君正®

Magik 开发平台

后量化使用指南

Date: Feb. 2022

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

君正®

深度神经网络开放平台

后量化使用指南

Release history

Date/Author	Revision	Change
Feb. 2022 czhang	1.0.1	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rightsis granted. Ingenic assumes no liability, provides no warranty either expressed or implied relatingto the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining orsupporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes tothis document without notice. Anyone relying on this documentation should contact Ingenic forthe current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10
Xibeiwang East Road, Haidian District, Beijing, China,
Tel: 86-10-56345000
Fax:86-10-56345001
Http: //www.ingenic.com

目录

1 简介

MagikTransformKit 工具支持对 onnx/tensorflow 格式模型进行后量化得到 magik 格式模型，该模型可在芯片端进行部署工作。

文档以 yolov5s 网络为基础讲解如何产生在芯片部署的 magik 格式模型，以及生成模型效果不理想时如何定位问题。

2 转换命令

```
./magik-transform-tools \
--framework onnx \
--target_device T40 \
--outputpath yolov5s_magik.mk.h \
--inputpath ./yolov5s.onnx \
--mean 0,0,0 \
--var 255.0,255.0,255.0 \
--img_width 640 \
--img_height 640 \
--img_channel 3 \
--input_nodes images \
--output_nodes 640,660,680 \
--post_training_quantization true \
--ptq_config_path ./ptq_yolov5_config.json
```

MagikTransformKit 工具通过命令行参数完成 onnx/tensorflow 模型后量化工作，同时支持训练量化模型转换工作，命令行参数分为通用部分，后量化部分。

2.1 通用的命令行参数

framework – 输入模型格式，现支持 onnx/tensorflow 格式模型，（caffe/mxnet/pytorch 格式模型可统一转换 onnx 格式模型）

target_device – 部署板子的型号

inputpath – 转换的输入模型

outputpath – 转换的输出，转换完成后会保存在当前目录下生成所需的.bin 文件

mean – 训练时的预处理的均值

var – 训练时的预处理的方差

img_width – 上板运行时的输入的宽

img_height – 上板运行时的输入的宽

img_channel – 上板运行时的输入的通道

input_nodes – 模型输入节点

output_node – 模型输出节点

2.2 后量化部分命令行参数

post_training_quantization – 转换工具后量化功能开关
ptq_config_path – 后量化部分解析 josn 文件路径

3 json 文件

json 文件两部分组成，MagikTransformKit 工具进行后量化工具的需要基本信息和调试模型的 debug 信息。

MagikTransformKit 工具生成后量化模型效果不理想时，可以通过 Json 文件添加 debug 信息进行调试工作。

3.1 json 的基本信息

format – 图片处理格式:RGB,BGR,GRAY

width – 网络输入宽

height – 网络输入高

path – 校准图片集目录路径

used_image_num – 校准图片的数量

feature_quantize_method – feature 量化方法:KL,ADMM,MSE

weight_quantize_method – 权重量化方法:MAX,ADMM

MagikTransformKit 工具拿到正确的命令行信息和 json 文件，可以生成 magik 格式后缀*.mk.h 后量化模型和后缀*.bin 后量化模型，可在板端进行部署工作，部署工作可参考 MagikTransformKit 使用手册。

```
[                    yolov5s_demo]$ls
magik_model_yolov5s_magik.mk.h  run.sh          yolov5s.onnx
magik-transform-tools           yolov5-20
ptq_yolov5_config.json          yolov5s_magik.bin
```

3.2 json 的 debug 选项(可选项)

```
{
    "format":"RGB",
    "width":640,
    "height":640,
    "path":"./yolov5-50",
    "used_image_num":2,
    "feature_quantize_method":"KL",
    "weight_quantize_method":"MAX_ABS",
    "feature_quantize_bits":8,
    "weight_quantize_bits":8,
    "debug": {
        "open":false,
        "debug_input_path":"./test_image/5050_1-112.jpg",
        "debug_output_dir":"./dump_data"
    },
    "bias_corr":{
        "weight_corr":false,
        "feature_corr":false,
        "vali_dir":"./validated_images",
        "corr_dir":"./test_image"
    },
    "weight_fake_quant": true,
    "feature_fake_quant": true,
    "skip_node_quant":[
    ],
    "skip_op_quant":[
    ]
}
```

feature_quantize_bits – feature 量化位宽，可选，默认 8 比特

weight_quantize_bits – 权重量化位宽，可选，默认 8 比特

debug – 定位量化效果差选项(详见 demo)，可选{

debug_input_path – debug 校准图片集路径

debug_output_dir – debug 结果存放路径

}

weight_fake_quant – 模拟权重量化效果，可选

feature_fake_quant – 模拟 feature 量化效果，可选

bias_corr – 微调网络 bias 信息，可选 {

weight_corr – 权重伪量化微调 bias

feature_corr – feature 伪量化微调 bias

vali_dir – 验证图片集目录路径

corr_dir – 微调图片集目录路径

}

skip_node_quant – 跳过指定算子名量化，可选

skip_op_quant – 跳过指定算子类型量化，可选

4 yolov5 后量化 demo

demo 路径信息：***your_root/magik-toolkit/Models/T40/post/yolov5s_demo***

4.1 模型准备工作

MagikTransformKit 工具直接支持 tensorflow/onnx 格式模型进行后量化工作，间接支持 Pytorch/mxnet/caffe 格式模型，这三类格式需要统一转换成 onnx 格式。

