



Eyecloud OpenNCC Software Development Kit (SDK)

Getting Started Guide

September 2021

Revision 1.1.0

技术支持

扫描下方二维码“eyecloud 小助手”二维码添加微信好友，成功后小助手会邀请您入群。

申请请备注“官方文档”

eyecloud小助手



联系方式

电话：0571-8535-2616

邮箱：info@eyecloud.tech

版权声明

本手册版权归杭州眼云智家科技有限公司所有。未经本公司书面许可，任何单位及个人无权以任何形式复制、传播、转载本手册的任何内容，违者将被追究法律责任。

修订历史

Version	Date	Editor	Description
1.0.0	June 2021	Zed	
1.1.0	Sept. 2021	Zed	更新目录结构，新增 NVIDIA 及 eMMC 内容
1.2.0	Nov.2022	ZL	增加 yolo data cloud push

目录

技术支持	2
联系方式	2
版权声明	2
修订历史	3
目录	4
第一章 介绍	6
1.1 概述	6
1.2 SDK 结构	6
1.3 支持的产品及平台	6
第二章 入门指导	7
2.1 快速入门之 Linux(Ubuntu)	7
2.1.1 环境搭建	7
2.1.2 OpenNCC_Linux 操作演示	8
2.2 快速入门之 Windows	13
2.2.1 环境搭建	13
2.2.2 OpenNCC_Windows 操作演示	13
2.3 快速入门之 Raspberry Pi	14
2.3.1 环境搭建	14

2.3.2 OpenNCC_Raspberry 操作演示	14
2.4 快速入门之 NVIDIA	15
2.4.1 环境搭建	15
2.5 Custom (自定义)	16
2.5.1 环境搭建	23
2.6 运行结果演示	24
第三章 软件概述	25
3.1 资源库	25
3.1.1 OpenNCC 固件	25
3.1.2 OpenNCC API 库	26
3.1.3 AI 模型	26
3.2 应用例程	29
3.3 OpenNCC 运行机制介绍	29
3.3.1 OpenNCC 独立模式	30
3.3.2 OpenNCC 混合模式	32
3.3.3 协处理计算棒模式	33
3.3.4 独立模式和协处理模式区别	35
常见问题	37

第一章 介绍

1.1 概述

该文档用于介绍 OpenNCC Software Development Kit (SDK) 并且包含了启动，运行及开发的所有必要信息。

1.2 SDK 结构

目录	内容
./Platform	包含不同平台生成运行环境的脚本。
./docs	包含 SDK 相关介绍和文档。
./SDK/Drivers	包含不同平台所必须安装的驱动。
./SDK/Example	包含 SDK 的相关例程。
./SDK/Source	包含固件，模型及 SDK 库文件。
./SDK/Tools	包含相关的模型转换及编译工具。
./Viewer	包含已编译的 Viewer 及 QT 源码。

1.3 支持的产品及平台

SDK 支持的产品如下：

- OpenNCC DK
- OpenNCC Lite
- OpenNCC USB

SDK 支持的开发平台如下 “

- Linux
 - Ubuntu 16.04, Ubuntu 18.04
 - Raspberry Pi (树莓派)
 - NVIDIA (英伟达)
- Windows 10

提示：其他平台可通过官方[技术支持](#)渠道联系我们实现定制化服务。

第二章 入门指导

进入 openncc/Platform , 目录如下 :

```
└─ Platform
   └─ Linux
      └─ NVIDIA
      └─ RaspberryPi
      └─ Ubuntu
   └─ Windows
   └─ Custom
   └─ README.md
```

选择需求的文件目录进入 , 对应文件夹内包含环境搭建的脚本。

警告 :环境搭建脚本会自动生成和覆盖相关文件 ,运行前请确认是首次运行或已经完成备份。

2.1 快速入门之 Linux(Ubuntu)

2.1.1 环境搭建

- 进入目录 openncc/Platform/Linux/Ubuntu。
- 右键打开终端。
- 输入命令 `./build_ubuntu.sh`。

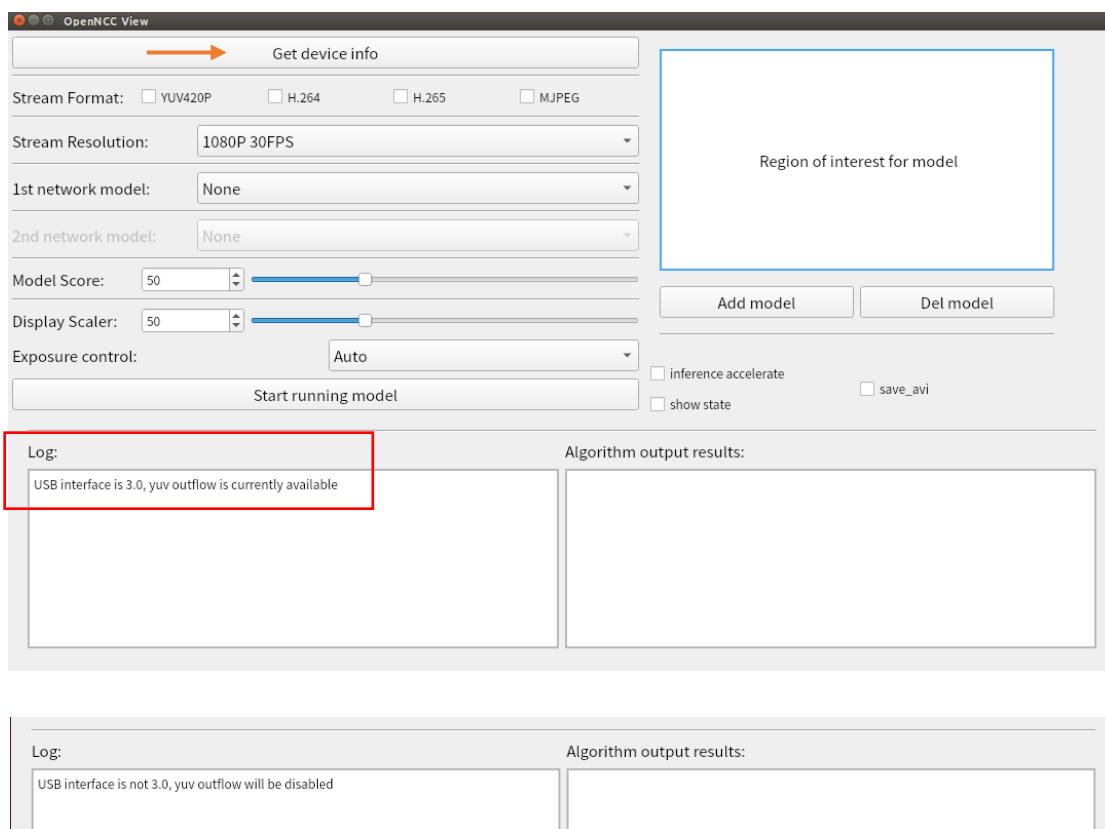
生成目录如下 :

```
└─ Ubuntu
   └─ Example
   └─ Source
   └─ Viewer
   └─ build_Ubuntu.sh
   └─ README.md
```

2.1.2 OpenNCC_Linux 操作演示

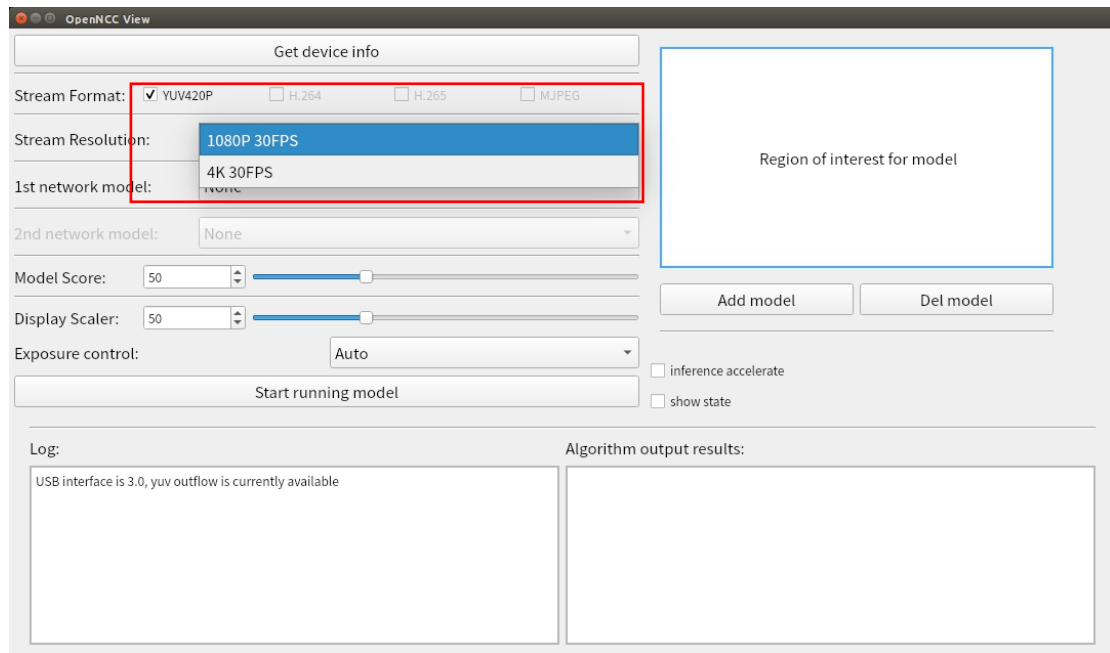
- 进入 openncc/Platform/Linux/Ubuntu/Viewer/OpenNcc_Linux 目录。
- 右键打开终端，执行 `sudo ./AppRun`，启动软件。
- 连接 OpenNCC 相机到电脑 USB 3.0 接口，点击 `Get device info` 按钮获取设备信息，此时 log 区域会有两种提示：
 - USB3.0 : USB interface is 3.0, yuv outflow is currently available.
 - USB2.0 :USB interface is not 3.0, yuv outflow will be disabled.

