模型转换运行用户指南

Revision: 1.0

Release Date: 2022-02-10

Contact Information

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

变更说明

版本 1.0（2022-02-10）

第 10 版。

1. 移除前面版本中一组的转换模型命令
2. 添加统一转换命令 Pegasus，添加 pegasus 对应的模型转换流程

版本 0.9（2021-08-10）

第 9 版。

1. 添加对新 IP 的量化指导
2. 添加 perchannel 量化对应指导以及限制条件

版本 0.8（2021-01-15）

第 8 版。

1. 添加环境安装注意事项
2. 添加量化时使用多张图片的指导

版本 0.7（2020-7-18）

第 7 版。

1. 增加多输入模型转换指导
2. 删除 IDE 工具使用指导

版本 0.6（2020-05-18）

第 6 版。

1.增加了 3.5 章节。

版本 0.5（2019-12-20）

第五版。

1. 增加使用 acuity 工具进行 inference 操作指导
2. 增加模型转换过程中量化参数细节指导
3. 移除 FAQ 章节，将对应章节内容移到 Amlogic-NN-FAQ 文档中
4. 增加 darknet、Keras 框架模型导入指导

版本 0.4 (2019-07-18)

第四版。

1. 优化环境安装步骤，提供一键式安装方式
2. 模型转换增加模型量化简介以及量化参数选取指导
3. 删除工程代码编译章节，case 代码编译在<Android&Linux 开发指导.docx>文档中体现
4. 删除通用知识章节
5. FAQ 章节中增加量化/反量化指导，以指导客户解决反馈的前处理时间过长的问题

版本 0.3 (2019-07-08)

第三版。

1. 添加 ONNX 模型转换命令
2. 模型转换工具添加 whl 包安装提醒说明
3. DDK_6.3.3.4 离线模型变更为 nbg 类型，修改对应模块说明
4. DDK_6.3.3.4 模型转换出 case 代码后直接生成 case 代码目录，取消代码整理章节。
5. 增加 case 代码简介章节

版本 0.2 (2018-12-20)

第二版。

1. 默认环境安装错误点/遗漏点修改
2. 不再使用 IDE 转换 case 代码，删除对应章节，部分章节介绍进行调整。
3. 将代码转换流程精简，部分流程删除，将问题解答移到第七章节。
4. 增加 IDE 工具使用介绍章节，介绍 case 代码导入，添加 jpeg 库，编译运行等流程。
5. 工程代码章节添加外部人员获取版本方式，增加基于内部代码和外部代码进行 case 代码编译介绍。
6. 添加 FAQ 章节，将常见问题归纳输出。

版本 0.1 (2018-11-12)

第一版。

目录

1. 简介

本文将仔细介绍使用 Acuity_tool 工具将模型转换出 case 代码的过程以及细节。指导公司内部或者外部相关开发人员进行模型转换。

当前我们的 NN 芯片只支持 tensorflow、caffe、tflite、darknet、onnx、pytorch, keras 类型的模型。如果开发者是其他类型的模型，需先将模型转换成上述支持的一种。

2. 工具介绍

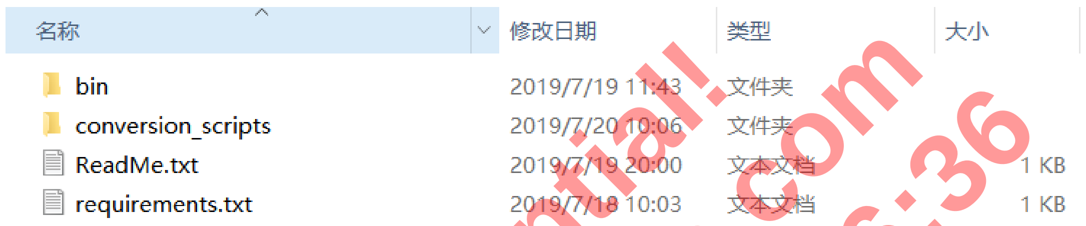
2.1 简介

当前将模型转换出 case 代码需要的工具：

1. acuity-toolkit 模型工具，用于将模型量化，并转换出 case 代码，编译后即可运行。

2.2 模型转换工具

2.2.1 工具目录简介



名称	修改日期	类型	大小
bin	2019/7/19 11:43	文件夹	
conversion_scripts	2019/7/20 10:06	文件夹	
ReadMe.txt	2019/7/19 20:00	文本文档	1 KB
requirements.txt	2019/7/18 10:03	文本文档	1 KB

bin：模型转换需要用到的可执行文件以及 config 配置文件目录

conversion_scripts：

1、模型转换 demo（包含转换脚本程序和 mobilenet_v1.pb 模型）

2、转换只需要在环境安装后之后按照顺序执行三个 sh 脚本即可。

3、转自己的模型，修改三个脚本对应的配置参数再进行转换。

requirements.txt：环境依赖的安装包列表。

ReadMe.txt：简单的文档简介

2.2.2 环境依赖

操作系统	Ubuntu16.04（x64）
Python 版本	Python3.5.2

依赖库	tensorflow==1.13.2 numpy==1.16.4 scipy==1.1.0 Pillow==5.3.0 protobuf==3.9.0 networkx==1.11 image==1.5.5 lmdb==0.93 onnx==1.4.1 h5py==2.9.0 flatbuffers==1.10 matplotlib==2.1.0 dill==0.2.8.2 ruamel.yaml==0.15.81 onnx_tf==1.2.1 ply==3.11 torch==1.2.0

注：上述只是将当前需要的一些库列出来，具体的依赖库以工具内部 acuity_toolkit-binary-xxx 里面的 requirements.txt 为准。

2.2.3 工具安装

Step 1. 环境准备

Ubuntu 16.04 系统 64 位的计算机（如果是虚拟机，RAM 内存设置>4GB），Python 需要的是 3.5.2 版本。

Step 2. 安装 python3 以及 pip 等工具

```
sudo apt-get install python3 python3-pip python3-virtualenv
```

Step 3. 安装相关依赖包

获取 acuity-toolkit 工具包，进入根目录。
执行：for req in $(cat requirements.txt); do pip3 install $req; done

Step 4. Check 环境

```
python3 bin/checkenv.py
```

注：check 成功会有：Env Pass: Env check SUCCESS!!!打印。

3. 模型转换

3.1 模型转换简介

模型转换过程是先将模型量化成 int8、int16 或者 uint8 数据格式，再转成基于我们平台运行的 nbg 格式文件，并导出运行的 case 代码。

3.2 量化简介

量化：量化是将以往用 32bit 或者 64bit 表达的浮点数用 16bit、8bit 或者更低的 2bit 方式进行存储。

3.2.1 量化详解

Int8：将 float32 数据量化成 int8 的数据，8 个 bit 位，一个为符号位。其他 7 个 bit 位来表示有效数字。需算出 fl 值。fl：表示小数的 bit 位个数。（int16 与 int8 类似）

可以参考 3.4 章节 step2 里面生成的 xxxx.quantize，分析如下图：

```
@InceptionResnetV1/Block8/concat_14:out0':
   dtype: dynamic_fixed_point
   method: layer
   max_value:
   -    6.883890151977539
   min_value:
   -    0.0
   fl:
   -    4
   qtype: i8
```

解析：qtype 表示量化方式为 int8。最小/最大值范围为（0.0~6.88）

输出结果的最大绝对值为 6.88，只需要 3 个 bit 就可以表示整数位。

所以用来表示小数的位数 fl=8-1-3=4。

u8：将 float32 数据量化成 unsigned int8 的数据。8 个 bit 位均用来表示数字，没有符号位。即：将最大/最小范围区间映射到 0~255 之间。需计算出两个参数，scale:映射比例 zero_point：零点的位置。计算公式如下图：

scale = (max_value - min_value)/255

zero_point = max_value - max_value/scale

参考 3.4 章节 step2 里面生成的 xxxx.quantize。

```
'@FeatureExtractor/MobilenetV1/MobilenetV1/Conv2d_0_1:weight':
    dtype: asymmetric_quantized
    method: layer
    max_value:
    -   2.7051823139190674
    min_value:
    -   -2.880463123321533
    zero_point:
    -   132
    scale:
    -   0.021990729495882988
    qtype: u8
```