针对 caffe 格式模型，我们提供 caffe2onnx 工具进行转换工作(详情可见 caffe2onnx_tools)

```
(test) [xxxx@1xx.xx.xx.xx caffe2onnx_tools]$python caffe2onnx/caffe2onnx.py model/resnet18.prototxt model/resnet18.caffemodel -o resnet18.onnx
```

caffe2onnx 工具路径信息：***your_root/magik-toolkit/Models/T40/post/caffe2onnx_tools***

当前使用的 yolov5s 模型从百度网盘下载：

链接：https://pan.baidu.com/s/1JZEtjRnUs6XF-nUcgNy3PA

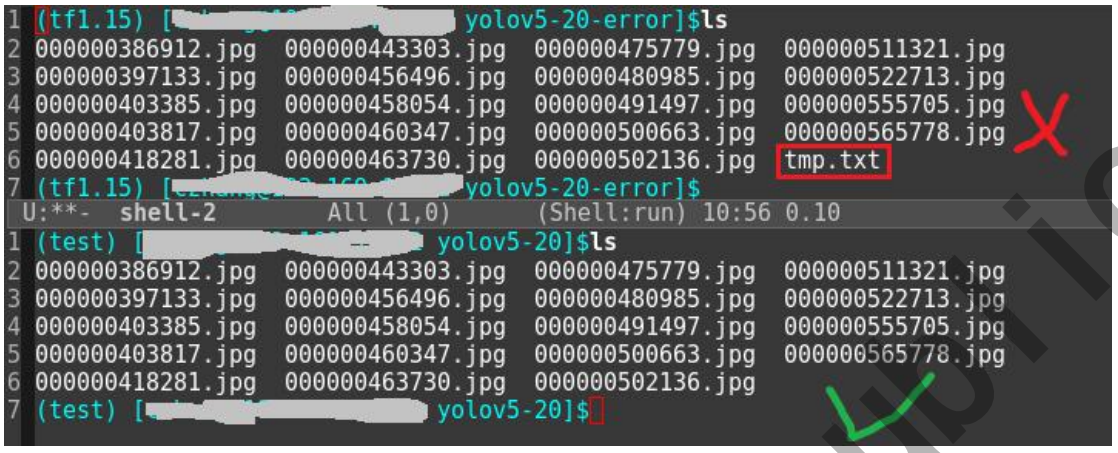
提取码：ad4v

模型下载至路径：***your_root/magik-toolkit/Models/T40/post/yolov5s_demo***

4.2 后量化校准数据准备

后量化过程校准图片通常在训练集或者验证集抽取，数目在 10~100 之间。

保证在校准图片目录下只有图片文件，不能有其他文件，否则量化过程会出错。



4.3 json 文件

1）保证 format/weight/height 参数与训练时保持一致

2）指定量化图片路径 path(共 19 张)和量化图片数量 12

3）常用量化组合 (feature:KL,weight:MAX_ABS)，量化效果不好可尝试(feature:MSE,weight:ADMM)组合

```
{
    "format":"RGB",
    "width":640,
    "height":640,
    "path":"./yolov5-20",
    "used_image_num":12,
    "feature_quantize_method":"KL",
    "weight_quantize_method":"MAX_ABS"
}
```

4.4 命令行书写

1）保证输出模型后缀 mk.h

2）设置 post_training_quantization 选项为真，保证 ptq_config_path 选项输入文件路径正确

```
./magik-transform-tools \
--framework onnx \
--target_device T40 \
--outputpath yolov5s_magik.mk.h \
--inputpath ./yolov5s.onnx \
--mean 0,0,0 \
--var 255.0,255.0,255.0 \
--img_width 640 \
--img_height 640 \
--img_channel 3 \
--input_nodes images \
--output_nodes 640,660,680 \
--post_training_quantization true \
--ptq_config_path ./ptq_yolov5_config.json
```

