OpenNCC Software Manual

1. Overview

This document introduces the basic concepts of OpenNCC deployment, OpenNCC CDK and OpenVINO, and the method of using OpenNCC CDK to develop and deploy OpenNCC DK independent operation mode and mixed mode with OpenVINO.

1.1 Support platform

- Hardware：OpenNCC USB, OpenNCC IPC, or OpenNCC IR+,
- PC OS：Ubuntu16.04, Ubuntu18.04, Raspberry Pi OS, ARM Linux (Need to provide toolchain cross compilation)
- Support language: C/C++、Python3.5、Python3.7
- OpenVINO: 2019.R1.144

1.2 Customer Support Center

Please visit https://www.openncc.com/ for more updates and documents.

2.CDK Introduction

OpenNCC CDK is a set of toolkits specifically developed for OpenNCC cameras for rapid deployment of deep learning in OpenNCC DK.

2.1 CDK development package directory structure

Contents	Abstract
ncc_cdk/Docs	OpenNCC Offline documentation
ncc_cdk/NCC_View/Linux	OpenNCC View for Linux
ncc_cdk/Public/Firmwares	OpeNCC Firmware file
ncc_cdk/Public/Library/For_C&C++/Linux	C/C++ OpenNCC CDK static library on Linux and VPU USB bootloader
ncc_cdk/Public/Library/For_C&C++/Windows	C/C++ OpenNCC CDK static library on Windows and VPU USB bootloader

ncc_cdk/Public/Library/For_Python	Python version OpenNCC CDK package, and demo program
ncc_cdk/Sample/Demo/Hello_NCC	C/C++ && Python demo program, using NCC camera to output video stream
ncc_cdk/Samples/How_to/Capture video	Sample program, use OpenNCC CDK library to get video stream
ncc_cdk/Samples/How_to/load a model	Sample program, using the OpenNCC CDK library to load a deep learning model in Blob format
ncc_cdk/Samples/Demo/work with OpenVINO	Sample program, using OpenNCC CDK library to make OpenNCC camera integration compatible with OpenVINO
ncc_cdk/Tools/myriad_compiler	IR file conversion Blob file tool
ncc_cdk/Tools/deployment	OpenNCC deployment script

CDK Download Link: www.OpenNCC.com/Downloads.

API reference "**OpenNCC API.docx**".

3.OpenVINO installation and use

To deploy a deep learning model on end-point target devices, you need to optimize and convert a trained model to the VPU characteristics to achieve higher operating performance. OpenNCC is compatible with OpenVINO's tool set and model format, and needs to rely on Intel OpenVINO's model optimizer to complete model optimization and conversion into Blob format. When using OpenNCC CDK, you need to install OpenVINO as follows:

- If you need to convert the trained model yourself, you need to install OpenVINO to run the model optimizer.
- When OpenVINO runs in a mixed mode with the OpenVINO inference engine, it also needs OpenVINO support.

3.1 Download and install OpenVINO

OpenNCC currently supports OpenVINO version: 2019R1.144,Download please refer to the "**Download link.docx**".

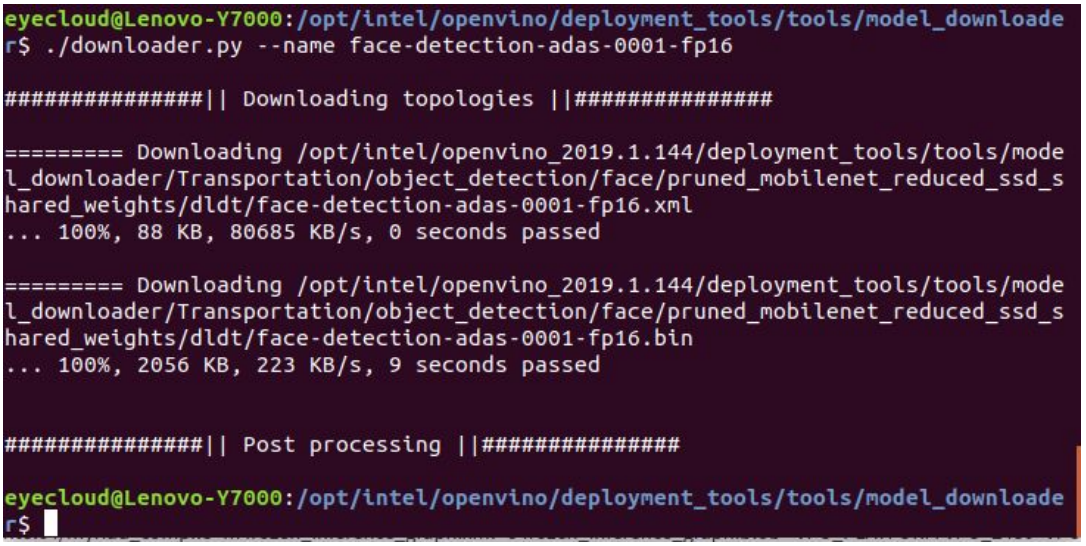
OpenVINO installation reference: "**OpenVINO Installation Guide.docx**"

