

CANN  
5.0.4.6

# ATC 工具使用指南 (开放态)

文档版本	01
发布日期	2023-03-17



**版权所有 © 华为技术有限公司 2023。保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## **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暗示的声明或保证。

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。

---

# 前言

## 概述

本文介绍如何将开源框架的网络模型（如Caffe、TensorFlow等）以及Ascend IR定义的单算子描述文件，通过昇腾张量编译器ATC（Ascend Tensor Compiler）转换成昇腾AI处理器支持的离线模型，并对模型转换过程中的参数以及AIPP等功能进行详细介绍。

## 读者对象

本文档适用于使用ATC工具进行模型转换的人员，通过本文档您可以达成：

- 了解不同框架原始网络模型转成昇腾AI处理器离线模型的方法。
- 能够基于本文档中的参数，转成满足不同定制要求的离线模型。
- 掌握AIPP配置文件的配置方法，以及如何使能AIPP功能。

掌握以下经验和技能可以更好地理解本文档：

- 熟悉Linux基本命令。
- 对机器学习、深度学习有一定的了解。

目 录

前言.....	ii
1 学习向导.....	1
2 环境搭建.....	2
3 快速入门.....	5
4 ATC 简介.....	9
4.1 工具介绍.....	9
4.2 运行流程.....	12
4.3 关键概念.....	13
5 初级功能.....	16
5.1 原始模型文件或离线模型转成 json 文件.....	16
5.2 离线模型支持动态 BatchSize/动态分辨率.....	17
5.3 离线模型支持动态维度.....	18
5.4 自定义离线模型的输入输出数据类型.....	19
5.5 借助离线模型查看软件基础版本号.....	20
6 高级功能.....	22
6.1 AIPP 使能.....	22
6.1.1 什么是 AIPP.....	22
6.1.2 如何使能 AIPP.....	23
6.1.3 AIPP 配置示例.....	29
6.1.3.1 静态 AIPP 配置示例.....	29
6.1.3.2 动态 AIPP 配置示例.....	31
6.1.4 色域转换配置说明.....	33
6.1.5 归一化配置说明.....	56
6.1.6 Crop/Padding 配置说明.....	56
6.1.7 AIPP 对模型输入大小的校验说明.....	57
6.1.8 配置文件模板.....	57
6.1.9 典型场景样例参考.....	62
6.1.9.1 YUV400_U8 转 GRAY 格式.....	62
6.1.9.2 YUV420SP_U8 转 BGR 格式.....	62
6.1.9.3 RGB888_U8 转 RGB ( 或 BGR ) 格式.....	63
6.2 单算子模型转换.....	64

6.2.1 什么是单算子描述文件.....	64
6.2.2 如何将算子描述文件转成离线模型.....	64
6.2.3 配置文件样例.....	64
6.2.3.1 单算子描述文件配置.....	64
6.2.3.2 多组算子描述文件配置.....	70
6.2.3.3 动态 Shape 单算子描述文件配置.....	71
6.2.4 描述文件参数说明.....	73
<b>7 参数说明.....</b>	<b>79</b>
7.1 参数概览.....	79
7.2 基础功能参数.....	85
7.2.1 总体选项.....	85
7.2.1.1 --help 或 --h.....	85
7.2.1.2 --mode.....	86
7.2.2 输入选项.....	88
7.2.2.1 --model.....	88
7.2.2.2 --weight.....	88
7.2.2.3 --om.....	89
7.2.2.4 --framework.....	91
7.2.2.5 --input_format.....	92
7.2.2.6 --input_shape.....	93
7.2.2.7 --input_shape_range.....	94
7.2.2.8 --dynamic_batch_size.....	95
7.2.2.9 --dynamic_image_size.....	97
7.2.2.10 --dynamic_dims.....	98
7.2.2.11 --singleop.....	99
7.2.3 输出选项.....	101
7.2.3.1 --output.....	101
7.2.3.2 --output_type.....	102
7.2.3.3 --check_report.....	103
7.2.3.4 --json.....	104
7.2.4 目标芯片选项.....	105
7.2.4.1 --soc_version.....	106
7.2.4.2 --core_type.....	107
7.2.4.3 --aicore_num.....	107
7.3 高级功能参数.....	108
7.3.1 功能配置选项.....	108
7.3.1.1 --out_nodes.....	108
7.3.1.2 --input_fp16_nodes.....	110
7.3.1.3 --insert_op_conf.....	110
7.3.1.4 --op_name_map.....	111
7.3.1.5 --is_input_adjust_hw_layout.....	112
7.3.1.6 --is_output_adjust_hw_layout.....	113

7.3.2 模型调优选项.....	113
7.3.2.1 --disable_reuse_memory.....	113
7.3.2.2 --fusion_switch_file.....	114
7.3.2.3 --enable_scope_fusion_passes.....	118
7.3.2.4 --enable_small_channel.....	119
7.3.2.5 --buffer_optimize.....	120
7.3.2.6 --mdl_bank_path.....	120
7.3.3 算子调优选项.....	121
7.3.3.1 --precision_mode.....	122
7.3.3.2 --op_precision_mode.....	124
7.3.3.3 --modify_mixlist.....	125
7.3.3.4 --op_select_implmode.....	127
7.3.3.5 --optypelist_for_implmode.....	129
7.3.3.6 --keep_dtype.....	130
7.3.3.7 --auto_tune_mode.....	131
7.3.3.8 --op_bank_path.....	134
7.3.3.9 --op_debug_level.....	135
7.3.4 调试选项.....	136
7.3.4.1 --dump_mode.....	136
7.3.4.2 --log.....	137
7.3.4.3 --debug_dir.....	139
7.3.4.4 --op_compiler_cache_mode.....	140
7.3.4.5 --op_compiler_cache_dir.....	141
7.3.4.6 --display_model_info.....	142
<b>8 定制网络专题.....</b>	<b>144</b>
8.1 定制网络修改 ( Caffe ) .....	144
8.1.1 简介.....	144
8.1.2 扩展算子列表.....	145
8.1.3 扩展算子规则.....	146
8.1.4 样例参考.....	157
8.1.4.1 FasterRCNN 网络模型 prototxt 修改.....	157
8.1.4.2 YOLOv3 网络模型 prototxt 修改.....	159
8.1.4.3 YOLOv2 网络模型 prototxt 修改.....	161
8.1.4.4 SSD 网络模型 prototxt 修改.....	163
8.1.4.5 BatchedMatMul 算子 prototxt 修改.....	164
8.1.4.6 SENet 网络模型 prototxt 修改.....	165
8.2 定制网络修改 ( TensorFlow ) .....	166
8.2.1 概述.....	166
8.2.2 编译可执行文件.....	167
8.2.3 转换模型.....	168
8.2.4 FAQ.....	169

8.2.4.1 使用 bazel 编译工具编译时提示 “An error occurred during the fetch of repository 'io_bazel_rules_docker'”，编译失败.....	170
8.2.4.2 使用 pip3.7.5 install 软件时提示" Could not find a version that satisfies the requirement xxx".....	171
8.2.4.3 获取 Switch_v1.pb 网络模型.....	172
8.2.4.4 获取 xlacompile.patch 补丁文件.....	173
<b>9 参考.....</b>	<b>178</b>
9.1 算子规格参考.....	178
9.1.1 Caffe 框架算子规格.....	178
9.1.2 TensorFlow 框架算子规格.....	178
9.1.3 ONNX 算子规格.....	178
9.2 错误码参考.....	179
9.2.1 Command Line Errors.....	179
9.2.1.1 E10001 Invalid Argument.....	179
9.2.1.2 E10002 Invalid --input_shape Argument.....	179
9.2.1.3 E10003 Invalid Argument.....	179
9.2.1.4 E10004 Invalid Argument.....	180
9.2.1.5 E10005 Invalid Argument.....	180
9.2.1.6 E10006 Invalid Argument.....	180
9.2.1.7 E10007 Invalid Required Argument.....	180
9.2.1.8 E10008 Invalid Argument.....	181
9.2.1.9 E10009 Invalid Dynamic Shape Argument.....	181
9.2.1.10 E10010 Invalid --log Argument.....	181
9.2.1.11 E10011 Invalid --input_shape Argument.....	182
9.2.1.12 E10012 Invalid --input_shape Argument.....	182
9.2.1.13 E10013 Invalid Argument.....	182
9.2.1.14 E10014 Invalid Argument.....	183
9.2.1.15 E10015 Invalid Argument.....	183
9.2.1.16 E10016 Invalid Argument, Node Not Found.....	183
9.2.1.17 E10017 Invalid Argument, Node Not Found.....	183
9.2.1.18 E10018 Invalid Dynamic Shape Argument.....	184
9.2.1.19 E10019 Invalid --input_shape Argument.....	184
9.2.1.20 E10020 Invalid --dynamic_image_size Argument.....	184
9.2.1.21 E10021 Invalid Argument.....	185
9.2.1.22 E10022 Invalid Argument.....	185
9.2.1.23 E10023 Invalid --singleop Argument.....	185
9.2.1.24 E10024 Invalid --singleop Argument.....	185
9.2.1.25 E10025 Invalid --singleop Argument.....	186
9.2.1.26 E10026 Invalid --singleop Argument.....	186
9.2.1.27 E10027 Invalid --singleop Argument.....	186
9.2.1.28 E10029 Invalid --singleop Argument.....	187
9.2.1.29 E10030 Invalid --singleop Argument.....	187
9.2.1.30 E10031 Invalid --input_shape Argument.....	187

9.2.1.31 E10034 Invalid --input_fp16_nodes Argument.....	188
9.2.1.32 E10035 Invalid Dynamic Shape Argument.....	188
9.2.1.33 E10036 Invalid Dynamic Shape Argument.....	188
9.2.1.34 E10037 Invalid Dynamic Shape Argument.....	189
9.2.1.35 E10038 Invalid Dynamic Shape Argument.....	189
9.2.1.36 E10039 Invalid Dynamic Shape Argument.....	189
9.2.1.37 E10040 Invalid --input_shape Argument.....	190
9.2.1.38 E10041 Invalid --framework or --model Argument.....	190
9.2.1.39 E10045 Invalid Dynamic Shape Argument.....	190
9.2.1.40 E10046 Invalid Dynamic Shape Argument.....	191
9.2.1.41 E10047 Invalid Argument.....	191
9.2.1.42 E10048 Invalid --input_shape_range Argument.....	191
9.2.1.43 E10049 Invalid --input_shape_range Argument.....	192
9.2.1.44 E10050 Invalid --input_shape_range Argument.....	192
9.2.1.45 E10052 Invalid AIPP Configuration.....	192
9.2.1.46 E10410 Invalid Argument.....	192
9.2.1.47 E11001 Caffe Model Data Error.....	193
9.2.1.48 E11002 Caffe Model Data Error.....	193
9.2.1.49 E11003 Caffe Model Data Error.....	193
9.2.1.50 E11004 Caffe Model Data Error.....	194
9.2.1.51 E11005 Invalid --input_shape Argument.....	194
9.2.1.52 E11008 Caffe Model Data Error.....	194
9.2.1.53 E11009 Unsupported Caffe Operator.....	195
9.2.1.54 E11012 Caffe Model Data Error.....	195
9.2.1.55 E11014 Caffe Model Data Error.....	195
9.2.1.56 E11015 Caffe Model Data Error.....	195
9.2.1.57 E11016 Invalid --out_node Argument.....	196
9.2.1.58 E11017 Invalid --out_node Argument.....	196
9.2.1.59 E11018 Caffe Model Data Error.....	196
9.2.1.60 E11021 Caffe Model Data Error.....	197
9.2.1.61 E11022 Caffe Model Data Error.....	197
9.2.1.62 E11023 Caffe Model Data Error.....	197
9.2.1.63 E11024 Caffe Model Data Error.....	197
9.2.1.64 E11027 Caffe Model Data Error.....	198
9.2.1.65 E11029 Caffe Model Data Error.....	198
9.2.1.66 E11032 Caffe File Error.....	198
9.2.1.67 E11033 The Caffe weight file is invalid.....	199
9.2.1.68 E11035 Caffe Model Data Error.....	199
9.2.1.69 E11036 Caffe Model Data Error.....	199
9.2.1.70 E11037 Caffe Model Data Error.....	199
9.2.1.71 E12009 TensorFlow Model Data Error.....	200
9.2.1.72 E12010 TensorFlow Model Data Error.....	200



9.2.1.73 E12013 TensorFlow Model Data Error.....	200
9.2.1.74 E12029 TensorFlow Model Data Error.....	201
9.2.1.75 E16001 ONNX Model Data Error.....	201
9.2.1.76 E16002 ONNX Model Data Error.....	201
9.2.1.77 E16004 ONNX Model Data Error.....	202
9.2.1.78 E16005 ONNX Model Data Error.....	202
9.2.1.79 E20100 Invalid GE Initialization Argument.....	202
9.2.1.80 E20101 Invalid GE Initialization Argument.....	202
9.2.1.81 E20102 Invalid GE Initialization Argument.....	203
9.2.1.82 E20103 Invalid Platform Initialization Argument.....	203
9.2.1.83 W21000 Empty Path.....	203
9.2.1.84 E21001 Failure to Open File.....	204
9.2.1.85 E21002 Failure to Read File.....	204
9.2.1.86 E21003 Empty Path.....	204
9.2.1.87 W40001 Invalid Path.....	205
9.2.1.88 W40002 Failure to Create Directory.....	205
9.2.1.89 E40000 Failure to Import te.platform.log_util.....	205
9.2.1.90 E40001 Failure to Import Python Module.....	205
9.2.1.91 E40002 Failure to Call Python Function.....	206
9.2.1.92 E40003 Failure to Import Auto Tiling Manager of Python Module.....	206
9.2.1.93 E40004 Failure to Import RL Tiling Manager of Python Module.....	206
9.2.1.94 E40008 Failure to Import Python Module.....	207
9.2.1.95 E40010 Invalid Auto Tune Mode.....	207
9.2.1.96 E40011 Invalid Argument.....	207
9.2.1.97 EE1000 Invalid SoC Version.....	208
9.2.1.98 E76002 ONNX Model Data Error.....	208
9.2.2 Resource Errors.....	208
9.2.2.1 E10044 Insufficient Memory.....	208
9.2.2.2 E19022 Insufficient Memory.....	209
9.2.2.3 E19023 Too Large OM Model.....	209
9.2.3 Invalid Argument.....	209
9.2.3.1 E10051 Invalid --job_id Argument.....	209
9.2.3.2 E10401 Invalid Operator Input Count.....	210
9.2.3.3 E10402 Invalid Input Buffer Allocation for Operator Execution.....	210
9.2.3.4 E10403 Invalid Operator Output Count.....	210
9.2.3.5 E10404 Invalid Output Buffer Allocation for Operator Execution.....	210
9.2.3.6 E10405 Inconsistent Input Buffer Count and Input Tensor Count for Operator Execution.....	211
9.2.3.7 E10406 Inconsistent Output Buffer Count and Output Tensor Count for Operator Execution.....	211
9.2.3.8 E14001 Invalid Argument for Operator Compilation.....	211
9.2.3.9 E19009 Operator Name Conflict.....	212
9.2.3.10 E19014 Operator Data Verification Failure.....	212
9.2.3.11 E19018 Failure to Parse File.....	212

9.2.3.12 E19025 Input Tensor Error.....	213
9.2.4 System Support Errors.....	213
9.2.4.1 E10501 Unsupported Operator.....	213
9.2.4.2 E13002 Unsupported Operator.....	213
9.2.4.3 E13003 Unsupported Operator.....	214
9.2.4.4 E19010 Unsupported Operator.....	214
9.2.4.5 EZ0501 Unsupported Operator.....	214
9.2.4.6 EZ3002 Unsupported Operator.....	214
9.2.4.7 EZ3003 Unsupported Operator.....	215
9.2.4.8 EZ9010 Unsupported Operator.....	215
9.2.5 Configuration Errors.....	215
9.2.5.1 E12004 Operator Prototype Registration Error.....	215
9.2.5.2 E19024 Invalid Environment Variable.....	216
9.2.6 File Errors.....	216
9.2.6.1 E19000 Invalid Directory.....	216
9.2.6.2 E19001 Failure to Open File.....	216
9.2.6.3 E19002 Too Long File Directory.....	217
9.2.6.4 E19003 Failure to Read File.....	217
9.2.6.5 E19004 Failure to Write File.....	217
9.2.6.6 E19005 Failure to Parse File.....	217
9.2.6.7 E19015 File size invalid.....	218
9.2.7 Invalid Path Name.....	218
9.2.7.1 E19026 Input Path Name Error.....	218
9.2.8 Minor Error.....	218
9.2.8.1 W11001 Operator Missing High-Priority Performance.....	218
9.2.9 ACL API Errors.....	219
9.2.9.1 EH0001 Invalid Argument.....	219
9.2.9.2 EH0002 Null Pointer.....	219
9.2.9.3 EH0003 Invalid Path.....	219
9.2.9.4 EH0004 Invalid File.....	220
9.2.9.5 EH0005 Invalid AIPP Argument.....	220
9.2.9.6 EH0006 Feature Unsupported.....	220
9.2.10 Environment Errors.....	220
9.2.10.1 E30000 Invalid Configuration File.....	221
9.2.10.2 E30001 Invalid Configuration File.....	221
9.2.10.3 E30002 Invalid Configuration File.....	221
9.2.10.4 EE2000 Device Error.....	221
9.2.10.5 EE2001 Failure to Allocate Memory.....	222
9.2.10.6 EE3001 Driver Failure.....	222
9.2.10.7 EI0001 Invalid Environment Variable Configuration.....	222
9.2.10.8 EI0002 SO File Not Found.....	223
9.2.10.9 EI0004 Invalid Ranktable Configuration.....	223

9.2.10.10 EI0007 Allocation Failure.....	223
9.2.11 Operator Compilation Errors.....	223
9.2.11.1 E20001 Failure to Precompile Op.....	224
9.2.11.2 E20003 Failure to Compile Op.....	224
9.2.11.3 E20004 Failure to Compile Op.....	224
9.2.11.4 E40009 Failure to Compile Op.....	224
9.2.12 Invalid Operator Arguments.....	225
9.2.12.1 E20002 Failure to Precompile Op.....	225
9.2.12.2 E20006 Failure to Calculate Data Edge Size.....	225
9.2.12.3 E40005 Failure to Execute Function check_supported.....	225
9.2.12.4 E40006 Failure to Obtain Op Format.....	226
9.2.12.5 E40007 Failure to Precompile Op.....	226
9.2.12.6 EI0003 Invalid Collective Communication Op Argument.....	226
9.2.12.7 EI0005 Inconsistent Collective Communication Arguments Between Ranks.....	227
9.2.12.8 EI0006 Invalid Collective Communication Allocation.....	227
9.2.13 Fusion Pass Errors.....	227
9.2.13.1 E20007 Failure to Execute Fusion Pass.....	227
9.2.13.2 E20008 Failure to Execute Fusion Pass.....	228
9.2.13.3 E20009 Failure to Execute Fusion Pass.....	228
9.2.14 Task Generation Errors.....	228
9.2.14.1 E20005 Failure to Generate Task.....	228
9.2.15 TBE Compiler Errors.....	229
9.2.15.1 EB0000 Invalid IR.....	229
9.2.16 Task Schedule Execution Errors.....	229
9.2.16.1 EE4001 Model Execution Error.....	229
9.2.16.2 EE4002 Model Execution Error.....	229
9.2.16.3 EE4003 Profiling Execution Error.....	230
9.2.16.4 EE4004 Profiling Execution Error.....	230
9.2.16.5 EE5001 Task Schedule Resource Error.....	230
9.2.16.6 EE5002 Task Schedule Resource Error.....	230
9.2.17 Default Error.....	231
9.2.17.1 E19999 System Terminated.....	231
9.3 一键收集故障信息.....	231
9.4 开启 Cast 算子自动插入特性.....	234
<b>10 FAQ.....</b>	<b>237</b>
10.1 开发环境架构为 Arm ( aarch64 ) , 模型转换耗时较长.....	237
10.2 使用 AIPP 色域转换模型时如何判断视频流的格式标准.....	238
10.3 如何确定原始框架网络模型中的算子与昇腾 AI 处理器或 BS9SX1A AI 处理器支持的算子的对应关系....	238
10.4 目标模型中包含算子原型库未注册的算子.....	239
10.5 目标模型中因为未找到算子归属引擎导致转换失败.....	240
10.6 缺省 input_shape 参数导致转换失败.....	240
10.7 out_nodes 参数指定的输出节点不存在.....	241

10.8 out_nodes 参数输入校验失败导致转换失败.....	242
10.9 input_fp16_nodes 参数指定的节点名字不存在.....	242
10.10 算子 infershape 失败导致模型转换失败.....	243
10.11 算子找不到算子信息库导致模型转换失败.....	243
10.12 insert_op_conf 参数文件路径不存在导致转换失败.....	244
10.13 insert_op_conf 参数指定的 aipp 配置文件校验失败导致转换失败.....	244
10.14 dynamic_image_size 或 dynamic_batch_size 设置分辨率数值过大或档位过多导致运行环境异常缓慢.....	245
10.15 算子信息库中缺少算子.....	246

# 1 学习向导

## 针对新手

3 快速入门	4 ATC简介	5 初级功能
本章节以ATC工具支持的所有框架网络模型为例，简单介绍如何进行基础功能的模型转换。	介绍ATC工具功能架构、运行流程以及关键概念。	简单介绍其他功能点场景下，比如将模型转成json文件，离线模型支持动态batch，动态分辨率等场景，如何组合各种ATC参数转换成满足要求的离线模型。

## 适合专家

6.1 AIPP使能	6.2 单算子模型转换	8 定制网络专题
介绍什么是AIPP、模型转换时如何使能AIPP、根据配置文件模板如何构造AIPP配置文件，以及根据色域转换功能如何输出满足要求的图片数据等功能，并给出典型场景写下的配置示例。	介绍什么是单算子描述文件、如何构造单算子描述文件，以及如何将该文件转成适配昇腾AI处理器的离线模型，用于验证单算子功能。	如果网络包含了一些原始Caffe中没有定义的算子结构，或者原始模型为TensorFlow框架时，含有控制流算子，上述场景是不能直接使用ATC工具进行模型转换的，需要定制Caffe框架的prototxt或者将TensorFlow框架控制流算子转成函数类算子。本章节给出详细说明。

# 2 环境搭建

## 获取 ATC 工具

参见《[CANN 软件安装指南](#)》进行开发环境搭建，并确保开发套件包Ascend-cann-toolkit安装完成。该场景下ATC工具安装在“*Ascend-cann-toolkit安装目录/ascend-toolkit/latest/bin*”下。

## 设置环境变量

### 须知

- 若开发环境架构为Arm（aarch64），模型转换耗时较长，则可以参考[10.1 开发环境架构为Arm（aarch64），模型转换耗时较长](#)解决。
- 该工具对Python版本的支持请参见《[CANN 软件安装指南](#)》中的“通过命令行方式安装>安装开发环境>安装OS依赖>依赖列表”章节，本手册以Python3.7.5为例进行介绍，相应环境变量和安装命令以实际安装Python版本为准。
- 使用ATC工具进行模型转换的过程中，会自动将ATC工具所在位置“../python/site-packages”目录下算子编译依赖的TBE Python库写入PYTHONPATH环境变量。  
若算子实现时用户引入了TBE模块外的其他Python依赖，请自行添加PYTHONPATH的环境变量，配置引入的Python依赖所在路径，如下所示：  
export PYTHONPATH=xxx:\$PYTHONPATH

### 1. 必选环境变量

#### – 设置公共环境变量

##### ■ 以root用户安装Ascend-cann-toolkit包

```
./usr/local/Ascend/ascend-toolkit/set_env.sh  
#若开发套件包Ascend-cann-toolkit在非昇腾设备上安装，则如下环境变量必须执行，用于设置动态链接库所在路径，否则无需执行  
export LD_LIBRARY_PATH=Ascend-cann-toolkit安装目录/ascend-toolkit/latest/<arch>-linux/ <arch>-linux/devlib
```

##### ■ 以非root用户安装Ascend-cann-toolkit包

```
./${HOME}/Ascend/ascend-toolkit/set_env.sh  
#若开发套件包Ascend-cann-toolkit在非昇腾设备上安装，则如下环境变量必须执行，用于设置动态链接库所在路径，否则无需执行  
export LD_LIBRARY_PATH=Ascend-cann-toolkit安装目录/ascend-toolkit/latest/<arch>-linux/ <arch>-linux/devlib
```

<arch>请替换为操作系统具体架构。

- 设置Python相关环境变量

模型编译依赖Python，以Python3.7.5为例，请以CANN软件包运行用户执行如下命令设置Python3.7.5相关环境变量

```
#如果用户环境存在多个python3版本，则指定使用python3.7.5版本
export PATH=/usr/local/python3.7.5/bin:$PATH
#设置python3.7.5库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
```

上述环境变量只在当前窗口生效，用户可以将上述命令写入~/.bashrc文件，使其永久生效，方法如下：

- a. 以安装用户在任意目录下执行`vi ~/.bashrc`，在该文件最后添加上述内容。
- b. 执行`wq!`命令保存文件并退出。
- c. 执行`source ~/.bashrc`使环境变量生效。

## 2. 可选环境变量

a. 日志落盘、打屏与重定向。

■ 日志落盘：

atc命令执行过程中，日志默认落盘到\$HOME/ascend/log/plog/plog-pid\_\*.log（pid代表进程ID，“\*”表示该日志文件创建时的时间戳）路径，由于7.3.4.2 --log默认值为null，既不输出日志，若该路径存在日志信息，则为atc进程之外的其他日志信息，比如依赖Python相关信息。

若想要日志体现atc进程相关信息，则模型转换时，需要设置7.3.4.2 --log参数（不能设置为null）。

■ 日志打屏：

atc命令执行过程中，日志默认不打屏，如需打屏显示，则请在执行atc命令的当前窗口设置如下环境变量，然后再执行atc命令：

```
export ASCEND_SLOG_PRINT_TO_STDOUT=1
```

关于日志的更多信息请参见《[日志参考（开放态）](#)》。若设置上述环境变量后，仍旧未打屏有效信息，则请在atc命令设置7.3.4.2 --log参数（不能设置为null）显示相应的日志级别。

■ 日志重定向

如果不想日志落盘，而是重定向到文件，则模型转换前需要设置上述的日志打屏环境变量，并且atc命令需要设置7.3.4.2 --log参数（不能设置为null），样例如下：

```
atc xxx --log=debug >log.txt
```

b. 开启算子并行编译功能。

若网络模型较大，模型转换过程中，可设置如下环境变量，开启算子的并行编译功能。

```
export TE_PARALLEL_COMPILER=xx
```

TE\_PARALLEL\_COMPILER的值代表算子并行编译进程数（配置为整数），取值范围为0~32，默认值为8，当取值=0，表示不开启算子的并行编译功能；当取值>0时，表示开启算子的并行编译。建议配置为：CPU核数\*80%/昇腾AI处理器个数，向上取整。

c. 打印模型转换过程中各个阶段的图描述信息。

```
export DUMP_GE_GRAPH=1
```

上述环境变量控制dump图的内容多少：

- 取值为1，全量dump。
- 取值为2，不含有权重等数据的基本版dump。
- 取值为3，只显示节点关系的精简版dump。

设置上述环境变量后，还可以设置如下环境变量，控制dump图的个数。

```
export DUMP_GRAPH_LEVEL=1
```

- 取值为1，dump所有图。
- 取值为2，dump除子图外的所有图，默认值为2。
- 取值为3，dump最后的生成图。

设置上述变量后，在执行atc命令的当前路径会生成如下文件：

- ge\_onnx\*.pbtxt：基于ONNX的开源模型描述结构，可以使用Netron等可视化软件打开。
- ge\_proto\*.txt：protobuf格式存储的文本文件，该文件可以转成json格式文件方便用户定位问题。该文件与ge\_onnx\*.pbtxt成对出现，但是比ge\_onnx\*.pbtxt文件会多string类型的属性信息，用户选择其中一种文件打开即可。

上述每个文件对应模型编译过程中的一个步骤，比如以ge\_onnx\_00000001\_graph\_0\_PreRunBegin.pbtxt开始，以ge\_onnx\_00000078\_graph\_0\_PreRunAfterBuild.pbtxt结尾。每个文件中包括完成该步骤所涉及的所有算子。



# 3 快速入门

本章节以各框架下模型转换为例，演示如何快速转换一个模型。

## 须知

- 使用高版本ATC工具转换的模型，在低版本环境上使用可能会出现不兼容问题，建议使用匹配的版本重新进行模型转换，如果用于想查看已有离线模型使用的ATC等基础版本信息，则请参见[7.2.3.4 --json](#)。
- 如果用户使用的网络模型中有自定义算子，则请优先参见《[TBE自定义算子开发指南（开放态）](#)》手册开发部署好自定义算子，模型转换时会优先去查找自定义算子库匹配模型文件中的算子；若匹配失败，则会去查找内置算子库。
- 如果用户使用Faster RCNN、YOLOv3、YOLOv2、SSD等Caffe框架网络模型进行模型转换，由于此类网络中包含了一些原始Caffe框架中没有定义的算子结构，如ROI Pooling、Normalize、PSROI Pooling和Upsample等。为了使昇腾AI处理器能支持这些网络，需要对原始的Caffe框架网络模型进行扩展，降低开发者开发自定义算子/开发后处理代码的工作量，详细扩展方法请参见[8.1 定制网络修改（Caffe）](#)。
- 针对TensorFlow原始网络模型，如果存在控制流算子，该类网络模型不能直接使用ATC工具进行模型转换，需要先将控制流算子的网络模型转成函数类算子的网络模型，然后利用ATC工具转换成适配昇腾AI处理器的离线模型，详细转换方式请参见[8.2 定制网络修改（TensorFlow）](#)。

## 开源框架的 Caffe 网络模型转换成离线模型

### 步骤1 获取Caffe网络模型。

您可以从以下链接中获取ResNet-50网络的模型文件（\*.prototxt）、预训练模型文件（\*.caffemodel），并以CANN软件包运行用户将获取的文件上传至开发环境任意目录，例如上传到\$HOME/module/目录下。

- ResNet-50网络的模型文件（\*.prototxt）：单击[Link](#)下载该文件。
- ResNet-50网络的预训练模型文件（\*.caffemodel）：单击[Link](#)下载该文件。

### 步骤2 执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version>
```

关于参数的详细解释以及使用方法请参见[7 参数说明](#)，请使用与芯片名相对应的 `<soc_version>` 取值进行模型转换，然后再进行推理，具体使用芯片查询方法请参见[7.2.4.1 --soc\\_version](#)。

- 步骤3** 若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

成功执行命令后，在 `--output` 参数指定的路径下，可查看离线模型（如：`caffe_resnet50.om`）。

模型编译时，若遇到AICPU算子不支持某种数据类型导致编译失败的场景，可通过启用Cast算子自动插入特性快速将输入转换为算子支持的数据类型，从而实现网络的快速打通，详细流程请参见[9.4 开启Cast算子自动插入特性](#)。

- 步骤4** （后续处理）如果想快速体验直接使用转换后的om离线模型文件进行推理，请准备好环境、om模型文件、符合模型输入要求的\*.bin格式的输入数据，单击[Link](#)，获取msame工具，参考该工具配套的README，进行体验。

----结束

## 开源框架的 TensorFlow 网络模型转换成离线模型

- 步骤1** 获取TensorFlow网络模型。

单击[Link](#)，获取ResNet V1 50网络的模型文件（\*.pb），并以CANN软件包运行用户将获取的文件上传至开发环境任意目录，例如上传到`$HOME/module/`目录下。

- 步骤2** 执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet_v1_50.pb --framework=3 --output=$HOME/module/out/tf_resnet_v1_50 --soc_version=<soc_version> --input_shape="input:1,224,224,3"
```

关于参数的详细解释以及使用方法请参见[7 参数说明](#)，请使用与芯片名相对应的 `<soc_version>` 取值进行模型转换，然后再进行推理，具体使用芯片查询方法请参见[7.2.4.1 --soc\\_version](#)。

- 步骤3** 若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

成功执行命令后，在 `--output` 参数指定的路径下，可查看离线模型（如：`tf_resnet_v1_50.om`）。

模型编译时，若遇到AICPU算子不支持某种数据类型导致编译失败的场景，可通过启用Cast算子自动插入特性快速将输入转换为算子支持的数据类型，从而实现网络的快速打通，详细流程请参见[9.4 开启Cast算子自动插入特性](#)。

- 步骤4** （后续处理）如果想快速体验直接使用转换后的om离线模型文件进行推理，请准备好环境、om模型文件、符合模型输入要求的\*.bin格式的输入数据，单击[Link](#)，获取msame工具，参考该工具配套的README，进行体验。

----结束

## ONNX 网络模型转换成离线模型

- 步骤1** 获取ONNX网络模型。

单击[Link](#)，获取ResNet-50网络的模型文件（\*.onnx），并以CANN软件包运行用户将获取的文件上传至开发环境任意目录，例如上传到\$HOME/module/目录下。

