Amlogic

NN Integration Guide

Revision: 0.2

Release Date: 2022-02-10

Copyright

Trademarks

Amlogic, and other Amlogic icons are trademarks of Amlogic companies. All other trademarks and registered trademarks are property of their respective companies.

Disclaimer

Amlogic may make improvements and/or changes in this document or in the product described in this document at any time.

This product is not intended for use in medical, life saving, or life sustaining applications.

Circuit diagrams and other information relating to products of Amlogic are included as a means or illustrating typical applications. Consequently, complete information sufficient for production design is not necessarily given. Amlogic makes no representations or warranties with respect to the accuracy or completeness of the contents presented in this document.

Contact Information

- Website: www.amlogic.com
- Pre-sales consultation: contact@amlogic.com
- Technical support: support@amlogic.com

Revision History

Issue 0.2 (2022-02-10)

This is the second official release.

Compared to last version, the following changes are made:

1. Add nn engine tool introduction
2. Add aml_perlayer_visual tool introduction

Issue 0.1 (2021-08-10)

This is the Initial release.

Contents

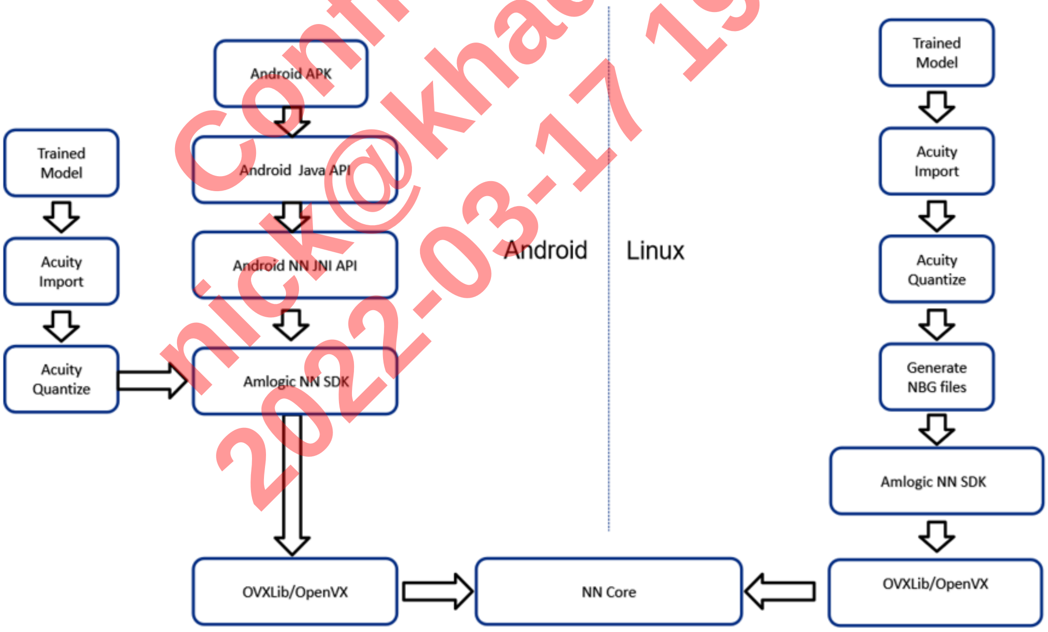
1. Introduction

This document describes the concept of AMLOGIC NN and its integration steps, to guide AMLOGIC developers and customers.

1. AMLOGIC NN chip uses GPU architecture to optimize and accelerate a series of neural network operators such as conv, deconv, pool and relu etc.
2. Low-level optimization and acceleration through openvx1.2
3. Support Caffe, tensorflow, tflite, darknet, onnx, keras, pytoch and other open-source frameworks
4. Support three quantization methods including int8, int16, uint8 etc., also support hybrid quantization to improve accuracy (the latest IP supports Perchannel quantization)
5. The supported open-source framework model needs to be quantified using the above three quantization method first, and then export the NBG (Network Binary Graph) file that can be recognized and loaded by the chips.
6. For the supported neural network, please refer to: https://github.com/VeriSilicon/acuity-models
7. Support Android, Linux, FreeRTOS and other systems

