Model

Transcoding and Running User Guide

Revision: 0.8

Release Date: 2021-01-15

Copyright

Trademarks

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

Revision History

Issue 0.8 (2021-01-15)

This is the eighth official release.

Compared to last version, the following changes are made:

- Add environmental installation notes
- Add guidance for using multiple pictures when quantifying

Issue 0.7 (2020-07-18)

This is the seventh official release.

Compared to last version, the following changes are made:

1. Added 3.7 to describe transformation instruction using multi-input model.
2. Deleted IDE tool related description.

Issue 0.6 (2020-05-18)

This is the sixth official release.

Compared to last version, 3.5, 3.6 and their topics are added.

Issue 0.5 (2019-12-20)

This is the fifth official release.

Compared to last version, the following changes are made:

1. Added a guide for using acuity tool to inference.
2. Added detailed guidance on quantization parameters during model conversion
3. Removed the FAQ chapter and moved the corresponding chapter to the Amlogic-NN-FAQ document.
4. Added darknet and Keras framework model imports instructions.
5. Add hybrid quantification guidance

Issue 0.4 (2019-07-18)

This is the fourth official release.

Compared to last version, the following changes are made:

1. Optimized the environment installation procedure and provided the one-click installation method.
2. Added introduction to model quantification and guidance to quantitative parameter selection for model transcoding.
3. Deleted the project code compilation chapter. For details about project code compilation, see *Android&Linux Development Guide.docx*.
4. Deleted the general knowledge chapter.
5. Added quantitative/anti-quantization guidance in the FAQ chapter to help customers solve the problem of exceedingly long processing time.

Issue 0.3 (2019-07-08)

This is the third official release.

Compared to last version, the following changes are made:

1. Added ONNX model transcoding commands.
2. Added whl package installation instructions for the model transcoding tool.
3. Changed the type of DDK_6.3.3.4 offline model to nbg and modified the corresponding module description.
4. Deleted the code summary chapter because the case code directory is directly generated after the case code is transcoded through the DDK_6.3.3.4 model.
5. Added the case code introduction chapter.

Issue 0.2 (2018-12-20)

This is the second official release.

Compared to last version, the following changes are made:

1. Modified errors of and missing information about the default environment installation procedure.
2. Deleted the chapter describing how to use IDE to transcode case code and adjust content in some chapters.
3. Simplified the transcoding process with some of the operations deleted and move the FAQ part to Chapter 7.
4. Added the IDE tool instructions chapter, introducing how to import case code, add the jpeg library, and compile and run the code.
5. Added the method for external personnel to obtain version information in the project code chapter and the introduction to case code compilation based on internal and external code.
6. Added the FAQ chapter.

Issue 0.1 (2018-11-12)

This is the Initial release.

Contents

1. Introduction

This document introduces how you can use the Acuity_tool to transcode case code from a model for the convenience of model transcoding for both internal and external developers.

Currently, NN chips only support models of tensorflow, caffe, tflite, darknet, onnx,pytorch and keras types. In this case, if you use another type of model, you need to convert the model type to one of the preceding types at first.

2. Tools

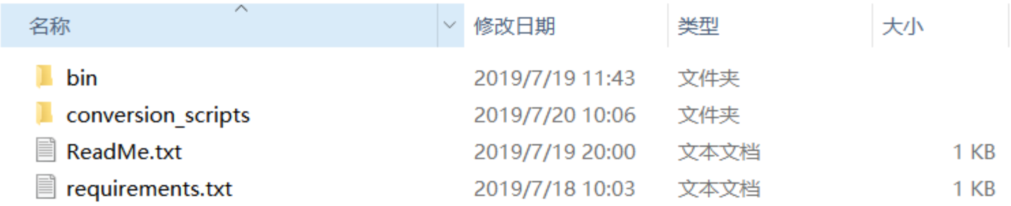
2.1 Overview

Currently, you can use the following tools to transcode case code.

1. The acuity-toolkit model tool: You can use it to quantize models and transcode case code, which can be run after compilation.

2.2 Model Transcoding Tool

2.2.1 Introduction to the Tool Directory



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

The **bin** folder includes executable files required for model transcoding and the configuration file directory.

The **conversion_scripts** folder includes model transcoding demos (including transcoding scripts and the mobilenet_v1.pb model). You only need to run the three sh scripts in sequence after the environment is ready. Before you transcode your own model, be sure to modify the configuration parameters of the three scripts.

The **requirements.txt** file lists all the software packages required by the environment.

The **ReadMe.txt** file simply introduces how the tool works.

2.2.2 Environmental Dependencies

Operating System	**Ubuntu16.04 （x64)**
Python	Python3
Dependent library	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

Note: The above only lists some libraries that are currently needed, and the specific dependent libraries are based on the requirements.txt in the acuity_toolkit-binary-xxx inside the tool.

2.2.3 Installing the Tool

Step 1 Prepare a computer with the 64-bit Ubuntu 16.04 system. (If it is a virtual machine, it's RAM memory space must be larger than 4 GB.),Python requires version 3.5.2.

Step 2 Run the following command to install python3 and pip:

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

Step 3 Do as follows to install the related dependencies:

```
Obtain the acuity-toolkit-binar toolkit and go to the root directory.

Execute:   for req in $(cat requirements.txt); do pip3 install $req; done
```

Step 4 Check the environment

```
Execute: python3 bin/checkenv.py

Note: if check success, will have Env Pass: Env check SUCCESS!!! print.
```

3. Model Transcoding

3.1 Introduction

During model transcoding, models are first quantized to data in the int8, int16, or uint8 format, and then are converted to nbg files that can be run in our platform. Then, case code can be exported.

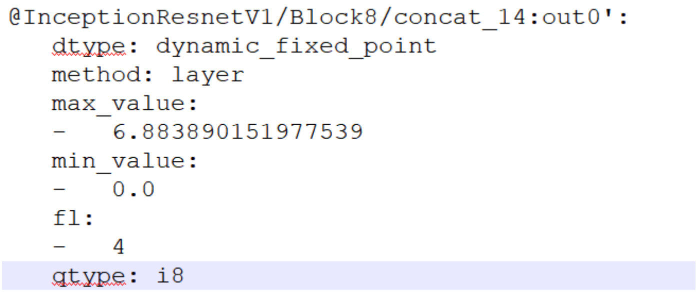
3.2 Quantization

Quantization is the process that 32- or 64-bit floating point numbers are stored as 2-bit data.

3.2.1 Introduction

Int8: indicates that 32-bit floating point numbers are quantized to int8 data. Int8 data includes eight bits, among which one is the sign bit and the other seven are valid numbers. You need to calculate the fl value that indicates the number of bits in the decimal. (The mechanism for int16 is similar to that of int8.)

You can refer to xxxx.quantize that is generated after Step 2 in section 3.4 is executed. The following figure shown the analysis process:



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

Analysis: qtype indicates that the quantization mode is int8. The value range of the calculation result is (0.0, 6.88).

The maximum absolute value of the result is 6.88, which only requires 3 bits for the integer part.

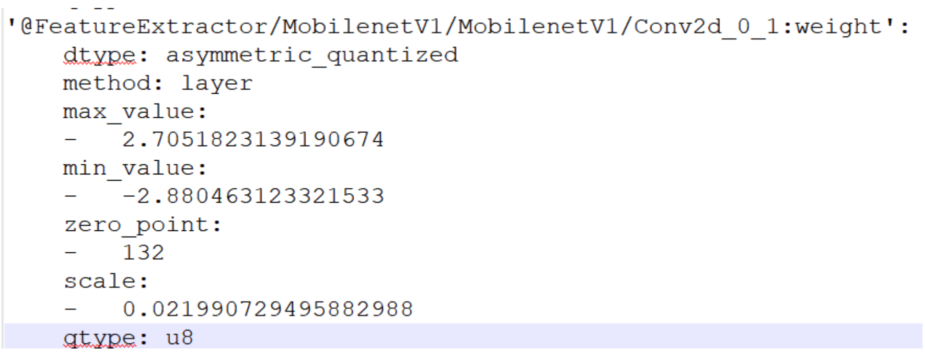
Therefore, the number of bits used to represent the decimal is fl=8-1-3=4.

u8: indicates that 32-bit floating point numbers are quantized to unsigned int8 data. All eight bits indicate numbers, and there is no sign bit. That is, the value range is 0 to 255. In this case, you need to calculate two parameters, scale and zero_point based on the following formula:

scale = (max_value - min_value)/255

zero_point = max_value - max_value/scale

You can refer to xxxx.quantize that is generated after Step 2 in section 3.4 is executed.



```
'@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
```

Analysis: qtype indicates that the quantization mode is u8. The value range of calculation results is (–2.88, 2.7).

