Eyecloud OpenNCC
Software Development Kit (SDK)

Getting Started Guide

June 2021

Revision 1.0.0

Technical Support

You can contact us through the channel on the official website.

https://www.openncc.com/contact

Contact information

Tel: +1-408-219-6489

E-mail: service@eyecloud.tech

Copyright statement

The copyright of this manual belongs to Hangzhou EyeCloud Technologies Co., Ltd. Without the written permission of our company, no unit or individual has the right to copy, transmit, or reproduce any content of this manual in any form. Violators will be held legally responsible.

Revision History

Version	Date	Editor	Description
1.0.0	June 2021	Zed	

Directory

Technical Support 2
Contact information 2
Copyright statement 2
Revision History 3
Directory 4
1 Introduction 6
1.1 Overview 6
1.2 SDK structure 6
1.3 Supported Products and Platforms 6
2 Getting Started Guide 7
2.1 Quick Start to Linux 7
2.1.1 Environment construction 7
2.1.2 OpenNCC_Linux Operation Demo 8
2.2 Quick Start to Windows 14
2.2.1 Environment construction 14
2.2.2 OpenNCC_Windows Operation Demo 14
2.3 Quick Start to Raspberry Pi 15
2.3.1 Environment construction 15
2.3.2 OpenNCC_Raspberry Operation Demo 15
2.4 Custom(Customization) 16
2.4.1 Environment construction 16
2.5 Run results demo 17
3 Software Overview 18
3.1 Resource library 18
3.1.1 OpenNCC Firmware 18
3.1.2 OpenNCC API Library 19
3.1.3 AI Models 19
3.2 Application Examples 22
3.3 OpenNCC Introduction of operation mechanism 23
3.3.1 OpenNCC standalone mode 23

3.3.2 OpenNCC mixed mode..... 25
3.3.3 Co-processing compute stick mode.....26
3.3.4 Difference between independent mode and co-processing mode..... 28

1 Introduction

1.1 Overview

This document is used to introduce the OpenNCC Software Development Kit (SDK) and contains all the necessary information to start, run and develop it.

1.2 SDK structure

Directory	Content
./Platform	Contains scripts for generating runtime environments for different platforms.
./docs	Contains SDK-related introduction and documentation.
./SDK/Drivers	Contains the drivers that must be installed for different platforms.
./SDK/Example	Contains SDK related routines.
./SDK/Source	Includes firmware, model and SDK library files.
./SDK/Tools	Includes relevant model conversion and compilation tools.
./Viewer	Includes compiled Viewer and QT source code.

1.3 Supported Products and Platforms

The SDK supports the following products:

- OpenNCC DK
- OpenNCC Lite
- OpenNCC USB

The SDK supports the following platforms:

- Ubuntu 16.04, Ubuntu 18.04
- Windows 10

- Raspberry Pi

Tip: Other platforms can contact us through the official technical support channel to achieve customized services.

2 Getting Started Guide

Go to openncc/Platform , the directory is as follows.

Select the required file directory to enter, the corresponding folder contains the scripts for environment building.

Warning: The environment builder script will automatically generate and overwrite related files, please make sure it is the first time you run it or have completed a backup before running it.

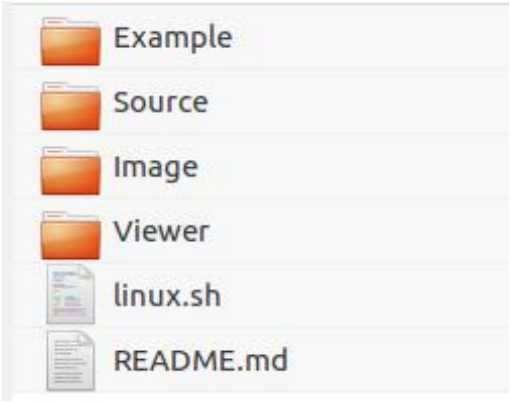
2.1 Quick Start to Linux

2.1.1 Environment construction

- Go to the directory: openncc/Platform/Linux
- Right click to open the terminal.
- Enter the command . /linux.sh

```
zed@zed:~/openncc/Platform/Linux$ ./linux.sh
Linux
Please make sure you have backed up what you need?(y or n)y
mkdir Source
copy Library ....
copy Model .....
copy Firmware .....
copy Example .....
copy Viewer ......
copy Linux lib to Viewer
```

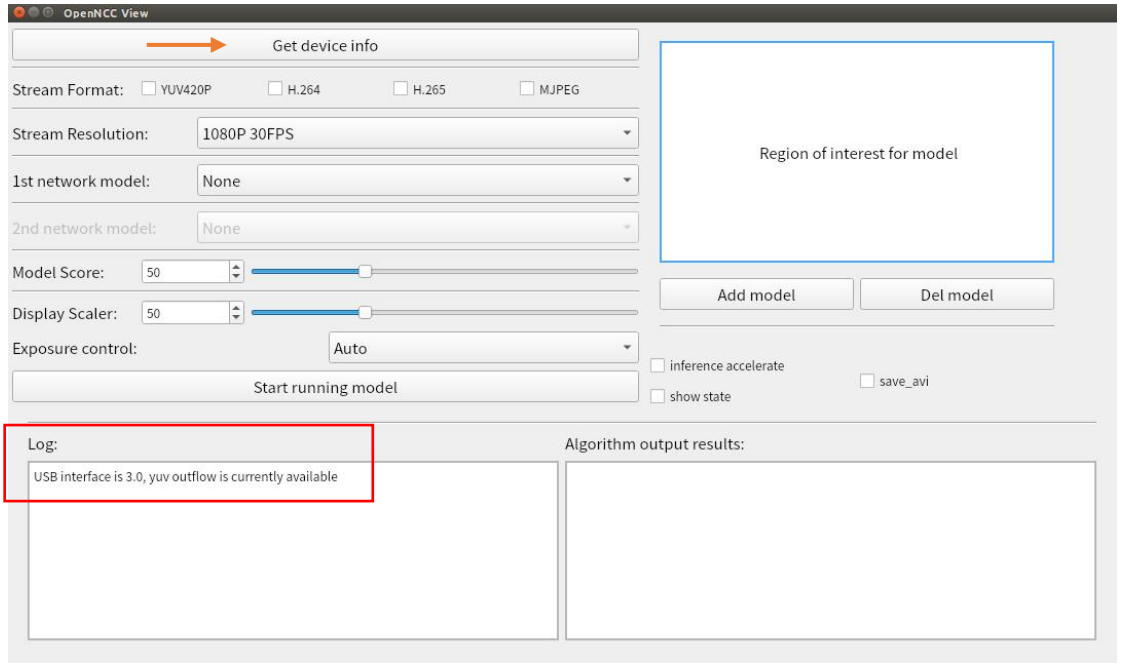
The generated directory is as follows.

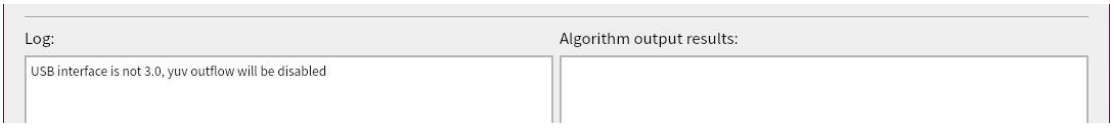


2.1.2 OpenNCC_Linux Operation Demo

- Go to the directory: openncc/Platform/Linux/Viewer/OpenNcc_Linux
- Right-click to open the terminal and execute `sudo . /AppRun` to start the software.
- Connect the OpenNCC camera to the USB 3.0 port of your computer and click on the Get device info button to get the device information, at this point there will be two prompts in the log area.
    - 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.