1.1 Amlogic NN Workflow

The workflow of AMLOGIC NN is as follows:



1.2 Integration Flow

The integration flow includes the following steps::

1. Select the model (model can be self-trained, or find trained model online)

2. Check the development environment (the DDK on the board, model tool and SDK versions need to be aligned.)

3. Model transcoding (The board only supports quantitative model. Model transcoding is required)

4. Choose SDK release (AMLOGIC provides two SDK releases which have different pros and cons.)

5. Take the provided demo as an example, do demo integration

6. Based on the integrated demo, integrate into your own application framework

2. Release Source Introduction

2.1 Introduction

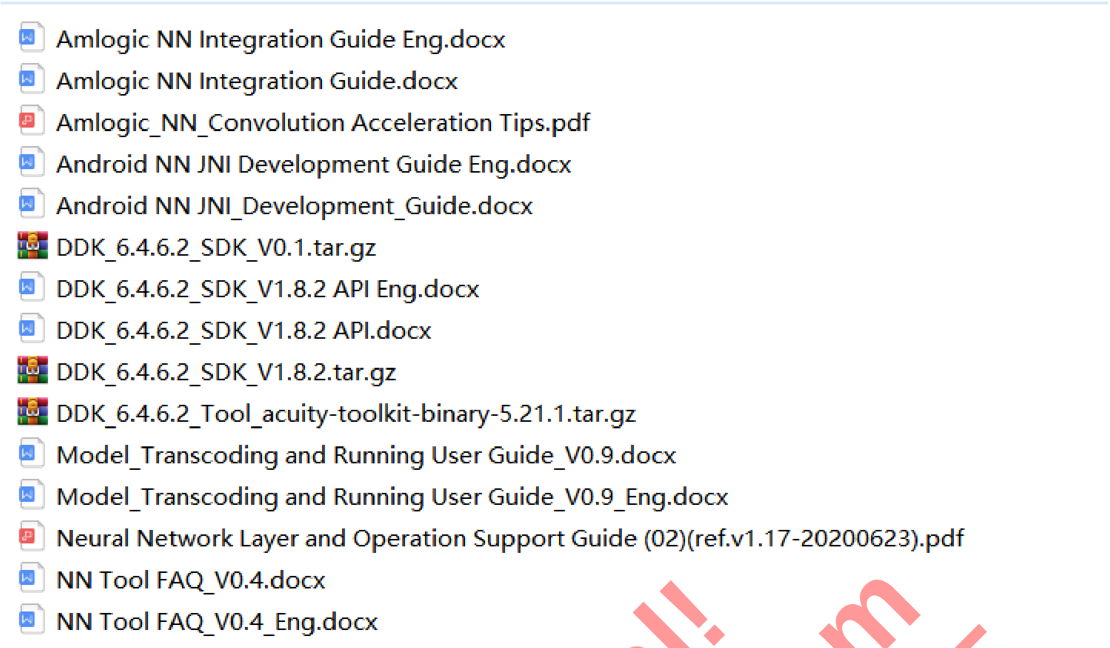
This chapter lists current AMLOGIC NN released documents, tools and their scope of use.

2.2 Introduction to documents and tools

The following table lists the current AMLOGIC released documents and tools. Contact your AMLOGIC representative If anything is missed.

NUM	Document	Remarks
1	AMLNN Convolution Acceleration Tips.pdf	Model design guide document
2	Neural Network Layer and Operation Support Guide.docx	Operators supported list document
3	Model_Transcoding and Running User Guide_V0.9.docx Model_Transcoding and Running User Guide_V0.9_Eng.docx	Model Transcoding
4	DDK_xxxx_SDK_V1.x.x API.docx DDK_xxxx_SDK_V1.x.x API Eng.docx	SDK_ V1.8 API in Chinese and English
5	NN Tool FAQ_V0.4.docx NN Tool FAQ_V0.4_Eng.docx	NN FAQ in Chinese and English
6	**DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz**	Model Transcoding Tools
7	**DDK_x.x.x.x_SDK_V0.1.tar.gz**	SDK_V0.1 development package
8	**DDK_x.x.x.x_SDK_V1.x.x.tar.gz**	SDK_V1.8 development package
9	Android NN JNI Development Guide.docx Android NN JNI Development Guide_Eng.docx	Android NN JNI Development Guide
10	Amlogic NN Integration Guide.docx Amlogic_NN_Integration_Guide_Eng.docx	Amlogic_NN Integration Guide
11	aml_nnengine_Tool_Introduction_v0.1.docx aml_nnengine_Tool_Introduction_v0.1_en.docx	Aml nn engine guide
12	aml_perlayer_visual_tool_v0.1.docx aml_perlayer_visual_tool_v0.1_en.docx	Aml nn visualizatoin tool