3.2 Intel Free model download

OpenNCC supports OpenVINO models, Intel has a large number of free trained models for learning reference and testing. After we have installed OpenVINO, we can use the Intel download tool to download the model collection. Model download tool path: `openvino/deployment_tools/tools/model_downloader/downloader.py`, common commands are as follows:

- View all downloadable models：`./downloader.py --print`
- Download the specified model：`./downloader.py --name *`

For example, download a face detection model ：`./downloader.py --name face-detection-adas-0001-fp16`



3.3 Model optimization and format conversion

When we need to deploy a trained model to OpenNCC, we need to optimize and transform the model. After installing OpenVINO, you can use the model optimization tool `/opt/intel/openvino/deployment_tools/model_optimizer/mo.py` to optimize the model. For specific documents, see the official Intel documents: Model Optimizer Developer Guide

After the model optimization is completed, the model needs to be converted to the Blob format before it can be deployed on OpenNCC. In the OpenVINO installation

directory: `/opt/intel/openvino/deployment_tools/inference_engine/lib/intel64`myriad_compile tool, the method of use is as follows:

Enter from the command line terminal : `./myriad_compile -m input_xxx-fp16.xml -o output_xxx.blob -VPU_PLATFORM VPU_2480 -VPU_NUMBER_OF_SHAVES 8 -VPU_NUMBER_OF_CMX_SLICES 8`

After the format conversion is completed, the model can be deployed on OpenNCC, refer to: ncc_cdk/Samples/How_to/load a model, or use the OpenNCC View interface program to add the model to deploy and test it.

4. OpenNCC operating mechanism

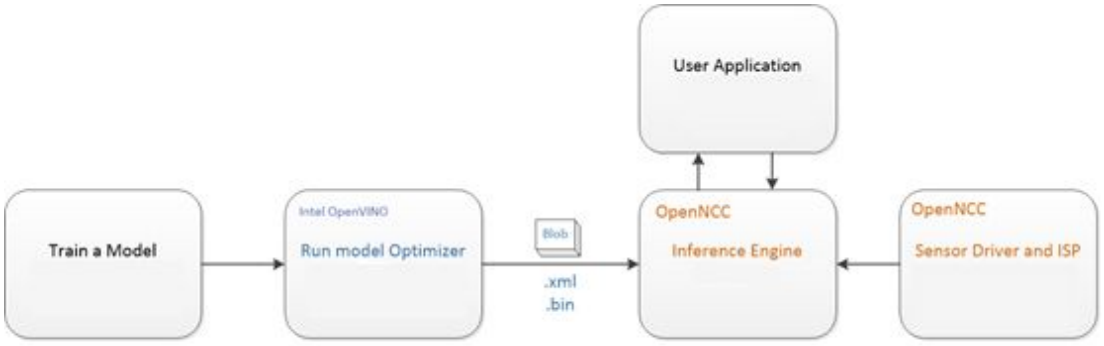
From a model training environment to embedded deployment, it is a very important task, which requires mastering the framework of deep learning, such as commonly used: Caffe*, TensorFlow*, MXNet*, Kaldi*, etc.In addition, it is very important to master the deployed embedded platform. You need to understand 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.

4.1 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 CDK API.