Therefore, the parameters are calculated as follows:

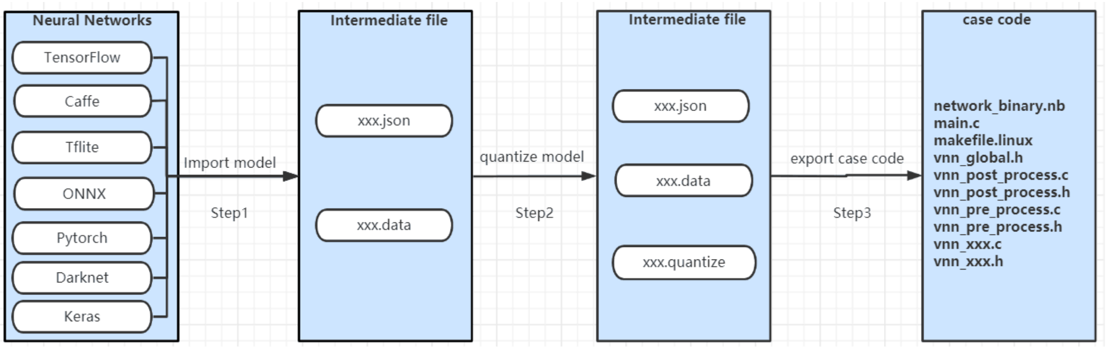
scale = (2.705 -(-2.88))/255= 0.0219

zero_point = 255 - 2.705/scale = 132

3.2.2 Rules of Selecting a Proper Quantitation Method

1. In normal cases, u8 and int8 are applicable to all models. However, Google recommends u8.
2. In normal cases, int16 is not recommended because it requires models with size that is twice the models used for 8-bit methods. Additionally, int16 has no advantage in accuracy.
3. Take the value ranges for pre- and post-processing into consideration. For example, if the value range for pre-processing is (0, 255), u8 is recommended.
4. Randomly select a quantization method. After quantization is completed, check the generated xxxx.quantize file and check the max_value and min_value of related statistics. If there is any value with its absolute value greater than 256, int16 is recommended. If there is any value with its absolute value greater than 128, u8 is recommended. In other cases, both int8 and u8 are applicable.

3.3 Model Conversion Flowchart



3.4 Model Transcoding Procedure

Model transcoding is performed in the **acuity-toolkit** directory. The executable files are all in bin directory.

Step 1 Run the following commands to convert the model into an intermediate file:

For Caffe models:

```
./bin/convertcaffe    --caffe-model xx.prototxt

                --caffe-blobs xx.caffemodel --data-output xxx.data --net-output xxx.json
```

For Darknet models:

```
./bin/convertdarknet    --net-input yolov2.cfg    \

              --weight-input yolov2.weights --data-output yolov2.data --net-output yolov2.json
```

For Tensorflow models:

```
./bin/convertensorflow --tf-pb inceptionV1.pb    \

                --inputs input    --outputs InceptionV1/Logits/Predictions/Reshape_1 \

                --net-output inceptionV1.json    --data-output inceptionV1.data      \

                --input-size-list '224,224,3'
```

For Tflite models:

```
./bin/convertflite    --tflite-mode    ssd_big_graph.tflite    \

                --net-output ssd_big_graph.json --data-output ssd_big_graph.data
```

For Onnx models:

```
./bin/convertonnx    --onnx-model    vgg16.onnx    \

                --net-output vgg16.json    --data-output vgg16.data
```

For Keras model:

```
./bin/convertkeras --keras-model simple_CNN.81-0.96.hdf5 \
                  --net-output cnn.json --data-output cnn.data
```

For Pytorch model:

```
./bin/convertpytorch --pytorch-model cnn_new.pt --net-output cnn.json \
                  --data-output cnn.data --input-size-list '1,28,28'
```

Result: Two intermediate files, **xxx.json** and **xxx.data** are generated.

Note

1. *For tensorflow models, names of the input and output nodes and the size of the input HWC are required. The information can be obtained through summarize_graph or tensorboard.*
2. ***--inputs/--outputs** indicates names of input/output nodes. If there are multiple input or output nodes, separate their names with spaces, for example, **output1 output2 output3**.*
3. ***--input-size-list** If there are multiple input nodes, separate their names with number signs (#), for example, **224,224,3#299,299,3#12,12**.*

4. *Currently, only Caffe/Tensorflow/Tflite/Darknet/Onnx models are supported. If you use models of other types, convert them to one of the preceding types first.*
5. If there are some logic control inputs for the tensorflow model, please refer to the 3.17 instructions in the Amlogic-NN-FAQ document for processing.

Step 2 Run the following commands to quantize the model:

```
./bin/tensorzonex    --action quantization \

       --quantized-dtype asymmetric_affine-u8 \

       --channel-mean-value '128 128 128 1'    \



       --source text --source-file dataset.txt        \

       --model-input xxx.json -- model-data xxx.data \

      --reorder-channel '0 1 2'    \

       --quantized-rebuild
```

Result: According to the input image in dataset.txt, the maximum and minimum values of each layer and those of the model weight can be calculated through forward reasoning. You can calculate the quantization parameters of each layer and save them as quantized result file **xxx.quantize**, which will be used when you export case code.

Note

1. ***--channel-mean-value*** *indicates that you need to set the pre-processing command-line parameters based on the pre-processing methods used for model training. It includes four values, m1, m2, m3, and scale. The first three are mean-value parameters and the last one is the scale parameter. The following uses the three-channel input data (data1, data2, data3) to show how pre-processing is performed:*

   Out1 = (data1-m1)/scale

   Out2 = (data2-m2)/scale

   Out3 = (data3-m3)/scale

For example: if the pre-processing requires that data is normalized to the range [–1, 1], set the parameter value range to (128 128 128 128).

If the pre-processing requires that data is normalized to the range [0, 1], set the parameter value range to (0 0 0 256).

For single-channel parameters, set the parameter value range to (m1, 0, 0, scale).

2. ***dataset.txt*** *specifies the quantized input image path, for example,* ***/data/test/cat.jpg****. You are advised to provide about 200 images showing the use and running conditions for quantization to ensure that the maximum and minimum values of the statistics are the same as those when the model is running for better quantization effects.*

*When the number of pictures is greater than 100, it is recommended to add quantization parameters --batch-size(default 100) and --epochs(default 1), epochs*batch-size=number of pitures.For example, there is 5000 pictures, then you can set parameters:--batch-size 100 -epochs 50. if the computer memory is small , you can set the value of batch-size to a small value.*

3. ***--quantized-dtype*** *indicates the data type for quantization. Supported types are as follows:*

   dynamic_fixed_point-i8 (int8 quantization)

   dynamic_fixed_point-i16 (int16 quantization)

   asymmetric_affine-u8 (unit8 quantization, default)

4. ***--quantized-rebuild** is the default parameter. You are advised to set it because if you do not set it and you use different quantization methods to quantize a model, the second quantization process does not take effect on the condition that you do not delete the generated **xxx.quantize** file.*
5. *--reorder-channel set the format of data placement*

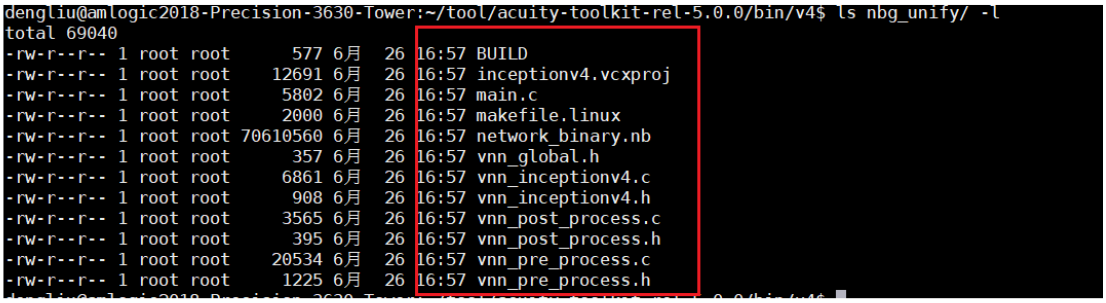
   Caffe model is in bgr format:that is '2 1 0'

   Tensorflow model is rgb format, that is '0 1 2'

Step 3 Run the following commands to the ovxgenerator.py file in the Vivante_Acuiti_Trainer tool to generate the case code:

```
./bin/ovxgenerator    --model-input 190328.json \

          --data-input 190328.data \

          --channel-mean-value '128 128 128 1' \

          --reorder-channel '0 1 2' \

          --export-dtype quantized \

          --optimize VIPNANOQI_PID0X88    \

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

          --pack-nbg-unify
```

