interpolate(input, self.size, self.scale_factor, self.mode, self.align_corners,
                        #recompute_scale_factor=self.recompute_scale_factor
                        )
```

Run step 3 again and it will work properly.