



Amlogic

NN 工具 FAQ


Revision: 0.5

Release Date: 2021-08-10

Copyright

© 2021 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

变更记录

版本 0.5 (2021-08-10)

第五版。

相对于上一版本，修改了以下内容。

- 修改了 1.1、3.4、4.6。

版本 0.4 (2021-01-15)

第四版。

相对于上一版本，修改了以下内容。

- 修改了 1.3、3.2 、4.3、3.4。
- 增加了 3.8、3.9、4.7。

版本 0.3 (2020-06-16)

第三版。

相对于上一版本，修改了以下内容。

- 修改了 4.1 、4.3。
- 增加了 1.6、4.5。

版本 0.2 (2020-05-12)

第二版。

相对于上一版本，修改了以下内容。

- 修改了 1.4, 4.2 和 4.3。
- 增加了 1.5 和 4.5。

版本 0.1 (2020-04-17)

第一版。

目录

变更记录.....	ii
1. 框架.....	4
1.1 当前 Amlogic NN 支持哪几种开源框架?	4
1.2 当前 Amlogic NN 支持什么格式的模型?	4
1.3 当前 Amlogic NN 支持哪些通用模型?	4
1.4 是否能脱离 acuity 量化这一套框架来使用 Amlogic NN?	4
1.5 NPU 上是否能同时运行多个 nn 任务.....	4
1.6 NPU 框架, NN core、PPU、TP 分别是什么, 处理哪些运算, 分别的优缺点?	5
2. 工具.....	6
2.1 当前模型 caffe 框架, 是不是可以不用安装 tensorflow?	6
2.2 Acuity 工具量化方式是否与 tensorflow 量化方式一致?	6
3. 模型转换	7
3.1 模型转换会有精度损失吗?	7
3.2 量化时候, datet 需要多少张图片? 数量和精度是否成正比?	7
3.3 对于不支持模型如何处理, 如何实现不支持 layer?	7
3.4 使用 acuity 工具转换模型, 在导出 case 代码时, 如何设置对应板子正确的参数?	7
3.5 使用 point-16 量化之后, 再次使用 point-8 时,发现系数文件始终无法缩小? ...	8
3.6 caffe 模型转换过程中报错: Not supported caffe net model version(v0 layer or v1 layer).....	8
3.7 模型 input 包含序列的话, shape 应该怎么定义?	8
4. 集成类.....	11
4.1 模型运行过程中出错, 如何确认问题点?	11
4.2 板子上运行结果精度差, 如何定位?	11
4.3 如何统计模型运行过程中的带宽以及 NPU 使用率?	12
4.4 demo 集成后, 发现前处理时间较长, 是整个流程中的计算瓶颈, 怎么解决? ..	13
4.5 从 vsi_nn_ConvertTensorToData 保存指针后再次调用了 vsi_nn_RunGraph, 然后再去使用指针是否安全?	15
4.6 如何得到每个 op 的运算结果和耗时?	15

1. 框架

1.1 当前 Amlogic NN 支持哪几种开源框架？

解答：当前支持的神经网络框架以及对应版本信息如下：

NN framework	Info
Caffe	https://github.com/verisilicon/caffe
TensorFlow	Version: 2.3.0
Darknet	
Onnx	Version: 1.6.0
Tflite	Version: 2.3.0
Keras	Version: 2.2.5
Pytorch	Version: 1.2.0

1.2 当前 Amlogic NN 支持什么格式的模型？

解答：当前 Amlogic NN 只支持量化后的模型，需将支持的开源框架对应的模型进行量化处理，量化方式有三种 Int8、int16、Uint8。

1.3 当前 Amlogic NN 支持哪些通用模型？

解答：SqueezeNet、VGG、Resnet*、Inception、RetinaNet*、MobileNet*、LSTM*、GRU*、SSD、Yolo*、FasterRCNN（MaskRCNN 不支持）。

支持的模型列表可参考：<https://github.com/VeriSilicon/acuity-models/tree/master/models>

1.4 是否能脱离 acuity 量化这一套框架来使用 Amlogic NN？

解答：对于有此需求的客户，当前方式也支持。当前已适配支持 amnn，可以参考 amnn 那套流程。

1.5 NPU 上是否能同时运行多个 nn 任务

解答：支持多线程多进程操作，但硬件是时分复用的。

1.6 NPU 框架，NN core、PPU、TP 分别是什么，处理哪些运算，各自的优缺点？

1、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。

NN core 优点：是运算速度快。缺点：支持的算子较少。

2、TP：tensor 处理单元，是 NN core 的补充，完成 tensor 的一些处理。

TP 优点：是支持的算子较 NN core 多。缺点：运算速度慢。

3、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。

PPU 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：分别处理哪些算子，这些信息体现在 release 的算子文档《[Layer and Operation Support Guide \(01\).docx](#)》中