提示：OpenNCC TYPE-C 接口有正反接入两种模式，分别对应 USB 3.0 和 USB 2.0。受传输速率影响，USB2.0 模式下会暂时禁用 YUV420P 格式的视频流输出。

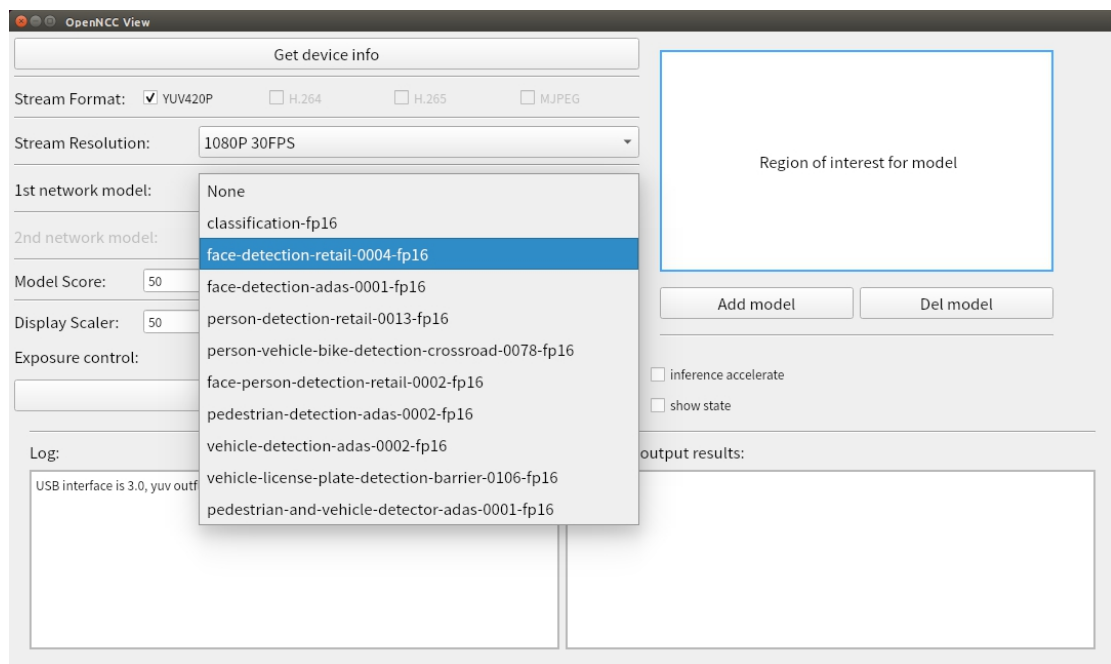


如果需要切换 USB 模式，旋转 TPYE-C 接口再次接入，重新点击 `Get device info` 按钮即可。如果仍然没有改变，请检查电脑 USB 接口类型或[联系我们](#)。

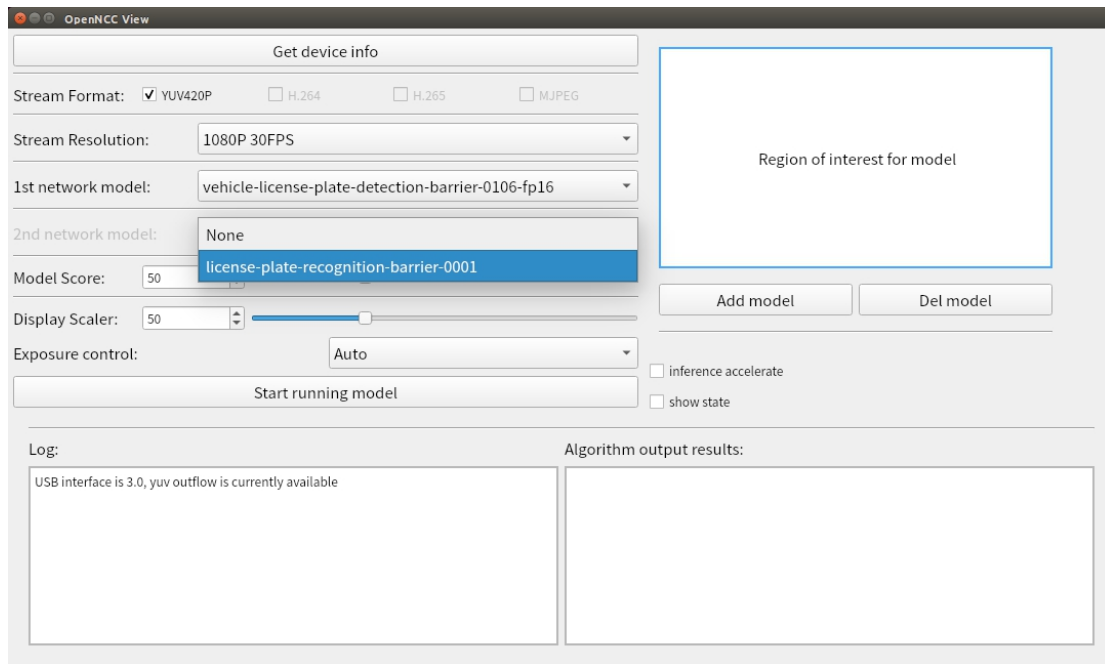
- 任意选择一种视频流格式 yuv420p/H.264/H.265/mjpeg。
- “Stream Resolution”：两种分辨率可供选择，1080p 和 4K。
(具体由 OpenNCC 产品类型决定)



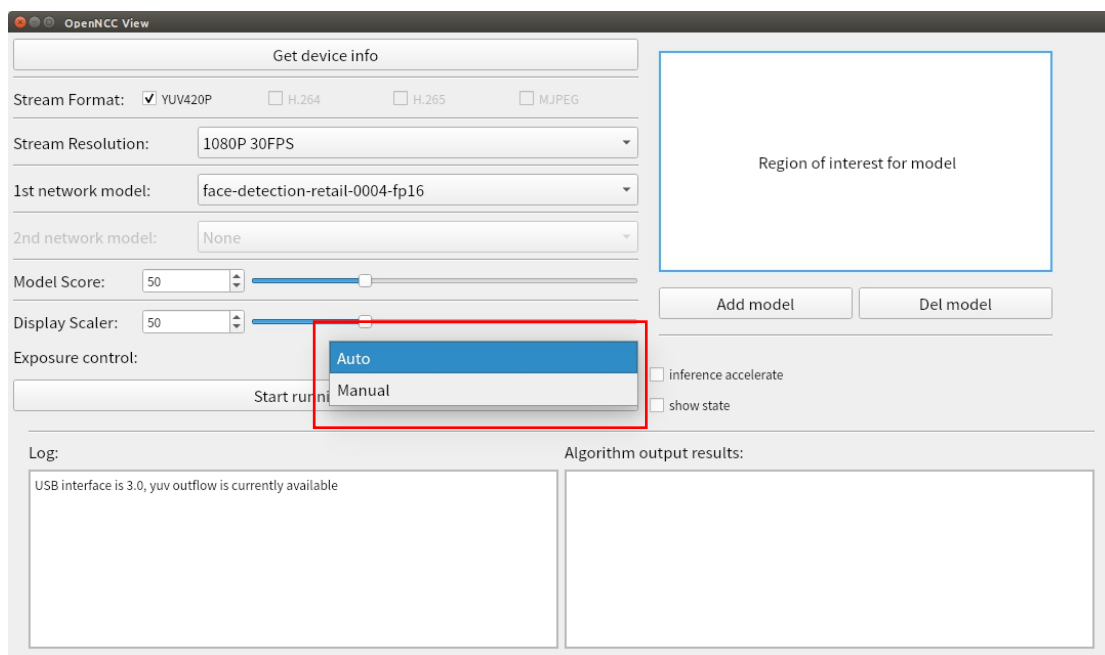
- “1st network model”：选择算法模型。（详细模型介绍见 [3.1.3](#)）
目前支持 10 多种模型可选，选择 None 即不加载模型，仅显示原始视频流。