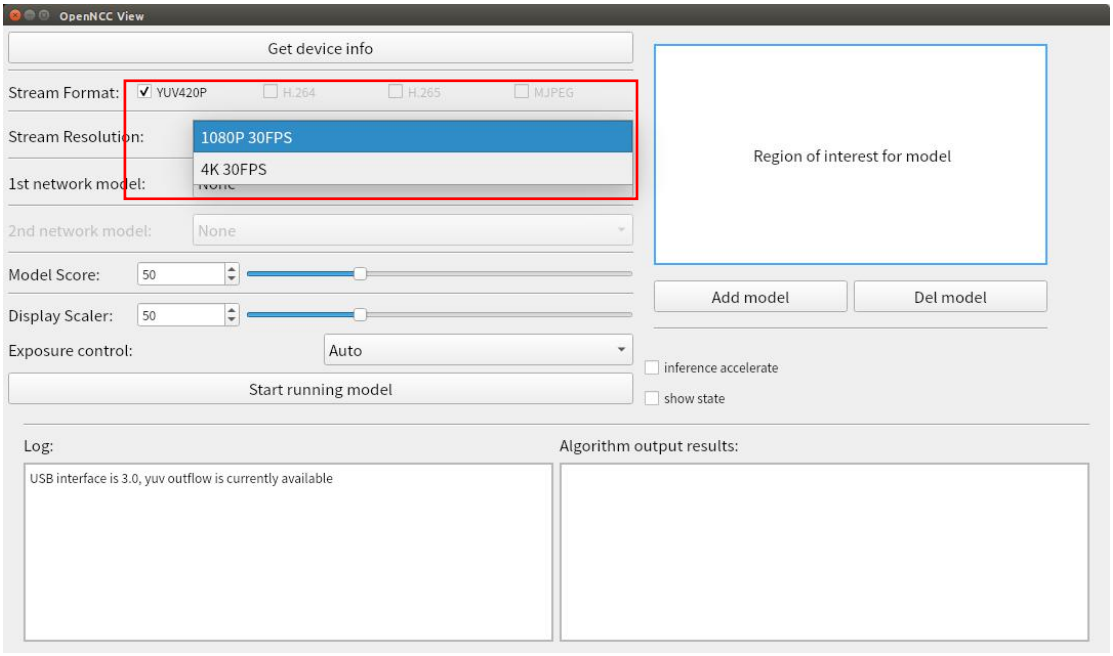
Tip: The OpenNCC TYPE-C interface has two modes of positive and negative access, corresponding to USB3.0 and USB2.0 respectively. The USB2.0 mode temporarily disables the YUV420P format video stream output due to the transfer rate.



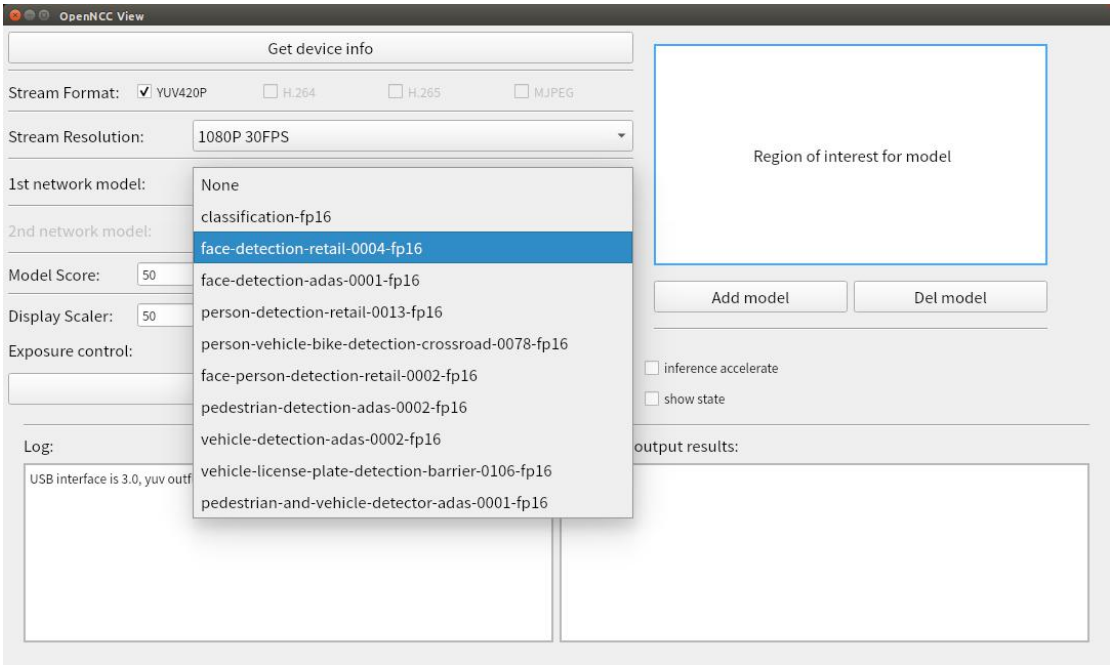


If you need to switch USB mode, rotate the TPYE-C port to access it again and click the Get device info button again. If there is still no change, please check your computer USB port type or contact us.

- Choose any of the video streaming formats yuv420p/H.264/H.265/mjpeg.
- ”Stream Resolution”: Two resolutions are available, 1080p and 4K.
  (Specifically determined by OpenNCC product type)



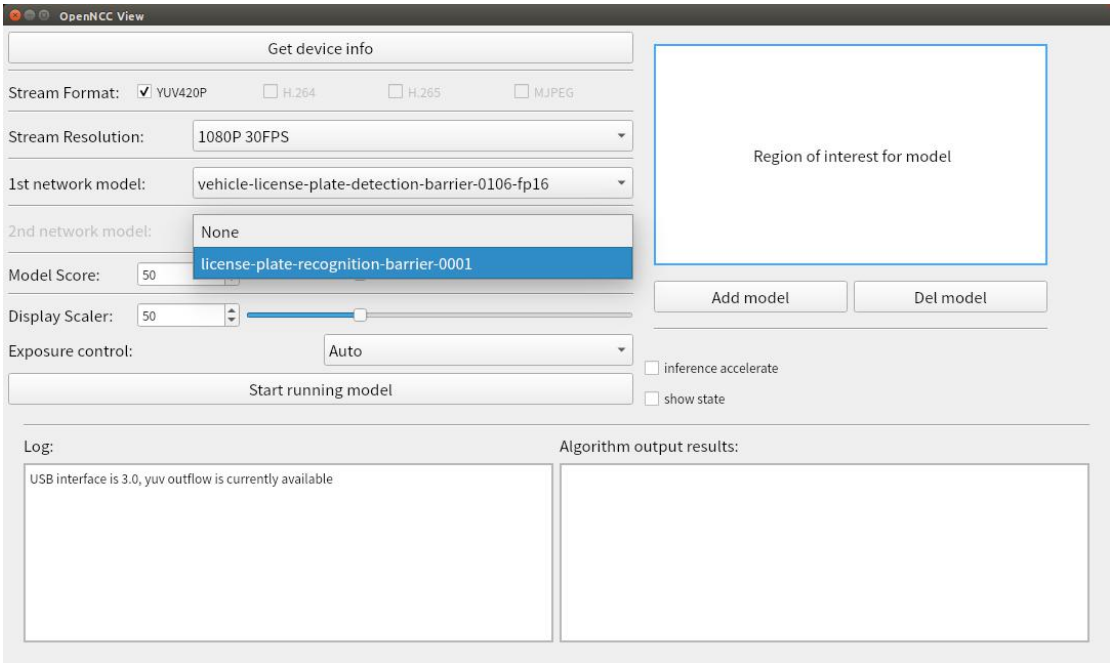
- "1st network model": Select the algorithm model. (See 3.1.3 for detailed model description)
  Currently supports more than 10 models to choose. If choose None that will not load models, only the original video stream could be displayed.