- 步骤2** 执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet50.onnx --framework=5 --output=$HOME/module/out/onnx_resnet50 --soc_version=<soc_version>
```

关于参数的详细解释以及使用方法请参见[7 参数说明](#)，请使用与芯片名相对应的<soc\_version>取值进行模型转换，然后再进行推理，具体使用芯片查询方法请参见[7.2.4.1 --soc\\_version](#)。

- 步骤3** 若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

成功执行命令后，在--output参数指定的路径下，可查看离线模型（如：onnx\_resnet50.om）。

模型编译时，若遇到AICPU算子不支持某种数据类型导致编译失败的场景，可通过启用Cast算子自动插入特性快速将输入转换为算子支持的数据类型，从而实现网络的快速打通，详细流程请参见[9.4 开启Cast算子自动插入特性](#)。

- 步骤4** （后续处理）如果想快速体验直接使用转换后的om离线模型文件进行推理，请准备好环境、om模型文件、符合模型输入要求的\*.bin格式的输入数据，单击[Link](#)，获取msame工具，参考该工具配套的README，进行体验。

----结束

## MindSpore 框架的网络模型转换成离线模型

- 步骤1** 获取MindSpore框架的网络模型。

单击[Link](#)，获取ResNet-50网络的模型文件（仅支持\*.air格式的模型文件进行模型转换），并以CANN软件包运行用户将获取的文件上传到开发环境任意路径，例如\$HOME/module/目录下。

- 步骤2** 执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/ResNet50.air --framework=1 --output=$HOME/module/out/ResNet50_mindspore --soc_version=<soc_version>
```

关于参数的详细解释以及使用方法请参见[7 参数说明](#)，请使用与芯片名相对应的<soc\_version>取值进行模型转换，然后再进行推理，具体使用芯片查询方法请参见[7.2.4.1 --soc\\_version](#)。

- 步骤3** 若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

成功执行命令后，在--output参数指定的路径下，可查看离线模型（如：ResNet50\_mindspore.om）。

模型编译时，若遇到AICPU算子不支持某种数据类型导致编译失败的场景，可通过启用Cast算子自动插入特性快速将输入转换为算子支持的数据类型，从而实现网络的快速打通，详细流程请参见[9.4 开启Cast算子自动插入特性](#)。

**步骤4** （后续处理）如果想快速体验直接使用转换后的om离线模型文件进行推理，请准备好环境、om模型文件、符合模型输入要求的\*.bin格式的输入数据，单击[Link](#)，获取msame工具，参考该工具配套的README，进行体验。

----结束

# 4 ATC 简介

工具介绍  
运行流程  
关键概念

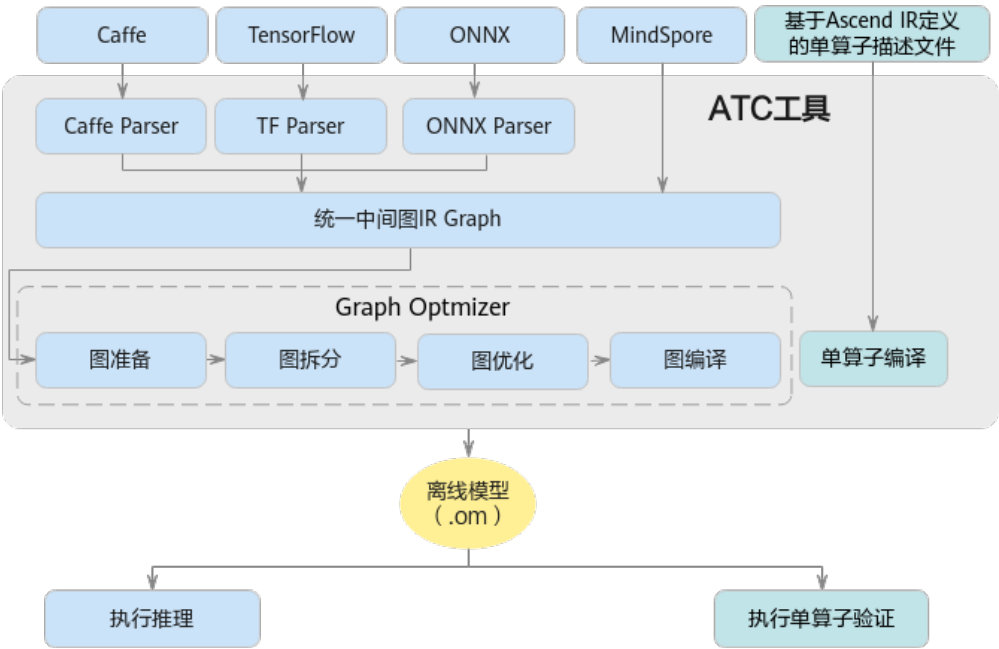
## 4.1 工具介绍

### ATC 简介

昇腾张量编译器（Ascend Tensor Compiler，简称ATC）是昇腾CANN架构体系下的模型转换工具，它可以将开源框架的网络模型或Ascend IR定义的单算子描述文件（json格式）转换为昇腾AI处理器支持的.om格式离线模型。其功能架构如图4-1所示。

模型转换过程中，ATC会进行算子调度优化、权重数据重排、内存使用优化等具体操作，对原始的深度学习模型进行进一步的调优，从而满足部署场景下的高性能需求，使其能够高效执行在昇腾AI处理器上。

图 4-1 ATC 工具功能架构



其中:

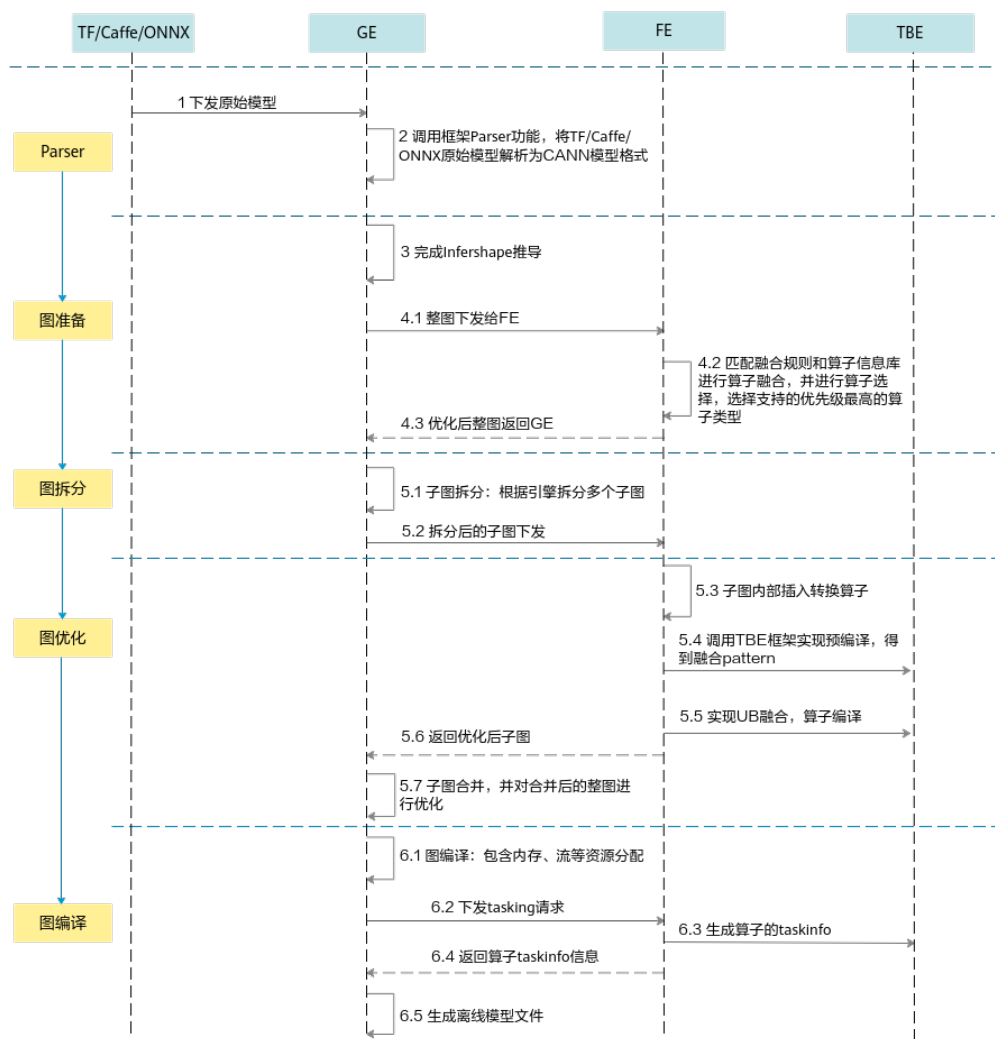
- 开源框架网络模型场景:
  - a. 开源框架网络模型经过Parser解析后, 转换为中间态IR Graph。
  - b. 中间态IR经过图准备, 图拆分, 图优化, 图编译等一系列操作后, 转成适配昇腾AI处理器的离线模型。
  - c. 转换后的离线模型上传到板端环境, 通过AscendCL接口加载模型文件实现推理过程。  
用户也可以将开源框架网络模型转换后的离线模型转成json文件, 方便文件查看, 也可以直接将开源框架网络模型通过ATC工具转成json文件。
- 单算子描述文件场景下:  
Ascend IR定义的单算子描述文件 ( json格式 ) 通过ATC工具进行单算子编译后, 转成适配昇腾AI处理器的单算子离线模型, 然后上传到板端环境, 通过AscendCL接口加载单算子模型文件用于验证单算子功能。

## 模型转换交互流程

下面以开源框架网络模型转换为.om离线模型为例, 详细介绍模型转换过程中与周边模块的交互流程。

- TBE算子模型转换交互流程

图 4-2 TBE 算子模型转换交互流程



a. 调用框架Parser功能，将主流框架的模型格式转换成CANN模型格式。

b. 图准备阶段：该阶段会完成原图优化以及InferShape推导（设置算子输出的shape和dtype）等功能。

原图优化时：GE向FE发送图优化请求，并将图下发给FE，FE匹配融合规则进行图融合，并进行算子选择，选择优先级最高的算子类型进行算子匹配，最后将优化后的整图返回给GE。

c. 图拆分阶段：GE根据图中数据将图拆分为多个子图。

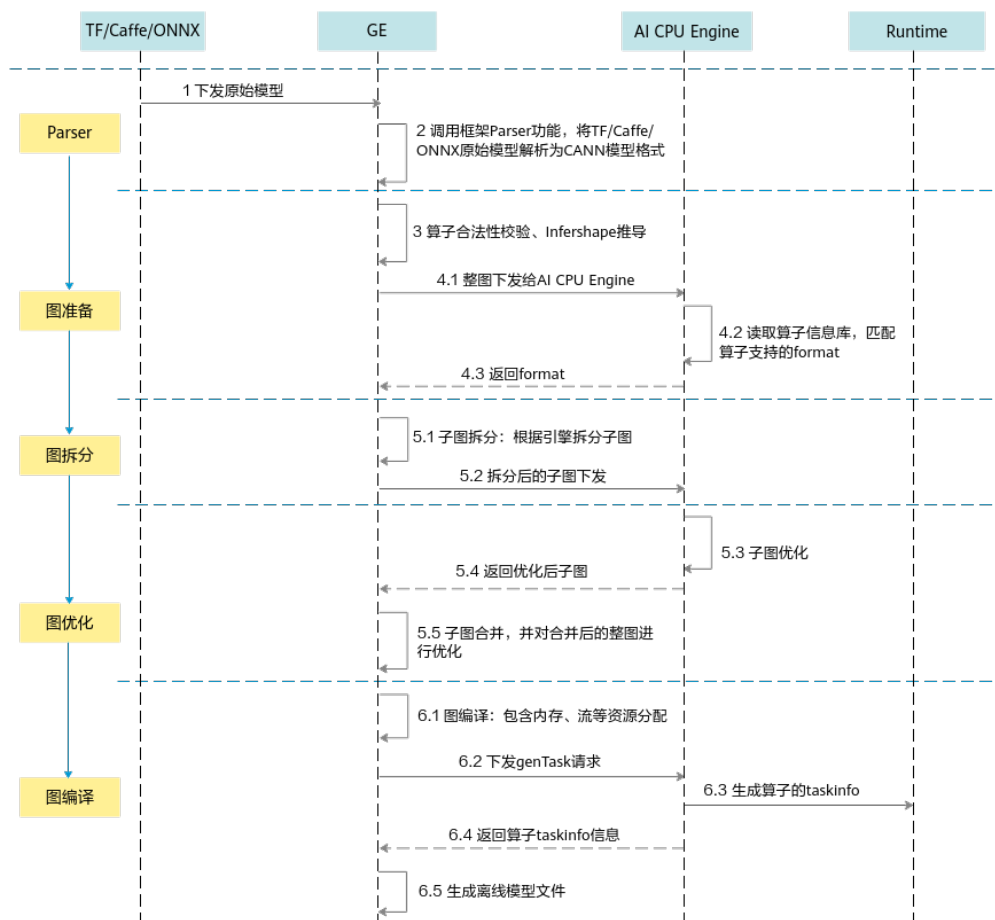
d. 图优化阶段：GE将拆分后的子图下发给FE，FE首先在子图内部插入转换算子，然后按照当前子图流程进行TBE算子预编译，对TBE算子进行UB融合，并根据算子信息库中算子信息找到算子实现将其编译成算子kernel（算子的\*.o与\*.json），最后将优化后子图返回给GE。

优化后的子图合并为整图，再进行整图优化。

e. 图编译阶段：GE进行图编译，包含内存分配、流资源分配等，并向FE发送tasking请求，FE返回算子的taskinfo信息给GE，图编译完成之后生成适配昇腾AI处理器的离线模型文件（\*.om）。

- AI CPU算子模型转换交互流程

图 4-3 AI CPU 算子模型转换交互流程



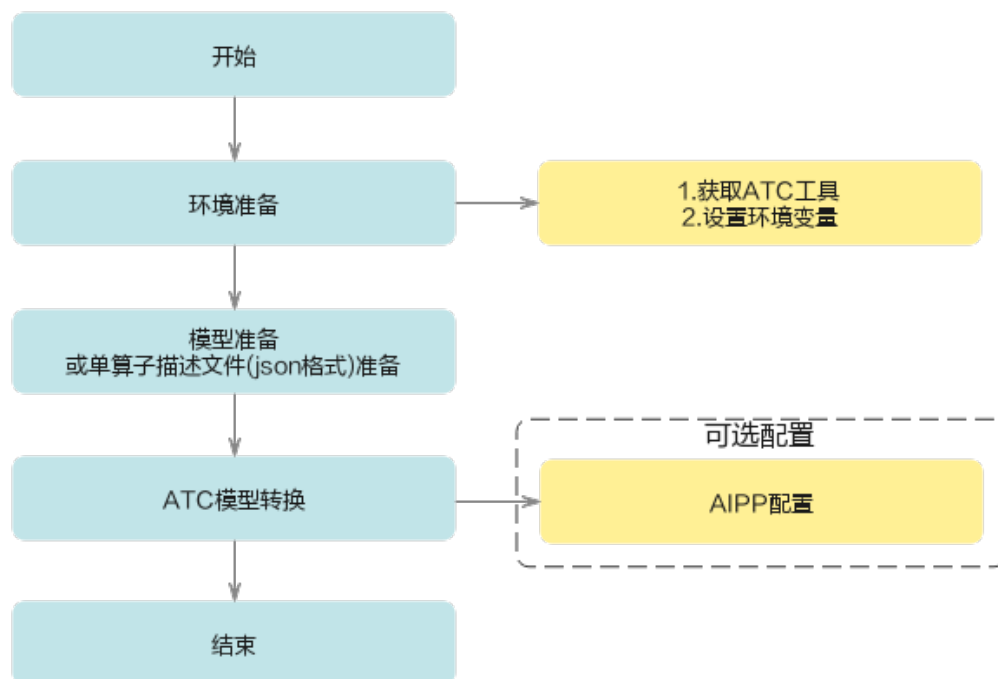
- 调用框架Parser功能，将主流框架的模型格式转换成CANN模型格式。
- 图准备阶段：该阶段会完成算子基本参数校验以及Infershape推导（设置算子输出的shape和dtype）等功能。  
另外，GE将整图下发给AI CPU Engine，AI CPU Engine读取算子信息库，匹配算子支持的format，并将format返回给GE。
- 图拆分阶段：GE根据图中数据将图拆分为多个子图。
- 图优化阶段：GE将拆分后的子图下发给AI CPU Engine，AI CPU Engine进行子图优化，并将优化后子图返回给GE。  
优化后的子图合并为整图，再进行整图优化。
- 图编译阶段：GE进行图编译，包含内存分配、流资源分配等，并向AI CPU Engine发送genTask请求，AI CPU Engine返回算子的taskinfo信息给GE，图编译完成之后生成适配昇腾AI处理器的离线模型文件（\*.om）。

## 4.2 运行流程

使用ATC工具进行模型转换的运行流程如图4-4所示。



图 4-4 运行流程



1. 使用ATC工具之前，请先在开发环境安装CANN软件包，获取相关路径下的ATC工具，详细说明请参见[获取ATC工具](#)。
2. 准备要进行转换的模型或单算子描述文件，并上传到开发环境。
3. 使用ATC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](#)。

## 4.3 关键概念

- AIPP  
AIPP (Artificial Intelligence Pre-Processing) AI预处理，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。
- YUV420SP  
有损图像颜色编码格式，常用为YUV420SP\_UV、YUV420SP\_VU两种格式。
- 知识库  
存储auto tune调优后的Schedule，在后续算子编译中直接使用。
- cost model  
评估器，auto tune过程中如果没有命中知识库，则通过cost model评估tiling空间中tiling的优劣，选择最优tiling数据。
- 数据排布格式 (format)  
Format为数据的物理排布格式，定义了解读数据的维度，比如1D，2D，3D，4D，5D等。
  - NCHW和NHWC

在深度学习框架中，多维数据通过多维数组存储，比如卷积神经网络的特征图（Feature Map）通常用四维数组保存，即4D，4D格式解释如下：

- N: Batch数量，例如图像的数目。
- H: Height，特征图高度，即垂直高度方向的像素个数。
- W: Width，特征图宽度，即水平宽度方向的像素个数。
- C: Channels，特征图通道，例如彩色RGB图像的Channels为3。

由于数据只能线性存储，因此这四个维度有对应的顺序。不同深度学习框架会按照不同的顺序存储特征图数据，比如Caffe，排列顺序为[Batch, Channels, Height, Width]，即NCHW。TensorFlow中，排列顺序为[Batch, Height, Width, Channels]，即NHWC。

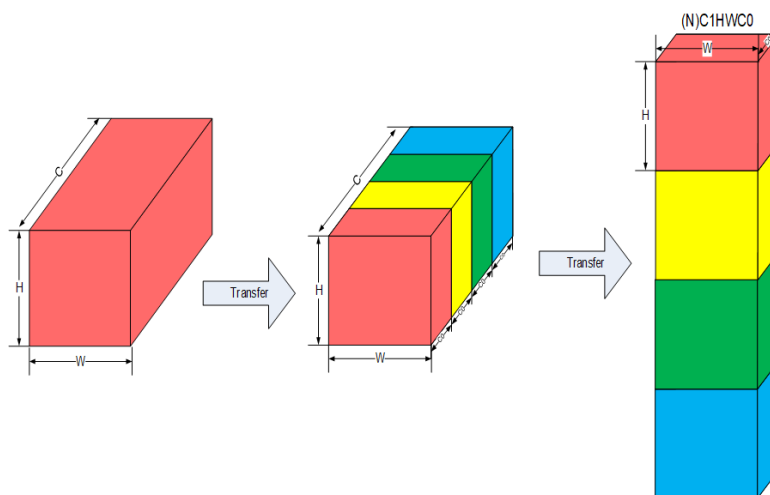
如图4-5所示，以一张格式为RGB的图片为例，NCHW实际存储的是“RRRRRRGGGGGGBBBBBB”，同一通道的所有像素值顺序存储在一起，而NHWC实际存储的则是“RGBRGBRGBRGBRGB”，多个通道的同一位置的像素值顺序存储在一起。

图 4-5 NCHW 和 NHWC



#### - NC1HWC0

昇腾AI处理器中，为了提高通用矩阵乘法（GEMM）运算数据块的访问效率，所有张量数据统一采用NC1HWC0的五维数据格式，如下图所示。



其中C0与微架构强相关，等于AI Core中矩阵计算单元的大小，对于FP16类型为16，对于INT8类型则为32，这部分数据需要连续存储； $C1=C/C0$ 。如果结果不整除，向上取整。

例如：NHWC -> NC1HWC0的转换过程为：

- 将NHWC数据在C维度进行分割，变成C1份NHWC0。
- 将C1份NHWC0在内存中连续排列，由此变成NC1HWC0。

NHWC->NC1HWC0的转换场景示例：

- i. 首层RGB图像通过AIPP转换为NC1HWC0格式。
- ii. 中间层Feature Map每层输出为NC1HWC0格式，在搬运过程中需要重排。

# 5 初级功能

原始模型文件或离线模型转成json文件  
离线模型支持动态BatchSize/动态分辨率  
离线模型支持动态维度  
自定义离线模型的输入输出数据类型  
借助离线模型查看软件基础版本号

## 5.1 原始模型文件或离线模型转成 json 文件

### 场景介绍

如果用户不方便查看原始模型或离线模型的参数信息时，可以将原始模型或离线模型转成json文件进行查看。

### 转换方法

本章节以Caffe框架ResNet-50网络模型为例，进行演示，参见[步骤1](#)获取原始模型文件。

- 原始模型文件转json文件

该场景下[7.2.2.3 --om](#)参数需要指定为原始模型文件，命令示例如下：

```
atc --mode=1 --om=$HOME/module/resnet50.prototxt --json=$HOME/module/out/  
caffe_resnet50.json --framework=0
```

- 离线模型转json文件

该场景下需要先将原始模型转成离线模型，然后再执行离线模型转成json的操作。

- 原始模型转成离线模型，命令示例如下：

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel  
--framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version>
```

- 离线模型转成json文件，命令示例如下：

```
atc --mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/  
caffe_resnet50.json
```

关于参数的详细解释以及使用方法请参见[9 参考](#)。若提示如下信息，则说明转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

ATC run success

成功执行命令后，在--json参数指定的路径下，可查看转换后的json文件信息。  
关于参数的详细解释以及使用方法请参见[9 参考](#)。

## 5.2 离线模型支持动态 BatchSize/动态分辨率

### 场景介绍

某些推理场景，如检测出目标后再执行目标识别网络，由于目标个数不固定导致目标识别网络输入BatchSize不固定。如果每次推理都按照最大的Batch或最大分辨率进行计算，会造成计算资源浪费。因此，模型转换需要支持动态BatchSize和动态分辨率的设置，使用ATC工具时，通过[7.2.2.8 --dynamic\\_batch\\_size](#)参数设置支持的Batch档位，通过[7.2.2.9 --dynamic\\_image\\_size](#)参数设置支持的分辨率档位。

### 转换方法

如下转换示例以Caffe框架ResNet50网络模型为例进行演示，参见[步骤1](#)获取原始模型文件。

- 步骤1** 以CANN软件包运行用户登录开发环境，并将模型转换过程中用到的模型文件（\*.prototxt）、权重文件（\*.caffemodel）等上传到开发环境任意路径，例如上传到\$HOME/module/目录下。
- 步骤2** atc命令中加入[7.2.2.8 --dynamic\\_batch\\_size](#)参数或者[7.2.2.9 --dynamic\\_image\\_size](#)，执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

- 动态BatchSize  
atc --model=\$HOME/module/resnet50.prototxt --weight=\$HOME/module/resnet50.caffemodel --framework=0 --output=\$HOME/module/out/caffe\_resnet50 --soc\_version=<soc\_version> --input\_shape="data:-1,3,224,224" --dynamic\_batch\_size="1,2,4,8"
- 动态分辨率  
atc --model=\$HOME/module/resnet50.prototxt --weight=\$HOME/module/resnet50.caffemodel --framework=0 --output=\$HOME/module/out/caffe\_resnet50 --soc\_version=<soc\_version> --input\_shape="data:1,3,-1,-1" --dynamic\_image\_size="224,224;448,448"

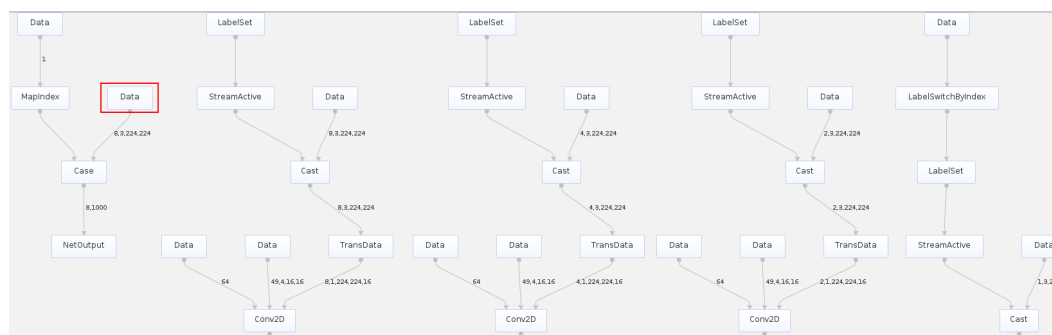
关于参数的详细解释以及使用方法请参见[9 参考](#)。若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

ATC run success

成功执行命令后，在--output参数指定的路径下，可查看离线模型（如：caffe\_resnet50.om）。

模型转换完成后，在生成的om模型中，会新增一个输入（如[图5-1](#)中红框中的Data输入），在模型推理时通过该新增的输入提供具体的Batch值（或分辨率值）。例如，a输入的BatchSize是动态的（或分辨率是动态的），在om模型中，会有与a对应的b输入来描述a的BatchSize（或分辨率取值）。

图 5-1 包含动态 BatchSize 功能的离线模型



----结束

## 5.3 离线模型支持动态维度

### 场景介绍

为支持Transformer等网络在输入格式的维度不确定的场景，通过[7.2.2.10 --dynamic\\_dims](#)参数实现ND格式下任意维度的档位设置。

ND表示支持任意格式，当前 $N \leq 4$ 。

### 转换方法

本章节以Caffe框架ResNet-50网络模型为例进行演示，参见[步骤1](#)获取原始模型文件。

**步骤1** 以CANN软件包运行用户登录开发环境，并将支持设置动态维度的模型文件上传到开发环境任意路径，例如上传到\$HOME/module/目录下。

**步骤2** atc命令中加入[7.2.2.10 --dynamic\\_dims](#)等相关参数，执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --  
framework=0 --output=$HOME/module/out/dynamic_dims --soc_version=<soc_version> --  
input_shape="data:-1,3,-1,-1" --dynamic_dims="1,224,224;8,448,448" --input_format=ND
```

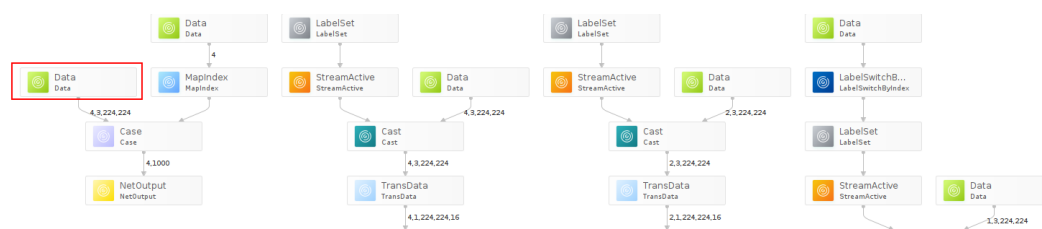
关于参数的详细解释以及使用方法请参见[9 参考](#)。若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

成功执行命令后，在--output参数指定的路径下，可查看离线模型。

模型转换完成后，在生成的om模型中，会新增一个输入（如[图5-1](#)中红框中的Data输入），在模型推理时通过该新增的输入提供具体的维度值。例如，a输入的维度为动态的，在om模型中，会有与a对应的b输入来描述a的维度值。

图 5-2 包含动态维度的离线模型



----结束

## 5.4 自定义离线模型的输入输出数据类型

### 场景介绍

模型转换时支持指定网络的输入节点、输出节点的Datatype、Format、模型转换支持精度选择等关键参数。

假如，针对Caffe框架ResNet50网络模型，要求转换后离线模型的输入数据为FP16类型，指定`Pooling`算子作为输出算子（对应的节点名称为`pool1`），并且指定该输出节点的数据类型为FP16。该场景下就需要分别使用[7.3.1.2 --input\\_fp16\\_nodes](#)、[7.3.1.1 --out\\_nodes](#)、[7.2.3.2 --output\\_type](#)等参数来实现上述功能。

### 转换方法

本章节以Caffe框架ResNet-50网络模型为例进行演示，参见[步骤1](#)获取原始模型文件。

**步骤1** 以CANN软件包运行用户登录开发环境，并将模型转换过程中用到的模型文件（\*.prototxt）、权重文件（\*.caffemodel）等上传到开发环境任意路径，例如上传到\$HOME/module/目录下。

**步骤2** atc命令中加入[7.3.1.2 --input\\_fp16\\_nodes](#)、[7.3.1.1 --out\\_nodes](#)、[7.2.3.2 --output\\_type](#)参数，执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version> --input_fp16_nodes="data" --out_nodes="pool1:0" --output_type="pool1:0:FP16"
```

关于参数的详细解释以及使用方法请参见[9 参考](#)。若提示如下信息，则说明模型转换成功，若模型转换失败，则请参见[9.2 错误码参考](#)进行定位。

```
ATC run success
```

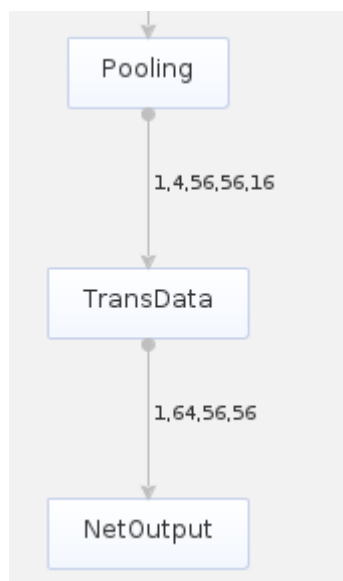
成功执行命令后，在output参数指定的路径下，可查看离线模型（如：caffe\_resnet50.om）。

**步骤3** （可选）如果用户想查看转换后离线模型中上述指定节点以及指定数据类型相关信息，则可以将上述离线模型转成json文件查看，命令如下：

```
atc --mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

[图5-3](#)为`Pooling`算子作为模型输出算子的示意图。

图 5-3 指定某个算子为离线模型输出



----结束

## 5.5 借助离线模型查看软件基础版本号

### 场景介绍

不同软件基础版本号场景下，由于软件功能差异，所转换出的离线模型功能也有差异，该场景下建议用户使用匹配软件版本的ATC工具重新进行模型转换。假如用户已有转换好的离线模型，想查看使用的软件基础版本号，则可以参见该章节完成。

### 查看方法

**步骤1** 获取已经转换好的离线模型，例如`caffe_resnet50.om`，并以CANN软件包运行用户将其上传至开发环境任意目录，例如上传到`$HOME/module/`目录下。

**步骤2** 将离线模型转成json文件：

```
atc --mode=1 --om=$HOME/module/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

在转换后的json文件中，可以查看原始模型转换为该离线模型时，使用的基础版本号，示例如下：（如下示例中的版本号都为样例，请以实际查询的为准）

说明：离线模型转换为json文件可以查看基础版本号，必须保证使用的软件为1.77.22.6.220及之后的版本，之前版本无法查阅该信息；`atc_cmdline`参数必须为1.78.23.34.230及之后的软件版本方可查阅。

```
{
  "key": "opp_version",
  "value": {
    "s": "1.78.23.34.230"
  }
},
...
{
  "key": "atc_version",
  "value": {
    "s": "1.78.23.34.230"
  }
}
```



```
},  
...  
  
{  
  "key": "atc_cmdline",  
  "value": {  
    "s": "xxx/atc.bin --model ./resnet50.prototxt --weight ./resnet50.caffemodel --framework 0 --  
output ./out/resnet50 --soc_version Ascend310"  
  }  
},  
...  
  
