

Amlogic

NN 集成指导


Revision: 0.2

Release Date: 2022-02-10

Copyright

© 2022 Amlogic. All rights reserved. No part of this document may be reproduced. Transmitted, transcribed, or translated into any language in any form or by any means with the written permission of Amlogic.

Trademarks

 , 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 of 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

变更说明

版本 0.2 (2022-01-05)

第二版。

1. 增加 nn engine 工具介绍
2. 增加 aml_perlayer_visual 工具介绍

版本 0.1 (2021-08-10)

第一版。

Confidential!
nick@khadas.com
2022-03-17 19:26:35

目录

变更说明.....	ii
1. 简介	4
1.1 Amlogic NN Workflow.....	4
1.2 集成流程简介	4
2. Release Source 介绍	6
2.1 简介.....	6
2.2 文档以及工具介绍.....	6
3. 模型设计指导.....	7
3.1 简介	7
3.2 Amlogic NN 芯片介绍	7
3.3 设计指导.....	7
4. 模型转换指导.....	8
4.1 简介.....	8
4.2 环境确认.....	8
4.2.1 版本确认	8
4.2.2 资料获取	9
4.3 模型转换.....	9
4.3.1 环境准备	9
4.3.2 模型转换	10
4.3.3 性能测试	10
5. 模型集成指导.....	12
5.1 SDK 版本选择.....	12
5.1.1 背景介绍.....	12
5.1.2 SDK 版本对比.....	12
5.2 模型集成.....	12
5.2.1 精度确认	12
5.2.2 板侧结果确认	13
5.2.3 模型运行信息可视化.....	13
5.2.4 模型集成	13

1. 简介

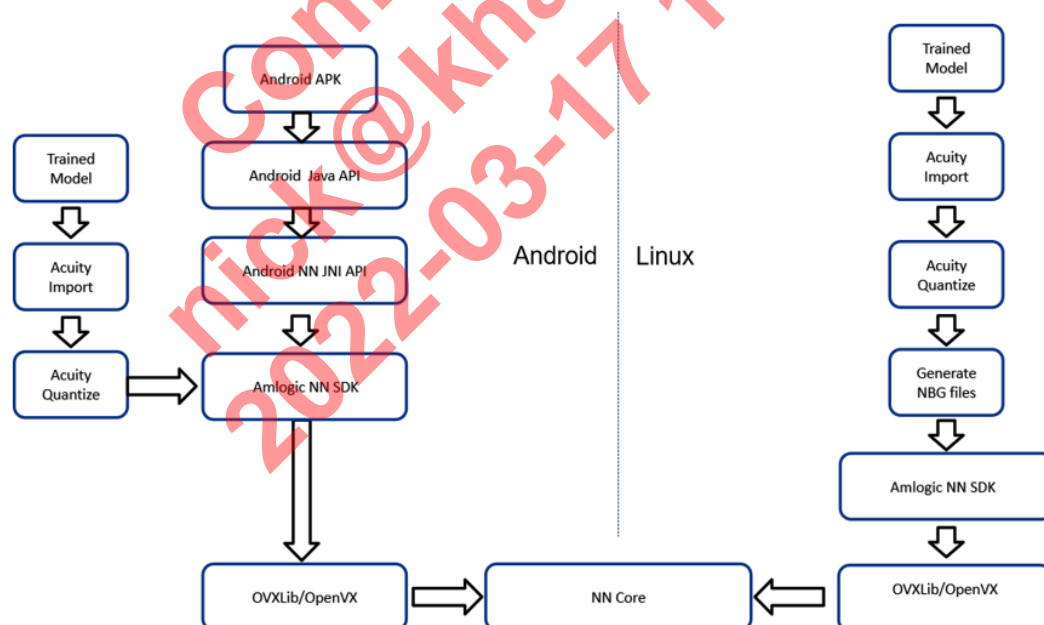
本文介绍 Amlogic NN 概念和 Amlogic NN 的集成步骤，用以指导 Amlogic 相关开发人员以及 Amlogic 客户。

Amlogic NN 简介如下：

1. Amlogic NN 芯片采用的是 GPU 架构，针对 conv、deconv、pool、relu 等一系列的神经网络算子进行优化加速
2. 通过 Openvx1.2 来进行底层优化加速
3. 支持 caffe、tensorflow、tflite、darknet、onnx、Keras、Pytorch 等开源框架
4. 支持 int8、int16、uint8 等三种量化方式，同时也支持通过混合量化来提升精度（最新 IP 支持 Perchannel 量化方式）
5. 支持的开源框架模型需先用上面列出的三种量化方式进行量化处理，然后导出芯片能识别并加载的 nbg(Network Binary Graph)文件
6. 支持的神经网络参照：<https://github.com/VeriSilicon/acuity-models>
7. 支持 Android、Linux、FreeRTOS 等系统。