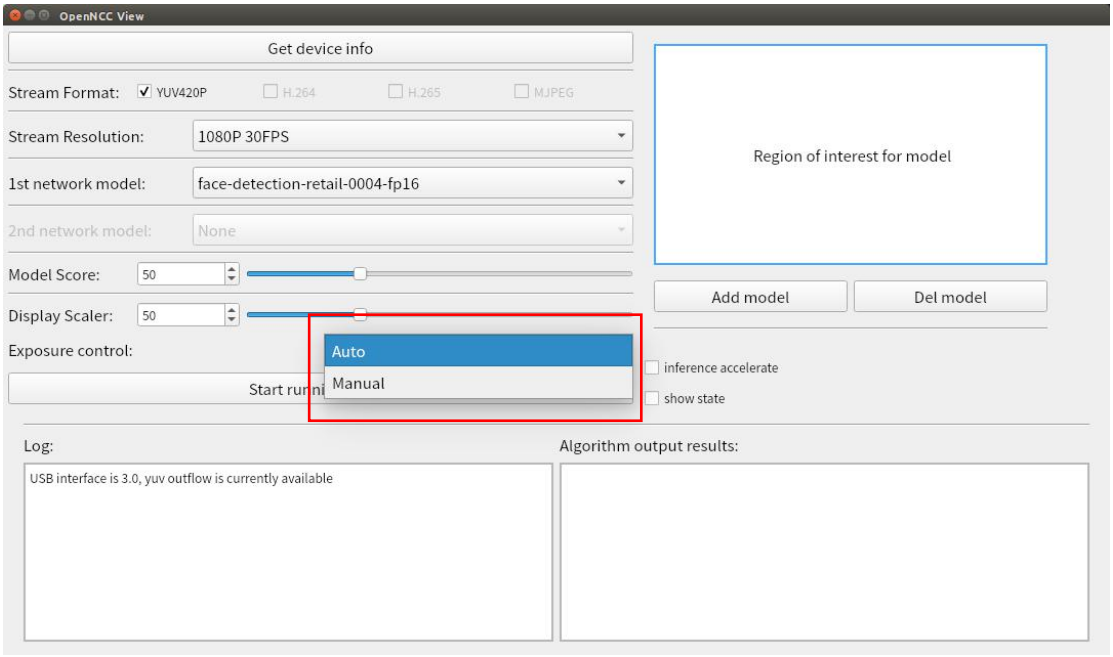
- The algorithm area can be restricted by boxing the ROI region, and the algorithm will only recognize scenes within the region (See the running result demonstration for details)

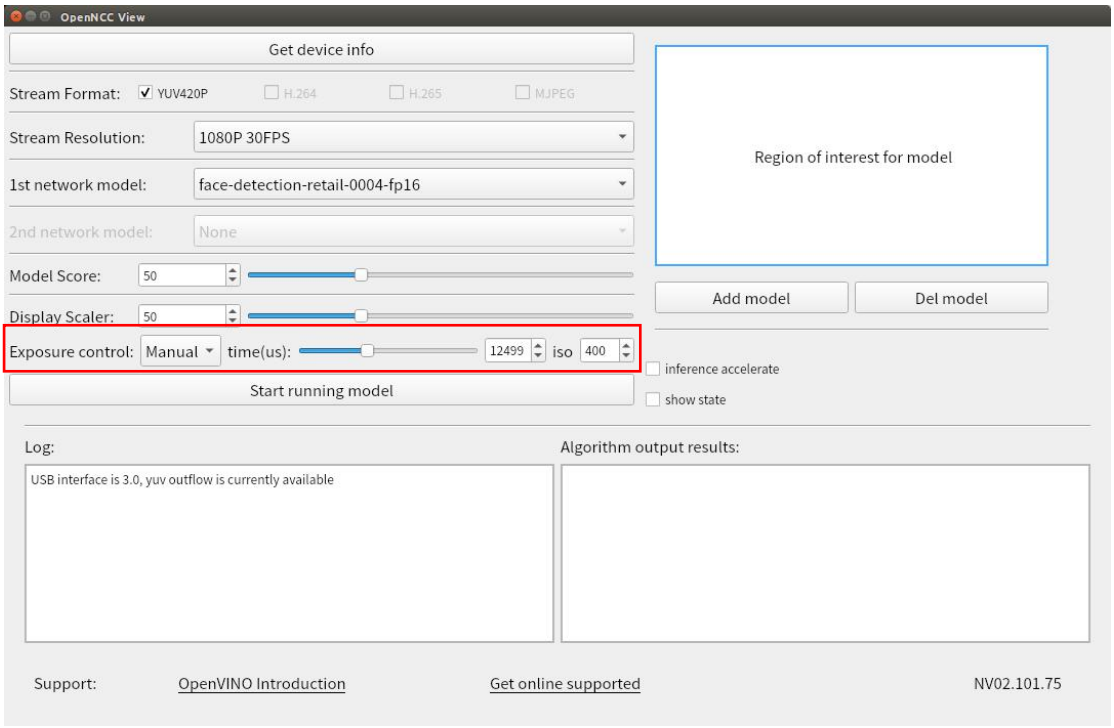
- "2nd network model": Select the secondary algorithm model.

