模型转换指导

文档版本 01

发布日期 2020-05-09

华为技术有限公司

商标声明

HUAWEI和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

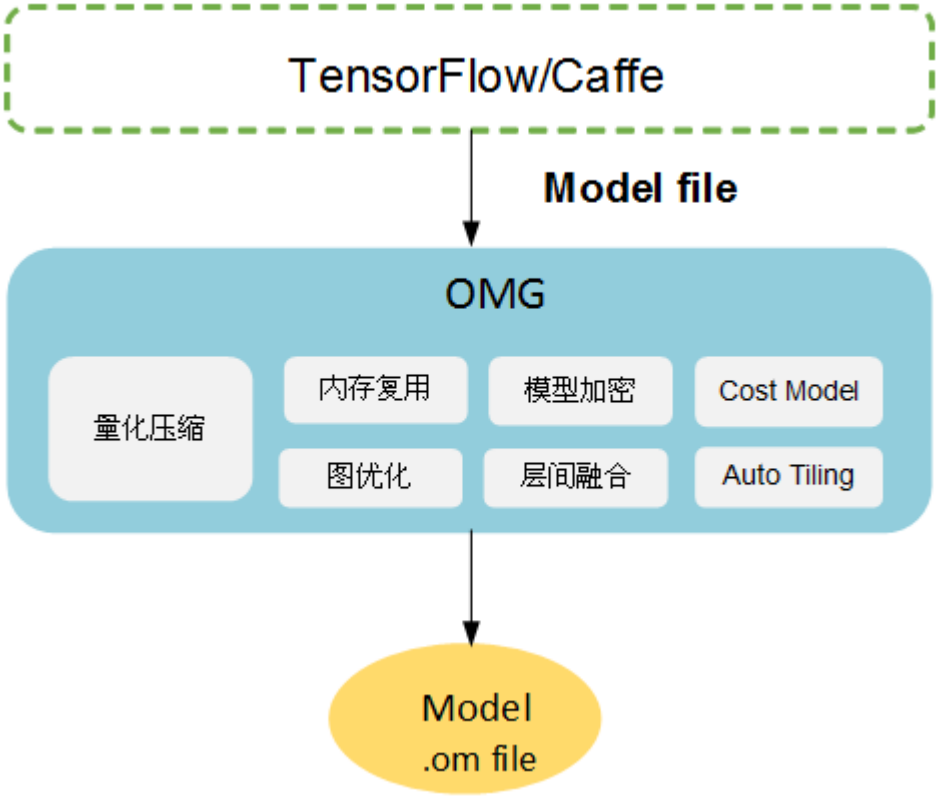
1 简介

本文介绍如何将开源框架的网络模型，例如Caffe、TensorFlow等框架训练好的模型，通过OMG（Offline Model Generator：离线模型生成器）将其转换成昇腾AI处理器支持的离线模型，模型转换过程中可以实现算子调度的优化、权值数据重排、量化压缩、内存使用优化等，可以脱离设备完成模型的预处理。

工具功能架构

OMG工具功能架构如图1-1所示。

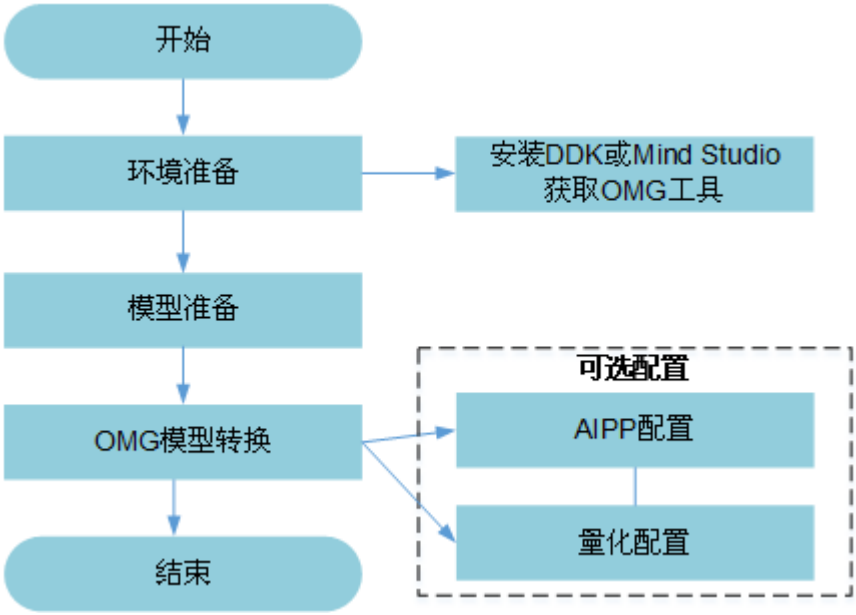
图 1-1 OMG 工具功能架构



工具运行流程

使用OMG工具进行模型转换的总体流程如图1-2所示。

图 1-2 运行流程



详细流程说明如下：

1. 使用OMG工具之前，请先在服务器安装DDK或Mind Studio，获取相关路径下的OMG工具，详细说明请参见准备动作中的环境准备。
2. 准备要进行转换的模型，并上传到DDK所在服务器，详细说明请参见使用示例。
3. 使用OMG工具进行模型转换，在配置相关参数时，根据实际情况选择是否进行AIPP配置或者量化配置。
   a. AIPP是昇腾AI处理器提供的硬件图像预处理模块，包括色域转换，图像归一化（减均值/乘系数）和抠图（指定抠图起始点，抠出神经网络需要大小的图片）等功能。DVPP模块输出的图片多为对齐后的YUV420SP类型，不支持输出RGB图片。因此，业务流需要使用AIPP模块转换对齐后YUV420SP类型图片的格式，并抠出模型需要的输入图片。
   b. 量化是指对高精度数据进行低Bit量化，在对模型大小和性能有更高要求的时候可以选择执行量化操作。模型转换过程中量化会将高精度数据向低比特数据进行量化，让最终生成的模型更加轻量化，从而达到节约网络存储空间、降低传输时延以及提高运算执行效率的目的。

2 转换 Caffe/TensorFlow 网络模型

本节介绍用户使用Caffe/Tensorflow等模型，如何通过OMG工具将其转换为昇腾AI处理器支持的离线模型。

2.1 约束及参数说明

约束说明

在进行模型转换前，请务必查看如下约束要求：

- 只支持原始框架类型为caffe和tensorflow的模型转换，当原始框架类型为caffe时，输入数据类型为FLOAT；当原始框架类型为tensorflow时，输入数据类型为INT32、BOOL、UINT8、FLOAT。
- 当原始框架类型为caffe时，模型文件（.prototxt）和权重文件（.caffemodel）的op name、op type必须保持名称一致（包括大小写）。
- 当原始框架类型为Caffe时，除了top与bottom相同的layer以外（例如BatchNorm，Scale，ReLU等），其他layer的top名称需要与其name名称保持一致。
- 当原始框架类型为tensorflow时，只支持FrozenGraphDef格式。
- 不支持动态shape的输入，例如：NHWC输入为[?,?,?,3]多个维度可任意指定数值。模型转换时需指定固定数值。
- 输入数据最大支持四维，转维算子（reshape、expanddim等）不能输出五维。
- 模型中的所有层算子除const算子外，输入和输出需要满足dim!=0。
- 模型转换不支持含有训练算子的模型。
- 量化（uint8）后的模型不支持模型转换。
- 模型中的算子只支持2D卷积，暂不支持3D卷积。
- 只支持《算子清单》中的算子，并需满足算子限制条件。

参数说明

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--mode	运行模式 ● 0：生成适配昇腾AI处理器的离线模型 ● 1：离线模型或模型文件转json ● 3：仅做预检，检查模型文件的内容是否合法。	否	0
--model	原始模型文件路径。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	是	不涉及
--weight	权重文件路径。 当原始模型是caffe时需要指定。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--framework	原始框架类型 ● 0：caffe ● 3：tensorflow **说明** ● 当mode为1时，该参数可选，可以指定caffe或tensorflow，不指定时默认为离线模型转json，如果指定时需要保证--om模型和--framework类型对应一致，例如：--framework=0 --om=/home/*username*/*test*/resnet18.prototxt ● 当mode为0或3时，该参数必选，可以指定caffe或tensorflow。	是	不涉及
--output	存放转换后的离线模型的路径（包含文件名），例如“out/caffe_resnet18”。 转换后的离线模型，会自动以“.om”的后缀结尾。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	是	不涉及
--encrypt_mode（预留，目前不支持）	加密模式 ● 0：加密 ● -1：不加密	否	-1

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--encrypt_key（预留，目前不支持）	用于加密的随机数文件所在的路径。 加密模式下必填。 **说明** ● 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)。 ● 测试时，您可以使用**openssl rand 32 -out ek_key**命令生成一个随机数文件。实际商用时，用户可根据实际需求选择其它工具生成随机数文件。	否	不涉及
--hardware_key（预留，目前不支持）	加密使用的ISV硬件密钥文件（这个文件是经过加密的）路径。 加密模式下必填。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--certificate（预留，目前不支持）	加密使用的ISV证书文件路径。 加密模式下必填。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--private_key（预留，目前不支持）	加密使用的ISV私钥文件路径。 加密模式下必填。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--cal_conf	量化配置文件路径。 **说明** ● 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.) ● 量化配置文件的内容示例如下： device：USE_CPU bin：150 type：JSD quantize_algo：NON_OFFSET inference_with_data_quantized：true inference_with_weight_quantized：true ● 量化配置文件的配置说明，请查看**4 量化配置**。	否	不涉及