Note: x.x.x.x is DDK version. For example, DDK6.4.6.2 has the following documents and tools:



Amlogic NN Integration Guide Eng.docx
Amlogic NN Integration Guide.docx
Amlogic_NN_Convolution Acceleration Tips.pdf
Android NN JNI Development Guide Eng.docx
Android NN JNI_Development_Guide.docx
DDK_6.4.6.2_SDK_V0.1.tar.gz
DDK_6.4.6.2_SDK_V1.8.2 API Eng.docx
DDK_6.4.6.2_SDK_V1.8.2 API.docx
DDK_6.4.6.2_SDK_V1.8.2.tar.gz
DDK_6.4.6.2_Tool_acuity-toolkit-binary-5.21.1.tar.gz
Model_Transcoding and Running User Guide_V0.9.docx
Model_Transcoding and Running User Guide_V0.9_Eng.docx
Neural Network Layer and Operation Support Guide (02)(ref.v1.17-20200623).pdf
NN Tool FAQ_V0.4.docx
NN Tool FAQ_V0.4_Eng.docx

3. Model Design Guide

3.1 Introduction

This chapter guides customers who have the development environment and are able to design and train the models, on how to design models better fit with AMLOGIC NN acceleration rules. This chapter can be skipped for customers who only do model integration.

3.2 Introduction to Amlogic NN Chips

AMLOGIC NN chip has three computing blocks: NN core, PPU and TP.

NN core:

The main computing block of NN, it includes large number of MAC units, mainly used for large-scale computing such as convolution computing.

Pros: Fast computing speed.

Cons: Support less operators.

TP:

Tensor processing unit, it is a supplementary to NN core, used for processing of tensor.

Pros: Supports more operators than NN core.

Cons: Lower computing speed.

PPU:

A GPU-like module. PPU is programmable. It supports float computing and is the main computing block for custom operators.

Pros: Programmable, high precision, supports all operators in theory.

Cons: Lower computing speed.

Note: < neural network layer and Operation Support Guide > lists the supported operators and describes which module of NN the operators run in.

3.3 Design Guide

General principles:

1. Try to use more continuous operators on the NN core to build model
2. Try to use operators that can be fused to build model
3. For other acceleration rules, please refer to *AMLNN Convolution Acceleration Tips*

Related documents:

AMLNN Convolution Acceleration Tips.pdf

Neural Network Layer and Operation Support Guide

4. Model Transcoding Guide

4.1 Introduction

This chapter describes how to check the version of NN driver on the board and apply for the documents and model transcoding tools according to the NN driver version.

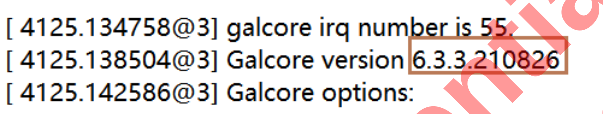
4.2 Environment Check

The AMLOGIC NN releases are updated from time to time. The NN driver on the board must be used in company with the correct version of model transcoding tool and SDK. However, due to the update on different branches might not be synced, before integration, you need to check which DDK version is used first, then choose the correct SDK and model transcoding tool for integration. Update to the latest version if necessary.

4.2.1 Version Check

1. Get the NN driver version on the board:

   Execute the command on the serial port or ADB window: dmesg |grep Galcore



```
[ 4125.134758@3] galcore irq number is 55.
[ 4125.138504@3] Galcore version 6.3.3.210826
[ 4125.142586@3] Galcore options:
```

   Note: if the board has started for a long time, the Galcore information may be flushed. If the Galcore information is not available, restart the board and execute the command again.