{  
  "key": "soc_version",  
  "value": {  
    "s": "Ascend310Ascend610SD3403"  
  }  
},  
...  
}
```

----结束

# 6 高级功能

AIPP使能

单算子模型转换

## 6.1 AIPP 使能

### 6.1.1 什么是 AIPP

AIPP ( Artificial Intelligence Pre-Processing ) 人工智能预处理，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），数据处理之后再行真正的模型推理。

该模块功能与DVPP相似，都是用于图像数据预处理，但与DVPP相比，由于DVPP各组件基于处理速度和处理占有量的考虑，对输入、输出有特殊的限制，如对输出图片的宽高有对齐要求，且其输出格式通常为YUV420SP等格式。这样的设定虽在视频分析的场景下有非常广阔的输入，但深度学习模型的输入通常为RGB或BRG，且输入图片尺寸各异，因此ATC工具流程中提供了AIPP功能模块。

与DVPP不同的是，AIPP主要用于在AI Core上完成数据预处理，通过AIPP提供的色域转换功能，输出满足要求的图片格式；通过改变图像尺寸中的补边（Padding）功能，输出满足长宽对齐的图片等，AIPP的出现是对DVPP能力的有效补充。

AIPP根据配置方式不同，分为静态AIPP和动态AIPP；如果要将原始图片输出为满足推理要求的图片格式，则需要使用色域转换功能；如果要输出固定大小的图片，则需要使用AIPP提供的Crop（抠图）、Padding（补边）功能。

#### 静态 AIPP 和动态 AIPP

在使能AIPP功能时，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，使能AIPP时可以通过[aipp\\_mode](#)参数控制。具体配置示例请参见[6.1.3 AIPP配置示例](#)，关于参数解释请参见[6.1.8 配置文件模板](#)。

- 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参数值被保存在离线模型中，每次模型推理过程采用固定的AIPP预处理参数进行处理，而且在之后的推理过程中无法通过业务代码进行直接的修改。  
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：模型转换时仅设置AIPP模式为动态，每次在执行推理前，根据需求动态修改AIPP参数值，然后在模型执行时可使用不同的AIPP参数。动态AIPP参数值会根据需求在不同的业务场景下选用合适的参数（如不同摄像头采用不同的归一化参数，输入图片格式需要兼容YUV420和RGB等）。

如果模型转换时设置了动态AIPP，则使用应用工程进行模型推理时，需要在接口之前，调用接口，设置模型推理的动态AIPP数据。设置动态AIPP参数值的接口请参见《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>[aclmdlSetInputAIPP](#)”章节。

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等参数。关于动态参数结构体，请参见[动态AIPP的参数输入结构](#)。

AIPP支持的图像输入格式包括：YUV420SP\_U8、RGB888\_U8、XRGB8888\_U8、YUV400\_U8。

## 色域转换

色域转换，用于将输入的图片格式，转换为模型需要的图片格式，在使能AIPP功能时，通过[csc\\_switch](#)参数控制色域转换功能是否开启，参数解释请参见[6.1.8 配置文件模板](#)。

一旦确认了AIPP处理前与AIPP处理后的图片格式，即可确定色域转换其他相关的参数值，本手册提供相关模板可以供用户使用，无需再次修改，配置示例请参见[6.1.4 色域转换配置说明](#)。

## 改变图像尺寸

AIPP功能中的改变图像尺寸操作由Crop（抠图）、Padding（补边）完成，分别对应配置模板中的crop、padding参数。参数解释请参见[6.1.8 配置文件模板](#)。

关于该功能的详细说明以及AIPP参数配置示例请参见[6.1.6 Crop/Padding配置说明](#)。

## 6.1.2 如何使能 AIPP

通过在模型转换过程中开启AIPP功能，可以在推理之前就完成所有的数据处理，专门的加速模块实现并保证性能，可以不让图像处理成为推理阶段的瓶颈，图像处理方式比较灵活。本章节给出如何在模型转换阶段开启AIPP功能。

## 使用约束

1. 动态AIPP的参数每次推理需要计算，计算需要耗时，所以动态AIPP的性能比静态AIPP性能要差。
2. 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，在进行推理业务时，输入图片数据要求为NHWC排布，该场景下最终与AIPP连接的输入节点的格式被强制改成NHWC，可能与模型转换命令中[7.2.2.5 --input\\_format](#)参数指定的格式不一致。
  - 模型转换时没有开启AIPP，模型转换如果没有使用[7.2.2.5 --input\\_format](#)指定模型的输入格式，则Caffe模型默认为NCHW排布，TensorFlow 模型输入默认为NHWC。在进行推理业务时，需要用户自行根据模型的输入格式来选取输入图片的排布类型。例如，输入图片数据为NHWC排布，模型为Caffe模型，模型转换时不指定[7.2.2.5 --input\\_format](#)，则默认模型输入格式

为NCHW，输入图片和模型输入的格式不一致。该场景下，需要用户自行把NHWC排布的原始图片数据转换为NCHW排布。

## 使用示例

模型转换时，AIPP功能会以aipp算子形式插入离线模型中，如下示例以Caffe框架ResNet50网络模型为例，演示模型转换时如何使用AIPP。

### 步骤1 获取Caffe网络模型。

您可以从以下链接中获取ResNet-50网络的模型文件 (\*.prototxt)、预训练模型文件 (\*.caffemodel)，并以CANN软件包运行用户将获取的文件上传至开发环境任意目录，例如上传到\$HOME/module/目录下。

- ResNet-50网络的模型文件 (\*.prototxt)：单击[Link](#)下载该文件。
- ResNet-50网络的预训练模型文件 (\*.caffemodel)：单击[Link](#)下载该文件。

### 步骤2 构造AIPP配置文件insert\_op.cfg。

您可以根据[6.1.3 AIPP配置示例](#)或[6.1.9 典型场景样例参考](#)章节获取AIPP配置示例，如果上述示例无法满足要求，则需要参见[6.1.8 配置文件模板](#)自行构造配置文件。

### 步骤3 atc命令中加入7.3.1.3 --insert\_op\_conf参数，用于插入aipp预处理算子，执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --framework=0 --insert_op_conf=$HOME/module/insert_op.cfg --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version>
```

关于参数的详细解释以及使用方法请参见[7 参数说明](#)。若提示如下信息，则说明模型转换成功。

```
ATC run success
```

成功执行命令后，在--output参数指定的路径下，可查看离线模型（如：caffe\_resnet50.om）。

### 步骤4 （可选）如果用户想查看转换后离线模型中aipp算子的相关信息，则可以将上述离线模型转成json文件查看，命令如下：

```
atc --mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

如下为json文件中带有aipp信息的样例（如下样例中所有aipp属性值都为样例，请以用户实际构造的配置文件为准）：

```
{
  "key": "aipp",
  "value": {
    "func": {
      "attr": [
        {
          "key": "mean_chn_0",
          "value": {
            "i": 0
          }
        },
        {
          "key": "mean_chn_1",
          "value": {
            "i": 0
          }
        },
        {
          "key": "mean_chn_2",
          "value": {
            "i": 0
          }
        }
      ]
    }
  }
}
```

```
        "i": 0
      }
    },
    {
      "key": "mean_chn_3",
      "value": {
        "i": 0
      }
    },
    {
      "key": "csc_switch",
      "value": {
        "b": true
      }
    },
    {
      "key": "input_format",
      "value": {
        "i": 1
      }
    },
    {
      "key": "input_bias_0",
      "value": {
        "i": 0
      }
    },
    {
      "key": "input_bias_1",
      "value": {
        "i": 128
      }
    },
    {
      "key": "input_bias_2",
      "value": {
        "i": 128
      }
    },
    {
      "key": "aipp_mode",
      "value": {
        "i": 1
      }
    },
    {
      "key": "src_image_size_h",
      "value": {
        "i": 0
      }
    },
    {
      "key": "crop_size_h",
      "value": {
        "i": 0
      }
    },
    {
      "key": "matrix_r0c0",
      "value": {
        "i": 256
      }
    },
    {
      "key": "matrix_r0c1",
      "value": {
        "i": 443
      }
    },
  ],
```

```
{
  "key": "matrix_r0c2",
  "value": {
    "i": 0
  }
},
{
  "key": "src_image_size_w",
  "value": {
    "i": 0
  }
},
{
  "key": "crop_size_w",
  "value": {
    "i": 0
  }
},
{
  "key": "rbuv_swap_switch",
  "value": {
    "b": false
  }
},
{
  "key": "padding",
  "value": {
    "b": false
  }
},
{
  "key": "ax_swap_switch",
  "value": {
    "b": false
  }
},
{
  "key": "top_padding_size",
  "value": {
    "i": 0
  }
},
{
  "key": "matrix_r1c0",
  "value": {
    "i": 256
  }
},
{
  "key": "matrix_r1c1",
  "value": {
    "i": -86
  }
},
{
  "key": "matrix_r1c2",
  "value": {
    "i": -178
  }
},
{
  "key": "resize",
  "value": {
    "b": false
  }
},
{
  "key": "resize_output_h",
  "value": {
```

```
        "i": 0
      }
    },
    {
      "key": "related_input_rank",
      "value": {
        "i": 0
      }
    },
    {
      "key": "load_start_pos_h",
      "value": {
        "i": 0
      }
    },
    {
      "key": "matrix_r2c0",
      "value": {
        "i": 256
      }
    },
    {
      "key": "matrix_r2c1",
      "value": {
        "i": 0
      }
    },
    {
      "key": "matrix_r2c2",
      "value": {
        "i": 350
      }
    },
    {
      "key": "resize_output_w",
      "value": {
        "i": 0
      }
    },
    {
      "key": "var_rec_i_chn_0",
      "value": {
        "f": "1"
      }
    },
    {
      "key": "var_rec_i_chn_1",
      "value": {
        "f": "1"
      }
    },
    {
      "key": "var_rec_i_chn_2",
      "value": {
        "f": "1"
      }
    },
    {
      "key": "load_start_pos_w",
      "value": {
        "i": 0
      }
    },
    {
      "key": "var_rec_i_chn_3",
      "value": {
        "f": "1"
      }
    },
  ],
```

```
{
  "key": "single_line_mode",
  "value": {
    "b": false
  }
},
{
  "key": "output_bias_0",
  "value": {
    "i": 16
  }
},
{
  "key": "output_bias_1",
  "value": {
    "i": 128
  }
},
{
  "key": "output_bias_2",
  "value": {
    "i": 128
  }
},
{
  "key": "right_padding_size",
  "value": {
    "i": 0
  }
},
{
  "key": "bottom_padding_size",
  "value": {
    "i": 0
  }
},
{
  "key": "min_chn_0",
  "value": {
    "f": "0"
  }
},
{
  "key": "min_chn_1",
  "value": {
    "f": "0"
  }
},
{
  "key": "min_chn_2",
  "value": {
    "f": "0"
  }
},
{
  "key": "min_chn_3",
  "value": {
    "f": "0"
  }
},
{
  "key": "crop",
  "value": {
    "b": false
  }
},
{
  "key": "cpadding_value",
  "value": {
```



```
        "f": "0"  
      },  
      {  
        "key": "left_padding_size",  
        "value": {  
          "i": 0  
        }  
      }  
    ]  
  }  
}
```

----结束

## 6.1.3 AIPP 配置示例

### 6.1.3.1 静态 AIPP 配置示例

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；如果模型只有一个输入，则只需要配置第一组aipp\_op即可，如下示例以网络模型为多输入时进行说明：

#### 须知

- 如果模型转换时，用户设置了[7.2.2.9 --dynamic\\_image\\_size](#)动态分辨率参数，即输入图片的宽和高不确定，同时又通过[7.3.1.3 --insert\\_op\\_conf](#)参数设置了静态AIPP功能：该场景下，AIPP配置文件中不能开启Crop和Padding功能，并且需要将配置文件中的src\_image\_size\_w和src\_image\_size\_h取值设置为0。
- 如果模型转换时，用户设置了[7.2.2.7 --input\\_shape\\_range](#)动态shape范围参数，同时又通过[7.3.1.3 --insert\\_op\\_conf](#)参数配置了AIPP功能，则AIPP输出的宽和高要在[7.2.2.7 --input\\_shape\\_range](#)所设置的范围内。

- 使用related\_input\_rank参数标识，对模型第几个输入进行AIPP处理，如下配置定义了两组AIPP参数，分别对模型第一个和第二个输入进行AIPP处理：

```
aipp_op {  
  aipp_mode : static  
  related_input_rank : 0 # 标识对第1个输入进行AIPP处理  
  src_image_size_w : 608  
  src_image_size_h : 608  
  crop : false  
  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  
  mean_chn_0 : 104  
  mean_chn_1 : 117
```

```
    mean_chn_2 : 123
  }
  aipp_op {
    aipp_mode : static
    related_input_rank : 1 # 标识对第2个输入进行AIPP处理
    src_image_size_w : 608
    src_image_size_h : 608
    crop : false
    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
    mean_chn_0 : 104
    mean_chn_1 : 117
    mean_chn_2 : 123
  }
}
```

- 使用related\_input\_name参数标识，对模型第几个输入进行AIPP处理，此处需要填写为模型输入的name（input对应的值）或者模型首层节点的输出（top参数对应的取值），该参数只适用于Caffe网络模型，且不能与related\_input\_rank参数同时使用。

如下配置定义了两组AIPP参数，分别对模型第一个和第二个输入进行AIPP处理：

```
aipp_op {
  aipp_mode : static
  related_input_name : "data" # 标识对第1个输入进行AIPP处理
  src_image_size_w : 608
  src_image_size_h : 608
  crop : false
  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
  mean_chn_0 : 104
  mean_chn_1 : 117
  mean_chn_2 : 123
}
aipp_op {
  aipp_mode : static
  related_input_name : "im_info" # 标识对第2个输入进行AIPP处理
  src_image_size_w : 608
  src_image_size_h : 608
  crop : false
  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
mean_chn_0 : 104
mean_chn_1 : 117
mean_chn_2 : 123
}
```

### 6.1.3.2 动态 AIPP 配置示例

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；如果模型只有一个输入，则只需要配置第一组aipp\_op即可，如下示例以网络模型为多输入时进行说明。

#### 配置示例

##### 须知

- 如果模型转换时，用户设置了7.2.2.8 `--dynamic_batch_size`动态Batch档位参数，同时又通过7.3.1.3 `--insert_op_conf`参数配置了动态AIPP功能：  
实际推理时，调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>`aclmdlSetInputAIPP`”接口，设置动态AIPP相关参数值时，需确保batchSize要设置为最大Batch数。
- 如果模型转换时，用户设置了7.2.2.9 `--dynamic_image_size`动态分辨率参数，同时又通过7.3.1.3 `--insert_op_conf`参数配置了动态AIPP功能：  
实际推理时，调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>`aclmdlSetInputAIPP`”接口，设置动态AIPP相关参数值时，不能开启Crop和Padding功能。该场景下，还需要确保通过应用软件开发指南 (C&C++, 开放态)》手册中的“AscendCL API参考>模型加载与执行>`aclmdlSetDynamicHWSIZE`”接口设置的宽、高相等，都必须设置成动态分辨率最大档位的宽、高。
- 如果模型转换时，用户设置了7.2.2.7 `--input_shape_range`动态shape范围参数，同时又通过7.3.1.3 `--insert_op_conf`参数配置了AIPP功能，则AIPP输出的宽和高要在7.2.2.7 `--input_shape_range`所设置的范围内。

动态AIPP场景下，用户无需手动配置csc\_switch、rbuv\_swap\_switch等参数，根据如下配置文件配置好相关参数后，模型转换时，ATC会为动态AIPP新增一个模型输入（以下简称AippData）。实际推理时，需要调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>`aclmdlSetInputAIPP`”接口，设置动态AIPP相关参数值，然后传给上述新增的AippData，AippData根据传入的参数值构造的结构体为[动态AIPP的参数输入结构](#)，该结构体无需用户手动处理。

```
aipp_op
{
  aipp_mode: dynamic
  related_input_rank: 0    # 标识对第1个输入进行AIPP处理
}
```

```
max_src_image_size: 752640 # 输入图像最大的size, 参数必填
}
aipp_op
{
    aipp_mode: dynamic
    related_input_rank: 1      # 标识对第2个输入进行AIPP处理
    max_src_image_size: 752640 # 输入图像最大的size, 参数必填
}
```

## 动态 AIPP 的参数输入结构

根据[配置示例](#)配置好动态AIPP文件后，模型推理时为动态AIPP新增模型输入（AippData）传入参数值后，自动形成的结构体如下，该结构体无需用户手动处理：

```
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/column copy
                                // 3: padding source picture data, block copy
                                // 4: padding source picture data, mirror copy
    int8_t rotateSwitch;        //rotate switch, 0: non-rotate, 1: rotate 90°clockwise, 2: rotate 180°clockwise, 3:
    rotate 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
    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
}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 batchSize;           //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;
```

## 6.1.4 色域转换配置说明

AIPP提供了更为方便的图像格式转换方式：色域转换，用于将输入的图片格式，转换为模型需要的图片格式，一旦确认了AIPP处理前与AIPP处理后的图片格式，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**）。

例如：将JPEG格式的图像文件（如后缀为jpg、jpeg、JPG、JPEG的图像文件）转为RGB格式；将视频解码后的YUV格式数据转为RGB格式。而根据不同的彩色视频数字化标准又可以将视频格式分为BT-601标准清晰度视频格式（定义于SDTV标准中）和BT-709高清标准视频格式（定义于HDTV标准中）。两种视频格式又分为NARROW和WIDE，其中：

	Ranges:		Ranges:
	Y[16 ... 235]		Y/Cb/Cr[0 ... 255]
	Cb/Cr[16 ... 240]		
NARROW取值范围为:	R/G/B[0 ... 255]	WIDE取值范围为:	R/G/B[0 ... 255]

关于如何判断输入数据的标准，请参见[10.2 使用AIPP色域转换模型时如何判断视频流的格式标准](#)。

YUV格式的数据转为RGB格式可以视作如下公式展示的矩阵乘法，这其中的转换矩阵就是待配置的参数和偏移量。

```
# 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 |
```

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、RGB888\_U8等），当前给出JPEG、BT-601NARROW、BT-601WIDE、BT-709NARROW、BT-709WIDE几种典型场景下的色域转换配置。

### 说明

DVPP处理后的数据，不会对色域配置产生影响，例如DVPP处理前的数据为BT-709格式，经过DVPP处理后，输入给AIPP，此时数据仍为BT-709格式。

## YUV420SP\_U8 转 YUV444

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

## YUV420SP\_U8 转 YVU444

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

## YUV420SP\_U8 转 RGB

- 输入数据为JPEG图像

### JPEG

```
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  
}
```

- 输入数据为BT-601NARROW视频

### BT-601NARROW

```
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  
}
```

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

```
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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : YUV420SP_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  matrix_r0c0 : 256  
  matrix_r0c1 : 0  
  matrix_r0c2 : 403  
  matrix_r1c0 : 256  
  matrix_r1c1 : -48  
  matrix_r1c2 : -120  
  matrix_r2c0 : 256  
  matrix_r2c1 : 475  
  matrix_r2c2 : 0  
  input_bias_0 : 0  
  input_bias_1 : 128  
  input_bias_2 : 128  
}
```

## YUV420SP\_U8 转 BGR

- 输入数据为JPEG图像

#### JPEG

```
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  
}
```

- 输入数据为BT-601NARROW视频



#### BT-601NARROW

```
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  
}
```

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

```
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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : YUV420SP_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  matrix_r0c0 : 256  
  matrix_r0c1 : 475  
  matrix_r0c2 : 0  
  matrix_r1c0 : 256  
  matrix_r1c1 : -48  
  matrix_r1c2 : -120  
  matrix_r2c0 : 256  
  matrix_r2c1 : 0  
  matrix_r2c2 : 403  
  input_bias_0 : 0  
  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  
}
```

## YVU420SP\_U8 转 RGB

- 输入数据为JPEG图像

### JPEG

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

- 输入数据为BT-601NARROW视频

### BT-601NARROW

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

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

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

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

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

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : YUV420SP_U8  
  csc_switch : true  
  rbuv_swap_switch : true  
  matrix_r0c0 : 256  
  matrix_r0c1 : 0  
  matrix_r0c2 : 403  
  matrix_r1c0 : 256  
  matrix_r1c1 : -48  
  matrix_r1c2 : -120  
  matrix_r2c0 : 256  
  matrix_r2c1 : 475  
  matrix_r2c2 : 0  
  input_bias_0 : 0  
  input_bias_1 : 128  
  input_bias_2 : 128  
}
```

### YVU420SP\_U8 转 BGR

- 输入数据为JPEG图像

#### JPEG

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

- 输入数据为BT-601NARROW视频

#### BT-601NARROW

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

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

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

- 输入数据为BT-709NARROW视频

**BT-709NARROW**

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

- 输入数据为BT-709WIDE视频

**BT-709WIDE**

```
aipp_op {  
  aipp_mode: static  
  input_format : YUV420SP_U8  
  csc_switch : true  
  rbuv_swap_switch : true  
  matrix_r0c0 : 256  
  matrix_r0c1 : 475  
  matrix_r0c2 : 0  
  matrix_r1c0 : 256  
  matrix_r1c1 : -48  
  matrix_r1c2 : -120  
  matrix_r2c0 : 256  
  matrix_r2c1 : 0  
  matrix_r2c2 : 403  
  input_bias_0 : 0  
  input_bias_1 : 128  
  input_bias_2 : 128  
}
```

**RGB888\_U8 转 RGB**

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

**RGB888\_U8 转 BGR**

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

## RGB888\_U8 转 YUV444

- 输入数据为JPEG图像

### JPEG

```
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  
}
```

- 输入数据为BT-601NARROW视频

### BT-601NARROW

```
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  
}
```

- 输入数据为BT-601WIDE视频



#### BT-601WIDE

```
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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : RGB888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  matrix_r0c0 : 55  
  matrix_r0c1 : 183  
  matrix_r0c2 : 19  
  matrix_r1c0 : -29  
  matrix_r1c1 : -99  
  matrix_r1c2 : 128  
  matrix_r2c0 : 128  
  matrix_r2c1 : -116  
  matrix_r2c2 : -12  
  output_bias_0 : 0  
  output_bias_1 : 128  
  output_bias_2 : 128  
}
```

## RGB888\_U8 转 YVU444

- 输入数据为JPEG图像

#### JPEG

```
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  
}
```

- 输入数据为BT-601NARROW视频

#### BT-601NARROW

```
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  
}
```

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

```
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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : RGB888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  matrix_r0c0 : 55  
  matrix_r0c1 : 183  
  matrix_r0c2 : 19  
  matrix_r1c0 : 128  
  matrix_r1c1 : -116  
  matrix_r1c2 : -12  
  matrix_r2c0 : -29  
  matrix_r2c1 : -99  
  matrix_r2c2 : 128  
  output_bias_0 : 0  
  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  
}
```

## BGR888\_U8 转 GRAY

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

## BGR888\_U8 转 RGB

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

## BGR888\_U8 转 BGR

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

## XRGB8888\_U8 转 RGB

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : false  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
}
```

## XRGB8888\_U8 转 BGR

```
aipp_op {  
  aipp_mode : static  
  input_format : XRGB8888_U8  
  csc_switch : false  
  rbuv_swap_switch : true  
  ax_swap_switch : true  
}
```

## XRGB8888\_U8 转 YUV444

- 输入数据为JPEG图像

### JPEG

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-601NARROW视频

### BT-601NARROW

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  matrix_r0c0 : 55  
  matrix_r0c1 : 183  
  matrix_r0c2 : 19  
  matrix_r1c0 : -29  
  matrix_r1c1 : -99  
  matrix_r1c2 : 128  
  matrix_r2c0 : 128  
  matrix_r2c1 : -116  
  matrix_r2c2 : -12  
  output_bias_0 : 0  
  output_bias_1 : 128  
  output_bias_2 : 128  
}
```

## XRGB8888\_U8 转 YVU444

- 输入数据为JPEG图像

#### JPEG

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-601NARROW视频



#### BT-601NARROW

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-601WIDE视频

#### BT-601WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-709NARROW视频

#### BT-709NARROW

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  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  
}
```

- 输入数据为BT-709WIDE视频

#### BT-709WIDE

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_swap_switch : true  
  matrix_r0c0 : 55  
  matrix_r0c1 : 183  
  matrix_r0c2 : 19  
  matrix_r1c0 : 128  
  matrix_r1c1 : -116  
  matrix_r1c2 : -12  
  matrix_r2c0 : -29  
  matrix_r2c1 : -99  
  matrix_r2c2 : 128  
  output_bias_0 : 0  
  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  
  ax_swap_switch : true  
  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  
}
```

## XBGR8888\_U8 转 GRAY

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : true  
  ax_swap_switch : true  
  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  
}
```

## RGBX8888\_U8 转 GRAY

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : false  
  ax_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  
}
```

## BGRX8888\_U8 转 GRAY

```
aipp_op {  
  aipp_mode: static  
  input_format : XRGB8888_U8  
  csc_switch : true  
  rbuv_swap_switch : true  
  ax_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  
}
```

## YUV400\_U8 转 GRAY

```
aipp_op {  
  aipp_mode: static  
  input_format : YUV400_U8  
  csc_switch : false  
}
```

### 6.1.5 归一化配置说明

归一化就是要把需要处理的数据经过处理后限制在一定范围内，方便后面数据的处理。AIPP支持的归一化设置，通过减均值和乘系数的操作完成，这样的能力不仅能用于常规的归一化，还能用于不同数据格式的转化。

比如在由unit8转为fp16时，其转换可以视作如下公式。其中，mean\_chn\_i表示每个通道的均值，min\_chn\_i表示每个通道的最小值，var\_reci\_chn表示每个通道方差的倒数，各通路的这三个值都是需要进行配置的参数。

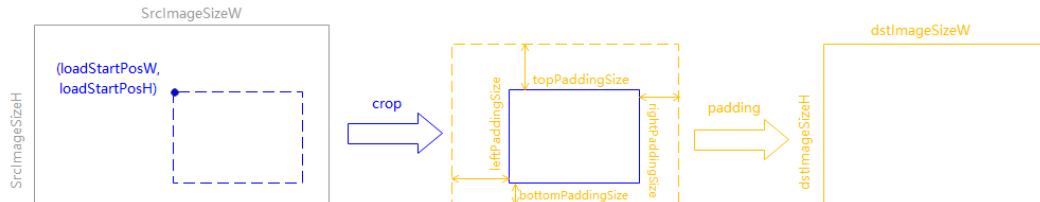
$$\text{pixel\_out\_chx}(i) = [\text{pixel\_in\_chx}(i) - \text{mean\_chn\_i} - \text{min\_chn\_i}] * \text{var\_reci\_chn}$$

### 6.1.6 Crop/Padding 配置说明

AIPP改变图片尺寸需要遵守如下图中的顺序，即先Crop再Padding，每个操作仅能执行一次。

原图大小为srcImageSizeW, srcImageSizeH的图像经过图像预处理后变为模型预期的dstImageSizeW、dstImageSizeH图像尺寸。

图 6-1 改变图像尺寸



#### 说明

图中实线框表示当前图片size，虚线框表示经过右侧箭头上的AIPP操作处理后的图片size。

从执行角度看，我们需要在配置文件中指出裁剪的起始位置左上点坐标loadStartPosW、loadStartPosH以及裁剪后的图像大小crop\_size\_w, crop\_size\_h。在padding环节，我们需要指明在裁剪后的图像四周padding的尺寸，即left\_padding\_size、right\_padding\_size、top\_padding\_size和bottom\_padding\_size。而经过图像尺寸改变之后最终图片大小，需要跟模型文件输入的图像大小即--**input\_shape**中的宽和高相等。

对于YUV420SP\_U8图片类型，load\_start\_pos\_w、load\_start\_pos\_h参数必须配置为偶数。配置样例如下：

```
aipp_op {  
  aipp_mode: static  
  input_format: YUV420SP_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: 20
right_padding_size: 15
top_padding_size: 20
bottom_padding_size: 15
padding_value: 0
}
```

## 6.1.7 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，最终生成离线模型的输入大小（即input\_size）均会被Crop、Padding等操作影响。假设模型的Batch数量为N（如果为动态batch场景，N为最大档位数的取值），模型输入图片的宽为src\_image\_size\_w，高为src\_image\_size\_h，最后模型输入的Size的计算公式如下所示。

### 静态 AIPP 对模型输入大小的校验

表 6-1 input\_size 校验公式

input_format	input_size
YUV400_U8	$N * \text{src\_image\_size\_w} * \text{src\_image\_size\_h} * 1$
YUV420SP_U8	$N * \text{src\_image\_size\_w} * \text{src\_image\_size\_h} * 1.5$
XRGB8888_U8	$N * \text{src\_image\_size\_w} * \text{src\_image\_size\_h} * 4$
RGB888_U8	$N * \text{src\_image\_size\_w} * \text{src\_image\_size\_h} * 3$

### 动态 AIPP 对模型输入大小的校验

如果为动态AIPP，模型转换时，ATC会为动态AIPP新增一个模型输入，用于接收模型推理阶段通过调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API 参考>模型加载与执行>aclmdlSetInputAIPP”接口后传入的AIPP参数，该场景下新增输入节点大小计算公式为：

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

kAippDynamicPara以及kAippDynamicBatchPara参数解释请参见[动态AIPP的参数输入结构](#)。

## 6.1.8 配置文件模板

### 模板使用整体约束

1. 使用配置文件模板时，请将需要配置的参数去注释，并改为合适的值。
2. 模板中参数取值都为默认值，实际使用时，如果配置文件中某些参数未配置，则模型转换时自动设置成该模板中相应参数的默认值。

3. 静态AIPP场景下, `input_format`属性为必选属性, 其余属性均为可选配置, 如果未配置, 则模型转换时自动设置成该模板中相应参数的默认值。
4. 由于硬件处理逻辑的限制, 配置文件中的参数有如下处理顺序要求: 图像裁剪 (`crop`) > 色域转换 (通道交换) > 数据减均值/归一化 > 图像边缘填充 (`padding`)。
5. 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RAW和UINT8格式。
6. 若输入图片为RGB (由R、G、B三个分量组成的图片), 其对应的输入、输出通道顺序, 从高地址到低地址依次为: {R,G,B}。
7. 经过AIPP处理后的图片, 统一采用NC1HWC0的五维数据格式进行存储:  
以原始模型要求的图片为RGB (由R、G、B三个分量组成的图片) 为例进行说明, 配置了AIPP功能场景下:
  - Caffe/ONNX框架数据格式只能设置为NCHW (数据存储格式为RRRRRRGGGGGGBBBBBB)
  - TensorFlow框架数据格式只能设置为NHWC (数据存储格式为RGBRGBRGBRGBRGB) 或NCHW (数据存储格式为RRRRRRGGGGGGBBBBBB)。实际提供的图片经过AIPP色域转换功能处理后, 输出的离线模型中图片为RGB, 并以NC1HWC0五维数据格式进行存储 (关于NC1HWC0详细介绍请参见[4.3 关键概念](#)): 若AIPP输出数据类型为FP16, 则C0=16, 从高位到低位依次为R,G,B, 其余位数补0; C1=1。
8. 对于AIPP支持的图像输入格式YUV420SP\_U8和RGB888\_U8, 根据[6.1.8 配置文件模板](#) `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) 参数的取值不同, AIPP输出不同:
  - 对于YUV420SP\_U8, 根据UV分量顺序不同, YUV420SP\_U8又分为YUV420SP\_UV(NV12)和YUV420SP\_VU(NV21, 对应[6.1.4 色域转换配置说明](#)中的YVU420SP\_U8), 默认为YUV420SP\_UV(NV12):
    - 若`rbuv_swap_switch`设置为false, 则AIPP输出为YUV420SP\_UV(NV12)。
    - 若`rbuv_swap_switch`设置为true, 则AIPP输出为YUV420SP\_VU(NV21)。
  - 对于RGB888\_U8, 根据`rbuv_swap_switch`参数的取值不同, AIPP输出不同:
    - 若`rbuv_swap_switch`设置为false, 则AIPP输出为RGB888\_U8。
    - 若`rbuv_swap_switch`设置为true, 则AIPP输出为BGR888\_U8。
9. 假如用户使用的AIPP图像输入格式为YUV422 sp或RGB package格式, 该场景下与AIPP输入格式对应关系为:
  - YUV420SP\_U8代表的是YUV420 sp格式。
  - RGB888\_U8代表的是RGB package格式。