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

2. 工具

2.1 当前模型 caffe 框架，是不是可以不用安装 tensorflow？

解答：TensorFlow 是必须要安装的，当前量化和 inference 都是用的 tensorflow 的 API。

2.2 Acuity 工具量化方式是否与 tensorflow 量化方式一致？

解答：是一致的。asymmetric_quantized-u8 是 google 支持的量化方式，根据”[Quantizing deep convolutional networks for efficient inference: A whitepaper](#)” 论文的描述，对于大部分网络，这种量化方式对精度的损失最小。

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

3. 模型转换

3.1 模型转换会有精度损失吗？

解答：模型转换涉及到 float 数据量化成 int8/uint8/int16 数据，会有一定程度上的精度损失。

3.2 量化时候，datet 需要多少张图片？数量和精度是否成正比？

解答：一般建议图片数量在 200~2000 张左右，尽量是模型运行时场景的图片，或者是训练模型验证集里的图片。这对量化效果有正向作用。但数量和精度不成正比关系。

注：当前量化时默认 --batch-size 是 100，默认-epochs 是 1，所以默认最多只会用到 100 张图片。当图片数量>100 时，建议添加量化参数--batch-size(默认 100)和--epochs(默认 1)，epochs*batch_size=图片数量。例如 5000 张图片，设置参数为：--batch-size 100 --epochs 50。电脑内存较小时，可以将 batch_size 的值设置改小。

3.3 对于不支持模型如何处理，如何实现不支持 layer？

解答：

1. 对于不支持的开源框架模型，需要将之转换成当前 Amlogic NN 支持的开源框架对应模型。
2. 对于不支持的 layer，当前工具是以 binary 方式 release 的，暂不支持客户自己添加自定义 layer。

3.4 使用 acuity 工具转换模型，在导出 case 代码时，如何设置对应板子正确的参数？

解答：执行命令：cat /proc/cpuinfo

查看倒第二行数 Serial 对应的数列码前四个字符(例如 290a, 290b 等)，根据如下对应关系来确定设置的值。

PID	Serial
0X7D	Serial : 290a 70004ba55adb38423231474c4d4
0X88	Serial : 290b 70004ba55adb38423231474c4d4
0X99	Serial : 2b0a 0f00df472d383156314d534c4d4
0XA1	Serial : 300a 010245bbf1e131305631434c4d4
	Serial : 300b 010200151d00000139365838535
0X99	Serial : 2f0a 0c00d2f1ef293156324d544c4d41
0XB9	Serial : 2f0b 06009f45301d3356324d544c4d41
0XBE	Serial : 330a 0304d93895a932305632434c4d4
	Serial : 330b 0304d93895a932305632434c4d4
0XE8	Serial : 380a 0201000000008d94ad5d3256335
	Serial : 380b 0201000000008d94ad5d3256335

例如：

序列号前四位为 290a 时，设置参数 “--optimize VIPNANOQI_PID0X7D”；

序列号前四位为 290b 时，设置参数 “--optimize VIPNANOQI_PID0X88”；

序列号前四位为 2b0a 时，设置参数 “--optimize VIPNANOQI_PID0X99”；

序列号前四位为 300a 时，设置参数 “--optimize VIPNANOQI_PID0XA1”；

序列号前四位为 2f0b 时，设置参数 “--optimize VIPNANOQI_PID0XB9”；

序列号前四位为 330a 时，设置参数 “--optimize VIPNANOQI_PID0XBE”；

序列号前四位为 380a 时，设置参数 “--optimize VIPNANOQI_PID0XE8”。

如果序列号前四位在上述表格中没有统计出来，当前建议使用序列号前三位的对应的参数尝试下。

3.5 使用 point-16 量化之后，再次使用 point-8 时，发现系数文件始终无法缩小？

解答：一步量化生成的文件未删除时，再次进行量化，不会生成新的文件。建议添加默认参数 --quantized-rebuild 到量化命令中，此时中间文件会默认被先删除，再做量化处理。也可以手动对量化中间文件进行删除。

3.6 caffe 模型转换过程中报错：Not supported caffe net model version(v0 layer or v1 layer)