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--check_report	预检结果保存文件路径。若不指定该路径，在模型转换失败或mode为3（仅做预检）时，将预检结果保存在当前路径下。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	check_result.json
--h或--help	显示帮助信息。	否	不涉及
--input_format	输入数据格式：NCHW和NHWC ● 当原始框架是tensorflow时，默认是NHWC。如果实际是NCHW的话，需要通过此参数指定NCHW。 ● 原始框架为Caffe时，只支持NCHW格式。	否	不涉及
--input_fp16_nodes	该参数与--is_output_fp16配合使用。 多网络串联时，指定下一个网络输入数据类型为fp16的输入节点名称。 例如：“node_name1; node_name2”。 举例说明：两个网络net1和net2串联，net1的输出作为net2的输入，则通过该参数指定net2网络接收net1输出数据类型为fp16的节点名称。	否	不涉及
--input_shape	模型输入数据的shape。 例如：“input_name1: n1, c1, h1, w1; input_name2: n2, c2, h2, w2”。 input_name必须是转换前的网络模型中的节点名称。 若原始模型为动态shape，例如input_name1:？,h,w,c，该参数必填。 其中“？”为batch数，表示一次处理的图片数量，需要用户根据实际情况填写，用于将动态shape的原始模型转换为固定shape的离线模型。	否	不涉及
--is_output_fp16	多网络串联时，指定前一个网络输出的数据类型是否为fp16。 例如：false,true,false,true。 举例说明：两个网络net1和net2串联，net1的输出作为net2的输入，则通过该参数指定net1的输出数据类型为fp16。	否	false

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--json	离线模型或模型文件转换为json格式文件的路径。 **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--om	当mode为1时必填。 需要转换为json格式的离线模型或模型文件的路径。例如/home/*username*/test/out/caffe_resnet18.om或/home/*username*/test/resnet18.prototxt **说明** 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)	否	不涉及
--op_name_map	算子映射配置文件路径，网络中包含DetectionOutput算子时需要指定。 例如：不同的网络中DetectionOutput算子的功能不同，可能指定DetectionOutput（Davinci模型中的算子）到如下算子的映射： ● FSRDetectionOutput：fasterrcnn网络中的算子 ● SSDDetectionOutput：SSD网络中的算子。 ● RefinedetDetectionOutput：Refinedet网络中的算子。 **说明** ● 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.) ● 算子映射配置文件的内容示例如下：DetectionOutput：SSDDetectionOutput。	否	不涉及
--out_nodes	指定输出节点。 如果不指定输出节点（算子名称），则模型的输出默认为最后一层的算子信息，某些情况下，用户想要查看某层算子参数是否合适，则需要将该层算子的参数输出，既可以在模型转换时通过该参数指定输出某层算子，模型转换后，在相应.om模型文件最后一层即可以看到指定输出算子的参数信息，如果通过.om模型文件无法查看，则可以将.om模型文件转换成json格式后查看。 例如：“node_name1:0;node_name1:1;node_name2:0”。 node_name必须是模型转换前的网络模型中的节点名称，冒号后的数字表示第几个输出，例如node_name1:0，表示节点名称为node_name1的第0个输出。	否	不涉及

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--plugin_path	自定义算子插件路径。 例如："/home/a1/b1;/home/a2/b2;/home/a3/b3" **说明** 自定义算子插件路径中可以包含多个路径，两个路径之间以分号分割，每个路径内不能包含分号，否则将导致解析得到的路径与预期不符。	否	./plugin
--target	当前仅支持设置为“mini”。 mini：量化中eltwise算子支持双输出；量化中roipooling算子支持int8输出；量化中conv算子支持混合精度。	否	mini
--ddk_version	指定自定义算子运行需要匹配的ddk环境的版本号。	否	不涉及
--net_format	指定网络算子优先选用的数据格式，ND（N<=4）和5D。仅在网络中算子的输入数据同时支持ND和5D两种格式时，指定该参数才生效。 ● ND：模型中算子按NCHW转换成通用格式。 ● 5D：模型中算子按华为自研的5维转换成华为格式。	否	不涉及

--insert_op_conf	输入预处理算子的配置文件路径，例如aipp算子。
	说明
	● 路径部分：支持大小写字母、数字，下划线；文件名部分：支持大小写字母、数字，下划线和点(.)
	● 配置文件的内容示例如下：
	aipp_op {
	aipp_mode: static
	input_format：YUV420SP_U8
	csc_switch：true
	var_reci_chn_0：0.00392157
	var_reci_chn_1：0.00392157
	var_reci_chn_2：0.00392157
	}
	● aipp配置文件的配置说明，请查看**3 AIPP配置**。
		否	不涉及

参数名称	参数描述	是否必选(以mode为0和3为准)	默认值
--fp16_high_prec	指定是否生成高精度FP16 Davinci模型。 ● 0：默认值，生成普通FP16 Davinci模型，推理性能更好。 ● 1：生成高精度FP16 Davinci模型，推理精度更好。 高精度当前仅支持如下算子Caffe：Convolution，Pooling，FullConnection；TensorFlow：tf.nn.conv2d，tf.nn.max_poo算子。	否	0
--output_type	网络输出数据类型： ● FP32：默认值，推荐分类网络、检测网络使用。 ● UINT8：图像超分辨率网络，推荐使用，推理性能更好。	否	FP32
--enable_l2dynamic	L2动态优化开关，该参数可能会影响网络模型的推理性能，如果性能不满足要求，可以尝试关闭该开关，验证性能影响。 ● true：是，打开L2动态优化开关。 ● false：否，关闭L2动态优化开关。	否	true

2.2 使用 OMG 工具转换模型

准备动作

- **环境准备**

  本手册介绍命令行方式的模型转换，如果用户想通过Mind Studio界面进行模型转换，则请参见《Ascend 310 Mind Studio基本操作》。

  本手册以DDK独立安装为例进行说明。

  如果DDK安装用户为HwHiAiUser，日志未输出到屏幕或者DDK所在服务器操作系统以及架构为Arm（aarch64），模型转换耗时较长，则可以分别参见**5.1 DDK安装用户为HwHiAiUser时，日志未输出到屏幕**和**5.2 DDK所在服务器操作系统以及架构为Arm（aarch64），模型转换耗时较长**解决。

- 以DDK安装用户将模型转换过程中使用到的模型文件（*.prototxt）、权重文件（*.caffemodel）等上传到DDK所在服务器/home/*username*/*test*/目录下。

使用示例

步骤1 以DDK安装用户登录DDK所在服务器。

步骤2 设置环境变量：

```
export LD_LIBRARY_PATH=DDK安装目录/ddk/uihost/lib
```

步骤3 进入DDK安装目录“ddk/uihost/bin”下获取omg工具。执行以下命令生成模型文件。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
./omg --model=/home/username/test/resnet18.prototxt --weight=/home/username/test/
resnet18.caffemodel --framework=0 --output=/home/username/test/out/caffe_resnet18
```

成功执行命令后，在output参数指定的路径下，可查看模型文件（如：caffe_resnet18.om）。

说明

如果您直接复制示例中的命令，由于PDF文档格式的限制，超过单行的命令，会自动换行，因此您需要手动将多行命令合并成一行，参数之间以空格分割。

模型转换过程中如果有如下错误提示信息“It is recommended to convert layers-structure to layer-structure by caffe tool”或“Type XXX unsupported”，则请参见**5 FAQ**解决。

步骤4 （可选）如果模型转换过程中指定了输出节点（即使用了--out_nodes参数），转换成.om模型后无法查看最后一层算子的输出信息，则可以将.om模型转换成json格式后查看，使用命令为：

```
omg --mode=1 --om=/home/username/test/caffe_resnet18.om  --json=/home/username/test/out/resnet.json
```

----结束

3 AIPP 配置

3.1 基本介绍

AIPP（AI Preprocessing）用于在AI Core上完成图像预处理，包括改变图像尺寸、色域转换（转换图像格式）、减均值/乘系数（改变图像像素）。

AIPP区分为静态AIPP和动态AIPP。您只能选择静态AIPP或动态AIPP方式来处理图片，不能同时配置静态AIPP和动态AIPP两种方式。

- 静态AIPP：模型转换时设置AIPP模式为静态，同时设置AIPP参数，模型生成后，AIPP参数值被保存在Davinci模型，每次模型推理阶段都使用相同的AIPP参数。

  如果使用静态AIPP方式，多batch情况下共用同一份AIPP参数。

- 动态AIPP：模型转换时设置AIPP模式为动态，每次模型推理前需要在推理Engine的代码中设置动态AIPP参数值，然后在模型推理时可使用不同的AIPP参数。在推理Engine的代码中设置动态AIPP参数值涉及的API及示例请参见《Matrix API参考》中的“模型管家接口”>“AIPP配置接口”。

  如果使用动态AIPP方式，多batch使用不同的参数，体现在动态参数结构体中，每个batch可以配置不同的crop、resize等参数。关于动态参数结构体，请参见**3.7 动态AIPP的参数输入结构**。

AIPP支持的图像输入格式包括：YUV420SP_U8、XRGB8888_U8、RGB888_U8、YUV400_U8。

- 对于YUV420SP_U8，根据UV交织顺序不同，YUV420SP_U8又分为YUV420SP_UV(NV12)和YUV420SP_VU(NV21)，默认为YUV420SP_UV(NV12)：

  - 若3.2 配置文件模板中rbuv_swap_switch（R通道与B通道交换开关/U通道与V通道交换开关）设置为false，则AIPP输出为YUV420SP_UV(NV12)
  - 若3.2 配置文件模板中rbuv_swap_switch设置为true，则AIPP输出为YUV420SP_UV(NV12)。
- 对于RGB888_U8，根据rbuv_swap_switch参数的取值不同，AIPP输出不同：
  - 若3.2 配置文件模板中rbuv_swap_switch设置为false，则AIPP输出为RGB888_U8。
  - 若3.2 配置文件模板中rbuv_swap_switch设置为true，则AIPP输出为BGR888_U8。

须知

- AIPP的输入格式为“YUV420SP_U8”（默认为“YUV420SP_UV”格式），若格式为“YUV420SP_VU”，请修改参数“rbuv_swap_switch”为false，否则会影响输出结果。
- 开启AIPP时，模型输入为“RGB888_U8”或“BGR888_U8”，对应不同的色域转换矩阵。
- 模型转换是否开启AIPP功能，执行推理业务时，对输入图片数据的要求：
  - 模型转换时开启AIPP：图像选择XRGB8888_U8或RGB888_U8，使用该种配置转换后的模型，在进行推理业务时，输入图片数据要求为NHWC排布。
  - 模型转换时没有开启AIPP，模型转换完毕，在进行推理业务时，输入图片数据要求为NCHW排布，因此需要用户自行把NHWC排布的原始图片数据转换为NCHW排布。

3.2 配置文件模板

配置文件说明如下：