2. Find the version of model transcoding tool and SDK according to the driver version retrieved above.

DDK_Version	Acuity Tool Version	Galcore Version	Release Time
DDK6.3.3.4	Acuity_5.0	6.3.3.210826	2019/06/05
DDK6.4.0.3	acuity_5.5.0	6.4.0.229426	2019/10/30
DDK6.4.0.10	acuity_5.7.0	6.4.0.10.239607	2020/03/10
DDK6.4.2.1	acuity_5.11.0	6.4.2.1.258160	2020/04/27
DDK6.4.3	acuity_5.14.1	6.4.3.279124	2020/07/24
DDK6.4.4.3	acuity_5.16.3	6.4.4.3.310723	2021/03/25
DDK6.4.6.2	acuity_5.21.1	6.4.6.2.345497	2021/06/02
DDK6.4.8.7	Acuity_6.0.12	6.4.8.7.415784	2022/01/05

For example:

If currently the driver version is 6.3.3.210826, then the DDK version is 6.3.3.4.

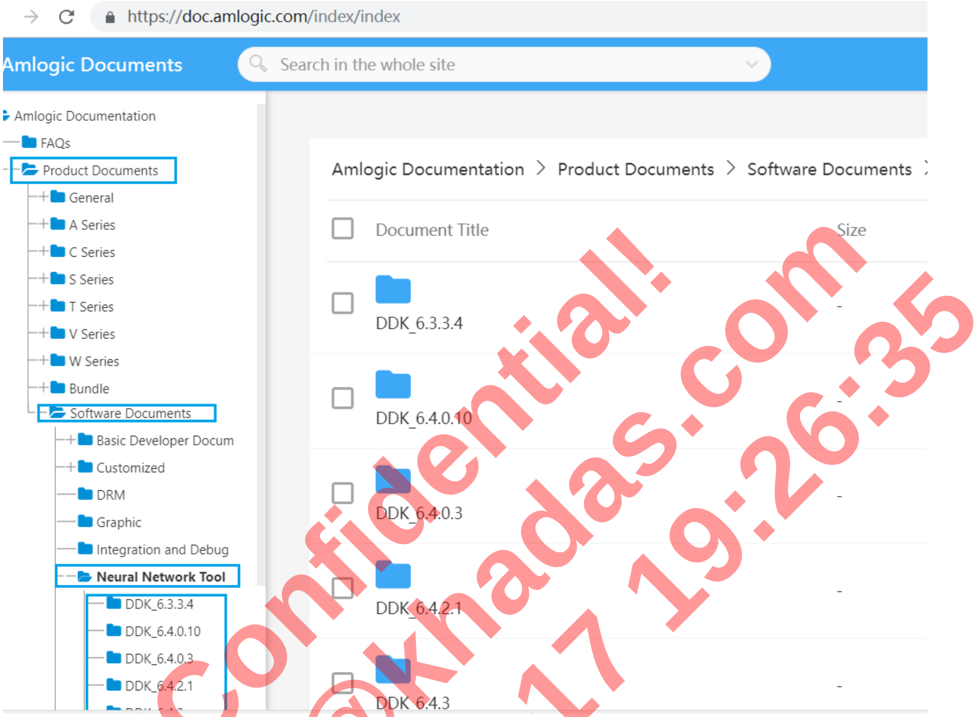
4.2.2 Get Documents

After DDK version is clear, you need to get the corresponding version of tools and documents.

If the DDK version is 6.3.3.4, then the corresponding model transcoding tool is acuity-toolkit-binary-5.14.1.tar.gz and SDK is DDK_6.4.3_SDK_V0.1.tar.gz/DDK_6.4.3_SDK_V1.x.x.tar.gz.

Generally, AMLOGIC sales representative will apply for the documents and tools on behalf of customers, then release them to customers. Customers need to check whether they received correct version of tools and documents. If they are not match, contact your AMLOGIC sales representative to re-apply for the correct version of documents and tools.