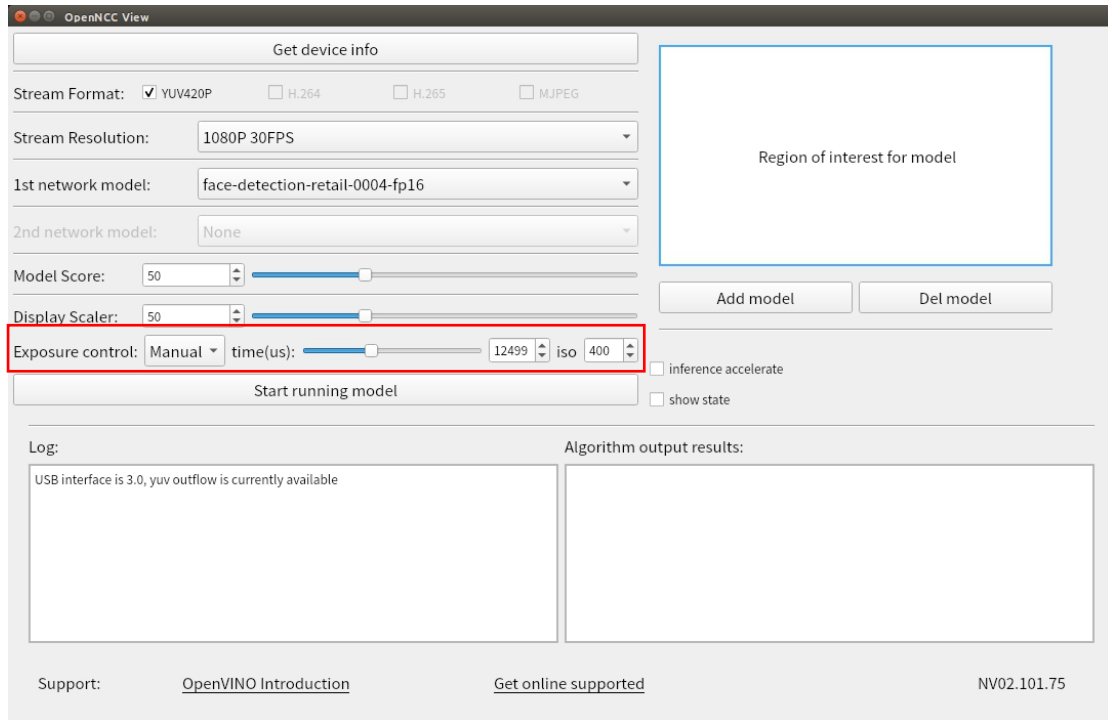


- 可以通过框选 ROI 区域限制算法区域，算法只对区域内的场景进行识别。（具体见[运行结果](#)展示）
- “2nd network model”：选择二级算法模型。
示例模型：- vehicle-license-plate-detection-barrier-0106-fp16

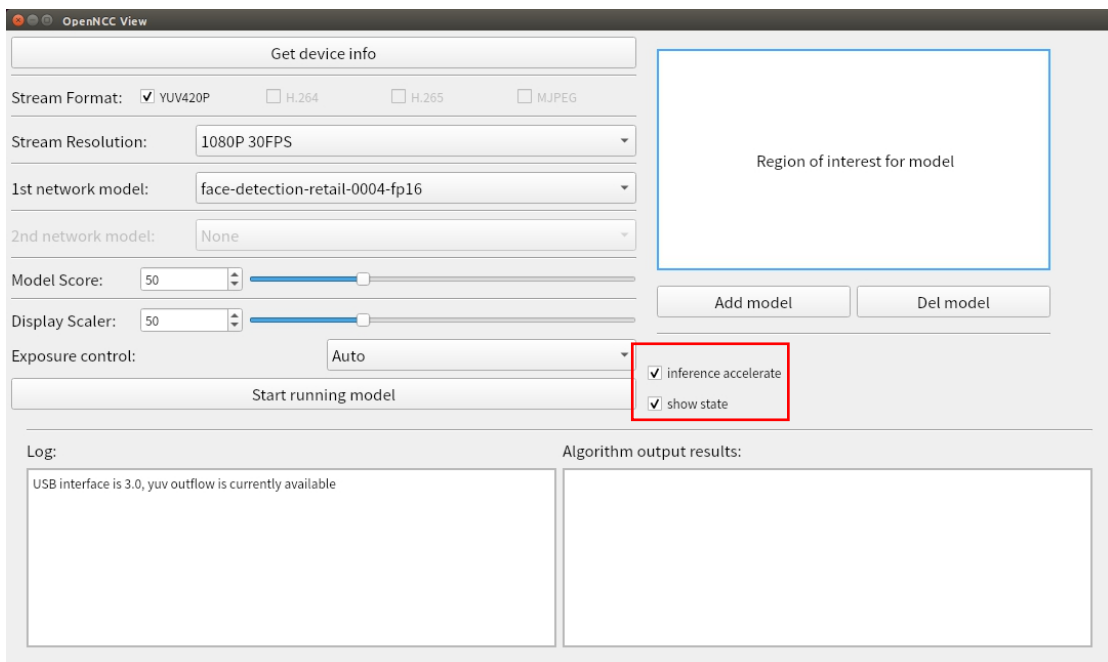


- “Model Score”：设置算法识别的最低分数，达到阈值后才会画面中框选出识别结果。
- “Display Scaler”：设置视频显示窗体大小，可以调节显示窗口分辨率。
- “Exposure control”：设置曝光，可以选择 Auto 和 Manual，当选择 Manual 时，可以自己设置 Exposure times 和 iso。

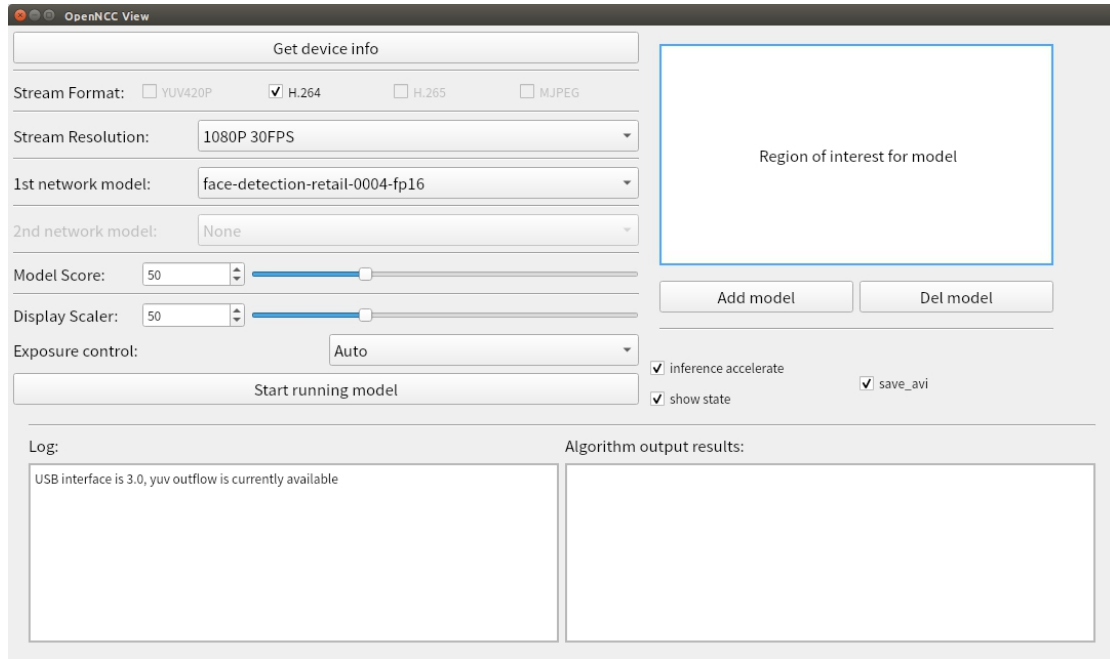




- 勾选“show state”，选择是否在画面上显示当前状态信息，包括视频流帧率、算法帧率、分辨率、设备 id。
- 勾选“inference accelerate”，选择是否启用算法加速。（必须在加载算法模型前选择）。



- 勾选“save_avi”，视频将在打开视频流后，以时间命名的 avi 文件保存到目录 openncc/Platform/Linux/Viewer/OpenNcc_Linux /avi 文件夹下。当关闭视频流后，视频会停止保存。（yuv420p 没有此项）



- 点击“Start running models”，打开视频流。运行结果见 [2.5](#)

2.2 快速入门之 Windows

2.2.1 环境搭建

- 进入 openncc/Platform/Windows
- 双击运行 `build_windows.bat`
- 结果如下：
 - └─ Windows
 - └─ Source
 - └─ Viewer
 - └─ build_windows.sh
 - └─ README.md
- 脚本运行成功后，进入 openncc/Platform/Windows/Drivers 目录，安装 Windows USB 驱动。详细安装步骤见 [OpenNCC_USB_Driver_install_guide_win.pdf](#)。

2.2.2 OpenNCC_Windows 操作演示

- 进入 openncc/Platform/Windows/Viewer/OpenNcc_Windows 目录。
- 双击运行 `OpenNCC.exe`。
- 后续操作同 [OpenNCC_Linux](#)。

2.3 快速入门之 Raspberry Pi

2.3.1 环境搭建

- 打开终端
- 进入 `openncc/Platform/Linux/RaspberryPi`
- 输入命令 `./build_raspberrypi.sh`

运行成功后，会生成目录如下：

```
└─ RaspberryPi
   └─ Example
   └─ Source
   └─ Viewer
   └─ build_raspberrypi.sh
   └─ README.md
```

2.3.2 OpenNCC_Raspberry 操作演示

提示：运行 OpenNCC_Raspberry 之前，请先查看设备号并[联系我们](#)获取密钥，设备号获取方法详见 `openncc/Platform/Linux/RaspberryPi/README.md`