解析：qtype 表示量化方式为 u8。输出的最小/最大值范围为（-2.88~2.7）

量化参数计算：

scale = (2.705 -（-2.88）)/255= 0.0219

zero_point = 255 - 2.705/scale = 132

3.2.2 量化方式选取指导

量化方式选取规则：

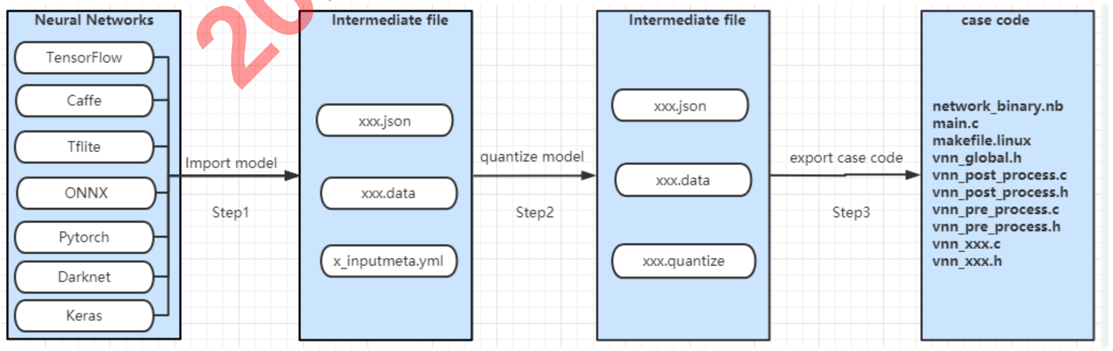
1. 通常情况下，选择 u8/int8 的方式均可。u8 是 google 推荐量化方式。

2. 一般建议不选取 int16，相较 8bit 量化，模型大一倍，并在精度上没有明显的优势。

3. 参考模型前处理后的取值范围选取量化方式。比如，如果模型前处理后的取值范围为（0~255），此时建议直接选取 u8 的量化方式。

4. 先随便选择一种量化方式，执行完量化操作后，查看生成的 xxxx.quantize 文件，确认统计的 max_value/min_value,如果出现绝对值>256 的情况，建议选取 int16 方式量化。如果绝对值出现>128 的情况，建议选取 u8 量化方式，其他情况选择 int8/u8 均可。

3.3 模型转换流程图



3.4 模型转换步骤

当前模型转换过程都是在 acuity-toolkit 目录下进行的。可执行文件均在 bin 目录下。

Step 1. 导入模型并生成量化配置文件

命令:

```
Step A：
        Caffe：
                ./bin/pegasus    import caffe    --model xxxx.prototxt --weights xxx.caffemodel \
                 --output-data    xxx.data    --output-model xxx.json

        Darknet：
                ./bin/pegasus    import    darknet      --model xxx.cfg --weights xxx.weights \
                                --output-data    xxx.data    --output-model xxx.json

        Tensorflow：
                ./bin/pegasus import      tensorflow    --model xxx.pb    \
                 --inputs input    --outputs InceptionV1/Logits/Predictions/Reshape_1 \
                 --input-size-list '224,224,3'    \
                 --output-data xxx.data --output-model xxx.json

        Tflite:
                ./bin/pegasus import    tflite    --model    xxxx.tflite    \
                 --output-data xxx.data --output-model xxx.json

        Onnx：
                ./bin/pegasus import    onnx --model    xxxx.onnx    \
                 --output-data xxx.data --output-model xxx.json    \

                 # --inputs "0 1 2"    --input-size-list "288,288,3#288,288,3#288,288,3" \

                 # --outputs "327 328"

        Keras:
                ./bin/pegasus import keras --model    xxxx..hdf5 \
                   --output-data xxx.data --output-model xxx.json

        Pytorch:
                ./bin/pegasus import pytorch --model    xxxx...pt    \
                 --output-data xxx.data --output-model xxx.json    --input-size-list '3,224,224'

Step B：
        生成配置文件：
                ./bin/pegasus    generate inputmeta --model ${NAME}.json \
                  --channel-mean-value "128 128 128 0.0078125"    --source-file dataset.txt \
                --input-meta-output ${NAME}_inputmeta.yml    \
                #    --separated-database
```

结果：1. 生成三个中间文件 xxx.json、xxx.data 和 xxx_inputmeta.yml

注：

1. 转换 tensorflow 模型时，需要知道模型的输入输出节点名称以及输入的 HWC 尺寸。tensorflow 模型可以使用 summarize_graph 或者 tensorboard 来找对应节点以及尺寸

2. --inputs/--outputs，输入输出节点名称。如果有多个输入或者输出，请使用空格分隔每个输出，例如“output1 output2 output3”

3. --input-size-list，如果有多个输入，用#分隔每个输入大小，例如：“224,224,3#299,299,3”

4. 当前只支持 Caffe/Tensorflow/Tflite/Darknet/Onnx/pytorch/keras 等模型，如果是其它类型模型，需要先将模型转成当前支持的类型

5. tensorflow 模型如果存在一些逻辑控制的输入，请参照 Amlogic-NN-FAQ 文档中 3.17 指导进行处理。

7. --channel-mean-value 根据训练模型时预处理方式来设置该预处理的命令行参数。包括四个值（m1, m2, m3, scale）. 前三个值为均值参数，最后一个值为 scale 参数。对于输入为三通道数据（data1，data2，data3），预处理过程为：

Out1 = (data1-m1)*scale

Out2 = (data2-m2)*scale

Out3 = (data3-m3)*scale

例如：如果前处理需将数据归一化到[-1, 1]之间，参数设置为（128 128 128 0.0078125）

如果前处理需将参数归一化到[0, 1]之间，参数设置为（0 0 0 0.00392156）。

如果是单通道参数，设置方式为（m1, 0, 0, scale）

最后一个数 scal 为小数，比如 0.0078125(1/128)、 0.00392156 (1/256)

7. dataset.txt 中是给定的量化输入图片路径，如：/data/test/cat.jpg。建议提供 200 张左右，同时也是模型使用运行场景的图片来进行量化，确保统计出的最大/最小值为实际运行场景的最大/最小值范围，这样量化效果会更佳。

8. 多输入模型转换注意点

1. *--separated-database* 模型为多输入模型时，添加次参数生成的 yml 文件中单独给每个 input 设置 dataset.txt。如果没设置此参数，所以的输入共一个 dataset.txt，此时 dataset.txt 中应将多个输入放一行并以空号间隔开，如一组输入： ./1.jpg ./2.jpg ./3.jpg 。 如果要设置多张图片量化，可以设置多组图片路径即可。

2. --channel-mean-value 设置输入对应的 channel-mean value，也可以直接修改生成的 xxx_inputmeta.yml 文件，多个输入对于的 channel-mean 不同时，以#号隔开，例如一组输入对于的 channel-mean: "128 128 128 0.00715#106 115 67 0.00715#121 99 112 0.00715"

3. *--source-file* 设置输入的 source，也可以直接修改生成的 xxx_inputmeta.yml 文件。如果设置了*--separated-database，此时输入对应的设置为："dataset1.txt#dataset2.txt#dataset3.txt"*

Step 2. 对模型进行量化

命令:

```
./bin/pegasus   quantize   --quantizer asymmetric_affine\
                --qtype uint8   --with-input-meta   ${NAME}_inputmeta.yml \
                --model   ${NAME}.json --model-data   ${NAME}.data \
                --rebuild   \
                # --batch-size   16   --iterations 10
```