Apply link: https://doc.amlogic.com/index/index



https://doc.amlogic.com/index/index

Amlogic Documents

Search in the whole site

- Amlogic Documentation
  - FAQs
  - Product Documents
    - General
    - A Series
    - C Series
    - S Series
    - T Series
    - V Series
    - W Series
    - Bundle
    - Software Documents
      - Basic Developer Docum
      - Customized
      - DRM
      - Graphic
      - Integration and Debug
      - Neural Network Tool
        - DDK_6.3.3.4
        - DDK_6.4.0.10
        - DDK_6.4.0.3
        - DDK_6.4.2.1

Amlogic Documentation > Product Documents > Software Documents >

□	Document Title	Size
□	DDK_6.3.3.4	-
□	DDK_6.4.0.10	-
□	DDK_6.4.0.3	-
□	DDK_6.4.2.1	-
□	DDK_6.4.3	-

4.3 Model Transcoding

After a customer selects the model to be integrated, the first step is to perform model transcoding using Acuity tool.

4.3.1 Environment Preparation

1. Unzip the received model transcoding tool on Ubuntu:

   tar -zxf DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz

   Note: The tool needs to be unzipped in Linux. There will be link issues when it is unzipped in windows and copied to Linux.

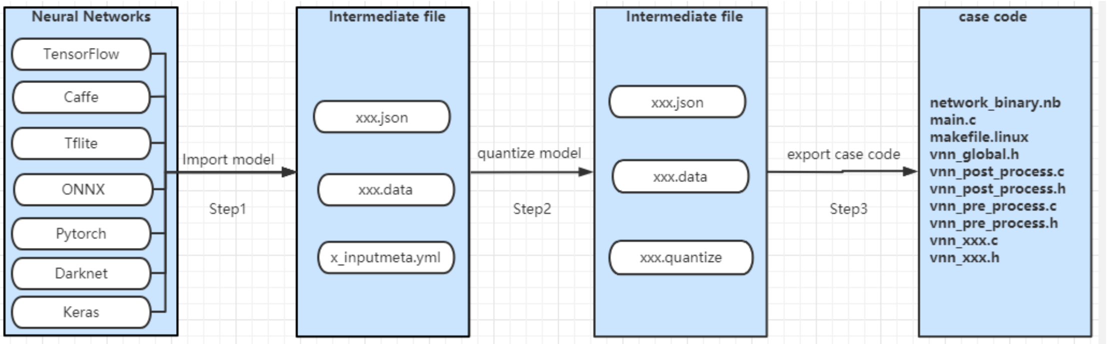
2. Install the tool

   Refer to the readme.txt under the acuity-tool-binary-xxxx directory to install the tool.

3. Check the tool

   After the tool is installed, use the demo under the acuity-tool-binary-xxxx/demo directory for model transcoding test.

4.3.2 Model Transcoding



There are three steps for model transcoding: import model, quantify model and export case demo. For more details, please refer to chapter 3.4 of *Model Transcoding and Running User Guide.docx*.

1、Key points in model transcoding:Set correct value of channel-mean-values in xxx_imputmeta.xml . The value provided should be equal to mean-values used by model training when it processes input pictures.

2、Quantitative data is best provided by 200 ~ 500 groups to ensure that the maximum / minimum value is the maximum / minimum value of the actual operation scenario during quantization step.

3、In case different channels use different scales, customer can do pre-processing to convert the input data into NPY files, then provide NPY files during quantization step. Note: Related documents and tools for this chapter:

1. Model transcoding documents:

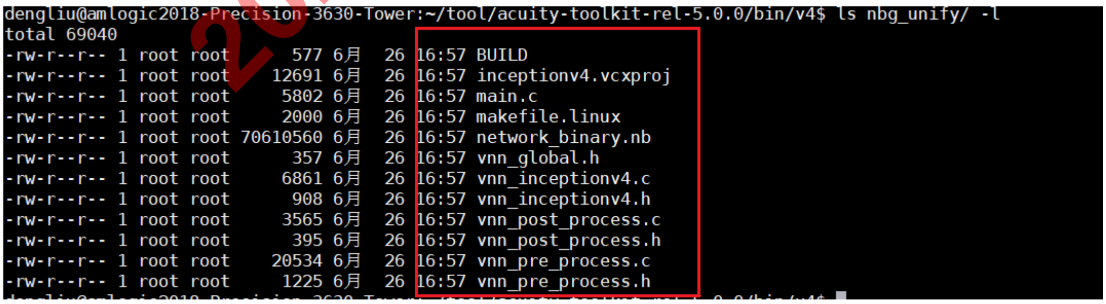
   <Model_Transcoding and Running User Guide_V1.0>