```
# AIPP的配置以aipp_op开始，标识这是一个AIPP算子的配置，aipp_op支持配置多个
aipp_op {
#
# AIPP当前支持色域转换、抠图、减均值、乘系数、通道数据交换、单行模式的能力。
# 输入图片的类型仅支持UINT8格式。
# 使用此配置文件时，请将需要配置的参数去注释，并改为合适的值。
# 模板中参数值为默认值，其中input_format属性为必选属性，其余属性均为可选配置。
#========================= 全局设置（start）
==========================================================================================
================================
# aipp_mode指定了AIPP的模式，必须配置
# 类型：enum
# 取值范围：dynamic/static，dynamic 表示动态AIPP，static 表示静态AIPP
# aipp_mode:

# related_input_rank参数为可选，标识对模型的第几个输入做AIPP处理，从0开始，默认为0。例如模型有两个输入，需要对第2个输入做AIPP，则配置related_input_rank为1。
# 类型: 整型
# 配置范围 >= 0
# related_input_rank: 0

# input_edge_idx参数为可选，如果一个模型输入为多个算子共有，即Data算子后面跟着多个算子，配置该参数，对Data算子的不同的输出边做不同的AIPP处理。
# 类型: 整型
# 配置范围 >= 0
# input_edge_idx: 0
```

```
#========================= 全局设置（end）
==========================================================================================
================================

#========================= 动态AIPP需设置，静态AIPP无需设置（start）
==========================================================================================
=======
# 输入图像最大的size，必须大于等于原始图像的大小
# 类型：int
# max_src_image_size: 0

# 是否支持旋转，保留字段，暂不支持该功能
# 类型：bool
# 取值范围：true/false，true表示支持旋转，false表示不支持旋转
# support_rotation: false
#========================= 动态AIPP需设置，静态AIPP无需设置（end）
==========================================================================================
========

#========================= 静态AIPP需设置，动态AIPP无需设置 （start）
==========================================================================================
======
# 输入图像类型
# 类型: enum
# 取值范围：YUV420SP_U8/XRGB8888_U8/RGB888_U8/YUV400_U8
# input_format :

# 图像的宽度、高度
# 类型：uint13
# 取值范围 & 约束： [0,4096]、对于YUV420SP_U8类型的图像，要求取值是偶数
# 说明：请根据实际图片的宽、高配置src_image_size_w、src_image_size_h，若src_image_size_w、
src_image_size_h同时不设置或同时设置为0，则会取网络输入定义的w和h
# src_image_size_w :0
# src_image_size_h :0

# c_padding_value :0.0
#========= crop参数设置（配置样例请参见AIPP配置 > Crop/Padding配置说明） =========
# AIPP处理图片时是否支持抠图
# 类型：bool
# 取值范围：true/false，true表示支持，false表示不支持
# crop :false

# 抠图起始位置水平、垂直方向坐标，抠图大小为网络输入定义的w和h
# 类型：uint13
# 取值范围 & 约束： [0,4096]、对于YUV420SP_U8类型的图像，要求取值是偶数
#说明：load_start_pos_w加上网络输入定义的w需要小于等于src_image_size_w，load_start_pos_h加上网络输
入定义的h需要小于等于src_image_size_h
# load_start_pos_w :0
# load_start_pos_h :0

# 抠图后的图像size
# 类型：uint13
# 取值范围 & 约束： [0,4096]、偶数、load_start_pos_w + crop_size_w <= src_image_size_w、
load_start_pos_h + crop_size_h <= src_image_size_h
# crop_size_w :0
# crop_size_h :0


#================================= resize参数设置 ================================
# AIPP处理图片时是否支持缩放，保留字段，暂不支持该功能
# 类型：bool
# 取值范围：true/false，true表示支持，false表示不支持
resize :false

# 缩放后图像的宽度和高度，保留字段，暂不支持该功能
# 类型：uint13
# 取值范围 & 约束： [0,4096]、偶数、小于src_image_size
resize_output_w :0
resize_output_h :0
```

```
#======== padding参数设置（配置样例请参见AIPP配置 > Crop/Padding配置说明） =========
# AIPP处理图片时padding使能开关
# 类型：bool
# 取值范围：true/false，true表示支持，false表示不支持
# padding :false

# C方向的填充值，静态AIPP配置
# 类型：  float16
# left_padding_size :0
# right_padding_size :0
# top_padding_size :0
# bottom_padding_size :0


#================================ rotation参数设置 ==================================
# AIPP处理图片时的旋转角度，保留字段，暂不支持该功能
# 类型：uint8
# 范围：{0, 1, 2, 3} 0不旋转，1顺时针90°，2顺时针180°，3顺时针270°
# rotation_angle :0


#========= 色域转换参数设置（配置样例请参见AIPP配置 > 色域转换配置说明） =============
# 色域转换开关，静态AIPP配置
# 类型：bool
# 取值范围：true/false，true表示开启色域转换，false表示关闭
# csc_switch :false

# 色域转换前，R通道与B通道交换开关/U通道与V通道交换开关
# 类型：bool
# 取值范围：true/false，true表示开启通道交换，false表示关闭
# rbuv_swap_switch :false

# 单行处理模式（只处理抠图后的第一行）开关
# 类型：bool
# 取值范围：true/false，true表示开启单行处理模式，false表示关闭
# single_line_mode :false

# 若色域转换开关为false，则本功能旁路。
# 若输入图片通道数为4，则忽略第一通道。
# YUV转BGR：
# | B |   | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | Y - input_bias_0 |
# | G | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | | U - input_bias_1 | >> 8
# | R |   | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | V - input_bias_2 |
# BGR转YUV：
# | Y |   | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | B |        | output_bias_0 |
# | U | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | | G | >> 8 + | output_bias_1 |
# | V |   | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | R |        | output_bias_2 |

# 3*3 CSC矩阵元素
# 类型：int16
# 取值范围：[-32768 ,32767]
# matrix_r0c0 :298
# matrix_r0c1 :516
# matrix_r0c2 :0
# matrix_r1c0 :298
# matrix_r1c1 :-100
# matrix_r1c2 :-208
# matrix_r2c0 :298
# matrix_r2c1 :0
# matrix_r2c2 :409

# RGB转YUV时的输出偏移
# 类型：uint8
# 取值范围：[0, 255]
# output_bias_0 :16
# output_bias_1 :128
# output_bias_2 :128
```

```
# YUV转RGB时的输入偏移
# 类型：uint8
# 取值范围：[0, 255]
# input_bias_0 :16
# input_bias_1 :128
# input_bias_2 :128


#============================ 减均值、乘系数设置 ================================
# 计算规则如下：
# 当uint8->uint8时，本功能旁路
# 当uint8->int8时，pixel_out_chx(i) = pixel_in_chx(i) – mean_chn_i
# 当uint8->fp16时，pixel_out_chx(i) = [pixel_in_chx(i) – mean_chn_i – min_chn_i] * var_reci_chn

# 通道n均值
# 类型：uint8
# 取值范围：[0, 255]
# mean_chn_0 :0
# mean_chn_1 :0
# mean_chn_2 :0

# 通道n最小值
# 类型：float16
# 取值范围：[-65504, 65504]
# min_chn_0 :0.0
# min_chn_1 :0.0
# min_chn_2 :0.0

# 通道n方差或(max-min)的倒数
# 类型：float16
# 取值范围：[-65504, 65504]
# var_reci_chn_0 :1.0
# var_reci_chn_1 :1.0
# var_reci_chn_2 :1.0
}
#======================== 静态AIPP需设置，动态AIPP无需设置 （ end ）
==========================================================================================
======
```

3.3 色域转换配置说明

一旦确认了AIPP处理前与AIPP处理后的图片格式（即模型输入的图片格式与模型训练阶段的图片格式），即可确定色域转换相关的参数值（matrix_r*c*配置项的值是固定的，不需要调整），配置可参考如下内容。

在AIPP处理前，针对模型输入的图片或视频（各颜色编码方式，如YUV420SP_U8、XRGB8888_U8、RGB888_U8、YUV400_U8），当前给出以下两种典型场景下的色域转换配置：

- 对于JPEG图像文件格式（如后缀为jpg、jpeg、JPG、JPEG的图像文件），可以选择以下表格中“JPEG”列的色域转换配置。
- 对于视频解码后的数据，可以根据不同的彩色视频数字化标准（如BT-601、BT-709等）配置色域转换参数。

YUV420SP_U8 转 YVU444SP_U8