3）MagikTransformKit 工具对 yolov5 网络后量化过程正常进行，会生成上板的 yolov5s_magik.bin 文件

```
[ yolov5s_demo]$ls
magik_model_yolov5s_magik.mk.h  run.sh          yolov5s.onnx
magik-transform-tools           yolov5-20
ptq_yolov5_config.json          yolov5s_magik.bin
```

yolov5s.onnx – 待量化的模型文件

yolov5-20 – 后量化校准图片集目录

ptq_yolov5_config.json – 后量化使用 json 文件

run.sh – 转换命令的脚本

yolov5s_magik.bin – 后量化过程产生上板文件

magik_model_yolov5s_magik.mk.h – 后量化过程产生上板文件

5 模型上板

5.1 代码编译

在 ***your_root/magik-toolkit/Models/T40/post/yolov5s_demo/venus_sample_ptq_yolov5s*** 目录下我们还提供了上板运行需要的 inference.cpp，Makfile、测试数据，另外我们还需要用到 venus 库及 mips 编译工具。

其中，

venus 库在 ***your_root/magik-toolkit/InferenceKit/T40/venus*** 下；

mips 编译工具由方案同事提供。

5.1.1 网络输入

这里为了用户快速实现流程，我们在这里加入了 opencv 的库函数方便调用 imread 等函数，所以测试的时候直接传入 jpg 图即可。

当前使用的 opencv 的库从百度网盘下载：

链接：https://pan.baidu.com/s/1Nx7yf1gzNzXCJPTTbLtvJw

提取码：8rdl

opencv 的库下载至：

your_root/magik-toolkit/ThirdParty

5.1.2 模型的加载

提供的实例 inference.cpp 中模型是通过参数传入的，运行前注意同步拷贝到板端对应的目录并在运行时传入。

5.1.3 超参数的设置

```
void generateBBox(std::vector<venus::Tensor> out_res, std::vector<magik::venus::ObjBbox_t>& candidate_boxes,
int img_w, int img_h)
{
  float person_threshold = 0.3;
  int classes = 80;
  float nms_threshold = 0.6;
  std::vector<float> strides = {8.0, 16.0, 32.0};
  int box_num = 3;
  std::vector<float> anchor = {10,13,  16,30,  33,23, 30,61,  62,45,  59,119, 116,90,  156,198,  373,326};
```

strides 和 anchor 按 yolov5 的实际使用设置，person_threshold 和 nms_threshold 分别对应原始代码中的 conf-thres（置信度阈值）和 iou-thres（iou 阈值），classes 是类别数。

5.1.4 编译

TOPDIR – venus 库的相对目录

opencv_path – mips 编译后的 opencv 库的路径，拷贝也是从这里

libtype – 根据实际板子的需求确定是否 muclibc（这里以 muclibc 为例）

build_type – release 模式，运行得到结果，默认设置

– profile 模式，运行时网络结构可视化及网络每层运行时间及 GOPs 统计

– debug 模式，运行得到结果的同时保存每层量化 feature 的结果

– nmem 模式，统计模型运行时 nmem 内存占用情况，运行程序，内存使用情况保存在/tmp/nmem_memory.txt

直接 make 编译 inference.cpp 即可生成 venus_yolov5s_bin_uclibc_*，即我们上板需要的可执行文件。

注意：我们同时提供了用于实际板端运行的输入为 nv12 的代码用例 inference_nv12.cpp 及输入数据 10_w1024_h714.nv12，如有需要，可修改 Makefile 进行编译及使用测试。

5.2 上板运行

opencv 库路径：***magik-toolkit/ThirdParty/opencv_mips_7.2.0_2.26/uclibc/lib*** 下

venus 库路径：***magik-toolkit/InferenceKit/T40/venus/7.2.0/lib/uclibc*** 下

5.2.1 release(发布库)

编译: make build_type=release

在当前文件夹下生成 venus_yolov5s_bin_uclibc_release 可执行文件，拷贝 venus 库(libvenus.so)、opencv 库(libopencv_core.so.3.3, libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3,libopencv_videoio.so.3.3)、可执行文件(venus_yolov5s_bin_uclibc_release)、模型文件(yolov5s_magik.bin)、测试图片(fall_1054_sys.jpg)至开发板运行即可：

./venus_yolov5s_bin_uclibc_release yolov5s_magik.bin fall_1054_sys.jpg

注：运行前添加库路径至 LD_LIBRARY_PATH:

export LD_LIBRARY_PATH=$lib_path:$LD_LIBRARY_PATH

清除 **make build_type=release clean**

```
[root@Ingenic-uc1_1:venus_sample_ptq_yolov5s]# ./venus_yolov5s_bin_ptq_uclibc_release .
s./yolov5s_magik.bin fall_1054_sys.jpg
./venus_yolov5s_bin_ptq_uclibc_re
lease ../yolov5s_magik.bin fall_1054_sys.jpg
Warning : The version number is not obtained. Please upgrade the soc-nna!
INFO(magik): venus version:0.9.0(00000900_8450a71)  built:20220211-1206(7.2.0 glibc2.26
s mips@aie)
w:388 h:574
ori_image w,h: 388 ,574
model-->640 ,640 4
input shape:
-->384 640
scale---> 0.668990
resize padding over:
resize valid_dst, w:260 h 384
padding info top :0 bottom 0 left:190 right:190
pad_x:190 pad_y:0 scale:0.668990
box:   7 41 352 406 0.88
box:   97 329 380 510 0.71
```

