英伟达嵌入式GPU jetson Nano软件安装说明

英伟达嵌入式GPU jetson Nano软件安装说明

一、jetson nano烧录操作系统

1.1 硬件准备：

1.2 下载OS镜像：

1.3 格式化TF存储卡

1.4 写入镜像

1.5 初始化OS

二、Windows远程控制jetson nano

2.1 准备工作

三、jetson nano安装YOLOv5

3.1 准备工作

3.2 安装pytorch+torchvision

3.3 安装YOLOv5

四、jetson nano安装TensorRTX

4.1 选择合适版本的tensorRTX

4.2 修改配置文件

4.3 编译运行

五、jetson nano 安装Deepstream（可选）

5.1 安装Deepstream

5.2 测试yolov3模型

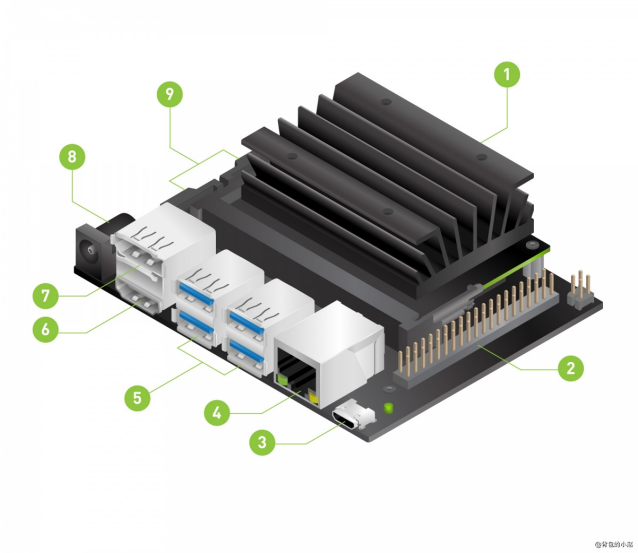
5.3 测试YOLOv5自定义模型

版本	作者	更新时间
V 1.0	@恩培-计算机视觉（抖音）	2022-03-29

> 说明：
>
> 1、硬件本身非课程内容，需自费购买
>
> 2、本课使用的是jetson nano 4G内存版
>
> 3、本课配合Windows系统使用，macOS暂未测试
>
> 4、因软件兼容问题，本文档不对其他版本相关软件做出效果承诺

一、jetson nano烧录操作系统

> 参考官网介绍：https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro



1.1 硬件准备：

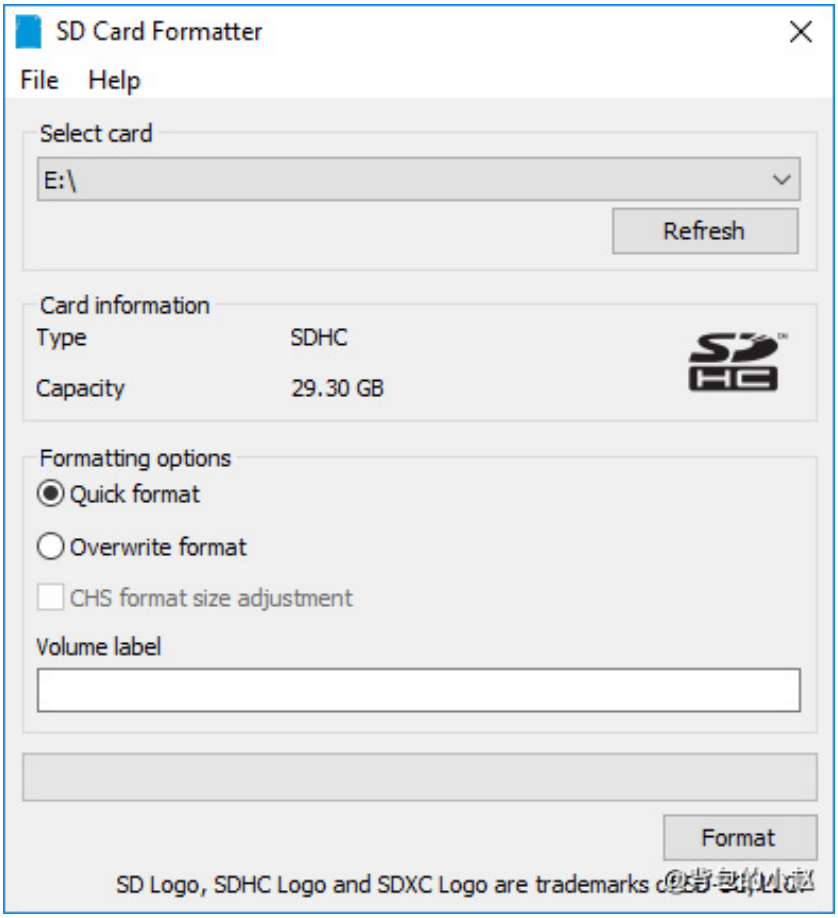
- Jetson Nano 4G 内存版
- Windows 电脑
- 32G以上空间的 TF存储卡（我用的128G后续软件消耗多）
- TF读卡器