结果：根据 dataset.txt 提供的输入图片，进行前向推理计算，统计每层 layer 输出的最大/最小值以及模型权值的最大/最小值，同时计算出每层的量化参数，保存成量化结果文件 xxx.quantize，导出 case 代码时会用到该量化文件。

注：

1. *--quantizer 和--qtype* 一起用于选择量化类型，支持的类型为：

asymmetric_affine ---> 对应 *qtype 选择设置* uint8

dynamic_fixed_point --->对应 qytpe 选择设置 int8 或者 int16

perchannel_symmetric_affine （perchannel 量化，对应 qytpe 设置 int8，当前只有 E8 对应芯片支持，参照 Step3 生成 case 代码中--optimize 参数的设置）
dynamic_fixed_point-i8

2. --rebuild 默认参数，建议携带。在前后使用不同的量化方式进行量化时，如果没有手动删除已生成的 xxx.quantize 文件，在没有携带该参数时，第二次量化未做任何处理。

3. *--batch-size 和--iterations 为多张图片量化时设置参数，*如果图片数量为>1，需添加量化参数*--batch-size*（默认 1）和*--iterations*（默认 1），*iterations* *batch_size=图片数量。例如 5000 张图片，设置参数为：--batch-size 100 --*iterations* 50。

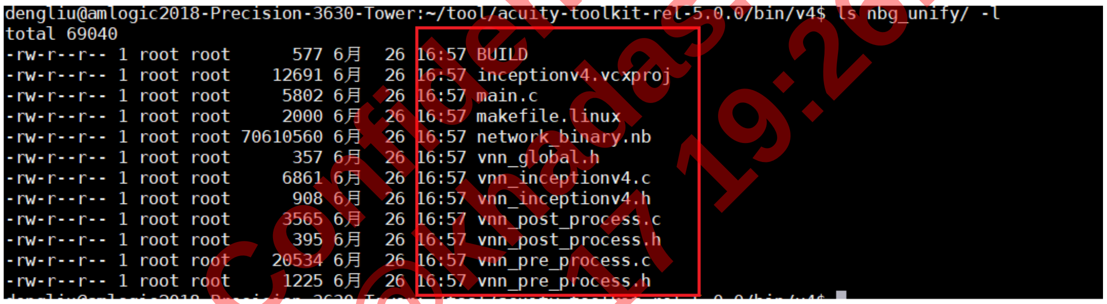
如果电脑内存较小，可以将 batch_size 的值设置成较小的值。

Step 3. 生成 case 代码

命令:

```
./bin/pegasus    export ovxlib   \
        --model xxx.json --model-data xxx.data \
        -model-quantize xxxx.quantize    --with-input-meta    xxxx_inputmeta.yml \\
         --dtype quantized    \
        --optimize VIPNANOQI_PIDOX88    \
        --viv-sdk    ./bin/vcmdtools \
        --pack-nbg-unify
```

结果：生成 case 代码，如下图：



```
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
```

注：

1. --optimize 当前设置值为 VIPNANOQI_PIDOX88。当前有八个有效值：

VIPNANOQI_PIDOX7D、VIPNANOQI_PIDOX88

VIPNANOQI_PIDOX99、VIPNANOQI_PIDOXA1

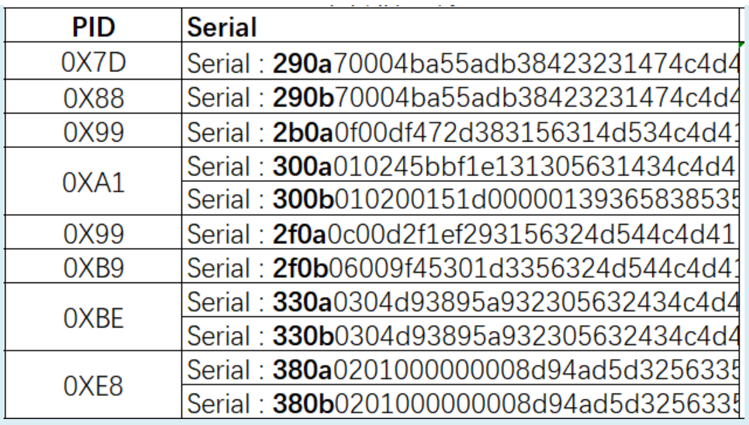
VIPNANOQI_PIDOXB9、VIPNANOQI_PIDOXBE

VIPNANOQI_PIDOXE8、VIPNANOQI_PIDOX1E

可以使用如下方式来确认应该设置的值，执行：cat /proc/cpuinfo

查看倒第二行数 Serial 对应的数列码**前四个字符（Serial 后面粗体字部分）**

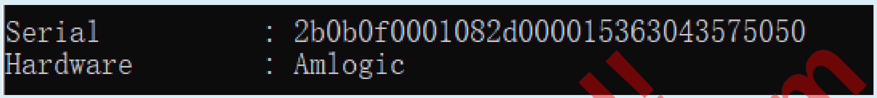
序列号和参数 VIPNANOQI_PIDOX？？ 的对应关系如下图：



PID	Serial
0X7D	Serial : **290a**70004ba55adb38423231474c4d4
0X88	Serial : **290b**70004ba55adb38423231474c4d4
0X99	Serial : **2b0a**0f00df472d383156314d534c4d41
0XA1	Serial : **300a**010245bbf1e131305631434c4d4
	Serial : **300b**010200151d000001393658385353
0X99	Serial : **2f0a**0c00d2f1ef293156324d544c4d41
0XB9	Serial : **2f0b**06009f45301d3356324d544c4d41
0XBE	Serial : **330a**0304d93895a932305632434c4d4
	Serial : **330b**0304d93895a932305632434c4d4
0XE8	Serial : **380a**0201000000008d94ad5d3256335
	Serial : **380b**0201000000008d94ad5d3256335

例如:

在串口或者 cmd 窗口执行命令: cat /proc/cpuinfo 后，Serial 如下图



```
Serial          : 2b0b0f0001082d000015363043575050
Hardware        : Amlogic
```

此时，参数应该设置为: --optimize VIPNANOQI_PID0XB9

2.--viv-sdk 依赖的 sdk 包，acuity_tool_xxx/bin 目录下 vcmdtools 目录，填写相对路径即可。

3.--pack-nbg-unify 生成 nbg 文件。

4、1/2/3 三个对应参数设置后，会生成 nbg 的 casedemo，我们通常使用的就 nbg case demo。

此时在模型目录下也会生成 normal case demo，执行下面两行命令，可以将 normal case 整理到一个目录中：

```
  mkdir normal_case_demo

  mv  *.h *.c .project .cproject *.vcxproj *.lib BUILD *.linux *.export.data
normal_case_demo
```

二者区别：

Normal_case: 加载模型时，会有在线编译时间，耗时较长。

android 平台：支持直接跑 normal case

linux 平台: 不支持直接跑 normal case。1.Linux 平台需将 acuity_tool_xx/bin/vcmdtools 目录 push 到板子 data 目录，然后设置环境变量：export VIVANTE_SDK_DIR=/data/vcmdtoos。2.还需要将 buildroot_sdk_64xx 目录里 buildroot_sdk/build/so/drivers_64_exportdata/对应 so push 到板子上，设置环境变量：export LD_LIBRARY_PATH=/xxx/xxx/drivers_64_exportdata

NBG case：在线编译这一步在 pc 端已经完成，板子上能直接加载 nb 文件，模型加载速度快。

5、nbg case 转换成功标志：case 代码目录下有 network_binary.nb 文件。

6、demo 是模型转换目录，会在它同级目录下生成 demo_nbg_unify 目录，此为 nbg case demo 目录

生成的 case 代码简介

生成的代码在 nbg_unify 目录下。

1. network_binary.nb 是生成的 nbg 文件，保存有模型的权值和 graph，该文件名称可以根据自己喜好修改。
2. vnn_inceptionv4.c 模型创建和释放实现对应的代码文件
3. vnn_pre_process.c 模型前处理文件，将读入图片数据量化成 8bit 数据，可以复用文件中接口，也可以自行实现量化。
4. vnn_post_process.c 模型后处理文件，当前转出的 case 代码后处理只是做了 top5 处理，可以根据模型需要，自行添加后处理流程。