5.2.2 debug（用于核对数据的库）

详见 Maigk 训练量化使用指南步骤 7.2，输入的处理有些不同。

5.2.3 profile（网络可视化及每层运行时间统计）

编译: make build_type=profile

在当前文件夹下生成 venus_yolov5s_bin_uclibc_profile 可执行文件，拷贝 venus 库(libvenus.p.so)、opencv 库(libopencv_core.so.3.3, libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3,libopencv_videoio.so.3.3)、可执行文件(venus_yolov5s_bin_uclibc_profile)、模型文件(yolov5s_magik.bin)、测试图片(fall_1054_sys.jpg)至开发板运行即可：

./venus_yolov5s_bin_uclibc_profile yolov5s_magik.bin fall_1054_sys.jpg

注：运行前添加库路径至 LD_LIBRARY_PATH:

export LD_LIBRARY_PATH=$lib_path:$LD_LIBRARY_PATH

清除

make build_type=profile clean

```
root@Ingenic-uc1_1:venus_sample_ptq_yolov5s]# ./venus_yolov5s_bin_ptq_uclibc_profile ../yolov5s_magik.bin fall_1054_sys.jpg
./venus_yolov5s_bin_ptq_uclibc_pr
ofile ../yolov5s_magik.bin fall_1054_sys.jpg
Warning : The version number is not obtained. Please upgrade the soc-nna!
INFO(magik): venus version:0.9.0(00000900_8450a71)  built:20220211-1107(7.2.0 glibc2.26 mips@aie)
w:388 h:574
ori_image w,h: 388 ,574
model-->640 ,640 4
input shape:
-->384 640
scale---> 0.668990
resize padding over:
resize valid_dst, w:260 h 384
padding info top :0 bottom 0 left:190 right:190
pad_x:190 pad_y:0 scale:0.668990
box:   7 41 352 406 0.88
box:   97 329 380 510 0.71
[I/magik::venus]:
Timing cycle = 10
===== Detailed DispatchProfiler Summary: N/A, Exclude 5warm-ups =====
LayerName            OperatorType        InputShape       OutputShape      FilterShape      Stride    Pad          Dila     Avg(ms) Max(ms) Min(ms) Last(ms) Avg(%)  GOPs    BandWidth(MB)
AppendInputOutputNOpOptimizer_input_nop_0_BatchNormScale Normalize           {1,384,640,4}    {1,384,640,4}    N/A                N/A      N/A      N/A      1.808   1.809   1.807   1.8070   2.82%   0.000   1.875
414_Quantize         Convolution         {1,384,640,4}    {1,192,320,32}   {3,3,4,32}       {1,1}     {1,1,1,1}    {1,1}    6.839   6.842   6.835   6.8400   10.66%  0.142   2.817
417_Quantize         Convolution         {1,192,320,32}   {1,96,160,64}    {3,3,32,64}      {2,2}     {1,1,1,1}    {1,1}    2.181   2.183   2.179   2.1830   3.40%   0.566   2.831
420_Quantize         Convolution         {1,96,160,64}    {1,96,160,32}    {1,1,64,32}      {1,1}     {0,0,0,0}    {1,1}    0.755   0.756   0.753   0.7540   1.18%   0.063   1.409
423_Quantize         Convolution         {1,96,160,32}    {1,96,160,32}    {1,1,32,32}      {1,1}     {0,0,0,0}    {1,1}    1.056   1.057   1.055   1.0550   1.65%   0.031   0.939
426_Quantize         Convolution         {1,96,160,32}    {1,96,160,32}    {3,3,32,32}      {1,1}     {1,1,1,1}    {1,1}    1.142   1.162   1.128   1.1290   1.78%   0.283   0.947
427                  Add                 {1,96,160,32}    {1,96,160,32}    N/A              N/A       N/A          N/A      0.609   0.611   0.608   0.6110   0.95%   0.000   1.406
432_0_Quantize       Convolution         {1,96,160,32}    {1,96,160,32}    {1,1,32,32}      {1,1}     {0,0,0,0}    {1,1}    1.093   1.096   1.090   1.0910   1.70%   0.031   0.939
432_1_Quantize       Convolution         {1,96,160,64}    {1,96,160,32}    {1,1,64,32}      {1,1}     {0,0,0,0}    {1,1}    0.755   0.757   0.754   0.7540   1.18%   0.063   1.409
432                  Concat              {1,96,160,64}    {1,96,160,64}    N/A              N/A       N/A          N/A      0.999   1.004   0.994   1.0000   1.56%   0.000   1.875
435_Quantize         Convolution         {1,96,160,64}    {1,96,160,64}    {1,1,64,64}      {1,1}     {0,0,0,0}    {1,1}    1.340   1.365   1.304   1.3610   2.09%   0.126   1.880
438_Quantize         Convolution         {1,96,160,64}    {1,48,80,128}    {3,3,64,128}     {2,2}     {1,1,1,1}    {1,1}    1.766   1.767   1.765   1.7660   2.75%   0.566   1.478
441_Quantize         Convolution         {1,48,80,128}    {1,48,80,64}     {1,1,128,64}     {1,1}     {0,0,0,0}    {1,1}    0.372   0.372   0.371   0.3720   0.58%   0.063   0.712
444_Quantize         Convolution         {1,48,80,64}     {1,48,80,64}     {1,1,64,64}      {1,1}     {0,0,0,0}    {1,1}    0.342   0.343   0.341   0.3430   0.53%   0.031   0.473
447_Quantize         Convolution         {1,48,80,64}     {1,48,80,64}     {3,3,64,64}      {1,1}     {1,1,1,1}    {1,1}    0.568   0.569   0.567   0.5680   0.89%   0.283   0.505
448                  Add                 {1,48,80,64}     {1,48,80,64}     N/A              N/A       N/A          N/A      0.337   0.340   0.335   0.3350   0.53%   0.000   0.703
```