1.2 下载OS镜像：

- 官方地址：https://developer.nvidia.com/jetson-nano-sd-card-image
- 附件OS位置：`/jetson软件安装/1.烧录系统/jetson-nano-jp461-sd-card-image.zip`

1.3 格式化TF存储卡

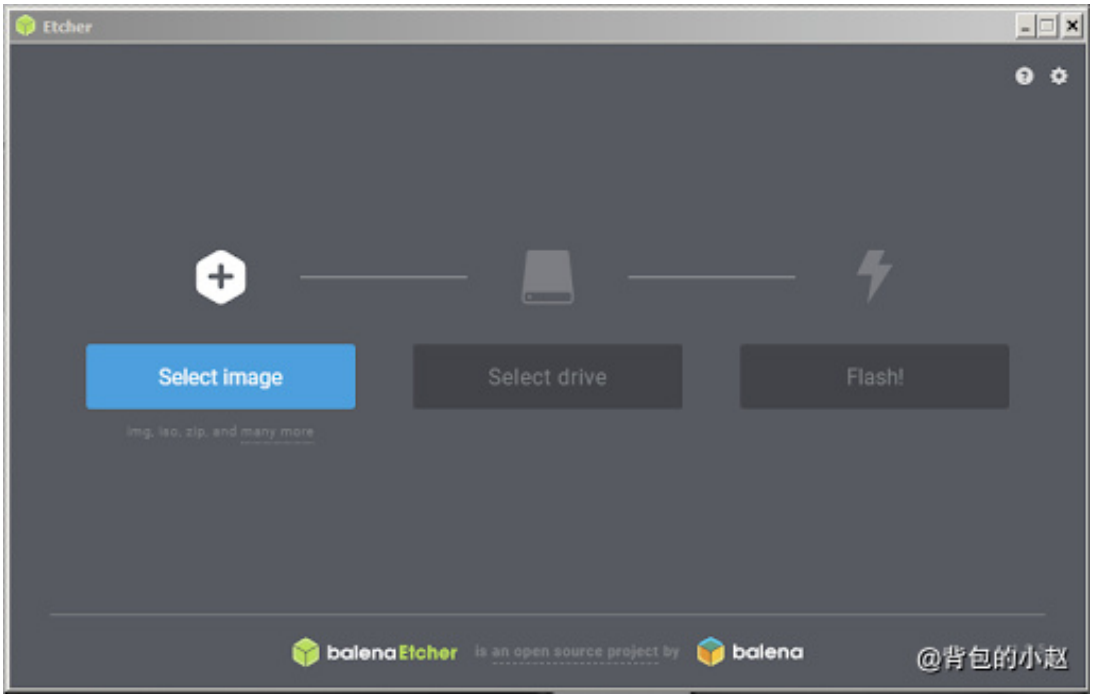
- Windows安装 SD Memory Card Formatter for Windows（附件位置：`/jetson软件安装/1.烧录系统/SDCardFormatterv5_WinEN.zip`）