   <Model_Transcoding and Running User Guide_V1.0_Eng>

2. Model transcoding tool: DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz

4.3.3 Performance test

After the model transcoding in chapter.4.3.2 is completed, a simple demo (which can be running after compilation) is generated, as shown in figure below:



```
dengliu@amlogic2018-Precision-3630-Tower:~/tool/acuity-toolkit-rel-5.0.0/bin/v4$ ls nbg_unify/ -l
total 69040
-rw-r--r-- 1 root root      577 6月   26 16:57 BUILD
-rw-r--r-- 1 root root    12691 6月   26 16:57 inceptionv4.vcxproj
-rw-r--r-- 1 root root     5802 6月   26 16:57 main.c
-rw-r--r-- 1 root root     2000 6月   26 16:57 makefile.linux
-rw-r--r-- 1 root root 70610560 6月   26 16:57 network_binary.nb
-rw-r--r-- 1 root root      357 6月   26 16:57 vnn_global.h
-rw-r--r-- 1 root root     6861 6月   26 16:57 vnn_inceptionv4.c
-rw-r--r-- 1 root root      908 6月   26 16:57 vnn_inceptionv4.h
-rw-r--r-- 1 root root     3565 6月   26 16:57 vnn_post_process.c
-rw-r--r-- 1 root root      395 6月   26 16:57 vnn_post_process.h
-rw-r--r-- 1 root root    20534 6月   26 16:57 vnn_pre_process.c
-rw-r--r-- 1 root root     1225 6月   26 16:57 vnn_pre_process.h
```

After using Acuity tool, the result of model transcoding is a simple demo which uses the classification model template. That is, the current demo only performs top5 post-processing on one output.

Performance test.:

1. Refer to the readme.txt under android or buildrootsdk in SDK V0.1 on how to compile the exported case demo into executable binary.
2. Push the executable binary onto the board and run it to get the performance data/frame rate

Note: Related documents and tools for this chapter:

Android sdk:	DDK_x.x.x.x_SDK_V0.1.tar.gz/android_sdk_xxx.tar.gz
Linux sdk:	DDK_x.x.x.x_SDK_V0.1.tar.gz/buildroot_sdk_xxx.tar.gz

SDKv1.x.x is a set of interfaces provided by AMLOGIC, which provides an abstraction for case demo, to hide the interface changes in case demo when DDK is upgraded.

Refer to ReadMe.txt under android_sdk/buildroot_sdk for general information.

5. Model integration guide

After the model transcoding is completed, the next step is to do the final model integration.

5.1 SDK version Choice

5.1.1 Background

SDKv0.1 is a software package for compliment and integration, its code is based on the case demo exported from model. SDKv1.x.x is a set of interfaces provided by AMLOGIC, which provides an abstraction for case demo, to hide the interface changes in case demo when DDK is upgraded.

Some existing processes are still using SDKv0.1. For compatible reasons, both SDK releases are published. Customer should choose which SDK release to use for their development.

5.1.2 Comparison between SDK releases

	SDK_V1.x.x	SDK_V0.1
Introduction	AMLOGIC encapsulated interfaces	Original SDK, Case demo exported by Acuity tool
Dependency	libnnsdk.so	libovxlib.so, libCLC.so, libGAL.so, libOpenCL.so, libOpenVX.so, libOpenVXU.so, libVivanteOpenCL.so, libVSC.so
Header file dependency	nn_sdk.h/nn_util.h	vsi related header files(there are too many header files and will not be listed here)
Integration Interface	Use encapsulated interfaces	Extracted interfaces based on the case demo
Version Upgrade	Only update NBG file	Need to update NGB file, model creation file, then re-compile

SDK1.x is starting supported from DDK6.4.3, earlier versions of DDK only supports SDKv0.1

sdkv0.1: DDK_x.x.x.x_SDK_V0.1.tar.gz

sdkv1.x: DDK_x.x.x.x_SDK_V1.x.x.tar.gz

5.2 Model Integration

Amlogic NN only provides a running platform. That is. it only runs the computation of the model.