## 配置文件模板

```
# 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 {
```

```
#===== 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

# related_input_name参数为可选，标识对模型的第几个输入做AIPP处理，此处需要填写为模型输入的名称
# (input对应的值)或者模型首层节点的输出(top参数对应的取值)。该参数只适用于Caffe网络模型，且不能与
# related_input_rank参数同时使用。
# 例如模型有两个输入，且输入name分别为data、im_info，需要对第二个输入做AIPP，则配置
# related_input_name为im_info。
# 类型: string
# 配置范围: 无
related_input_name: ""

===== 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，动态AIPP必须配置（如果为动态batch场景，N为最大档位数的取值）
# 类型: int
max_src_image_size: 0
# 若输入图像格式为YUV400_U8，则max_src_image_size>=N * src_image_size_w * src_image_size_h * 1。
# 若输入图像格式为YUV420SP_U8，则max_src_image_size>=N * src_image_size_w * src_image_size_h * 1.5。
# 若输入图像格式为XRGB8888_U8，则max_src_image_size>=N * src_image_size_w * src_image_size_h * 4。
# 若输入图像格式为RGB888_U8，则max_src_image_size>=N * src_image_size_w * src_image_size_h * 3。

# 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)
=====

# 输入图像格式，必选
# 类型: enum
# 取值范围: YUV420SP_U8、XRGB8888_U8、RGB888_U8、YUV400_U8
input_format:
# 说明: 模型转换完毕后，在对应的*.om模型文件中，上述参数分别以1、2、3、4枚举值呈现。

# 原始图像的宽度、高度
# 类型: int32
# 取值范围 & 约束: 宽度取值范围为[2,4096]或0；高度取值范围为[1,4096]或0，对于YUV420SP_U8类型的图
# 像，要求原始图像的宽和高取值是偶数
src_image_size_w: 0
src_image_size_h: 0
# 说明: 请根据实际图片的宽、高配置src_image_size_w和src_image_size_h；只有crop, padding功能都没有开
# 启的场景，src_image_size_w和src_image_size_h才能取值为0或不配置，该场景下会取网络模型输入定义的w和
# h，并且网络模型输入定义的w取值范围为[2,4096]，h取值范围为[1,4096]。
# C方向的填充值，保留字段，暂不支持该功能
# 类型: float16
# 取值范围: [-65504, 65504]
cpadding_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: int32
# 取值范围 & 约束: [0,4095]、对于YUV420SP_U8类型的图像, 要求取值是偶数
# 说明: load_start_pos_w<src_image_size_w, load_start_pos_h<src_image_size_h
load_start_pos_w: 0
load_start_pos_h: 0

# 抠图后的图像size
# 类型: int32
# 取值范围 & 约束: [0,4096]、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
说明: 若开启抠图功能, 并且没有配置padding, 该场景下crop_size_w和crop_size_h才能取值为0或不配置, 此
时抠图大小 ( crop_size[W|H] ) 的宽和高取值来自模型文件--input_shape中的宽和高, 并且--input_shape中的
宽和高取值范围为[1,4096]。

# 抠图约束如下:
# 若input_format取值为YUV420SP_U8, 则load_start_pos_w、load_start_pos_h必须为偶数。
# 若input_format取值为其他值, 对load_start_pos_w、load_start_pos_h无约束。
# 若开启抠图功能, 则src_image_size[W|H] >= crop_size[W|H]+load_start_pos[W|H]。

#===== 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: int32
# 取值范围 & 约束: resize_output_h: [16,4096]或0; resize_output_w: [16,1920]或0; resize_output_w/
resize_input_w ∈ [1/16,16]、resize_output_h/resize_input_h ∈ [1/16,16]
resize_output_w: 0
resize_output_h: 0
# 说明: 若开启了缩放功能, 并且没有配置padding, 该场景下resize_output_w和resize_output_h才能取值为0
或不配置, 此时缩放后图像的宽和高取值来自模型文件--input_shape中的宽和高, 并且--input_shape中的高取
值范围为[16,4096], 宽取值范围为[16,1920]。

#===== 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

# H和W的填充值, 静态AIPP配置
# 类型: int32
# 取值范围: [0,32]
left_padding_size: 0
right_padding_size: 0
top_padding_size: 0
bottom_padding_size: 0
# 说明: AIPP经过padding后, 输出的H和W要与模型需要的H和W保持一致, 其中W取值要<=1080。

# 上下左右方向上padding的像素取值, 静态AIPP配置
# 类型: uint8/int8/float16
# 取值范围分别为: [0,255]、[-128, 127]、[-65504, 65504]
padding_value: 0
# 说明: 该参数取值需要与最终AIPP输出图片的数据类型保持一致。

#===== 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

# 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

# RGBA->ARGB, YUVA->AYUV交换开关
# 类型：bool
# 取值范围：true/false, true表示开启，false表示关闭
ax_swap_switch: false

# 单行处理模式（只处理抠图后的第一行）开关，保留字段，暂不支持该功能
# 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，则本功能不起作用。
# 若输入图片通道数为4，则忽略A通道或X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：[-32677,32676]
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，本功能不起作用
# 当uint8->fp16时，pixel_out_chx(i) = [pixel_in_chx(i) - mean_chn_i - min_chn_i] * var_reci_chn
# 每个通道的均值
```

```
# 类型: uint8
# 取值范围: [0, 255]
mean_chn_0: 0
mean_chn_1: 0
mean_chn_2: 0
mean_chn_3: 0

# 每个通道的最小值
# 类型: float16
# 取值范围: [0, 255]
min_chn_0: 0.0
min_chn_1: 0.0
min_chn_2: 0.0
min_chn_3: 0.0

# 每个通道方差的倒数
# 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
var_reci_chn_3: 1.0
}

#===== 静态AIPP需设置, 动态AIPP无需设置 (end)
=====
```

## 6.1.9 典型场景样例参考

### 6.1.9.1 YUV400\_U8 转 GRAY 格式

该场景以AIPP输入YUV400\_U8图像格式, 输出GRAY格式为例进行说明, 输入图像尺寸为224\*224, 有效数据区域从左上角(0, 0)像素开始; 原始网络模型Caffe/TF的C=1, H和W小于224, 例如220, 该场景下需要开启抠图功能参数crop, 并且抠图起始位置水平、垂直方向坐标load\_start\_pos\_h、load\_start\_pos\_w为0, 执行推理时, 将从(0, 0)点开始选取220\*220区域的数据。

该场景下无需配置色域转换开关csc\_switch, 并且对于同一个原始网络模型Caffe/TF, 如果AIPP输入的是YUV420SP\_U8图像, 则可以使用同一套AIPP配置, 即只取了Y通道的数据。

AIPP配置文件如下:

```
aipp_op{
  aipp_mode: static
  csc_switch: false
  crop: true
  input_format: YUV400_U8
  load_start_pos_h: 0
  load_start_pos_w: 0
  src_image_size_w: 224
  src_image_size_h: 224
  # 归一化系数需要根据用户模型实际需求配置, 如下所取常见值仅作为示例
  mean_chn_0: 128
  min_chn_0: 0.0
  var_reci_chn_0: 0.00390625
}
```

### 6.1.9.2 YUV420SP\_U8 转 BGR 格式

该场景以AIPP输入YUV420SP\_U8 ( NV12 ) 图像格式, 输出BGR格式为例进行说明, 输入图像尺寸为256\*256; 原始网络模型Caffe/TF的C=3, H和W与AIPP输入图像尺寸

相同，该场景下无需配置抠图功能参数crop，但需要配置色域转换开关csc\_switch和相应的CSC矩阵参数。

AIPP配置文件如下：

```
aipp_op {
  related_input_rank: 0
  src_image_size_w: 256
  src_image_size_h: 256
  crop: false
  input_format: YUV420SP_U8
  aipp_mode: static
  csc_switch: true
  # 如果输入的是YUV420SP_U8 ( NV21 ) 图像，则需要将rbuv_swap_switch参数设置为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
  # 归一化系数需要根据用户模型实际需求配置，如下所取常见值仅作为示例
  # 归一化系数应用于色域转换和通道交换之后的通道
  mean_chn_0: 104
  mean_chn_1: 117
  mean_chn_2: 123
  min_chn_0: 0.0
  min_chn_1: 0.0
  min_chn_2: 0.0
  var_reci_chn_0: 1.0
  var_reci_chn_1: 1.0
  var_reci_chn_2: 1.0
}
```

### 6.1.9.3 RGB888\_U8 转 RGB ( 或 BGR ) 格式

该场景以AIPP输入RGB888\_U8图像格式，输出RGB格式为例进行说明，输入图像尺寸为250\*250，有效数据区域从左上角(0, 0)像素开始；原始网络模型Caffe/TF的C=3，H和W小于250，例如240；该场景下需要开启抠图功能参数crop，并且抠图起始位置水平、垂直方向坐标load\_start\_pos\_h、load\_start\_pos\_w为0，推理时将从(0, 0)点开始选取240\*240区域的数据。

该场景下无需配置通道交换开关参数rbuv\_swap\_switch、色域转换开关参数csc\_switch和CSC矩阵参数。

AIPP配置文件如下：

```
aipp_op {
  related_input_rank: 0
  src_image_size_w: 250
  src_image_size_h: 250
  crop: true
  load_start_pos_w: 0
  load_start_pos_h: 0
  aipp_mode: static
  input_format: RGB888_U8
  csc_switch: false
  # 如果原始Caffe/TF模型需要的是BGR格式，则需要将rbuv_swap_switch参数设置为true
  rbuv_swap_switch: false
  # 归一化系数需要根据用户模型实际需求配置，此处取默认值，即不改变像素的值
}
```

```
# 若配置归一化系数，将应用于通道交换之后的通道  
}
```

## 6.2 单算子模型转换

### 6.2.1 什么是单算子描述文件

单算子描述文件是基于Ascend IR定义的单个算子的定义文件，包括算子的输入、输出及属性信息；借助该文件转换成适配昇腾AI处理器的离线模型后，可以验证单算子的功能。

### 6.2.2 如何将算子描述文件转成离线模型

**步骤1** 参见[6.2.3 配置文件样例](#)和[6.2.4 描述文件参数说明](#)构造单算子描述文件。本章节以构造format为ND的GEMM单算子为例进行说明。

**步骤2** 以CANN软件包运行用户，将[步骤1](#)构造的单算子描述文件上传到开发环境任意目录，例如\$HOME/singleop/目录下。

**步骤3** 执行如下命令生成离线模型。（如下命令中使用的目录以及文件均为样例，请以实际为准）

```
atc --singleop=$HOME/singleop/gemm.json --output=$HOME/singleop/out/op_model --  
soc_version=<soc_version>
```

关于参数的详细解释以及使用方法请参见[7.2.2.11 --singleop](#)。

**步骤4** 若提示如下信息，则说明模型转换成功。

```
ATC run success
```

成功执行命令后，在output参数指定的路径下，可查看离线模型，例如：  
0\_GEMM\_1\_2\_16\_16\_1\_2\_16\_16\_1\_2\_16\_16\_1\_2\_1\_2\_1\_2\_16\_16.om。

其中生成的离线模型文件命名规则为：序号\_opType\_输入的描述  
(dataType\_format\_shape)\_输出的描述(dataType\_format\_shape)。

dataType以及format对应枚举值请从\${INSTALL\_DIR}/include/graph/types.h文件中查看，枚举值依次递增。其中，\${INSTALL\_DIR}请替换为CANN软件安装后文件存储路径。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：  
\$HOME/Ascend/ascend-toolkit/latest。

----结束

### 6.2.3 配置文件样例

#### 6.2.3.1 单算子描述文件配置

本章节中的单算子是基于Ascend IR定义的，描述文件为json格式。关于单算子的定义请参见《[算子清单（开放态）](#)》。

单算子生成的离线模型文件命名规则为：序号\_opType\_输入的描述  
(dataType\_format\_shape)\_输出的描述(dataType\_format\_shape)。

dataType以及format对应枚举值请从\${INSTALL\_DIR}/include/graph/types.h文件中查看，枚举值从0开始依次递增。其中，\${INSTALL\_DIR}请替换为CANN软件安装后文件

存储路径。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：  
\$HOME/Ascend/ascend-toolkit/latest。

- **format为ND:**

```
[
{
  "op": "GEMM",
  "input_desc": [
    {
      "format": "ND",
      "shape": [16, 16],
      "type": "float16"
    },
    {
      "format": "ND",
      "shape": [16, 16],
      "type": "float16"
    },
    {
      "format": "ND",
      "shape": [16, 16],
      "type": "float16"
    },
    {
      "format": "ND",
      "shape": [],
      "type": "float16"
    },
    {
      "format": "ND",
      "shape": [],
      "type": "float16"
    }
  ],
  "output_desc": [
    {
      "format": "ND",
      "shape": [16, 16],
      "type": "float16"
    }
  ],
  "attr": [
    {
      "name": "transpose_a",
      "type": "bool",
      "value": false
    },
    {
      "name": "transpose_b",
      "type": "bool",
      "value": false
    }
  ]
}
```

上述单算子转换后的离线模型为：

0\_GEMM\_1\_2\_16\_16\_1\_2\_16\_16\_1\_2\_16\_16\_1\_2\_1\_2\_1\_2\_16\_16.om

- **format为NCHW:**

```
[
{
  "op": "Conv2D",
  "input_desc": [
    {
      "format": "NCHW",
      "shape": [1, 3, 16, 16],
      "type": "float16"
    },
  ],
}
```

```
{
  "format": "NCHW",
  "shape": [3, 3, 3, 3],
  "type": "float16"
},
"output_desc": [
  {
    "format": "NCHW",
    "shape": [1, 3, 16, 16],
    "type": "float16"
  }
],
"attr": [
  {
    "name": "strides",
    "type": "list_int",
    "value": [1, 1, 1, 1]
  },
  {
    "name": "pads",
    "type": "list_int",
    "value": [1, 1, 1, 1]
  },
  {
    "name": "dilations",
    "type": "list_int",
    "value": [1, 1, 1, 1]
  }
]
}
```

上述单算子转换后的离线模型为：

0\_Conv2D\_1\_0\_1\_3\_16\_16\_1\_0\_3\_3\_3\_1\_0\_1\_3\_16\_16.om

- **Tensor的实现format与原始format不同**

ATC模型转换时，会将**origin\_format**与**origin\_shape**转成离线模型需要的**format**与**shape**。

```
[
  {
    "op": "Add",
    "input_desc": [
      {
        "format": "NC1HWC0",
        "origin_format": "NCHW",
        "shape": [8, 1, 16, 4, 16],
        "origin_shape": [8, 16, 16, 4],
        "type": "float16"
      },
      {
        "format": "NC1HWC0",
        "origin_format": "NCHW",
        "shape": [8, 1, 16, 4, 16],
        "origin_shape": [8, 16, 16, 4],
        "type": "float16"
      }
    ],
    "output_desc": [
      {
        "format": "NC1HWC0",
        "origin_format": "NCHW",
        "shape": [8, 1, 16, 4, 16],
        "origin_shape": [8, 16, 16, 4],
        "type": "float16"
      }
    ]
  }
]
```

上述单算子转换后的离线模型为：

0\_Add\_1\_3\_8\_1\_16\_4\_16\_1\_3\_8\_1\_16\_4\_16\_1\_3\_8\_1\_16\_4\_16.om

- **输入指定为常量**

该场景下，支持设置为常量的输入，新增**is\_const**和**const\_value**两个参数，分别表示是否为常量以及常量取值，**const\_value**当前仅支持一维list配置，具体配置个数由shape取值决定，例如，如下样例中shape为2，则**const\_value**中列表个数为2；**const\_value**中取值类型由type决定，假设type取值为float16，则单算子编译时会自动将**const\_value**中的取值转换为float16格式的取值。

```
[
  {
    "op": "ResizeBilinearV2",
    "input_desc": [
      {
        "format": "NHWC",
        "name": "x",
        "shape": [
          4,
          16,
          16,
          16
        ],
        "type": "float16"
      },
      {
        "format": "NHWC",
        "is_const": true,
        "const_value": [49, 49],
        "name": "size",
        "shape": [
          2
        ],
        "type": "int32",
        "test": [7, 7.0]
      }
    ],
    "output_desc": [
      {
        "format": "NHWC",
        "name": "y",
        "shape": [
          4,
          48,
          48,
          16
        ],
        "type": "float"
      }
    ],
    "attr": [
      {
        "name": "align_corners",
        "type": "bool",
        "value": false
      },
      {
        "name": "half_pixel_centers",
        "type": "bool",
        "value": false
      }
    ]
  }
]
```

上述单算子转换后的离线模型为：

0\_ResizeBilinearV2\_1\_1\_4\_16\_16\_16\_3\_1\_2\_0\_1\_4\_48\_48\_16.om

- 可选输入 ( optional input ) :

```
[
  {
    "op": "MatMulV2",
    "input_desc": [
      {
        "format": "ND",
        "shape": [16, 16],
        "type": "float"
      },
      {
        "format": "ND",
        "shape": [16, 16],
        "type": "float"
      },
      {
        "format": "RESERVED",
        "shape": [16, 16],
        "type": "UNDEFINED"
      },
      {
        "format": "RESERVED",
        "shape": [16, 16],
        "type": "UNDEFINED"
      }
    ],
    "attr": [
      {
        "name": "transpose_x1",
        "type": "bool",
        "value": false
      },
      {
        "name": "transpose_x2",
        "type": "bool",
        "value": false
      }
    ],
    "output_desc": [
      {
        "format": "ND",
        "shape": [16, 16],
        "type": "float"
      }
    ]
  }
]
```

上述单算子转换后的离线模型为：

0\_MatMulV2\_0\_2\_16\_16\_0\_2\_16\_16\_26\_40\_16\_16\_26\_40\_16\_16\_0\_2\_16\_16.om

- 输入个数不确定 ( 动态输入场景 ) :

该场景下，单算子的输入个数不确定。比如AddN单算子：

- 构造的单算子json文件使用动态输入dynamic\_input参数，而不使用Tensor的名称name参数，该场景下构造的描述文件为：

该场景下算子的dynamic\_input取值必须和算子信息库中该算子定义的输入name的取值相同。

具体设置几个输入，由AddN单算子描述文件属性参数中N的取值决定，用户可以自行修改输入的个数，但是必须和属性中N的取值匹配。（该说明仅针对AddN算子生效，其他动态输入算子的约束以具体算子为准。）

```
[
  {
    "op": "AddN",
    "input_desc": [
      {
        "dynamic_input": "x",
```



```
        "format": "NCHW",
        "shape": [1,3,166,166],
        "type": "float32"
    },
    {
        "dynamic_input": "x",
        "format": "NCHW",
        "shape": [1,3,166,166],
        "type": "int32"
    },
    {
        "dynamic_input": "x",
        "format": "NCHW",
        "shape": [1,3,166,166],
        "type": "float32"
    }
],
"output_desc": [
    {
        "format": "NCHW",
        "shape": [1,3,166,166],
        "type": "float32"
    }
],
"attr": [
    {
        "name": "N",
        "type": "int",
        "value": 3
    }
]
}
```

- 构造的单算子json文件使用Tensor的名称name参数，而不使用动态输入dynamic\_input参数，该场景下构造的描述文件为：

该场景下算子的name取值必须和算子原型定义中算子的输入名称相同，根据输入的个数自动生成x0、x1、x2……。

具体设置几个Tensor名称，由AddN单算子描述文件属性参数中N的取值决定，用户可以自行修改Tensor名称的个数，但是必须和属性中N的取值匹配，例如N取值为3，则name取值分别设置为x0、x1、x2。（该说明仅针对AddN算子生效，其他动态输入算子的约束以具体算子为准。）

```
[
    {
        "op": "AddN",
        "input_desc": [
            {
                "name": "x0",
                "format": "NCHW",
                "shape": [1,3,166,166],
                "type": "float32"
            },
            {
                "name": "x1",
                "format": "NCHW",
                "shape": [1,3,166,166],
                "type": "int32"
            },
            {
                "name": "x2",
                "format": "NCHW",
                "shape": [1,3,166,166],
                "type": "float32",
            }
        ],
        "output_desc": [
```

```
{
  "format": "NCHW",
  "shape": [1,3,166,166],
  "type": "float32"
},
"attr": [
  {
    "name": "N",
    "type": "int",
    "value": 3
  }
]
}
```

上述单算子转换后的离线模型为：

0\_AddN\_0\_0\_1\_3\_166\_166\_3\_0\_1\_3\_166\_166\_0\_0\_1\_3\_166\_166\_0\_0\_1\_3\_166\_166.om

- **模糊编译场景**

如果用户无法获取算子的shape范围，又想编译一次达到多次执行推理的目的时，可以使用该特性。该场景下构造的单算子描述文件如下，当前仅支持Transformer网络模型涉及的算子：

```
[
  {
    "compile_flag":1
    "op": "Add",
    "input_desc": [
      {
        "format": "ND",
        "shape": [3,3],
        "type": "int32"
      },
      {
        "format": "ND",
        "shape": [3,3],
        "type": "int32"
      }
    ],
    "output_desc": [
      {
        "format": "ND",
        "shape": [3,3],
        "type": "int32"
      }
    ]
  }
]
```

### 6.2.3.2 多组算子描述文件配置

描述文件支持定义多组算子Json文件配置，一组配置包括算子类型、算子输入和输出信息、视算子情况决定是否包括属性信息。

如果Json文件配置了多组算子，则模型转换完成后，会生成多组算子对应的.om离线模型文件，分别以序号0\_xx，1\_xx，...形式呈现。如下配置文件只是样例，请根据实际情况进行修改。

```
[
  {
    "op": "MatMul",
    "input_desc": [
      {
        "format": "ND",
        "shape": [
          16,
```

```
        16
      ],
      "type": "float16"
    },
    ...
  ],
  "output_desc": [
    {
      "format": "ND",
      "shape": [
        16,
        16
      ],
      "type": "float16"
    }
  ],
  "attr": [
    {
      "name": "alpha",
      "type": "float",
      "value": 1.0
    },
    ...
  ]
},
{
  "op": "MatMul",
  "input_desc": [
    {
      "format": "ND",
      "shape": [
        256,
        256
      ],
      "type": "float16"
    },
    ...
  ],
  "output_desc": [
    {
      "format": "ND",
      "shape": [
        256,
        256
      ],
      "type": "float16"
    }
  ],
  "attr": [
    {
      "name": "alpha",
      "type": "float",
      "value": 1.0
    },
    ...
  ]
}
]
```

### 6.2.3.3 动态 Shape 单算子描述文件配置

该场景下的单算子分为如下类型：

- 模型编译时不指定Shape，模型执行时根据输入固定Shape，能推导出具体输出Shape：

```
[
  {
    "op": "Add",
```

```
"input_desc": [
  {
    "format": "ND",
    "shape": [-1,16],
    "shape_range": [[0, 32]],
    "type": "int64"
  },
  {
    "format": "ND",
    "shape": [-1,16],
    "shape_range": [[0, 32]],
    "type": "int64"
  }
],
"output_desc": [
  {
    "format": "ND",
    "shape": [-1,16],
    "shape_range": [[0,32]],
    "type": "int64"
  }
]
```

- 模型编译时不指定Shape，模型执行时根据输入固定Shape和常量，能推导出具体输出Shape：

```
[
  {
    "op": "TopK",
    "input_desc": [
      {
        "format": "ND",
        "shape": [-1],
        "shape_range": [[1,-1]],
        "type": "int32"
      },
      {
        "format": "ND",
        "shape": [], #推理时会传入常量
        "type": "int32"
      }
    ],
    "output_desc": [
      {
        "format": "ND",
        "shape": [-1],
        "shape_range": [[1,-1]],
        "type": "int32"
      },
      {
        "format": "ND",
        "shape": [-1],
        "shape_range": [[1,-1]],
        "type": "int32"
      }
    ],
    "attr": [
      {
        "name": "sorted",
        "type": "bool",
        "value": true
      }
    ]
  }
]
```

- 模型编译时不指定Shape，模型执行时根据输入固定Shape，无法得到算子的准确输出Shape，但可以得到输出Shape的范围。

该场景下在输出参数output\_desc中将算子输出TensorDesc中Shape为动态维度的纬度值记为“-1”，并对其“-1”的维度给出shape\_range取值范围：

```
[
  {
    "op": "Where",
    "input_desc": [
      {
        "format": "ND",
        "shape": [-1],
        "shape_range": [[1,-1]],
        "type": "int32"
      }
    ],
    "output_desc": [
      {
        "format": "ND",
        "shape": [-1, 1],
        "shape_range": [[1,-1]],
        "type": "int64"
      }
    ]
  }
]
```

## 6.2.4 描述文件参数说明

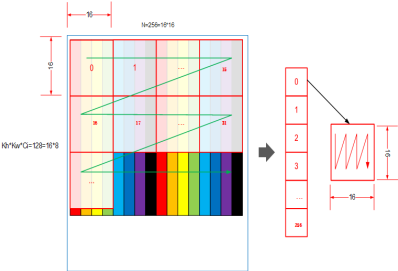
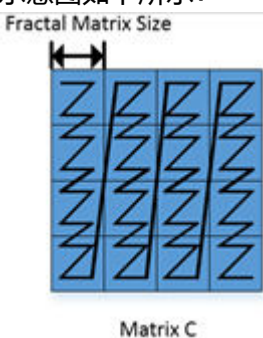
单算子描述文件是由OpDesc的数组构成的json文件，参数解释如下：

表 6-2 OpDesc 参数说明

属性名	类型	说明	是否必填
compile_flag	INT32	编译类型。取值如下： <ul style="list-style-type: none"><li>0：表示进行精确编译。精确编译是指按照用户指定的维度信息、在编译时系统内部不做任何转义直接编译，其中，AI CPU算子不受该标记影响。</li><li>1：表示进行模糊编译。模糊编译是指对于支持动态Shape的算子，在编译时系统内部对可变维度做了泛化后再进行编译。如果用户无法获取算子的Shape范围，又想编译一次达到多次执行推理的目的时，可以使用模糊编译特性。</li></ul> 默认值为0。 使用约束：当前仅支持transformer网络模型涉及的算子。	否
op	string	算子类型。	是
input_desc	TensorDesc数组	算子输入描述。	是
output_desc	TensorDesc数组	算子输出描述。	是
attr	Attr数组	算子属性。	否

表 6-3 TensorDesc 数组参数说明

属性名	类型	说明
dynamic_input	string	<p>可选。动态输入，取值必须和算子信息库中该算子定义的输入name相同。</p> <p>该参数用于设置算子动态输入的分组与动态输入的个数，例如算子原型定义中某算子的动态输入为：</p> <p>.DYNAMIC_INPUT(x,...) .DYNAMIC_INPUT(y,...)</p> <p>则表示动态输入有两组，分别为x, y。每一组的输入个数，根据dynamic_input的个数确定。具体设置原则可以参见TensorDesc数组中<b>name</b>参数的说明。</p> <ul style="list-style-type: none"><li>• 如果构造的单算子描述文件中已经设置过name参数，则该参数可选。</li><li>• 如果构造的单算子描述文件中没有name参数，则该参数必填。</li><li>• 如果同时存在dynamic_input和name参数，则以name参数设置的为准。</li></ul>

属性名	类型	说明
format	string	<p>必填。Tensor的排布格式，配置为算子原始框架支持的Format。</p> <p>当前支持的Format格式以及对应的枚举如下：</p> <ul style="list-style-type: none"><li>• NCHW: 0</li><li>• NHWC: 1</li><li>• ND: 2，表示支持任意格式。</li><li>• NC1HWC0: 3，5维数据格式。其中，C0与微架构强相关，该值等于cube单元的size，例如16；C1是将C维度按照C0切分：<math>C1=C/C0</math>。如果结果不整除，向上取整。</li><li>• FRACTAL_Z: 4，用于定义卷积权重数据格式，由FT Matrix（FT：Filter，卷积核）变换得到。FRACTAL_Z是送往Cube的最终数据格式，采用“C1HW,N1,N0,C0”的4维数据排布。数据有两层Tiling，如下图所示：</li></ul>  <p>第一层与Cube的Size相关，数据按照列的方向连续（小n）；第二层与矩阵的Size相关，数据按照行的方向连续（大Z）。</p> <ul style="list-style-type: none"><li>• FRACTAL_NZ: 29，分形格式，如Feature Map的数据存储，在cube单元计算时，输出矩阵的数据格式为NW1H1H0W0。整个矩阵被分为（H1*W1）个分形，按照column major排布，形状如N字形；每个分形内部有（H0*W0）个元素，按照row major排布，形状如z字形。考虑到数据排布格式，将NW1H1H0W0数据格式称为Nz格式。其中，H0,W0表示一个分形的大小，示意图如下所示：</li></ul>  <p>Fractal Matrix Size</p> <p>Matrix C</p> <ul style="list-style-type: none"><li>• RESERVED: 40，预留，当配置为该值，type必须配置为UNDEFINED，表示该输入为可选。</li></ul>

属性名	类型	说明
		模型转换完毕，上述Format在对应om文件名中以对应的枚举呈现，例如若输入为NHWC格式，则展示为1。
origin_format	string	可选。Tensor的原始Format。 不带此字段时，默认Tensor的实现Format与原始Format一致。
name	string	可选。Tensor的名称。算子的输入为动态输入时，需要设置该字段。 该参数用于设置每一组动态输入中，具体输入的名称，每一个输入名称为算子原型中定义的输入名称+编号，编号根据dynamic_input的个数确定，从0开始依次递增。 <ul style="list-style-type: none"><li>如果构造的单算子描述文件中已经设置过dynamic_input参数，则该参数可选。</li><li>如果构造的单算子描述文件中没有dynamic_input参数，则该参数必填。</li><li>如果同时存在dynamic_input和name参数，则以name参数设置的为准。</li></ul>
shape	int数组	必填。Tensor的Shape，例如[1, 224, 224, 3]，实际Shape乘积不能超过int32最大值（2147483647）。 <ul style="list-style-type: none"><li>静态Shape场景： Shape维度以及取值都为固定值，该场景下不需要再配置shape_range参数。</li><li>Shape为常量场景： 如果希望指定算子输入、输出Shape为标量，则该参数需要设置为"[]"形式，比如"shape": []。该场景下不需要再配置shape_range参数。</li><li>动态Shape场景，Shape取值有如下场景：<ul style="list-style-type: none"><li>Shape维度确定，但是某一维度的取值不确定，则该不确定的维度取值设置为“-1”，例如[16,-1,20,-1]，该场景下还需要与shape_range参数配合使用，用于给出“-1”维度的取值范围。例如：<pre>"shape": [-1,16], "shape_range": [[0,32]],</pre></li></ul></li></ul>
origin_shape	string	可选。Tensor的原始Shape。 不带此字段时，默认Tensor的实现Shape与原始Shape一致。



属性名	类型	说明
type	string	<p>必填。Tensor的数据格式，支持的type以及对应的枚举如下：</p> <ul style="list-style-type: none"><li>• bool: 12</li><li>• int8: 2</li><li>• uint8: 4</li><li>• int16: 6</li><li>• uint16: 7</li><li>• int32: 3</li><li>• uint32: 8</li><li>• int64: 9</li><li>• uint64: 10</li><li>• float16/fp16/half: 1</li><li>• float/float32: 0</li><li>• double: 11</li><li>• UNDEFINED: 27，未知数据类型，当配置为该值，Format必须配置为RESERVED，表示该输入为可选。</li></ul> <p>模型转换完毕，上述type在对应om文件名中以对应的枚举呈现，例如若输入为int8类型，则展示为2。</p>
shape_range	int[2]数组	<p>可选。Shape为动态时，unknow shape的取值范围。 例如，若Shape取值为[16,-1,20,-1]：其中的-1表示unknow shape。 shape_range取值为[1,128],[1,-1]：[1,128]表示从1到128的取值范围，对应Shape参数中第一个-1；[1,-1]表示从1到无穷大的取值范围，对应Shape参数中第二个-1。</p>
is_const	bool	<p>可选，表示输入是否为常量： true：常量。 false：非常量。</p>
const_value	list	<p>可选，常量取值。 当前仅支持一维list配置，list中具体配置个数由Shape取值决定。例如，Shape取值为2，则<b>const_value</b>中列表个数为2。 取值类型由type决定，假设type取值为float16，则单算子编译时会自动将const_value中的取值转换为float16格式的取值。</p>

表 6-4 Attr 数组参数说明

属性名	类型	说明
name	string	必填。属性名。

属性名	类型	说明
type	string	必填。属性值的类型，支持的类型有： <ul style="list-style-type: none"><li>• bool</li><li>• int</li><li>• float</li><li>• string</li><li>• list_bool</li><li>• list_int</li><li>• list_float</li><li>• list_string</li><li>• list_list_int</li><li>• data_type</li></ul>
value	由type的取值决定	必填。属性值，根据type不同，属性值不同，举例如下： <ul style="list-style-type: none"><li>• bool: true/false</li><li>• int: 10</li><li>• float: 1.0</li><li>• string: “NCHW”</li><li>• list_bool: [false, true]</li><li>• list_int: [1, 224, 224, 3]</li><li>• list_float: [1.0, 0.0]</li><li>• list_string: ["str1","str2"]</li><li>• list_list_int: [[1, 3, 5, 7], [2, 4, 6, 8]]</li><li>• data_type: "DT_FLOAT"或该枚举值对应的数字，例如0。</li></ul> 其他取值请参见\${INSTALL_DIR}/include/graph/types.h中DataType的枚举值或枚举值对应的数字。其中，\${INSTALL_DIR}请替换为CANN软件安装后文件存储路径。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：\$HOME/Ascend/ascend-toolkit/latest。

# 7 参数说明

[参数概览](#)

[基础功能参数](#)

[高级功能参数](#)