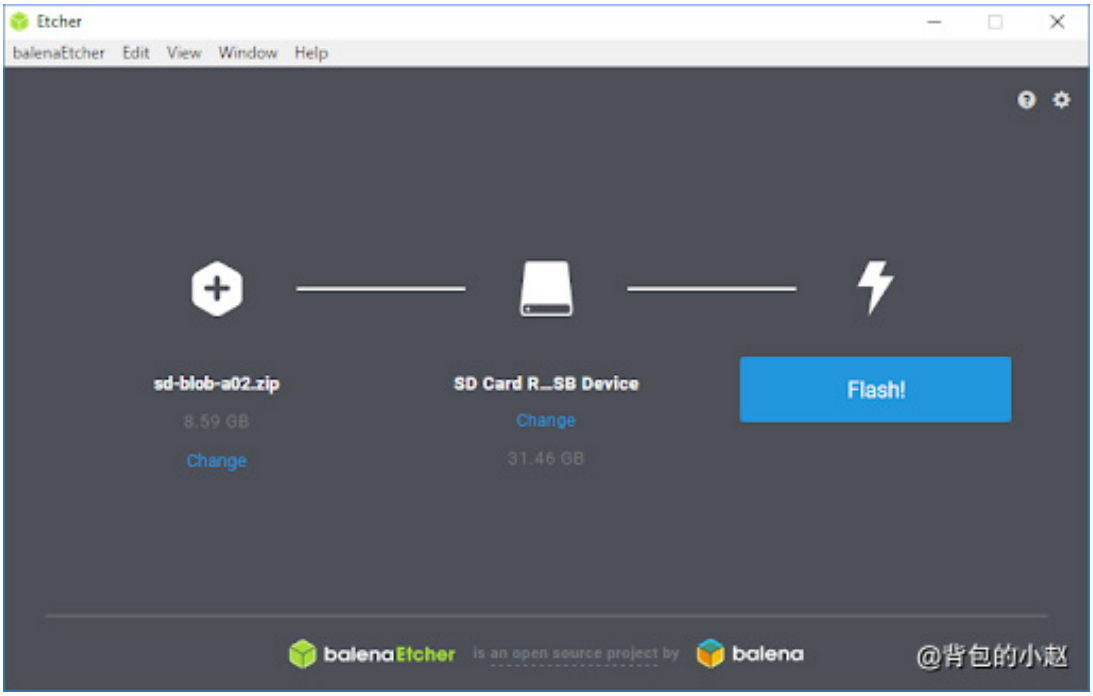
- 选择TF卡盘符（图中是E:\）
- 选择Quick format
- 点击`Format`开始格式化
- 耗时10分钟左右

1.4 写入镜像

- Windows安装Etcher（附件位置`/jetson软件安装/1.烧录系统/balenaEtcher-Setup-1.7.8.exe`）



- 点击`select image`选择镜像，选择`jetson-nano-jp461-sd-card-image.zip`
- 点击`selct drive`选择TF卡
- 点击`Flash`写入



1.5 初始化OS

- 插上键盘、鼠标、TF卡
- 插上网线（nano不带无线网卡，且不建议用USB网卡，不稳定）
- 插上电
- 设置好用户名密码

二、Windows远程控制jetson nano

2.1 准备工作

- Windows安装`putty`远程SSH远程命令工具、`winSCP`远程文件传输工具、`VNC`远程控制工具，（附件位置：`jetson软件安装/2.远程控制工具/`）
- 使用`putty`登录Jetson Nano，换源（最好有梯子，则不需要换源）：

```
# 首先备份好原来的source.list文件
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak

# 修改source.list，更换清华源
sudo vim /etc/apt/sources.list


# 按dG删除所有内容，复制下面内容加入
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic main multiverse restricted
universe
```

```
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-security main multiverse
restricted universe
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-updates main multiverse
restricted universe
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-backports main multiverse
restricted universe
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic main multiverse
restricted universe
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-security main
multiverse restricted universe
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-updates main
multiverse restricted universe
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-backports main multiv

# 保存

# 更新软件列表
sudo apt-get update




# pip换源
执行以下语句

cd ~
mkdir .pip
cd .pip
vi pip.conf

#pip.conf写入

[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple/
[install]
trusted-host = pypi.tuna.tsinghua.edu.cn

#保存pip.conf
```