5.2.4 nmem 模式（用于统计网络运行时 nmem 占用情况，保存在/tmp/nmem_memory.txt）

编译: make build_type=nmem

在当前文件夹下生成 venus_yolov5s_bin_uclibc_nmem 可执行文件，拷贝 venus 库(libvenus.m.so)、opencv 库(libopencv_core.so.3.3 ,libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3,libopencv_videoio.so.3.3)、可执行文件(venus_yolov5s_bin_uclibc_nmem)、模型文件(yolov5s_magik.bin)、测试图片(fall_1054_sys.jpg)至开发板运行即可：

./venus_yolov5s_bin_uclibc_nmem yolov5s_magik.bin fall_1054_sys.jpg

注：运行前添加库路径至 LD_LIBRARY_PATH:

export LD_LIBRARY_PATH=$lib_path:$LD_LIBRARY_PATH

清除

make build_type=nmem clean

```
[root@Ingenic-uc1_1:venus_sample_ptq_yolov5s]# ./venus_yolov5s_bin_ptq_uclibc_nmem ../yolov5s_magik.bin fall_1054_sys.jpg
./venus_yolov5s_bin_ptq_uclibc_nm
em ../yolov5s_magik.bin fall_1054_sys.jpg
Warning : The version number is not obtained. Please upgrade the soc-nna!
INFO(magik): venus version:0.9.0(00000900_8450a71)  built:20220211-1007(7.2.0 glibc2.26 mips@aie)
w:388 h:574
ori_image w,h: 388 ,574
model-->640 ,640 4
input shape:
-->384 640
scale---> 0.668990
resize padding over:
resize valid_dst, w:260 h 384
padding info top :0 bottom 0 left:190 right:190
pad_x:190 pad_y:0 scale:0.668990
box:   7 41 352 406 0.88
box:   97 329 380 510 0.71
```

6 数据核对

验证 PC 端和板端的数据是否能对齐，可按如下步骤操作：

6.1 PC 端

使用由 Ingenic 提供的 PC 端推理库，通过设置相关环境变量可以实现算子逐层数据保存成二进制文件与 SOC 开启对应模式之后的二进制文件进行比对，PC 端的具体操作步骤如下：

为了保持输入数据的高度一致，我们使用了 opencv 库作为图像解析工具，如果您是需要比对 python 与 pc 端数据结果一致性我们建议您在 python 端也使用 opencv 来作为解析图像的工具。所以我们整理了运行 pc 端工具所必要的库、模型放在了百度云上。

