Amlogic

NN Tool 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

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 0.5 (2021-08-10)

This is the 0.5 version.

Compared to the previous version, the following contents are changed.

- Modified 1.1, 3.4 and 4.6.

Issue 0.4 (2021-01-15)

This is the 0.4 version.

Compared to the previous version, the following contents are changed.

- Modified 1.3, 3.2, 4.3 and 3.4.
- Added 3.8, 3.9 and 4.7.

Issue 0.3 (2020-06-16)

This is the 0.3 version.

Compared to the previous version, the following contents are changed.

- Modified 4.1, 4.2 and 4.3.
- Added 1.6, and 4.6.

Issue 0.2 (2020-05-12)

This is the 0.2 version.

Compared to the previous version, the following contents are changed.

- Modified 1.4, 4.2 and 4.3.
- Added 1.5 and 4.5.

Issue 0.1 (2020-04-17)

This is the initial release.

Contents

1. Framework

1.1 What open source frameworks does Amlogic NN currently support？

Answer: The currently supported neural network framework and the corresponding version information are as follows:

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 What formats does Amlogic NN currently support?

Answer: Amlogic NN only supports quantized models. The models corresponding to the supported open source framework need to be quantized. There are three quantization methods: Int8, int16, and Uint8.

1.3 What models does Amlogic NN currently support?

Answer: SqueezeNet, VGG, Resnet*, Inception, RetinaNet*, MobileNet*, LSTM*, GRU*, SSD, Yolo*, and FasterRCNN (MaskRCNN not support).

For the list of supported models, please refer to: https://github.com/VeriSilicon/acuity-models/tree/master/models

1.4 Can I use Amlogic NN without the acuity quantization framework?

Answer: For customers with this need, the current method is also supported. At present, it has been adapted to support armnn. You can refer to the amrnn process.

1.5 Can NPU run several NNs in parallel?

Answer: Support multi-threaded multi-process operation, but the hardware is time-multiplexed

1.6 What are the functions, pros and cons of NN core, PPU and TP?

Answer:

- **NN core**: the main module of NN operation, contains massive macs units, and mainly performs large-scale operations such as convolution.
- **TP**: tensor processing unit, which is a supplement to NN core and operates some processing of tensor.
- **PPU**: a module like GPU, programmable, supports float, and is the main computing platform for customized operators.

Unit	Pros	Cons
NN core	Fast operation speed.	Only limited operators are supported.
TP	More operators are supported than NN core	Slow operation speed.
PPU	Programmable, theoretically all operators are supported, and high accuracy.	Slow operation speed.

Note:

Please refer to *Layer and Operation Support Guide* for information on which operator is handled by which unit

2. Tools

2.1 Can the current model caffe framework be installed without tensorflow?

Answer: TensorFlow must be installed, and the current quantization and inference are using tensorflow API.

2.2 Is the quantization method of Acuity tool the same as tensorflow?

Answer: It is the same. Asymmetric_quantized-u8 is a quantization method supported by Google. According to the description of the paper "Quantizing deep convolutional networks for efficient inference: A whitepaper", for most networks, this quantization method has the smallest loss of accuracy.

3. Model Conversion

3.1 Will there be a loss of accuracy in model conversion?

Answer: Model conversion involves quantization of float data into int8 / uint8 / int16 data, and there will be a certain degree of accuracy loss.

3.2 When quantifying, how many pictures does datet need? Is quantity and precision proportional?

Answer: It is generally recommended that the number of pictures is about 50 ~ 200, try to be pictures of the scene when the model is running, or pictures in the validation set of the training model. This has a positive effect on the quantification effect. But the quantity and precision are not proportional.

Note: The current quantization default --batch-size is 100, and the default -epochs is 1, so by default only 100 pictures will be used at most. When the number of pictures is >100, it is recommended to add quantization parameters --batch-size (default 100) and --epochs (default 1), epochs*batch_size=number of pictures. For example, 5000 pictures, the setting parameters are: --batch-size 100 --epochs 50. When the computer memory is small, the batch_size value can be set to a smaller value.

3.3 How to deal with unsupported models and how to implement unsupported layers?

Answer: 1. For the unsupported open source framework model, you need to convert it to the corresponding model of the open source framework supported by the current Amlogic NN.