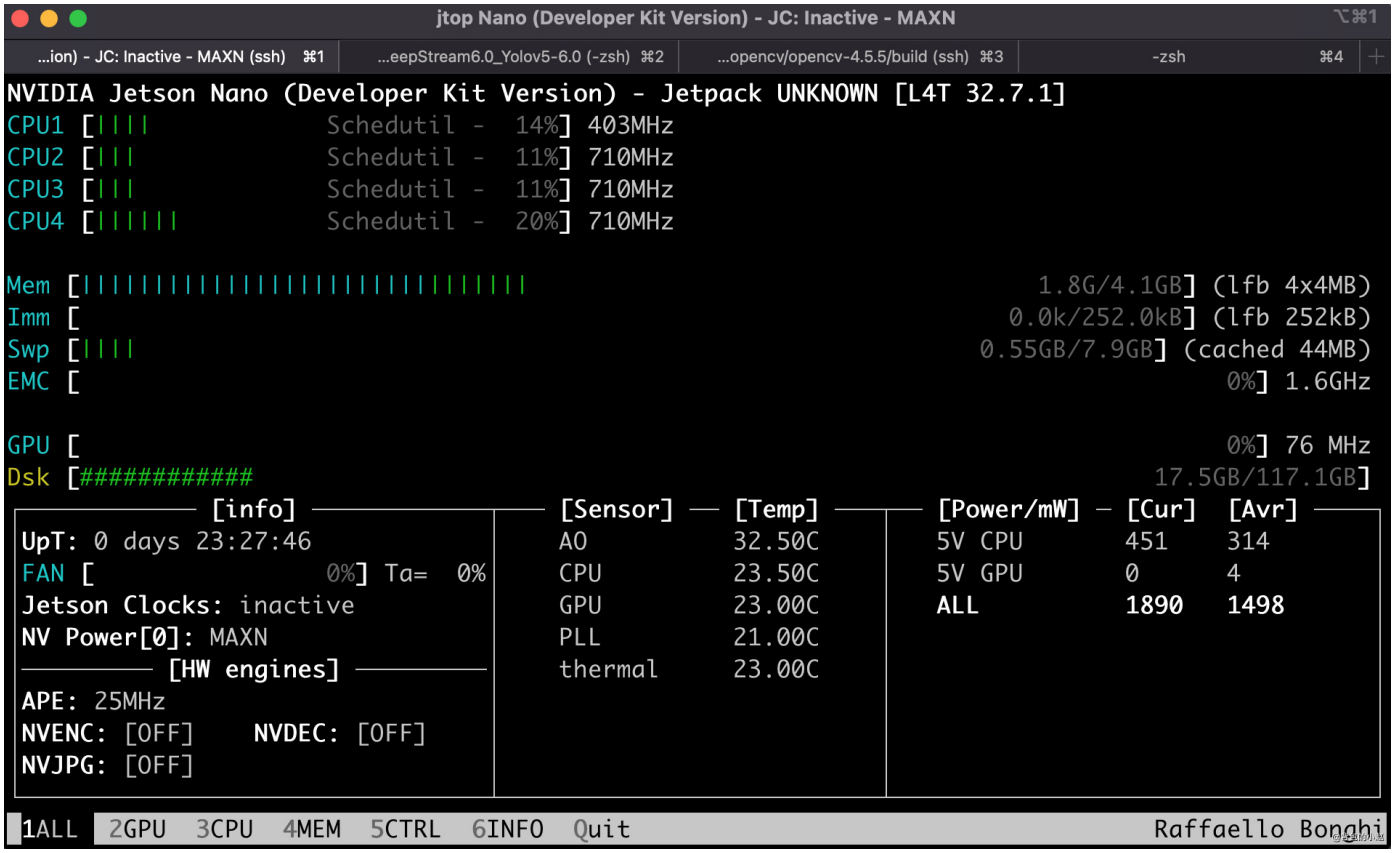
- 安装`jtop`监控

```
#升级pip3
python3 -m pip install --upgrade pip

# 安装jtop
sudo -H pip3 install -U jetson-stats

# 重启
sudo reboot

# 查看jtop，效果如下图
jtop
```