Example model: - vehicle-license-plate-detection-barrier-0106-fp16

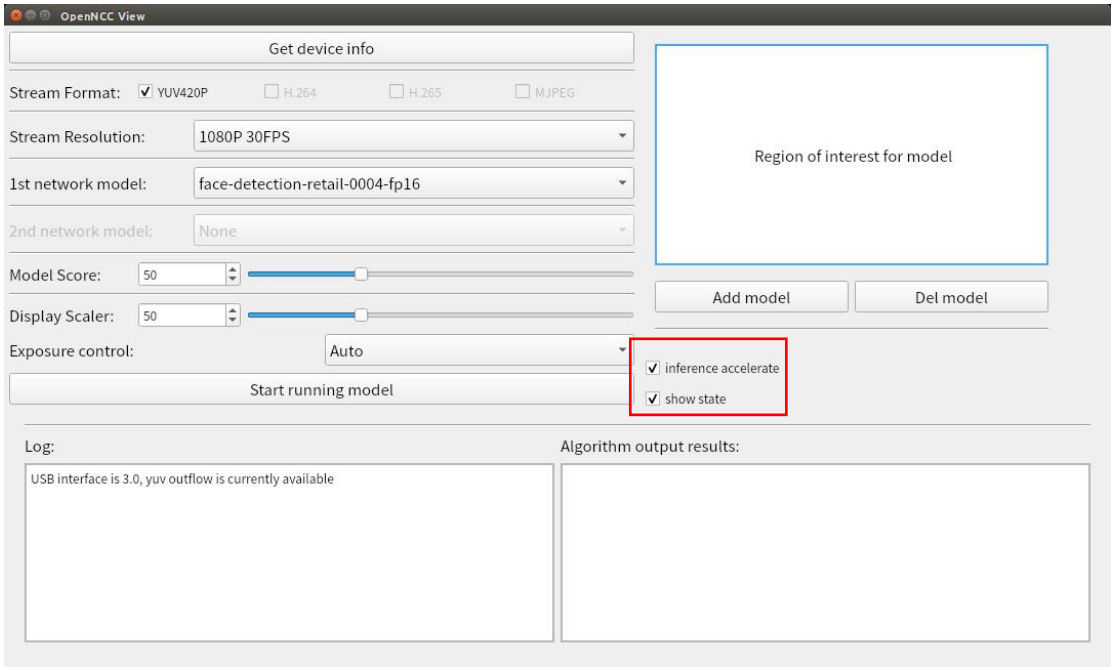


- "Model Score": Set the minimum score for the algorithm to recognize and reach the threshold before the recognition result will be framed in the screen.
- "Display Scaler": Set the video display window size, and you can adjust the display window resolution.
- "Exposure control": Set the exposure, you can choose Auto and Manual, when you choose Manual, you can set your own Exposure times and iso.

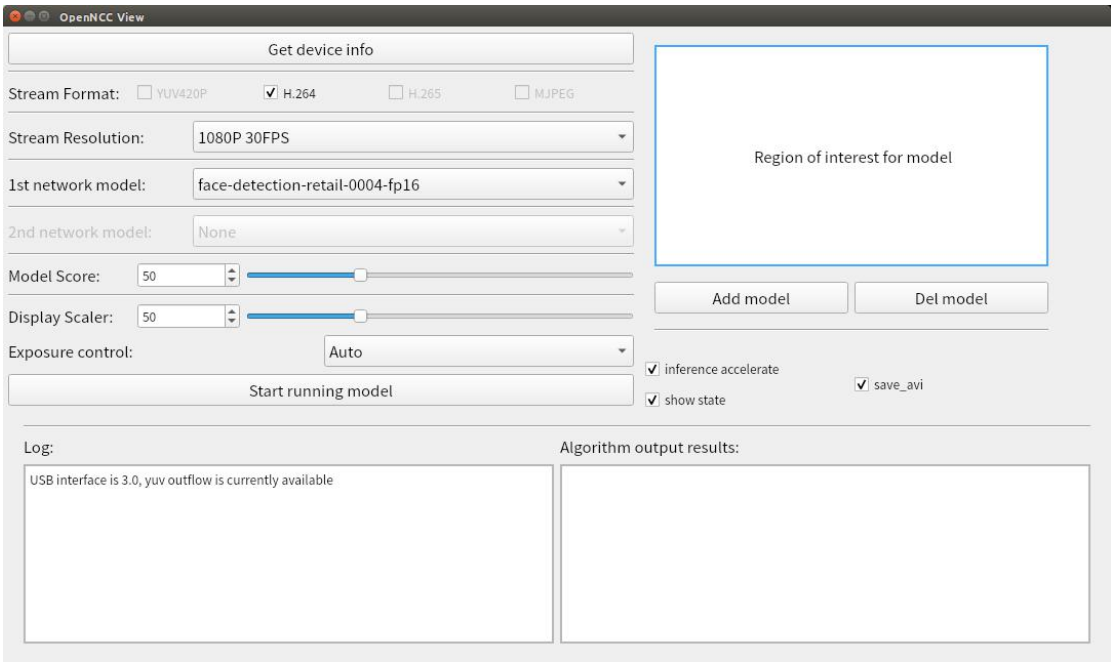




- Switch “show state”, Select whether to display current status information on the screen, including video stream frame rate, algorithm frame rate, resolution, and device id.
- Switch “inference accelerate”,Select whether to enable algorithm acceleration. (Must be selected before loading the algorithm model).



- Switch “save_avi”, The video will be saved to the directory openncc/Platform/Linux/Viewer/OpenNcc_Linux/avi folder with the time named avi file after opening the video stream. When the video stream is closed, the video will stop saving. (yuv420p does not have this item)

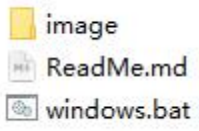


- Click ”Start running models”, Open the video stream. See 2.5 for running results.

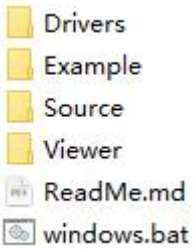
2.2 Quick Start to Windows

2.2.1 Environment construction

- Go to directory: openncc/Platform/Windows.



- Double-click to run windows.bat.
- The results are as follows:



- After the script runs successfully, go to the openncc/Platform/Windows/Drivers directory and install the Windows USB driver. Detailed installation steps can be found in OpenNCC_USB_Driver_install_guide_win.pdf.

2.2.2 OpenNCC_Windows Operation Demo

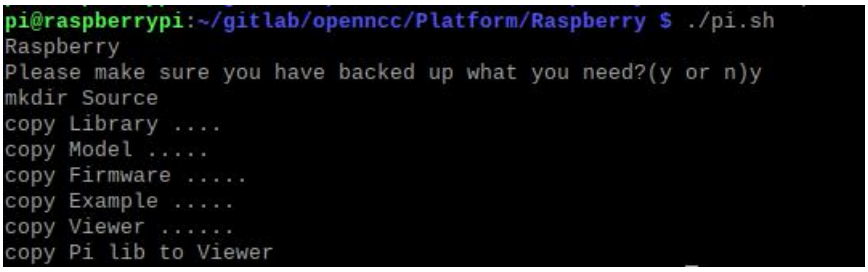
- Go to the directory: openncc/Platform/Windows/Viewer/OpenNcc_Windows.
- Double-click to run OpenNCC.exe.
- Following steps are the same as OpenNCC_Linux.

2.3 Quick Start to Raspberry Pi

2.3.1 Environment construction

- Open terminal
- Go to directory: openncc/Platform/Raspberry

- Enter command ./pi.sh



After running successfully, the following directory will be generated:

```
└── Raspberry
    ├── Example
    ├── Source
    ├── Viewer
    ├── pi.sh
    ├── ReadMe.md
```

2.3.2 OpenNCC_Raspberry Operation Demo

Tip: Before running OpenNCC_Raspberry, please check the device number and contact us for the key, see openncc/Platform/Raspberry/ReadMe.md for details.

- Copy the key file (eyecloud.key) to the directory: openncc/Platform/Raspberry/Viewer/OpenNcc_Raspberry/Configuration/fw.
- Go back to the directory: openncc/Platform/Viewer/OpenNcc_Raspberry.
- Open a terminal and execute sudo . /AppRun to start the software.
- Follow up with the same operation as OpenNCC_Linux.