## 7.1 参数概览

### 总体约束

在进行模型转换前，请务必查看如下约束要求：

- 如果要将FasterRCNN等网络模型转成适配昇腾AI处理器的离线模型，则务必参见[8.1 定制网络修改 \(Caffe\)](#)先修改prototxt模型文件。
- 支持原始框架类型为Caffe、TensorFlow、MindSpore、ONNX的模型转换，当原始框架类型为Caffe、MindSpore、ONNX时，输入数据类型为FP32、FP16（通过设置入参--input\_fp16\_nodes实现，MindSpore框架不支持该参数）、UINT8（通过配置数据预处理实现）；当原始框架类型为TensorFlow时，输入数据类型为FP16、FP32、UINT8、INT32、INT64、BOOL。
- 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。
- 对于Caffe框架网络模型：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。
- 只支持[9.1 算子规格参考](#)中的算子，并需满足算子限制条件。
- 由于软件约束（动态shape场景下暂不支持输入数据为DT\_INT8），量化后的部署模型使用ATC工具进行模型转换时，不能使用动态shape相关参数，例如[7.2.2.8 --dynamic\\_batch\\_size](#)和[7.2.2.9 --dynamic\\_image\\_size](#)等，否则模型转换会失败。

**图7-1**列出了所有芯片共用的ATC参数（参数只在某些芯片下使用的未列出），其中表示参数互斥，不能同时使用；关联参数表示需要相互配合或者某些场景下需要配合使用。通过该图，您可以快速了解ATC中的部分参数，更多详细参数请参见**表7-1**。

[illegible]

- 如果通过**atc --help**命令查询出的参数未解释在**表7-1**，则说明该参数预留或适用于其他芯片版本，用户无需关注。
- 使用atc命令进行模型转换时，命令有两种方式，用户根据实际情况进行选择：
  1. **atc param1=value1 param2=value2 ...**（value值前面不能有空格，否则会导致截断，param取的value值为空）
  2. **atc param1 value1 param2 value2 ...**
- 参数是否必选以--mode为0和3为准。

ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
7.2.1.1 --help或--h	显示帮助信息。	否	不涉及
7.2.1.2 --mode	运行模式。	否	0
7.2.2.1 --model	原始模型文件路径与文件名。	是	不涉及
7.2.2.2 --weight	权重文件路径与文件名。	否	不涉及

ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
<a href="#">7.2.2.3 --om</a>	需要转换为json格式的离线模型或原始模型文件的路径和文件名。	否	不涉及
<a href="#">7.2.2.4 --framework</a>	原始框架类型。	是	不涉及
<a href="#">7.2.2.5 --input_format</a>	输入数据格式。	否	Caffe默认为NCHW；TensorFlow默认为NHWC
<a href="#">7.2.2.6 --input_shape</a>	模型输入数据的shape。	否	不涉及
<a href="#">7.2.2.7 --input_shape_range</a>	指定模型输入数据的shape范围。	否	不涉及
<a href="#">7.2.2.8 --dynamic_batch_size</a>	设置动态batch档位参数，适用于执行推理时，每次处理图片数量不固定的场景。	否	不涉及
<a href="#">7.2.2.9 --dynamic_image_size</a>	设置输入图片的动态分辨率参数。适用于执行推理时，每次处理图片宽和高不固定的场景。	否	不涉及
<a href="#">7.2.2.10 --dynamic_dims</a>	设置ND格式下动态维度的档位。适用于执行推理时，每次处理任意维度的场景。	否	不涉及
<a href="#">7.2.2.11 --singleop</a>	单算子定义文件，将单个算子Json文件转换成适配昇腾AI处理器的离线模型。	否	不涉及

ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
<b>7.2.3.1 --output</b>	<ul style="list-style-type: none"><li>如果是开源框架的网络模型，存放转换后的离线模型的路径以及文件名。</li><li>如果是单算子描述文件，存放转换后的单算子模型的路径。</li></ul>	是	不涉及
<b>7.2.3.2 --output_type</b>	指定网络输出数据类型或指定某个输出节点的输出类型。	否	不涉及
<b>7.2.3.3 --check_report</b>	预检结果保存文件路径和文件名。	否	check_result.json
<b>7.2.3.4 --json</b>	离线模型或原始模型文件转换为json格式文件的路径和文件名。	否	不涉及
<b>7.2.4.1 --soc_version</b>	模型转换时指定芯片版本。	是	不涉及
<b>7.2.4.2 --core_type</b>	设置网络模型使用的Core类型，若网络模型中包括Cube算子，则只能使用AiCore。	否	AiCore
<b>7.2.4.3 --aicore_num</b>	设置模型编译时使用的aicore数目。	否	默认值为最大值
<b>7.3.1.1 --out_nodes</b>	指定输出节点。	否	不涉及
<b>7.3.1.2 --input_fp16_nodes</b>	指定输入数据类型为FP16的输入节点名称。	否	不涉及
<b>7.3.1.3 --insert_op_conf</b>	插入算子的配置文件路径与文件名，例如aipp预处理算子。	否	不涉及

ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
<a href="#">7.3.1.4 --op_name_map</a>	扩展算子（非标准算子）映射配置文件路径和文件名，不同的网络中某扩展算子的功能不同，可以指定该扩展算子到具体网络中实际运行的扩展算子的映射。	否	不涉及
<a href="#">7.3.1.5 --is_input_adjust_hw_layout</a>	用于指定网络输入数据类型是否为FP16，数据格式是否为NC1HWC0	否	false
<a href="#">7.3.1.6 --is_output_adjust_hw_layout</a>	用于指定网络输出的数据类型是否为FP16，数据格式是否为NC1HWC0	否	false
<a href="#">7.3.2.1 --disable_reuse_memory</a>	内存复用开关。	否	0
<a href="#">7.3.2.2 --fusion_switch_file</a>	融合开关配置文件路径以及文件名。	否	不涉及
<a href="#">7.3.2.3 --enable_scope_fusion_passes</a>	指定编译时需要生效的融合规则列表。	否	不涉及
<a href="#">7.3.2.4 --enable_small_channel</a>	是否使能small channel的优化，使能后在channel<=4的首层卷积会有性能收益。	否	0
<a href="#">7.3.2.5 --buffer_optimize</a>	是否开启数据缓存优化。	否	l2_optimize
<a href="#">7.3.2.6 --mdl_bank_path</a>	加载模型调优后自定义知识库的路径。	否	\${HOME}/ascend/latest/data/aoe/custom/graph/<soc_version>
<a href="#">7.3.3.1 --precision_mode</a>	设置网络模型的精度模式。	否	force_fp16

ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
<b>7.3.3.2 --op_precision_mode</b>	设置具体某个算子的精度模式，通过该参数可以为不同的算子设置不同的精度模式。	否	不涉及
<b>7.3.3.3 --modify_mixlist</b>	混合精度场景下，修改算子使用混合精度名单。	否	不涉及
<b>7.3.3.6 --keep_dtype</b>	保持原始模型编译时个别算子的计算精度不变。	否	不涉及
<b>7.3.3.7 --auto_tune_mode</b>	设置算子的自动调优模式。	否	不涉及
<b>7.3.3.8 --op_bank_path</b>	加载Auto Tune调优后自定义知识库的路径。	否	<b>参数默认值：</b> <ul style="list-style-type: none"><li>GA调优后默认自定义知识库路径：\$ {HOME}/ascend/latest/data/aoe/custom/op/&lt;soc_version&gt;/cube</li><li>RL调优后默认自定义知识库路径：\$ {HOME}/ascend/latest/data/aoe/custom/op/&lt;soc_version&gt;/vector</li></ul>
<b>7.3.3.4 --op_select_implmode</b>	选择算子是高精度实现还是高性能实现。	否	high_performance
<b>7.3.3.5 --optypelist_for_implmode</b>	列举算子optype的列表，该列表中的算子使用--op_select_implmode参数指定的模式。	否	不涉及
<b>7.3.3.9 --op_debug_level</b>	TBE算子编译debug功能开关。	否	0
<b>7.3.4.1 --dump_mode</b>	是否生成带shape信息的json文件。	否	0



ATC参数名称	参数简述（具体说明见参数描述章节）	是否必选	默认值
7.3.4.2 --log	设置ATC模型转换过程中显示日志的级别。	否	null
7.3.4.3 --debug_dir	用于配置保存模型转换、网络迁移过程中算子编译生成的调试相关过程文件的路径，包括算子.o/.json/.cce等文件。	否	./kernel_meta
7.3.4.4 --op_compiler_cache_mode	用于配置算子编译磁盘缓存模式。	否	disable
7.3.4.5 --op_compiler_cache_dir	用于配置算子编译磁盘缓存的目录。	否	\$HOME/atc_data
7.3.4.6 --display_model_info	模型编译时或对已有的离线模型，查询模型占用的关键资源信息、编译与运行环境等信息。	否	0

## 7.2 基础功能参数

### 7.2.1 总体选项

#### 7.2.1.1 --help 或--h

##### 功能说明

显示帮助信息。

##### 关联参数

无。

##### 参数取值

无。

##### 推荐配置及收益

无。

## 示例

```
atc --help
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.1.2 --mode

## 功能说明

运行模式。

## 关联参数

- 若**7.2.1.2 --mode**取值为1或5，则需要与**7.2.2.3 --om**、**7.2.3.4 --json**参数配合使用。如果将原始模型文件转换成带shape信息的json文件，则还需要与**7.3.4.1 --dump\_mode**参数配合使用。
- 若**7.2.1.2 --mode**取值为6，则只需要与**7.2.2.3 --om**参数配合使用。

## 参数取值

参数值：

- 0：生成适配昇腾AI处理器的离线模型。
- 1：离线模型或原始模型文件转json，方便查看模型中的参数信息。
- 3：仅做预检，检查模型文件的内容是否合法。
- 5：GE（Graph Engine）dump图结构文件转json，用于解析GE图编码过程中产生的dump图结构（ge\_proto\*.txt格式文件，ge\_onnx\*.pbtxt暂不支持），然后将dump图结构转换成json文件，方便用户定位。
- 6：针对已有的**离线模型**，显示模型信息，包括模型占用的关键资源信息、编译与运行环境等信息。

参数值约束：

若**7.2.1.2 --mode**取值为5，需要先设置如下环境变量，生成dump图结构文件，然后才能进行下一步的转换：

打印模型转换过程中各个阶段的图描述信息。

```
export DUMP_GE_GRAPH=1
```

上述环境变量控制dump图的内容多少：取值为1，全量dump；取值为2：不含权重等数据的基本版dump；取值为3：只显示节点关系的精简版dump。

设置上述环境变量后，还可以设置如下环境变量，控制dump图的个数。

```
export DUMP_GRAPH_LEVEL=1
```

取值为1，dump所有图；取值为2：dump除子图外的所有图；取值为3：dump最后的生成图；默认值为2。

设置上述变量后，在执行atc命令的当前路径会生成相应的图文件。

参数默认值：0

## 推荐配置及收益

无。

## 示例

### 📖 说明

使用atc命令进行模型转换时，命令有两种方式，用户根据实际情况进行选择，本章节以选择第一种方式为例进行说明：

1. **atc param1=value1 param2=value2 ...** (value值前面不能有空格，否则会导致截断，param取的value值为空)
2. **atc param1 value1 param2 value2 ...**

- 参数值取值为0：

```
atc --mode=0 --framework=0 --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version>
```

- 参数值取值为1：

- 离线模型转换为json

```
--mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

- 原始模型文件转换为json

```
--mode=1 --om=$HOME/module/resnet50.prototxt --json=$HOME/module/out/caffe_resnet50.json --framework=0
```

- 参数取值为5：

GE dump图结构文件转json

```
--mode=5 --om=$HOME/module/ge_proto_00000000_PreRunBegin.txt --json=$HOME/module/out/ge_proto.json
```

- 参数值取值为6：

```
atc --mode=6 --om=$HOME/module/out/caffe_resnet50.om
```

命令执行完毕，屏幕会打印类似如下信息：

```
===== Display Model Info start =====
# 模型转换使用的atc命令
Original Atc command line: ${INSTALL_DIR}/bin/atc.bin --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version> --display_model_info=1
# ATC软件版本信息、soc_version版本信息、原始框架信息
system info: atc_version[xxx], soc_version[xxx], framework_type[xxx].
# 运行时的占用内存、逻辑stream数目、event数目
resource info: memory_size[xxx B], weight_size[xxx B], stream_num[xxx], event_num[xxx].
# 离线模型文件中各分区大小、包括ModelDef、权重、tbekernels、taskinfo占用的大小等
om info: modeldef_size[xxx B], weight_data_size[xxx B], tbe_kernels_size[xxx B], cust_aicpu_kernel_store_size[xxx B], task_info_size[xxx B].
===== Display Model Info end =====
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

## 7.2.2 输入选项

### 7.2.2.1 --model

#### 功能说明

原始网络模型文件路径与文件名。

#### 关联参数

当原始模型为Caffe框架时，需要和7.2.2.2 --weight参数配合使用。

#### 参数取值

**参数值：**模型文件路径与文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

#### 推荐配置及收益

无。

#### 示例

```
--model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel
```

#### 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.2 --weight

#### 功能说明

原始网络模型权重文件路径与文件名。

当原始网络模型是Caffe时需要指定。

#### 关联参数

当原始模型为Caffe框架时，需要和7.2.2.1 --model参数配合使用。

## 参数取值

**参数值：**权重文件路径与文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

## 推荐配置及收益

无。

## 示例

```
--model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.3 --om

## 功能说明

离线模型（.om）、原始模型文件（例如Caffe框架的.prototxt，TensorFlow框架的.pb等）、GE dump图结构文件（.txt）的路径和文件名。

## 关联参数

- 若7.2.1.2 --mode取值为1：
  - 离线模型转换为json  
该参数需要与7.2.1.2 --mode=1、7.2.3.4 --json参数配合使用。
  - 原始模型文件转换为json  
该参数需要与7.2.1.2 --mode=1、7.2.3.4 --json参数、7.2.2.4 --framework配合使用。
- 若7.2.1.2 --mode取值为5：  
GE dump图结构文件转json，该参数需要与7.2.1.2 --mode=5、7.2.3.4 --json参数配合使用。
- 若7.2.1.2 --mode取值为6：  
则只需要与7.2.1.2 --mode参数配合使用。

## 参数取值

**参数值：**离线模型（.om）、原始模型文件（例如Caffe框架的.prototxt，TensorFlow框架的.pb）或GE dump图结构文件（.txt）的路径。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

## 推荐配置及收益

无。

## 示例

- 若7.2.1.2 --mode取值为1

- 离线模型转换为json:

```
--mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

在转换后的json文件中，可以查看原始模型转换为该离线模型时，使用的基础版本号，以及当时转换使用的atc命令，示例如下：（如下示例中的版本号都为样例，请以实际查询的为准）

说明：离线模型转换为json文件可以查看基础版本号，必须保证使用的软件为1.77.22.6.220及之后的版本，之前版本无法查阅该信息；atc\_cmdline参数必须为1.78.23.34.230及之后的软件版本方可查阅。

```
{
  "key": "opp_version",
  "value": {
    "s": "1.78.23.34.230"
  }
},
...
{
  "key": "atc_version",
  "value": {
    "s": "1.78.23.34.230"
  }
},
...
{
  "key": "atc_cmdline",
  "value": {
    "s": "xxx/atc.bin --model ./resnet50.prototxt --weight ./resnet50.caffemodel --framework 0
--output ./out/resnet50 --soc_version Ascend310"
  }
},
...
{
  "key": "soc_version",
  "value": {
    "s": "Ascend310Ascend610SD3403"
  }
},
...
```

- 原始模型文件转换为json

```
--mode=1 --om=$HOME/module/resnet50.prototxt --json=$HOME/module/out/caffe_resnet50.json --framework=0
```

- 若7.2.1.2 --mode取值为5

GE dump图结构文件转json:

```
--mode=5 --om=$HOME/module/ge_proto_00000000_PreRunBegin.txt --json=$HOME/module/out/ge_proto.json
```

- 若7.2.1.2 --mode取值为6

```
atc --mode=6 --om=$HOME/module/out/caffe_resnet50.om
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.4 --framework

## 功能说明

原始网络模型框架类型。

## 关联参数

无。

## 参数取值

参数值：

- 0: Caffe
- 1: MindSpore
- 3: TensorFlow
- 5: ONNX

参数值约束：

- 当**7.2.1.2 --mode**为1时，该参数可选，可以指定Caffe、TensorFlow、ONNX原始模型转成json，不指定时默认为离线模型转json，如果指定时需要保证**7.2.2.3 --om**模型和**7.2.2.4 --framework**类型对应一致，例如：  

```
--mode=1 --framework=0 --om=$HOME/module/resnet50.prototxt  
--mode=1 --framework=3 --om=$HOME/module/resnet_v1_50.pb  
--mode=1 --framework=5 --om=$HOME/module/resnet50.onnx
```
- 当**7.2.1.2 --mode**为0或3时，该参数必选，可以指定Caffe、TensorFlow、MindSpore或ONNX。
- 当取值为0时，即为Caffe框架网络模型，模型包括后缀为prototxt的模型文件和后缀为caffemodel的权重文件，并且此两个文件的op name和op type必须保持名称一致（包括大小写）。
- 当取值为3时，即为TensorFlow框架网络模型，只支持FrozenGraphDef格式，即尾缀为pb的模型文件，pb文件采用protobuf格式存储，网络模型和权值数据都储存在同一个文件中。
- 当取值为5时，即为ONNX格式网络模型，仅支持ai.onnx算子域中opset v11版本的算子，用户也可以将其他opset版本的算子（比如opset v9），通过PyTorch转换成opset v11版本的ONNX算子；而使用PyTorch训练出的pth模型需要转化为ONNX格式的模型，才能进行模型转换。
- 当取值为1时，即为MindSpore框架网络模型时，请务必查看如下限制：
  - 模型转换时，仅支持后缀为\*.air的模型文件。

- [7.2.1.2 --mode](#)只支持配置为0；[7.2.2.5 --input\\_format](#)只支持配置为NCHW、[7.3.1.1 --out\\_nodes](#)、[7.3.1.6 --is\\_output\\_adjust\\_hw\\_layout](#)、[7.3.1.2 --input\\_fp16\\_nodes](#)、[7.3.1.5 --is\\_input\\_adjust\\_hw\\_layout](#)、[7.3.1.4 --op\\_name\\_map](#)参数不支持在MindSpore框架下使用。

## 推荐配置及收益

无。

## 示例

- Caffe框架：  
`--mode=0 --framework=0 --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version>`
- MindSpore框架：  
`--mode=0 --framework=1 --model=$HOME/module/ResNet50.air --output=$HOME/module/out/ResNet50_mindspore --soc_version=<soc_version>`
- TensorFlow框架：  
`--mode=0 --framework=3 --model=$HOME/module/resnet_v1_50.pb --output=$HOME/module/out/tf_resnet_v1_50 --soc_version=<soc_version>`
- ONNX网络模型：  
`--mode=0 --framework=5 --model=$HOME/module/resnet50.onnx --output=$HOME/module/out/onnx_resnet50 --soc_version=<soc_version>`

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.5 --input\_format

## 功能说明

输入数据格式。

## 关联参数

无。

## 参数取值

参数值：

- 当原始框架为Caffe时，支持NCHW、ND（表示支持任意维度格式， $N \leq 4$ ）两种格式，默认为NCHW。
- 当原始框架为ONNX时，支持NCHW、NCDHW、ND（表示支持任意维度格式， $N \leq 4$ ）三种格式，默认为NCHW。
- 当原始框架是TensorFlow时，支持NCHW、NHWC、ND、NCDHW、NDHWC五种输入格式，默认为NHWC。



- 如果TensorFlow模型是通过ONNX模型转换工具输出的，则该参数必填，且值为NCHW。
- 如果原始模型中含有带data\_format入参的算子，则该参数必填，推荐取值为ND，模型转换过程中会根据data\_format属性的算子，推导出具体的format。若用户无法确定输入数据格式，则推荐指定为ND。
- 当原始框架为MindSpore时，只支持配置为NCHW。

**参数默认值：**Caffe、ONNX默认为NCHW；TensorFlow默认为NHWC。

**参数值约束：**

- 如果模型转换时开启AIPP，在进行推理业务时，输入图片数据要求为NHWC排布，该场景下最终与AIPP连接的输入节点的格式被强制改成NHWC，可能与atc模型转换命令中**7.2.2.5 --input\_format**参数指定的格式不一致。
- 如果同时配置了**7.3.1.3 --insert\_op\_conf**参数，则**7.2.2.5 --input\_format**参数只能配置为NCHW、NHWC。

## 推荐配置及收益

无。

## 示例

```
--input_format=NCHW
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.6 --input\_shape

## 功能说明

指定模型输入数据的shape。

## 关联参数

无。

## 参数取值

**参数值：**模型输入的shape信息，例如：

"input\_name1:n1,c1,h1,w1;input\_name2:n2,c2,h2,w2"。指定的节点必须放在双引号中，节点中间使用英文分号分隔。input\_name必须是转换前的网络模型中的节点名称。

**参数值约束：**若原始模型中输入数据的某个维度不固定，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，该场景下用户可以进行如下操作：

1. 设置为固定取值，例如，取值为“1, 2, 3...” ，用于将输入数据某个维度不固定的原始模型转换为固定维度的离线模型。
2. 如果模型输入数据为标量（例如BOOL类型），该场景下无需配置对应节点的7.2.2.6 --input\_shape参数。
3. 设置为“-1”，与7.2.2.8 --dynamic\_batch\_size参数配合使用，用于设置动态batch档位参数，详细使用请参见7.2.2.8 --dynamic\_batch\_size参数。

## 推荐配置及收益

无。

## 示例

```
--input_shape="input_name1:n1,c1,h1,w1;input_name2:n2,c2,h2,w2"
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.2.7 --input\_shape\_range

## 功能说明

指定模型输入数据的shape范围。该参数预留，当前版本暂不支持。

## 关联参数

该参数不能与7.2.2.8 --dynamic\_batch\_size、7.2.2.9 --dynamic\_image\_size、7.2.2.10 --dynamic\_dims同时使用。

## 参数取值

**参数值：**模型输入数据的shape范围信息，例如："input\_name1:[n1~n2,c1,h1,w1];input\_name2:[n2,c2,h2,w2]"。指定的节点必须放在双引号中，节点中间使用英文分号分隔。input\_name必须是转换前的网络模型中的节点名称。

**参数值约束：**

- shape范围信息必须放在英文[]中。
- 该参数不限定维度，维度中的任一值都可以由用户指定取值范围。
- 如果用户不想指定维度的取值，则可以将设置为-1，表示此维度可以使用>=1的任意取值。

## 推荐配置及收益

无。

## 示例

```
--input_shape_range="input1:[8~20,3,5,-1];input2:[5,3~9,10,-1]"
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

- **使用约束：**
  - 该参数只适用于TensorFlow和ONNX网络模型，不适用于Caffe和MindSpore网络模型。
  - 若使用该参数时，同时通过[7.3.1.3 --insert\\_op\\_conf](#)设置了AIPP功能，则AIPP输出图片的宽和高要在[7.2.2.7 --input\\_shape\\_range](#)所设置的范围内。
- **接口约束：**

如果模型转换时通过该参数设置了shape的范围，则使用应用工程进行模型推理时，需要在[aclmdlExecute](#)接口之前，增加[aclmdlSetDatasetTensorDesc](#)接口，用于设置真实的shape范围。

关于[aclmdlSetDatasetTensorDesc](#)接口的具体使用方法，请参见《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册“AscendCL API参考”。

### 7.2.2.8 --dynamic\_batch\_size

## 功能说明

设置动态BatchSize参数，适用于执行推理时，每次处理图片数量不固定的场景。

在某些推理场景，如检测出目标后再执行目标识别网络，由于目标个数不固定导致目标识别网络输入BatchSize不固定。如果每次推理都按照最大的BatchSize或最大分辨率进行计算，会造成计算资源浪费。因此，推理需要支持动态BatchSize和动态分辨率的场景，使用ATC工具时，通过该参数设置支持的BatchSize，通过[7.2.2.9 --dynamic\\_image\\_size](#)参数设置支持的分辨率档位。

模型转换完成后，在生成的om模型中，会新增一个输入，在模型推理时通过该新增的输入提供具体的BatchSize值。例如，a输入的BatchSize是动态的，在om模型中，会有与a对应的b输入来描述a的具体BatchSize。

## 关联参数

该参数需要与[7.2.2.6 --input\\_shape](#)配合使用，不能与[7.2.2.9 --dynamic\\_image\\_size](#)、[7.2.2.10 --dynamic\\_dims](#)、[7.2.2.7 --input\\_shape\\_range](#)同时使用。

## 参数取值

**参数值：**档位数，例如"1,2,4,8"。

**参数值格式：**指定的参数必须放在双引号中，每一组参数中间使用英文逗号分隔。

**参数值约束：**档位数取值范围为(1,100]，既必须设置至少2个档位，最多支持100档配置；档位之间通过英文逗号分隔，每个档位数值建议限制为：[1~2048]。

## 推荐配置及收益

- 如果用户设置的档位数值过大或档位过多，可能会导致模型转换失败，此时建议用户减少档位或调低档位数值。
- 如果用户设置的档位数值过大或档位过多，在运行环境执行推理时，建议执行 **swapon -a** 命令关闭swap交换区间作为内存的功能，防止出现由于内存不足，将swap交换空间作为内存继续调用，导致运行环境异常缓慢的情况。

## 示例

```
--input_shape="data:-1,3,416,416;img_info:-1,4" --dynamic_batch_size="1,2,4,8"
```

其中，“--input\_shape”中的“-1”表示设置动态BatchSize。则ATC在模型编译时，支持的输入组合档数分别为：

第0档：data(1,3,416,416)+img\_info(1,4)

第1档：data(2,3,416,416)+img\_info(2,4)

第2档：data(4,3,416,416)+img\_info(4,4)

第3档：data(8,3,416,416)+img\_info(8,4)

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

- **使用约束：**
  - 若用户执行推理业务时，每次处理的图片数量不固定，则可以通过配置该参数来动态分配每次处理的图片数量。例如用户执行推理业务时需要每次处理2张，4张，8张图片，则可以配置为2,4,8，申请了档位后，模型推理时会根据实际档位申请内存。
  - 如果用户设置了动态BatchSize，同时又通过**7.3.1.3 --insert\_op\_conf**参数设置了动态AIPP功能：  
实际推理时，调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>**aclmdlSetInputAIPP**”接口，设置动态AIPP相关参数值时，需确保batchSize要设置为最大Batch数。
  - 某些场景下，通过该参数设置动态BatchSize特性后，生成的离线模型网络结构会与固定BatchSize场景下的不同，推理性能可能存在差异。
- **接口约束：**

如果模型转换时通过该参数设置了动态BatchSize，则使用应用工程进行模型推理时，需要在**aclmdlExecute**接口之前，增加**aclmdlSetDynamicBatchSize**接口，用于设置真实的BatchSize档位。

关于**aclmdlSetDynamicBatchSize**接口的具体使用方法，请参见《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册“AscendCL API参考>模型加载与执行”章节。

### 7.2.2.9 --dynamic\_image\_size

#### 功能说明

设置输入图片的动态分辨率参数。适用于执行推理时，每次处理图片宽和高不固定的场景。

#### 关联参数

该参数需要与[7.2.2.6 --input\\_shape](#)配合使用，不能与[7.2.2.8 --dynamic\\_batch\\_size](#)、[7.2.2.10 --dynamic\\_dims](#)、[7.2.2.7 --input\\_shape\\_range](#)同时使用。

#### 参数取值

**参数值：**动态分辨率参数，例如  
"imagesize1\_height,imagesize1\_width;imagesize2\_height,imagesize2\_width"。

**参数值格式：**指定的参数必须放在双引号中，每一组参数中间使用英文分号分隔，组内参数使用英文逗号分隔。

**参数值约束：**档位数取值范围为(1,100]，既必须设置至少2个档位，最多支持100档配置；档位之间通过英文逗号分隔。

#### 推荐配置及收益

- 如果用户设置的分辨率数值过大或档位过多，可能会导致模型转换失败，此时建议用户减少档位或调低档位数值。
- 如果用户设置的分辨率数值过大或档位过多，在运行环境执行推理时，建议执行 **swapoff -a** 命令关闭swap交换区间作为内存的功能，防止出现由于内存不足，将swap交换空间作为内存继续调用，导致运行环境异常缓慢的情况。

#### 示例

```
--input_shape="data:8,3,-1,-1;img_info:8,4,-1,-1" --dynamic_image_size="416,416;832,832"
```

其中，“--input\_shape”中的“-1”表示设置动态分辨率。则ATC在编译模型时，支持的输入组合档数分别为：

第0档：data(8,3,416,416)+img\_info(8,4,416,416)

第1档：data(8,3,832,832)+img\_info(8,4,832,832)

#### 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

#### 依赖约束

- **使用约束：**
  - 如果用户设置了动态分辨率，则请确保不同档位的分辨率能在原生框架下正常推理。

- 如果用户设置了动态分辨率，实际推理时，使用的数据集图片大小需要与具体使用的分辨率相匹配。
  - 如果用户设置了动态分辨率，即输入图片的宽和高不确定，同时又通过 [7.3.1.3 --insert\\_op\\_conf](#) 参数设置了静态AIPP功能：该场景下，AIPP配置文件中不能开启Crop和Padding功能，并且需要将配置文件中的 `src_image_size_w` 和 `src_image_size_h` 取值设置为0。
  - 如果用户设置了动态分辨率，同时又通过 [7.3.1.3 --insert\\_op\\_conf](#) 参数设置了动态AIPP功能：  
实际推理时，调用《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册中的“AscendCL API参考>模型加载与执行>`aclmdlSetInputAIPP`”接口，设置动态AIPP相关参数值时，不能开启Crop和Padding功能。该场景下，还需要确保通过[aclmdlSetInputAIPP](#)接口设置的宽和高与《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册“AscendCL API参考>模型加载与执行>`aclmdlSetDynamicHWSIZE`”接口设置的宽、高相等，都必须设置成动态分辨率最大档位的宽、高。
  - 某些场景下，通过该参数设置动态分辨率特性后，生成的离线模型网络结构会与固定分辨率场景下的不同，推理性能可能存在差异。
- **接口约束：**  
如果模型转换时通过该参数设置了动态分辨率，则使用应用工程进行模型推理时，需要在[aclmdlExecute](#)接口之前，增加[aclmdlSetDynamicHWSIZE](#)接口，用于设置真实的分辨率。  
关于[aclmdlSetDynamicHWSIZE](#)接口的具体使用方法，请参见《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册“AscendCL API参考>模型加载与执行”章节。

### 7.2.2.10 --dynamic\_dims

#### 功能说明

设置ND格式下动态维度的档位。适用于执行推理时，每次处理任意维度的场景。

为支持Transformer等网络在输入格式的维度不确定的场景，通过该参数实现ND格式下任意维度的档位设置。ND表示支持任意格式，当前 $N \leq 4$ 。

#### 关联参数

该参数需要与[7.2.2.6 --input\\_shape](#)、[7.2.2.5 --input\\_format](#)配合使用，不能与[7.2.2.8 --dynamic\\_batch\\_size](#)、[7.2.2.9 --dynamic\\_image\\_size](#)、[7.2.2.7 --input\\_shape\\_range](#)、[7.3.1.3 --insert\\_op\\_conf](#)同时使用。

#### 参数取值

**参数值：**通过“dim1,dim2,dim3;dim4,dim5,dim6;dim7,dim8,dim9”的形式设置。

**参数值格式：**所有档位必须放在双引号中，每档中间使用英文分号分隔，每档中的dim值与[7.2.2.6 --input\\_shape](#)参数中的-1标识的参数依次对应，[7.2.2.6 --input\\_shape](#)参数中有几个-1，则每档必须设置几个维度。

**参数值约束：**

- 当前支持的输入shape的动态维度档位最多为4维，每个维度档位数取值范围为(1,100]，既必须设置至少2个档位，最多支持100档配置，建议配置为3~4档。

## 推荐配置及收益

无。

## 示例

- 若网络模型只有一个输入：

每档中的dim值与**7.2.2.6 --input\_shape**参数中的-1标识的参数依次对应，**7.2.2.6 --input\_shape**参数中有几个-1，则每档必须设置几个维度。例如：

ATC参数取值为：

```
--input_shape="data:1,-1" --dynamic_dims="4;8;16;64" --input_format=ND
```

则ATC在编译模型时，支持的data算子的shape为1,4; 1,8; 1,16;1,64。

ATC参数取值为：

```
--input_shape="data:1,-1,-1" --dynamic_dims="1,2;3,4;5,6;7,8" --input_format=ND
```

则ATC在编译模型时，支持的data算子的shape为1,1,2; 1,3,4; 1,5,6; 1,7,8。

- 若网络模型有多个输入：

每档中的dim值与网络模型输入参数中的-1标识的参数依次对应，网络模型输入参数中有几个-1，则每档必须设置几个维度。例如网络模型有三个输入，分别为data(1,1,40,T)，label(1,T)，mask(T,T)，其中T为动态可变。则配置示例为：