- jetson安装VNC

```
# 更新软件源
sudo apt update

# Enable VNC 服务
sudo ln -s ../vino-server.service /usr/lib/systemd/user/graphical-session.target.wants

#配置 VNC server
gsettings set org.gnome.Vino prompt-enabled false
gsettings set org.gnome.Vino require-encryption false

#编辑 org.gnome.Vino.gschema.xml文件
sudo vi /usr/share/glib-2.0/schemas/org.gnome.Vino.gschema.xml
```

```
# 文件最后，</schema>之前添加以下内容

<key name='enabled' type='b'>
  <summary>Enable remote access to the desktop</summary>
  <description>
    If true, allows remote access to the desktop via the RFB
    protocol. Users on remote machines may then connect to the
    desktop using a VNC viewer.
  </description>
  <default>false</default>
</key>


#设置为 Gnome 编译模式
sudo glib-compile-schemas /usr/share/glib-2.0/schemas


#设置 VNC
gsettings set org.gnome.Vino authentication-methods "['vnc']"

gsettings set org.gnome.Vino vnc-password $(echo -n '123456'|base64)

#重启
sudo reboot

# Windows测试VNC是否能连接
```

三、jetson nano安装YOLOv5

3.1 准备工作

- jtop 中 power mode选择maxn
- jtop中打开Jetson_clock
- 修改swap（4G后面编译或运行不够）

```
sudo vim /etc/systemd/nvzramconfig.sh

# 修改
mem=$((("${totalmem}" / 2 / "${NRDEVICES}") * 1024))
# 为

为mem=$((("${totalmem}" * 2 / "${NRDEVICES}") * 1024))

#重启
sudo reboot

#再进入输入jtop可以查看是否变化
```

3.2 安装pytorch+torchvision

> 如果需要下载其他版本，请进入官网：https://forums.developer.nvidia.com/t/pytorch-for-jetson-version-1-10-now-available/72048
>
> 本课使用的是：PyTorch v1.7 - torchvision v0.8.1，离线安装包文件位置：`jetson软件安装/3.pytorch_torchvision`

```
# 进入离线安装包目录

# 安装PyTorch
sudo apt-get install python3-pip libopenblas-base libopenmpi-dev
pip3 install Cython
pip3 install numpy torch-1.7.0-cp36-cp36m-linux_aarch64.whl

# 安装torchvision
sudo apt-get install libjpeg-dev zlib1g-dev libpython3-dev libavcodec-dev libavformat-dev libswscale-dev

# 解压vision-0.8.1.zip
unzip vision-0.8.1.zip
# 重命名
mv vision-0.8.1 torchvision
# 进入目录
cd torchvision
# 安装
export BUILD_VERSION=0.8.0  # where 0.x.0 is the torchvision version
python3 setup.py install --user


# 验证
>>> import torch
>>> print(torch.__version__)
>>> print('CUDA available: ' + str(torch.cuda.is_available()))
>>> print('cuDNN version: ' + str(torch.backends.cudnn.version()))
```

```
>>> a = torch.cuda.FloatTensor(2).zero_()
>>> print('Tensor a = ' + str(a))
>>> b = torch.randn(2).cuda()
>>> print('Tensor b = ' + str(b))
>>> c = a + b
>>> print('Tensor c = ' + str(c))

>>> import torchvision
>>> print(torchvision.__version__)
```

