YOLOv5实战-Jetson Nano部署

1 烧录系统镜像

1） 下载系统镜像

英伟达官方地址 https://developer.nvidia.com/embedded/downloads

下载系统镜像JetPack 4.5.1（注意有4GB和2GB两个版本，根据Jetson Nano的型号选择相应的版本）

百度网盘下载链接:

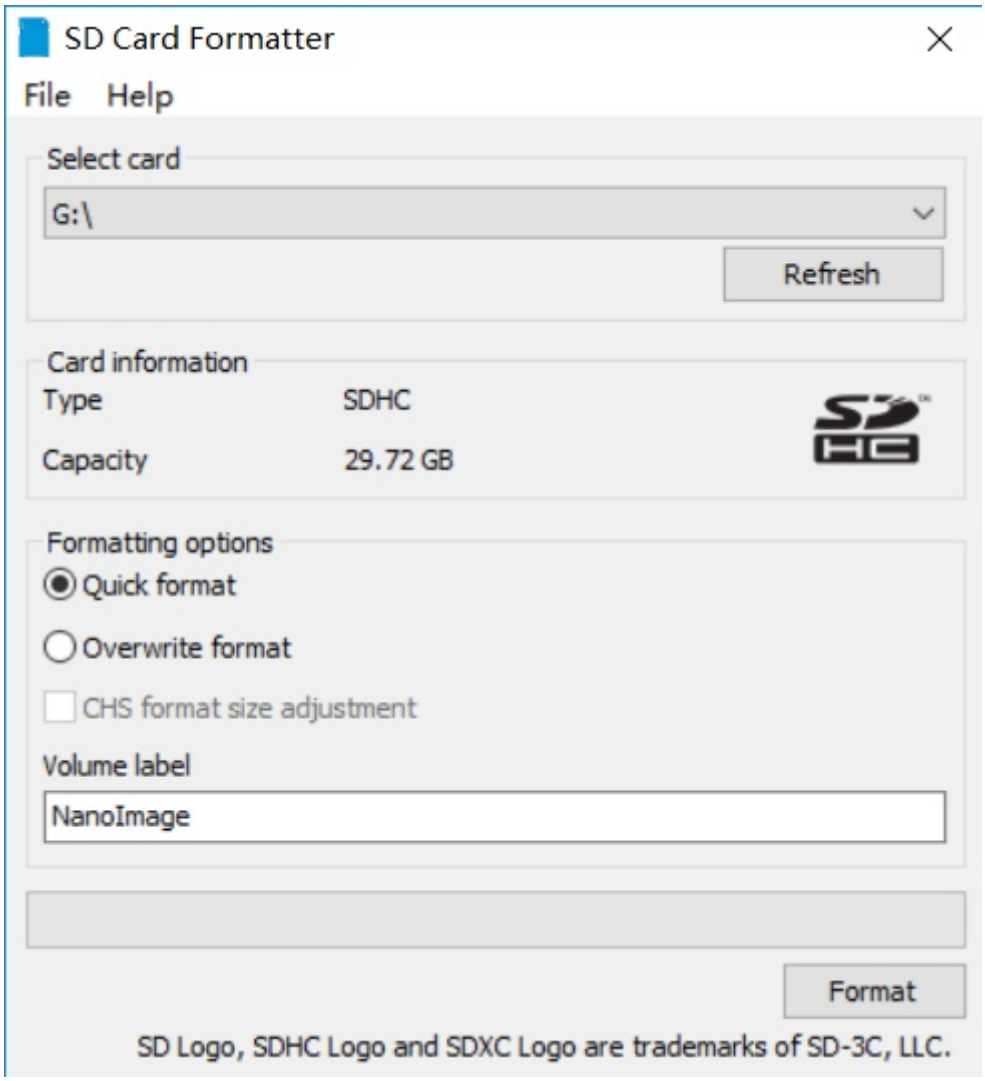
链接：https://pan.baidu.com/s/1gpg9coTXgSyGgc5CDfv1iQ 提取码：sj5x

2) 格式化 SD 卡

下载SD Card Formatter软件并安装

https://www.sdcard.org/downloads/formatter/sd-memory-card-formatter-for-windows-download/