Result: Case code is generated, as shown in the following figure:



```
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
```

Note

1. *The **xxx.quantize** file is not used as an input parameter and is read by default.*
2. ***--reorder-channel** indicates the data sequence.*

   *The data sequence for caffe models is in bgr format, for example, **2 1 0**.*

   *The data sequence for tensorflow models is in rgb format, for example, **0 1 2**.*
3. *Set the **--channel-mean-value** parameter to the same value as that set during quantization. For details, see Step 2.*
4. *Currently, **--optimize** is set to **VIPNANOQI_PID0X88**. Its value range is as follows:*

   VIPNANOQI_PID0X7D, VIPNANOQI_PID0X88

   VIPNANOQI_PID0X99, VIPNANOQI_PID0XA1

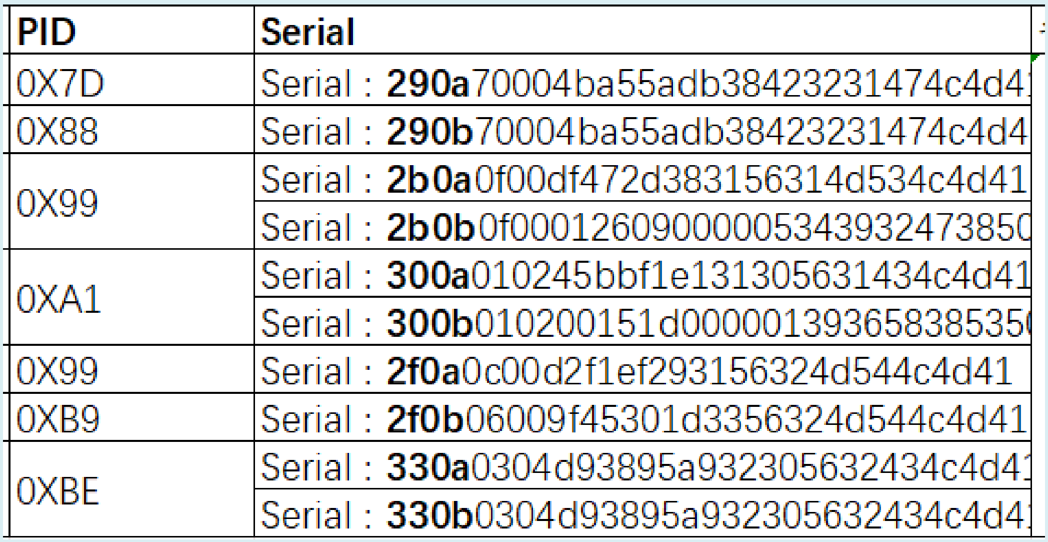
   VIPNANOQI_PID0XB9, VIPNANOQI_PID0XBE

   You can use the following method to confirm the values the value that should be set,

   Execute: cat /proc/cpuinfo

Check the first four characters of the serial code corresponding to Serial in the penultimate line (the part in bold after Serial)

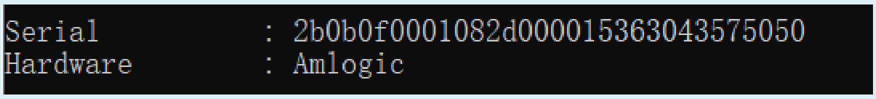
The mapping between Serial parameter `VIPNANOQI_PIDOX? ?` *and the corresponding relationship are shown in the figure below.*



PID	Serial
0X7D	Serial : **290a**70004ba55adb38423231474c4d4
0X88	Serial : **290b**70004ba55adb38423231474c4d4
0X99	Serial : **2b0a**0f00df472d383156314d534c4d41
	Serial : **2b0b**0f0001260900000534393247385
0XA1	Serial : **300a**010245bbf1e131305631434c4d41
	Serial : **300b**010200151d00000139365838535
0X99	Serial : **2f0a**0c00d2f1ef293156324d544c4d41
0XB9	Serial : **2f0b**06009f45301d3356324d544c4d41
0XBE	Serial : **330a**0304d93895a932305632434c4d4
	Serial : **330b**0304d93895a932305632434c4d4

For example:

After executing the command cat / proc / cpuinfo on the serial port or the CMD window, the serial is shown in the figure below.



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

At this point, the parameter should be set to: `--optimize VIPNANOQI_PID0X99`

5. ***--viv-sdk*** *depends on the sdk package, which is stored in the* `acuity_tool_xxx/`***bin\vcmdtools*** *directory. You can enter its relative path.*
6. ***--pack-nbg-unify*** *is used to generate a nbg file.*
7. *After the parameters described in 4, 5, and 6 are set, case demo code for the nbg-type model is generated. We usually use the nbg case demo. At this time, the normal case demo will also be generated in the model directory. Execute the following two lines of commands to organize the normal case into one directory:*

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

The difference between the two commands:

Normal_case: *When loading the model, online compilation may takes a long time. Android platform supports running normal case directly, but Linux platform does not. If Linux platform is used, you need to push* `acuity_tool_xx/bin/vcmdtoos` *directory to the board data directory, and then set the environment variable:*`export VIVANTE_SDK_DIR=/data/vcmdtoos.` *Then it can run normally.*

NBG case: *This step of on-line compilation has been completed on the PC. nb file can be loaded directly on the board, and the model loading speed is fast.*

8. *After the nbg-type model is successfully transcoded, there will be a* ***network_binary.nb*** *file in the case code directory.*

9. *Conversion_scripts is a model conversion directory, and conversion_scripts_nbg_unify will be generated in the same level directory. This is the nbg case demo directory.*

Introduction to the generated case code

The generated code is stored in the **nbg_unify** directory. Details are as follows:

1. **network_binary.nb** is the generated nbg file, which stores the weight and graph of the model. You can change the file name as needed.
2. **vnn_inceptionv4.c** is the code file used for model creation and release.
3. **vnn_pre_process.c** is the pre-processing model file, where read images are quantized to 8-bit data. Data about interfaces can be duplicated, or the model can be quantized itself.
4. **vnn_post_process.c** is the post-processing model file. The current transcoded case code only experiences top5 processing, and you can add post-processing to the model as needed.

Note

1. *Check the logs for Step 3 to confirm which files are generated. For example, a **mbox_priorbox_185.bin** file is generated for caffe ssd networks.*
2. *The demo for model transcoding of conversion_scripts is provided in the tool package. The execution sequence for the scripts is as follows: 0_import_model.sh -> 1_quantize_model.sh -> 2_export_case_code.sh.*
3. *After Step 3, you can transcode the mobilenet_v1 model in the demo to case code.*
4. *For the models that are transcoded themselves, place the models in the directory and modify the parameters in the sh scripts.*
5. *The **python extractoutput.py xxx.json** command in **extractoutput.py** of the demo can generate **outnamelist.txt**, which records mapping between **Blob name** and **tensor ID**. You can use this as references when obtaining transcoding results.*

Step 4 Inference on the PC side

```
./bin/tensorzonex    --action inference \
                  --source text    \
                        --source-file test.txt \
--channel-mean-value '127.5 127.5 127.5 128'        \
--model-input xxx.json \
--model-data xxx.data \

--dtype quantized
```

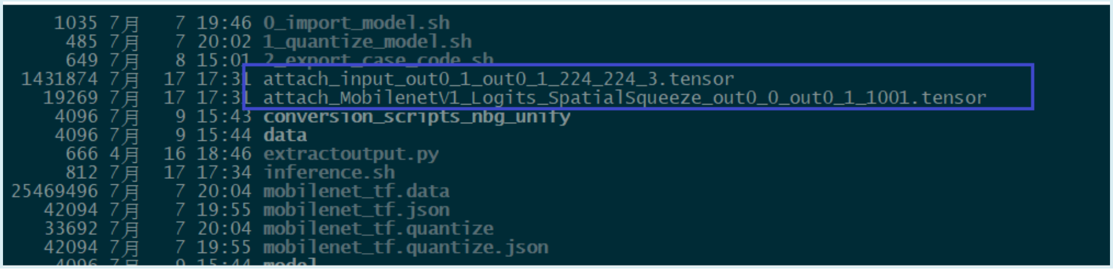
Note

1. The picture used when confirming the accuracy is generally stored in test.txt file.
2. Executing step 4 after executing step 1 importing model gets the result of non quantitative model of PC inference. That is float result.
3. Executing step 4 after executing step 2 quantifying model gets the result of quantitative model of PC inference. If a quantization operation has been performed, there is a `xxx.quantize` file. You can set the parameter `--dtype float32` to infer the result of float.
4. --channel-mean-value the value is same as the value set during quantization.
5. After execution of inference, the tensor file corresponding to input and output will be saved in the current directory. The input file is the float data after preprocessing.