```
--input_shape="data:1,1,40,-1; label:1,-1; mask:-1,-1" --dynamic_dims="20,20,1,1; 40,40,2,2; 80,60,4,4" --input_format=ND
```

在ATC编译模型时，支持的输入dims组合档数分别为：

第0档：data(1,1,40,20)+label(1,20)+mask(1,1)

第1档：data(1,1,40,40)+label(1,40)+mask(2,2)

第2档：data(1,1,40,80)+label(1,60)+mask(4,4)

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 接口约束

如果模型转换时通过该参数设置了动态维度，则使用应用工程进行模型推理时，需要在[aclmdlExecute](#)接口之前，增加[aclmdlSetInputDynamicDims](#)接口，用于设置真实的维度。

关于[aclmdlSetInputDynamicDims](#)接口的具体使用方法，请参见《[应用软件开发指南 \(C&C++, 开放态\)](#)》手册“AscendCL API参考>模型加载与执行”章节。

### 7.2.2.11 --singleop

## 功能说明

单算子描述文件，将单个算子描述文件（json格式）转换成适配昇腾AI处理器的离线模型，以便进行后续的单算子功能验证。



## 关联参数

使用该参数时，只有如下参数可以配合使用，其中[7.2.3.1 --output](#)、[7.2.4.1 --soc\\_version](#)为必填。

昇腾310 AI处理器可配合使用的参数：

- [7.2.3.1 --output](#)
- [7.2.4.1 --soc\\_version](#)
- [7.3.3.7 --auto\\_tune\\_mode](#)
- [7.3.3.1 --precision\\_mode](#)
- [7.3.3.4 --op\\_select\\_implmode](#)
- [7.3.3.5 --optypelist\\_for\\_implmode](#)
- [7.3.2.4 --enable\\_small\\_channel](#)
- [7.3.4.2 --log](#)
- [7.3.4.3 --debug\\_dir](#)
- [7.3.4.4 --op\\_compiler\\_cache\\_mode](#)
- [7.3.4.5 --op\\_compiler\\_cache\\_dir](#)

昇腾710 AI处理器可配合使用的参数：

- [7.2.3.1 --output](#)
- [7.2.4.1 --soc\\_version](#)
- [7.2.4.2 --core\\_type](#)
- [7.3.3.7 --auto\\_tune\\_mode](#)
- [7.3.3.1 --precision\\_mode](#)
- [7.3.3.4 --op\\_select\\_implmode](#)
- [7.3.3.5 --optypelist\\_for\\_implmode](#)
- [7.3.4.2 --log](#)
- [7.3.4.3 --debug\\_dir](#)
- [7.3.4.4 --op\\_compiler\\_cache\\_mode](#)
- [7.3.4.5 --op\\_compiler\\_cache\\_dir](#)

## 参数取值

**参数值：**单算子描述文件（json格式）格式以及参数配置请参见[6.2 单算子模型转换](#)。

**参数值约束：**该参数指定的单算子都是基于Ascend IR定义的，关于单算子的详细定义请参见《[算子清单（开放态）](#)》手册。

## 推荐配置及收益

无。



## 示例

下面以GEMM单算子为例进行说明，该单算子对应的描述文件为`gemm.json`，将该文件上传到ATC工具所在服务器任意目录，例如上传到`$HOME/singleop`，使用示例如下：

```
--singleop=$HOME/singleop/gemm.json --output=$HOME/singleop/out/op_model --  
soc_version=<soc_version>
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

单算子json文件转换成离线模型场景，如果希望模型转换时只使用TBE算子，还需设置如下环境变量：

```
export ASCEND_ENGINE_PATH=${INSTALL_DIR}/lib64/plugin/opskernel/libfe.so:${INSTALL_DIR}/lib64/  
plugin/opskernel/libge_local_engine.so
```

执行上述命令后：如果用户想要使用AICPU算子的能力或再次执行开源框架的网络模型转换成离线模型命令，如[开源框架的Caffe网络模型转换成离线模型](#)，转换之前，需要删除上述环境变量：执行`unset ASCEND_ENGINE_PATH`命令，使`ASCEND_ENGINE_PATH`环境变量失效。

## 7.2.3 输出选项

### 7.2.3.1 --output

#### 功能说明

- 如果是开源框架的网络模型：  
存放转换后的离线模型的路径以及文件名，例如：`$HOME/module/out/caffe_resnet50`，转换后的模型文件名以指定的为准，自动以`.om`后缀结尾，例如：`caffe_resnet50.om`。
- 如果是单算子描述文件（json格式）：  
存放转换后的单算子模型的路径，例如：`$HOME/singleop/out/op_model`。转换后的模型文件命名规则为：序号\_opType\_输入的描述(dataType\_format\_shape)\_输出的描述(dataType\_format\_shape)。

#### 关联参数

无。

#### 参数取值

参数值：

- 如果是开源框架的网络模型：存放转换后的离线模型的路径以及文件名。
- 如果是单算子描述文件（json格式）：存放转换后的单算子模型的路径。

**参数值格式：**路径和文件名：支持大小写字母（a-z, A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

## 推荐配置及收益

无。

## 示例

- Caffe框架网络模型：  
`--output=$HOME/module/out/caffe_resnet50`
- TF框架网络模型：  
`--output=$HOME/module/out/tf_resnet_v1_50`
- ONNX网络模型：  
`--output=$HOME/module/out/onnx_resnet50`
- 单算子描述文件：  
`--output=$HOME/singleop/out/op_model`

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.3.2 --output\_type

## 功能说明

指定网络输出数据类型或指定某个输出节点的输出类型。

## 关联参数

若指定某个输出节点的输出类型，则需要和[7.3.1.1 --out\\_nodes](#)参数配合使用。

## 参数取值

**参数值：**

- 指定网络输出类型
  - FP32：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。
  - FP16：推荐分类网络、检测网络使用。通常用于一个网络输出作为另一个网络输入场景。

模型转换完毕，在对应的\*.om模型文件中，上述数据类型分别以DT\_FLOAT、DT\_UINT8、DT\_FLOAT16值呈现。
- 指定某个输出节点的输出类型

例如: `--output_type="node1:0:FP16;node2:0:FP32"`, 表示node1节点第一个输出设置为FP16, node2第一个节点输出设置为FP32。指定的节点必须放在双引号中, 节点中间使用英文分号分隔。

该场景下, 该参数需要与`--out_nodes`参数配合使用。

#### 参数值约束:

若在模型转换时不指定网络具体输出数据类型, 则以原始网络模型最后一层输出的算子数据类型为准; 若指定了类型, 则以该参数指定的类型为准, 此时**7.3.1.6 --is\_output\_adjust\_hw\_layout**参数指定的类型不生效。

## 推荐配置及收益

无。

## 示例

- 指定网络输出类型  
`--output_type=FP32`
- 指定某个输出节点的输出类型  
`--model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50caffemodel --framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version> --output_type="conv1:0:FP16" --out_nodes="conv1:0"`

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.3.3 --check\_report

#### 功能说明

预检结果保存文件路径和文件名。

#### 关联参数

**7.2.1.2 --mode**: 当**7.2.1.2 --mode**=0时解析图失败或**7.2.1.2 --mode**=3仅做预检时, 用于生成check\_result.json预检结果文件, 如果不指定路径, 则将预检结果保存在当前路径下。

#### 参数取值

**参数值**: 预检结果保存文件路径和文件名。

**参数默认值**: check\_result.json

**参数值格式**: 路径和文件名: 支持大小写字母 (a-z, A-Z)、数字 (0-9)、下划线 ( \_ )、中划线 ( - )、句点 ( . )、中文字符。

## 推荐配置及收益

无。

## 示例

```
--check_report=$HOME/module/out/check_result.json
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.2.3.4 --json

## 功能说明

离线模型、原始模型文件、GE dump图结构文件转换为json格式文件的路径和文件名。

如果是已有的离线模型转换为json格式文件，在转换后的文件中还可以查看原始模型转换为该离线模型时，使用的基础版本号（比如ATC软件版本信息，OPP算子包版本信息等）以及当时模型转换使用的atc命令。

## 关联参数

- 离线模型转换为json  
该参数需要与[7.2.1.2 --mode=1](#)、[7.2.2.3 --om](#)参数配合使用。
- 原始模型文件转换为json  
该参数需要与[7.2.1.2 --mode=1](#)、[7.2.2.3 --om](#)参数、[7.2.2.4 --framework](#)配合使用。
- GE dump图结构文件转json  
该参数需要与[7.2.1.2 --mode=5](#)、[7.2.2.3 --om](#)参数配合使用。

## 参数取值

**参数值：** json格式文件的路径和文件名。

**参数值格式：** 路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

## 推荐配置及收益

无。

## 示例

- 离线模型转换为json：

```
--mode=1 --om=$HOME/module/out/caffe_resnet50.om --json=$HOME/module/out/caffe_resnet50.json
```

在转换后的json文件中，可以查看原始模型转换为该离线模型时，使用的基础版本号，以及当时转换使用的atc命令，示例如下：（如下示例中的版本号都为样例，请以实际查询的为准）

说明：离线模型转换为json文件可以查看基础版本号，必须保证使用的软件为1.77.22.6.220及之后的版本，之前版本无法查阅该信息；atc\_cmdline参数必须为1.78.23.34.230及之后的软件版本方可查阅。

```
{
  "key": "opp_version",
  "value": {
    "s": "1.78.23.34.230"
  }
},
...
{
  "key": "atc_version",
  "value": {
    "s": "1.78.23.34.230"
  }
},
...
{
  "key": "atc_cmdline",
  "value": {
    "s": "xxx/atc.bin --model ./resnet50.prototxt --weight ./resnet50.caffemodel --framework 0 --output ./out/resnet50 --soc_version Ascend310"
  }
},
...
{
  "key": "soc_version",
  "value": {
    "s": "Ascend310Ascend610SD3403"
  }
},
...
```

- 原始模型文件转换为json

```
--mode=1 --om=$HOME/module/resnet50.prototxt --json=$HOME/module/out/caffe_resnet50.json --framework=0
```

- GE dump图结构文件转json

```
--mode=5 --om=$HOME/module/ge_proto_00000000_PreRunBegin.txt --json=$HOME/module/out/ge_proto.json
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

## 7.2.4 目标芯片选项

### 7.2.4.1 --soc\_version

#### 功能说明

指定模型转换时昇腾AI处理器的版本。

#### 关联参数

无。

#### 参数取值

**参数值：** <soc\_version>

昇腾AI处理器的版本，可从ATC工具安装路径的“compiler/data/platform\_config”目录查看。

".ini"文件的文件名即为对应的<soc\_version>。如果用户根据上述方法仍旧无法确定具体使用的<soc\_version>，则：

1. 单击如下手册中的链接并进入该手册，《[Ascend-DMI工具用户指南](#)》。
2. 完成“使用工具>使用前准备”，然后进入“使用工具>设备实时状态查询”章节。
3. 使用相关命令查看芯片的详细信息，例如使用**ascend-dmi -i -dt**命令查看芯片的详细信息，返回信息中“Chip Name”对应取值即为具体使用的<soc\_version>。  
使用atc命令转换模型时，实际配置的<soc\_version>值，要去掉“Chip Name”对应取值中的空格，例如“Chip Name”对应取值为Ascend xxx yyy，实际配置的<soc\_version>值为Ascendxxxyyy。

**参数值约束：** 请使用与芯片名相对应的<soc\_version>取值进行模型转换，然后再进行推理。

#### 推荐配置及收益

无。

#### 示例

昇腾310 AI处理器使用示例：

```
--soc_version=Ascend310
```

昇腾710 AI处理器使用示例：

```
--soc_version=Ascend710
```

#### 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

#### 依赖约束

无。

### 7.2.4.2 --core\_type

#### 功能说明

设置网络模型使用的Core类型。

#### 关联参数

无。

#### 参数取值

参数值：

- VectorCore
- AiCore，默认为AiCore。

**参数值约束：**若网络模型中包括Cube算子，则只能使用AiCore。

#### 推荐配置及收益

无。

#### 示例

```
--core_type=VectorCore
```

### 支持的芯片型号

昇腾710 AI处理器

#### 依赖约束

无。

### 7.2.4.3 --aicore\_num

#### 功能说明

用于设置模型编译时使用的aicore数目。

#### 关联参数

无。

#### 参数取值（昇腾 710 AI 处理器）

**参数值：**aicore数目支持配置，取值范围为{1,2,4,8}

**参数默认值：**默认值即为最大值：8

#### 参数取值（昇腾 310 AI 处理器）（昇腾 910 AI 处理器）

**参数值：**（当前版本该参数预留，不建议配置；如需配置，则只能配置为默认值）

不同芯片版本该参数默认值不同，可从“\${INSTALL\_DIR}/x86\_64-linux/data/platform\_config/<soc\_version>.ini”文件中查看，该文件中如下参数的取值即为aicore\_num的默认取值：

```
[SoCInfo]
#aicore_num默认值，默认值即为最大值，实际模型转换时，xx请替换为具体取值
ai_core_cnt=xx
```

其中，\${INSTALL\_DIR}请替换为CANN软件安装后文件存储路径。例如，\$HOMR/Ascend。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：\$HOME/Ascend/ascend-toolkit/latest/x86\_64-linux。

**参数默认值：**默认值即为最大值

## 推荐配置及收益

无。

## 示例

昇腾310 AI处理器使用示例：

```
--aicore_num=2
```

昇腾710 AI处理器使用示例：

```
--aicore_num=8
```

## 支持的芯片型号

昇腾710 AI处理器

## 依赖约束

无。

# 7.3 高级功能参数

## 7.3.1 功能配置选项

### 7.3.1.1 --out\_nodes

#### 功能说明

指定网络模型中的输出节点（算子）或指定网络模型的输出。

如果不指定输出节点，则模型的输出默认为最后一层的算子信息，如果指定，则以指定的为准。

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。



## 关联参数

无。

## 参数取值

### 参数值：

- 网络模型中的节点（node\_name）名称  
指定的输出节点必须放在双引号中，节点中间使用英文分号分隔。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1个输出。
- 某层输出节点的topname（该场景仅支持Caffe网络模型）  
指定的输出节点必须放在双引号中，节点中间使用英文分号分隔。topname必须是模型编译前的Caffe网络模型中layer的某个top名称；若几个layer有相同的topname，则以最后一个layer的topname为准。
- 指定网络模型输出的名称（output的名称）（该场景仅支持ONNX网络模型）  
指定的output的名称必须放在双引号中，多个output的名称中间使用英文分号分隔。output必须是网络模型的输出。

### 参数值约束：

- 参数值中的三种方式，使用时只能取其一，不能同时存在。
- 若参数值取值为某层输出节点的topname，该种方式仅限于Caffe网络模型。
- 若参数值取值为网络模型输出的名称（output的名称），该种方式仅限于ONNX网络模型。
- 如果模型转换过程中该算子被融合掉，则该算子不能作为输出节点。

## 推荐配置及收益

无。

## 示例

- 参数值取网络模型中的节点（node\_name）名称  
`--out_nodes="node_name1:0;node_name1:1;node_name2:0"`
- 参数值取某层输出节点的topname  
`--out_nodes="topname0;topname1;topname2"`
- 参数值取网络模型输出的名称（output的名称）  
`--out_nodes="output1;output2;output3"`

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.1.2 --input\_fp16\_nodes

#### 功能说明

指定输入数据类型为FP16的输入节点名称。

#### 关联参数

若配置了该参数，则不能对同一个输入节点同时使用[7.3.1.3 --insert\\_op\\_conf](#)参数。

#### 参数取值

**参数值：**数据类型为FP16的输入节点名称。

**参数值约束：**指定的节点必须放在双引号中，节点中间使用英文分号分隔。

#### 推荐配置及收益

无。

#### 示例

```
--input_fp16_nodes="node_name1;node_name2"
```

#### 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

#### 依赖约束

无。

### 7.3.1.3 --insert\_op\_conf

#### 功能说明

插入算子的配置文件路径与文件名，例如aipp预处理算子。

#### 关联参数

- 若配置了该参数，则不能对同一个输入节点同时使用[7.3.1.2 --input\\_fp16\\_nodes](#)参数。
- 该参数不能与[7.2.2.10 --dynamic\\_dims](#)同时使用。

#### 参数取值

**参数值：**插入算子的配置文件路径与文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z, A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**文件后缀不局限于.cfg格式，但是配置文件中的内容需要满足prototxt格式。

## 推荐配置及收益

无。

## 示例

下面以插入AIPP预处理算子为例进行说明，配置文件内容示例如下（文件名为举例为：*insert\_op.cfg*），关于AIPP预处理配置文件的详细配置说明，请查看[6.1 AIPP使能](#)章节。

```
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  
}
```

将配置好的*insert\_op.cfg*文件上传到ATC工具所在服务器任意目录，例如上传到\$HOME/module，使用示例如下：

```
--insert_op_conf=$HOME/module/insert_op.cfg
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.1.4 --op\_name\_map

## 功能说明

扩展算子（非标准算子）映射配置文件路径和文件名，不同的网络中某扩展算子的功能不同，可以指定该扩展算子到具体网络中实际运行的扩展算子的映射。

## 关联参数

无。

## 参数取值

**参数值：**扩展算子映射配置文件路径和文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z, A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

## 推荐配置及收益

无。

## 示例

扩展算子映射配置文件内容示例如下（文件名举例为：*opname\_map.cfg*）：

```
OpA:Network1OpA
```

将配置好的*opname\_map.cfg*上传到ATC工具所在服务器任意目录，例如上传到*\$HOME/module*，使用示例如下：

```
--op_name_map=$HOME/module/opname_map.cfg
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.1.5 --is\_input\_adjust\_hw\_layout

#### 功能说明

用于指定网络输入数据类型是否为FP16，数据格式是否为NC1HWC0。

#### 关联参数

该参数需要与[7.3.1.2 --input\\_fp16\\_nodes](#)配合使用。

若[7.3.1.5 --is\\_input\\_adjust\\_hw\\_layout](#)参数设置为true，对应[7.3.1.2 --input\\_fp16\\_nodes](#)节点的输入数据类型为FP16，输入数据格式为NC1HWC0。

#### 参数取值

**参数值：**false或true

**参数默认值：**false

## 推荐配置及收益

无。

## 示例

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --  
framework=0 --output=$HOME/module/out/caffe_resnet50 --is_input_adjust_hw_layout=true --  
input_fp16_nodes="data" --soc_version=<soc_version>
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.1.6 --is\_output\_adjust\_hw\_layout

## 功能说明

用于指定网络输出的数据类型是否为FP16，数据格式是否为NC1HWC0。

## 关联参数

该参数需要与[7.3.1.1 --out\\_nodes](#)配合使用。

若[7.3.1.6 --is\\_output\\_adjust\\_hw\\_layout](#)参数设置为true，对应[7.3.1.1 --out\\_nodes](#)中输出节点的输出数据类型为FP16，数据格式为NC1HWC0。

## 参数取值

参数值：false或true

参数默认值：false

## 推荐配置及收益

无。

## 示例

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --  
framework=0 --output=$HOME/module/out/caffe_resnet50 --is_output_adjust_hw_layout=true --  
out_nodes="prob:0" --soc_version=<soc_version>
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

## 7.3.2 模型调优选项

### 7.3.2.1 --disable\_reuse\_memory

## 功能说明

内存复用开关。

## 关联参数

无。

## 参数取值

参数值：

- 1：关闭内存复用。
- 0：开启内存复用。

参数默认值：0

**参数值约束：**如果网络模型较大，关闭内存复用开关，模型转换时可能会造成内存不足，导致模型转换失败。

## 推荐配置及收益

无。

## 示例

```
--disable_reuse_memory=0
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

在内存复用场景下（默认开启内存复用），支持基于指定算子（节点名称/算子类型）单独分配内存。通过OP\_NO\_REUSE\_MEM环境变量指定要单独分配的一个或多个节点，支持混合配置。配置多个节点时，中间通过英文逗号（“,”）隔开。

- 基于节点名称配置  
`export OP_NO_REUSE_MEM=gradients/logits/semantic/kernel/Regularizer/l2_regularizer_grad/Mul_1,resnet_v1_50/conv1_1/BatchNorm/AssignMovingAvg2`
- 基于算子类型配置  
`export OP_NO_REUSE_MEM=FusedMulAddN,BatchNorm`
- 混合配置  
`export OP_NO_REUSE_MEM=FusedMulAddN,resnet_v1_50/conv1_1/BatchNorm/AssignMovingAvg`

### 7.3.2.2 --fusion\_switch\_file

## 功能说明

融合规则（包括图融合和UB融合）开关配置文件路径以及文件名，通过该参数关闭配置文件中指定的融合规则。

- 图融合：是FE根据融合规则进行改图的过程。图融合用融合后算子替换图中融合前算子，提升计算效率。图融合的场景如下：

- 在某一些算子的数学计算量可以进行优化的情况下，可以进行图融合，融合后可以节省计算时间。例如：conv+biasAdd，可以融合成一个算子，直接在l0c中完成累加，从而省去add的计算过程。
- 在融合后的计算过程可以通过硬件指令加速的情况下，可以进行图融合，融合后能够加速。例如：conv+biasAdd的累加过程，就是通过l0c中的累加功能进行加速的，可以通过图融合完成。
- UB融合：UB即昇腾AI处理器上的Unified Buffer，UB融合指A算子的计算结果在Unified Buffer上，需要搬移到Global Memory。B算子再执行时，需要将A算子的输出由Global Memory再搬移到Unified Buffer，进行B的计算逻辑，计算完之后，又从Unified Buffer搬移回Global Memory。  
从这个过程会发现A的结果从Unified Buffer->Global Memory->Unified Buffer->Global Memory。这个经过Global Memory进行数据搬移的过程是浪费的，因此将A和B算子合并成一个算子，省去了数据搬移的过程叫UB融合。UB融合可以减少整网中数据搬移的时间（Global Memory>Unified Buffer,Unified Buffer->Global Memory），提高运算效率，有效降低带宽。

## 关联参数

无。

## 参数取值

**参数值：**配置文件路径以及文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**

系统内置的图融合和UB融合规则，均为默认开启，用户可以根据需要通过该参数关闭指定的融合规则。当前可以关闭的融合规则请参见《[图融合和UB融合规则参考](#)》，由于系统机制，其他融合规则无法关闭。

## 推荐配置及收益

无。

## 示例

### • 场景1：逐条配置待关闭融合规则

配置文件样例如下，冒号前面为融合规则名，后面字段表示融合规则是否开启（融合规则开关配置文件名举例为 *fusion\_switch.cfg*）：

```
xxxFusionPass:off
yyyFusionPass:off
....
```

### • 场景2：一键式关闭融合规则

该参数支持用户一键式关闭融合规则，配置文件样例如下：

```
{
  "Switch":{
    "GraphFusion":{
      "ALL":"off"
    },
    "UBFusion":{
      "ALL":"off"
    }
  }
}
```

```
}  
}
```

说明：

- 关闭某些融合规则可能会导致功能问题，因此此处的一键式关闭仅关闭系统部分融合规则，而不是全部融合规则。
- 一键式关闭融合规则时，可以同时开启部分融合规则，样例如下：

```
{  
  "Switch":{  
    "GraphFusion":{  
      "ALL":"off",  
      "SoftmaxFusionPass":"on"  
    },  
    "UBFusion":{  
      "ALL":"off",  
      "TbePool2dQuantFusionPass":"on"  
    }  
  }  
}
```

将上述配置好的 *fusion\_switch.cfg* 文件上传到 ATC 工具所在服务器任意目录，例如上传到 *\$HOME/module*，使用示例如下：

```
--fusion_switch_file=$HOME/module/fusion_switch.cfg
```

模型转换完毕，在执行 atc 命令的当前目录，会生成 "fusion\_result.json" 文件，用于记录模型转换过程中除去 *fusion\_switch.cfg* 文件中关闭的融合规则外，仍旧使用的融合规则，其中，"match\_times" 字段表示模型转换过程中匹配到的融合规则次数，"effect\_times" 字段表示实际生效的次数。如果未使用 [7.3.2.2 --fusion\\_switch\\_file](#) 参数，则生成的 "fusion\_result.json" 文件中记录模型转换过程中匹配到的所有融合规则。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

- 若网络模型中 Convolution 算子的 "group" 属性取值 == 模型文件 prototxt 中 "num\_output" 属性的取值，则上述配置文件中 *VxxxRequantFusionPass* 必须打开。
- 昇腾模型压缩工具对原始框架模型进行量化时，会插入量化和反量化算子，而使用 ATC 工具进行模型转换过程中，会对插入的量化和反量化算子进行融合，此情况下再进行量化后模型 dump 结果与原始模型 dump 结果的比对可能不准确，因此如果用户想使用昇腾模型压缩工具量化后的模型进行精度比对，则需要通过 [7.3.2.2 --fusion\\_switch\\_file](#) 参数关闭部分融合功能，该场景下需要关闭的融合规则如下：

昇腾310 AI处理器、昇腾910 AI处理器场景必须关闭的融合规则：

```
V100RequantFusionPass:off  
ConvConcatFusionPass:off  
SplitConvConcatFusionPass:off  
TbeEltwiseQuantFusionPass:off  
TbeConvDequantVaddReluQuantFusionPass:off  
TbeConvDequantVaddReluFusionPass:off  
TbeConvDequantQuantFusionPass:off  
TbeDepthwiseConvDequantFusionPass:off  
TbeFullyconnectionElemwiseDequantFusionPass:off  
TbeConv2DAddMulQuantPass:off  
TbePool2dQuantFusionPass:off
```



```
TbeCommonRules0FusionPass:off  
TbeCommonRules2FusionPass:off
```

昇腾710 AI处理器必须关闭的融合规则：

```
V200RequantFusionPass:off  
ConvConcatFusionPass:off  
SplitConvConcatFusionPass:off  
TbeEltwiseQuantFusionPass:off  
TbeConvDequantVaddReluQuantFusionPass:off  
TbeConvDequantVaddReluFusionPass:off  
TbeConvDequantQuantFusionPass:off  
TbeDepthwiseConvDequantFusionPass:off  
TbeFullyconnectionElemwiseDequantFusionPass:off  
TbeConv2DAddMulQuantPass:off  
TbePool2dQuantFusionPass:off  
TbeCommonRules0FusionPass:off  
TbeCommonRules2FusionPass:off
```

融合规则解释如下：

- V100RequantFusionPass  
图融合规则。V100量化场景下，满足反量化(dequant)和量化(quant)相关pattern时，进行部署优化，提升推理性能。
- V200RequantFusionPass  
图融合规则。V200量化场景下，满足反量化(dequant)和量化(quant)相关pattern时，进行部署优化，提升推理性能。
- ConvConcatFusionPass  
图融合，支持conv2d\*N+concat算子的图融合规则，conv2d后面可以连接dequant和Relu类算子。
- SplitConvConcatFusionPass  
图融合，支持split+conv2d\*N+concat算子的融合规则，conv2d后面可以连接dequant和Relu类算子。
- TbeEltwiseQuantFusionPass  
UB融合，支持elemwise+quant算子的UB融合，quant算子为可选节点。
- TbeConvDequantVaddReluQuantFusionPass  
UB融合规则。量化场景下，对Conv-dequant-vadd-relu-quant连续的节点，标记UB融合，提升推理性能。
- TbeConvDequantVaddReluFusionPass  
UB融合，支持conv2d+dequant+vadd+relu/conv2d+dequant+(leakyrelu)+vadd算子的融合节点。
- TbeConvDequantQuantFusionPass  
UB融合规则。量化场景下，对Conv-dequant-quant连续的节点，标记UB融合，提升推理性能。
- TbeDepthwiseConvDequantFusionPass  
UB融合，支持depthwiseConv2d+dequant+(relu/mul)+quant/depthwiseConv2d+dequant+(sigmoid)+mul/depthwiseConv2d+requant/depthwiseConv2d+(power+relu6+power)+elemwise+(quant)算子的融合节点。
- TbeFullyconnectionElemwiseDequantFusionPass  
UB融合，支持如下两种形式的融合：
  - i. 固定shape场景BatchMatMul/BatchMatMulV2 + elemwise的融合。

- ii. 固定shape场景MatMul/MatMulV2/BatchMatMul/BatchMatMulV2 + AscendDequant + elemwise1(+ elemwise2)的融合。
- TbeConv2DAddMulQuantPass  
UB融合，支持conv+dequant+add+quant融合，add算子除quant外还必须有另两路任意输出才可以进行融合。
- TbePool2dQuantFusionPass  
UB融合规则。量化场景下，对Pool2d-quant连续的节点，标记UB融合，提升推理性能。
- TbeCommonRules0FusionPass  
UB融合，支持StridedRead+Conv2D+dequant+elemwise+quant+StridedWrite算子的UB融合，除Conv2D外，其他节点都是可选节点。
- TbeCommonRules2FusionPass  
UB融合，支持StridedRead+Conv2D+dequant+elemwise+quant+StridedWrite算子的UB融合，除Conv2D外，其他节点都是可选节点；elemwise支持多输出场景下的融合。

### 7.3.2.3 --enable\_scope\_fusion\_passes

#### 功能说明

指定编译时需要生效的融合规则列表。

无论是内置还是用户自定义的Scope融合规则，都分为如下两类：

- 通用融合规则（General）：各网络通用的Scope融合规则；默认生效，不支持用户指定失效。
- 定制化融合规则（Non-General）：特定网络适用的Scope融合规则；默认不生效，用户可以通过[7.3.2.3 --enable\\_scope\\_fusion\\_passes](#)指定生效的融合规则列表。

当前支持的融合规则请参见《[TensorFlow Parser Scope融合规则参考](#)》。

#### 关联参数

无。

#### 参数取值

**参数值：**注册的融合规则名称。

**参数值约束：**允许传入多个，中间使用英文逗号分隔，例如ScopePass1,ScopePass2,...。

#### 推荐配置及收益

无。

#### 示例

```
--enable_scope_fusion_passes=ScopePass1,ScopePass2
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

该参数只适用于TensorFlow网络模型。如果要查看模型转换过程中融合规则相关的日志信息，则[7.3.4.2 --log](#)至少要设置为warning级别。

### 7.3.2.4 --enable\_small\_channel

## 功能说明

是否使能small channel的优化，使能后在channel $\leq$ 4的卷积层会有性能收益。

建议用户在推理场景下打开此开关。

## 关联参数

该参数使能后，建议与[7.3.1.3 --insert\\_op\\_conf](#)参数（AIPP功能）配合使用，可以获得更优的性能。

## 参数取值

参数值：

- 0：关闭。
- 1：使能。

参数默认值：0

**参数值约束：**如果模型Input的channel $\leq$ 4，建议开启该参数，并配合AIPP（[7.3.1.3 --insert\\_op\\_conf](#)）使用，可获得更优的性能；如果开启之后出现性能下降，建议进行Tiling调优。

## 推荐配置及收益

无。

## 示例

```
--enable_small_channel=1
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

该参数使能后，建议与AIPP功能[7.3.1.3 --insert\\_op\\_conf](#)同时使用，否则可能没有收益。

### 7.3.2.5 --buffer\_optimize

#### 功能说明

数据缓存优化开关。

#### 关联参数

无。

#### 参数取值

参数值：

- l1\_optimize：表示开启l1优化。当前版本该参数无效，等同于off\_optimize。
- l2\_optimize：表示开启l2优化。
- off\_optimize：表示关闭数据缓存优化。

参数默认值：l2\_optimize

#### 推荐配置及收益

建议打开数据缓存优化功能：开启数据缓存优化可提高计算效率、提升性能，但由于部分算子在实现上可能存在未考虑的场景，导致影响精度，因此在出现精度问题时可以尝试关闭数据缓存优化。如果关闭数据缓存优化功能后，精度达标，则需要识别出问题算子，反馈给华为工程师进一步分析、解决算子问题，解决算子问题后，建议仍旧保持开启数据缓存优化功能。

#### 示例

昇腾310 AI处理器：

```
--buffer_optimize=l2_optimize
```

昇腾710 AI处理器：

```
--buffer_optimize=l2_optimize
```

#### 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

#### 依赖约束

无。

### 7.3.2.6 --mdl\_bank\_path

#### 功能说明

加载子图调优后自定义知识库的路径。

## 关联参数

无。

## 参数取值

**参数值：**子图调优后自定义知识库路径。

**参数值格式：**支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）。

**参数默认值：**`${HOME}/ascend/latest/data/aoe/custom/graph/<soc_version>`

## 推荐配置及收益

无。

## 示例

例如子图调优后自定义知识库的路径为`$HOME/custom_module_bank`，则使用示例为：