2.4 Custom(Customization)

2.4.1 Environment construction

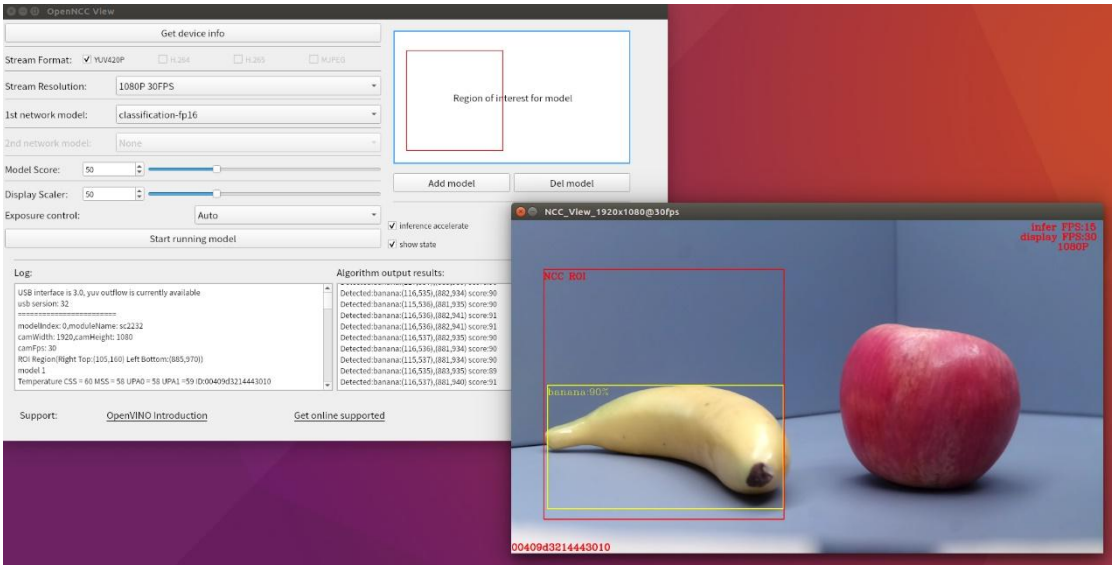
- Go to directory: openncc/Platform/Custom
- Right click to open the terminal
- Enter command ./custom.sh

Tip: To run on a custom platform, please contact us for customization services.

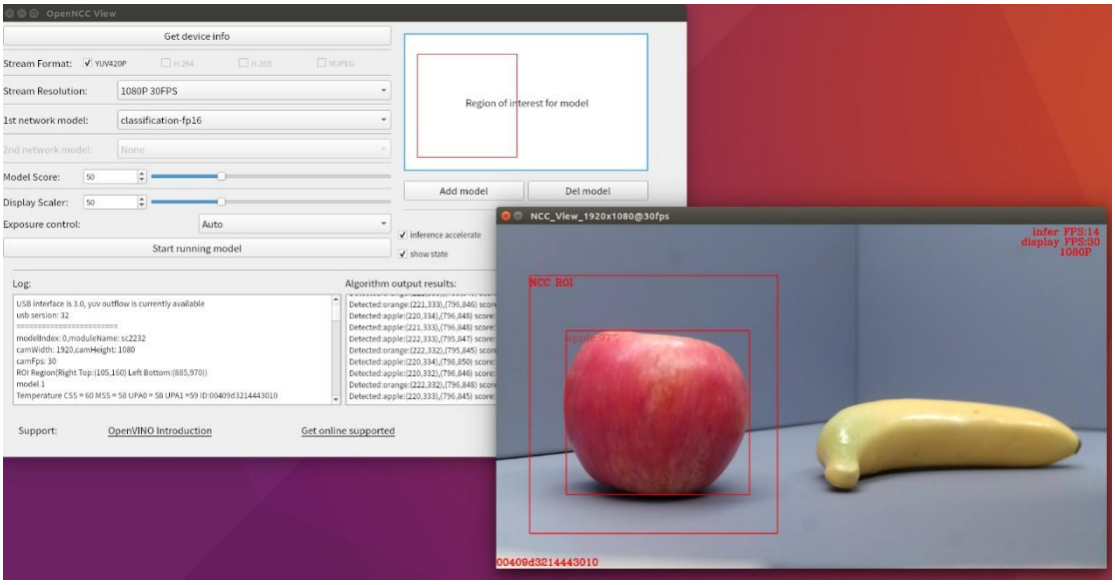
2.5 Running result

Take the object classification algorithm model as an example.

Banana results in the algorithm area:



Apple results in the algorithm area:



3 Software Overview

This chapter introduces the SDK which contains the library of resources necessary for the application and the related basic application routines.

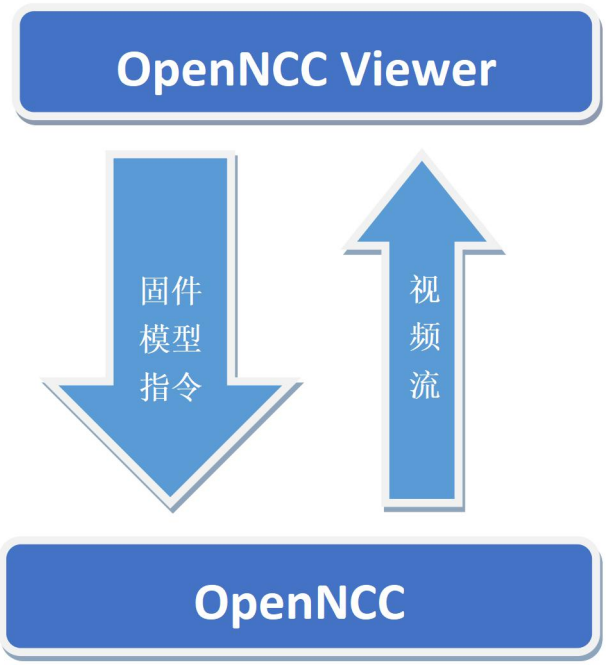
3.1 Resource library

The SDK repository contains three types of files, Firmware, OpenNCC API library and AI model.

Directory	Contents
Source/Firmware	Contains the necessary firmware for OpenNCC.
Source/Library	Contains OpenNCC API libraries and firmware bootloaders compiled in different languages.
Source/Model	Contains different types of AI models already integrated.

3.1.1 OpenNCC Firmware

OpenNCC firmware is the basis for OpenNCC operation. OpenNCC needs to call moviUsbBoot (Movidius official bootloader) from the application to distribute the firmware before it can boot



3.1.2 OpenNCC API Library

The OpenNCC API library is the bridge between the application and OpenNCC. The application sends commands to OpenNCC through the API library, and OpenNCC outputs video streams and AI algorithm results through the API library.

3.1.3 AI Models

Currently, the AI models that OpenNCC has integrated and a brief description are as follows.