The output file is the data hat has been inversed quantization to float. In general, it is recommended to use the tensor file saved by inference as the input file when you confirm the accuracy problem of board running time.

For example:

After the Mobilenet model executes the inference, the tensor file of input and output is saved. You can use the input corresponding to tensor file `attach_input_out0_1_out0_1_224_224_3. tensor` as input file of demo execution on board.



```
    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
```

3.5 Extended Parameter Description

3.5.1 Introduction

The parameters in the process of the model transcoding steps introduced in 3.4 are the most concise usage parameters currently organized. In general, only the above parameters can be used to convert the case demo normally. At present, some customers may have more needs and need to know the other parameters corresponding to the quantization and export commands. This chapter expands all the parameters of the quantization command tensorzonex and the export command ovxgenerator command for use.

3.5.2 Tensorzonex Extended Parameters

This command line tool (tensorzonex / tensorzonex.py) can be used for cropping, verification and other operations. All parameters are optional, and some parameters are necessary to perform specific tasks. Allowed parameters are shown in bold. This command-line tool is used for model quantization and PC-side inference operations.

```
  tensorzonex [-h] [-h] [--action ACTION] [--debug] [--dtype DTYPE]
                          [--device DEVICE] --model-input MODEL_INPUT
                          [--model-data MODEL_DATA]
                          [--model-quantize MODEL_QUANTIZE]
                          [--model-data-format MODEL_DATA_FORMAT]
                          [--validation-output VALIDATION_OUTPUT]
                          [--source SOURCE] [--source-file SOURCE_FILE]
                          [--restart] [--batch-size BATCH_SIZE]
                          [--samples SAMPLES] [--config CONFIG]
                          [--output-num OUTPUT_NUM] [--data-output DATA_OUTPUT]
                          [--epochs EPOCHS] [--optimizer OPTIMIZER] [--lr LR]
                          [--epochs-per-decay EPOCHS_PER_DECAY]
                          [--quantized-dtype QUANTIZED_DTYPE]
                          [--quantized-moving-alpha QUANTIZED_MOVING_ALPHA]
                          [--quantized-algorithm QUANTIZED_ALGORITHM]
```

```
[--quantized-divergence-nbins QUANTIZED_DIVERGENCE_NBINS]
[--quantized-rebuild] [--quantized-rebuild-all]
[--quantized-hybrid] [--reorder-channel REORDER_CHANNEL]
[--input-fitting INPUT_FITTING]
[--input-normalization INPUT_NORMALIZATION]
[--channel-mean-value CHANNEL_MEAN_VALUE]
[--mean-file MEAN_FILE]
[--caffe-mean-file CAFFE_MEAN_FILE] [--random-crop]
[--random-mirror] [--random-flip]
[--random-contrast RANDOM_CONTRAST]
[--random-brightness RANDOM_BRIGHTNESS] [--force-gray]
[--task TASK] [--prune-epochs PRUNE_EPOCHS]
[--prune-loss PRUNE_LOSS] [--pfps-epochs PFPS_EPOCHS]
[--pfps-reduce-target PFPS_REDUCE_TARGET]
[--pfps-delta0 PFPS_DELTA0]
[--without-update-masked-grad]
[--capture-format CAPTURE_FORMAT] [--capture-quantized]
[--output-dir OUTPUT_DIR] [--pb-name PB_NAME]
```

Arguments:

General Purpose:

-h help file

--action What to do for the network.(Optional)

The following values are valid for the related activity:

inference - for inference model

measure - for calcuate MACCs and PARAMs

prune - for iterative pruning and retraining

quantization - for Quantization

snapshot - for dump result layer by layer

test - for testing(Default)

train - for training

--batch-size Integer value which specifies the batch size (number of images in a batch). (Optional)

100 (Default).

CAUTION: Set this value smaller if the RAM or GPU cannot support such batch size.

--caffe-mean-file Input caffe mean file. Default is None. Example file:~/IMAGENET_1300_VAL/imagenet_mean.binaryproto

--config Reserved

--debug Set full debug model on.Produces additonal debug output.

Default is off.

--device Sepecify the compute device:

gpu:0 /cpu:0

if more than one GPU is present, specify by incrementing the number e.g, gpu:2.Note:The current tool is cpu,can’t use gpu.

--dtype Data type used for calculation.(Optional)

Allowed values are:

float32 - for float32 networks(Use this value to speed quantize.Default)

quantized - for quantized networks.

--epochs Integer value specifying the number of batches of images
used to train the model in every PFP-S interation. Each epoch
uses

The entire training set. Default is 1

--lr Learning rate in float format.Default is **0.1.** Required when **ACTION**=train

--mean-file Input mean patch and filename.Default is None.

--model-data Neural Network coefficient data input path and filename. Default uses the same patch and replaces the .json of MODEL_INPUT with .data.Rqquired when **ACTION**=quantization or train.

--model- Neural Network model data input format file type.

data-format Valid options are: **zone**(Default) or **acuity**

--model-input Neural Network model data input path and filename.

Required when **ACTION**=prune,quantization,train,inference,

test,snapshot,or validate.

--restart When training,do not load the coefficients and retrain.Use this option if you want to start from scratch.

--samples Sample total size.Default is -1.

--source Dataset source type.Currently supported:

sqlite - the source file has a .dsx extension and is a self-contained set of images. The file size will likely be large.

text - the source file has a .txt extension and contains a list of images,one per line,and additional parameters for the image which together make a dataset.

Required when **ACTION**=prune,quantization or train.

--source-file Dataset path and filename. The file extension indicates the dataset:

.dsx (implies source is **sqlite**)

.txt(implies source is a comma delimited **text** file)

Required when **ACTION**=prune,quantization or train.

--validation-output Validation output file.Default is in the tensorzonex binary floder with the name of “va;odatopm.csv”

--pb-name Save Graph and Coefficients in Tensorflow PB format.

Inference Argument:

--output-num Number of outputs(integer value). Default is 5.Only valid for Use **ACTION**=inference.

Quantization Argument:

--model-quantize Neural Network quantized tensors’ description file(Optional)
A *.quantize file. If not specified,the default uses the same path as MODEL_INPUT and replace the .json with .quantize
Please specify this parameter if EXPORT_DTYPE is specified as “quantized”

--quantized-dtype Data type used for quantized.(Optional)
Valid **dtype** values are:
asymmetric_affine-u8 - asymmetric affine unsigned8bit
dynamic_fixed_point-i8 – dynamic fixed point signed 8bit
dynamic_fixed_point-i16 –ynamic fixed point signed 16bit
Default is u8

--quantized-algorithm Quantization algorithm(Optional)
Valid values are:
kl_divergence
moving_average
n**ormal**(default)

--quantized-moving-alpha If **QUANTIZED_ALGORITHM** is specified as "moving_average", please set this parameter

--quantized-divergence-nbins If **QUANTIZED_ALGORITHM** is specified as "kl_divergence", please set this parameter

--quantized-rebuild Rebuild quantize table containing default specified tensors, mutually exclusive with argument – **quantized-rebuild-all, -- quantized-hybird**

--quantized-rebuild-all Rebuild quantize table containing all tensors, mutually exclusive with argument **--quantized-rebuild, - -quantized-hybrid**

--quantized-hybrid Setup a hybrid quantize network by quantize table.

mutually exclusive with argument **--quantized-rebuild, --quantized-rebuild-all**

This parameter needs to be specified if you want to generate a hybrid mode application

Transformation Arguments:

--channel-mean-value input channel mean value

--force-gray Force channel to gray.Enabled by default.

--input-fittling How to fit the image input to the network.

Crop -crop or pad the image.

Scale -scale the image(Default)

--input-Normalization Normalization method for input into the network:**image,pixel**,or **None**.Default is **None**.

--random-brightness Augment training images with randomized brightness (provide max delta) Format type is float.

--random-contrast Augment training images with randomized contrast (provide min-max contrast).

--random-crop Augment the training images with randomized crop.

--random-flip Augment training images with randomized flip (vertical). Enabled by default.

--random-mirror Augment training images with randomized mirroring (horizontal). Enabled by default.