- 将密钥文件（`eyecloud.key`）复制到目录
`openncc/Platform/Linux/RaspberryPi/Viewer/OpenNcc_Raspberry/Configuration/fw`
- 回到目录
`openncc/Platform/Linux/RaspberryPi/Viewer/OpenNcc_Raspberry`
- 打开终端，执行 `sudo ./AppRun`，启动软件。
- 后续操作同 [OpenNCC_Linux](#)。

2.4 快速入门之 NVIDIA

2.4.1 环境搭建

- 打开终端
- 进入 openncc/Platform/Linux/NVIDIA
- 输入命令 `./build_nvidia.sh`

运行成功后，会生成目录如下：

```
└─ NVIDIA
    └─ Example
    └─ Source
    └─ build_nvidia.sh
    └─ README.md
```

2.5 快速入门之 YOLO 数据的云端推送

提示：本操作基于上述几个平台的 opencv, libusb 等基础环境已经搭建完毕

2.5.1 环境搭建

- 在此基础上，针对 ubuntu 和 Raspberry

install curl:

```
cd ~  
wget http://curl.haxx.se/download/curl-7.57.0.tar.gz  
tar -xzf curl-7.57.0.tar.gz  
cd curl-7.57.0/  
./configure  
make  
sudo make install
```

树莓派环境下 make 时可能出现报错，如果有，请执行如下操作：

```
cd curl-7.57.0/lib  
打开 content_encoding.c 文件  
将第 549 行删除
```

- 针对 NVIDIA

1.install zlib:

```
$ down from http://zlib.net/zlib-1.2.5.tar.gz  
$ tar xzf zlib-1.2.5  
$ cd zlib-1.2.5  
$ mkdir zlib_install  
$ export CC=gcc  
$ ./configure --prefix=/home/zlib-1.2.5/zlib_install  
$ make -j8  
$ make install
```

2.install openssl:

```
$ down from https://www.openssl.org/source/openssl-3.0.1.tar.gz
```



```
$ tar xzf openssl-3.0.1.tar.gz
$ cd openssl-3.0.1
$ mkdir openssl_install
$ ./config no-asm shared
--prefix=/home/openssl-3.0.1/openssl_install
$ make -j8
$ make install
```

3.install curl:

```
$ wget http://curl.haxx.se/download/curl-7.57.0.tar.gz
$ tar -xzvf curl-7.57.0.tar.gz
$ cd curl-7.57.0/
$ ./configure --with-openssl=/home/openssl/openssl_install
$ make
$ make install
```

2.5.1 简介

该程序旨在帮助用户方便地在 OpenNCC 上使用 YOLO 模型，包括 YOLO 的训练和转化，模型在 OpenNCC 设备上的部署，识别结果的打包并上传云端。

提示：本演示已带默认生成的 yolo 模型数据。

2.5.2 操作演示

- 打开终端
针对不同平台进入 openncc/Platform/NVIDIA
或者 openncc/Platform/Raspberry
或者 openncc/Platform/Ubuntu
- 相对应输入命令 `./build_nvidia.sh` 或者 `build_raspberrypi.sh` 或者 `./build_ubuntu.sh`。
- 运行成功后，会相应生成目录
- 进入 `Example/How_to/OpenNCC/C&C++/How_to_use_sdk` 目录
- 输入命令 `make all`
- 编译成功后进入 bin 目录 `cd ./bin`
- 输入命令 `sudo ldconfig`
- 输入命令 `sudo ./OpenNCC 5` 执行程序

提示：程序首次运行请查看终端打印获取该设备的设备号，请联系我们获取密钥为该设备开通云端权限。

```
test example:yoloV4
sdk version:0 OpenNCCSV02.103.07_Nov 29 2022
./moviUsbBoot ./fw/OpenNcc.mvcmd
Performing bulk write of 9903056 bytes from ./fw/OpenNcc.mvcmd...
Successfully sent 9903056 bytes of data in 507.580625 ms (18.606483 MB/s)
device opened
1D6B:0003 (usbver:30, bus 2, device 1)
0CF3:E500 (usbver:20, bus 1, device 4) path: 7
inEndPoint:[82]
outEndPoint:[2]

5986:210E (usbver:20, bus 1, device 3) path: 5

046D:C52F (usbver:20, bus 1, device 2) path: 3

03E7:F63D (usbver:21, bus 1, device 8) get our self usb device ver:21
path: 2
outEndPoint:[1]
inEndPoint:[81]
outEndPoint:[2]
inEndPoint:[82]
outEndPoint:[3]
inEndPoint:[83]

1D6B:0002 (usbver:20, bus 1, device 1)
Firmware version: MV02.101.26 Device SN:0020546618442010
usb version:21
10:21:16 : sdk/sdk.cpp(666) enter watchdog task....
list num:1
[0/1]camera: sc2232, 1920X1080@30fps, AFmode:0, maxEXP:33333us,gain[100, 1600]

-----
Camera detection succeeded!(sc2232)
```

关于 YOLO 模型训练

介绍如何进行模型训练，训练获得的数据提供给 [OpenNCC_YOLO](#) 使用。

安装环境依赖

CMake >= 3.18:

<https://cmake.org/download/>

Powershell (already installed on windows):

<https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell>

CUDA >= 10.2 (可选) :

<https://developer.nvidia.com/cuda-toolkit-archive>

cuDNN >= 8.0.2 (可选) :

<https://developer.nvidia.com/rdp/cudnn-archive>

(on **Linux** follow steps described here) :

```
https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#installlinux-tar
```

(onWindows follow steps described here) :

```
https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#installwindows)
```

GPU with CC \geq 3.0 (可选) :

```
https://en.wikipedia.org/wiki/CUDA#GPUs_supported
```

训练工具编译

```
git clone https://github.com/AlexeyAB/darknetcd darknet
mkdir build_released build_release
cmake ..
cmake --build . --target install --parallel 8
```

准备数据集

- 训练集图片放在 train 文件夹内，验证集放在 val 文件夹内

数据标注

```
git clone https://hub.fastgit.org/AlexeyAB/Yolo_mark.git
cmake .
make
./linux_mark.sh
```

- 关于标注的使用方法可查看 Yolo_mark 目录内的 readme.md

参数配置

- 除两个数据集外，启动训练还需要配置几个参数文件，包括
- obj.data , obj.name , train.txt。

以上三个文件,会在数据标注时自动生成`Yolo_mark/x64/Release/data`目录下：

obj.name 文件包含所有目标的类别名，train.txt 包含所有训练图片路径，val.txt 非必须，可以手动从 train 文件中分割出 30% 的图片用于验证。而 obj.data 文件申明了上述所有文件的路径和类别总数，如果使用自己的数据集，对应参数修改请在标注前完成修改。

● yolo.cfg(拓扑)

Yolo.conv (预训练权重)

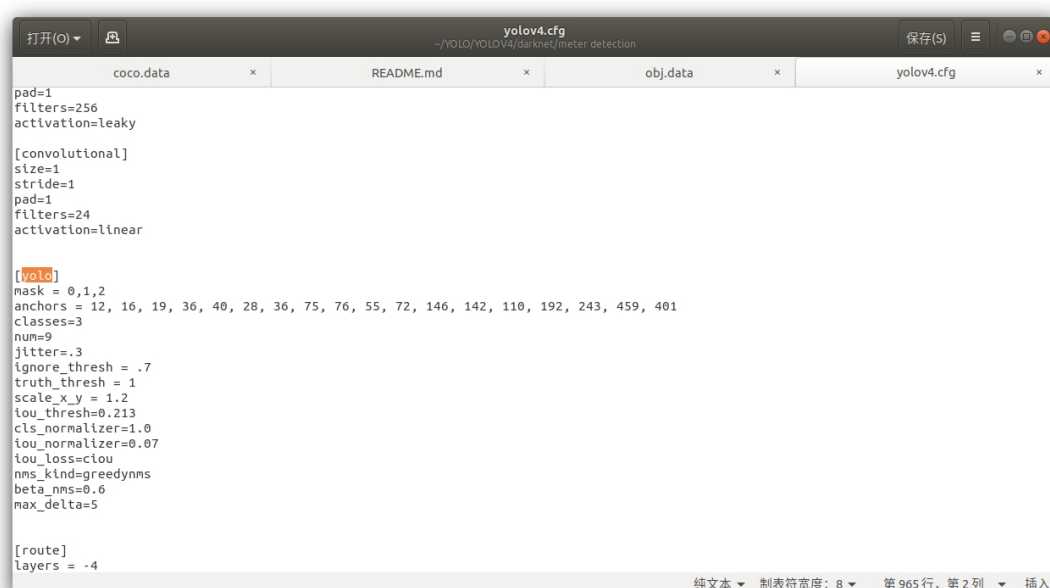
cfg 和 conv 存在一定对应关系，考虑此处训练的模型最终需部署在 openncc 上，推荐使用 (yolov4-tiny.cfg+yolov4-tiny.conv.29) 或 (yolov3-tiny.cfg+yolov3-tiny.conv.11) 的搭配，cfg 文件可以直接在 darknet/cfg 目录下找到。

● Cfg 文件修改

如果目标类别数量不等于 80，则必须修改 cfg 文件。

搜索 cfg 文件中所有 yolo 层的位置，若总共有 3 类目标，则将[yolo]层 classes 参数定义为 3，再将[yolo]上一层[convolutional]层的 filters 定义为 24。计算方式为 $\text{filters} = (\text{classes} + 5) * 3$ 。