注：

1. 根据 step3 的 log 确认生成了哪些文件，需要注意下（如 caffe ssd 网络会生成一个 mbox_priorbox_185.bin 文件）
2. 工具包 demo 目录为提供的模型转换 demo：

   Demo 中执行顺序：

   Step1： 0_import_model.sh

   Setp2： 1_quantize_model.sh

   Step3： 2_export_case_code.sh
3. 当前执行完三步，可将 demo 中的 mobilenet_v1 模型转出 case 代码。
4. 转换自己的模型，可以将模型放到该目录下，然后修改 sh 脚本中的参数即可。
5. Demo 中的 extractoutput.py，执行命令：python extractoutput.py xxx.json 可以生成 outnamelist.txt，里面记录的是 Blob name 与 tensor ID 的 map 关系，在取结果时候，可以参考下。

Step 4. PC 端推理仿真

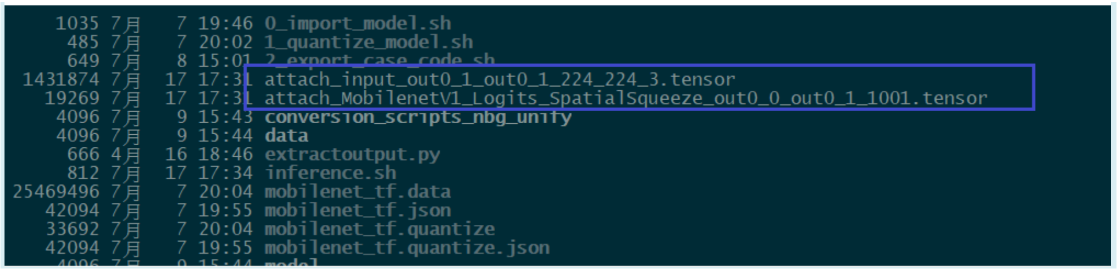
```
./bin/pegasus inference \
                    --dtype quantized   --model xxx.json \
                    --model-data xxx.data    --with-input-meta xxxx_inputmeta.yml \
```

注：

1. --dtype 设置 PC 端推理的方式， 参数为 quantized 、float32
2. Inference 的图片是 xxx_imputmeta.xml 中设置的量化时用的 source 文件，一般推理是只仿真一张，如果量化时用的多张图片，在推理时，请修改下 yml 中对应的 sourefile
3. 执行完 inference 之后，会在当前目录保存输入输出对应的 tensor 文件。输入文件为做完前处理之后的 float 数据。输出文件为已经反量化成 float 的数据。一般确认板侧运行的精度问题，建议使用 inference 保存的 tensor 文件作为输入文件。

   例如：

   Mobilenet 模型执行 inference 之后，保存了输入输出的 tensor 文件。可以使用 inference 时保存的输入 tensor 文件：attach_input_out0_1_out0_1_224_224_3.tensor 作为板子上 demo 执行时的输入文件。



```
    1035 7月    7 19:46 0_import_model.sh
     485 7月    7 20:02 1_quantize_model.sh
     649 7月    8 15:01 2_export_case_code.sh
 1431874 7月   17 17:31 attach_input_out0_1_out0_1_224_224_3.tensor
   19269 7月   17 17:31 attach_MobilenetV1_Logits_SpatialSqueeze_out0_0_out0_1_1001.tensor
    4096 7月    9 15:43 conversion_scripts_nbg_unify
    4096 7月    9 15:44 data
     666 4月   16 18:46 extractoutput.py
     812 7月   17 17:34 inference.sh
25469496 7月    7 20:04 mobilenet_tf.data
   42094 7月    7 19:55 mobilenet_tf.json
   33692 7月    7 20:04 mobilenet_tf.quantize
   42094 7月    7 19:55 mobilenet_tf.quantize.json
    4096 7月    9 15:44 model
```

Step 5. 量化模型

对于量化模型(如 tflite 量化模型),acuity_tool 工具也支持转换。当前执行 step1/step3 便可完成转换,此时可能需要用到--mean-values,--std-values，具体参照 3.6 章节 Pegasus 参数介绍，直接使用量化模型自带的量化参数，step1 中生成的 xxx_imputmeta.xml 不会对量化参数产生影响，只是 step3 生成 nbg case 时需要用到。也可以按顺序执行 step1/step2/step3，这样就会重新量化模型，量化模型本身的量化参数不会被用到。

3.5 混合量化

混合量化是对同一个模型的不同 layer 根据精度来使用不同的量化方式。例如，某个模型的某层 layer 的 8bit 量化效果差，导致最终结果误差大，我们可以使用混合量化方式，将对应的 layer 设置不同的量化方式，来提高精度.

Note：避免理解偏差，当前章节直接使用原始的英文介绍流程。

3.5.1 简要流程

1、Use the **--rebuild** and **--compute-entropy** arguments to quantize the network(.data and .json files) with a support quantization type. This generates a .quantize file and an entropy.txt files. In the xxx.quantize file, the **customized_quantize_layers** section provides suggested layer for a futher quantization with the **dynamic_fixed_point-i16** quantization type.

2、Update the customized_quantize_layers section to add, modify, or remove layers for a further quantization..To determine new quantization types for these layers, reference the entropies in entropy.txt. For high entropy layers, you can use the float32 type to improve the precision.. Note: he supported data types in customized_quantize_layers are dynamic_fixed_point-i16 and float32.

3、Use --hybrid and --model-quantize arguments with the updated .quantize file to quantize the network with the same quantization type as in step1.This performs a hybrid quantization and generates .quantize .json and .quantize files.

4、Export such hybrid-quantized model with the .data and the generated .quantize .json files. In the exported model, the dbype_converter layers are inserted.

3.5.2 详细步骤

Use lenet as an example to show how to change a layer from quantized dtype to a float dtype:

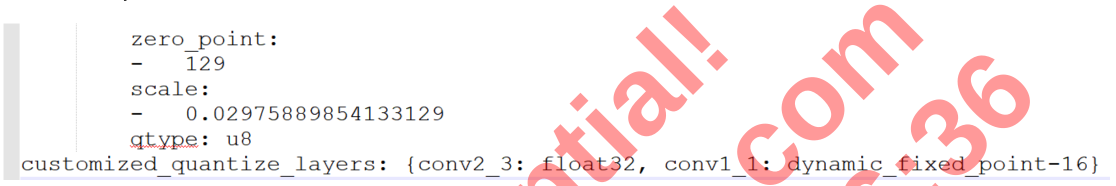
1、Use --rebuild and --compute-entropy to quantize an ACUITY model called lenet to generate a lenet.quantize file as flows:

```
$ ./bin/pegasus quantize \
        --quantizer asymmetric_affine \
        --qtype uint8 \
        --rebuild \
        --with-input-meta   lenet_inputmeta.yml \
        --model   lenet.json \
        --model-data   lenet.data
        --compute-entropy
```

2、Add layer name ‘conv2_3’ and corresponding quantized_dtype ‘float32’ to **customized_quantize_layers** of lenet.quantize.