```
--mdl_bank_path=$HOME/custom_module_path
```

## 依赖约束

加载子图调优后自定义知识库路径优先级：[7.3.2.6 --mdl\\_bank\\_path](#)参数加载路径>`TUNE_BANK_PATH`环境变量设置路径>默认子图调优后自定义知识库路径。

1. 如果模型转换前，通过`TUNE_BANK_PATH`环境变量指定了子图调优自定义知识库路径，模型转换时又通过[7.3.2.6 --mdl\\_bank\\_path](#)参数加载了自定义知识库路径，该场景下以[7.3.2.6 --mdl\\_bank\\_path](#)参数加载的路径为准，`TUNE_BANK_PATH`环境变量设置的路径不生效。
2. [7.3.2.6 --mdl\\_bank\\_path](#)参数和环境变量指定路径都不生效或无可用自定义知识库，则使用默认自定义知识库路径。
3. 如果默认自定义知识库路径下也无可用知识库，则atc工具会查找`${INSTALL_DIR}/compiler/data/fusion_strategy/custom`路径下是否有可用知识库，其中，`${INSTALL_DIR}`请替换为CANN软件安装后文件存储路径。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：`$HOME/Ascend/ascend-toolkit/latest`。
4. 如果上述路径下都无可用的自定义知识库，则atc工具会查找子图调优内置知识库，该路径为：`${INSTALL_DIR}/compiler/data/fusion_strategy/built-in`

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 7.3.3 算子调优选项

### 7.3.3.1 --precision\_mode

#### 功能说明

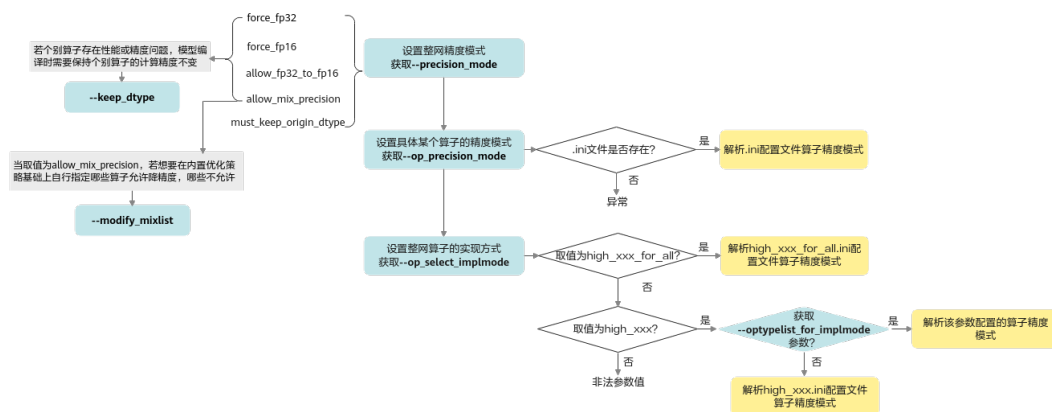
设置网络模型的精度模式。

#### 关联参数

- 当取值为`allow_mix_precision`时，如果用户想要在内置优化策略基础上进行调整，自行指定哪些算子允许降精度，哪些算子不允许降精度，则需要参见7.3.3.3 `--modify_mixlist`参数设置。
- 推理场景下，使用7.3.3.1 `--precision_mode`参数设置整个网络模型的精度模式，可能会有个别算子存在性能或精度问题，该场景下可以使用7.3.3.6 `--keep_dtype`参数，使原始网络模型编译时保持个别算子的计算精度不变，但7.3.3.1 `--precision_mode`参数取值为`must_keep_origin_dtype`时，7.3.3.6 `--keep_dtype`不生效。

关联参数示意图如图7-2所示。

图 7-2 关联参数示意图



设置具体算子精度模式场景下：

- 首先读取7.3.3.2 `--op_precision_mode`参数，校验该参数的ini配置文件是否存在，若存在则解析文件并读取算子的精度模式，否则上报异常。
- 7.3.3.2 `--op_precision_mode`不存在则读取7.3.3.4 `--op_select_implmode`参数：
  - 首先检测是否配置为`high_xxx_for_all`参数，若是则解析`high_xxx_for_all.ini`文件并读取算子的精度模式。
  - 若配置为`high_xxx`参数，则检测是否配置7.3.3.5 `--optypelist_for_implmode`参数，若是，则读取该参数配置的算子精度模式；否则解析`high_xxx.ini`文件并读取算子的精度模式。

#### 参数取值

##### 参数值：

- `force_fp32`：表示网络模型中算子支持float16和float32时，强制选择float32，若原图精度为float16，也会强制转为float32。

如果网络模型中存在部分算子，并且该算子实现不支持float32，比如Conv2D卷积算子仅支持float16类型，则该参数不生效，仍然使用支持的float16；如果该算子不支持float32，且又配置了黑名单（precision\_reduce = false），则会使用float32的aicpu算子。

- force\_fp16：表示网络模型中算子支持float16和float32时，强制选择float16。
- allow\_fp32\_to\_fp16：优先保持原图精度，如果网络模型中算子支持float32，则保留原始精度float32；如果网络模型中算子不支持float32，则直接降低精度到float16。
- must\_keep\_origin\_dtype：表示保持原图精度：如果网络模型中存在部分算子，并且该算子实现不支持float32，比如Conv2D卷积算子仅支持float16类型，则不能使用该参数。
- allow\_mix\_precision：表示使用混合精度模式（混合精度模式是通过混合使用float16和float32数据类型来处理神经网络的过程）：

针对网络模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到float16，从而在精度损失很小的情况下提升系统性能并减少内存使用。

若配置了该种模式，则可以在OPP软件包安装路径\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/config/<soc\_version>/aic-<soc\_version>-ops-info.json内置优化策略文件中查看“precision\_reduce”参数的取值：

- 若取值为true（白名单），则表示允许将当前float32类型的算子，降低精度到float16。
- 若取值为false（黑名单），则不允许将当前float32类型的算子降低精度到float16，相应算子仍旧使用float32精度。
- 若网络模型中算子没有配置该参数（灰名单），当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。

**参数默认值：**force\_fp16

## 推荐配置及收益

所配置的精度模式不同，网络模型精度以及性能有所不同，具体为：

精度高低排序：

force\_fp32>must\_keep\_origin\_dtype>allow\_fp32\_to\_fp16>allow\_mix\_precision>force\_fp16

性能优劣排序：

force\_fp16>=allow\_mix\_precision>allow\_fp32\_to\_fp16>must\_keep\_origin\_dtype>force\_fp32

## 示例

```
--precision_mode=force_fp16
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.3.2 --op\_precision\_mode

## 功能说明

设置具体某个算子的精度模式，通过该参数可以为多个算子设置不同的精度模式。

## 关联参数

该参数不能与[7.3.3.4 --op\\_select\\_implmode](#)、[7.3.3.5 --optypelist\\_for\\_implmode](#)参数同时使用，若三个参数同时配置，则只有[7.3.3.2 --op\\_precision\\_mode](#)参数指定的模式生效。

## 参数取值

**参数值：**设置算子精度模式配置文件（.ini格式）的路径以及文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z, A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**

- 当前仅支持通过.ini配置文件方式设置算子精度，配置文件中的内容以key\_value（算子类型=精度模式）形式呈现，每一行设置一个算子的精度模式。
- 算子类型必须为基于Ascend IR定义的算子的OpType，算子类型查看方法请参见[10.3 如何确定原始框架网络模型中的算子与昇腾AI处理器或BS9SX1A AI处理器支持的算子的对应关系](#)。

## 推荐配置及收益

该参数不建议配置，若使用高性能或者高精度模式，网络性能或者精度不是最优，则可以使用该参数，通过配置ini文件调整具体某个算子的精度模式。

## 示例

构造算子精度模式配置文件`op_precision.ini`，并在该文件中设置算子的精度模式，每一行设置一个算子的精度模式，样例如下：

```
optype1=high_precision  
optype2=high_performance
```

将配置好的`op_precision.ini`文件上传到ATC工具所在服务器任意目录，例如上传到`$HOME/conf`，使用示例如下：

```
--op_precision_mode=$HOME/conf/op_precision.ini
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器



## 依赖约束

无。

### 7.3.3.3 --modify\_mixlist

## 功能说明

混合精度场景下，修改算子使用混合精度名单，该文件包括黑名单，白名单和灰名单：

- 白名单：表示混合精度模式下，允许将当前float32类型的算子，降低精度到float16。
- 黑名单：表示混合精度模式下，不允许将当前float32类型的算子降低精度到float16，相应算子仍旧使用float32精度。
- 灰名单：表示混合精度模式下，当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。

## 关联参数

该参数需要与[7.3.3.1 --precision\\_mode](#)参数配合使用，并且该参数取值需要设置为“allow\_mix\_precision”，表示使用混合精度模式，该场景下网络模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到float16；而使用[7.3.3.3 --modify\\_mixlist](#)参数后，用户可以在内置优化策略基础上进行调整，自行指定哪些算子允许降精度，哪些算子不允许降精度。

## 参数取值

**参数值：**混合精度名单路径以及文件名。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**

- 名单格式为\*.json格式，文件中的算子列表由用户指定，多个算子使用英文逗号分隔。
- 配置的算子类型必须为基于Ascend IR定义的算子的OpType，算子类型查看方法请参见[10.3 如何确定原始框架网络模型中的算子与昇腾AI处理器或BS9SX1A AI处理器支持的算子的对应关系](#)。

## 推荐配置及收益

无。

## 示例

混合精度名单样例如下，文件名举例为ops\_info.json：

```
{
  "black-list": {
    "to-remove": [
      // 黑名单
      // 黑名单算子转换为灰名单算子，配置该参数时，请确保被转换的算子已经存在于黑名单中
    ]
  }
}
```

```
"Yolo"
],
"to-add": [           // 白名单或灰名单算子转换为黑名单算子
  "Matmul"
]
},
"white-list": {       // 白名单
  "to-remove": [      // 白名单算子转换为灰名单算子，配置该参数时，请确保被转换的算子已经存在于
白名单中
  "Conv2D"
],
"to-add": [           // 黑名单或灰名单算子转换为白名单算子
  "Bias"
]
}
}
```

- 假设算子A默认在白名单中，如果您希望将该算子配置为黑名单算子，则配置示例和系统处理逻辑为：

- a. 将该算子添加到黑名单中：

```
{
  "black-list": {
    "to-add": [A]
  }
}
```

则系统会将该算子从白名单中删除，并添加到黑名单中，最终该算子在黑名单中。

- b. 将该算子从白名单中删除，同时添加到黑名单中：

```
{
  "black-list": {
    "to-add": [A]
  }
  "white-list": {
    "to-remove": [A]
  }
}
```

则系统会将该算子从白名单中删除，并添加到黑名单中，最终该算子在黑名单中。

- 对于只从黑/白名单中删除，而不添加到白/黑名单的场景，系统会将该算子添加到灰名单中，配置示例如下（例如，从白名单删除某个算子）：

```
{
  "white-list": {
    "to-remove": [A]
  }
}
```

则系统会将该算子从白名单中删除，然后添加到灰名单中，最终该算子在灰名单中。

将配置好的`ops_info.json`文件上传到ATC工具所在服务器任意目录，例如上传到`$HOME/module`，使用示例如下：

```
--precision_mode=allow_mix_precision --modify_mixlist=$HOME/module/ops_info.json
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.3.4 --op\_select\_implmode

#### 功能说明

设置网络模型中算子是高精度实现模式还是高性能实现模式。

高精度是指在float16输入场景，通过泰勒展开/牛顿迭代等手段进一步提升算子的精度；高性能是指在float16输入的情况下，不影响网络精度前提的最优性能实现。

#### 关联参数

无。

#### 参数取值

参数值：

- **high\_precision**：表示网络模型中算子采用高精度实现模式。  
该参数采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/impl\_mode/high\_precision.ini。  
为保持兼容，该参数仅对high\_precision.ini文件中算子列表生效，通过该列表可以控制算子生效的范围并保证之前版本的网络模型不受影响。
- **high\_performance**：表示网络模型中算子采用高性能实现模式。  
该参数采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/impl\_mode/high\_performance.ini。  
为保持兼容，该参数仅对high\_performance.ini文件中算子列表生效，通过该列表可以控制算子生效的范围并保证之前版本的网络模型不受影响。
- **high\_precision\_for\_all**：表示网络模型中算子采用高精度实现模式。  
该参数采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/impl\_mode/high\_precision\_for\_all.ini，该文件中列表后续可能会跟随版本更新。  
**该实现模式不保证兼容**，如果后续新的软件包中有算子新增了实现模式（即配置文件中新增了某个算子的实现模式），之前版本使用high\_precision\_for\_all的网络模型，在新版本上性能可能会下降。
- **high\_performance\_for\_all**：表示网络模型中算子采用高性能实现模式。  
该参数采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/impl\_mode/high\_performance\_for\_all.ini，该文件中列表后续可能会跟随版本更新。  
**该实现模式不保证兼容**，如果后续新的软件包中有算子新增了实现模式（即配置文件中新增了某个算子的实现模式），之前版本使用high\_performance\_for\_all的网络模型，在新版本上精度可能会下降。

上述精度模式，根据算子的dtype进行区分。

参数默认值：high\_performance

## 推荐配置及收益

不建议用户使用**7.3.3.4 --op\_select\_implmode**参数设置算子的精度模式，该参数仅作为调测使用；推荐通过**7.3.3.2 --op\_precision\_mode**参数ini配置文件方式设置算子精度模式：

- 如果用户对性能有更高要求，则建议优先使用**high\_performance\_for\_all**参数，若经过验证性能满足要求，则建议用户复制一份high\_performance\_for\_all.ini文件，并且重命名为“网络模型.ini”文件，跟随网络使用，不同网络模型使用不同的ini文件，后续模型转换时，可以直接使用**7.3.3.2 --op\_precision\_mode**参数加载保存的“网络模型.ini”配置文件。
- 如果用户对精度有更高要求，则建议优先使用**high\_precision\_for\_all**参数，若经过验证精度满足要求，则建议用户复制一份high\_precision\_for\_all.ini文件，并且重命名为“网络模型.ini”文件，跟随网络使用，不同网络模型使用不同的ini文件，后续模型转换时，可以直接使用**7.3.3.2 --op\_precision\_mode**参数加载保存的“网络模型.ini”配置文件。
- 如果用户在使用**high\_performance\_for\_all**时，虽然性能得到很大的提升，但是发现精度不满足要求，发现是由于xxx算子使用了高性能模式引起的，则需要复制一份high\_performance\_for\_all.ini文件，重命名为“网络模型.ini”文件，并将文件中该xxx算子的实现模式调整为高精度模式，后续模型转换时，直接使用**7.3.3.2 --op\_precision\_mode**参数加载“网络模型.ini”配置文件。
- 如果用户在使用**high\_precision\_for\_all**时，虽然精度得到很大的提升，但是发现性能下降较厉害，发现是由于xxx算子使用了高精度模式引起的，则需要复制一份high\_precision\_for\_all.ini文件，重命名为“网络模型.ini”文件，并将文件中该xxx算子的实现模式调整为高性能模式，后续模型转换时，直接使用**7.3.3.2 --op\_precision\_mode**参数加载“网络模型.ini”配置文件。

high\_\*.ini文件中算子的实现模式以all\_ops\_impl\_mode.ini文件（路径为\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/impl\_mode）所列出的为准，不在该文件中的实现模式不支持配置。

上述路径中的\${INSTALL\_DIR}请替换为CANN软件安装后文件存储路径。例如，若安装的Ascend-cann-toolkit软件包，则安装后文件存储路径为：\$HOME/Ascend/ascend-toolkit/latest。

## 示例

```
--op_select_implmode=high_precision
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

- 如果有新支持精度模式的算子也选择高性能或者高精度模式，又不想破坏已有网络的精度或性能，则可以通过如下两种方式进行配置：
  - 通过**7.3.3.5 --optypelist\_for\_implmode**参数指定新增的具体算子  
`--op_select_implmode=high_precision --optypelist_for_implmode=算子optype`
  - 通过**7.3.3.2 --op\_precision\_mode**参数设置算子的精度模式  
构造算子精度模式配置文件op\_precision.ini，并在该文件中设置算子的精度模式，每一行设置一个算子的精度模式，样例如下：

```
optype1=high_precision  
optype2=high_performance
```

将配置好的 *op\_precision.ini* 文件上传到 ATC 工具所在服务器任意目录，例如上传到 *\$HOME/conf*，使用示例如下：

```
--op_precision_mode=$HOME/conf/op_precision.ini
```

- **7.3.3.4 --op\_select\_implmode** 参数表示设置网络模型中所有算子的高精度或高性能模式，如果算子实现了高精度和高性能，则运行时选择 **7.3.3.4 --op\_select\_implmode** 参数指定的模式；如果算子只实现了一种，则按照算子实现的方式运行，例如：

某个算子当前只支持高精度，而 **7.3.3.4 --op\_select\_implmode** 设置为高性能，则 **7.3.3.4 --op\_select\_implmode** 参数对于该算子不生效，使用该算子当前实现的高精度方式运行。

### 7.3.3.5 --optypelist\_for\_implmode

#### 功能说明

设置 optype 列表中算子的实现方式，该参数当前仅支持设置某个具体算子的实现方式，不支持设置多个算子。当前仅支持配置的算子为 Pooling、SoftmaxV2、LRN、ROIAlign。

#### 关联参数

- 该参数需要与 **7.3.3.4 --op\_select\_implmode** 参数配合使用，通过 **7.3.3.5 --optypelist\_for\_implmode** 参数设置的算子，统一使用 **7.3.3.4 --op\_select\_implmode** 参数指定的精度模式。
- 该参数配合 **7.3.3.4 --op\_select\_implmode** 参数使用时，不能与 **7.3.3.2 --op\_precision\_mode** 参数同时使用，若同时配置，则只有 **7.3.3.2 --op\_precision\_mode** 参数指定的模式生效。上述三个算子运行流程请参见 **7.3.3.1 --precision\_mode** 中的图 7-2。

#### 参数取值

**参数值：** 算子列表。

**参数值约束：**

- 该列表中的算子 OpType 必须为基于 Ascend IR 定义的算子的 OpType，算子类型查看方法请参见 **10.3 如何确定原始框架网络模型中的算子与昇腾 AI 处理器或 BS9SX1A AI 处理器支持的算子的对应关系**。
- 该列表中的算子使用 **7.3.3.4 --op\_select\_implmode** 参数指定的精度模式，且仅支持指定为 high\_precision、high\_performance 两种精度模式。
- 该参数仅对指定的算子生效，不指定的算子按照默认实现方式选择。

#### 推荐配置及收益

无。

#### 示例

```
--op_select_implmode=high_precision --optypelist_for_implmode=Pooling
```

上述配置示例表示仅对Pooling算子使用高精度模式，未指定算子使用算子的默认实现方式。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.3.6 --keep\_dtype

## 功能说明

保持原始网络模型编译时个别算子的计算精度不变。

推理场景下，使用[7.3.3.1 --precision\\_mode](#)参数设置整个网络模型的精度模式，可能会有个别算子存在性能或精度问题，为此引入[7.3.3.6 --keep\\_dtype](#)参数，保持原始网络模型编译时个别算子的计算精度不变，具体算子可以通过该参数指定的文件进行配置。

## 关联参数

该参数需要与[7.3.3.1 --precision\\_mode](#)参数配合使用，但当[7.3.3.1 --precision\\_mode](#)参数取值为must\_keep\_origin\_dtype时，该参数不生效。

## 参数取值

**参数值：**算子配置文件路径以及文件名，配置文件中列举需保持计算精度的算子名称或算子类型，每个算子单独一行。

**参数值约束：**若为算子类型，则以OpType::typeName格式进行配置，每个OpType单独一行，且算子OpType必须为基于Ascend IR定义的算子的OpType，算子类型查看方法请参见[10.3 如何确定原始框架网络模型中的算子与昇腾AI处理器或BS9SX1A AI处理器支持的算子的对应关系](#)。

**参数值格式：**路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、英文冒号(:)、中文字符。

## 推荐配置及收益

无。

## 示例

- 若配置文件中为算子名称，则配置样例如下（文件名举例为 *execeptionlist.cfg*）：

```
Opname1
Opname2
...
```
- 若配置文件中为算子类型，则配置样例为（文件名举例为 *execeptionlist.cfg*）：

```
OpType::TypeName1  
OpType::TypeName2  
...
```

将配置好的 *execeptionlist.cfg* 文件上传到 ATC 工具所在服务器任意目录，例如上传到 \$HOME，使用示例如下：

```
--keep_dtype=$HOME/execeptionlist.cfg --precision_mode=force_fp16
```

模型编译时，*execeptionlist.cfg* 文件中的算子，保持原始网络模型精度，即精度不会改变，其余网络模型中的算子以 [7.3.3.1 --precision\\_mode](#) 参数指定的精度模式进行编译。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无

### 7.3.3.7 --auto\_tune\_mode

## 功能说明

设置算子的自动调优模式：控制 TBE 算子编译时，是否对算子进行调优，以便在昇腾 AI 处理器上寻找最好的性能配置。

关于 Auto Tune 工具的原理，支持调优的算子以及详细使用请参见《[CANN 开发工具指南（开放态）](#)》中的“Auto Tune 工具使用指南”章节。

## 关联参数

该参数不能与 [7.3.3.8 --op\\_bank\\_path](#) 参数同时使用。

## 参数取值

参数值：

- GA ( Genetic Algorithm )：遗传算法，用于设置 Cube 算子的调优性能。
- RL ( Reinforcement Learning )：强化学习，用于设置 Vector 算子的调优性能。

**参数值格式：**支持配置多种模式，多种模式放在双引号中，中间用英文逗号分隔，例如 "RL,GA"。

## 推荐配置及收益

使用 ATC 工具进行模型转换时，如果使能该参数，则模型转换时间会比不使能长（通常一个算子大约需要 20 分钟左右，具体时间和网络模型大小相关），但使用转换后的离线模型进行推理，性能会优于不调优的离线模型。

## 示例

```
--auto_tune_mode="RL,GA"
```



- GA调优：
  - 内置知识库以及cost model路径：  
`${INSTALL_DIR}/opp/data/tiling/<soc_version>/built-in/`：负责存储常用shape调优后的知识库和cost model，如果网络模型中某个shape没有命中内置知识库，则会发起调优。
  - 自定义知识库默认路径：  
`${HOME}/ascend/latest/data/aoe/custom/op/<soc_version>/cube`：用于存放调优过程中生成的知识库。如果该路径已经有知识库，则对该知识库进行追加；如果当前路径没有知识库，则新建知识库。该路径安装完CANN软件包后默认不存在，会在首次执行调优时自动创建。  
该路径为默认路径，用户也可以通过设置“TUNE\_BANK\_PATH”环境变量，[指定调优后自定义知识库的存储路径](#)。
- RL调优：
  - 内置知识库路径：  
`${INSTALL_DIR}/opp/data/rl/<soc_version>/built-in/`：负责存储常用shape调优后的知识库。
  - 自定义知识库默认路径：  
`${HOME}/ascend/latest/data/aoe/custom/op/<soc_version>/vector`：用于存放调优过程中调优性能优于built-in或built-in中没有的知识库。该路径安装完CANN软件包后默认不存在，会在首次执行调优时自动创建。  
该路径为默认路径，用户也可以通过设置“TUNE\_BANK\_PATH”环境变量，[指定调优后自定义知识库的存储路径](#)。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

- 环境变量设置：

使用该功能时，只支持带NPU设备的安装场景，详细介绍请参见《[CANN 软件安装指南](#)》>常见部署场景推荐。

CANN软件包安装完成后，请以CANN软件包运行用户，执行如下命令使环境变量生效：

```
source ${INSTALL_DIR}/bin/setenv.bash
```
- 指定调优后自定义知识库的存储路径：

可以通过设置如下环境变量实现：

```
export TUNE_BANK_PATH=/home/username/custom_tune_bank
```

若配置此环境变量，则：

  - GA调优后的策略存储在配置路径的<soc\_version>/ga目录下，调优脚本会自动在配置的TUNE\_BANK\_PATH路径下创建<soc\_version>/ga两级目录。  
例如TUNE\_BANK\_PATH=/home/username/custom\_tune\_bank，则GA调优后的最优策略存入/home/username/custom\_tune\_bank/<soc\_version>/ga目录中。
  - RL调优后的策略存储在配置路径的<soc\_version>/rl目录下，调优脚本会自动在配置的TUNE\_BANK\_PATH路径下创建<soc\_version>/rl两级目录。



例如TUNE\_BANK\_PATH=/home/username/custom\_tune\_bank, 则RL调优后的最优策略存入/home/username/custom\_tune\_bank/<soc\_version>/rl目录中。

路径支持绝对路径或相对于执行Auto Tune工具所在路径的相对路径, 配置的路径需要为已存在的目录, 且执行用户具有读、写、可执行权限。

### 说明

若调优时指定了自定义知识库路径, 后续进行模型转换时, 若想直接使用指定的自定义知识库, 也需要配置上此环境变量。

例如指定调优后自定义知识库存储路径为/home/username/custom\_tune\_bank/<soc\_version>/ga, 则使用ATC工具进行模型转换前需要配置如下环境变量 (该场景下用户无需再次开启Auto Tune功能):

```
export TUNE_BANK_PATH=/home/username/custom_tune_bank
```

- 强制调优:

如果custom路径已经有用户调优过的知识库, 由于某种原因导致算子逻辑变更, 例如GEMM算子新增支持的ND输入, 该情况下需要设置如下环境变量后, 重新发起调优。

```
export REPEAT_TUNE=True
```

- 指定层调优:

使用ATC工具进行Auto Tune在线调优时 (基于DUMP数据离线调优不支持该场景), 可以通过设置如下环境变量, 对指定层进行调优。

```
export TUNE_OPS_NAME=conv_layers/Pad_1
```

某些场景下, 例如在对网络进行Profiling性能分析后, 发现某个算子性能较低, 则可以通过设置上述环境变量, 对性能较低的算子单独进行调优。

OPS\_NAME配置为网络模型中需要调优的节点的name, 可配置为原始网络模型中的OP Name或者配置为经过GE/FE处理过的适配昇腾AI处理器的网络模型中的节点的OP Name (此OP Name可从Profiling调优数据中获取, 详细可参见《[CANN 开发工具指南 \(开放态\)](#)》中的“Profiling工具使用指南”章节)。若指定多个节点, 则使用英文逗号分隔。

### 说明

如果指定算子已经调优过, 即在custom路径已经有调优过的知识库, 用户想重新发起该算子的调优, 则需要同时设置如下两个环境变量:

```
export REPEAT_TUNE=True
export TUNE_OPS_NAME=conv_layers/Pad_1
```

- 开启Auto Tune时, 由于上板评估需要申请内存, 内存占用相比ATC工具要大, 额外的内存占用大小与同时使用的device设备数有关, 可近似估算为: 2\*device数量\*输入数据大小。如果内存超过ATC运行内存大小, 则会导致调优失败。
- 使用Auto Tune工具进行GA模式调优时, 需要独占Device资源, 不可以进行使用Device资源的其他操作, 请将其他进程停止后, 重新进行调优, 否则将会出现未知异常 (例如 run.o.failed)。如果Host有编译业务执行, 则会影响调优时长。
- 当前在同一个开发环境上允许开启多个ATC进程同时调优, 多进程并行在一定范围内会提升调优效率, 但由于资源的限制, 达到一定峰值后调优效率会随着进程数的增多下降, ATC进程数配置建议满足如下规则:
  - ATC进程数 \* TE\_PARALLEL\_COMPILER \* 2 < Host侧CPU核数。其中TE\_PARALLEL\_COMPILER代表算子并行编译进程数。
  - TBE算子并行编译场景下 (即TE\_PARALLEL\_COMPILER>1), 建议一个ATC进程对应一个Device。

### 7.3.3.8 --op\_bank\_path

#### 功能说明

加载Auto Tune算子调优后自定义知识库的路径。

#### 关联参数

该参数不能与[7.3.3.7 --auto\\_tune\\_mode](#)参数同时使用。

#### 参数取值

**参数值：**Auto Tune算子调优后自定义知识库路径。

**参数值格式：**支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）。

**参数默认值：**

- GA调优后默认自定义知识库路径：\${HOME}/ascend/latest/data/aoe/custom/op/<soc\_version>/cube
- RL调优后默认自定义知识库路径：\${HOME}/ascend/latest/data/aoe/custom/op/<soc\_version>/vector

#### 推荐配置及收益

无。

#### 示例

例如Auto Tune算子调优后自定义知识库的路径分别为：

- GA调优后：\$HOME/custom\_tune\_bank/op/<soc\_version>/cube
- RL调优后：\$HOME/custom\_tune\_bank/op/<soc\_version>/vector

则使用示例为：

- 单独加载GA调优后知识库：**  
`--op_bank_path=$HOME/custom_tune_bank/op/<soc_version>/cube`
- 单独加载RL调优后知识库：**  
`--op_bank_path=$HOME/custom_tune_bank/op/<soc_version>/vector`
- 同时加载GA和RL调优后知识库：**  
`--op_bank_path=$HOME/custom_tune_bank/op`

#### 使用约束

加载算子调优后自定义知识库路径优先级：**TUNE\_BANK\_PATH**环境变量设置路径>[7.3.3.8 --op\\_bank\\_path](#)参数加载路径>默认算子调优后自定义知识库路径。

- 如果模型转换前，通过**TUNE\_BANK\_PATH**环境变量指定了算子调优自定义知识库路径，模型转换时又通过[7.3.3.8 --op\\_bank\\_path](#)参数加载了自定义知识库路径，该场景下以**TUNE\_BANK\_PATH**环境变量设置的路径为准，[7.3.3.8 --op\\_bank\\_path](#)参数加载的路径不生效。
- [7.3.3.8 --op\\_bank\\_path](#)参数和环境变量指定路径都不生效前提下，使用默认自定义知识库路径。

- 如果上述路径下都无可用的自定义知识库，则atc工具会查找算子调优内置知识库。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

### 7.3.3.9 --op\_debug\_level

#### 功能说明

TBE算子编译debug功能开关。

#### 关联参数

无。

#### 参数取值

**参数值：**

- 0：不开启算子debug功能，在执行atc命令当前路径**不生成**算子编译目录kernel\_meta。
- 1：开启算子debug功能，在执行atc命令当前路径算子编译生成的kernel\_meta文件夹中生成TBE指令映射文件（算子cce文件\*.cce和python-cce映射文件\*\_loc.json），用于后续工具进行AICore Error问题定位。
- 2：开启算子debug功能，在执行atc命令当前路径算子编译生成的kernel\_meta文件夹中生成TBE指令映射文件（算子cce文件\*.cce和python-cce映射文件\*\_loc.json），并关闭编译优化开关并且开启ccec调试功能（ccec编译器选项设置为-O0-g），用于后续工具进行AICore Error问题定位。
- 3：不开启算子debug功能，在执行atc命令当前路径算子编译生成的kernel\_meta文件夹中**保留**.o（算子二进制文件）和.json文件（算子描述文件）。
- 4：不开启算子debug功能，在执行atc命令当前路径算子编译生成的kernel\_meta文件夹中**保留**.o（算子二进制文件）和.json文件（算子描述文件），并生成TBE指令映射文件（算子cce文件\*.cce）和UB融合计算描述文件（{\$kernel\_name}\_compute.json）。

**参数默认值：**0

**参数值约束：**进行模型转换时，建议配置为0、3或4。如果需要定位AICore Error问题，则需要将参数值设置为1或2。设置为1或2后，由于加入了调试功能，会导致网络性能下降。

#### 推荐配置及收益

无。

#### 示例

```
--op_debug_level=1
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 使用约束

该参数优先级高于算子编译接口（TBE DSL的build接口或者TBE TIK的BuildCCE接口）中的**tbe\_debug\_level**的值。

## 7.3.4 调试选项

### 7.3.4.1 --dump\_mode

#### 功能说明

是否生成带shape信息的json文件。

#### 关联参数

该参数需要与[7.2.3.4 --json](#)、[7.2.1.2 --mode=1](#)、[7.2.2.4 --framework](#)、[7.2.2.3 --om](#)参数（需要为原始模型文件，如果为Caffe框架模型文件，还需要增加[7.2.2.2 --weight](#)参数）配合使用。

#### 参数取值

参数值：

- 0：不使能。
- 1：使能。

参数默认值：0

#### 推荐配置及收益

无。

#### 示例

```
atc --mode=1 --om=$HOME/module/resnet50.prototxt --json=$HOME/module/out/resnet50.json --framework=0 --weight=$HOME/module/resnet50.caffemodel --dump_mode=1
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

7.3.4.2 --log

功能说明

设置ATC模型转换过程中日志的级别。

关联参数

无。

参数取值

- 参数值：
- debug：输出debug/info/warning/error/event级别的运行信息。
  - info：输出info/warning/error/event级别的运行信息。
  - warning：输出warning/error/event级别的运行信息。
  - error：输出/error/event级别的运行信息。
  - null：不输出日志。默认为null。
- 参数默认值： null

推荐配置及收益

无。

示例

```
--log=debug
```

如果模型转换失败，则可以通过分析日志定位问题。日志格式如下，更多日志信息请参见《[日志参考（开放态）](#)》。

```
[Level] ModuleName(PID,PName):DateTimeMS [FileName:LineNumber]LogContent
```

各字段解释如下：

表 7-2 日志字段说明

字段	说明
Level	日志级别。运行日志存在5种日志级别：ERROR、WARNING、INFO、DEBUG、EVENT。
ModuleName	产生日志的模块的名称。
PID	进程ID。
PName	进程名称。
DateTimeMS	日志打印时间，格式为：yyyy-mm-dd-hh:mm:ss.SSS。
FileName:LineNumber	调用日志打印接口的文件及对应的行号。

字段	说明
LogContent	各模块具体的日志内容。

样例如下：