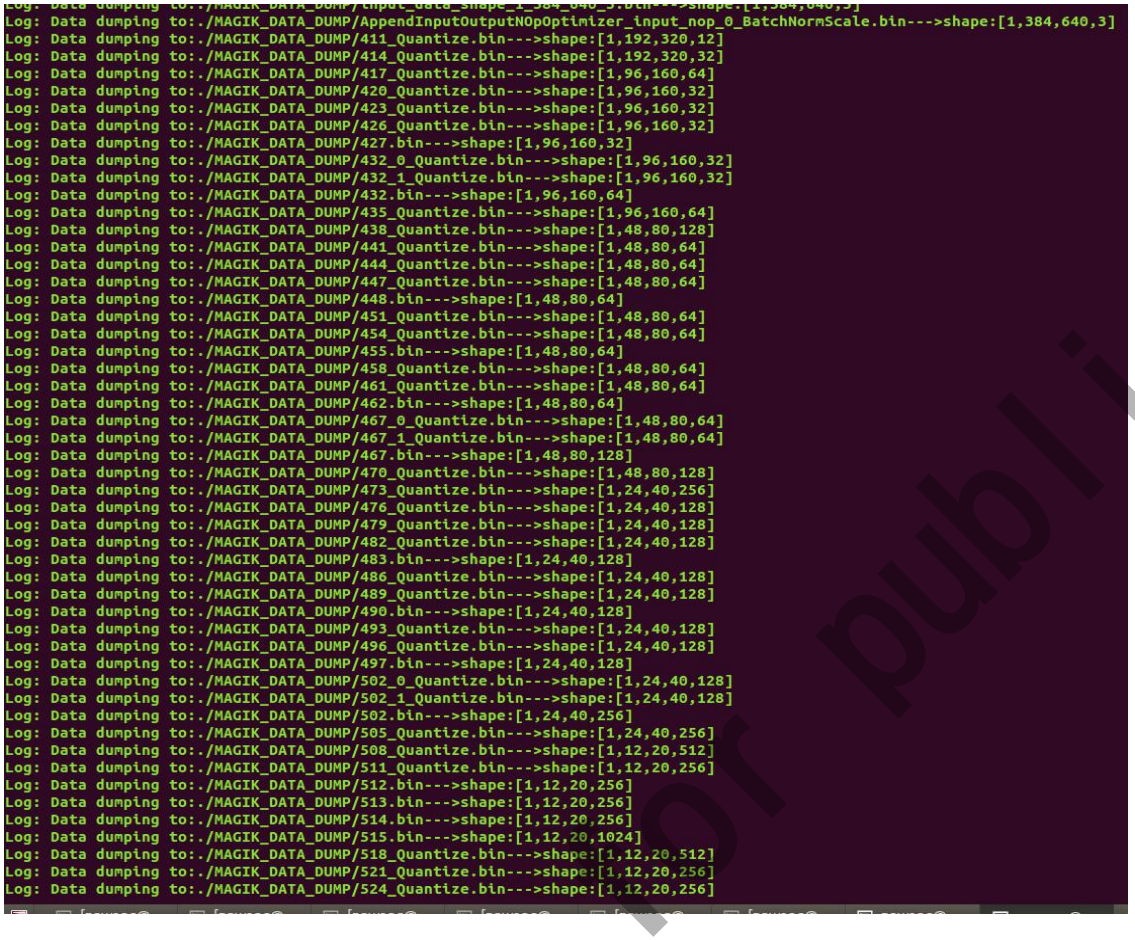
首先需要切换到***./git_magik-toolkit/Models/T40/post/pc_inference*** 文件夹下，在 PC 端通过设置环境变量

export MAGIK_CPP_DUMPDATA=true

可保存每层量化结果到./MAGIK_DATA_DUMP/文件夹下，具体运行命令为:

```
export   MAGIK_CPP_DUMPDATA=true
make   clean
make   -j8
./pc_inference_bin   ./model_quant.mgk   fall_1054_sys.jpg   640 384
```

上述命令中 model_quant.mgk 是转化工具转化模型时生成的.mgk 文件，fall_1054_sys.jpg 是输入的图片，640 ,384 分别对应输入的宽和高。最终保存./MAGIK_DATA_DUMP/下（具体见下图），可见输入层有保存为 input_data_shape_1_384_640_3.bin，后面的 shape 命名的规则是 n,h,w,c, 即高为 384，宽为 640，最后面是输出层；如果您不希望分步执行的话，可以运行我们提供的 **exector.sh** 脚本来一键运行。



6.2 板端

(1) 板端需要的输入数据为 4 通道，对于上面 pc 端测试的 fall_1054_sys.jpg，我们用运行时保存的输入 input_data_shape_1_384_640_3.bin 处理成.h 之后拿来做板端的测试，以保证二者的输入完全一致，可先做线下处理

(脚本在 yolov5s_demo/venus_sample_ptq_yolov5s/generate_img_input.py):