使用 SD Card Formatter 格式化 SD 卡

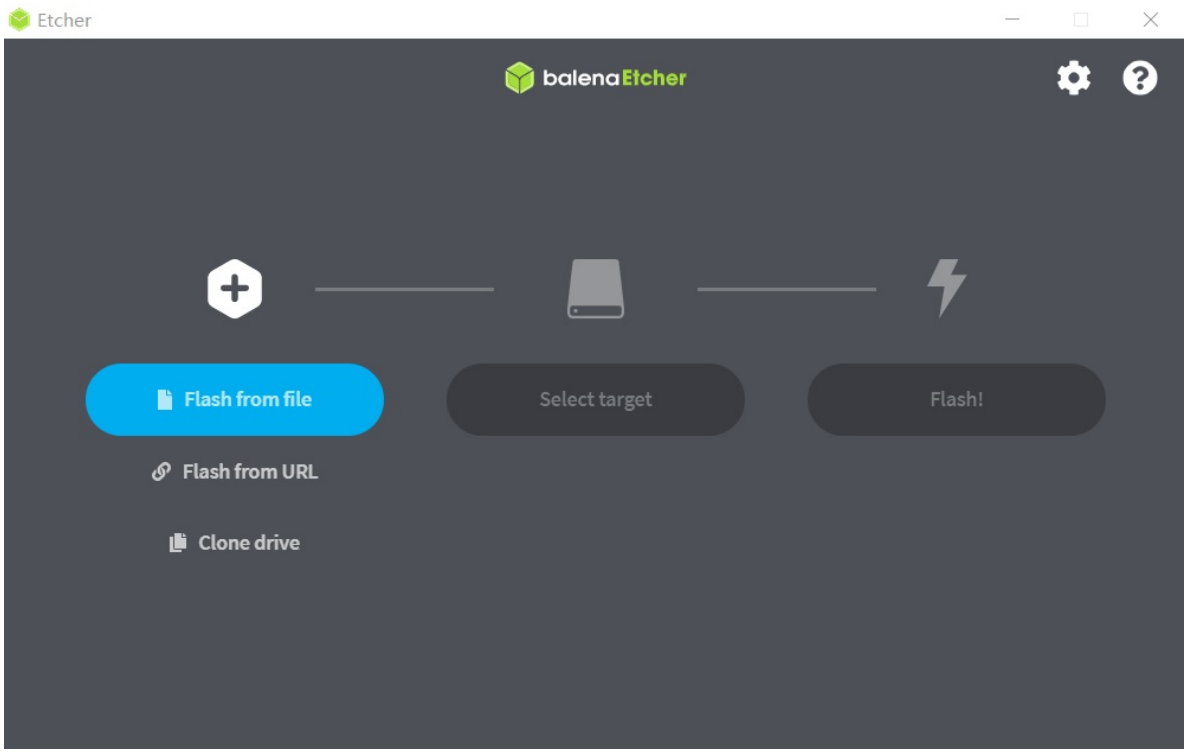


3) 使用 Etcher 写入镜像

下载Etcher软件并安装 https://www.balena.io/etcher/

使用Etcher软件写入镜像

大概30分钟，镜像写入完毕



4）使用SD卡开机

烧写完成后，将 SD 卡插入 Jetson Nano, 开机，完成一些设置，如时区、语言、输入法等

2 远程登录工具

1）安装和使用远程登录工具PuTTY

(1）下载和安装PuTTY

https://www.chiark.greenend.org.uk/~sgtatham/putty/

(2) 确定自己开发板的 IP 地址

在安装系统中，在界面通过 ctr+Alt+T 打开命令提示符，输入：ifconfig 找到对应的有线网卡 eth0 的 IP 地址

(3）使用PuTTY远程登录Nano开发板

打开 PuTTY , 输入IP地址和端口号，默认开启 ssh 服务。

打开 Open；输入用户名和密码后进入终端模式

2）安装和使用远程文件传输软件WinSCP

WinSCP 是一个 Windows 环境下使用的 SSH 的开源图形化 SFTP 客户端。同时支持 SCP 协议。它的主要功能是在本地与远程计算机间安全地复制文件，并且可以直接编辑文件。