The application deployment process is as follows:



According to the OpenVINO documentation, for a specific training framework Configure Model Optimizer

Run Model Optimizer to produce an optimized Intermediate Representation (IR) of the model based on the trained network topology, weights and biases values, and other optional parameters.

The IR is a pair of files that describe the whole model:

◦ .xml: The topology file - an XML file that describes the network topology

◦ .bin: The trained data file - a .bin file that contains the weights and biases binary data

Then run myriad_compile to generate a BLOB file from the IR file.

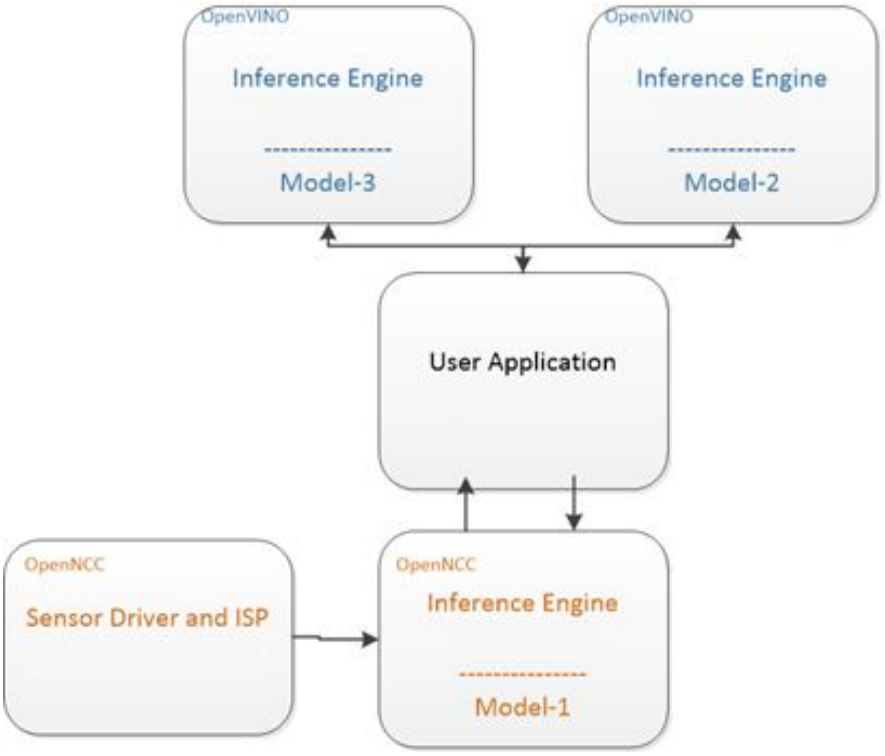
To integrate the BLOB model file generated after optimization using OpenNCC CDK, see the demo program of Samples/How_to/Load a model under CDK.

OpenNCC View is an application demonstration program with an operating interface integrated with OpenNCC CDK. You can also use OpenView to deploy models and obtain test results. Refer to OpenNCC View Guide Because different depth models have differentiated inference output results, if users cannot find a suitable post-processing analytical model under the CDK, they need to refer to ncc_cdk/Samples/How_to/load a model and write post-processing code in combination with their own application scenarios.

4.2 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.

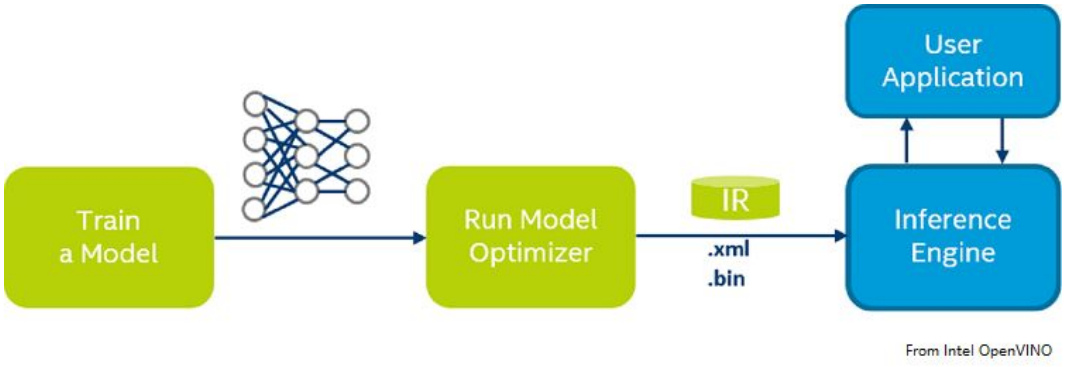
As shown in the figure, Model-1 runs on the OpenNCC end-point to complete the pre-processing of the video stream. OpenNNC returns the results of the first-level processing model to the user application. Model-1 and Model-2 fully run under the OpenVINO inference engine to implement subsequent processing.