3.3 安装YOLOv5

```
# 克隆地址
git clone https://github.com/ultralytics/yolov5.git
# 如果下载速度慢，将附件：
jetson软件安装/5.yolov5相关/yolov5-master.zip
#用winSCP复制到jetson nano上，并解压重命名

# 进入目录
cd yolov5
# 安装依赖
pip3 install -r requirements.txt

# 可能matplotlib 安装出错，将：5.yolov5相关/matplotlib-3.3.3-cp36-cp36m-linux_aarch64.whl传输到nano，直接离线安装
pip3 install matplotlib-3.3.3-cp36-cp36m-linux_aarch64.whl


# 再检查一遍（之前可能安装中断）
pip3 install -r requirements.txt


# 将训练好的yolov5n.pt传输到yolov5目录下


# 以下指令可以用来测试yolov5
python3 detect.py --source data/images/bus.jpg --weights yolov5n.pt --img 640 #图片测试
python3 detect.py --source video.mp4 --weights yolov5n.pt --img 640 #视频测试,需要自己准备视频
python3 detect.py --source 0 --weights yolov5n.pt --img 640 #摄像头测试
```

- 报错处理

```
# 可能报错
ImportError: The _imagingft C module is not installed

# 重新安装pillow
pip3 install 'pillow==8.4.0'
```

四、jetson nano安装TensorRTX

> 参考资料：https://github.com/wang-xinyu/tensorrtx/tree/master/yolov5

4.1 选择合适版本的tensorRTX

- 对照YOLOv5版本，选择合适版本的tensorRTX，具体可参考 https://github.com/wang-xinyu/tensorrtx/tree/master/yolov5介绍
- 本课使用YOLOv5 6.0 版，所以需要clone 最新的tensortx：`git clone https://github.com/wang-xinyu/tensorrtx.git`，附件位置：`jetson软件安装/6.tensorRTX/tensorrtx-master.zip`

4.2 修改配置文件

```
# 在yololayer.h修改检测类别数量、输入画面大小

static constexpr int CLASS_NUM = 80;
static constexpr int INPUT_H = 640;
static constexpr int INPUT_W = 640;




# 在yolov5.cpp 修改数据类型、GPU、NMS、BBox confidence、Batch size

#define USE_FP16  // set USE_INT8 or USE_FP16 or USE_FP32，需要注意jetson nano GPU 不支持int8 型运算，所以这里不需要改动
#define DEVICE 0  // GPU id
#define NMS_THRESH 0.4
#define CONF_THRESH 0.5
#define BATCH_SIZE 1
#define MAX_IMAGE_INPUT_SIZE_THRESH 3000 * 3000 // ensure it exceed the maximum size in the input images !
```

4.3 编译运行

- 生成`.wts`文件

```
# 将tensorRTX yolov5 下的gen_wts.py复制到ultralytics yolov5目录下（注意不要弄混）
cp {tensorrtx}/yolov5/gen_wts.py {ultralytics}/yolov5

# 进入ultralytics yolov5目录
cd {ultralytics}/yolov5

# 运行，生成
python gen_wts.py -w {.pt 模型文件} -o {.wts 文件}
# 如：
python gen_wts.py -w yolov5n.pt -o yolov5n.wts

# 可以看到目录下有一个文件：
yolov5n.wts
```

- 编译

```
# 进入tensorRTX yolov5目录
cd {tensorrtx}/yolov5/

# 确保 yololayer.h 识别类别CLASS_NUM已经修改

mkdir build
cd build

# 复制yolov5n.wts到build目录
cp {ultralytics}/yolov5/yolov5n.wts {tensorrtx}/yolov5/build

cmake ..
make
```

- 生成engine 文件

```
sudo ./yolov5 -s [.wts] [.engine] [n/s/m/l/x/n6/s6/m6/l6/x6 or c/c6 gd gw]  //
serialize model to plan file

# 如
sudo ./yolov5 -s yolov5n.wts yolov5n.engine n
```

- 测试

```
sudo ./yolov5 -d [.engine] [image folder]  // deserialize and run inference, the images
in [image folder] will be processed.

sudo ./yolov5 -d yolov5n.engine ../samples

# 可以查看是否检测出
```

- 运行python 脚本`python yolo_trt_demo.py`（附件位置：`jetson软件安装/6.tensorRTX/yolo_trt_demo.py`），可能需要安装`pycuda`

```
# 安装pycuda
export CPATH=$CPATH:/usr/local/cuda-10.2/targets/aarch64-linux/include
export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/cuda-10.2/targets/aarch64-linux/lib

pip3 install pycuda --user

#或者:
sudo pip3 install --global-option=build_ext --global-option="-I/usr/local/cuda/include"
--global-option="-L/usr/local/cuda/lib64" pycuda
```