python generate_img_input.py bin_path w h c bin

bin_path -- 要转换的 bin 文件，即 pc 端运行保存的输入 bin

w, h -- bin 文件对应的宽和高，bin 文件名字里有对应，分别是 1_h_w_c

bin -- flag 标志，表示由 bin 文件转头文件，还有一种是 img 转头文件

说明：用 bin 的话可以完全保证两边的输入一致，方便对数据。

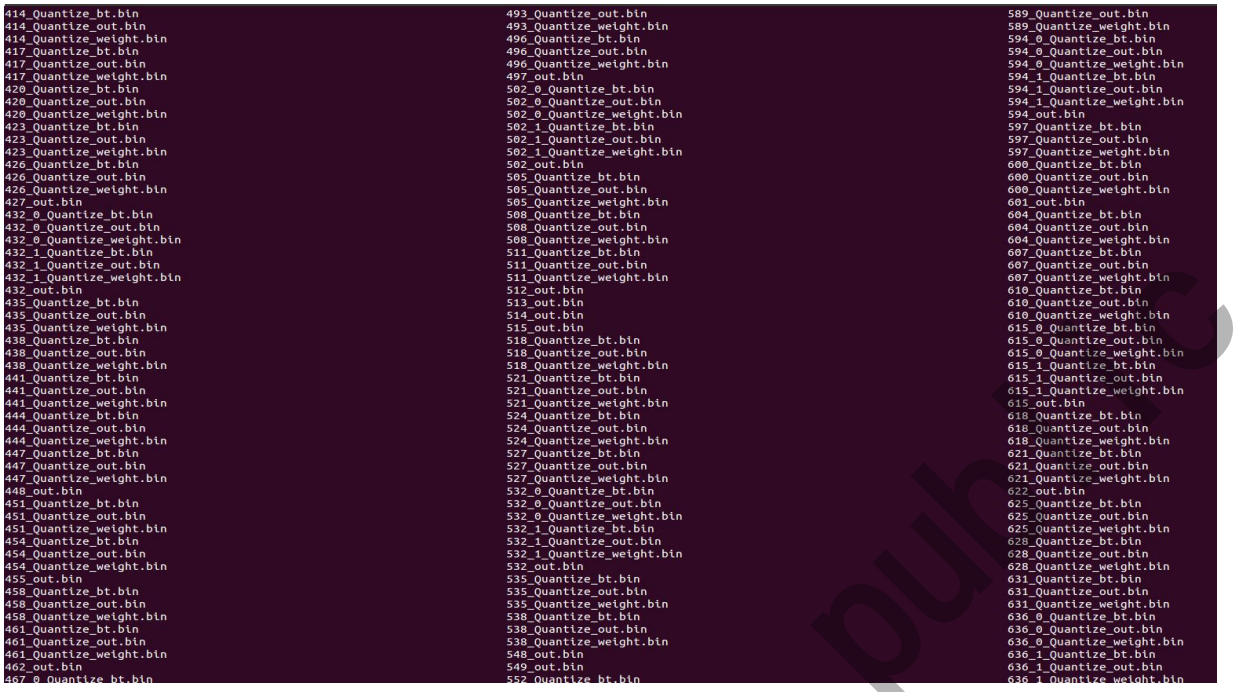
提供的实例中模型是通过参数传入的(argv[1])，运行时注意同步拷贝到板端对应的目录，运行时按需传入。宽(w)、高(h)也要从外部传入，具体值同(1)中。

编译 c 代码：make build_type=debug ，在当前文件夹下生成

venus_yolov5s_bin_ptq_uclibc_debug 可执行文件，拷贝 venus 库(libvenus.d.so)、可执行文件(venus_yolov5s_bin_ptq_uclibc_debug)、模型文件(yolov5s_magik.bin)至开发板运行：

./venus_yolov5s_bin_ptq_uclibc_debug yolov5s_magik.bin 640 383

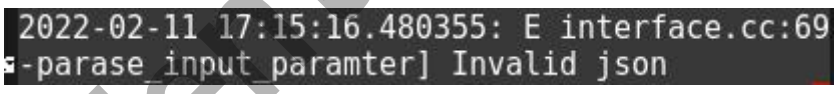
其中，640 对应为 w，383 对应 h，运行的时候就会自动保存每层的输出及其他信息（如下图）：



其中 _*_QuantizeFeature_out.bin 和 PC 端运行时保存在 ./MAGIK_DATA_DUMP/ 下的 _*_QuantizeFeature.bin
是一一对应的，直接核对 md5 值是否一致即可，在保证输入完全一致的情况下中间有层不对应及时反馈。

7 后量化过程常见问题

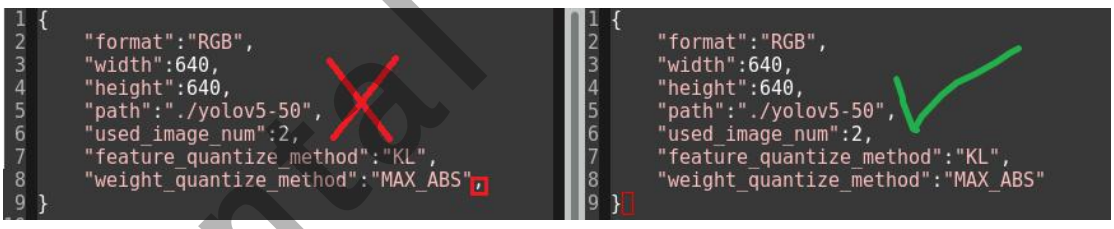
问：转换过程提示 invalid json



答：检查 josn 文件路径、命令是否正确

```
./magik-transform-tools \
--framework onnx \
--target_device T40 \
--outputpath yolov5s_magik.mk.h \
--inputpath ./yolov5s.onnx \
--mean 0,0,0 \
--var 255.0,255.0,255.0 \
--img_width 640 \
--img_height 640 \
--img_channel 3 \
--input_nodes images \
--output_nodes output \
--post_training_quantization true \
--ptq_config_path ./ptq_yolov5_config.jso
> > > > > > > > > > > > > INFO(magik): magik-
transform-tools version:1.0.0(00010000_fbdaa2
8)  cuda version:9.0.176  built:20220209-1705
_GPU
2022-02-11 17:15:09.308386: W clip_model_by_n
op_optimizer.cc:48-collect_remove_nodes] Mult
iple output NOp exists on the current pathway
!
2022-02-11 17:15:16.480355: E interface.cc:69
-parase_input_paramter] Invalid json
```

