yolov5部署到jetson nano

一、Jetson Nano烧录系统及系统初始化设置

比较简单：详见参考地址

参考地址：https://blog.csdn.net/weixin_44293881/article/details/128554780

二、Windows远程控制jetson nano

2.1 准备工作

- 使用putty登录Jetson Nano，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


#Jetson Nano—安装pip

# 更新软件列表
sudo apt-get update
#升级一下pip保证是最新版的：
sudo apt-get upgrade
```

```
#安装pip
sudo apt-get install python3-pip
pip3 install --upgrade pip


# 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
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
>
> 推荐使用：PyTorch v1.8 - torchvision v0.9.0
>
> 下面是1.7版本的，根据下载好的文件名改一下命令就可以了

```
# 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__)
```

- 解决:Jetson系列python3 import 报错 "Illegal instruction[cpre dumped]"
  - **解决方法**

    编辑环境变量

    ```
    sudo gedit ~/.bashrc
    ```

    在最后一行添加

    ```
    export OPENBLAS_CORETYPE=ARMV8
    ```

    最后，激活环境变量

    ```
    source ~/.bashrc
    ```

    即可!
  - 参考链接：https://blog.csdn.net/FriendshipTang/article/details/115445902
- 由于numpy版本与opencv-python版本不匹配才从torch1.7换到了torch1.8
  - 如果numpy与torch版本不匹配可能还会有no muddle torch.fx报错
  - numpy与opencv-python和torch1.7不能同时满足
  - 更换为torch1.8解决
- 需要注意的是在安装torch时,一定要安装numpy,不然的话显示你安装torch成功,但是你conda list是找不到的.

  安装完torch,在测试时报错:非法指令,核心已转储(让人头疼!),查询很久才发现怎么解决,命令如下

  export OPENBLAS_CORETYPE=ARMV8
  - 参考链接：https://blog.csdn.net/weixin_48057700/article/details/121782986

参考链接:

https://blog.openpilot.cc/archives/2271

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
- 本人使用的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
#根据需要

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 (可选)

#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
80→6

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
```