2. For the unsupported layer, the current tool is released in binary mode, and customers are currently not supported to add custom layers.

3.4 Using the acuity tool to convert the model, when exporting the case code, how to set the correct parameters of the corresponding board?

Answer: execute command: cat / proc / cpuinfo

Check the first four characters of the serial code corresponding to the second to last serial number, e.g. 290a, 290b, and determine the set value according to the following correspondence.

PID	Serial
0X7D	Serial : **290a**70004ba55adb38423231474c4d4
0X88	Serial : **290b**70004ba55adb38423231474c4d4
0X99	Serial : **2b0a**0f00df472d383156314d534c4d41
0XA1	Serial : **300a**010245bbf1e131305631434c4d4
	Serial : **300b**010200151d0000013936583853
0X99	Serial : **2f0a**0c00d2f1ef293156324d544c4d41
0XB9	Serial : **2f0b**06009f45301d3356324d544c4d41
0XBE	Serial : **330a**0304d93895a932305632434c4d4
	Serial : **330b**0304d93895a932305632434c4d4
0XE8	Serial : **380a**0201000000008d94ad5d3256335
	Serial : **380b**0201000000008d94ad5d3256335

- When the first four digits of the serial number are 290a, the parameter is set to "--optimize VIPNANOQI_PID0X7D".
- When the first four digits of the serial number are 290b, the parameter is set to "--optimize VIPNANOQI_PID 0X88".
- When the first four digits of the serial number are 2b0a, the parameter is set to "--optimize VIPNANOQI_PID 0X99".
- When the first four digits of the serial number are 300a, the parameter is set to "--optimize VIPNANOQI_PID 0XA1".
- When the first four digits of the serial number are 2f0b, the parameter is set to "--optimize VIPNANOQI_PID 0XB9".
- When the first four digits of the serial number are 330a, the parameter is set to "--optimize VIPNANOQI_PID 0XBE".
- When the first four digits of the serial number are 380a, the parameter is set to "--optimize VIPNANOQI_PID 0XE8".
- If the first four digits of the serial number are not counted in the above table, it is recommended to use the corresponding parameters of the first three digits of the serial number to try.

3.5 After using point-16 quantization, when using point-8 again, the coefficient file cannot be reduced?

Answer: When the file generated by one-step quantization is not deleted, quantization is performed again, and no new file is generated. It is recommended to add the default parameter --quantized-rebuild to the quantization command. At this time, the intermediate file will be deleted by default first, and then be quantized. You can also manually delete the quantized intermediate files.

3.6 Errors reported during conversion of caffe model: Not supported caffenet model version (v0 layer or v1 layer)

Answer: The current caffemodel or prototxt file version is too old, you need to use the tool to update the prototxt or caffe model file.

Solution: Update the prototxt tool: upgrade_net_proto_text-d. Update caffemodel tool: upgrade_net_proto_binary-d.

Obtaining method: Compile the caffe source code, and the corresponding executable file will be generated under the tool directory.

3.7 If the model input contains a sequence, how should the shape be defined?

Answer: Pass the sequence value with --predef-file xxx.npz, and add this parameter value when importing the model. To support some logic controls, a predefined file in npz format needs to be provided, which is generated by np.savez (\ 'prd.npz \', placeholder_name = predefine_value). If there are arrays in the placeholders that do not support characters, you can use map as a key to store the dictionary as a map to a normal key value. For example: "inference / train_satge" is a placeholder, numpy is not supported, you can use: np.savez ('prd.npz', stage = False, map = {'stage': 'inference / train_stage'}.

3.8 Is the quantation input data only pictures?How to give quantation data to some models whose input data is not a picturre?