对 yolov4-tiny.cfg 来说有两个 yolo 层，所以一共需要修改 4 个参数。



```

pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear

[yolo]
mask = 0,1,2
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=3
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
scale_x_y = 1.2
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5

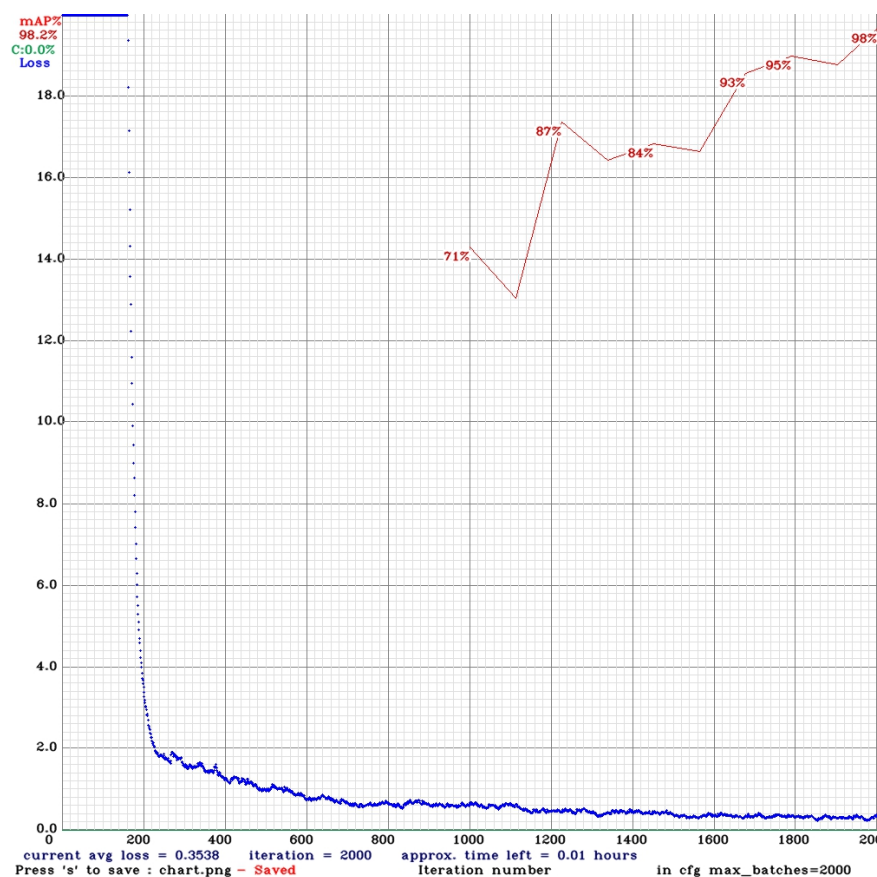
[route]
layers = -4
  
```

启动训练

第二步编译成功后，会在 darknet 目录下生成 ./darknet 工具。

输入命令：

```
./darknet detector train ./obj.data ./yolov4-tiny.cfg ./yolov4-tiny.conv.29
```



如果你使用显卡训练，显存小于 8G 的显卡，需要在 cfg 的第一层[net]中将 batch 参数改到 8 以下（8，4，2，1）。

若训练进行顺利，可看到如下图的训练日志图表。

训练结束后，可看到一系列.weights 文件。这里还是建议制作数据集时设置一个验证集，这样可以直接锁定验证集中 map 最高的权重 yolov4-tiny_best.weights 作为后续使用。

● Darknet to tensorflow

```
git clone https://github.com/RenLuXi/tensorflow-yolov4-tiny.git
cd tensorflow-yolov4-tiny
python convert_weights_pb.py --class_names obj.names --weights_file yolov4-tiny_best.weights --tiny
```

这里需要用到第五步中 obj.names 和 yolov4-tiny_best.weights。

● Tensorflow to IR(opencvino 推理格式)

修改 json 配置文件

打开 tensorflow-yolov4-tiny 目录下的 yolo_v4_tiny.json，将其中的 classes 值修改为你自己的类别数，opencvino 进行 tensorflow 转换需要用到这个文件。

然后替换 json 配置文件

```
cp ./yolo_v4_tiny.json
```

```
/opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf
```

进入 openvino 模型转换工具目录

```
cd /opt/intel/openvino/deployment_tools/model_optimizer
```

转换命令

```
python mo.py --input_model yolov4-tiny.pb
```

```
--transformations_config ./extensions/front/tf/yolo_v4_tiny.json
```

```
--batch 1 --data_type FP32 --reverse_input_channels
```

● IR to blob

老办法，先初始化 openvino 环境，然后把上一步生成的 xml 和 bin 文件丢过去转换

```
source /opt/intel/openvino_2020.3.194/bin/setupvars.sh
```

```
cd
```

```
/opt/intel/openvino_2020.3.194/deployment_tools/inference_engine/lib/intel64
```

```
cp
```

```
/opt/intel/openvino/deployment_tools/model_optimizer/yolov4-tiny.xml ./
```

```
cp
```

```
/opt/intel/openvino/deployment_tools/model_optimizer/yolov4-tiny.bin ./
```

```
/opt/intel/openvino_2020.3.194/deployment_tools/inference_engine/lib/intel64/myriad_compile -m yolov4-tiny.xml -o yolov4-tiny.blob  
-VPU_MYRIAD_PLATFORM VPU_MYRIAD_2480  
-VPU_NUMBER_OF_SHAVES 6 -VPU_NUMBER_OF_CMX_SLICES 6
```

使用模型

- 转化完成后，至此，已经获得了在 OpenNCC 上部署所需的全部模型文件，xml，bin 和 blob。

2.6 Custom（自定义）

2.6.1 环境搭建

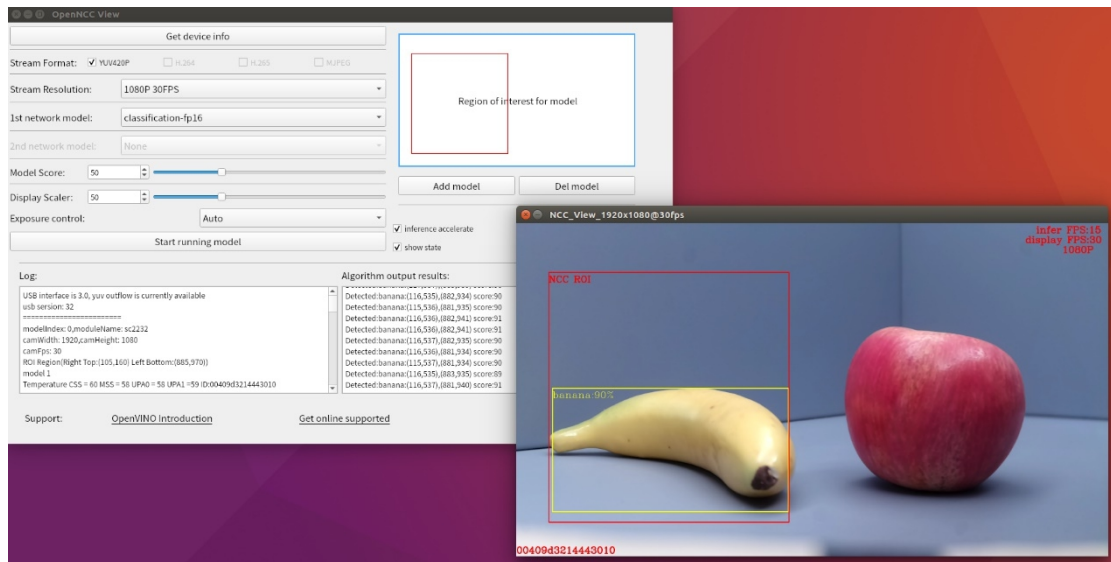
- 进入 openncc/Platform/Custom
- 右键打开终端
- 输入命令 `./build_custom.sh`