--reorder-channel Use the same channel reorder value as used for network train/test/inference to change channel to BGR or RGB. A string value: Use "**2 1 0**" for BGR,"**0 1 2**" for RGB

Snapshot Only

Arguments:

--capture-format Define snapshot tensor file data sequence.
n**chw** (Default) , **nhwc**

--capture-quantized Define whether or not snapshot tensor file data should be in quantized format.If this parameter is specified, data will be saved to quantized format instead of float.

--output-dir Snapshot data file store path. If not specified, default is storage path is folder ‘./snapshot/’ .

Train Only Arguments:

--epochs-per-decay Integer value indicating the number of epochs required for next decay for learning rate decaying algorithms. Default is 100. Only valid for use with ACTION=train.

--data-output Network coefficient data output file path and filename where trained data should be saved. Default uses the location of the .json file and replaces the .json of MODEL_INPUT with .data. If not set, the original data (which is set using –model-data) is overwritten. Used only with **ACTION**=train, **ACTION**=prune.

--optimizer Stochastic Gradient Descent (SGD) Optimizer:
Static
Momentum (Default).
Only valid for use with **ACTION**=train

3.5.3 Ovxgenerator Extended Parameters

This command-line tool (ovxgenerator or ovxgenerator.py) takes the input of the model and data and converts it into an application that can be built and run using ovxlib. This command will be used in the step of exporting the case demo.

```
ovxgenerator    [-h] --model-input MODEL_INPUT --data-input DATA_INPUT
                    [--model-quantize MODEL_QUANTIZE]
                    [--model-output MODEL_OUTPUT] [--optimize OPTIMIZE]
                    [--export-dtype EXPORT_DTYPE]
                    [--channel-mean-value CHANNEL_MEAN_VALUE]
                    [--reorder-channel REORDER_CHANNEL]
                    [--save-fused-graph] [--pack-nbg-unify]
                    [--pack-nbg-viplite] [--viv-sdk VIV_SDK]
                    [--build-platform BUILD_PLATFORM]
                    [--target-ide-project TARGET_IDE_PROJECT]
                    [--batch-size BATCH_SIZE] [--force-remove-permute]
```

Required Arguments

--data-input Neural Network coefficient data input filename. Required.

--model-input Neural Network model input filename. Required.

Optional Arguments

-h Show help test on the console

--model-quantize Neural Network quantized tensors’ description file. (Optional).A *.quantize file. If not specified, the default uses the same path as MODEL_INPUT and replaces the .json with .quantize. Please specify this parameter if EXPORT_DTYPE is specified as ‘quantized’

--model-output Neural Network model output name. (Optional) If not specified, the default uses the network name as the model output name.

--optimize Optimize the exporting network according to the specified hardware configuration path or configuration name. (Optional).

none – no optimization

Default (Default) - If a configuration file or configuration name is not specified, it will use default export rule to export application code.If --pack-* is given, please specify a configuration file path or configuration name instead of **none** or **Default**.

--export-dtype Export case to given data type. (Optional)

String values in [‘float’, ‘quantized’]

float (Default) - specify to export float16 case.

quantized - specify to export quantized case.

--channel-mean-Value Input channel mean value. (Optional)
This parameter should always be given according to running models.The same channel mean value would be used for network train/test/inference/snapshot/validate.

--reorder-channel Use the same channel reorder value as used for network train/test/inference/snapshot/validate to change the channel to BGR or RGB. (Optional) This is a string value and should always be given according to running models
Use "**2 1 0**" for BGR. Use **"0 1 2**" for RGB.
Default is "", which stands for do not do any reorder channel operation at all.

--save-fused-graph Whether to save fused graph. (Optional).A fused graph is a network description file which contains the network structure of exported application code. It is useful for debug.

--viv-sdk SDK dir, if one of --pack-* is given, please specify a folder containing the binary sdk of vSimulator.

--pack-nbg-unify Pack binary graph for unify driver.(Optional) mutually exclusive with other --pack-*.If this feature is enabled, two cases will be generated,a unify case and a nbg_unify case.
If no --pack-* is given, only the unify case will be generated.

--batch-size Batch size for exported application code. (Optional)
Integer value in[1,+nan] **1** (Default)

--force-remove-permutes Force remove the head permutation layers inserted by T2C and directly connected to input graph layers; tails permutated layers inserted by T2C and directly connected to graph output layers.(Optional) This parameter should only be specified for models in Tensorflow/TensorflowLite formats, because models in Tensorflow sequence would trigger T2C.
CAUTION: If specified, please make sure to feed the correct tensor data to the application. For example, a

Tensorflow network with input of [1,224,224,3](nhwc)
and meet the remove permute condition.
-If this parameter is not specified, the user should feed
tensor with shape [1,224,224,3](nhwc) to the application.
-If this parameter is specified, the user should feed the
tensor with shape [1,3,224,224](nchw) to application.
Similarly, corresponding transpositions are also needed
for every output layer that removed the permutations
in post-process code.

3.6 Hybrid Quantization

Hybrid quantization is to use different quantization methods for different layers of the same model according to accuracy. For example, the 8bit quantization effect is poor, and resulting in a large error in the finale result of the model, we can use a hybrid quantization method to set the corresponding layer to a different quantization method to improve accuracy.

Note:

To avoid misunderstanding, this chapter uses the original English introduction process directly.

3.6.1 Process Summary

1、Use original xxx.data xxx.json file to quantize network into asymmetric_affine-u8 dtype as usual with parameter '--quantized-rebuild', which results in an xxx.quantize file.

2、In the xxx.quantize file, add the layer names and their corresponding quantized_dtype (choose from dynamic_fixed_point-i8, dynamic_fixed_point-i16, asymmetric_affine-u8, float32) to be changed to customized_quantize_layers.

3、Use the xxx.quantize file generated in Step 2, and the original xxx.data xxx.json file to quantize the network into asymmetric_affine-u8 dtype with parameter '--quantized-hybrid'.

4、Use the xxx.data, xxx.quantize.json, and xxx.quantize files generated in Step 3 to export application code as usual. DATACONVERT layers will be inserted into the graph.

3.6.2 Operation Procedure

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

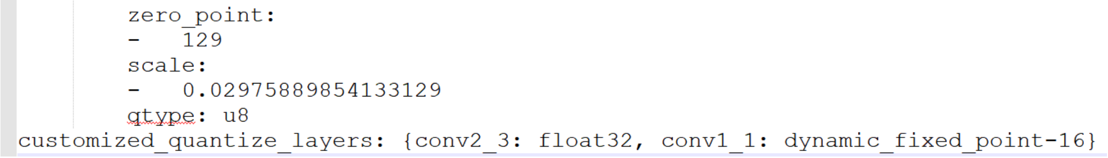
1、Quantize as usual to generate the lenet.quantize file:

```
$ ./tensorzonex --action quantization \
        --dtype float32 \
        --model-input ~/lenet/lenet.json \
        --model-data ~/lenet/lenet.data \
        --quantized-dtype asymmetric_affine-u8 \
        --quantized-rebuild \
        --source text \
        --source-file ~/lenet/dataset.txt \
        --reorder-channel '2 1 0' \
        --channel-mean-value '0 0 0 256':
```

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}
```

3、 Quantize the network into asymmetric_affine-u8 dtype again with parameter ‘--quantizedhybrid’. This will result in two new files lenet.quantize and lenet.quantize.json as well as some dtype_converter layers added into lenet.quantize.json.

```
$ ./tensorzonex --action quantization \
        --dtype float32 \
        --model-input ~/lenet/lenet.json \
        --model-data ~/lenet/lenet.data \
        --model-quantize ~/lenet/lenet.quantize
        --quantized-dtype asymmetric_affine-u8 \
        --quantized-hybrid \
        --source text \
        --source-file ~/lenet/dataset.txt \
        --reorder-channel '2 1 0' \
        --channel-mean-value '0 0 0 256'
```

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.