```
aipp_op {
    aipp_mode: static
    input_format : YUV420SP_U8
    csc_switch : false
    rbuv_swap_switch : true
}
```

YUV420SP_U8 转 RGB888_U8

表 3-1 YUV420SP_U8 转 RGB888_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : YUV420SP_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 256
matrix_r0c1 : 0
matrix_r0c2 : 359
matrix_r1c0 : 256
matrix_r1c1 : -88
matrix_r1c2 : -183
matrix_r2c0 : 256
matrix_r2c1 : 454
matrix_r2c2 : 0
input_bias_0 : 0
input_bias_1 : 128
input_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : YUV420SP_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 298
	matrix_r0c1 : 0
	matrix_r0c2 : 409
	matrix_r1c0 : 298
	matrix_r1c1 : -100
	matrix_r1c2 : -208
	matrix_r2c0 : 298
	matrix_r2c1 : 516
	matrix_r2c2 : 0
	input_bias_0 : 16
	input_bias_1 : 128
	input_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : YUV420SP_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 298
		matrix_r0c1 : 0
		matrix_r0c2 : 459
		matrix_r1c0 : 298
		matrix_r1c1 : -55
		matrix_r1c2 : -136
		matrix_r2c0 : 298
		matrix_r2c1 : 541
		matrix_r2c2 : 0
		input_bias_0 : 16
		input_bias_1 : 128
		input_bias_2 : 128
		}

YUV420SP_U8 转 BGR888_U8

表 3-2 YUV420SP_U8 转 BGR888_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : YUV420SP_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 256
matrix_r0c1 : 454
matrix_r0c2 : 0
matrix_r1c0 : 256
matrix_r1c1 : -88
matrix_r1c2 : -183
matrix_r2c0 : 256
matrix_r2c1 : 0
matrix_r2c2 : 359
input_bias_0 : 0
input_bias_1 : 128
input_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : YUV420SP_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 298
	matrix_r0c1 : 516
	matrix_r0c2 : 0
	matrix_r1c0 : 298
	matrix_r1c1 : -100
	matrix_r1c2 : -208
	matrix_r2c0 : 298
	matrix_r2c1 : 0
	matrix_r2c2 : 409
	input_bias_0 : 16
	input_bias_1 : 128
	input_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : YUV420SP_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 298
		matrix_r0c1 : 541
		matrix_r0c2 : 0
		matrix_r1c0 : 298
		matrix_r1c1 : -55
		matrix_r1c2 : -136
		matrix_r2c0 : 298
		matrix_r2c1 : 0
		matrix_r2c2 : 459
		input_bias_0 : 16
		input_bias_1 : 128
		input_bias_2 : 128
		}

YUV420SP_U8 转 GRAY

```
aipp_op {
    aipp_mode: static
    input_format : YUV420SP_U8
    csc_switch : true
    rbuv_swap_switch : false
    matrix_r0c0 : 256
    matrix_r0c1 : 0
    matrix_r0c2 : 0
```

```
    matrix_r1c0 : 0
    matrix_r1c1 : 0
    matrix_r1c2 : 0
    matrix_r2c0 : 0
    matrix_r2c1 : 0
    matrix_r2c2 : 0
    input_bias_0 : 0
    input_bias_1 : 0
    input_bias_2 : 0
}
```

XRGB8888_U8 转 YUV444SP_U8

表 3-3 XRGB8888_U8 转 YUV444SP_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : XRGB8888_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 77
matrix_r0c1 : 150
matrix_r0c2 : 29
matrix_r1c0 : -43
matrix_r1c1 : -85
matrix_r1c2 : 128
matrix_r2c0 : 128
matrix_r2c1 : -107
matrix_r2c2 : -21
output_bias_0 : 0
output_bias_1 : 128
output_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : XRGB8888_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 66
	matrix_r0c1 : 129
	matrix_r0c2 : 25
	matrix_r1c0 : -38
	matrix_r1c1 : -74
	matrix_r1c2 : 112
	matrix_r2c0 : 112
	matrix_r2c1 : -94
	matrix_r2c2 : -18
	output_bias_0 : 16
	output_bias_1 : 128
	output_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : XRGB8888_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 47
		matrix_r0c1 : 157
		matrix_r0c2 : 16
		matrix_r1c0 : -26
		matrix_r1c1 : -87
		matrix_r1c2 : 112
		matrix_r2c0 : 112
		matrix_r2c1 : -102
		matrix_r2c2 : -10
		output_bias_0 : 16
		output_bias_1 : 128
		output_bias_2 : 128
		}

XRGB8888_U8 转 YVU444SP_U8

表 3-4 XRGB8888_U8 转 YVU444SP_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : XRGB8888_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 77
matrix_r0c1 : 150
matrix_r0c2 : 29
matrix_r1c0 : 128
matrix_r1c1 : -107
matrix_r1c2 : -21
matrix_r2c0 : -43
matrix_r2c1 : -85
matrix_r2c2 : 128
output_bias_0 : 0
output_bias_1 : 128
output_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : XRGB8888_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 66
	matrix_r0c1 : 129
	matrix_r0c2 : 25
	matrix_r1c0 : 112
	matrix_r1c1 : -94
	matrix_r1c2 : -18
	matrix_r2c0 : -38
	matrix_r2c1 : -74
	matrix_r2c2 : 112
	output_bias_0 : 16
	output_bias_1 : 128
	output_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : XRGB8888_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 47
		matrix_r0c1 : 157
		matrix_r0c2 : 16
		matrix_r1c0 : 112
		matrix_r1c1 : -102
		matrix_r1c2 : -10
		matrix_r2c0 : -26
		matrix_r2c1 : -87
		matrix_r2c2 : 112
		output_bias_0 : 16
		output_bias_1 : 128
		output_bias_2 : 128
		}

XRGB8888_U8 转 GRAY

```
aipp_op {
    aipp_mode: static
    input_format : XRGB8888_U8
    csc_switch : true
    rbuv_swap_switch : false
    matrix_r0c0 : 76
    matrix_r0c1 : 150
    matrix_r0c2 : 30
    matrix_r1c0 : 0
    matrix_r1c1 : 0
    matrix_r1c2 : 0
    matrix_r2c0 : 0
    matrix_r2c1 : 0
    matrix_r2c2 : 0
    output_bias_0 : 0
    output_bias_1 : 0
    output_bias_2 : 0
}
```

RGB888_U8 转 BGR888_U8

```
aipp_op {
    aipp_mode: static
    input_format : RGB888_U8
    csc_switch : false
    rbuv_swap_switch : true
}
```

RGB888_U8 转 YUV444SP_U8

表 3-5 RGB888_U8 转 YUV444SP_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : RGB888_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 77
matrix_r0c1 : 150
matrix_r0c2 : 29
matrix_r1c0 : -43
matrix_r1c1 : -85
matrix_r1c2 : 128
matrix_r2c0 : 128
matrix_r2c1 : -107
matrix_r2c2 : -21
output_bias_0 : 0
output_bias_1 : 128
output_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : RGB888_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 66
	matrix_r0c1 : 129
	matrix_r0c2 : 25
	matrix_r1c0 : -38
	matrix_r1c1 : -74
	matrix_r1c2 : 112
	matrix_r2c0 : 112
	matrix_r2c1 : -94
	matrix_r2c2 : -18
	output_bias_0 : 16
	output_bias_1 : 128
	output_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : RGB888_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 47
		matrix_r0c1 : 157
		matrix_r0c2 : 16
		matrix_r1c0 : -26
		matrix_r1c1 : -87
		matrix_r1c2 : 112
		matrix_r2c0 : 112
		matrix_r2c1 : -102
		matrix_r2c2 : -10
		output_bias_0 : 16
		output_bias_1 : 128
		output_bias_2 : 128
		}

RGB888_U8 转 YVU444SP_U8

表 3-6 RGB888_U8 转 YVU444SP_U8

JPEG	BT-601NARROW	BT-709NARROW

aipp_op {
aipp_mode: static
input_format : RGB888_U8
csc_switch : true
rbuv_swap_switch : false
matrix_r0c0 : 77
matrix_r0c1 : 150
matrix_r0c2 : 29
matrix_r1c0 : 128
matrix_r1c1 : -107
matrix_r1c2 : -21
matrix_r2c0 : -43
matrix_r2c1 : -85
matrix_r2c2 : 128
output_bias_0 : 0
output_bias_1 : 128
output_bias_2 : 128
}
	aipp_op {
	aipp_mode: static
	input_format : RGB888_U8
	csc_switch : true
	rbuv_swap_switch : false
	matrix_r0c0 : 66
	matrix_r0c1 : 129
	matrix_r0c2 : 25
	matrix_r1c0 : 112
	matrix_r1c1 : -94
	matrix_r1c2 : -18
	matrix_r2c0 : -38
	matrix_r2c1 : -74
	matrix_r2c2 : 112
	output_bias_0 : 16
	output_bias_1 : 128
	output_bias_2 : 128
	}
		aipp_op {
		aipp_mode: static
		input_format : RGB888_U8
		csc_switch : true
		rbuv_swap_switch : false
		matrix_r0c0 : 47
		matrix_r0c1 : 157
		matrix_r0c2 : 16
		matrix_r1c0 : 112
		matrix_r1c1 : -102
		matrix_r1c2 : -10
		matrix_r2c0 : -26
		matrix_r2c1 : -87
		matrix_r2c2 : 112
		output_bias_0 : 16
		output_bias_1 : 128
		output_bias_2 : 128
		}

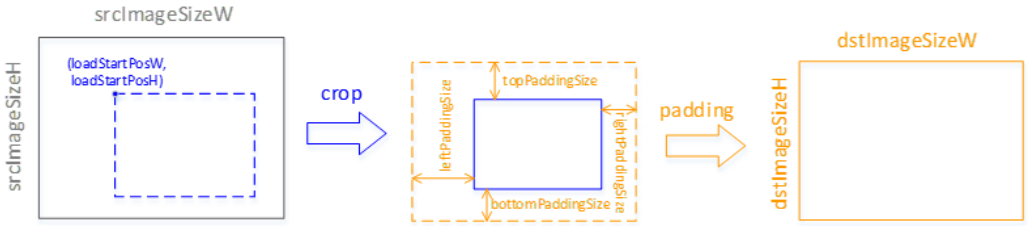
RGB888_U8 转 GRAY

```
aipp_op {
    aipp_mode: static
    input_format : RGB888_U8
    csc_switch : true
    rbuv_swap_switch : false
    matrix_r0c0 : 76
    matrix_r0c1 : 150
    matrix_r0c2 : 30
    matrix_r1c0 : 0
    matrix_r1c1 : 0
    matrix_r1c2 : 0
    matrix_r2c0 : 0
    matrix_r2c1 : 0
    matrix_r2c2 : 0
    output_bias_0 : 0
    output_bias_1 : 0
    output_bias_2 : 0
}
```

3.4 Crop/Padding 配置说明

目前AIPP支持Crop（抠图）、Padding两种改变图像尺寸的操作。

图 3-1 改变图像尺寸



对于YUV420SP_U8图片类型，load_start_pos_w、load_start_pos_h、crop_size_w与crop_size_h四个参数必须配置为偶数。抠图后的图像的宽、高需和网络输入定义的w和h相等。

配置样例如下：