1.1 Amlogic NN Workflow

Amlogic NN 工作流程如下图



1.2 集成流程简介

整个集成流程大致分为如下几步：

1. 选定模型(可自行训练或者网上找已经训练好的模型)

2. 确认开发环境(单板环境需跟模型工具以及 sdk 包一一对应)
3. 模型转换(单板上只支持量化模型, 需进行模型转换)
4. 选定集成 sdk 版本(当前提供了俩个 sdk 版本, 优劣不同)
5. 参照提供的 demo, 进行 demo 集成
6. 基于已集成好的 demo, 整理集成到自己的应用框架里

Confidential!
nick@khadas.com
2022-03-17 19:26:35

2.Release Source 介绍

2.1 简介
















本章节会介绍当前 Amlogic NN 对外 release 那些文档、工具以及对应的使用范围。

2.2 文档以及工具介绍

以下是当前 Amlogic 对外 release 的文档以及工具，可能均能获取到。如有遗漏，请找对应接口人帮忙申请即可。

NUM	Document	Remarks
1	AMLNN Convolution Acceleration Tips.pdf	模型设计指导文档
2	Neural Network Layer and Operation Support Guide.docx	算子支持列表文档
3	Model_Transcoding and Running User Guide_V0.9 Model_Transcoding and Running User Guide_V0.9_Eng	模型转换指导中英文文档
4	DDK_xxxx_SDK_V1.x.x API.docx DDK_xxxx_SDK_V1.x.x API Eng.docx	SDK_V1.8 API 中英文文档
5	NN Tool FAQ_V0.4.docx NN Tool FAQ_V0.4_Eng.docx	NN 常见问题中英文文档
6	DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz	模型转换工具
7	DDK_x.x.x.x_SDK_V0.1.tar.gz	SDK_V0.1 开发包
8	DDK_x.x.x.x_SDK_V1.x.x.tar.gz	SDK_V1.8 开发包
9	Android NN JNI 开发流程_v1.3.doc	Android jni 开发指导
10	Amlogic NN Integration Guide.docx Amlogic NN Integration Guide Eng.docx	Amlogic NN 集成指导文档
11	aml_nnengine_Tool_Introduction_v0.1.docx aml_nnengine_Tool_Introduction_v0.1_en.docx	Aml nn engine 使用指导文档
12	aml_perlayer_visual_tool_v0.1.docx aml_perlayer_visual_tool_v0.1_en.docx	Aml nn 可视化工具

Note: x.x.x.x 为对应的 DDK 版本号。如 DDK6.4.6.2 版本，会有以下 source:

-  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. 模型设计指导

3.1 简介

本章节只用于指导有环境有能力进行模型设计、训练的客户，指导客户如何设计出更符合 Amlogic NN 芯片加速规则的模型。对只设计到模型集成的客户，可以跳过这一章节。

3.2 Amlogic NN 芯片介绍

Amlogic NN 芯片里包含有三个计算模块，分别为 NN core、PPU、TP，对应介绍如下：

NN core：

NN 运算的主要模块，包含大量的 mac 单元，主要执行卷积等大规模的运算。

优点：是运算速度快。缺点：支持的算子较少。

TP：

tensor 处理单元，是 NN core 的补充，完成 tensor 的一些处理。

优点：是支持的算子较 NN core 多。缺点：运算速度慢。

PPU：

类似 gpu 的模块，可编程，支持 float，是自定义算子的主要运算平台。

优点：可编程，理论上支持所有算子，精度高。缺点：运算速度慢。

Note: <Neural Network Layer and Operation Support Guide>中有列出支持的算子以及算子运行在 NN 的哪个模块。

3.3 设计指导

设计原则：

1. 尽可能多、且连续的使用跑在 NN core 上的算子来构建模型
2. 尽可能使用能融合的算子来构造模型
3. 其他加速规则请参照 AMLNN Convolution Acceleration Tips 文档