```
$ ./ovxgenerator --model-input lenet.quantize.json \
        --data-input lenet.data \
        --model-quantize lenet.quantize
        --export-dtype quantized \
        --channel-mean-value '0 0 0 256' \
        --optimize VIPNANOQI_PID0X88   \
        --viv-sdk ../bin/vcmdtools \
        --pack-nbg-unify

Note: --optimize VIPNANOQI_PID0X88   Refer to chapter 3.4 step3
```

3.7 Multi-input Model Transformation

3.7.1 Introduction

In the conversion process, the difference between the multi-input model and the single-input model is that there will be an extra step to assign the quantized data set to multiple inputs, so that it can be quantified normally.
The multi input model will take another step in the transformation process. "generate inputmeta" will generate a xxx_inputmeta.yml file. You can set different input quantitative data sets, parameters of mean value by this file.
In the generate inputmeta step, a xxx_inputmeta.yml file will be generated, through which the quantitative data sets and mean parameters of different inputs can be set.
This chapter introduces a super command line tool: Pegasus. It integrates all the features described in sections 3.4 - 3.5, with some additional features added. The model with single-input of four dimensions can be transformed following section 3.4 or Pegasus, but multi-input model can only be transformed by Pegasus.

3.7.2 Operation Procedure

Step 1 Import Model

```
Caffe:
./bin/pegasus   import caffe \
--model     ./model_lw3.prototxt \
--weights model_lw3.caffemodel \
--output-model ./model_lw3.json \
--output-data ./model_lw3.data

ONNX:
./bin/pegasus   import onnx \
--model     ./model_UnetBased_0620v8-ep44-seg3ch.onnx \
--inputs "0 1 2" \
--input-size-list "288,288,3#288,288,3#288,288,3" \
--outputs "327 328" \
--output-model ./${NAME}.json \
--output-data ./${NAME}.data
```

Note:

The models(caffe/tensorflow/keras/tflite/onnx/pytorch) that supported by the current tool can be imported through this script.Only three models are listed above, Other framework models can be imported by referring to chapter 3.7.3

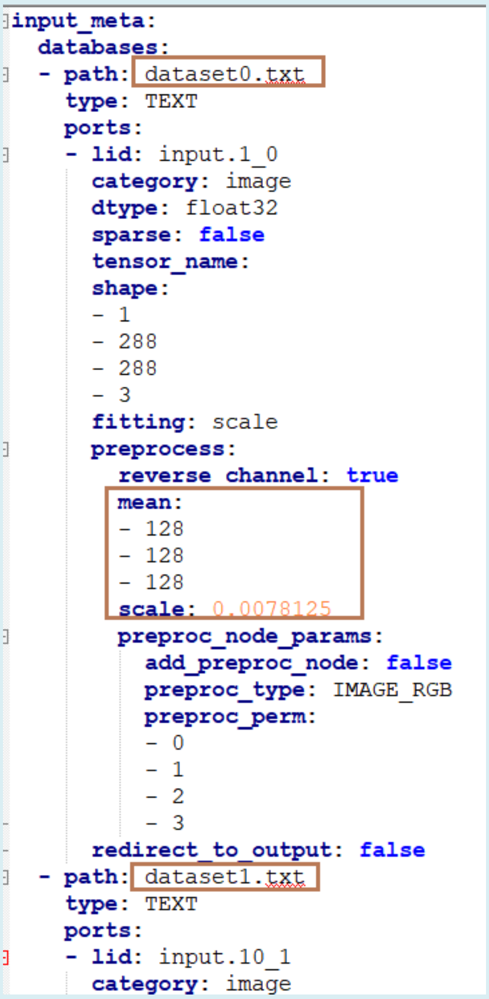
Step 2 Generate inputmeta

```
./bin/pegasus generate inputmeta --model ./${NAME}.json   \
--separated-database True
```

Result: generated${NAME}_inputmeta.yml

Note:

- The default value of the parameter **--separated-database** is false. There is only one dataset.txt path in the generated xxx_inputmeta.yml file. This setting is suitable for the case where the size of multiple input tensors is the same. There are multiple images in the dataset.txt line, separated by spaces: echo "space_shuttle_224x224.jpg goldfish_224x224.jpg" > dataset.txt
- When **--separated-database** is set to true, each input in the generated xxx_inputmeta.yml file has a corresponding dataset.txt file, which contains different input quantitative data separately. For example, if the tensor size of multiple inputs is different, it needs to be set to true.



```
input_meta:
  databases:
  - path: dataset0.txt
    type: TEXT
    ports:
    - lid: input.1_0
      category: image
      dtype: float32
      sparse: false
      tensor_name:
      shape:
      - 1
      - 288
      - 288
      - 3
      fitting: scale
      preprocess:
        reverse channel: true
        mean:
        - 128
        - 128
        - 128
        scale: 0.0078125
        preproc_node_params:
          add_preproc_node: false
          preproc_type: IMAGE_RGB
          preproc_perm:
          - 0
          - 1
          - 2
          - 3
      redirect_to_output: false
  - path: dataset1.txt
    type: TEXT
    ports:
    - lid: input.10_1
      category: image
```

- You can also set the mean value during the conversion process by changing the YML file. The mean value in the conversion process is also set by changing the yml file. As shown in the figure above, mean and scale correspond to the mean value setting. Unlike the previous section 3.4, the value of the decimal place is a float decimal, such as 1.0 / 128.

Step 3 Step3 Quantize Model

```
./bin/pegasus quantize \
--model ${NAME}.json \
--model-data ${NAME}.data \
--with-input-meta model_lw3_inputmeta.yml \
--quantizer asymmetric_quantized   \
--qtype uint8 --rebuild
```

Note:

- The quantization parameters are set in xxx_imputmeta.xml file.
- Refer to section 3.7.3 for other parameters.
- If the quantized data is multiple pictures, you can use --batch-size and --iterations to set, batch-size*iterations=number of pictures.

Step 4 Export Model:

```
./bin/pegasus  export ovxlib\
--model ${NAME}.json \
--model-data ${NAME}.data \
--model-quantize ${NAME}.quantize \
--with-input-meta model_lw3_inputmeta.yml \
--dtype quantized \
--optimize VIPNANOQI_PID0XB9   \
--viv-sdk ${ACUITY_PATH}vcmdtools \
--pack-nbg-unify
```

Note:

- Refer to step 3 in Section 3.4 for setting --optimize VIPNANOQI_PID0XB9.
- Refer to step 3 in Section 3.4 for setting --viv-sdk ${ACUITY_PATH}vcmdtools
- Refer to section 3.7.3 for other parameters

3.7.3 PASUS Extended Parameters

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

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

--input-size-list INPUT_SIZE_LIST  –outputs

[--size-with-batch SIZE_WITH_BATCH]
OUTPUTS [--mean-values MEAN_VALUES]
[--std-values STD_VALUES]
```

```
[--predef-file PREDEF_FILE]
[--subgraphs SUBGRAPHS]
[--output-model OUTPUT_MODEL]
[--output-data OUTPUT_DATA]

./bin/pegasus import tflite
[-h] --model MODEL [--output-model OUTPUT_MODEL]
[--output-data OUTPUT_DATA] [--outputs OUTPUTS]


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

./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]

./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]

./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]

./bin/pegasus generate inputmeta
[-h] [--input-meta-output INPUT_META_OUTPUT]
[--separated-database SEPARATED-DATABASE]
```

```
--model MODEL

./bin/pegasus quantize
[-h] --model MODEL --model-data MODEL_DATA
[--model-quantize MODEL_QUANTIZE]
[--batch-size BATCH_SIZE] [--iterations ITERATIONS]
[--device {GPU,CPU}]
[--with-input-meta WITH_INPUT_META]
[--quantizer {dynamic_fixed_point,asymmetric_quantized,asymmetric_affine,
symmetric_affine,perchannel_symmetric_affine}]
[--qtype QTYPE] [--hybrid] [--rebuild]
[--algorithm {normal,moving_average,kl_divergence}]
[--moving-average-weight MOVING_AVERAGE_WEIGHT]
[--divergence-nbins DIVERGENCE_NBINS]

./bin/pegasus export ovxlib
[-h] --model MODEL --model-data MODEL_DATA
[--model-quantize MODEL_QUANTIZE]
[--output-path OUTPUT_PATH]
[--with-input-meta WITH_INPUT_META]
[--optimize OPTIMIZE] [--dtype {float,quantized}]
[--save-fused-graph]
[--pack-nbg-unify] [--pack-nbg-viplite]
[--viv-sdk VIV_SDK]
[--build-platform {make}]
[--target-ide-project TARGET_IDE_PROJECT]
[--batch-size BATCH_SIZE]

./bin/pegasus inference
[-h] --model MODEL --model-data MODEL_DATA
[--model-quantize MODEL_QUANTIZE]
[--batch-size BATCH_SIZE] [--iterations ITERATIONS]
[--device {GPU,CPU}]
[--with-input-meta WITH_INPUT_META]
[--dtype {float32,quantized}]
[--postprocess {classification_classic,print_topn,dump_results}]
[--postprocess-file POSTPROCESS_FILE]
[--output-dir OUTPUT_DIR]
```

3.7.3.1 Pegasus Arguments Description (for gegasus arguments)

Import Caffe Arguments

-h Help file.

--model Caffe model filename. (Required)

--output-model Acuity net Neural Network layer set output file. (Optional).

If not specified, uses the path where the Caffe proto file is located, the output filename is the name defined in proto file.