解答：当前 caffemodel 或者 prototxt 文件版本过老，需要使用工具更新下 prototxt 或者 caffe 模型文件

解决方案：更新 prototxt 工具:upgrade_net_proto_text-d。

更新 caffemodel 工具:upgrade_net_proto_binary-d

获取方式：编译 caffe 源码，tools 目录下会生成对应的可执行文件

3.7 模型 input 包含序列的话，shape 应该怎么定义？

解答：将序列值用--predef-file xxx.npz 来传入，在 import 导入模型的时候添加该参数值。要支持某些逻辑控制，需提供一个 npz 格式的预定义文件，通过 np.savez('\prd.npz', placeholder_name=predefine_value) 来生成。如果占位符中有数组不支持字符，你可以用 map 作为关键字来存储字典作为映射到普通键值。例如：“inference/train_satge”是占位符，不支持 numpy，你可以使用：

np.savez('prd.npz', stage=False, map={'stage':'inference/train_stage'}).

3.8 量化输入数据只能是图片么？有的模型输入数据不是图片的怎么给量化数据？

解答：量化数据可以不是图片。当前可以将数组或者 txt 格式文件数据保存成 npy 文件，然后量化的时候，dataset.txt 里面使用 npy 文件的路径替换图片路径。如：

1、使用或者参照 acuity_tools_xxx/bin/tools/npv_input_generate.py 脚本，保存 npy 文件。

```
python3 npv_input_generate.py --action tensor2npv --src-platform tensorflow --input-file xxx.tensor --shape "224 224 3" --numpy-data-file input.npv
```

2、量化时，source 仍旧设置 txt，source_file 对应的 dataset.txt 里面列出 npv 的路径即可。

3.9 Caffe 模型常见问题

解答：

a、caffe 模型 input 层的格式：

Error	Right
<pre> name: "MobileNet-SSD" input: "data" input_shape { dim: 1 dim: 3 dim: 300 dim: 300 } layer { name: "conv0" type: "Convolution" bottom: "data" top: "conv0" param { lr_mult: 0.1 decay_mult: 0.1 } } </pre>	<pre> 1 name: "MobileNet-SSD" 2 layer { 3 name: "data" 4 type: "Input" 5 top: "data" 6 input_param { 7 shape { 8 dim: 1 9 dim: 3 10 dim: 300 11 dim: 300 12 } 13 } 14 } 15 layer { 16 name: "conv0" 17 type: "Convolution" </pre>

b、prototxt 中出现不支持 layer 时容易导致转换失败，例如检测类模型，不支持 DetectionOutput layer，需要手动删掉该 layer。

```
0 layer {  
1   name: "detection_out"  
2   type: "DetectionOutput"  
3   bottom: "mbox_loc"  
4   bottom: "mbox_conf_flatten"  
5   bottom: "mbox_priorbox"  
6   top: "detection_out"  
7   include {  
8     phase: TEST  
9   }  
0   detection_output_param {  
1     num_classes: 21  
2     share_location: true  
3     background_label_id: 0  
4     nms_param {  
5       nms_threshold: 0.45  
6       top_k: 100  
7     }  
8   }  
9 }
```

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

4. 集成类

4.1 模型运行过程中出错，如何确认问题点？

解答：

1. 确认板子上是否有 NPU 环境
确认 NN 模型加载：lsmod 结果中有 galcore
确认有 NN 设备节点：ls -l /dev/galcore 权限为 664

2. 确认当前转的 nbg 代码是否与板子一致

参照 3.1.4 问题

3. 查看具体错误点

设置环境变量，然后再次执行程序。

Linux:

```
export VSI_NN_LOG_LEVEL=4
```

```
export VIV_VX_DEBUG_LEVEL=1
```

Android:

```
setprop VSI_NN_LOG_LEVEL 4
```

```
setprop VIV_VX_DEBUG_LEVEL 1
```

4. 问题分析

- 1) 如果第一步、第三步出错，直接反馈给 Amlogic 工程师
- 2) 如果第二步出错，重新转换 nbg 文件再次运行。

4.2 板子上运行结果精度差，如何定位？

解答：

1. 确认 float 类型的精度和原始平台测试结果相近

在执行完 import 导入模型之后，使用工具中提供的 demo 里的 inference.sh 脚本，运行出 float 类型的结果，将此和原始平台结果对比。

2. 确认量化的精度是否和 float 类型的精度相近

在执行完量化步骤后，使用 inferece.sh 脚本，运行处量化后的结果，将之与 float 类型结果进行比较。可以使用 acuity_tool_xxx/bin/tools/compute_tensor_similarity.py 来比较，会输出俩个文件的余弦相似度以及欧式距离。