提示：如需在自定义平台运行，请[联系我们](#)获取定制化服务。

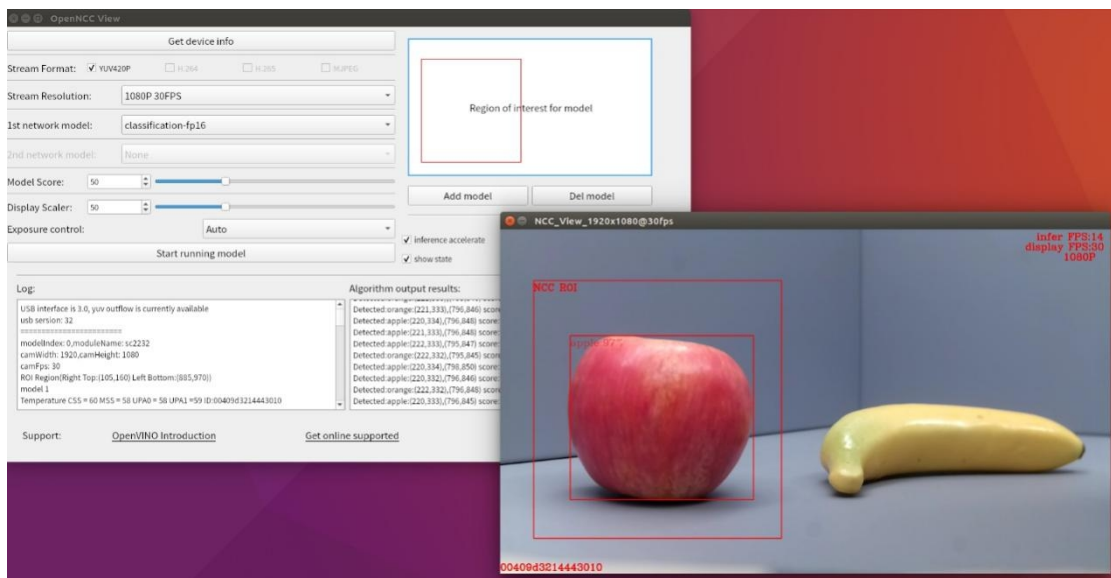
2.7 运行结果演示

以物体分类算法模型为例：

香蕉在算法区域内结果：



苹果在算法区域内结果：



第三章 软件概述

本章介绍了 SDK 中包含了应用必须的资源库及相关的基本应用例程。

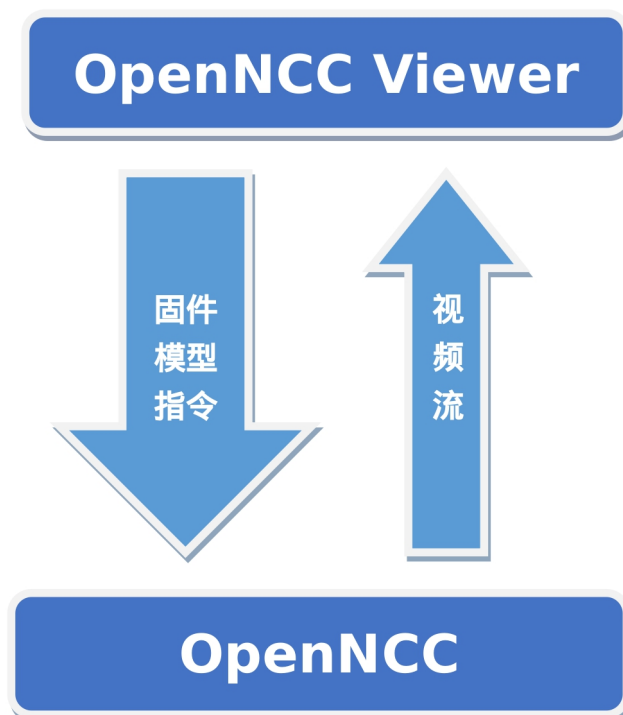
3.1 资源库

SDK 的资源库中包含了 Firmware (固件), Library (OpenNCC API 库), Model (AI 模型) 三类文件。

目录	内容
Source/Firmware	包含 OpenNCC 必要固件。
Source/Library	包含不同语言编译的 OpenNCC API 库 及固件引导程序。
Source/Model	包含已集成的不同类型 AI 模型。

3.1.1 OpenNCC 固件

OpenNCC 固件是 OpenNCC 运行的基础。OpenNCC 需要通过应用程序调用 moviUsbBoot (Movidius 官方引导程序) 下发固件, 才能够引导启动。



3.1.2 OpenNCC API 库

OpenNCC API 库是应用程序与 OpenNCC 连接的桥梁。应用程序通过 API 库下发指令给 OpenNCC ,OpenNCC 也通过 API 库输出视频流及 AI 算法结果。

3.1.3 AI 模型

目前，OpenNCC 已集成的 AI 模型及简单描述如下：

模型类别	名称	简介
物体分类	classification-fp16	ssd_mobilenet_v1_coco model can detect almost 90 objects
人脸、人形检测	face-detection-adas-0001-fp16	Face detector for driver monitoring and similar scenarios. The network features a default MobileNet backbone that includes depth-wise convolutions to reduce the amount of computation for the 3x3 convolution block
	face-detection-retail-0004-fp16	Face detector based on SqueezeNet light (half-channels) as a backbone with a single SSD for indoor/outdoor scenes shot by a front-facing camera
	face-person-detection-retail-0002-fp16	This is a pedestrian detector based on backbone with hyper-feature + R-FCN for the Retail scenario
	person-detection-retail-0013-fp16	This is a pedestrian detector for the Retail scenario. It is based on MobileNetV2-like backbone that includes depth-wise

模型类别	名称	简介
		convolutions to reduce the amount of computation for the 3x3 convolution block
	pedestrian-detection-adas-0002-fp16	Pedestrian detection network based on SSD framework with tuned MobileNet v1 as a feature extractor.
人车、自行车	person-vehicle-bike-detection-crossroad-0078-fp16	Person/Vehicle/Bike detector is based on SSD detection architecture, RMNet backbone, and learnable image downscale block (like person-vehicle-bike-detection-crossroad-0066, but with extra pooling)
	pedestrian-and-vehicle-detector-adas-0001-fp16	Pedestrian and vehicle detection network based on MobileNet v1.0 + SSD.
车辆检测	vehicle-detection-adas-0002-fp16	This is a vehicle detection network based on an SSD framework with tuned MobileNet v1 as a feature extractor.
车牌识别	vehicle-license-plate-detection-barrier-0106-fp16	This is a MobileNetV2 + SSD-based vehicle and (Chinese) license plate detector for the "Barrier" use case.
人脸属性	interactive_face_detection_demo	This demo executes four parallel infer requests for the Age/Gender Recognition, Head Pose Estimation, Emotions

模型类别	名称	简介
		Recognition, and Facial Landmarks Detection networks that run simultaneously
人体骨骼提取	human-pose-estimation-001-fp16	A multi-person 2D pose estimation network (based on the OpenPose approach) with tuned MobileNet v1 as a feature extractor.

3.2 应用例程

SDK 中包含了 How_to 及 Linkage_demo 两类例子,下面表格中包含了对示例的简单描述。详细开发细节见各平台及例程目录内 ReadMe 及 [OpenNCC_SDK_API.pdf](#)。

提示 : ReadMe 推荐用 **Typora** 查看 下载地址 : <https://www.typora.io/>

目录	内容
Example/How_to/... /How_to_use_sdk	示例程序, 如何在项目中使用 SDK 库。
Example/How_to/... /Capture_video	示例程序, 使用 SDK 库获取视频流。
Example/How_to/... /Load_a_model	示例程序, 使用 SDK 库下载一个 Blob 格式的深度学习模型。
Example/How_to/... /work_with_multiple_models	示例程序, 二级模型的应用。
Example/How_to/... /python3	Python 的相关示例。
Example/How_to/OpenNCC-EMMC	包含 eMMC 版本的相关示例程序。
Example/Linkage_demo/work with AlwaysAI / pedestrian_tracking_demo	人脸模型, 使用 AlwaysAI 解析结果显示, 并统计通过识别区域的人数。
Example/Linkage_demo/work_with_OpenVINO/human_pose_estimation_demo	人体骨骼模型, 使用 OpenVINO 解析结果显示。
Example/Linkage_demo/work_with_OpenVINO/interactive_face_detection_demo	人脸、年龄、性别、心情模型, 使用 OpenVINO 解析结果显示。
Example/Linkage_demo/work_with_PaddlePaddle/	OCR 文字识别示例, 包含指向 OpenNCC PaddlePaddle-OCR 仓库的链接。

3.3 OpenNCC 运行机制介绍

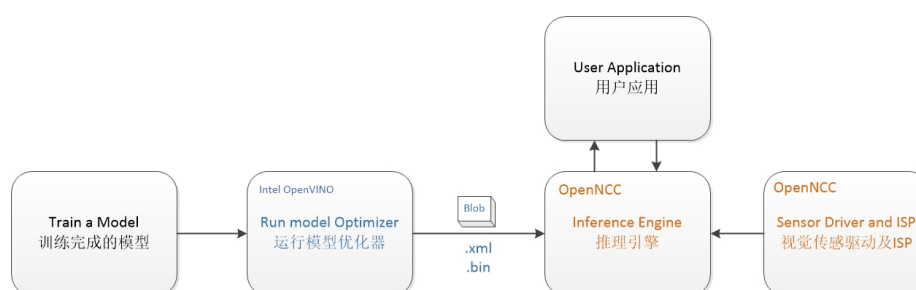
从一个模型训练环境到嵌入式部署, 是一个非常重要的工作, 需要对深度学习的框架掌握, 如常用的: Caffe*, TensorFlow*, MXNet*, Kaldi* 等。此外掌握部署的嵌入式平台非常重要, 需要了解平台性能, 系统架构特点, 结合平台特点需要对训练的模型框架进行优化, 并最后调优移植部署到嵌入式平台。OpenNCC 专注于深度学习模型的快速部署, 兼容 Intel OpenVINO, 并针对嵌

入式图形图像应用场景,在端侧完成了从 2MP 到 20MP 不同分辨率传感器集成,端侧实现了可部署专业级别的 ISP,可将 OpenVINO 优化转换后的模型文件动态下载到端侧 OpenNCC 相机,实现深度学习模型的快速部署。同时 OpenNCC 设计了独立工作模式、混合开发模式和协处理计算棒模式来适配不同的工作应用场景。

3.3.1 OpenNCC 独立模式

独立模式下, OpenNCC 独立运行一个深度学习模型,并将推理结果通过 OpenNCC CDK API 反馈给用户。

应用程序部署流程如下图:



按照 OpenVINO 文档,为特定的训练框架[配置模型优化器\(Configure Model Optimizer\)](#), [运行模型优化器\(Model Optimizer\)](#), 基于训练好的网络拓扑、权值和偏差值等可选参数产生一个优化后的 IR 文件, IR 是一对描述整个模型的文件, 包括.xml 文件--拓扑文件-描述网络拓扑的 XML 文件, 以及.bin 文件--训练后的数据文件-一个包含权重并偏置二进制数据的.bin 文件, 然后再运行 myriad_compile 将 IR 文件生成 BLOB 文件。

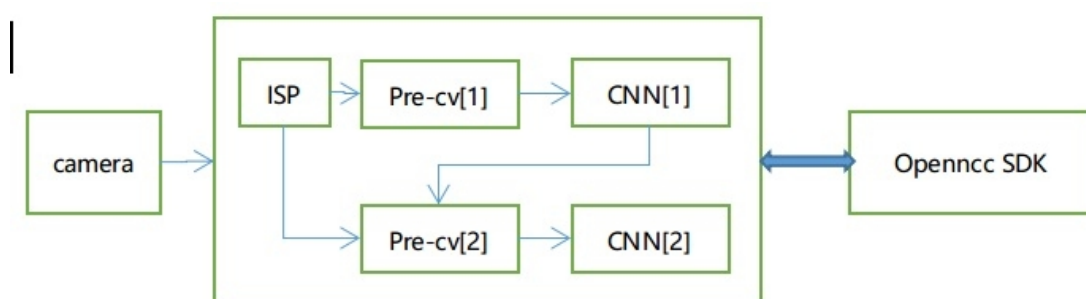
在应用程序上,集成使用 OpenNCC SDK 下载优化完成后的 BLOB 模型文件, 见 SDK 下 Example/How_to/OpenNCC/C&C++/Load_a_model 的演示程序。

OpenNCC View 是集成了 OpenNCC SDK 的带操作界面的应用演示程序, 也可以使用 OpenView 来部署模型, 获取测试结果。

由于不同的深度模型有差异化的推理输出结果，OpenNCC SDK 对不同格式结果支持在不断增加中，如果用户无法在 CDK 下找到合适的后处理解析模型，需要自己参考 Example/How_to/OpenNCC/C&C++/Load_a_model 并结合自己应用场景来编写后处理代码。

3.3.1.1 二级模型运行

考虑到端侧算力，目前 CDK 多级模型支持到两级模型级联，如图：



第一级模型必须为目标检测或者分类模型，且输出定义如下：

1. The net outputs blob with shape: [1, 1, N, 7], where N is the number of detected bounding boxes. For each detection, the description has the format: [image_id, label, conf, x_min, y_min, x_max, y_max]
 - image_id - ID of the image in the batch
 - label - predicted class ID
 - conf - confidence for the predicted class
 - (x_min, y_min) - coordinates of the top left bounding box corner
 - (x_max, y_max) - coordinates of the bottom right bounding box corner.

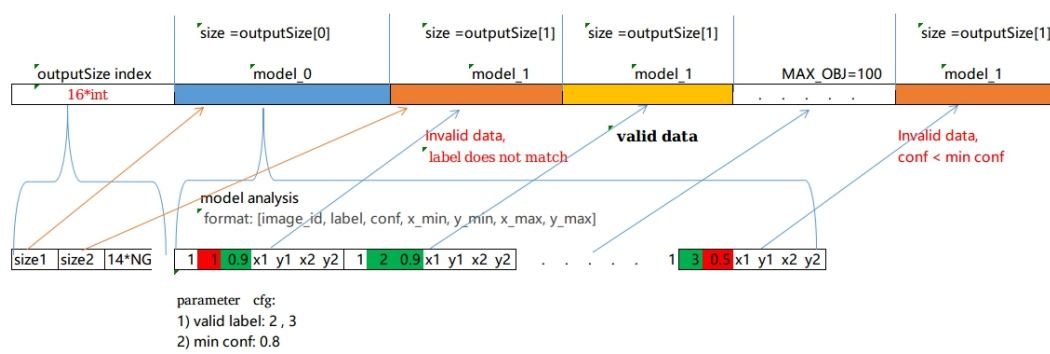
推理流程：

1) 图像先经过 Pre-cv[1]，把原图 scale 到一级模型输入大小，并做相应的格式转换，然后做一级模型推理计算，并且把一级推理结果输出到 Pre-cv[2]。

2) Pre-cv[2] 模块解析第一级模型的推理结果，把符合条件的 label 和 conf 的检测目标，根据坐标起点(x_min, y_min)，终点(x_max, y_max)从原图 crop 和 scale 到二级模型输入大小，并做相应的格式转换，进入第二级模型推理。

3) 最后把一级模型和全部的二级模型推理结果打包在一起输出。

模型输出解析（图示参数配置为，有效 label：2,3，conf=0.8）：



示例程序：Example/How_to/... /Multiple_models,第一级模型为车辆和车牌检测，第二级模型是车牌检测，设置有效的 label 为 2。

基于第一级的检测结果，适当微调第一级的检测坐标，有利于识别：

*起点向左上方微调（startXAdj，startYAdj）

*底点向右下方微调（endXAdj，endYAdj）

cnn2PrmSet.startXAdj = -5;

cnn2PrmSet.startYAdj = -5;

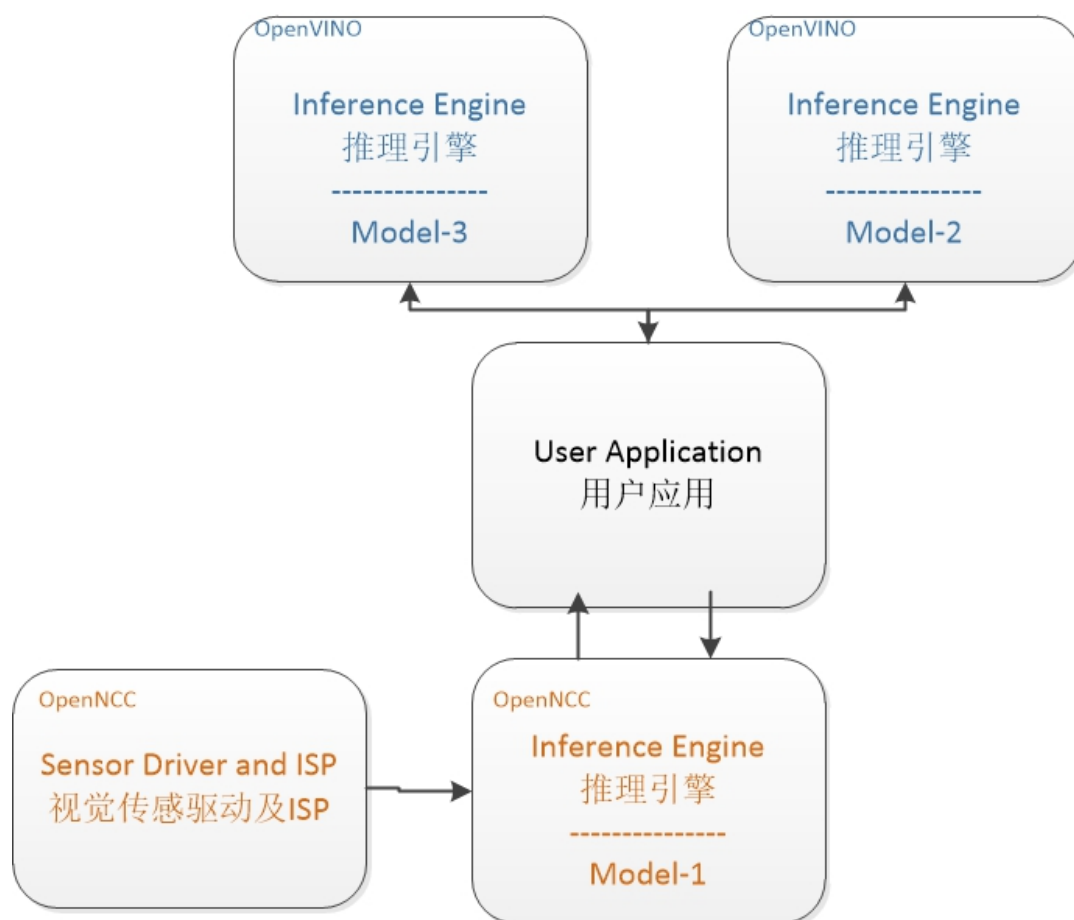
cnn2PrmSet.endXAdj = 5;

cnn2PrmSet.endYAdj = 5;

3.3.2 OpenNCC 混合模式

当需要解决一些复杂应用场景，需要多个网络模型组合处理、OpenNCC 端侧计算性能无法满足、或者端侧处理完成后需要到边缘侧集中后处理时，往往需要进行系统扩增。将实时性诉求高的模型运行在 OpenNCC 端侧，其他模型运行在后处理边缘机或云端。