```
aipp_op {
    aipp_mode: static
    input_format : YUV400_U8

    src_image_size_w :320
    src_image_size_h :240

    crop :true
    load_start_pos_w :10
    load_start_pos_h :20
    crop_size_w :50
    crop_size_h :60

    padding : true
    left_padding_size :10
    right_padding_size :20
    top_padding_size :10
    bottom_padding_size :20
}
```

3.5 根据模型输入指定 AIPP 配置说明

AIPP配置文件支持定义多组AIPP配置，对不同的模型输入进行不同的AIPP处理，配置多组AIPP参数时，将一组AIPP配置放到一个aipp_op配置项里，举例如下：

```
aipp_op {
        related_input_rank: 0
        aipp_mode : dynamic
        max_src_image_size: 60000
}
aipp_op {
        related_input_rank : 1
        aipp_mode : dynamic
        max_src_image_size: 80000
}
```

上述配置定义了两组AIPP参数，分别对模型第一个和第二个输入进行AIPP处理，AIPP的模式为动态AIPP，两个输入可以接受的最大的图片大小分别是60000字节和80000字节。

3.6 AIPP 对模型输入大小的校验说明

如果有配置AIPP，无论静态AIPP还是动态AIPP，最终生成的Davinci模型的接收的图片的大小（即input_size）均会被Crop、Padding等操作影响。在模型推理阶段，OME会对传入的图片的大小进行校验，如果是动态AIPP，要求模型推理时传入的图片的大小小于等于动态AIPP配置的max_src_image_size大小；如果是静态AIPP，要求模型推理时传入的图片的大小与**表3-7**中的计算的input_size相等。

对于静态AIPP，假设模型的Batch数量为N，图片的宽为src_image_size_w，高为src_image_size_h，最后模型输入的Size的计算公式如**表3-7**所示。

表 3-7 input_size 校验公式

input_format	input_size
YUV400_U8	N * src_image_size_w * src_image_size_h
YUV420SP_U8	N * src_image_size_w * src_image_size_h * 1.5
XRGB8888_U8	N * src_image_size_w * src_image_size_h * 4
RGB888_U8	N * src_image_size_w * src_image_size_h * 3

生成带动态AIPP的模型时，OMG会为动态AIPP添加模型输入。新增动态AIPP输入的input_size计算公式为：

sizeof(kAippDynamicPara) + (batch_count - 1) * sizeof(kAippDynamicBatchPara)

3.7 动态 AIPP 的参数输入结构

```
typedef struct tagAippDynamicBatchPara
{
    int8_t cropSwitch;              //crop switch
    int8_t scfSwitch;               //resize switch
    int8_t paddingSwitch;   // 0: unable padding,
                            // 1: padding config value,sfr_filling_hblank_ch0 ~    sfr_filling_hblank_ch2
                            // 2: padding source picture data, single row/collumn copy
                            // 3: padding source picture data, block copy
                            // 4: padding source picture data, mirror copy
    int8_t rotateSwitch;  //rotate switch，0: non-ratate，1: ratate 90°clockwise，2: ratate 180°clockwise，3:
ratate 270° clockwise
    int8_t reserve[4];
    int32_t cropStartPosW;          //the start horizontal position of cropping
    int32_t cropStartPosH;          //the start vertical position of cropping
    int32_t cropSizeW;              //crop width
    int32_t cropSizeH;              //crop height
    int32_t scfInputSizeW;          //input width of scf
    int32_t scfInputSizeH;          //input height of scf
    int32_t scfOutputSizeW;         //output width of scf
    int32_t scfOutputSizeH;         //output height of scf
    int32_t paddingSizeTop;         //top padding size
    int32_t paddingSizeBottom;      //bottom padding size
    int32_t paddingSizeLeft;        //left padding size
    int32_t paddingSizeRight;       //right padding size
    int16_t dtcPixelMeanChn0;       //mean value of channel 0
    int16_t dtcPixelMeanChn1;       //mean value of channel 1
    int16_t dtcPixelMeanChn2;       //mean value of channel 2
    int16_t dtcPixelMeanChn3;       //mean value of channel 3
#ifndef DAVINCI_TINY
    uint16_t dtcPixelMinChn0;       //min value of channel 0
    uint16_t dtcPixelMinChn1;       //min value of channel 1
    uint16_t dtcPixelMinChn2;       //min value of channel 2
    uint16_t dtcPixelMinChn3;       //min value of channel 3
    uint16_t dtcPixelVarReciChn0;   //sfr_dtc_pixel_variance_reci_ch0
    uint16_t dtcPixelVarReciChn1;   //sfr_dtc_pixel_variance_reci_ch1
    uint16_t dtcPixelVarReciChn2;   //sfr_dtc_pixel_variance_reci_ch2
    uint16_t dtcPixelVarReciChn3;   //sfr_dtc_pixel_variance_reci_ch3
    int8_t reserve1[16];            //32B assign, for ub copy
#endif
}kAippDynamicBatchPara;
typedef struct tagAippDynamicPara
{
    uint8_t inputFormat;        //input format：YUV420SP_U8/XRGB8888_U8/RGB888_U8
```

```
    //uint8_t outDataType; //output data type: CC_DATA_HALF,CC_DATA_INT8, CC_DATA_UINT8
    int8_t cscSwitch;                //csc switch
    int8_t rbuvSwapSwitch;           //rb/ub swap switch
    int8_t axSwapSwitch;             //RGBA->ARGB, YUVA->AYUV swap switch
    int8_t batchNum;                 //batch parameter number
    int8_t reserve1[3];
    int32_t srcImageSizeW;           //source image width
    int32_t srcImageSizeH;           //source image height
    int16_t cscMatrixR0C0;           //csc_matrix_r0_c0
    int16_t cscMatrixR0C1;           //csc_matrix_r0_c1
    int16_t cscMatrixR0C2;           //csc_matrix_r0_c2
    int16_t cscMatrixR1C0;           //csc_matrix_r1_c0
    int16_t cscMatrixR1C1;           //csc_matrix_r1_c1
    int16_t cscMatrixR1C2;           //csc_matrix_r1_c2
    int16_t cscMatrixR2C0;           //csc_matrix_r2_c0
    int16_t cscMatrixR2C1;           //csc_matrix_r2_c1
    int16_t cscMatrixR2C2;           //csc_matrix_r2_c2
    int16_t reserve2[3];
    uint8_t cscOutputBiasR0;   //output bias for RGB to YUV, element of row 0, unsigned number
    uint8_t cscOutputBiasR1;   //output bias for RGB to YUV, element of row 1, unsigned number
    uint8_t cscOutputBiasR2;   //output bias for RGB to YUV, element of row 2, unsigned number
    uint8_t cscInputBiasR0;    //input bias for YUV to RGB, element of row 0, unsigned number
    uint8_t cscInputBiasR1;    //input bias for YUV to RGB, element of row 1, unsigned number
    uint8_t cscInputBiasR2;    //input bias for YUV to RGB, element of row 2, unsigned number
    uint8_t reserve3[2];
    int8_t reserve4[16];             //32B assign, for ub copy
    kAippDynamicBatchPara aippBatchPara;  //allow transfer several batch para.
} kAippDynamicPara;
```

4 量化配置

4.1 基本介绍

此处量化是指对高精度数据进行低Bit量化，从而达到节约网络存储空间、降低传输时延以及提高运算执行效率的目的。

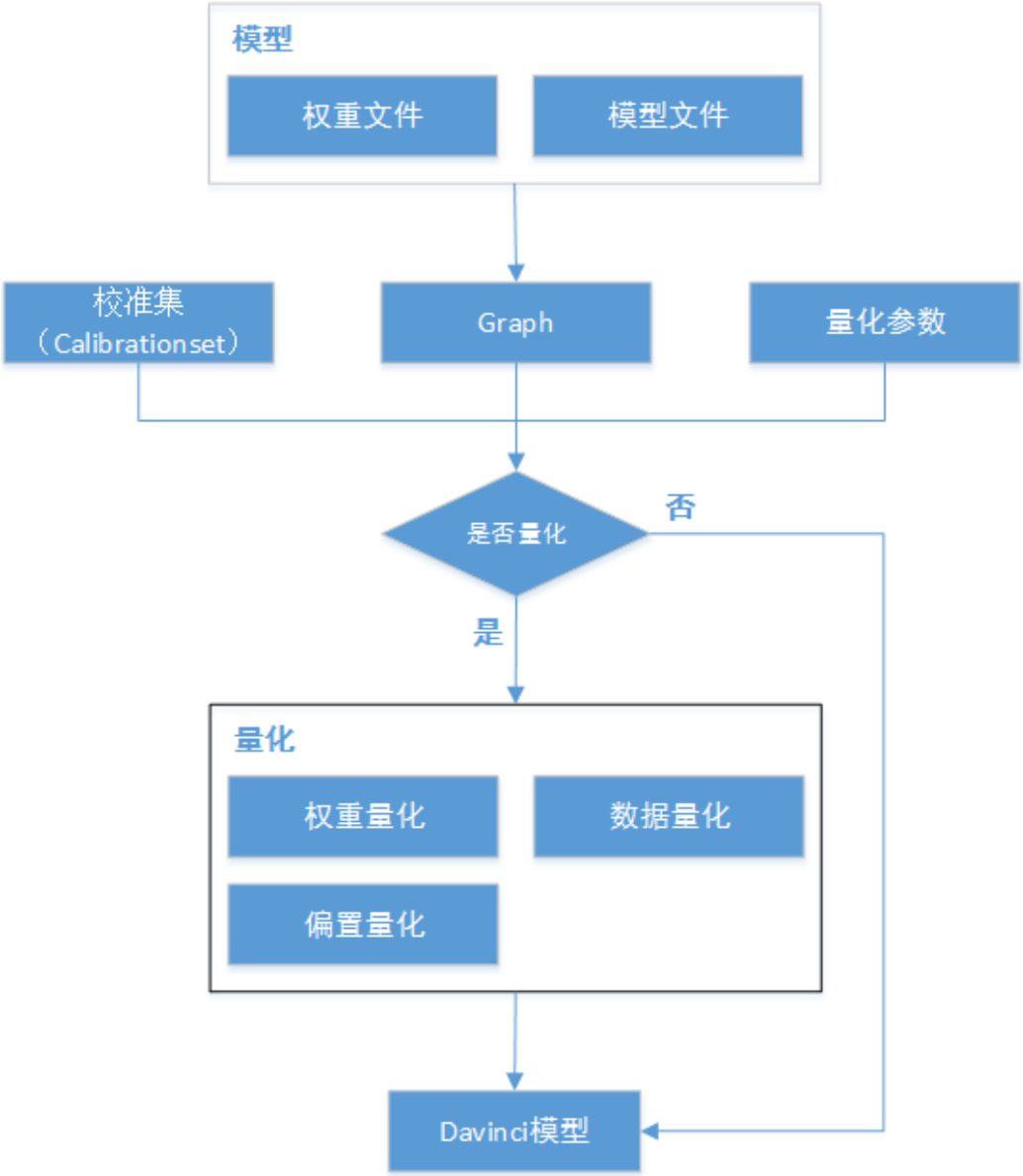
当前支持Convolution、Full Connection、ConvolutionDepthwise三种类型算子的量化，包括权重、偏置、数据量化。量化模式分为：无offset、数据offset。

量化后的权重、偏置在Davinci模型生成阶段即被保存在Davinci模型中，在模型推理阶段使用量化后权重、偏置对数据进行计算。

各概念的说明如下：

- 无offset：权重和数据都采用无偏移模式。
- 数据offset：数据采用有偏移模式，权重采用无偏移模式。
- 权重量化：根据量化算法对权重文件进行Int8量化。若是有offset量化模式，则输出Int8权重、scale和offset；若是无offset量化模式，则输出Int8权重和scale。当前权重量化仅支持无offset量化模式。
- 数据量化：经过有限输入（校准集，用于训练量化参数、保证精度）进行推理执行，根据频率统计和散度计算算法等，若是数据offset量化模式，则计算出量化数据的scale和offset；如果无offset量化模式，则计算出量化数据的scale。
- 偏置量化：根据权重量化的scale、数据量化的scale，将FP32偏置数据量化为Int32输出。

图 4-1 量化处理流程



4.2 配置文件模板

量化支持网络单输入或多输入的场景。使用此配置文件时，可以使用配置文件模板中默认值，也可以根据实际情况将需要的参数改成合适的值。另外，配置文件中的参数分为必选参数和可选参数，必选参数您必须配置对应的参数值，可选参数您可以不配置。配置文件中各参数的含义、是否必选请参见**4.3 参数配置说明**。

配置文件中的内容必须是使用英文格式。

网络单输入场景

输入为图片，通过OMG模型转换命令指定量化配置，配置文件举例如下。