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

4.3 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:



Configure Model Optimizer for specific training framework according to OpenVINO documentation.

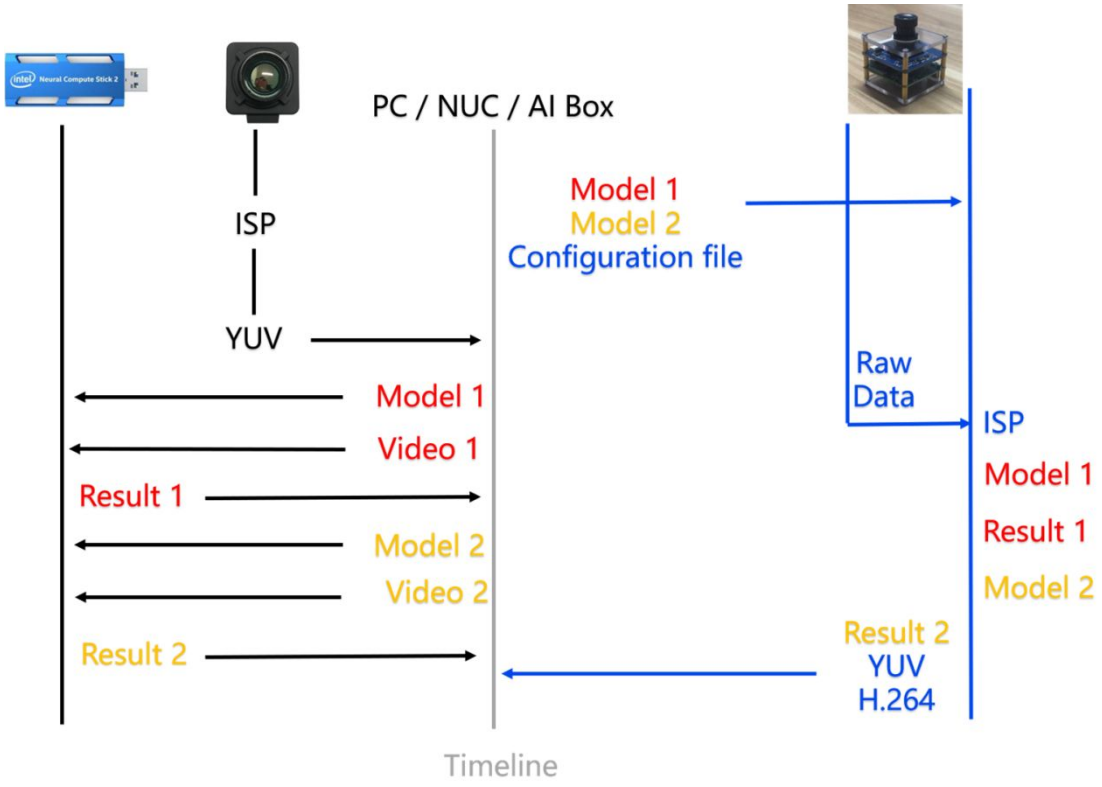
Run Model Optimizer to produce an optimized Intermediate Representation (IR) of the model based on the trained network topology, weights and biases values, and other optional parameters.

Download the optimized IR file to OpenNCC to run the Inference Engine. For details, refer to OpenVINO documents: Inference Engine validation application and sample applications.

Copy `Public/Firmwares/MvNCAPI-ma2480.mvcmd` and replace `openvino/inference_engine/lib/intel64/MvNCAPI-ma2480.mvcmd` in the openvino installation directory.(Remarks: MvNCAPI-ma2480.mvcmd in the openvino installation directory must be backed up before replacement. This file needs to be restored when using NCS2 inference)

4.4 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/