涉及文档：

AMLNN Convolution Acceleration Tips.pdf

Neural Network Layer and Operation Support Guide

4. 模型转换指导

4.1 简介

本章节主要介绍如何确认单板上 NN 的版本并申请对应的资料以及模型转换过程中涉及到的工具文档。

4.2 环境确认

当前 Amlogic NN 的 release 版本一直在迭代更新，单板上 NN driver 的版本均有对应的模型转换工具版本、sdk 版本。但由于不同分支上更新不及时，所以集成之前，首先需要确认当前的集成环境是哪个 ddk 版本，然后使用对应的 sdk 以及模型转换工具去集成，或者确认是否需要更新到最新版本。

4.2.1 版本确认

1、获取单板上 NN driver 版本：

在串口或者 adb 窗口执行命令：`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：如果单板启动时间很长，Galcore 信息可能会被冲掉。如果没有对应信息，可以重启单板再执行一次。

2、根据上面查到的 Driver 版本，明确需使用的模型转换工具以及 SDK 的版本

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

例如：

当前查看到的 driver 版本号是 6.3.3.210826, 那么当前的开发环境对应的是 DDK6.3.3.4 版本。

4.2.2 资料获取

明确了集成环境的 DDK 版本之后, 需拿到对应版本的工具以及文档。

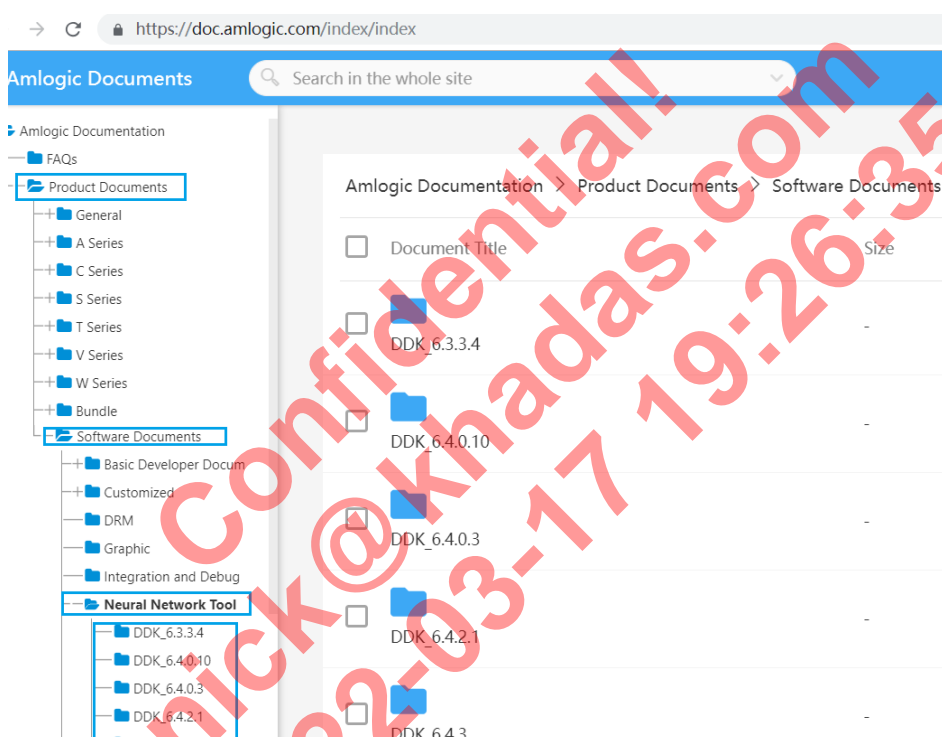
如客户查到自己单板上版本为 **DDK6.4.3**, 那么需要拿到:

模型转换工具 `acuity-toolkit-binary-5.14.1.tar.gz`

SDK 包: `DDK_6.4.3_SDK_V0.1.tar.gz/DDK_6.4.3_SDK_V1.x.x.tar.gz`

一般 Amlogic 接口人都会帮忙申请并 release 给客户, 客户需要自己确认下拿到的是否是对应的版本工具。如果不匹配, 需找对应接口人重新申请对应版本的文档以及工具。