```
device:USE_CPU
quantize_algo:HALF_OFFSET
```

```
weight_type:VECTOR_TYPE
bin:150
type:KL
inference_with_data_quantized:false
inference_with_weight_quantized:true
super_parameter:
{
min_percentile:PERCENTILE_HIGH
max_percentile:PERCENTILE_MID
start_ratio:0.7
end_ratio:1.3
step_ratio:0.01
}
exclude_op:'fc1000'
batch_count:50
preprocess_parameter:
{
input_type:IMAGE
image_format:BGR
input_file_path:'calibration/image_set'
mean_value:104.0
mean_value:117.0
mean_value:123.0
standard_deviation:1.0
}
```

网络多输入场景

第一个输入为图片，第二个输入为二进制文件，通过OMG模型转换命令指定量化配置，配置文件举例如下。

```
device:USE_CPU
quantize_algo:HALF_OFFSET
weight_type:VECTOR_TYPE
bin:150
type:KL
inference_with_data_quantized:false
inference_with_weight_quantized:true
super_parameter:
{
min_percentile:PERCENTILE_HIGH
max_percentile:PERCENTILE_MID
start_ratio:0.7
end_ratio:1.3
step_ratio:0.01
}
exclude_op:'fc1000'
batch_count:50
preprocess_parameter:
{
input_type:IMAGE
image_format:BGR
input_file_path:'calibration/image_set'
mean_value:104.0
mean_value:117.0
mean_value:123.0
standard_deviation:1.0
}
preprocess_parameter:
{
input_type:BINARY
input_file_path:'calibration/img_info.bin'
}
```

4.3 参数配置说明

表 4-1 device 参数说明

作用	推理模式
类型	enum
取值范围	USE_CPU
参数说明	USE_CPU：使用CPU做推理。
推荐配置	USE_CPU
必选或可选	必选

表 4-2 quantize_algo 参数说明

作用	量化模式
类型	enum
取值范围	● NON_OFFSET ● HALF_OFFSET
参数说明	● NON_OFFSET：权重和数据都采用无偏移模式 ● HALF_OFFSET：数据采用有偏移模式，权重采用无偏移模式 根据权重和数据在量化时是否有偏移，量化的映射公式有两种： ● 有偏移，公式为：q_uint8 = round(d_float/scale)-offset ● 无偏移，公式为：q_int8 = round(d_float/scale)
推荐配置	HALF_OFFSET
必选或可选	必选

表 4-3 weight_type 参数说明

作用	权重量化模式
类型	enum
取值范围	● VECTOR_TYPE ● SCALAR_TYPE

参数说明	对于卷积算子，可能有多个卷积核，多个对应的量化参数可能不一样。 ● VECTOR_TYPE：表示一个卷积核对应一组量化参数 ● SCALAR_TYPE：表示多个卷积核使用同一组量化参数
推荐配置	VECTOR_TYPE
必选或可选	必选

表 4-4 preprocess_parameter 参数说明

作用	预处理及输入相关参数
类型	Struct
取值范围	无。
参数说明	指定预处理及输入相关参数。preprocess_parameter参数的数量与网络的输入算子数量必须相同。 在preprocess_parameter内部包含如下参数： ● input_type ● image_format ● input_file_path ● mean_value ● standard_deviation
推荐配置	无
必选或可选	必选

表 4-5 input_type 参数说明

作用	输入数据类型
类型	enum
取值范围	● IMAGE ● BINARY

参数说明	指定输入类型，当前支持图片（IMAGE）和二进制文件（BINARY）两种格式。 ● IMAGE：图片格式。 支持的图片格式包括：".bmp", ".dib", ".jpeg", ".jpg", ".jpe", ".jp2", ".png", ".webp",".pbm", ".pgm", ".ppm", ".sr", ".ras", ".tiff", ".tif",".BMP", ".DIB", ".JPEG", ".JPG", ".JPE", ".JP2", ".PNG", ".WEBP",".PBM", ".PGM", ".PPM", ".SR", ".RAS", ".TIFF", ".TIF"。 ● BINARY：二进制文件（Binary格式）格式，请参见**表4-6**。对于非四维（num、channels、height、width）的数据，需要补齐到四维，补齐的维度值为1。
推荐配置	无
必选或可选	必选

表 4-6 二进制文件格式

文件头/数据	地址偏移	type	value	description
文件头（共20字节）	0000	32bit int	510	magic number 当magic number = 510，用来校验文件的合法性。
	0004	32bit int	50	input num
	0008	32bit int	3	input channels
	0012	32bit int	28	input height
	0016	32bit int	28	input width
数据	…	float	126	数据数量等于num*channels*height*width

表 4-7 input_file_path 参数说明

作用	输入数据地址，即校准集图片所在的路径
类型	string
取值范围	无

参数说明	指定量化输入的目录或文件。路径不要有中文、特殊字符（包括**|;&$><`**）以及空格。 根据实际网络的输入配置。 ● 如果是图片，指定到目录 ● 如果是二进制文件，指定到文件
推荐配置	建议使用与应用场景相关的图片或二进制文件。
必选或可选	必选

表 4-8 image_format 参数说明

作用	图像输入数据三通道排序方式
类型	enum
取值范围	● BGR ● RGB
参数说明	模型训练时候的图片三通道排序格式。 当input_type参数值为IMAGE时需要配置该值。该参数根据实际网络训练时候的输入通道排序方式确定，通常网络训练时候的排序方式是BGR。
推荐配置	BGR
必选或可选	必选

表 4-9 mean_value 参数说明

作用	图像预处理的均值
类型	float
取值范围	[0, 255.0]
参数说明	图片预处理单个通道的均值。 当input_type参数值为IMAGE时需要配置该值。多个通道需要配置多个该参数，通常图片有RGB三个通道，则需要配置三个该参数，如下： ● mean_value：104.0 ● mean_value：117.0 ● mean_value：123.0
推荐配置	无
必选或可选	必选

表 4-10 standard_deviation 参数说明

作用	图像预处理的均方差
类型	float
取值范围	[0,FLOAT_MAX]
参数说明	图片预处理的方差。 当input_type参数值为IMAGE时需要配置该值。多个通道使用统一方差。 如果输入范围大于float型能表示的范围，模型量化精度不能保证。 如果0<=standard_deviation<=0.00001，standard_deviation取1.0。
推荐配置	1.0（图片取值区间大小不发生变化）或者255.0（对于图片区间大小为[0, 255]的场景，可将值压缩到[0, 1.0]）。
必选或可选	必选

表 4-11 bin 参数说明

作用	数据映射直方图范围
类型	uint32
取值范围	[0,1000]
参数说明	数据直方图统计的范围。 在计算散度过程中需要统计数据直方图，该参数的值决定了直方图统计的最大值。如果不配置或者配置为0，则使用默认配置150。
推荐配置	100/150/200/250。
必选或可选	可选

表 4-12 type 参数说明

作用	散度计算指标类型
类型	enum
取值范围	● KL ● SYMKL ● JSD