A post-processing is required to convert the data output from the last node of the NPU, this post-processing needs to be done by customer. For example, for classification model, the data generated by NN is a (1,1000) array, this data needs to be further processed by top1 to get the final classification result.

5.2.1 Accuracy Check

It is generally recommended to use the simplest demo to test the accuracy of the lower plate side. If you don't do this operation and find that the result is wrong, you can also return to this operation.

Generally, it is recommended to test with the simplest demo to get the accuracy data of board test result.

You might return to this step (if it hasn't been done before) at any time if you find the final result is not correct.

1. Inference the Input and output for float and quantize data on PC using Acuity tool.

   Refer to step4 in chapter 3.4 of *Model Transcoding and Running User Guide*

2. For SDK v0.1, refer to the steps in chapter 4.2 of *NN Tool FAQ* document to confirm the accuracy

Note: Use the tensor file inferenced on PC as the input to board to keep consistent on input. The tensor file inferenced on PC contains the result after pre-processing. There is no secondary processing for alignment on board at run time.

Related documents:

NN Tool FAQ_V0.4.docx

NN Tool FAQ_V0.4_Eng.docx

DDK_x.x.x.x_SDK_V0.1.tar.gz

5.2.2 Confirmation the results of the Board

Chapter 5.2.1 can only quickly confirm the similarity between the board-side result of a single frame input and the PC-size result. Some model are cumbersome to add post-processing based on the C language, and it is impossible to quickly confirm whether the final display result corresponding the board-side running result is correct. Currently, a way is provided to allow customers to quickly confirm whether the final display result corresponding to the board-side running result is correct.

nnEngine is a set of model reasoning engine based on nnsdk development kit developed by Amlogic. It uses ADB protocol to realize data transmission and model operation control between embedded device and server. Users can use the provided aml_nn_engine python library to implement model inference and performance index acquisition functions.

Related documents:

aml_nnengine_Tool_Introduction_v0.1.docx

5.2.3 Visualization of Model operation infromation

Some customers need to confirm the execution time and bandwidth of each layer of the model running on the board. Perlayer visual can meet this requirement.

Perlayer Visual is a visualization tool for each layer of the model based on hardware information printing developed by Amlogic. It is convenient for users to evaluate the running time and bandwidth occupation of the model, as well as the problem of time-consuming and excessive bandwidth occupation of the positioning model.

Related documents:

aml_perlayer_visual_tool_v0.1.docx

5.2.4 Model Integration

The main effort of model integration is to add post-processing code to process the data from the last node and convert the data into the final display result.

1、Add post-processing codes

1. SDKV0.1:

1) Refer to the converted case demo, add post-processing code for current model in vnn_post_process.c: show_top5

2) Verify that the result is correct on board

2. SDKV1.X:

1）Refer to DDK_xxxx_SDK_V1.x.x API.doc and the demo program in DDK_x.x.x.x_SDK_V1.x.x.tar.gz, build demo case and add post-processing codes

2）Verify that the result is correct on board

2、Function Integration

1. SDKV0.1:

1）call the abstract interfaces based on case demo and compile it into so file

2）Call the interfaces encapsulated in your own application

2. SDKV1.X

1）Call the interfaces in the current demo in your own application directly

Related documents: DDK_xxxx_SDK_V1.x.x API.doc, DDK_x.x.x.x_SDK_V1.x.x.tar.gz

3、Android APK Development

After the previous development based on demo is completed, the next step is to migrate the corresponding demo flow into APK.

Refer to <Android NN JNI_Development_Guide_v1.3> for APK JNI development

For SDKv1.x, please refer to <DDK_x.x.x.x_SDK_Vx.x.x API> document and SDKV1.X development package to perform development