五、jetson nano 安装Deepstream（可选）

> Deepsream 可以更充分地利用硬件资源，但是他的API（C++、Python）门槛较高，这里流程仅做Demo参考；

5.1 安装Deepstream

- 参考官网安装：https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_Quickstart.html

```
# 安装依赖
sudo apt install \
libssl1.0.0 \
libgstreamer1.0-0 \
gstreamer1.0-tools \
gstreamer1.0-plugins-good \
gstreamer1.0-plugins-bad \
gstreamer1.0-plugins-ugly \
gstreamer1.0-libav \
libgstrtspserver-1.0-0 \
libjansson4=2.11-1

#安装librdkafka
git clone https://github.com/edenhill/librdkafka.git
cd librdkafka
git reset --hard 7101c2310341ab3f4675fc565f64f0967e135a6a
```

```
./configure
make
sudo make install

# 复制文件
sudo mkdir -p /opt/nvidia/deepstream/deepstream-6.0/lib
sudo cp /usr/local/lib/librdkafka* /opt/nvidia/deepstream/deepstream-6.0/lib


# 下载SDK，并复制到jetson nano上：
https://developer.nvidia.com/deepstream_sdk_v6.0.1_jetsontbz2

# 附件位置：/jetson软件安装/7.deepstream/deepstream_sdk_v6.0.1_jetson.tbz2

# 解压DeepStream SDK
sudo tar -xvf deepstream_sdk_v6.0.1_jetson.tbz2 -C /
# 安装
cd /opt/nvidia/deepstream/deepstream-6.0
sudo ./install.sh
sudo ldconfig
```

5.2 测试yolov3模型

```
#进入YOLO目录
cd sources/objectDetector_Yolo/
#编译
sudo CUDA_VER=10.2 make -C nvdsinfer_custom_impl_Yolo

#备份
sudo cp prebuild.sh prebuild.sh.back
# 只保留最后两句（下载yolo-tiny权重文件）

sudo vi prebuild.sh

#运行下载
sudo ./prebuild.sh

# 在jetson nano上运行，此时需要连接显示器（通过VNC看不到画面）
cd /opt/nvidia/deepstream/deepstream-6.0/sources/objectDetector_Yolo
# 测试
deepstream-app -c deepstream_app_config_yoloV3_tiny.txt

# 在命令行可以看到帧率
```

5.3 测试YOLOv5自定义模型

```
# 进入sources路径
cd  /opt/nvidia/deepstream/deepstream-6.0/sources/

# clone YOLOv5 deepstream
git clone https://github.com/enpeizhao/DeepStream6.0_Yolov5-6.0.git
# 附件位置：/jetson软件安装/7.deepstream/DeepStream6.0_Yolov5-6.0-main.zip

# 解压进入目录
cd DeepStream6.0_Yolov5-6.0
#拷贝tensorRTX生成的engine文件（yolov5n.engine）到DeepStream6.0_Yolov5-6.0目录
cp {tensorrtx}/yolov5/build/yolov5n.engine ./

# 编译
sudo CUDA_VER=10.2 make -C nvdsinfer_custom_impl_Yolo

# 修改参数
sudo vi config_infer_primary.txt
num-detected-classes=6 # 改为对应类别
pre-cluster-threshold=0.25 #改变CONF_THRESH

# 修改对应的类别标签
sudo vi labels.txt

# 如果修改 nvdsinfer_custom_impl_Yolo/ 下的文件都需要重新编译
sudo vi yololayer.h
# static constexpr int CLASS_NUM = 6;

#如要改变NMS_THRESH, 编辑文件nvdsinfer_custom_impl_Yolo/nvdsparsebbox_Yolo.cpp并重新编译

#define kNMS_THRESH 0.45

# 重新编译
sudo CUDA_VER=10.2 make -C nvdsinfer_custom_impl_Yolo

# 推理
deepstream-app -c deepstream_app_config.txt

# 使用USB摄像头
deepstream-app -c source1_usb_dec_infer_yolov5.txt
```