Model categories	Name	Introduction
Object Classification	classification-fp16	ssd_mobilenet_v1_coco model can detect almost 90 objects
Face and human shape detection	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

Model categories	Name	Introduction
		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.
person-vehicle-bike-detection	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.
Vehicle Inspection	vehicle-detection-adas-0002-fp16	This is a vehicle detection network based on an SSD framework with tuned MobileNet v1 as a feature extractor.
License plate recognition	vehicle-license-plate-detection-barrier-0106-fp16	This is a MobileNetV2 + SSD-based vehicle and (Chinese) license plate detector for the “Barrier” use case.
Face Properties	interactive_face_detection_demo	This demo executes four parallel infer requests for the Age/Gender Recognition, Head Pose Estimation, Emotions

Model categories	Name	Introduction
		Recognition, and Facial Landmarks Detection networks that run simultaneously
Human Bone Extraction	human-pose-estimation-0001-fp16	A multi-person 2D pose estimation network (based on the OpenPose approach) with tuned MobileNet v1 as a feature extractor.

3.2 Application Examples

The SDK contains two types of examples, How_to and Linkage_demo, and a brief description of the examples is included in the table below. Detailed development details can be found in ReadMe and OpenNCC_SDK_API.pdf in each platform and example directory.

Tip: ReadMe files are recommended to be viewed with Typora the download at: https://www.typora.io/

Directory	Contents
Example/How_to/How_to_use_sdk	Sample application, how to use the SDK library in your project.
Example/How_to/Capture_video	Sample application, how to get a video stream using the SDK library.
Example/How_to/Load_a_model	Sample application, how to download a deep learning model in Blob format using the SDK library.
Example/How_to/work_with_multiple_models	Sample application, how to use a second-level model.
Example/How_to/Python_demo	Python related examples.
Example/Linkage_demo/ work with AlwaysAI / pedestrian_tracking_demo	Face model, using AlwaysAI to parse the result display and count the number of people passing the recognition area.
Example/Linkage_demo/ work_with_OpenVINO/ human_pose_estimation_demo	Human skeleton model, using OpenVINO to parse the result display.
Example/Linkage_demo/ work_with_OpenVINO/ interactive_face_detection_demo	Face, age, gender, and mood models, using OpenVINO parsing results.
Example/Linkage_demo/ work_with_PaddlePaddle	OCR Sample application, include a network link to the OCR warehouse.

3.3 OpenNCC Introduction of operation mechanism

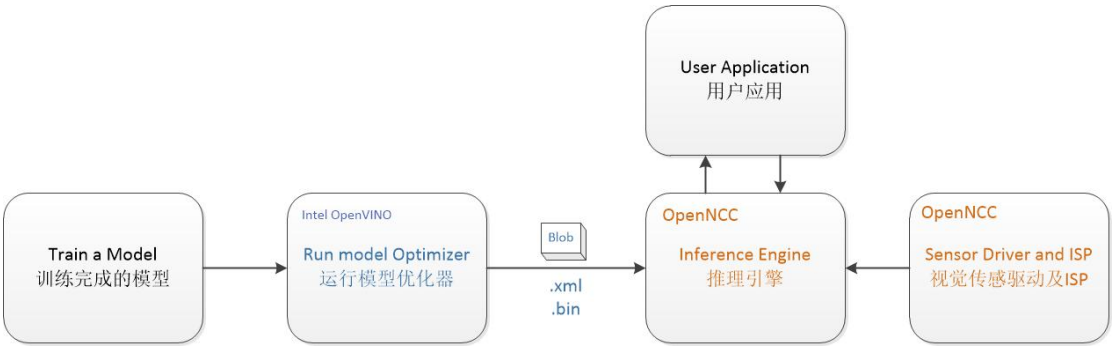
From a model training environment to embedded deployment, it is a very important task, which requires mastering the framework of deep learning, such as commonly used: Caffe*, TensorFlow*, MXNet*, Kaldi*, etc.In addition, it is very important to master the deployed embedded platform. You need to understand the platform performance, system architecture characteristics, and then combine the platform characteristics to optimize the training model framework, and finally tune, transplant, and deploy to the embedded platform.

OpenNCC focuses on the rapid deployment of deep learning models, is compatible with Intel OpenVINO tools, and for embedded graphics and image application scenarios, it has completed the integration of different resolution sensors from 2MP to 20MP on end-point target devices, and the end-point target devices has realized the deployment of professional-level ISP. OpenVINO optimized converted model files can be dynamically downloaded to the end-point OpenNCC camera to achieve rapid deployment of deep learning models.OpenNCC has designed independent working mode, mixed development mode and co-processing compute stick mode to adapt to different work application scenarios.

3.3.1 OpenNCC standalone mode

In the independent mode, OpenNCC independently runs a deep learning model, and feeds back the inference results to the user through the OpenNCC SDK API.

The application deployment process is as follows:



Following the OpenVINO documentation, configure the Model Optimizer for a specific training framework, run the Model Optimizer, and generate an optimized IR file based on the trained network topology, weights and bias values, and other optional parameters, IR is a pair of files describing the entire model The IR is a pair of files describing the entire model, including .xml file - topology file - an XML file describing the network topology, and .bin file - trained data file - a .bin file containing weights and biased binary data, and then run myriad_compile to generate the IR file into a BLOB file.

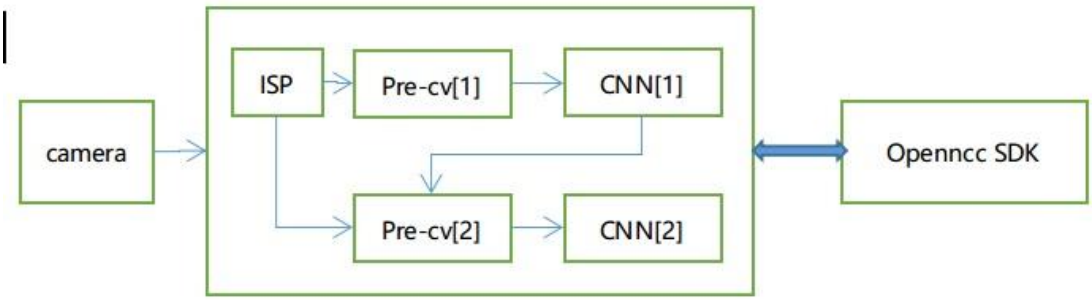
On the application, the integration uses the OpenNCC SDK to download the optimized BLOB model file, see the demo program at Example/How_to/Load_a_model under the SDK.

OpenNCC View is an application demonstrator with an operator interface integrated with the OpenNCC SDK. You can also use OpenView to deploy models and get test results.

Since different deep models have differentiated inference output results, OpenNCC SDK support for different formats of results is increasing. If users cannot find a suitable post-processing parsing model under SDK, they need to write their own post-processing code by referring to Example/How_to/Load_a_model and combining with their own application scenarios.

3.3.1.1 Secondary model runs

Considering the end-side arithmetic power, the current SDK multi-level model supports up to two levels of model cascading, as shown in Fig.:



The first level model must be a target detection or classification model and the output is defined as follows:

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.