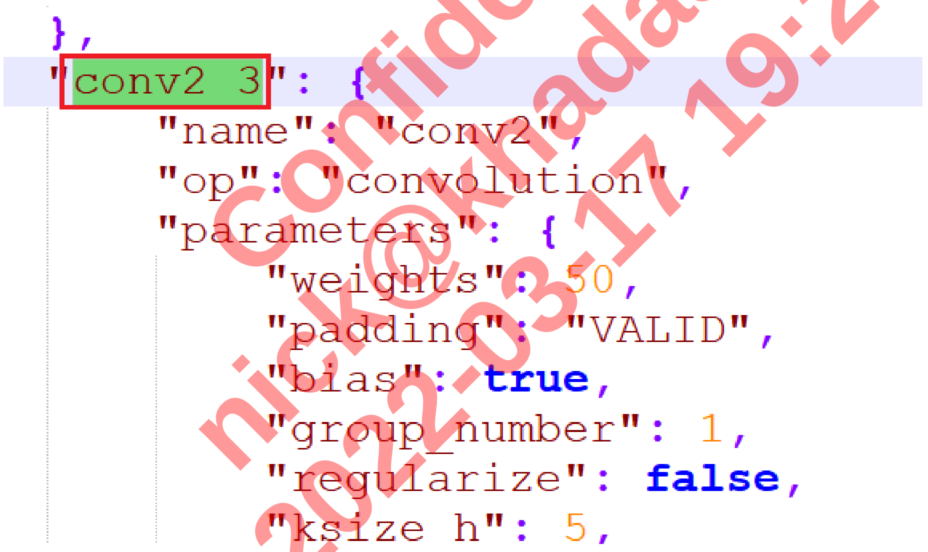
```
customized_quantize_layers: {conv2_3: float32}

Or customized_quantize_layers: {conv2_3: float32, conv1_1: dynamic_fixed_point-16}

CAUTION: if you want to change a connected subgraph of the network to non-quantized layers, set all the layers in this subgraph to  ‘float32’ , and you can get the layer name from the *.json file.
```

For example：



```
        zero_point:
        -   129
        scale:
        -   0.02975889854133129
        qtype: u8
customized_quantize_layers: {conv2_3: float32, conv1_1: dynamic_fixed_point-16}
```

The layer name from xxx.json file ， Specific as shown below:



```
},
"conv2_3": {
    "name": "conv2",
    "op": "convolution",
    "parameters": {
        "weights": 50,
        "padding": "VALID",
        "bias": true,
        "group_number": 1,
        "regularize": false,
        "ksize_h": 5,
```

3、 Use the --hybird argument and the updated lenet.quantize file to quantize the network into the asymmetric_affine-u8 data type again as follows:

```
$   ./bin/pegasus quantize \
      --model 'lenet.json' \
      --model-data 'lenet.data' \
      --model-quantize 'lenet.quantize' \
      --quantizer 'asymmetric_affine' \
      --qtype 'uint8' \
      --with-input-meta 'lenet_inputmeta.yml' \
      --hybrid
```

This generates hybrid network files lenet.quantize.json and lenet.quantize. The lenet.quantize.json file contains the newly added dtype_converter layers.

4、Use the lenet.data, lenet.quantize.json, lenet.quantize files generated by Step 3 to export application code (refer to Section 0) which wil result in some DATACONVERT layers inserted into the graph in the exported case.

```
$ ./bin/pegasus     export ovxlib
        --model lenet.quantize.json \
        --model-data lenet.data \
        --model-quantize lenet.quantize \
        --with-input-meta ${NAME}_inputmeta.yml
        --dtype quantized \
        --optimize VIPNANOQI_PID0X88   \
        --viv-sdk ../bin/vcmdtools \
        --pack-nbg-unify
```

1、Note: --optimize VIPNANOQI_PID0X88 Refer to chapter 3.4 step3

3.6 PASUS 扩展参数

3.6.1 Import Caffe

```
      ./bin/pegasus import caffe
          [-h] --model MODEL [--weights WEIGHTS]
          [--proto PROTO] [--output-model OUTPUT_MODEL]
          [--output-data OUTPUT_DATA]
```

Argument	Description
--model(required)	Requires a string value to denote the file path of the imported Caffe model, a .prototxt file
--weights	Requires a string value to denote the file path of the imported weights file, a .caffemodel file. If the weights file does not exist, the system generates a .data file which contains fake data
--output-model	Requires a string value to denote the file path of the generated ACUITY network model json file, When this argument is omitted, the system uses the default file path
--output-data	Requires a string value to denote the file path of the generated ACUITY network coefficient data data file, When this argument is omitted, the system uses the default file path
--proto	Requires a string value to represent the protocol used by the imported Caffe model. • 'caffe': (Default) Represents the standard Caffe format protocol. • 'lstm_caffe': Represents the LSTM layer protocol.

3.6.2 Import Tensorflow

```
      ./bin/pegasus import tensorflow
          [-h] --model MODEL
```

```
--inputs INPUTS
--input-size-list INPUT_SIZE_LIST
- - outputs OUTPUTS
[--size-with-batch SIZE_WITH_BATCH]
[--mean-values MEAN_VALUES]
[--std-values STD_VALUES]
[--predef-file PREDEF_FILE]
[--subgraphs SUBGRAPHS]
[--output-model OUTPUT_MODEL]
[--output-data OUTPUT_DATA]
```

--model(required)	Requires a string value to denote the file path of the TensorFlow frozen protocol buffer (protobuf) file, a `.pb` file. **Note:** Models trained and frozen by the following TensorFlow versions have been proven compatible with ACUITY: 1.4.x, 2.0.x, and 2.3.x.
--inputs (required)	Multiple points are separated with spaces. For example, 'input_point_1 input_point_2'.
--input-size-list(required)	Tensor shape sizes of the same input point are separated with commas. For example, '3,224,224', which represents the tensor shape sizes of one input point. Sizes between different input points are separated with hashtags. For example, '3,224,224#3,299,299#12,1', tensor shape sizes of three input points.
--outputs (required)	Multiple points are separated with spaces. For example, 'output_1 output_2'.
--outputmodel	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file
--size-with-batch	Requires one of the following values to specify whether the sizes listed in the **--input-size-list** argument contain batch dimension sizes of the input tensors. The batch dimension is the first dimension of an input tensor. • **True**: Contains. • **False**: Does not contain. **Note: True** or **False** for different input points are separated with commas and enclosed by ''. For example, 'True,False,True'. When this argument is omitted, the system uses **False** for all input points listed. Example 1: The sizes of the input tensor shape are (100, 243, 243, 3). Given the shape sizes listed in the **--input-size-list** argument are '243,243,3', set **--size-with-batch** to '**False**'.

	Example 2: Three input ports with only one input point at each port. The tensor shape sizes of each input point are (?, 243, 243, 3), (11, 88), and (10, 7). If the sizes listed in the **--input-size-list** argument are '243,243,3#88#10,7', set **--size-with-batch** to 'False,False,True'.
--mean-values	Requires a string to specify the mean value of each input point listed in the **--inputs** argument. Multiple mean values are separated with commas. For example, '128.0,128.0'. **Note:** Specify this argument only for quantized TensorFlow models.
--std-values	Requires a string to specify the standard value of each input point listed in the **--inputs** argument. Multiple standard values are separated with commas. For example, '128.0,128.0'. **Note:** Specify this argument only for quantized TensorFlow models.
--predef-file	Requires a string value to denote the file path of a predef file, an `.npz` file. Specify this argument to import complex models and enable custom control logics. To generate a predef file, you can use the NumPy function `np.savez(\'prd.npz', <path name>=<predefined value>)` where <path name> refers to the name of a network path for a specific compute stage. For example, `np.savez(\'prd.npz', train_stage=False)`. If a placeholder name contains unsupported characters, map the placeholder name to a supported alias. For example, NumPy does not support forward slashes. If the placeholder name is 'inference/train_stage', then use the following function to generate the predef file: `np.savez(\'prd.npz', stage=False, map={'stage':'inference/train_stage'})`. For more information about the `np.savez()` function, visit *NumPy documentation*.
--subgraphs	Requires a string to list the input points and output points of subgraphs. Specify this argument to import complex models. Use the following value syntax: • Point lists between different subgraphs are separated with semicolons. • For each subgraph, the input point list is followed by the output point list. The lists are separated with a hashtag.