--output-data Acuity net Neural Network coefficient data output file.

(Optional)

If not specified, default uses the path where the Caffe proto file is located, the output filename is the name defined in proto file.

--proto Switch protocol used for the Caffe binary protocol buffer file (binaryproto.caffemodel) associated with the CAFFE_MODEL. (Optional)

caffe – to import Caffe models from standard Caffe format protocol (Default) (http://caffe.berkeleyvision.org/)

lstm_caffe – use networks containing LSTM layer protocol.

CAUTION: To import non-standard Caffe models, please put relevant protocol file xxx_pb2.py into the same folder as caffe_pb2.py, and then specify this parameter as 'xxx' to import models.

--weights Caffe weights filename. (Optional)

Default assumes path and filename will be the same as

that of the **CAFFE_MODEL**. If the blob file does not

exist, random fake data will be generated.

Import Tensorflow Arguments

-h Help file.

--output-model Acuity net Neural Network layer set output file. (Optional)

If not specified, the default uses the path where the Tensorflow protobuf file is located, and the output filename is the same as the ptotobuf file.

CAUTION: For quant models, there will be an extra .quantize

file (quantize table) generated, after imported, DO NOT

quantize model again, just use imported model

inference/snapshot/export.

--output-data Acuity net Neural Network coefficient data output file.

(Optional)

If not specified, the default uses the path where the Tensorflow protobuf file is located, the output file name is the same as the ptotobuf file.

--model

Tensorflow frozen protobuf file. (Required)

CAUTION: Because every version of Tensorflow may have slight difference about APIs, use of 1.10.x to train and freeze protobuf file is highly recommended.

From test reports for a limited number of test cases, models rained and frozen by the following Tensorflow versions are known to work well: 1.4.x, 1.10.x, 1.13.x.

-- inputs

Input points of Tensorflow graph. (Required)

-If there is only one input point, enclose the input point name in quotation marks, such as “input_point”. -If there are two or more input points, use a space to separate each input point “input_point_1 input_point_2 input_point_3”.

--input-size-list

Input size list for corresponding input points. (Required)

- If there is only one input point, enclose the input point size list in quotation marks, and use a comma to separate the numbers in the size list, such as “224,224,3”.

-If there are two or more input points, use a hashtag to separate each input size, such as “224,224,3#299,299,3#12,12”.

CAUTION: the given input size of each input should or shouldn’t contain the batch size is determined by --size-with-batch.

--size-with-batch

Describes if the --input-size-list contains the highest batch dimension. (Optional)

None (Default) means --input-size-list should be given without batch.

Enclose the bool values in quotation marks, and use a comma to separate the bool values. Take a graph with two input layer as an example, the three input layers with shape “?x224x224x3” , “11x88” , “10” , input-size list and size with batch could be given as:

A:

--input-size-list “224,224,3#11,88#10”

--size-with-batch “False,True,True”

B:

--input-size-list “224,224,3#88#10”

--size-with-batch “False,False,True”

-- outputs

Output points of Tensorflow graph. (Required)

-If there is only one output point, enclose the output point name in quotation marks, such as “output_point” .

-If there are two or more output points, use a space to separate each output point, such as “output_point_1 output_point_2 output_point_3” .

--mean-values

Mean values for Tensorflow quant models. (Optional)

Needs to be provided when converting Tensorflow quant models. Comma-separated list of doubles, each entry in the list should match an entry in '--inputs'.

--std-values

Standard values for Tensorflow quant models. (Optional)

Needs to be provided when converting Tensorflow quant models. Comma-separated list of doubles, each entry in the list should match an entry in '--inputs'.

--predef-file

Pre-define file to import some complex models. (Optional)

To support some control logic, provide a npz format predef-file.

Use np.savez(\'prd.npz', [placeholder name]=predefine_value) to generate the file. If the filename contains characters which are not supported in, use ‘map’ as a keyword to store a dict to map it to normal key value.

Import TFlite Arguments

-h Help file.

--output-model Acuity net Neural Network layer set output file. (Optional)

If not specified, the default uses the path where the tflite model

file is located, and the output filename is the same as the tflite

model.

CAUTION: For quant models, there will be an extra .quantize

file (quantize table) generated, after imported, DO NOT

quantize model again, just use imported model

inference/snapshot/export.

--output-data Acuity net Neural Network coefficient data output file.(optional)

If not specified, the default uses the path where the tflite model

file is located, and the output filename is the same as the tflite

model.

--model Neural Network model input filename. (Required)

CAUTION: Acuity uses the tflite schema commits as in link:

https://github.com/tensorflow/tensorflow/commits/

master/tensorflow/lite/schema/schema.fbs

Commit hash:

0c4f5dfea4ceb3d7c0b46fc04828420a344f7598.

Because the tflite schema may not compatible with each other,

tflite models in older or newer schema may not be imported

successfully.

--outputs Output points of tflite graph. (Optional, available from 5.11.0)

-If there is only one output point, enclose the output point

name in quotation marks, such as “output_point”.

-If there are two or more output points, use a space to

separate each output point, such as “output_point_1

output_point_2 output_point_3”.

Import Darknet Arguments

-h Help file.

--model Darknet model filename. (Required)

--weights Darknet weights filename. (Required)

--output-model Acuity net Neural Network layer set output file. (Optional)

If not specified, the default uses the path where the darknet model file is located.

--output-data Acuity net Neural Network coefficient data output file.(optional)

If not specified, the default uses the path where the darknet model file is located.

Import Onnx Arguments

-h Help file.

--model model filename. (Required)

CAUTION: Supports onnx models in operator set 1-7.

-- inputs Input points of ONNX graph. (Optional)

--input-size-list Input size list for corresponding input points. (Optional)

None (Default): If there is only one input point, enclose the input point size list in quotation marks, and use a comma to separate the numbers in the size list, such as “224,224,3”. If there are two or more input points, use hashtag to separate each input size, such as “224,224,3#299,299,3#12,12”

CAUTION: whether the given input size of each input should or shouldn’t contain batch size is determined by --size-with-batch.

--size-with-batch Describe if the --input-size-list contain the highest batch dimension.(Optional, available from 5.11.0, March 2020)

None (Default): --input-size-list should be given without batch.Enclose the bool values in quotation marks, and use a hashtag to separate the bool values.

--input-dtype-list Input tensors dtype for corresponding input points.(Optional)

None (Default): Default input dtype is float. String values in [‘float’，‘int8’，’uint8’，’int16’，‘uint16’].

--output-data Acuity net Neural Network coefficient data output file.

(Optional) If not specified, the default uses the path where

The onnx model file is located.

--outputs Output points of ONNX graph. (Optional)

--output_model Acuity net Neural Network layer set output file. (Optional)

If not specified, the default uses the path where the onnx

model file is located.

Import Pytorch Arguments

-h Help file.

--model Pytorch model filename. The pytorch model must created

from torch.jit.trace(). (Optional)

CAUTION: Supports Pytorch 1.2

--inputs Model inputs.(Optional) Query from model (Default)

Only supports nodes which have one input tensor as input node.

--outputs Model outputs.(Optional) Query from model (Default).

Only supports nodes which have one output tensor as output node.

--input-size-list Input size list for corresponding input points. (Optional)

“None” (Default) If there is only one input point, enclose

the input point size list in quotation marks, and use a

comma to separate the numbers in the size list, such as

“3,224,224” . If there are two or more input points, use a

hashtag to separate each input size, such as

“3,224,224#3,299,299#12,12”

CAUTION: the given input size of each input should NOT

contain batch size.

--size-with-batch Describes if the --input-size-list contains the highest

batch dimension. (Optional)

None (Default) means --input-size-list should be given without batch.

--output-model Acuity net Neural Network layer set output file.(Optional)

If not specified, the default uses the path where the Pytorch model file is located, and the output filename is the same as the Pytorch model.

--output-data Acuity net Neural Network coefficient data output file.

If not specified, the default uses the path where the Pytorch model file is located, and the output filename is

the same as the Pytorch model.

--config A json file that describes the pytorch model information.

(Optional) It can convert pytorch model only by specifying

config file. Do not specify parameter ‘config’ if using

parameter ‘pytorch_model’ to convert pytorch model.

Import Keras Arguments

-h Help file.

--model Keras model filename. (Required)

CAUTION: Support Keras model generated by Tensorflow

1.13.2.

--convert-engine Convert engine for Keras model. (Optional)

String value in [‘Keras’, ‘TFLite’] Keras (Default)

--inputs Input points of Keras graph. (Optional)

-If there is only one input point, enclose the input point

name in quotation marks, such as “input_point”.