Reasoning process:

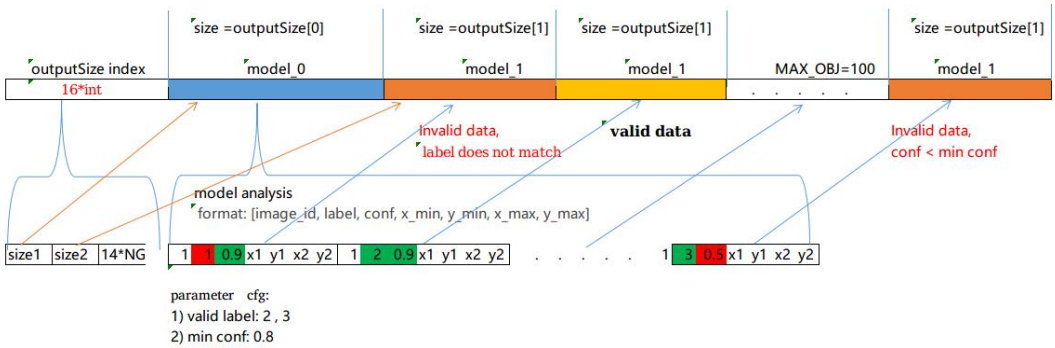
1) The images are first passed through Pre-cv [1], which scales the original image to the first-level model input size and does the corresponding format conversion, then does the first-level model inference calculation and outputs the first-level

inference results to Pre-cv [2].

2) The Pre-cv[2] module parses the inference results of the first-level model, takes the eligible labels and conf detection targets, crop and scale them from the original map to the input size of the second-level model according to the starting point (x_min, y_min) and the ending point (x_max, y_max), and does the corresponding format conversion to enter the second-level model inference.

3) Finally, the first-level model and all the second-level model inference results are packaged together for output.

The model output is parsed (the parameters shown are configured as, valid label: 2,3, conf=0.8).



Example program: Example/How_to/Multiple_models, the first level model is vehicle and license plate detection, the second level model is license plate detection, set the valid label as 2.

Based on the detection results of the first level, the detection coordinates of the first level are fine-tuned appropriately to facilitate recognition:

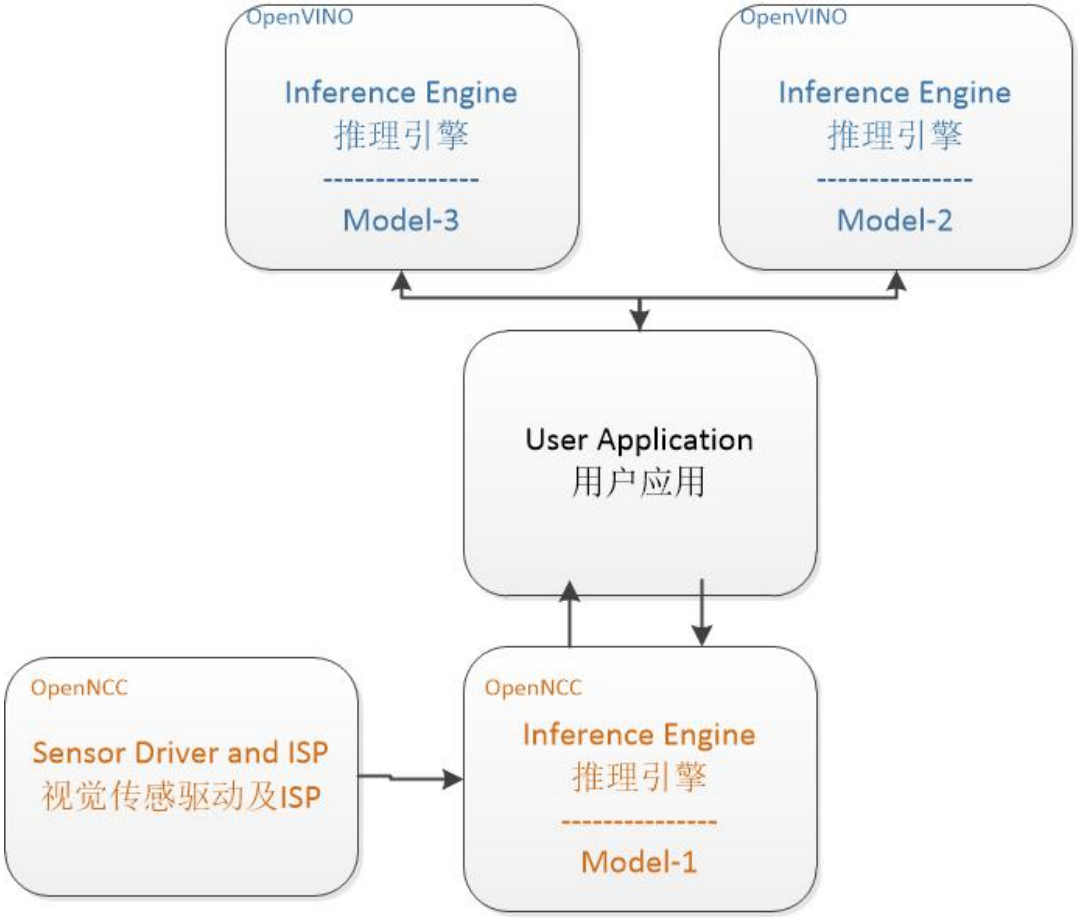
* The starting point is fine-tuned to the upper left (startXAdj, startYAdj)
* The bottom point is fine-tuned to the lower right (endXAdj, endYAdj)

```
cnn2PrmSet.startXAdj = -5;
cnn2PrmSet.startYAdj = -5;
cnn2PrmSet.endXAdj = 5;
cnn2PrmSet.endYAdj = 5;
```

3.3.2 OpenNCC mixed mode

When it is necessary to solve some complex application scenarios, multiple network model combination processing is required, OpenNCC end-point computing performance cannot be met, or the end-side processing needs to be concentrated on the edge side for post-processing, system expansion is often required. Run the models with high real-time requirements on the OpenNCC end-point, and the other models on the post-processing edge machine or cloud.

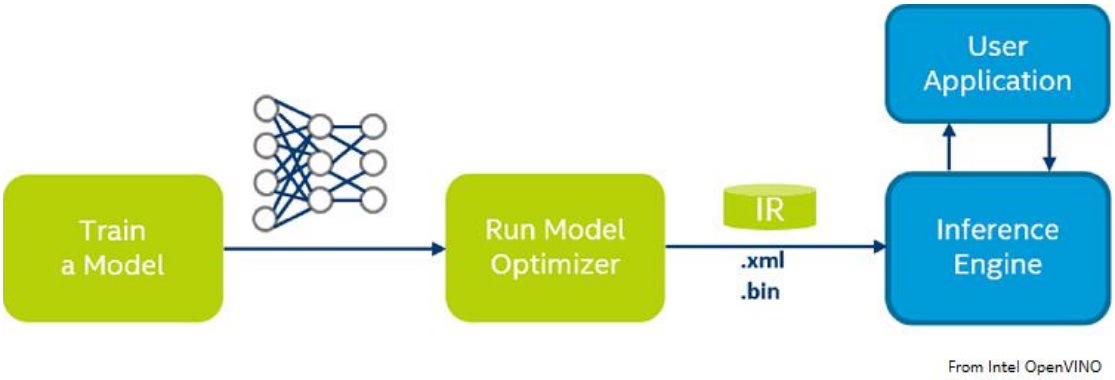
As shown in the figure, Model-1 runs on the OpenNCC end-point to complete the pre-processing of the video stream. OpenNNC returns the results of the first-level processing model to the user application. Model-1 and Model-2 fully run under the OpenVINO inference engine to implement subsequent processing.