3. 确认板侧结果是否和量化结果相近

- 1) 将转换的 case 代码中 vnn_post_process.c 里面的 save_output_data 函数做如下修改：

```
snprintf(filename, _DUMP_FILE_LENGTH, "output%u_%s.dat", i, shape);
```

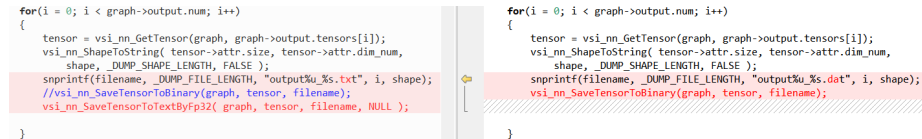
```
vsi_nn_SaveTensorToBinary(graph, tensor, filename);
```

改成:

```
snprintf(filename, _DUMP_FILE_LENGTH, "output%u_%s.txt", i, shape);
```

```
vsi_nn_SaveTensorToTextByFp32( graph, tensor, filename, NULL );
```

例如 :



```
for(i = 0; i < graph->output.num; i++)
{
    tensor = vsi_nn_GetTensor(graph, graph->output.tensors[i]);
    vsi_nn_ShapeToString( tensor->attr.size, tensor->attr.dim_num,
        shape, _DUMP_SHAPE_LENGTH, FALSE );
    snprintf(filename, _DUMP_FILE_LENGTH, "output%u_%s.txt", i, shape);
    //vsi_nn_SaveTensorToTextByFp32( graph, tensor, filename, NULL );
    vsi_nn_SaveTensorToTextByFp32( graph, tensor, filename, NULL );
}

for(i = 0; i < graph->output.num; i++)
{
    tensor = vsi_nn_GetTensor(graph, graph->output.tensors[i]);
    vsi_nn_ShapeToString( tensor->attr.size, tensor->attr.dim_num,
        shape, _DUMP_SHAPE_LENGTH, FALSE );
    snprintf(filename, _DUMP_FILE_LENGTH, "output%u_%s.dat", i, shape);
    vsi_nn_SaveTensorToBinary(graph, tensor, filename);
}
```

2) 编译出可执行文件并在板侧运行处结果。

4. 问题分析

- 1) 如果第一步出问题, 需检测 import 导入模型时候的输入输出节点与原始平台跑时是否一致。
- 2) 如果第二步出问题, 余弦相似度<0.95 或者欧氏距离值较大, 则表示量化出错或者量化效果不理想, 此时可以检查当前量化参数是否有错误或者反馈给 Amlogic 工程师。
- 3) 如果第三步出问题, 则表示板侧结果与量化后结果不一致, 问题出现在 nbg 文件打包过程或者是 driver 里面, 此时需将问题反馈给 Amlogic 工程师。

Note: 有时会存在如下情况: 板侧结果和 PC 端 inference 量化后的结果有一些差异, 当前没有一个明确判断余弦相似度以及欧氏距离过大的标准。除了那种一下就能判断出结果完全错误的情况, 其他情况下, 需要客户自己添加对应后处理代码, 如果得到的最终结果有差异: 仿真工具正确/板侧错误, 即可认为是板侧和仿真结果差异大, 此时可以将问题反馈给 Amlogic。

4.3 如何统计模型运行过程中的带宽以及 NPU 使用率?

解答: 运行 nbg case 之前重新加载 npu ko 以及设置一些环境变量即可获得对应信息打印。

1) 设置环境变量

2) Linux:

a. 卸载 npu 模块: `rmmod galcore`

b. 加载 npu 模块: `insmod /xxx/xxx/galcore.ko gpuProfiler=1 showArgs=1`

galcore.ko 路径: `/lib/modules/4.19.xxx/kernel/amlogic/npu/ (C1)`

`/lib/modules/4.9.113/kernel/amlogic/npu/ (W400/SM1)`

Note: 如果上面路径没有 galcore.ko, 在/lib 下执行 `find -name galcore.ko` 搜 ko 的路径。

c. 添加环境变量:

```
export VIV_VX_PROFILE=1
```

```
export VIV_VX_DEBUG_LEVEL=1
```

Android:

- a. 卸载 npu 目录: `pkill network && rmmod galcore`
- b. 重新加载 npu 模块: `insmod /xxx/xxx/galcore.ko gpuProfiler=1 showArgs=1`
galcore.ko 路径: `/vendor/lib/modules/galcore.ko`
- d. 添加环境变量:
 - `setprop VIV_VX_PROFILE 1`
 - `setprop VIV_VX_DEBUG_LEVEL 1`
 - `setprop VP_PROCESS_NAME xxxx` (xxx 为可执行文件名称)

Note: 1、当前 android 版本上如果 NPU DDK 版本 < 6.4.3, 带宽信息打印部分打印函数失效, 如有需要, Amlogic 工程师会提供一个新编译的 libGAL.so。

2、确认 DDK 版本方式: 串口或者 adb 窗口执行 `dmesg |grep Galcore` 就能查看到版本号。

3) 运行模型并根据打印 log 统计带宽

运行中会有如下打印:

```
[ 1] READ_BANDWIDTH (MByte): 63.048161
[ 2] WRITE_BANDWIDTH (MByte): 34.781154
[ 3] READOCB_BANDWIDTH (MByte): 21.768029
[ 4] WRITEOCB_BANDWIDTH (MByte): 21.041409
[ 5] GPUTOTALCYCLES: 23352854
[ 6] GPUIDLECYCLES: 1042473
```

OCB 是 SW TILING 节约的带宽, 实际的带宽等于 `READ_BANDWIDTH + WRITE_BANDWIDTH - READOCB_BANDWIDTH - WRITEOCB_BANDWIDTH`

使用率: $(TOTALCYCLES - IDLECYCLES) / TOTALCYCLES$

4.4 demo 集成后, 发现前处理时间较长, 是整个流程中的计算瓶颈, 怎么解决?

解答: 模型前处理一般是输入数据减均值乘以倍率之后, 将得到的 float 数转为 INT16 或者 Int8 的数, 然后传给 NPU 进行处理。当前转换出的 case 代码中, float 转 Int16 和 Int8 用的接口是 `vsi_nn_Float32ToDtype`, 建议将该接口替换掉, 替换方式如下:

Int8/int16: (fl 根据输入输出 tensor 可以获取: `tensor->attr.dtype.fl`)

```
float fl = pow(2., tensor->attr.dtype.fl);
float32 ---> int8: value_int8 = value_float * fl;
float fl = pow(2., -tensor->attr.dtype.fl);
Int8 ---> float: value_float = value_int * fl;
```

u8: (scale/zero_point 根据输入输出 tensor 可以获取:

```

    scale = tensor->attr.dtype.scale
    zero_point= tensor->attr.dtype.zero_point。
    float32 ---> u8:    value_uint8 = value_float/scale + zero_point;
Int    ---> float:    value_float = (value_uint8 -zero_point)*scale;
FP16: 有时有的输出 tensor 是 float16 格式的, 此时需将 float16 反量化成 float32
if ( attr.dtype.vx_type == VSI_NN_TYPE_FLOAT16 ) {
    uint8_t *tensor_data = (uint8_t *)vsi_nn_ConvertTensorToData(graph, tensor);
    vx_int16* data_ptr_32 = (vx_int16*)tensor_data;
    Float16ToFloat32(data_ptr_32 ,buffer ,length);
}
static vx_float32 Float16ToFloat32(const vx_int16* src , float* dst ,int lenth) {
    vx_int32 t1;
    vx_int32 t2;
    vx_int32 t3;
    vx_float32 out;
    for (int i = 0 ;i < lenth ;i++) {
        t1 = src[i] & 0x7fff;           // Non-sign bits
        t2 = src[i] & 0x8000;           // Sign bit
        t3 = src[i] & 0x7c00;           // Exponent

        t1 <<= 13;                       // Align mantissa on MSB
        t2 <<= 16;                       // Shift sign bit into position

        t1 += 0x38000000;                 // Adjust bias
        t1 = (t3 == 0 ? 0 : t1);          // Denormals-as-zero
        t1 |= t2;

        *((uint32_t*)&out) = t1;         // Re-insert sign bit
        dst[i] = out;
    }
    return out;
}

```

4.5 从 vsi_nn_ConvertTensorToData 保存指针后再次调用了 vsi_nn_RunGraph，然后再去使用指针是否安全？

解答：是安全的，vsi_nn_ConvertTensorToData 返回的指针是该接口内部 malloc 出来的一块 buffer，不受其他接口影响。只是使用了该指针内容之后，需要调用 vsi_nn_Free 去是否它。