参数说明	● KL：Kullback-Leibler Divergence ● SYMKL：Symmetric Kullback-Leibler Divergence ● JSD：Jensen-Shannon Divergence。 不同的散度类型对应的计算方式不一样。默认为KL。
推荐配置	无
必选或可选	可选

表 4-13 inference_with_data_quantized 参数说明

作用	推理过程中是否使用量化后的输入数据
类型	bool
取值范围	● true ● false
参数说明	控制推理过程中的输入数据是否经过量化。 该参数开启，模拟了输入数据量化的过程。默认值为false。
推荐配置	false
必选或可选	可选

表 4-14 inference_with_weight_quantized 参数说明

作用	推理过程中是否使用量化后的权重数据
类型	bool
取值范围	● true ● false
参数说明	控制推理过程中的权重数据是否经过量化。 该参数开启，模拟了权重数据量化反量化的过程。默认值为true。
推荐配置	true
必选或可选	可选

表 4-15 super_parameter 参数说明

作用	搜索相关参数
类型	Struct
取值范围	无

参数说明	搜索相关参数。 super_parameter参数内部包含如下参数： ● min_percentile ● max_percentile ● start_ratio ● end_ratio ● step_ratio
推荐配置	无
必选或可选	可选

表 4-16 min_percentile 参数说明

作用	最小值搜索位置
类型	enum
取值范围	● PERCENTILE_HIGH ● PERCENTILE_MID ● PERCENTILE_LOW
参数说明	在从小到大排序的一组数中，决定取第多少小的数，比如有100个数，1.0表示取第100-100*1.0=0，对应的就是第一个小的数。 ● PERCENTILE_HIGH：1.0 ● PERCENTILE_MID：0.99999 ● PERCENTILE_LOW：0.9999
推荐配置	PERCENTILE_HIGH
必选或可选	可选

表 4-17 max_percentile 参数说明

作用	最大值搜索位置
类型	enum
取值范围	● PERCENTILE_HIGH ● PERCENTILE_MID ● PERCENTILE_LOW

参数说明	在从大到小排序的一组数中，决定取第多少大的数，比如有100个数，1.0表示取第100-100*1.0=0，对应的就是第一个大的数。 ● PERCENTILE_HIGH：1.0 ● PERCENTILE_MID：0.99999 ● PERCENTILE_LOW：0.9999
推荐配置	PERCENTILE_MID或PERCENTILE_HIGH
必选或可选	可选

表 4-18 start_ratio、end_ratio、step_ratio 参数说明

作用	参数说明
类型	float
取值范围	end_ratio>start_ratio>0 && step_ratio>0（如果输入范围大于float型能表示的范围，模型量化精度不能保证）
参数说明	● start_ratio：决定搜索开始的位置 ● end_ratio：决定搜索结束的位置 ● step_ratio：决定搜索步长 在算法中找到d_max/d_min之后，会根据此参数，取d_max/d_min前后多少范围之内的数，然后根据step_ratio决定，每次增加的步长，说明参考如下： 以d_max =100，start_ratio=0.8，end_ratio=1.2，step_ratio=0.01为例，其定义的d_max搜索空间为从100*0.8=80到100*1.2=120的范围，每次步进100*0.01=1，一共41个d_max搜索值。
推荐配置	推荐配置有两组 ● start_ratio:0.7 end_ratio:1.3 step_ratio:0.01 ● start_ratio:0.3 end_ratio:1.7 step_ratio:0.01
必选或可选	可选

表 4-19 batch_count 参数说明

作用	量化校准集图片处理的图片数量
类型	uint32
取值范围	[0,UINT32_MAX)

参数说明	决定量化的读取的校准集图片数量。 如果不配置或者配置为0，那么会把校准集所有图片都作为校准集数据。如果配置了大于0的数，那么会根据校准集路径下图片总数和该参数取较小值作为校准集的图片实际数量。建议校准集图片数量不超过500张。
推荐配置	50
必选或可选	可选

表 4-20 exclude_op 参数说明

作用	量化算子黑名单
类型	string
取值范围	算子名称
参数说明	配置该参数，算子不进行量化。 ● 只支持Convolution、Full Connection、ConvolutionDepthwise算子； ● 多个算子，需要配置多个该参数，每个参数一行，如：exclude_op:'aaa' exclude_op:'bbb'
推荐配置	无
必选或可选	可选

4.4 参数调优说明

按照模型转换时设置的量化参数对权重、偏置、数据进行量化，若精度不满足要求，您可以按照如下步骤调整量化参数值。

1. 按照**4.2 配置文件模板**中的默认参数值进行校准，完成模型数据和参数的量化。
2. 若按照**1**中的量化配置进行量化后，精度满足要求，则调参结束，否则继续**3**。
3. 手动调整以下量化参数后，再进行量化。
   a. 修改搜索范围参数。

表 4-21 搜索范围参数

修改前	修改后
start_ratio:0.7 end_ratio:1.3 step_ratio:0.01	start_ratio:0.3 end_ratio:1.7 step_ratio:0.01

b. 将bin参数值改大，从150改成200或者250。

c. 如果是检测类网络，建议将max_percentile参数值设置为PERCENTILE_MID（表示0.99999）。

4. 若按照3中的量化配置进行量化后，精度满足要求，则调参结束，否则继续5。
5. 按照量化参数的可选配置项，遍历表4-22中的参数，并按“配置说明”配置参数（可根据calibration时间要求，调整搜索空间大小）。

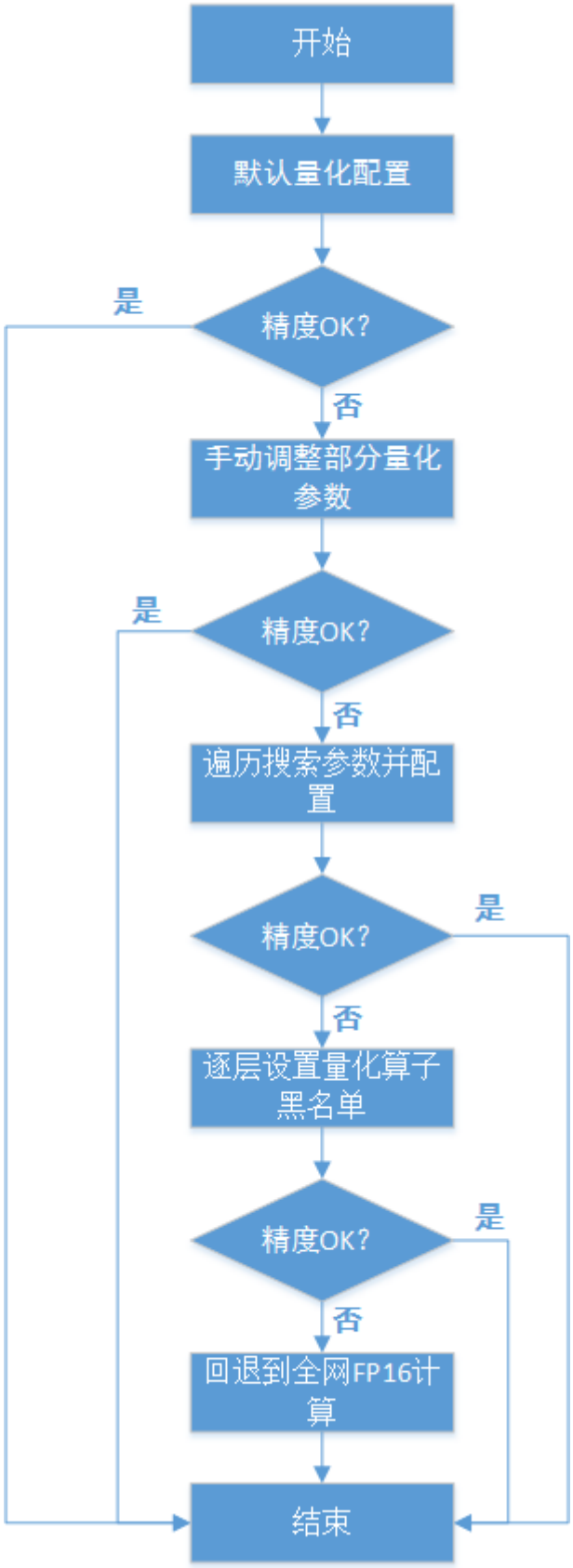
表 4-22 参数配置

参数	配置说明
quantize_algo	配置为HALF_OFFSET，表示数据采用有偏移模式，权重采用无偏移模式。
weight_type	配置为VECTOR_TYPE，表示对于卷积算子，可能有多个卷积核的情况下，一个卷积核对应一组量化参数。
bin	在计算散度过程中需要统计数据直方图，该参数的值决定了直方图统计的最大值。如果不配置或者配置为0，则使用默认配置150。建议选100, 150, 200这三档。
type	不同的散度类型对应的计算方式不一样。默认为KL。 ● KL：Kullback-Leibler Divergence； ● SYMKL：Symmetric Kullback-Leibler Divergence； ● JSD：Jensen-Shannon Divergence。
● start_ratio ● end_ratio ● step_ratio	start_ratio决定搜索开始的位置，end_ratio决定搜索结束的位置，step_ratio决定搜索步长。 推荐配置有两组 ● start_ratio:0.7 end_ratio:1.3 step_ratio:0.01 ● start_ratio:0.3 end_ratio:1.7 step_ratio:0.01
max_percentile	在从大到小排序的一组数中，决定取第多少大的数。 建议配置为1.0, 0.99999这两档。
batch_count	决定量化的读取的校准集图片数量。 可配置为大于1的整数，建议选10, 20, 50这几档。

6. 若按照5中的量化配置进行量化后，精度满足要求，则调参结束，否则继续7。
7. 通过exclude_op参数设置量化算子黑名单，从首层开始逐层将算子设置成算子黑名单，设置成算子黑名单的算子不进行量化，通过排除法，判断是哪个算子的量化对精度有影响。

   除设置到量化算子黑名单的算子不进行量化，其它算子默认进行量化，这时会存在int8计算和FP16计算混合的情况。
8. 若按照7中的量化配置进行量化后，精度满足要求，则调参结束，否则表明量化对精度没有影响，无需设置量化，去除量化配置，退回全网FP16的计算。

图 4-2 调参流程


开始
默认量化配置
精度OK?
是
否
手动调整部分量化参数
精度OK?
是
否
遍历搜索参数并配置
精度OK?
是
否
逐层设置量化算子黑名单
精度OK?
是
否
回退到全网FP16计算
结束

5 FAQ

5.1 DDK 安装用户为 HwHiAiUser 时，日志未输出到屏幕

DDK安装用户为HwHiAiUser，该场景下日志默认输入到host侧/var/dlog目录中，也可以通过设置如下环境变量，使日志直接输出到屏幕上。