检查 json 格式是否正确



8 json 的 debug 过程

debug 选项是转换工具模拟网络原生模型（onnx 或 tf）推理和板端推理，保存每层算子推理结果、计算每一层算子误差（余弦相似度，欧式距离，绝对值最大值）。

网络进行后量化，精度损失在 0.05%~1%是可接受范围，通常余弦相似度表现：

A.第一层到最后一层的余弦相似度都是 99.0%+；

B.或者中间层存在少许 98.0%+，第一层和最后一层余弦相似度都是 99.0%+。

8.1 效果差的现象

使用转换工具的 json 的 debug 选项定位问题，现象可分为以下四种：

A.第一层算子 cosine_avg 没有达到 99.0%+

B.第一层量化达到 99.0%+，某层开始下降，最后一层余弦相似度低于 95.0%

C.第一层和中间层量化正常，最后几层量化余弦相似度下降

D.余弦相似度表现正常，板端验证效果差

8.2 解决方法

打算

对于以上总结现象信息，我们整理定位解决问题的步骤方法。

A.第一层算子 cosine_avg 没有达到 99.0%+或者 max_absolu 很大：

原因：网络输入存在问题，常规情况下，第一层算子 cosine_avg 大于 99%

解决：检查 json 文件中图片格式（BRG、RGB）、均值方差，保证这些参数和训练时保持一致

B.第一层量化达到 99.0%+，某层开始下降，最后一层余弦相似度低于 95.0%：

原因 1：量化带来误差，这种误差必然存在

解决：使用 feature_quantize_bit 或者 weight_quantize_bit 选项提升位宽(设置为 10、12、14，默认 8 比特)

原因 2：某层算子量化方法存在问题

解决：使用 skip_node_quant 选项跳过 cosine_avg 下降严重层，使用机制后，其他层余弦相似度正常

```
"skip_node_quant":[
    "node_name_0"
],
```

反馈转换工具对接人员，将"result/path/ptq_model"路径下和"result/path/float_model"路径下存在问题层的输入输出的 dump 结果，和问题层量化参数发送给对接人员

原因 3：模拟原生网络推理存在问题

转换工具模拟网络原生模型（onnx 或 tf）推理存在问题。使用转换工具生成的"./result/path/input_uint8.bin"数据作为原生模型推理的输入，保存原生模型推理的结果；和"result/path/float_model"路径存放模拟结果(float 二进制形式 NHWC)进行对比，确认是否模拟原生模型推理存在问题；

如果存在问题反馈对接人员，将"result/path/float_model"路径下问题层输入输出 bin 文件和正确输出的文件发送对接人员，修复问题

C.第一层和中间层余弦相似度正常，最后几层量化余弦相似度下降：

原因 1：参考现象 2）

原因 2：网络量化存在累积误差，逐层量化带来误差在某一层表现明显

解决：通过"weight_fake_quant"选项、"feature_fake_quant"选项确认是 weight 量化还是 feature 量化带来的误差，在使用"bias_corr"选项消除误差

```
"feature_fake_quant": true,
"weight_fake_quant": true,
"bias_corr":{
    "weight_corr":false,
    "feature_corr":false,
    "vali_dir":"./validated_images",
```

"corr_dir":"./test_image"

}

"weight_corr"选项：消除 weight 量化带来的误差

"feature_corr"选项：消除 feature 量化带来的误差

"corr_dir"选项：传入进行 bias 校准的图片集

"vali_dir"选项：传入验证的图片集，完成 bias 校准后，验证 bias 校准效果

使用"bias_corr"选项依然无法解决问题，可通过提升量化位宽选项消除量化误差。

"feature_quantize_bits":12,

"weight_quantize_bits":10,

D.算子余弦相似度表现正常，但是精度较差：

原因 1：转换工具模拟原生网络推理存在问题，和现象 2）的原因 3 相同，但是现象表现形式不同

解决：可参考现象 2）原因 3

原因 2：板端推理存在问题

解决：使用转换工具生成的"./result/path/input_uint8.bin"数据作为板端推理的输入，保存板端推理输出的结果信息，与和"result/path/ptq_model"路径存放模拟结果(uint8_t 或者 uint16_t 二进制形式 NHWC)进行对比，确认是否板端推理是否存在问题；

如果存在问题反馈转换工具对接人员，将转换工具生成的"./result/path/input_uint8.bin"数据和"result/path/ptq_model"路径下存在问题层的输入输出的 dump 结果发送对接人员，修复板端问题。