申请链接: <https://doc.amlogic.com/index/index>



4.3 模型转换

客户选定好需要集成的模型后, 第一步就需要使用 Acuity_tool 工具进行模型转换。

4.3.1 环境准备

1. Ubuntu 上解压获取到的模型转换工具:

```
tar -xzf DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz
```

Note:需直接在 linux 环境解压, 在 windows 环境解压后再拷贝到 linux 环境会出现一些 link 问题。

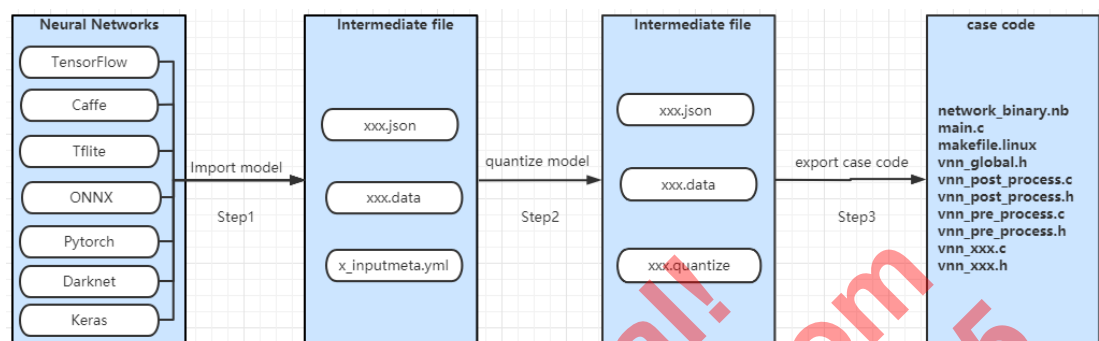
2. 安装环境

参照 acuity-tool-binary-xxxx 目录下的 ReadMe.txt 文档进行环境安装

3. Check 环境

环境安装 ok 之后，可以使用 acuity-tool-binary-xxxx/demo 目录里提供的 demo 进行模型转换测试

4.3.2 模型转换



模型转换里流程分三步：导入模型、量化模型、导出 case demo。

具体详细步骤参照模型转换文档 3.4 章节。

模型转换需注意的点：

- 1、在 xxx_inputmeta.yml 中设置 channel-mean-values，对应值需跟模型训练时对输入图片的处理时用到的 mean-values 保持一致。
- 2、量化数据最好提供 200~500 组，确保量化过程中统计出的最大/最小值为实际运行场景的最大/最小值范围。
- 3、对于不同通道使用不同 scale 的情况，客户可以自行做前处理将输入转换成 npy 文件，然后量化时提供 npy 文件即可。

Note：本章节涉及到工具文档为：

1. 模型转换文档：

<Model_Transcoding and Running User Guide_V1.0>

<Model_Transcoding and Running User Guide_V1.0_Eng>

2. 模型转换工具：

模型转换工具：DDK_x.x.x.x_Tool_acuity-toolkit-binary-x.xx.x.tar.gz

4.3.3 性能测试

完成 4.3.2 章节中的模型转换之后，能获取到一个简易的 demo(编译后即可运行)，如下图：

```
dengliu@amlogic2018-Precision-3630-Tower:~/tool/acuity-toolkit-rel-5.0.0/bin/v4$ ls nbq_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
```

当前使用 Acuity_tool 工具进行模型转换之后转出的都是一个简易的 demo，都是使用的分类模型的模板，即：当前 demo 只对输出做了 top5 的后处理。

性能测试：

1. 参照 sdkV0.1 压缩包中 android 或者 buildrootsdk 里 readme.txt 文件将转换出的 case demo 编译出可执行文件
2. 然后 push 到板子上执行，即可获取性能数据-帧率。

Note：本章节涉及到工具文档为：

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

android_sdk/buildroot_sdk 均有 ReadMe.txt 指导文档。

5. 模型集成指导

完成模型转换之后，需进行最后的集成工作。

5.1 SDK 版本选择

5.1.1 背景介绍

Sdkv0.1 版本是基于模型转换出的 case demo 整理的一个 sdk 开发包。提供一个基于 case demo 去编译、集成的开发环境。

SdkV1.x.x 版本是 Amlogic 基于 case demo 提取并封装的一套接口，用以屏蔽 DDK 版本升级时转出的 case demo 中一些接口的变动。

因为当前有些流程涉及到使用 sdkv0.1 开发包，所以当前两套 sdk 均对外 release。客户需自行选择使用哪套 sdk 进行集成开发。

5.1.2 SDK 版本对比

	SDK V1.x.x	SDK V0.1
Introduction	Amlogic 封装的接口	原始 sdk, Acuity_tool 导出的 case demo
so 依赖	libnnsdk.so	libovxlib.so、libCLC.so、libGAL.so、 libOpenCL.so、libOpenVX.so、 libOpenVXU.so、libVivanteOpenCL.so、 libVSC.so
头文件依赖	nn_sdk.h/nn_util.h	vsi 相关头文件(较多, 暂不列出)
集成接口	使用已封装好的接口	需基于 case demo 自行提取接口
版本升级	只需更新 nbg 文件	需更新 nbg 文件以及模型创建文件以及需重新编译