如图 ,Model-1 运行在 OpenNCC 端侧 ,完成对视频流的前处理。OpenNCC

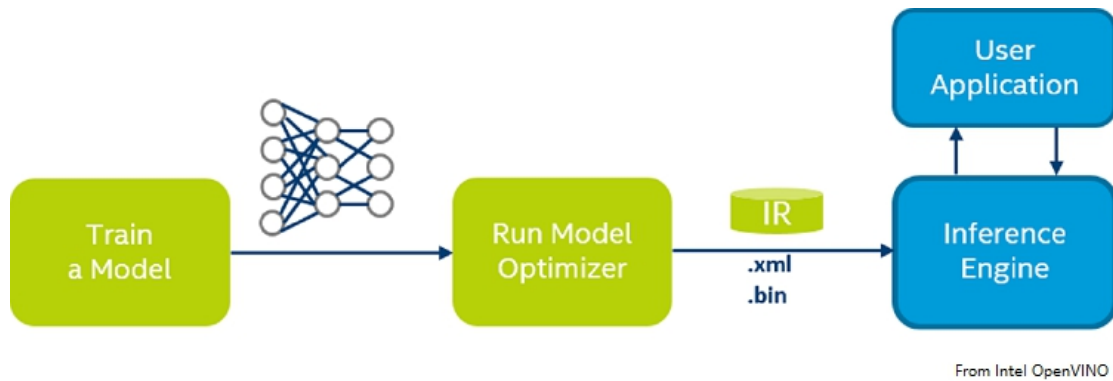


将一级处理模型结果返回用户应用程序，Model-1 和 Model-2 完全运行于 OpenVINO 推理引擎下，实现后续处理。

Examples/Linkage_demo/work_with_OpenVINO 演示了如何让 OpenNCC 和 Host PC 上 OpenVINO 组合实现一个分布式 AI 系统。

3.3.3 协处理计算棒模式

OpenNCC 的协处理模式，类似与 Intel NCS2 计算棒。这种工作模式下，OpenNCC 的视觉传感器不工作，用户可以单独使用 OpenNCC 来实现完全兼容 OpenVINO 环境。OpenVINO 典型的深度学习模型部署流程如下：



按照 OpenVINO 文档，为特定的训练框架[配置模型优化器\(Configure Model Optimizer\)](#)产生一个优化后的 IR 文件，基于训练好的网络拓扑、权值和偏差值等可选参数。

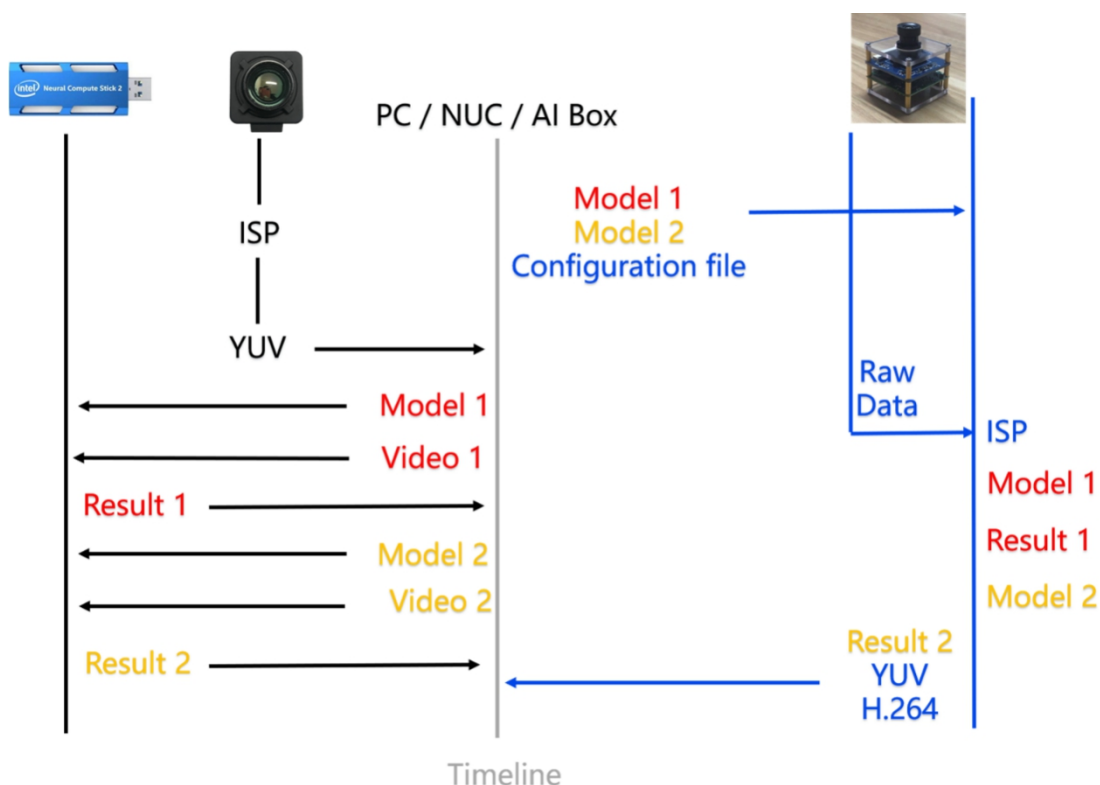
将优化生成的 IR 文件下载到 OpenNCC 上运行[推理引擎\(Inference Engine\)](#)，具体参考 OpenVINO 文档：[Inference Engine validation application](#) 和 [sample applications](#)。将 Source/Firmware/fw

/usb-ma2x8x.mvcmd 复制并且替换 openvino 安装目录下的
openvino/inference_engine/lib/intel64/usb-ma2x8x.mvcmd

提示：替换前必须备份 usb-ma2x8x.mvcmd，使用 NCS2 推理时需要恢复该文件

3.3.4 独立模式和协处理模式区别

如下图右侧是 OpenNCC 的独立模式，左侧是 OpenNCC 的协处理模式(类



同 Intel NCS2)。

当我们需要部署一个基于视觉的深度学习模型时，首先我们需要获取一个高质量的视频流，然后运行推理引擎来把输入的图像数据进行计算，最后输出结果。

左侧的协处理模式，我们需要一个 OpenNCC 或者 Intel NCS2 实现端侧推理，同时我们需要从一个摄像机获取视频流，并将视频帧通过 USB 发送给 OpenNCC。

而右侧的独立模式，不需要额外的摄像机来获取视频流，我们只需要将模型下载到 OpenNCC 后，就可以获取到推演结果。

参考 OpenVINO 官网：<https://docs.openvinotoolkit.org/>

常见问题

Q: 其他版本的 OpenVINO 被支持吗？

除了 2020.3.194 被 OpenNCC 正式支持外,其他新版本的 OpenVINO 也可以被 OpenNCC 支持,但是未做完整的覆盖性评测,部分模型存在运行异常情况。

Q: 如何确保 OpenNCC 相机开启了固件算法加速？

首先只有搭配 AR0234 模组的 OpenNCC 相机才能开启固件双引擎算法加速功能,使用 OpenNCC View 软件,可以直接通过勾选"inference acceleration"框激活,如果开发者需要自己开发的程序当中开启固件算法加速,可以在 sdk 初始化时将传入结构体 Network1Par 的末位成员 inferenceACC 置 1。除了观察画面帧率变化,开启加速时也可以测试 USB 接口处变高的电压确认。

Q: 固件双引擎算法加速有何效果？

固件双引擎算法加速对 OpenNCC View 当中所有的模型都有不俗的效果,大部分能提升近百分百的性能,0004 人脸识别模型在测试过程中可以稳定在 80 帧左右,而未开启加速时只能达到 43 帧。此外,开发者基于 OpenNCC 软件开发框架调用的模型都可以被正确作用。

Q: OpenNCC 支持的 OpenVINO 支持哪些深度框架和模型？

Caffe* (most public branches)

TensorFlow*

MXNet*

Kaldi*

ONNX*

更多信息见：[Supported Frameworks and Formats](#)

Q: OpenNCC 有支持自启动的硬件版本吗？

开发完成后,如果部署需要 OpenNCC 脱离主控独立运行,有带 Flash 的量产版本,请向 OpenNCC 咨询。

Q: 我有模型转换和解析的技术问题，哪里可以得到支持？

[Intel · OpenVINO 中文社区](#)可以得到支持

Q: OpenNCC 支持开发者自己更换镜头吗？

对于一般应用，开发者可以根据自己场景选配不同的镜头。如果更换镜头后，图像质量不能满足场景的有严苛要求，可以向 OpenNCC 寻求技术定制支持。

Q: OpenNCC 支持 Ubuntu 虚拟机开发吗？

支持，需要把 OpenNCC 的 USB 设备在启动时就被虚拟机检测到，如果先被主机检测到，再手工切换到虚拟机会出现 OpenNCC 开发接口检索设备失败的通信异常。

Q: OpenNCC 定制的固件应当如何替换？

方法一（全局更新）：将接收到的 OpenNCC.mvcmd 替换 Source/Firmware/fw 中的 OpenNCC.mvcmd，然后执行[环境搭建](#)脚本，完成自动更新。

方法二（局部更新）：找到需要运行的工程下的 OpenNCC.mvcmd，直接替换即可。

Q：如何替换一个自己的模型？

具体方法详见：

[SDK/Viewer/QT_Package/doc/OpenNCC_AI_Model_Integration.pdf](#)

Q：USB 连接异常怎么办？

例如：log：libusb_bulk_transfer 3 read error!-1
请尝试重新插拔 USB 接口，并确保稳定连接。

Q：开发平台没有界面应该如何开发？

请直接阅读[应用例程](#)中的 ReadMe , 内含详细说明。