-If there are two or more input points, use a space to separate each input point “input_point_1 input_point_2

input_point_3”.

None (Default) means use all of the header node for import.

--input-size-list Input size list for corresponding input points. (Required)

None (Default) means let converter detect the input shapes. -If there is only one input point, enclose the input

point size list in quotation marks, and use a comma to

separate the numbers in the size list, such as “224,224,3”.

-If there are two or more input points, use a hashtag to separate each input size, such as

“224,224,3#299,299,3#12,12”.

CAUTION: the given input size of each input should NOT

contain the batch size.

--outputs Output points of Keras graph. (Optional)

None (Default) means use all of the tail node for import.

-If there is only one output point, enclose the output

point name in quotation marks, such as “output_point”.

-If there are two or more output points, use a space to separate each output point, such as “output_point_1 output_point_2 output_point_3” .

--output-model Acuity net Neural Network layer set output file. (Optional) If not specified, the default uses the path where the Keras model file is located, and the output filename is the same as the Keras model.

--output-data Acuity net Neural Network coefficient data output file. (Optional) If not specified, the default uses the path where the Keras model file is located, and the output filename is the same as the Keras model.

3.7.3.2 Generate Inputmeta

-h Help file.

--model Acuity model file input. (Required)

--input-meta-output Model input meta output path. (Optional) If not specified, put the generated input meta file in the same folder as Acuity model file.

--separated-database whether to generate more database for multi-inputs network. (Optional) False (Default) means generate one database in input_meta yaml file.

3.7.3.3 Quantize Arguments

-h Help file.

--model Network model input file. (Required)

--model-data Network coefficient input file. (Required)

--model-quantize Quantized tensor description file(quantize table), such as *.quantize etc. (Optional) Please specify this parameter if DTYPE is specified as ‘quantized’

--batch-size Integer value which specifies the batch size (number of images in a batch). (Optional) 100 (Default).
CAUTION: Set this value smaller if the RAM or GPU cannot support such a batch size.

--iterations Number of iterations to run, integer value. (Optional)

1 (Default) The iteration times for networks with cycle such as lstm, etc.

--device Specify the compute device. (Optional) String value in GPU or CPU.

--with-input-meta Merge input meta into MODEL. (Optional)

Input meta provides descriptions about every input layer’s dataset, pre-process, etc. -If there’ s no input meta information in MODEL, this parameter need to be pecified.

-If the input meta information is existed in MODEL, and this parameter is specified, use the specified input meta to override.

--quantizer Quantizer type. (Optional) Specify the quantizer to quantize network tensors. String value options are:

asymmetric_affine (Default)

dynamic_fixed_point

perchannel_symmetric_affine

symmetric_affine

asymmetric_quantized (will be deprecated, use asymmetric_affine instead)

CAUTION: Quantizers without the prefix ‘perchannel_’ are per-layer quantizers. asymmetric_quantized will be deprecate in a future release. asymmetric_quantized and asymmetric_affine are equivalent, and asymmetric_affine now replaces asymmetric_quantized.

--qtype Quantizer data type. (Optional)

String value options are:

int8 (Default), **int16**, and **uint8**.

The following quantizer + qtype is supported:

asymmetric_affine + uint8

dynamic_fixed_point + int8/int16

perchannel_symmetric_affine + int8

symmetric_affine + int8

asymmetric_quantized + uint8

--hybrid Setup a hybrid quantize network. (Optional)

Mutually exclusive with －rebuild. In the hybrid quantize condition, please specify this parameter.

--rebuild (Optional) Rebuild quantize table with a default quantize rules.

Mutually exclusive with --hybrid. In most scenarios, this parameter should be given to let Acuity build a default quantize table.

--algorithm Quantization algorithm. (Optional) String value options are: **normal** (Default),**kl_divergence**, **moving_average.**

--moving-average-weight Moving average coefficient. (Optional) Positive float value. 0.01 (Default) Please specify this parameter if ALGORITHM is specified as ‘moving_average.’

--divergence-nbins KL divergence histogram nbins. (Optional) Positive integer value. **1024** (Default) Please specify this parameter if ALGORITHM is specified as ‘kl_divergence.’

3.7.3.4 Export Ovxlib Arguments

-h Help file.

--model Neural Network model input filename. (Required)

--model-data (Required) Neural Network coefficient data input filename.

--with-input-meta Merge input meta into MODEL. (Required)

Input meta contains the description about every input layer’ s dataset, pre-process etc. If there is no input meta information in MODEL, this parameter needs to be specified. If the input meta information already exists in MODEL and this parameter is specified, use this specified input meta to override.

--model-quantize Quantized tensor description file (quantize table), such as *.quantize, etc. (Optional) Please specify this parameter if DTYPE is specified as ‘quantized’

-- output_path output filename or path. (Optional)

--optimize Optimize the exporting network according to the specified hardware configuration path or configuration name. (Optional)

none － no optimization

Default (Default) - If a configuration file or configuration name is not specified, the default export rule will be used to export application code. If --pack-* is given, please specify a configuration file path or configuration name instead of none or

Default.

--dtype

Export case to given data type. (Optional)

String value options are:

float (Default) - specify this value to export a float16 case.

float16 - specify this value to export a float16 case.

float32 - specify this value to export a float32 case.

quantized - specify this value to export a quantized case.

--save-fused-graph

Whether to save fused graph. (Optional)

A fused graph is a network description file which contains the network structure of exported application code. It is useful at the debug level.

--pack-nbg-unify

Pack binary graph for unify driver. Mutually exclusive with other --pack-*. -If this feature is enabled, three cases will be generated: a unify case, a nbg_unify case and an openvx case. The openvx case is in the nbg_unify case folder, and tnbg_unify and openvx cases share the same .nb file. -If no --pack-* is specified, only the unify case will be generated.

CAUTION: DOES NOT support graph with CPU node.

--pack-nbg-viplite

Pack binary graph for VIPLite driver. (Optional)

Mutually exclusive with other --pack-*. -If this feature is enabled, two cases will be generated, a unify case and a nbg_viplite case. -If no --pack-* is specified, only the unify case will be generated.

CACUTION: DOES NOT support graph with CPU node.

--viv-sdk

SDK directory.

-If one --pack-* is given, please specify the folder which contains the binary sdk for vSimulator. If VivanteIDE is

installed, the SDK path may be something like /home/xxx/Verisilicon/VivanteIDEx.x.x/*cmdtools.

--build-platform (Optional)

Build platform for generating network cases for nbg.

Value options are:

make (Default), consistent with the chosen pack parameter

--pack-*.

--target-ideproject Target IDE environment for application code. (Optional)
Valid string values are:
linux64 (Default), **win32**

--batch-size Batch size for exported application code. (Optional)
Integer value in[1,+nan] **None** (Default)
CAUTION: if set ‘－batch-size’, using the parameter set,
otherwise set the shape[0] in the input_meta file as batch-size.

3.7.3.5 Inference Arguments

-h Help file.

--model Network model input file in Acuity format, such as *.json *.yml
etc. (Required)

--model-data Network coefficient input file in Acuity format, such as *data
etc. (Required)

--model-quantize Quantized tensor description file (quantize table), such as
*.quantize etc. (Optional) Please specify this parameter if
DTYPE is specified as ‘quantized’.

--batch-size Integer value which specifies the batch size (number of
images in a batch). (Optional) 100 (Default).
CAUTION: Set this value smaller if the RAM or GPU cannot
support such a batch size.

--iterations Running iterations, integer value. (Optional) 1 (Default)
The iteration times for networks with cycle such as lstm, etc.

--with-input-meta Merge input meta into MODEL. (Optional) Input meta is
description about every input layer’s dataset, pre-process,
etc. -If there is no input meta information in MODEL, this
parameter needs to be specified. -If the input meta information
exists in MODEL, and this parameter is specified, use the
specified input meta to override.

--dtype Data type used for inference. (Optional) String values in [‘float32’, ‘quantized’] **float32** (Default) - specify this value

to inference floa32 model. **quantized** - specify this value to

inference quantized model.

--postprocess Postprocess task. (Optional) Built-in post-process task value in

[‘print_topn’, ‘dump_result’,’ classification_classic’]

classification_classic(Default) ONLY ONE task could be

chosen at the same time. Mutually exclusive with

--postprocess-file Usage see 0

--postprocess-file Postprocess task configuration file. (Optional)

According to POSTPROCESS, the user needs to create a

yaml file manually or use pegasus to generate a postprocess-file to

define the output tensors’ post process tasks. One or more tasks

could be given at the same time. This argument is mutually

exclusive with －postprocess. For usage see Section 0

--output-dir Output directory of result files. (Optional)