	• Multiple points in each list are separated with commas. For example, 'graph1in1,graph1in2#graph1out1,graph1out2;graph2in1#graph2out1'.

3.6.3 Import Tflite

```
./bin/pegasus import tflite
    [-h]   --model MODEL
    [--inputs INPUTS]
    [--input-size-list INPUT_SIZE_LIST]
    [--size-with-batch SIZE_WITH_BATCH]
    [--output-model OUTPUT_MODEL]
    [--output-data OUTPUT_DATA]
    [--outputs OUTPUTS]
```

--model(required)	Requires a string value to denote the file path of the imported TensorFlow Lite (TFLite) model, a `.tflite` file. The supported TFLite schema is 0c4f5dfea4ceb3d7c0b46fc04828420a344f7598, committed on *https://github.com/tensorflow/tensorflow/commits/master/tensorflow/lite/schema/schema.fbs*. **Note:** Import failures may occur if the TFLite model uses an unsupported TFLite schema
--inputs	Multiple points are separated with spaces. For example, 'input_1 input_2'.
--input-size-list	Requires a string to represent the tensor shape sizes of the input points listed in the **--inputs** argument. • Tensor shape sizes of the same input point are separated with commas. For example, '3,224,224', which represents the tensor shape sizes of one input point. • Sizes between different input points are separated with hashtags. For example, '3,224,224#3,299,299#12,1', which represents the tensor shape sizes of three input points.
--size-with-batch	Requires one of the following values to specify whether the sizes listed in the **--input-size-list** argument contain batch dimension sizes of the input tensors. The batch dimension is the first dimension of an input tensor. • **True**: Contains. • **False**: Does not contain. **Note: True** or **False** for different input points are separated with commas and enclosed by ''. For example, 'True,False,True'.

	When this argument is omitted, the system uses **False** for all input points listed. Example 1: The sizes of the input tensor shape are (100, 243, 243, 3). Given the shape sizes listed in the **--input-size-list** argument are '243,243,3', set **--size-with-batch** to '**False**'. Example 2: Three input ports with only one input point at each port. The tensor shape sizes of each input point are (?, 243, 243, 3), (11, 88), and (10, 7). If the sizes listed in the **--input-size-list** argument are '243,243,3#88#10,7', set **--size-with-batch** to 'False,False,True'
--output-model	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file
--outputs	Multiple points are separated with spaces. For example, 'output_1 output_2' When this argument is omitted, the system uses all tail points of this model.

3.6.4 Import Darknet

```
./bin/pegasus import darknet
        [-h] --model MODEL --weights WEIGHTS
        [--output-model OUTPUT_MODEL]
        [--output-data OUTPUT_DATA]
```

--model(required)	a string value to denote the file path of the imported Darknet model, a `.cfg` file.
--weights	a string value to denote the file path of the imported weights file, a `.weights` file
--output-model	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file

3.6.5 Import Onnx

```
./bin/pegasus import onnx
        [-h] --model MODEL [--inputs INPUTS] [--outputs OUTPUTS]
        [--input-size-list INPUT_SIZE_LIST] [--size-with-batch SIZE_WITH_BATCH]
        [--input-dtype-list INPUT_DTYPE_LIST]
        [--output-model OUTPUT_MODEL]
        [--output-data OUTPUT_DATA]
```

--model(required)	a string value to denote the file path of the imported ONNX model, an `.onnx` file **Note:** ONNX operator sets 1 through 11 are supported.
--inputs	Requires a string to list the input points of the ONNX model. Multiple points are separated with spaces. For example, 'input_1 input_2'. If this argument is omitted, the system uses all head points of the imported model.
--input-size-list	Requires a string to represent the tensor shape sizes of the input points listed in the --inputs argument. • Tensor shape sizes of the same input point are separated with commas. For example, '3,224,224', which represents the tensor shape sizes of one input point. • Sizes between different input points are separated with hashtags. For example, '3,224,224#3,299,299#12,1', which represents the tensor shape sizes of three input points. When this argument is omitted, the system automatically detects the tensor shape sizes of the input points.
--size-with-batch	Requires one of the following values to specify whether the sizes listed in the **--input-size-list** argument contain batch dimension sizes of the input tensors. The batch dimension is the first dimension of an input tensor. • **True**: Contains. • **False**: Does not contain. **Note: True** or **False** for different input points are separated with commas and enclosed by ''. For example, 'True,False,True'. When this argument is omitted, the system uses **False** for all input points listed. Example 1: The sizes of the input tensor shape are (100, 243, 243, 3). Given the shape sizes listed in the **--input-size-list** argument are '243,243,3', set **--size-with-batch** to '**False**'. Example 2: Three input ports with only one input point at each port. The tensor shape sizes of each input point are (?, 243, 243, 3), (11, 88), and (10, 7). If the sizes listed in the **--input-size-list** argument are '243,243,3#88#10,7', set **--size-with-batch** to 'False,False,True'.
--input-dtype-list	Requires a string to denote data types of the input tensors on the input points. The allowed strings for the supported data types are '**float**', '**int8**', '**uint8**', '**int16**', and '**uint16**'. String values for

	the data types of different input tensors are separated with hashtags. For example, 'float#int8#uint16'. When this argument is omitted, the system uses the default value **'float'** for all input tensors.
--output-model	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file
--outputs	Requires a string to specify the output points of the ONNX model. Multiple points are separated with spaces. For example, 'output_1 output_2'. When this argument is omitted, the system uses all tail points of this model.

3.6.6 Import Pytorch

```
./bin/pegasus import pytorch
        [-h] [--model MODEL]
        [--inputs INPUTS]
        [--outputs OUTPUTS]
        [--input-size-list INPUT_SIZE_LIST]
        [--size-with-batch SIZE_WITH_BATCH]
        [--output-model OUTPUT_MODEL]
        [--output-data OUTPUT_DATA]
        [--config CONFIG]
```

--model(required)	a string value to denote the file path of the imported PyTorch model, a `.pt` file.
--inputs	Requires a string to list the input points of the PyTorch model. Multiple points are separated with space. For example, 'input_1 input_2'. If this argument is omitted, the system uses all head points of the imported model.
--outputs	Multiple points are separated with spaces. For example, 'output_1 output_2'.
--input-size-list	Requires a string to represent the tensor shape sizes of the input points listed in the **--inputs** argument. • Tensor shape sizes of the same input point are separated with commas. For example, '3,224,224', which represents the tensor shape sizes of one input point.

	• Sizes between different input points are separated with hashtags. For example, '3,224,224#3,299,299#12,1', which represents the tensor shape sizes of three input points. When this argument is omitted, the system automatically detects the tensor shape sizes of the input points.
--size-with-batch	Requires one of the following values to specify whether the sizes listed in the **--input-size-list** argument contain batch dimension sizes of the input tensors. The batch dimension is the first dimension of an input tensor. • **True**: Contains. • **False**: Does not contain. **Note: True** or **False** for different input points are separated with commas and enclosed by ''. For example, 'True,False,True'. When this argument is omitted, the system uses **False** for all input points listed. Example 1: The sizes of the input tensor shape are (100, 243, 243, 3). Given the shape sizes listed in the **--input-size-list** argument are '243,243,3', set **--size-with-batch** to '**False**'. Example 2: Three input ports with one input point at each port. The tensor shape sizes of each input point are (?,243, 243, 3), (11, 88), and (10, 7). If the sizes listed in the **--input-size-list** argument are '243,243,3#88#10,7', set **--size-with-batch** to 'False,False,True'.
--output-model	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file
-config	Requires a string value to denote the file path of the configuration file, a `.json` file. This is an alternative method to translate a PyTorch model to ACUITY formats. With this argument, you do not need to use any other configuration arguments listed above.