In ncc_SDK/Samples/Demo/work with OpenVINO demonstrated how to combine OpenNCC and OpenVINO on Host PC to implement a distributed AI system.

3.3.3 Co-processing compute stick mode

OpenNCC's co-processing mode is similar to Intel NCS2. In this mode of operation, OpenNCC's vision sensor does not work, and users can use OpenNCC alone to achieve full compatibility with the OpenVINO environment. The typical deep learning model deployment process of OpenVINO is as follows:



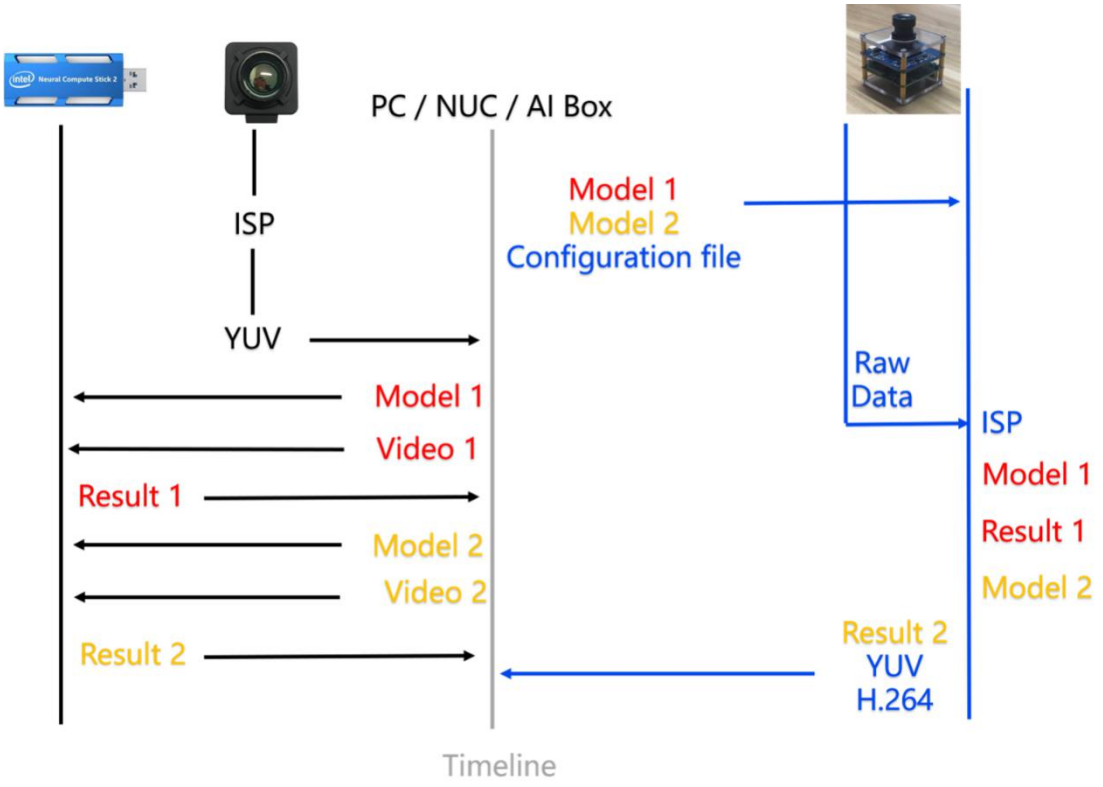
Following the OpenVINO documentation, Configure Model Optimizer for a specific training framework produces an optimized IR file based on the trained network topology, optional parameters such as weights and bias values.

Download the optimized IR file to OpenNCC and run the Inference Engine, refer to the OpenVINO documentation: Inference Engine validation application and sample applications. Copy Source/Firmware/fw /usb-ma2x8x.mvcmd and replace the openvino/inference_engine/lib/intel64/usb-ma2x8x.mvcmd in the openvino installation directory.

Tip: you must backup usb-ma2x8x.mvcmd before replacement, you need to restore this file when using NCS2 reasoning.

3.3.4 Difference between independent mode and co-processing mode

The right side of the figure below is the independent mode of OpenNCC, and the left side is the co-processing mode of OpenNCC (similar to Intel NCS2).



When we need to deploy a vision-based deep learning model, first we need to obtain a high-quality video stream, then run the inference engine to calculate the input image data, and finally output the result.

For the co-processing mode on the left, we need an OpenNCC DK or Intel NCS2 implements end-to-side reasoning. At the same time, we need to obtain a video stream from a camera and send the video frame to OpenNCC DK via USB.

In the independent mode on the right, no additional camera is needed to obtain the video stream. We only need to download the model to OpenNCC to obtain the deduction results.

Refer to OpenVINO official website: https://docs.openvinotoolkit.org/

FAQ

Q: Are other versions of openvino supported?

In addition to 2020.3.194 is officially supported by openncc, other new versions of openvino can also be supported by openncc, but there is no complete coverage evaluation, and some models run abnormally.

Q：How to ensure that openncc camera has firmware algorithm acceleration enabled?

First of all, only openncc camera with ar0234 module can enable firmware dual engine algorithm acceleration function. With openncc view software, it can be activated directly by checking the "inference acceleration" box. If developers need to enable firmware algorithm acceleration in their own programs, they can set the last member of the structure network1par, inferenceacc, to 1 when the SDK is initialized. In addition to observing the frame rate change, you can also test the higher voltage confirmation at the USB interface when accelerating.

Q: What is the effect of firmware dual engine algorithm acceleration?

Firmware dual engine algorithm acceleration has a good effect on all models in openncc view, most of which can improve the performance by nearly 100%. 0004 face recognition model can be stable at about 80 frames in the test process, but it can only reach 43 frames before accelerating. In addition, the models that developers call based on openncc software development framework can be used correctly.

Q: What deep learning frameworks and models does openvino support?

* Caffe* (most public branches)
* TensorFlow*
* MXNet*
* Kaldi*
* ONNX*

more： Supported Frameworks and Formats

Q: Does openncc have a hardware version that supports booting from Flash?

After the development is completed, if the deployment requires openncc to run independently from the SOC, and there is a mass production version with flash, please consult openncc.

Q: I have technical problems with model transformation and parsing. Where can I get support?

Intel • OpenVINO community could help you.

Q: Does openncc support developers to change lenses themselves?

For general applications, developers can choose different shots according to their own scenes. If the image quality can not meet the strict requirements of the scene after lens replacement, you can seek technical customization support from openncc.

Q: Does openncc support the development of Ubuntu virtual machine?

Yes, the USB device of openncc is detected by the virtual machine when it is started. If it is detected by the host first, and then manually switched to the virtual opportunity, the communication exception that the openncc development interface retrieval device fails occurs.

Q： How should OpenNCC custom firmware be replaced?

Method 1(Global update):

Replace OpenNCC.mvcmd in Source/Firmware/fw with new OpenNCC.mvcmd received from us, then execute the script to build environment automatically.

Method 2(Local update):

Find OpenNCC.mvcmd in your project, and just replaced it.

Q: What should I do if the USB connection is abnormal?

For example: log: libusb_bulk_transfer 3 Read error! - 1

Reinsert the USB port and ensure that the USB port is securely connected.

Q: How to develop on a platform without interface?

Please read the ReadMe in the application example for detailed reference.