Answer:The quantation input data may not be pictures.Currently you can save array or txt format file data as npy files.And then when quantizing ,use the path of the npy file to replace the image path in dataset.txt.Such as :

1、use or refer to the acuity_tools_xxx/bin/tools/npy_input_generate.py script to save npy file.

   python3 npy_input_generate.py   --action tensor2npy   --src-platform tensorflow --input-file xxx.tensor   --shape "224 224 3" --numpy-data-file   input.npy

2、When quantizing , the source is still set to txt, and the path of npy is listed in the dataset.txt corresponding to source file

3.9 Common issue of Caffe model

Answer:

   a.The format of the input layer of the caffe model:

b.If the prototxt does not support the layer, it is easy to cause the conversion to fail. For example, the detection model does not support the DetectionOutput layer, and the layer needs to be deleted manually.

```
layer {
  name: "detection_out"
  type: "DetectionOutput"
  bottom: "mbox_loc"
  bottom: "mbox_conf_flatten"
  bottom: "mbox_priorbox"
  top: "detection_out"
  include {
    phase: TEST
  }
  detection_output_param {
    num_classes: 21
    share_location: true
    background_label_id: 0
    nms_param {
      nms_threshold: 0.45
      top_k: 100
    }
```

4. Integration

4.1 An error occurred during the running of the model, how to confirm the problem?

Answer:

1. Confirm whether there is NPU environment on the board.

   Confirm NN model loading: galcore in lsmod results.

   Confirm that there is a NN device node: ls -l / dev / galcore permission is 664.

2. Confirm whether the currently transferred nbg code is consistent with the board.

   Refer to the question of 3.1.4.

3. Check the specific error points.

   Set the environment variables, and then execute the program again.

   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. Problem analysis.

   1) If the first and third steps are wrong, directly feedback to Amlogic engineer.

   2) If there is an error in the second step, re-convert the nbg file and run it again.

4.2 The accuracy of the running results on the board is poor, how to locate?

Answer:

1. Confirm that the precision of the float type is similar to the original platform test results.

   After executing the imported model, use the inference.sh script in the demo provided in the tool to run the result of the float type and compare this with the original platform result.

2. Confirm whether the precision of quantization is similar to the precision of float type.

   After performing the quantization step, use the inferece.sh script to compare the quantized result with the float type result.You can use acuity_tool_xxx/bin/tools/compute_tensor_similarity.py to compare, and output the cosine similarity and Euclidean distance of the two files.

3. Confirm whether the results on the board side are similar to the quantitative results.

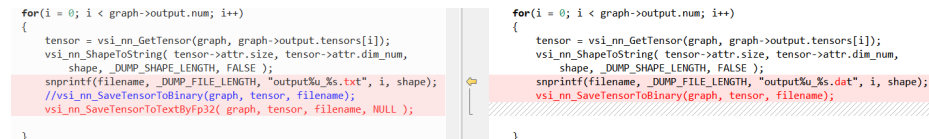1) Modify the **save_output_data** function in **vnn_post_process.c** in the converted case code as follows:

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

vsi_nn_SaveTensorToBinary(graph, tensor, filename);

Change to:

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

vsi_nn_SaveTensorToTextByFp32(graph, tensor, filename, NULL);

Such as :