3.6.7 Import Keras

```
    ./bin/pegasus import keras
        [-h] --model MODEL
        [--convert-engine {Keras,TFLite}]
        [--inputs INPUTS]
        [--input-size-list INPUT_SIZE_LIST]
        [--outputs OUTPUTS]
        [--output-model OUTPUT_MODEL]
        [--output-data OUTPUT_DATA]
```

--model(required)	a string value to denote the file path of the imported Keras model, a `.h5` file
--convert-engine	Reserved for future use.
--inputs	Multiple input points are separated with spaces. For example, 'input_1 nput_2'.
--input-size-list	Requires a string to represent the tensor shape sizes of the input points. These sizes must not contain batch sizes, which are the first dimensions of the input tensor shapes. • Tensor shape sizes of the same input point are separated with commas. For example, '3,224,224', which represents the tensor shape sizes of one input point. • Sizes between different input points are separated with hashtags. For example, '3,224,224#3,299,299#12,1', which represents the tensor shape sizes of three input points. When this argument is omitted, the system automatically detects the tensor shape sizes of the input points
--outputs	Multiple output points are separated with spaces. For example, 'output_1 output_2'.
--output-model	a string value to denote the file path of the generated ACUITY network model file
--output-data	a string value to denote the file path of the generated ACUITY network data file

3.6.8 Generate Inputmeta

```
./bin/pegasus generate inputmeta
        [-h] [--input-meta-output INPUT_META_OUTPUT]
        [--separated-database SEPARATED-DATABASE]
        --model MODEL --source-file SOURCE_FILE --channel-mean-value
```

--model(required)	a string value to denote the file path of an ACUITY network model file, a `.json` file
--input-meta-output	a string value to denote the file path of the generated inputmeta file. When this argument is omitted, the system uses the default file path .

--separated-database	separate the database into multiple ones in the generated inputmeta file for a multi input ACUITY network. This argument is valid only for multi-input networks. When this argument is added,separate the database into multiple ones
--source-file	a string value to denote the data file for the inputs.if you separate the multi-input, you can set “dataset1.txt#dataset2.txt#dataset3.txt”.you can also modify the yml file directly
--channel-mean-value	input channel mean value.if you separate the multi-input, you can set "128 128 128 0.00715#128 128 128 0.00715#128 128 128 0.00715".you can also modify the yml file directly

3.6.9 Quantize

```
./bin/pegasus quantize
        [-h] --model MODEL
        --model-data MODEL_DATA
        [--model-quantize MODEL_QUANTIZE]
        [--batch-size BATCH_SIZE] [--iterations ITERATIONS]
        [--device DEVICE] --with-input-meta WITH_INPUT_META
        [--quantizer QUANTIZER] [--qtype QTYPE]
        [--hybrid] [--rebuild] [--rebuild-all]
        [--compute-entropy] [--algorithm ALGORITHM]
        [--moving-average-weight MOVING_AVERAGE_WEIGHT]
        [--divergence-nbins DIVERGENCE_NBINS]
        [--divergence-first-quantize-bits DIVERGENCE_FIRST_QUANTIZE_BITS]
        [--MLE]
```

--model(required)	a string value to denote the file path of an ACUITY network model file, a `.json` file
--model-data(required)	a string value to denote the file path of an ACUITY model coefficient data file
--model-quantize	Requires a string value to denote the file path of the generated `.quantize` file when the **--rebuild** argument is specified, or the file path of the `.quantize` file to use when the **--hybrid** argument is specified. If this argument is omitted when **–rebuild** is specified, the `.quantize` file is generated in the same directory as the model file.
--batch-size	Requires an integer to specify the number of sample images per batch. • If the original network uses a fixed batch size, use the fixed batch size.

	• If the original network uses a variable batch size, set this argument to 1. When this argument is omitted, the system uses the value of **shape[0]** in the inputmeta file. **Note:** This argument is used together with **--iterations**.
--iterations	Requires an integer to specify the number of sample image batches. For KL divergence, moving-average, and automatic hybrid quantization algorithms, 500 to 1000 iterations are recommended. When this argument is omitted, the system uses the default value 1. **Note:** This argument is used together with **--batch-size**
--device	Requires one of the following values to specify the compute device type. • '**GPU**' • '**CPU**': Default When this argument is omitted, the system uses the default device type.
--with-input-meta(Required)	Requires a string value to denote the file path of the inputmeta file, a `.yml` file
--quantizer	Requires one of the following values to specify the quantizer for quantizing network tensors: • '**asymmetric_affine**': Default • '**dynamic_fixed_point**' • '**perchannel_symmetric_affine**' • '**qbfloat16**' When this argument is omitted, the system uses the default quantizer. **Note:** This argument is used together with **--qtype**.
--qtype	Requires one of the following values to specify the quantization data type: • '**int8**' • '**int16**' • '**uint8**': Default • '**qbfloat16**' When this argument is omitted, the system uses the default data type. **Note**: This argument is used together with **--quantizer**.

--hybrid	Performs hybrid quantization on the network model. This argument is mutually exclusive with the **–rebuild** and **--rebuild-all** arguments.
--rebuild	Performs quantization with the quantization file generated. To quantize the network with a quantizer other than **qbfloat16**, specify this argument. For details about quantizer types, see the description of --quantizer. This argument is mutually exclusive with the **--hybrid** and **--rebuild-all** arguments.
--rebuild-all	Rebuilds a quantization table with the default quantization rules and quantizes all activations. To quantize the network with the **qbfloat16** quantizer, specify this argument. For details about quantizer types, see the description of --quantizer. This argument is mutually exclusive with the **--hybrid** and **--rebuild** arguments.
--algorithm	Requires one of the following values to specify the quantization algorithm: • '**normal**': The default algorithm. • '**kl_divergence**': The KL divergence algorithm. If this value is chosen, specify either **--divergence-nbins** or **--divergence-first-quantize-bits**. • '**moving_average**': The moving-average algorithm. If this value is chosen, specify the **--moving-average-weight** argument. • '**auto**': The KL-divergence-based hybrid quantization algorithm with quantized layers automatically determined. ▪ If **--MLE** is specified, the quantized layers are determined by the Cosine Similarity measure. In this case, the automatic hybrid quantization algorithm and the MLE function together ensure a high quantization precision. ▪ If **--MLE** is not specified, the quantized layers are determined based on layer entropy. When this argument is omitted, the system uses the default algorithm.
--moving-average-weight	Requires a positive floating-point value to specify the coefficient for the moving average model. This argument is valid only if the **--algorithm** argument is set to '**moving_average**'.

	When this argument is omitted, the system uses the default coefficient 0.01.
--divergence-nbins	Requires an integer to specify the number of bins in the Kullback-Laiber divergence (KL divergence) histogram. The integer must equal 2_N where N is a positive integer. Specify this argument only if **--algorithm** is set to '**kl_divergence**'. When this argument is used, do not specify the **--divergence-first-quantize-bits** argument. When this argument is omitted, the system uses the value 2_{11}. **Note: --divergence-nbins** will be deprecated. Use **--divergence-first-quantize-bits** instead.
--divergence-first-quantize-bits	Requires a positive integer to calculate the number of bins in the KL divergence histogram. Given an integer m, the calculated KL bin count is 2_m. When this argument is omitted, the system uses the default value 11. **Note:** Specify this argument only if **--algorithm** is set to '**kl_divergence**'. When this argument is used, do not specify the **--divergence-nbins** argument
--compute-entropy	Measures the precision of the current quantization by calculating the entropy of each layer in the range of 0 to 1. The system stores these values in the `entropy.txt` file in the current workspace. If the entropy is low, the precision is high. If the entropy is high, the precision is low. When this argument is omitted, the system does not perform such measurement. **Note:** This argument does not require values. It is valid only if the **--batch-size** argument is set to 1.
--MLE	Minimizes per-layer quantization errors of the network by automatically adjusting quantization parameters. This function can be applied to all the supported quantization algorithms. Among them, automatic hybrid quantization together with the MLE function ensures a high quantization precision. **Note:** If this argument is specified, the quantization process may be time-comsuming