```
export  SLOG_PRINT_TO_STDOUT=1
```

若DDK安装用户非HwHiAiUser，该场景下日志信息会输出到屏幕上。

5.2 DDK 所在服务器操作系统以及架构为 Arm（aarch64），模型转换耗时较长

若DDK所在服务器操作系统以及架构为Arm（aarch64），如果模型转换的耗时较长，可以使用numactl工具指定CPU核后进行模型转换，步骤如下：

1. 以DDK安装用户登录DDK所在服务器，执行**su root**命令切换到root用户。
2. 确保Host侧服务器已连接网络后，执行以下命令安装numactl工具。
   ```
   yum -y install numactl
   ```
3. 切换到DDK安装用户执行如下命令，通过**numactl -C**指定编号16到31的CPU核来处理模型转换操作。

   此处建议指定编号16到31的CPU核，处理性能更好，用户也可以根据实际情况修改。
   ```
   numactl -C 16-31 --localalloc omg <args>
   ```

5.3 模型转换时提示 Unrecognized layer:xxx, layer type xxx 错误

问题描述

使用omg命令转换模型，模型转换失败时，提示"Type XXX unsupported"、"Unrecognized layer:xxx, layer type XXX"等信息，如图5-1所示。

图 5-1 模型转换失败提示信息

```
[INFO] FMK:2019-07-04-06:47:01.467.332 CheckFlags:framework/domi/omg_main/main.cpp:289:"domi will run without encrypt!"
[INFO] FMK:2019-07-04-06:47:01.471.981 LoadCustomOpLib:framework/domi/omg_main/main.cpp:619:"plugin load lib_caffe_parser.so success."
[INFO] FMK:2019-07-04-06:47:01.476.887 Register:framework/domi/omg/../omg/parser/op_parser_factory.cpp:14:"register caffe parser adapter success"
[INFO] FMK:2019-07-04-06:47:01.477.319 LoadPluginLib:framework/domi/omg_main/main.cpp:655:"load plugin libfusion_te.so success."
[INFO] FMK:2019-07-04-06:47:01.477.429 Generate:framework/domi/omg/omg.cpp:707:"set rtContext RT_CTX_GEN_MODE..."
[INFO] RUNTIME:2019-07-04-06:47:01.477.451 23143 runtime/feature/src/stream.cc:101 Setup:alloc stream id in default mode
[INFO] RUNTIME:2019-07-04-06:47:01.477.457 23143 runtime/feature/src/stream.cc:114 Setup:streamId_=384, firstTaskId_=65535, IsTaskSink=0
[INFO] RUNTIME:2019-07-04-06:47:01.477.485 23143 runtime/feature/src/stream.cc:101 Setup:alloc stream id in default mode
[INFO] RUNTIME:2019-07-04-06:47:01.477.491 23143 runtime/feature/src/stream.cc:114 Setup:streamId_=640, firstTaskId_=65535, IsTaskSink=0
[INFO] CCE:2019-07-04-06:47:01.477.512 cce/common/cce_sys.cc:34 cceSysInit cceSysInit begin!
[INFO] CCE:2019-07-04-06:47:01.478.065 cce/common/cce_sys.cc:43 cceSysInit cceSysInit success!
[INFO] CCE:2019-07-04-06:47:01.478.074 cce/optimizer/optimizer_sys.cc:29 optimizerSysInit optimizerSysInit begin!
[INFO] CCE:2019-07-04-06:47:01.478.202 cce/optimizer/optimizer_sys.cc:33 optimizerSysInit optimizerSysInit success!
[INFO] CCE:2019-07-04-06:47:01.478.209 cce/common/cce_sys.cc:46 cceSysInit optimizerSysInit success!
[INFO] FMK:2019-07-04-06:47:01.478.224 Parse:framework/domi/omg/parser/caffe/caffe_parser.cpp:733:"Caffe Parse model file deploy_mylenet_1.prototxt"
[INFO] FMK:2019-07-04-06:47:01.479.405 CheckTypeSupported:framework/domi/omg/../omg/pre_checker/pre_checker.cpp:312:"Type Reduction unsupported."
[INFO] FMK:2019-07-04-06:47:01.479.514 ConvertDim:framework/domi/common/op/ge_op_utils.cpp:2055:"Convert format , src size 4 <3 ,not need convert"
[INFO] RUNTIME:2019-07-04-06:47:01.479.595 23143 runtime/feature/src/logger.cc:262 HostMalloc:host memory malloc, size = 92
[INFO] RUNTIME:2019-07-04-06:47:01.479.619 23143 runtime/feature/src/logger.cc:270 HostMalloc:host memory alloc ok
[ERROR] FMK:2019-07-04-06:47:01.480.290 AddNode:framework/domi/omg/parser/caffe/caffe_parser.cpp:233:"Unrecognized layer:reduction, layer type Reduction"
[ERROR] FMK:2019-07-04-06:47:01.480.299 Parse:framework/domi/omg/parser/caffe/caffe_parser.cpp:812:"Caffe parser add node fail."
[INFO] FMK:2019-07-04-06:47:01.480.590 SaveJsonToFile:framework/domi/omg/../omg/model/model_saver.cpp:40:"file check_result.json does not exit, it will be created.
"
[ERROR] FMK:2019-07-04-06:47:01.480.600 SaveJsonToFile:framework/domi/omg/../omg/model/model_saver.cpp:47:"open file failed. file path : check_result.json"
[ERROR] FMK:2019-07-04-06:47:01.480.606 Save:framework/domi/omg/../omg/pre_checker/pre_checker.cpp:293:"Save failed."
[ERROR] FMK:2019-07-04-06:47:01.480.619 Generate:framework/domi/omg/omg.cpp:724:"Generate pre-checking report failed."
[ERROR] FMK:2019-07-04-06:47:01.480.757 main:framework/domi/omg_main/main.cpp:815:"OMG Generate execute failed!!"
OMG generate offline model failed. Please see the log or pre-checking report for more details.
```

说明

图5-1中的提示信息是以Reduction类型的算子为例，请以实际使用的模型为准。

解决方法

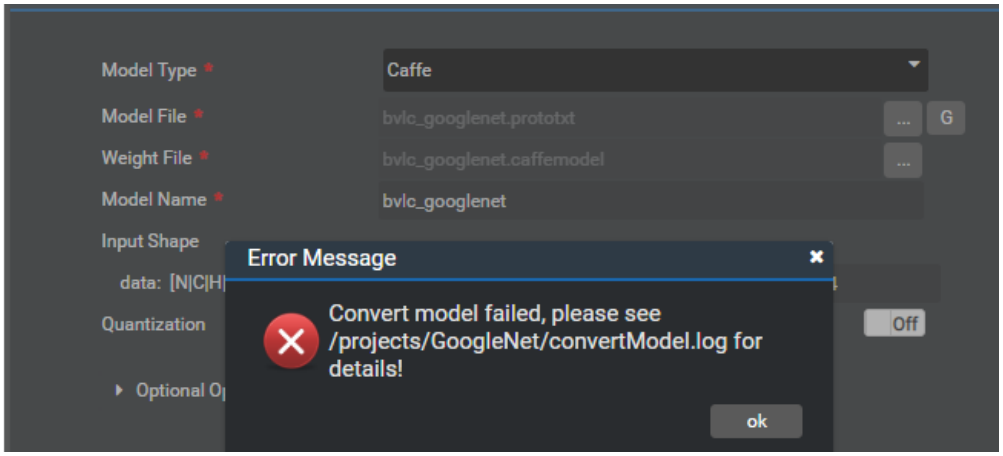
产生此错误原因是原始模型中有Ascend 310不识别的算子，需要用户根据《TE自定义算子开发指导》开发不识别的算子、开发算子插件、加载算子插件转换模型。

5.4 模型转换时提示 It is recommended to convert layers-structure to layer-structure by caffe tool 错误

问题描述

从https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet下载了googlenet的prototxt与caffemodel，导入时失败，如图5-2所示。

图 5-2 模型转化失败



“convertModel.log”日志报错如下：

```
[ERROR] FMK:2018-12-22-23:47:48.147.318 ConvertNetParameter:framework/domi/omg/parser/caffe/caffe_parser.cpp:776:"The weight file is consisted of layers-structure which is deprecated in caffe and unsupport in OMG. It is recommended to convert layers-structure to layer-structure by caffe tool. Error Code:0xFFFFFFFF(failed)"
```

解决方法

产生此错误原因为模型文件版本过低，需要用Caffe提供的工具将模型prototxt文件与caffemodel文件升级为最新版本。Caffe工具可从**Link**获取。

步骤1 用户自行下载Caffe的upgrade_net_proto_text工具与upgrade_net_proto_binary工具至Mind Studio所在服务器任一目录。

步骤2 参见**Link**中的说明自行编译获得**upgrade_net_proto_text**和**upgrade_net_proto_binary**工具。

步骤3 使用upgrade_net_proto_text工具升级prototxt。

upgrade_net_proto_text *model_old.prototxt model_new.prototxt*

步骤4 使用upgrade_net_proto_binary升级caffemodel。

upgrade_net_proto_binary *model_old.caffemodel model.new.caffemodel*

步骤5 使用转换后的protoxt文件与caffemodel文件重新进行模型导入。

----结束

6 附录

6.1 修订记录

文档版本	发布日期	修改说明
01	2020-05-09	第一次正式发布。