方便在 windows 和 linux 两个不同的系统之间传输文件。

下载和使用WinSCP https://winscp.net/

3） VNC 远程桌面控制

(1) 安装 vino

利用PuTTY, 在Nano上执行

```
sudo apt update
```

```
sudo apt install vino
```

(2) Enable VNC 服务

执行

```
sudo ln -s ../vino-server.service /usr/lib/systemd/user/graphical-
session.target.wants
```

配置 VNC server:

执行

```
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

执行

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

文件最后添加

```
<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
```

(3) 设置 VNC

登陆密码('thepassword' 修改为自己的密码)

```
gsettings set org.gnome.Vino authentication-methods "['vnc']"
```

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

(4) 重启机器，验证是否设置 vnc 成功

```
sudo reboot
```

这种方式是属于手动启动。

每次都需要手动启动会比较麻烦，下面会设置开机自启动的形式。

（5）设置开机自启动 VNC Server

VNC 服务器只有在您本地登录到 Jetson Nano之后才可用。如果希望 VNC 自动可用，可使用系统设置应用程序来启用自动登录。

```
gsettings set org.gnome.Vino enabled true
```

```
mkdir -p ~/.config/autostart
```

```
vi ~/.config/autostart/vino-server.desktop
```

将下面的内容添加到该文件中，保存并退出。

```
[Desktop Entry]
Type=Application
Name=Vino VNC server
Exec=/usr/lib/vino/vino-server
NoDisplay=true
```

(6) 连接 VNC Server

使用 vnc viewer 软件进行 VNC 连接，首先需要查询 IP 地址； 输入 IP 地址后点击 OK，双击对应的 VNC 用户输入密码，最后进入到 VNC 界面

3 安装和测试DeepStream

1) Install Dependencies

执行以下命令以安装需要的软件包：

```
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
```

2) Install the DeepStream SDK

Using the DeepStream tar package

1. 下载 DeepStream 5.1 Jetson tar package deepstream_sdk_v5.1.0_jetson.tbz2, 到 Jetson Nano.
2. 输入以下命令以提取并安装DeepStream SDK：

```
sudo tar -xvf deepstream_sdk_v5.1.0_jetson.tbz2 -C /
```

```
cd /opt/nvidia/deepstream/deepstream-5.1
```

```
sudo ./install.sh
```

```
sudo ldconfig
```

3) DeepStream测试

执行命令

```
cd /opt/nvidia/deepstream/deepstream-5.1/sources/objectDetector_Yolo
```

执行编译命令：

```
CUDA_VER=10.2 make -C nvdsinfer_custom_impl_Yolo
```

编辑文件prebuild.sh，注释掉除yolov3-tiny的语句

执行：

```
./prebuild.sh
```

下载yolov3-tiny.cfg和yolov3-tiny.weights

执行命令：

```
deepstream-app -c deepstream_app_config_yoloV3_tiny.txt
```

4 yolov5项目克隆和安装（电脑上完成！）

1）安装Anaconda和PyTorch

安装pytorch1.7以上版本

2） 克隆yolov5项目

克隆项目到本地

```
git clone https://github.com/ultralytics/yolov5.git
```

或直接下载V4.0版本的源码

3） 安装所需库

使用清华镜像源：

在yolov5路径下执行：

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple -r requirements.txt
```

注意：simple 不能少, 是 https 而不是 http

4） 下载预训练权重文件

下载yolov5s.pt，yolov5m.pt，yolov5l.pt，yolov5x.pt权重文件，并放置在weights文件夹下

百度网盘下载链接:

请见课件

5） 安装测试

测试图片：

在yolov5路径下执行

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

6）训练自己的数据集

可参考课程《YOLOv5目标检测实战：训练自己的数据集》

5 生成wts文件（在电脑上完成！）

1） 克隆tensorrtx

```
git clone https://github.com/wang-xinyu/tensorrtx.git
```

2） 生成yolov5s.wts文件

// 下载权重文件yolov5s.pt // 将文件tensorrtx/yolov5/gen_wts.py 复制到ultralytics/yolov5 // ensure the file name is yolov5s.pt and yolov5s.wts in gen_wts.py // go to ultralytics/yolov5 执行

```
python gen_wts.py
```

// a file 'yolov5s.wts' will be generated.