3.6.10 Export Ovxlib

```
./bin/pegasus export ovxlib
        --model    --model-data MODEL_DATA
        [--model-quantize MODEL_QUANTIZE]
        [--postprocess-file POSTPROCESS_FILE]
        [--output-path OUTPUT_PATH]
        --with-input-meta WITH_INPUT_META
        [--optimize OPTIMIZE] [--dtype DTYPE]
        [--save-fused-graph] [--pack-nbg-unify]
```

```
          [--pack-nbg-viplite] [--viv-sdk VIV_SDK]
          [--build-platform 'make']
          [--batch-size BATCH_SIZE] [--force-remove-permute]
          [--customer-lids] [--customer-ops]
```

--model(required)	a string value to denote the file path of an ACUITY network model file
--model-data(required)	a string value to denote the file path of an ACUITY model coefficient data file
--with-input-meta(required)	Requires a string value to denote the file path of the inputmeta file, a `.yml` file.
--model-quantize	a string value to denote the file path of a quantized tensor description file
--postprocess-file	a string value to denote the file path of the post-processing configuration file, a `.yml` file. Multiple tasks can be set in one configuration file. You can either create a post-processing file or use the generation command to generate a post-processing file
--output-path	a string value to denote the directory of the exported applications with the prefix specified
--optimize	Requires one of the following values to specify the optimization method during the export. • '**None**': Does not optimize. • '**default**': (Default) Optimizes the model based on the default rules. • '<configuration file path or configuration name>': Optimizes the model based on the specified configuration file. Specify a configuration file or a configuration name if the **--pack-nbg-unify** or **--pack-nbg-viplite** argument is specified
--dtype	Requires one of the following values to specify the tensor data type of the exported OVXLIB application: • '**float**' or '**float16**': '**float**' and '**float16**' are equivalents. '**float**' is the default data type. • '**float32**' • '**quantized**': A quantization data type specified in the `.quantize` file when the input model is a quantized model. If this value is chosen, specify the **--model-quantize** argument. When this argument is omitted, the system uses the default data type

--save-fused-graph	(Experimental use only) Saves a fused model to a `.json` file for debugging use. It is mutually exclusive with **-- pack-nbg-unify** and **--pack-nbg-viplite** arguments. The generated fused model contains network structures only of the exported unify applications. When this argument is omitted, no fused model is saved. **Note:** This argument does not require values.
--force-remove-permute	(Experimental use only) Removes the head permutation layers inserted after the input layer and the tail permutated layers inserted before the output layer. This argument is used only for export of unify applications from TensorFlow, TensorFlow Lite, and Keras models. It is mutually exclusive with **--pack-nbg-unify** and **--pack-nbg-viplite** arguments. When this argument is omitted, permutation layers are kept and the tensor shape is in the NHWC sequence. When this argument is specified, the tensor shape may be in the NCHW sequence. Make sure that the data sequence of the tensor shape is NHWC before application deployment onto devices with Vivante NPUs. For example, for a TensorFlow model with input of (1,224,224,3) in the NHWC sequence: • If this argument is not specified, the tensor with the shape (1,224,224,3) in the NHWC sequence is used. • If this argument is specified, the tensor shape may be (1,3,224,224) in the NCHW sequence. When the data sequence is NCHW, convert it into NHWC before application deployment. **Note:** This argument does not require values
--pack-nbg-unify	Packs binary graphs for the unified driver and generates three applications: • unify application: An `.nb` file in the output directory. • nbg_unify application: An `.nb` file in the unify subdirectory of the output directory. • nbg_unify_ovx application: An `.nb` file in the unify subdirectory of the output directory. **Important:** nbg_unify_ovx applications will be obsoleted at the end of 2021. The output directory is specified by the **--output-path** argument. **Note:** • If neither **--pack-nbg-unify** nor **--pack-nbg-viplite** is specified, only the unify application is generated.

	• This argument is mutually exclusive with the **--pack-nbg-viplite** argument. • Packing is not supported for edge devices with CPU nodes. If CPU nodes exist, use PPU nodes instead
--pack-nbg-viplite	Packs binary graphs for the VIPLite driver and generates two applications: • unify application: An `.nb` file in the output path. • nbg_unify application: An `.nb` file in the output path. The output directory is specified by the **output_path** argument. **Note:** • If neither **--pack-nbg-unify** nor **--pack-nbg-viplite** is specified, only the unify application is generated. • This argument is mutually exclusive with the **--pack-nbg-unify** argument. • Packages are not supported for edge devices with CPU nodes. If CPU nodes exist, use PPU nodes instead
--viv-sdk	Requires a string value to denote the file path of the directory that contains the binary SDK of VSim. During execution, VSim generates NBG files. For example, the file path may be '`/home/xxx/Verisilicon/VivanteIDEx.x.x/*cmdtools`' if VivanteIDE is installed. Specify this argument if the **--pack-nbg-unify** or **--pack-nbg-viplite** argument is specified
--build-platform	Builds a compiling tool to generate NBG applications. The available value is '**make**'. When this argument is omitted, the system uses the default value '**make**'.
--batch-size	Requires a positive integer to specify the batch size that the exported application supports. When this argument is omitted, the system uses the value of **shape[0]** in the inputmeta file.
--customer-lids	Reserved for future use.
--customer-ops	Reserved for future use.

3.6.11 Inference

```
./bin/pegasus inference
        [-h] --model MODEL --model-data MODEL_DATA
        [--model-quantize MODEL_QUANTIZE]
        [--batch-size BATCH_SIZE] [--iterations ITERATIONS]
        [--device DEVICE] [--with-input-meta WITH_INPUT_META]
```

```
          [--dtype DTYPE] [--postprocess POSTPROCESS]
          [--postprocess-file POSTPROCESS_FILE]
                    [--output-dir OUTPUT_DIR]
```

--model(required)	a string value to denote the file path of an ACUITY network model file, a `.json` file
--model-data(required)	a string value to denote the file path of an ACUITY model coefficient data file
--model_quantize	a string value to denote the file path of a quantized tensor description file, a `.quantize` file. **Note:** Specify this argument if the **--dtype** argument is set to '**quantized**'.
--batch-size	Requires an integer to specify the number of sample images per batch. When this argument is omitted, the system uses the value of **shape[0]** in the inputmeta file.Set this argument to a small value if the RAM or GPU does not support a large batch size. **Note:** This argument is used together with **--iterations**
--iterations	Requires an integer to specify the number of sample image batches. When this argument is omitted, the system uses the default value 1. **Note:** This argument is used tog
--devices	Requires one of the following values to specify the compute device type: • '**GPU**' • '**CPU**': Default When this argument is omitted, the system uses the default device type.
--with-input-meta(required)	Requires a string value to denote the file path of the inputmeta file, a `.yml` file.
--dtype	Requires one of the following values to specify the tensor data type of the input model: • '**float32**': The default data type. • '**quantized**': A quantization data type specified in the `.quantize` file when the input model is a quantized model. If this value is chosen, specify the **--model-quantize** argument. When this argument is omitted, the system uses the default data type

--postprocess	Requires one of the following values to specify the post-processing task: • '**print_topn**': Prints the top 5 output tensors of the output layers. • '**dump_result**': Dumps output tensors for input layers and output layers. • '**classification_classic**': (Default) Performs both of the actions. When this argument is omitted, the system uses the default value. **Note:** This argument is mutually exclusive with the **--postprocess-file** argument
--postprocess-file	Requires a string value to denote the file path of the post-processing configuration file, a `.yml` file. Multiple tasks can be set in one configuration file. You can either create a post-processing file or use the generation command to generate a post-processing file. **Note:** This argument is mutually exclusive with the **--postprocess** argument
--output-dir	Requires a string value to denote the directory for the generated files.