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

}                                                         }
```

2) Compile the executable file and run the result on the board side.

Problem analysis:

1. If there is a problem in the first step, you need to check whether the input and output nodes when importing the model are consistent with the original platform.

2. If there is a problem in the second step, it means that the quantization is wrong or the quantization effect is not ideal. At this time, you can check whether the current quantization parameter is wrong or feedback it to the Amlogic engineer.

3. If there is a problem in the third step, it means that the results on the board side are inconsistent with the quantized results. The problem appears in the nbg file packaging process or the driver. You can feedback it to the Amlogic engineer.

Note:

Sometimes there is the following situation: there is a slight difference between the result of the board side and the quantized result of the inference on the PC side, and the corresponding post-processing code needs to be added.

If final result is the simulation tool is correct / the board side is wrong, it can be considered as the board side There is a big difference from the simulation results. At this time, you can feedback the problem to Amlogic.

4.3 How to count the bandwidth and NPU utilization rate during the running of the model?

Answer: Reload npu ko before running nbg case, and set some environment variables to print out the corresponding information.

1. Set environment variables

Linux：

a. Uninstall the npu module: rmmod galcore

b. Load npu module: insmod /xxx/xxx/galcore.ko gpuProfiler=1 showArgs=1

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

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

Note:

If there is no galcore.ko in the above path, execute ***find -name galcore.ko*** under /lib to search the path of ko.

c. Add environment variables:

export VIV_VX_PROFILE=1

export VIV_VX_DEBUG_LEVEL=1

Android：

a. Uninstall npu directory: pkill network && rmmod galcore

b. Reload the npu module: insmod /xxx/xxx/galcore.ko gpuProfiler=1 showArgs=1

galcore.ko directory: /vendor/lib/modules/galcore.ko

d. Add environment variables:

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.In the current Android version,if NPU DDK version <6.4.3, some functions of bandwidth information may be printed invalid. This problem has not been solved yet. If necessary, Amlogic engineers will provide a newly compiled libGAL.so.

2. Confirm the DDK version: execute dmesg |grep Galcore in the serial port or adb window to check the version number.

2.   Run the model and count the bandwidth based on the printed log.

The following prints will be made during operation:

[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 is the bandwidth saved by SW TILING, the actual bandwidth is equal to: READ_BANDWIDTH + WRITE_BANDWIDTH -   READOCB_BANDWIDTH - WRITEOCB_BANDWIDTH

Usage rate:（TOTALCYCLES-IDLECYCLES）/TOTALCYCLES

4.4 After the demo is integrated, it is found that the pre-processing time is longer, which is the calculation bottleneck in the entire process. How to solve it?

Answer: The pre-processing of the model is generally after the input data minus the mean multiplied by the magnification, the obtained float number is converted to an INT16 or Int8 number, and then passed to the NPU for processing. In the currently converted case code, the interface used by float to Int16 and Int8 is vsi_nn_Float32ToDtype. It is recommended to replace this interface, as follows:

Int8/int16：(fl can be obtained according to the input and output tensor: 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 can be obtained according to the input and output tensor:

```
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：Some output tensor is in float16 format, you need to inverse quantize float16 into 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 Is it safe to use data received from vsi_nn_ConvertTensorToData if I call vsi_nn_RunGraph after I saved pointer from vsi_nn_ConvertTensorToData?

a. (Code snippet):

```
vsi_nn_RunGraph(graph);

auto bytes = vsi_nn_ConvertTensorToData(graph->output.tensor[0]);

vsi_nn_RunGraph(graph);

process(bytes)
```

Answer: is safe. the byte is a buffer create by malloc. you need vsi_nn_Free(bytes) after post_process.

4.6 How to get the result and usage time of each operation?

Answer: Run a normal case (non-nbg) on the board to start the online compilation process, and the loading process is time-consuming.

1. Generate normal case.

    After exporting the case code in the last step of the model conversion process, delete the following three parameters:

```
--optimize VIPNANOQI_PID0X88  \

--viv-sdk ../bin/vcmdtools \

--pack-nbg-unify
```

Case code will be generated in the directory where the model is located. Please organize *.c *.h *.export.data *.linux and other files into one folder.

2. Compile the case code generated in the first step into an executable file.

3. Refer to 4.3 to reload galcore.ko, and set environment variables.

    ■   Linux：

    Get OP usage time:

    export VIV_VX_DEBUG_LEVEL=1

    export CNN_PERF=1

    Dump result of each layer:

    export NN_LAYER_DUMP=1

    ■   Android：

    Get OP usage time:

    setprop VIV_VX_DEBUG_LEVEL 1

    setprop CNN_PERF 1

    Dump result of each layer:

    setprop NN_LAYER_DUMP 1

    Note:

    •   If it is in Linux environment:

    1. Push acuity-toolkit-xxx/bin/vcmdtoos in the conversion tool to the data directory on the board.

    2. Set the environment variable: export VIVANTE_SDK_DIR=/data/vcmdtoos

    3. Push buildroot_sdk\build\so\drivers_64_exportdata or drivers_32_exportdata in the sdkv0.1 to the data direcotry on the board.

    4.Set environment variable: export LD_LIBRARY_PATH=/xx/drivers_64_exportdata

    •   If the model is large, you need to carefully set the environment variables to obtain the results of each layer.

4. Run command **./xxxx   xxx.export.data   xxx.jpeg** to get operation usage time of every layer, the log is shown below:

```
layer id: 43 layer name:TensorMul operation[0]:unkown 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 What is the data format of input and output data when running the model on board?

Answer: for each framework model, the input and output data format corresponding to NPU reasoning is:

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