6 生成yolov5s.engine文件（在Nano上完成！）

1） 克隆tensorrtx

```
git clone https://github.com/wang-xinyu/tensorrtx.git
```

2） 编译tensorrtx/yolov5

// go to tensorrtx/yolov5 // ensure the macro NET in yolov5.cpp is s // update CLASS_NUM in yololayer.h if your model is trained on custom dataset

```
mkdir build
cd build
cmake ..
make
```

注意: by default, yolov5 script generate model with batch size = 1, FP16 mode and s model.

```
#define USE_FP16  // comment out this if want to use FP32
#define DEVICE 0  // GPU id
#define NMS_THRESH 0.4
#define CONF_THRESH 0.5
#define BATCH_SIZE 1

#define NET s  // s m l x
```

如果你需要改变上述默认参数，可在编译前修改文件 yolov5.cpp

另外，可改变yololayer.h中的参数

```
static constexpr int CLASS_NUM = 80;
static constexpr int INPUT_H = 608;
static constexpr int INPUT_W = 608;
```

3) copy文件'yolov5s.wts' 文件到tensorrtx/yolov5/build目录下

4) 生成yolov5s.engine

执行

```
// For example yolov5s
sudo ./yolov5 -s yolov5s.wts yolov5s.engine s
sudo ./yolov5 -d yolov5s.engine ../samples
```

可参考课程《YOLOv5目标检测实战：TensorRT加速部署》

7 使用DeepStream部署yolov5s

1）拷贝yolov5 deepstream文件

```
sudo chmod -R 777 /opt/nvidia/deepstream/deepstream-5.1/sources/
```

拷贝yolov5 deepstream文件yolov5.zip到/opt/nvidia/deepstream/deepstream-5.1/sources/

然后，解压文件

```
cd /opt/nvidia/deepstream/deepstream-5.1/sources/

unzip yolov5.zip
```

2) 拷贝yolov5.engine

```
cp /home/nano/tensorrtx/yolov5/build/yolov5s.engine
/opt/nvidia/deepstream/deepstream-5.1/sources/yolov5
```

备注：yolov5 deepstream文件yolov5.zip的来源

https://github.com/marcoslucianops/DeepStream-Yolo

修改了文件夹DeepStream-Yolo/external/yolov5/nvdsinfer_custom_impl_Yolo下的文件yololayer.h和yololayer.cu使其可以在YOLOv5 V4.0上工作

3）编译

```
cd /opt/nvidia/deepstream/deepstream-5.1/sources/yolov5
CUDA_VER=10.2 make -C nvdsinfer_custom_impl_Yolo
```

注意: 针对自己的数据集

可修改config_infer_primary.txt中的参数

```
num-detected-classes=80
```

以及类别名称文件labels.txt

另外，

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
```

如要改变CONF_THRESH, 编辑文件config_infer_primary.txt

```
[class-attrs-all]
pre-cluster-threshold=0.25
```

8 部署测试

1）测试视频文件推理

```
cd /opt/nvidia/deepstream/deepstream-5.1/sources/yolov5

deepstream-app -c deepstream_app_config.txt
```

2）USB摄像头视频测试

摄像头简单检测指令：

```
ls /dev/video*
```

安装v4l-utils工具：

```
sudo apt install v4l-utils
```

检测摄像头比较完整信息的指令：

```
v4l2-ctl --list-devices
```

摄像头更细致规格的查看指令：

```
v4l2-ctl --device=/dev/video0 --list-formats-ext

v4l2-ctl --device=/dev/video1 --list-formats-ext
```

YOLOv5 USB摄像头视频测试命令：

```
deepstream-app -c source1_usb_dec_infer_yolov5.txt
```

3） CSI摄像头视频测试

```
deepstream-app -c source1_csi_dec_infer_yolov5.txt
```