较早版本只能使用 sdkv0.1 版本，DDK6.4.3 及更新的版本才支持 sdkv1.x 版本。

如果使用 sdkv0.1: DDK_x.x.x.x_SDK_V0.1.tar.gz

如果使用 sdkv1.x: DDK_x.x.x.x_SDK_V1.x.x.tar.gz

5.2 模型集成

Amlogic NN 只是提供一个运行平台，即：只运行模型的推理部分。

模型运行在 NPU 上的最后一个节点数据转换成最终的显示结果的过程为后处理部分，这部分需客户自行去实现。例如：分类模型，当前 NN 上出来的数据即为一个(1,1000)的数组，客户获取到这个数组后，需做 top1 的后处理，然后才呈现出最终分类结果。

5.2.1 精度确认

一般建议使用最简洁的 demo 去测试下板侧精度。如果没有做该操作，发现最终结果不对，也可返回来做本次操作。

1. 基于 Acuity_tool 工具，在 pc 端 inference 出 float 和 quantize 的输入及输出

参照 Model_Transcoding and Running User Guide 文档 3.4 章节 Step4

2. sdkv0.1: 参照 FAQ 文档 4.2 章节步骤确认精度。

Note: 使用 pc 端 inference 出的 tensor 文件做板侧输入, 是为了保持输入的一致性。Pc 端 inference 出的 tensor 文件是做完前处理后的数据, 板侧运行时不会对其进行二次处理。

涉及到文档:

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 板侧结果确认

5.2.1 章节只能快速确认单帧输入的板侧结果跟 PC 端结果的相似性。有的模型基于 C 语言添加后处理比较繁琐, 无法快速确认板侧结果对应的最终显示结果是否正确。当前提供一种方式能让客户快速确认板侧结果对应的最终显示结果是否正确。

nnEngine 是 Amlogic 开发的一套基于 nnsdk 开发包的模型推理引擎, 利用 ADB 协议实现嵌入式设备端和服务器端的数据传输和模型运行控制。用户可通过提供的 aml_nn_engine python 库实现模型推理和性能指标获取功能。

涉及文档:

aml_nnengine_Tool_Introduction_v0.1.docx

5.2.3 模型运行信息可视化

有些客户需要确认模型运行在单板上的每个 layer 的执行时间以及带宽, Perlayer visual 能满足此需求。

Perlayer Visual 是 Amlogic 开发的基于硬件信息打印的模型各层信息可视化工具。方便用户评估模型的运行时间和带宽占用, 以及定位模型耗时和带宽占用过大的问题。

涉及文档:

aml_perlayer_visual_tool_v0.1.docx

5.2.4 模型集成

模型集成的主要工作量就是, 获取到最后一个节点的数据后, 自行添加后处理代码, 将数据转换成最终显示结果。

1、添加后处理代码

1. SDKV0.1:

1) 参照转换出的 case demo, 修改 vnn_post_process.c 文件中 show_top5 接口, 添加当前模型对应的后处理代码

2) 板端验证最终结果是否正确

2. SDKV1.X:

1) 参照 DDK_xxxx_SDK_V1.x.x API.doc 文档以及 DDK_x.x.x.x_SDK_V1.x.x.tar.gz 中的 demo 程序, 构建 demo case 并添加后处理代码。

2) 板端验证最终结果是否正确

2、功能集成

1. SDKV0.1:

1) 基于 case demo 提取调用接口并编译成 so

2) 在自己应用中调用自己封装的接口

2. SDKV1.X

1) 直接基于自己应用调用当前的 demo 中的接口即可

涉及文档: DDK_xxxx_SDK_V1.x.x API.doc、DDK_x.x.x.x_SDK_V1.x.x.tar.gz

3、Android apk 开发

前面基于 demo 的开发完成之后, 当前只需要将对应 demo 流程移植到 APK 中。

参照<Android NN JNI_Development_Guide_v1.3>进行 APK JNI 开发。

如果使用 sdkv1.X 版本, 请结合<DDK_x.x.x.x_SDK_V1.x.x API>文档以及 SDKV1.X 开发包进行开发。