4.6 如何得到每个 op 的运算结果和耗时？

解答：测试每个 op 的运行结果和耗时，需要运行 normal case（非 nbg），normal case 在板上运行时，会有一个在线编译过程，加载过程会比较慢。

1、生成 normal case：

在模型转换过程中，最后一步导出 case code 的时候，删除下面三个参数：

```
--optimize VIPNANOQI_PID0X88 \
--viv-sdk ../bin/vcmdtools \
--pack-nbg-unify
```

此时，会在模型所在目录生成 case code，将 *.c *.h *.export.data *.linux 等文件整理到一个目录下。

2、编译出可执行文件

将第一步中生成的 case code 编译出可执行文件。

3、设置环境变量

参照 4.3 中重新加载 galcore.ko。设置环境变量

Linux：

获取 OP 时间：

```
export VIV_VX_DEBUG_LEVEL=1
```

```
export CNN_PERF=1
```

dump 每层 layer 结果：

```
export NN_LAYER_DUMP=1
```

Android：

获取 OP 时间：

```
setprop VIV_VX_DEBUG_LEVEL 1
```

```
setprop CNN_PERF 1
```

dump 每层 layer 结果：

```
setprop NN_LAYER_DUMP 1
```

Note：

1、linux 环境：a.将转换工具里 acuity-toolkit-xxx/bin/vcmdtoos push 到板子 data 目录。b.设置环境变量：export VIVANTE_SDK_DIR=/data/vcmdtoos。c.将

sdkv0.1 里 buildroot_sdk\build\so\drivers_64_exportdata 或 drivers_32_exportdata
push 到板子 data 目录。d.设置环境变量: export
LD_LIBRARY_PATH=/xx/drivers_64_exportdata

2、如果模型较大，需谨慎设置获取每层结果的环境变量。

4、运行模型

执行命令: ./xxxx xxx.export.data xxx.jpeg

运行后得到每层 op 执行时间, log 如下图

```
layer id: 43 layer name:TensorMul operation[0]:unknown operation type target:VXNNE_OPERATION_TARGET_SH.
uid: -1
execution time:          125 us
VPC ELAPSETIME: 13317
*****
layer id: 80 layer name:ConvolutionReluPoolingLayer2 operation[0]:VXNNE_OPERATOR_CONVOLUTION target:VXNNE_OPERATION_TARGET_NN.
uid: 6
execution time:          72 us
VPC ELAPSETIME: 13431
*****
layer id: 75 layer name:ConvolutionReluPoolingLayer2 operation[0]:VXNNE_OPERATOR_CONVOLUTION target:VXNNE_OPERATION_TARGET_NN.
uid: 5
execution time:          86 us
VPC ELAPSETIME: 13559
*****
layer id: 44 layer name:ActivationLayer operation[0]:VXNNE_OPERATOR_ACTIVATION target:VXNNE_OPERATION_TARGET_TP.
uid: 3
execution time:          87 us
VPC ELAPSETIME: 13688
*****
layer id: 45 layer name:PoolingLayer2 operation[0]:VXNNE_OPERATOR_POOLING target:VXNNE_OPERATION_TARGET_SH.
uid: 2
execution time:          133 us
VPC ELAPSETIME: 13864
*****
layer id: 76 layer name:FullyConnectedReluLayer operation[0]:VXNNE_OPERATOR_FULLYCONNECTED target:VXNNE_OPERATION_TARGET_TP.
uid: 1
```

4.7 在板子上运行模型时，输入输出数据的数据格式是怎样的？

解答：各个框架模型，NPU 推理对应的输入输出数据格式为：

Framework	Input Format	Output Format
Caffe	NCHW(bbbgggrrr)	NCHW
Pytorch	NCHW(rrrgggbbb)	NCHW
Darknet	NCHW(rrrgggbbb)	NCHW
Onnx	NCHW(bbbgggrrr)	NCHW
Tensorflow	NHWC(rgbbrgb)	NHWC
Tflite	NHWC(rgbbrgb)	NHWC
Keras	NHWC(rgbbrgb)	NHWC

NPU 在推理计算的时候，只支持 NCHW 格式，TensorFlow/Tflite/Keras 三种框架对应模型在 NPU 推理时，数据输入之后有个 Permute 操作将数据格式 NHWC -> NCHW，输出之前也有个 Permute 操作将数据格式 NCHW->NHWC。