```
[INFO] FE(30741,atc.bin):2021-12-09-16:10:22.539.141 [fe_type_utils.cc:52]30741 GetRealPath:"path /usr/local/Ascend/opp/op_impl/built-in/ai_core/tbe/config/ascendxxx is not exist."  
[WARNING] FE(30741,atc.bin):2021-12-09-16:10:22.539.146 [sub_op_info_store.cc:52]30741 Initialize:"The config file[/usr/local/Ascend/opp/op_impl/built-in/ai_core/tbe/config/ascendxxx] of op information library[tbe-builtin] is not existed. "  
[ERROR] GE(30741,atc.bin):2021-12-09-16:10:22.539.201 [error_manager.cc:263]30741 ReportErrMsgage:[INIT][OPS_KER][Report][Error]error_code: W21000, arg path is not existed in map
```

问题定位思路：

表 7-3 问题定位思路

字段	说明	解决思路
GE	GE图编译或校验问题。	校验类报错，通常会给出明确的错误原因，此时需要针对性的修改模型转换使用的参数，以满足相关要求。
FE	算子融合问题。	无。
TEFUSION	<ul style="list-style-type: none"><li>算子预编译/编译问题。</li><li>融合算子编译问题。</li></ul>	常见错误信息以及解决思路： 1. ModuleNotFoundError: No module named 'decorator' 解决思路：根据提示信息安装pip包。 2. ModuleNotFoundError: No module named 'te' 解决思路：安装ATC工具所在软件包时，安装命令没有使用--pylocal，建议使用该参数重新安装相应软件包。
TBE	算子编译问题。	无。

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

- 日志落盘：  
atc命令执行过程中，日志默认落盘到\$HOME/ascend/log/plog/plog-pid\_\*.log（pid代表进程ID，“\*”表示该日志文件创建时的时间戳）路径，由于7.3.4.2 --log默认值为null，既不输出日志，若该路径存在日志信息，则为atc进程之外的其他日志信息，比如依赖Python相关信息。

若想要日志体现atc进程相关信息，则模型转换时，需要设置7.3.4.2 --log参数（不能设置为null）。

- 日志打屏：

atc命令执行过程中，日志默认不打屏，如需打屏显示，则请在执行atc命令的当前窗口设置如下环境变量，然后再执行atc命令：

```
export ASCEND_SLOG_PRINT_TO_STDOUT=1
```

关于日志的更多信息请参见《[日志参考（开放态）](#)》。若设置上述环境变量后，仍旧未打屏有效信息，则请在atc命令设置7.3.4.2 --log参数（不能设置为null）显示相应的日志级别。

- 日志重定向

如果不想日志落盘，而是重定向到文件，则模型转换前需要设置上述的日志打屏环境变量，并且atc命令需要设置7.3.4.2 --log参数（不能设置为null），样例如下：

```
atc xxx --log=debug >log.txt
```

### 7.3.4.3 --debug\_dir

#### 功能说明

用于配置保存模型转换、网络迁移过程中算子编译生成的调试相关过程文件的路径。

过程文件包括但不限于算子.o（算子二进制文件）、.json（算子描述文件）、.cce等文件，具体生成的文件以7.3.3.9 --op\_debug\_level参数设置的取值为准。

#### 关联参数

该参数需要与7.3.3.9 --op\_debug\_level参数配合使用，且7.3.3.9 --op\_debug\_level参数取值不能为0。

#### 参数取值

**参数值：**存放模型转换过程中生成的调试相关文件的路径。

**参数值格式：**路径：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**如果使用该参数，则在执行atc命令之前，请先创建该参数要指定的目录。

**参数默认值：**在执行atc命令的当前路径./kernel\_meta文件夹中生成算子编译的过程文件。

#### 推荐配置及收益

无。

#### 示例

例如创建的目录名为debug\_info，则执行命令为：

```
--debug_dir=$HOME/module/out/debug_info
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.4.4 --op\_compiler\_cache\_mode

## 功能说明

用于配置算子编译磁盘缓存模式。

## 关联参数

- 该参数需要与[7.3.4.5 --op\\_compiler\\_cache\\_dir](#)配合使用。
- 如果模型转换命令同时使用了[7.3.3.9 --op\\_debug\\_level](#)参数，则只有[7.3.3.9 --op\\_debug\\_level](#)参数配置为0、3或4才会启用编译缓存功能，其他取值禁用编译缓存功能。

## 参数取值

参数值：

- enable：表示启用算子编译缓存。启用后可以避免针对相同编译参数及算子参数的算子重复编译，从而提升编译速度。
- disable：表示禁用算子编译缓存。
- force：表示强制刷新缓存，即先删除已有缓存，再重新编译并加入缓存。当用户的python或者依赖库等发生变化时，需要指定为force用于清理已有的缓存。

参数默认值：disable

参数值约束：

- 由于force选项会先删除已有缓存，所以不建议在程序并行编译时设置，否则可能会导致其他模型使用的缓存内容被清除而导致失败。
- 建议模型最终发布时设置编译缓存选项为disable或者force。

## 推荐配置及收益

- 推荐配置为enable：启用后可以避免针对相同编译参数及算子参数的算子重复编译，从而提升编译速度。
- 首次使用带有该参数的atc命令，用于生成缓存，再次使用带有该参数的atc命令，会利用上次生成的缓存进行编译，从而提升编译速度。

## 示例

```
--op_compiler_cache_mode=enable --op_compiler_cache_dir=$HOME/atc_data/kernel_cache --  
op_debug_level=3
```



## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

### 使用约束：

使用该参数时，可以通过设置如下环境变量，限制某个芯片下缓存文件夹的磁盘空间的大小。如下环境变量可选，默认为500，单位为MB。

```
export MAX_OP_CACHE_SIZE=xx
```

另外，当编译缓存空间大小达到MAX\_OP\_CACHE\_SIZE设置的取值，而需要删除旧的kernel文件时，可以通过如下环境变量，设置需要保留缓存的空间大小比例。如下环境变量可选，默认为50，单位为百分比。

```
export REMAIN_CACHE_SIZE_RATIO=xx%
```

### 7.3.4.5 --op\_compiler\_cache\_dir

## 功能说明

用于配置算子编译磁盘缓存的目录。

## 关联参数

该参数需要与[7.3.4.4 --op\\_compiler\\_cache\\_mode](#)参数配合使用。

## 参数取值

**参数值：**存放算子编译磁盘缓存的路径。

**参数值格式：**路径支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（\_）、中划线（-）、句点（.）、中文字符。

**参数值约束：**如果[7.3.4.5 --op\\_compiler\\_cache\\_dir](#)参数指定的路径存在且有效，则在指定的路径下自动创建子目录kernel\_cache；如果指定的路径不存在但路径有效，则先自动创建目录，然后在该路径下自动创建子目录kernel\_cache。

**参数默认值：**\$HOME/atc\_data

## 推荐配置及收益

无。

## 示例

```
--op_compiler_cache_dir=$HOME/atc_data --op_compiler_cache_mode=enable
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

### 7.3.4.6 --display\_model\_info

## 功能说明

编译原始框架网络模型时，查询模型占用的关键资源信息、编译与运行环境等信息，查询出的信息直接在屏幕打印显示。

## 关联参数

无。

## 参数取值

参数值：

- 0：关闭查询功能。
- 1：打开查询功能。

**参数值约束：**该参数不支持单算子描述文件转离线模型时信息的查看，即不支持与[7.2.2.11 --singleop](#)参数同时使用。

**参数默认值：**0

## 推荐配置及收益

无。

## 示例

下面示例以Caffe框架网络模型为例进行说明：

```
atc --model=$HOME/module/resnet50.prototxt --weight=$HOME/module/resnet50.caffemodel --  
framework=0 --output=$HOME/module/out/caffe_resnet50 --soc_version=<soc_version> --  
display_model_info=1
```

命令执行完毕，屏幕会打印类似如下信息：

```
===== Display Model Info start =====  
# 模型转换使用的atc命令  
Original Atc command line: ${INSTALL_DIR}/bin/atc.bin --model=$HOME/module/resnet50.prototxt --  
weight=$HOME/module/resnet50.caffemodel --framework=0 --output=$HOME/module/out/caffe_resnet50  
--soc_version=<soc_version> --display_model_info=1  
# ATC软件版本信息、soc_version版本信息、原始框架信息  
system info: atc_version[xxx], soc_version[xxx], framework_type[xxx].  
# 运行时的占用内存、逻辑stream数目、event数目  
resource info: memory_size[xxx B], weight_size[xxx B], stream_num[xxx], event_num[xxx].  
# 离线模型文件中各分区大小、包括ModelDef、权重、tbekernels、taskinfo占用的大小等  
om info: modeldef_size[xxx B], weight_data_size[xxx B], tbe_kernels_size[xxx B],  
cust_aicpu_kernel_store_size[xxx B], task_info_size[xxx B].  
===== Display Model Info end =====
```

## 支持的芯片型号

昇腾310 AI处理器

昇腾710 AI处理器

## 依赖约束

无。

# 8 定制网络专题

[定制网络修改 \(Caffe\)](#)

[定制网络修改 \(TensorFlow\)](#)

## 8.1 定制网络修改 (Caffe)

### 8.1.1 简介

本章节修改只适用于Caffe网络模型。

网络的算子可以分为如下几类：

- 标准算子：昇腾AI处理器支持的Caffe标准算子，比如Convolution等。
- 扩展算子：昇腾AI处理器支持的公开但非Caffe标准算子，分为 2 种：
  - 一种是基于Caffe框架进行自定义扩展的算子，比如Faster RCNN中的ROI Pooling、SSD中的归一化算子Normalize等。
  - 另外一种是来源于其他深度学习框架的自定义算子，比如YOLOv2中Passthrough等。

Faster RCNN、SSD等网络模型都包含了一些原始Caffe框架中没有定义的算子结构，如ROI Pooling、Normalize、PSROI Pooling和Upsample等。为了使昇腾AI处理器能支持这些网络，需要对原始的Caffe框架网络模型进行扩展，降低开发者开发自定义算子/开发后处理代码的工作量。若开发者的Caffe框架网络模型中使用了这些扩展算子，在进行模型转换时，需要先在prototxt中修改/添加扩展层的定义，才能成功进行模型转换。

本章节提供了昇腾AI处理器的扩展算子列表，并给出了如何根据扩展算子修改prototxt文件方法。

## 8.1.2 扩展算子列表

表 8-1 扩展算子列表

分类	算子类型	说明
过程算子	<b>Reverse</b>	将输入tensor指定的轴进行反转。
	<b>ROI Pooling</b>	用于Faster R-CNN中，对Roi（Region of interest）进行池化操作，主要用于目标检测任务。
	<b>PSROI Pooling</b>	用于R-FCN中，进行位置敏感的候选区域池化操作，主要用于目标检测任务。
	<b>Upsample</b>	使用pooling mask的上采样。用于yolo网络。
	<b>Normallize</b>	将SSD网络中Channel维度中的元素在L2进行归一化。
	<b>Reorg</b>	Reorg layer in Darknet，将通道数据转移到平面上，或反过来操作。 <b>对应算子规格中的PassThrough算子。</b>
	<b>Proposal</b>	用于Faster R-CNN，根据rpn_cls_prob的foreground，rpn_bbox_pred中的bounding box regression修正anchors获得精确的proposals。
	<b>ROI Align</b>	从feature map中获取ROI（range of interest）的特征矩阵。
	<b>ShuffleChannel</b>	把channel维度分为[group, channel/Group]，再在channel维度中进行数据转置[channel/Group,group]。
	<b>Yolo</b> （Yolo/Detection/Region）	Yolo/Detection/Region算子，都需要替换为Yolo算子，对卷积网络输出的feature map生成检测框的坐标信息，置信度信息及类别概率。
	<b>PriorBox</b>	用于SSD网络，根据输入参数，生成prior box。
	<b>SpatialTransformer</b>	用于仿射变换。

分类	算子类型	说明
后处理算子	YoloV3DetectionOutput	此算子用于YOLOv3的后处理过程，对卷积网络输出的feature map生成检测框的坐标信息，置信度信息及类别概率。
	YoloV2DetectionOutput	此算子用于YOLOv2的后处理过程，对卷积网络输出的feature map生成检测框的坐标信息，置信度信息及类别概率。
	SSDDetectionOutput	此算子用于SSD网络的后处理过程，用于整合预选框、预选框偏移以及得分三项结果，最终输出满足条件的目标检测框、目标的label和得分。
	FSRDetectionOutput	此算子用于Faster R-CNN网络的后处理过程，对结果进行分类，并对每个类输出最终的bbox数量、坐标、类别概率及类别索引。

### 8.1.3 扩展算子规则

扩展算子的实现可以在Caffe、TensorFlow、MindSpore深度学习框架中实现，扩展算子中的第一种是基于Caffe框架进行扩展实现的，如ROI Pooling、Normalize、PSROI Pooling和Upsample层等，因此有公开的prototxt标准定义。而扩展算子中的第二种并不是在Caffe框架下实现的，对于这类网络中的自定义算子，也需要给出prototxt的标准定义，用于在prototxt中定义网络中相应的算子。

#### Reverse

在指定的轴上反序，如输入为[1,2,3]，其反序为[3,2,1]。

算子定义如下：

1. 在LayerParameter中添加ReverseParameter

```
message LayerParameter {  
  ...  
  optional ReverseParameter reverse_param = 157;  
  ...  
}
```

2. 定义ReverseParameter类型以及属性参数

```
message ReverseParameter{  
  repeated int32 axis = 1;  
}
```

## ROI Pooling

在目标检测算法中，region proposal产生的ROI大小不一，而分类网络的FC层要固定的输入，所以ROI Pooling起到一个连接作用。

ROI Pooling层为Faster RCNN网络中用于将不同图像经过卷积层后得到的feature map进行维度统一的操作，输入为一个feature map以及需要从中提取的ROI框坐标值，输出为一个维度归一化的feature map。

ROI Pooling层需要对caffe.proto文件进行扩展，定义ROI PoolingParameter，扩展方式如下所示，定义的参数包括：

- spatial\_scale，即输入feature map与原始图片的尺寸比例，用于转换ROI的坐标，因为ROI坐标是在原图尺寸上的。
- pooled\_h、pooled\_w，输出feature map在spatial维度上h和w的大小。

1. 在LayerParameter中添加ROI PoolingParameter

```
message LayerParameter {  
    ...  
    optional ROI PoolingParameter roi_pooling_param = 161;  
    ...  
}
```

2. 定义ROI PoolingParameter类型以及属性参数

```
message ROI PoolingParameter {  
    required int32 pooled_h = 1;  
    required int32 pooled_w = 2;  
    optional float spatial_scale = 3 [default=0.0625];  
    optional float spatial_scale_h = 4;  
    optional float spatial_scale_w = 5;  
}
```

基于上述原则，ROI Pooling层在prototxt中定义的代码样例如下：

```
layer {  
    name: "roi_pooling"  
    type: "ROI Pooling"  
    bottom: "res4f"  
    bottom: "rois"  
    bottom: "actual_rois_num"  
    top: "roi_pool"  
    roi_pooling_param {  
        pooled_h: 14  
        pooled_w: 14  
        spatial_scale: 0.0625  
        spatial_scale_h: 0.0625  
        spatial_scale_w: 0.0625  
    }  
}
```

## PSROI Pooling

PSROI Pooling层的操作与ROI Pooling层类似，不同之处在于不同空间维度输出的图片特征来自不同的feature map channels，且对每个小区域进行的是Average Pooling，不同于ROI Pooling的Max Pooling。

对于一个输出  $k \times k$  的结果，不同空间维度的特征取自输入feature map中不同的组，即将输入的feature map均匀分为 $k \times k$ 组，每组的channel数与输出的channel一致，得到上述输出。

PSROI Pooling层需要对caffe.proto文件进行扩展，定义PSROI PoolingParameter，扩展方式如下所示，定义的参数包括：

- spatial\_scale, 即输入feature map与原始图片的尺寸比例。
- output\_dim, 输出feature map的channel。
- group\_size, 输出的spatial维度, 即上述的k。

1. 在LayerParameter中添加PSROIPoolingParameter

```
message LayerParameter {  
  ...  
  optional PSROIPoolingParameter psroi_pooling_param = 207;  
  ...  
}
```

2. 定义PSROIPoolingParameter类型以及属性参数

```
message PSROIPoolingParameter {  
  required float spatial_scale = 1;  
  required int32 output_dim = 2; // output channel number  
  required int32 group_size = 3; // number of groups to encode position-sensitive score maps  
}
```

基于上述原则, PSROIPooling层在prototxt中定义的代码样例如下:

```
layer {  
  name: "psroipooling"  
  type: "PSROIPooling"  
  bottom: "some_input"  
  bottom: "some_input"  
  top: "some_output"  
  psroi_pooling_param {  
    spatial_scale: 0.0625  
    output_dim: 21  
    group_size: 7  
  }  
}
```

## Upsample

Upsample层为Pooling层的逆操作, 其中每个Upsample层均与网络之前一个对应大小输入、输出Pooling层一一对应, 完成feature map在spatial维度上的扩充。

Upsample层需要对caffe.proto文件进行扩展, 定义UpsampleParameter, 扩展方式示例如下所示。定义的参数包括stride, 即输出与输入的尺寸比例, 如2。

1. 在LayerParameter中添加UpsampleParameter

```
message LayerParameter {  
  ...  
  optional UpsampleParameter upsample_param = 160;  
  ...  
}
```

2. 定义UpsampleParameter类型以及属性参数

```
message UpsampleParameter {  
  optional float scale = 1 [default = 1];  
  optional int32 stride = 2 [default = 2];  
  optional int32 stride_h = 3 [default = 2];  
  optional int32 stride_w = 4 [default = 2];  
}
```

基于上述原则, Upsample在prototxt中定义的代码样例如下:

```
layer {  
  name: "layer86-upsample"  
  type: "Upsample"  
  bottom: "some_input"  
  top: "some_output"  
  upsample_param {  
    scale: 1  
    stride: 2  
  }  
}
```



```
}  
}
```

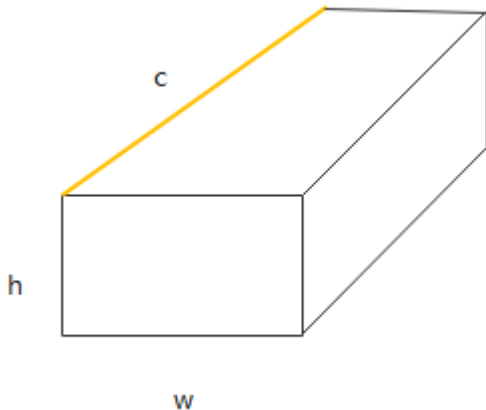
## Normalize

Normalize层为SSD网络中的一个归一化层，主要作用是将空间或者通道内的元素归一化到0到1之间，其进行的操作为对于一个 $c \times h \times w$ 的三维tensor，输出是同样大小的tensor，其中计算为每个元素以channel方向的平方和的平方根求 normalize，其具体计算公式为：

$$x'_i = \frac{x_i}{\sqrt{\sum_{j=1}^c x_j^2}}$$

其中分母位置的平方和的累加向量为同一 $h$ 与 $w$ 位置的所有 $c$ 方向的向量，如[图8-1](#)中的橙色区域。

图 8-1 Normalize 层原理图



经过上述计算归一化后，再对每个feature map做scale，每个channel对应一个scale值。

Normalize层需要对caffe.proto文件进行扩展，定义NormalizeParameter，扩展方式如下所示。定义的参数包括：

- across\_spatial参数表示是否对整个图片进行归一化，归一化的维度为： $1 \times c \times h \times w$ ，否则对每个像素点进行归一化： $1 \times c \times 1 \times 1$ 。
- channels\_shared表示scale是否相同，如果为true，则scale都是一样的，否则对于channel一样，对不同channel像素点是不一样的，默认为True。
- eps防止normalize过程中除以0的一个很小数值，默认为 $1e-10$ ，可配置。

normalize的计算公式转换为：

$$x_i = \frac{x_i}{\left(\sum_1^n x_i^2 + \text{eps}\right)^{\frac{1}{2}}} \times \text{scale}_i$$

算子定义如下：

## 1. 在LayerParameter中添加NormalizeParameter

```
message LayerParameter {  
  ...  
  optional NormalizeParameter norm_param = 206;  
  ...  
}
```

## 2. 定义NormalizeParameter类型以及属性参数

```
message NormalizeParameter {  
  optional bool across_spatial = 1 [default = true];  
  // Initial value of scale. Default is 1.0 for all  
  optional FillerParameter scale_filler = 2;  
  // Whether or not scale parameters are shared across channels.  
  optional bool channel_shared = 3 [default = true];  
  // Epsilon for not dividing by zero while normalizing variance  
  optional float eps = 4 [default = 1e-10];  
}
```

基于上述原则，Normalize在prototxt中定义的代码样例如下：

```
layer {  
  name: "normalize_layer"  
  type: "Normalize"  
  bottom: ""some_input"  
  top: "some_output"  
  norm_param {  
    across_spatial: false  
    scale_filler {  
      type: "constant"  
      value: 20  
    }  
    channel_shared: false  
  }  
}
```

## Reorg

Reorg算子在昇腾AI处理器内部以PassThrough算子呈现，将通道数据转移到平面上，或反过来操作。

PassThrough层为Yolo v2中的一个自定义层，由于Yolo v2并不是使用Caffe框架实现，因此对于该层没有标准的定义。该层实现的功能为将feature map在spatial维度上的数据展开到channel维度上，原始在channel维度上连续的元素在展开后的feature map中依然是连续的。

算子定义如下：

## 1. 在LayerParameter中添加ReorgParameter

```
message LayerParameter {  
  ...  
  optional ReorgParameter reorg_param = 155;  
  ...  
}
```

## 2. 定义ReorgParameter类型以及属性参数

```
message ReorgParameter {  
  optional uint32 stride = 2 [default = 2];  
  optional bool reverse = 1 [default = false];  
}
```

基于上述原则，Reorg在prototxt中定义的代码样例如下：

```
layer {  
  bottom: "some_input"  
  top: "some_output"  
  name: "reorg"
```

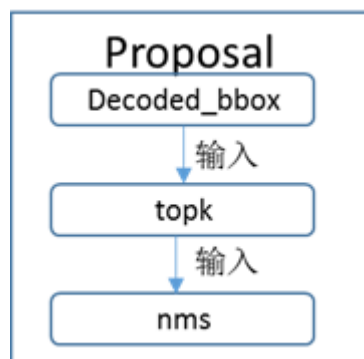
```
type: "Reorg"
reorg_param {
  stride: 2
}
```

## Proposal

proposal算子根据rpn\_cls\_prob的foreground, rpn\_bbox\_pred中的bounding box regression修正anchors获得精确的proposals。

具体可以分为3个算子decoded\_bbox、topk和nms，实现如图8-2所示。

图 8-2 proposal 算子实现



算子定义如下：

1. 在LayerParameter中添加ProposalParameter

```
message LayerParameter {
  ...
  optional ProposalParameter proposal_param = 201;
  ...
}
```

2. 定义ProposalParameter类型以及属性参数

```
message ProposalParameter {
  optional float feat_stride = 1 [default = 16];
  optional float base_size = 2 [default = 16];
  optional float min_size = 3 [default = 16];
  repeated float ratio = 4;
  repeated float scale = 5;
  optional int32 pre_nms_topn = 6 [default = 3000];
  optional int32 post_nms_topn = 7 [default = 304];
  optional float iou_threshold = 8 [default = 0.7];
  optional bool output_actual_rois_num = 9 [default = false];
}
```

基于上述原则，Proposal在prototxt中定义的代码样例如下：

```
layer {
  name: "faster_rcnn_proposal"
  type: "Proposal"           // 算子Type

  bottom: "rpn_cls_prob_reshape"
  bottom: "rpn_bbox_pred"
  bottom: "im_info"
  top: "rois"
  top: "actual_rois_num"     // 增加的算子输出
  proposal_param {
```

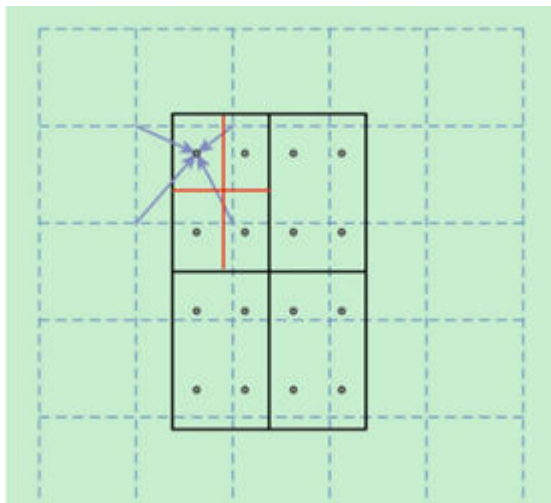
```
feat_stride: 16
base_size: 16
min_size: 16
pre_nms_topn: 3000
post_nms_topn: 304
iou_threshold: 0.7
output_actual_rois_num: true
}
```

## ROIAlign

ROI Align 是在Mask-RCNN论文里提出的一种区域特征聚集方式, 与ROI Pooling算法进行改进: 用双线性插值替换ROI Pooling操作中两次量化, 以解决ROI Pooling造成的区域不匹配的问题, 提高检测准确性。

ROIAlign算子是从feature map中获取ROI ( range of interest ), 分为pooled\_h x pooled\_w 个单元格, 每个单元格均分为sampling\_ratio\*sampling\_ratio个小方格, 每个小方格的中心点就是采样点。如图8-3所示, 虚线部分表示feature map, 实线表示ROI, 这里将ROI切分成2x2的单元格。如果采样点数是4, 则首先将每个单元格子均分成四个小方格 ( 如红色线所示 ), 每个小方格中心就是采样点。由于采样点的坐标通常是浮点数, 所以需要对该像素点进行双线性插值 ( 如图8-3中的四个箭头所示 ), 就可以得到该像素点的值了。然后对每个单元格内的四个采样点取均值, 就可以得到最终的ROIAlign的结果。

图 8-3 ROIAlign 原理图



算子定义如下:

1. 在LayerParameter中添加ROIAlignParameter

```
message LayerParameter {
...
    optional ROIAlignParameter roi_align_param = 154;
...
}
```

2. 定义ROIAlignParameter类型以及属性参数

```
message ROIAlignParameter {
    // Pad, kernel size, and stride are all given as a single value for equal
    // dimensions in height and width or as Y, X pairs.
    optional uint32 pooled_h = 1 [default = 0]; // The pooled output height
    optional uint32 pooled_w = 2 [default = 0]; // The pooled output width
    // Multiplicative spatial scale factor to translate ROI coords from their
```

```
// input scale to the scale used when pooling
optional float spatial_scale = 3 [default = 1];
optional int32 sampling_ratio = 4 [default = -1];
optional int32 roi_end_mode = 5 [default = 0];
}
```

根据上述类型以及属性用户可以自定义prototxt。

## ShuffleChannel

ShuffleChannel是把channel维度分为[group, channel/Group]，然后再在channel维度中进行数据转置[channel/Group,group]。

例如对于channel=4，group=2，则执行此算子后，就是把channel[1]、channel[2]的数据交换位置。

算子定义如下：

1. 在LayerParameter中添加ShuffleChannelParameter

```
message LayerParameter {
...
  optional ShuffleChannelParameter shuffle_channel_param = 159;
...
}
```

2. 定义ShuffleChannelParameter类型以及属性参数

```
message ShuffleChannelParameter{
  optional uint32 group = 1[default = 1]; // The number of group
}
```

基于上述原则，ShuffleChannel在prototxt中定义的代码样例如下：

```
layer {
  name: "layer_shuffle"
  type: "ShuffleChannel"
  bottom: "some_input"
  top: "some_output"
  shuffle_channel_param {
    group: 3
  }
}
```

## Yolo

YOLO算子出现在YOLO V2网络，且目前仅在YOLO V2、V3网络中使用，对数据做sigmoid和softmax操作。

- 在YOLO V2中，根据background和softmax的参数，有4种场景：
  - a. background=false，softmax=true，  
对(x,y,h,w)中的(x,y)做sigmoid，对b做sigmoid，对classes做softmax。
  - b. background=false，softmax=false，  
对(x,y,h,w)中的(x,y)做sigmoid，对b做sigmoid，对classes做sigmoid。
  - c. background=true，softmax=false，  
对(x,y,h,w)中的(x,y)做sigmoid，对b不做计算，对classes做sigmoid。
  - d. background=true，softmax= true，  
对(x,y,h,w)中的(x,y)做sigmoid，对b和classes放在一起做softmax。
- 在YOLO V3中，只有一种场景：对(x,y,h,w)中的(x,y)做sigmoid，对b做sigmoid，对classes做sigmoid。

输入数据格式为Tensor(n, coords+backgroup+classes,l,h,l.w)，其中n是anchor box的数量，coords表示x,y,w,h。

算子定义如下：

1. 在LayerParameter中添加YoloParameter

```
message LayerParameter {  
    ...  
    optional YoloParameter yolo_param = 199;  
    ...  
}
```

2. 定义YoloParameter类型以及属性参数

```
message YoloParameter {  
    optional int32 boxes = 1 [default = 3];  
    optional int32 coords = 2 [default = 4];  
    optional int32 classes = 3 [default = 80];  
    optional string yolo_version = 4 [default = "V3"];  
    optional bool softmax = 5 [default = false];  
    optional bool background = 6 [default = false];  
    optional bool softmaxtree = 7 [default = false];  
}
```

基于上述原则，Yolo在prototxt中定义的代码样例如下：

```
layer {  
    bottom: "layer82-conv"  
    top: "yolo1_coords"  
    top: "yolo1_obj"  
    top: "yolo1_classes"  
    name: "yolo1"  
    type: "Yolo"  
    yolo_param {  
        boxes: 3  
        coords: 4  
        classes: 80  
        yolo_version: "V3"  
        softmax: true  
        background: false  
    }  
}
```

## PriorBox

根据输入的参数，生成prior box。

下面以conv7\_2\_mbox\_priorbox为例，根据对应的参数生成prior box的数量。定义如下：

```
layer{  
    name:"conv7_2_mbox_priorbox"  
    type:"PriorBox"  
    bottom:"conv7_2"  
    bottom:"data"  
    top:"conv7_2_mbox_priorbox"  
    prior_box_param{  
        min_size:162.0  
        max_size:213.0  
        aspect_ratio:2  
        aspect_ratio:3  
        flip:true  
        clip:false  
        variance:0.1  
        variance:0.1  
        variance:0.2  
        variance:0.2  
        img_size:300  
        step:64
```

```
    offset:0.5  
  }  
}
```

1. 宽高都为minsize生成prior box。
2. 如果存在max\_size, 则用 $\sqrt{\text{min\_size} \times \text{max\_size}}$ 确定宽高生成一个框(约束,  $\text{max\_size} > \text{min\_size}$ )。
3. 根据aspect\_ratio(如定义所示aspect\_ratio为2, 3, flip是true, 自动添加aspect\_ratio=1/2、1/3), 生成对应的prior box。

因此,  $\text{num\_priors}(\text{prior box的数量}) = \text{min\_size的数量} + \text{aspect\_ratio的数量} \text{ (这里为4)} \times \text{min\_size的数量} \text{ (这里为1)} + \text{max\_size的数量} \text{ (max\_size的数量和min\_size的数量一一对应)}$ 。

算子定义如下:

1. 在LayerParameter中添加PriorBoxParameter

```
message LayerParameter {  
  ...  
  optional PriorBoxParameter prior_box_param = 203;  
  ...  
}
```

2. 定义PriorBoxParameter类型以及属性参数

```
message PriorBoxParameter {  
  // Encode/decode type.  
  enum CodeType {  
    CORNER = 1;  
    CENTER_SIZE = 2;  
    CORNER_SIZE = 3;  
  }  
  // Minimum box size (in pixels). Required!  
  repeated float min_size = 1;  
  // Maximum box size (in pixels). Required!  
  repeated float max_size = 2;  
  // Various of aspect ratios. Duplicate ratios will be ignored.  
  // If none is provided, we use default ratio 1.  
  repeated float aspect_ratio = 3;  
  // If true, will flip each aspect ratio.  
  // For example, if there is aspect ratio "r",  
  // we will generate aspect ratio "1.0/r" as well.  
  optional bool flip = 4 [default = true];  
  // If true, will clip the prior so that it is within [0, 1]  
  optional bool clip = 5 [default = false];  
  // Variance for adjusting the prior bboxes.  
  repeated float variance = 6;  
  // By default, we calculate img_height, img_width, step_x, step_y based on  
  // bottom[0] (feat) and bottom[1] (img). Unless these values are explicitly  
  // provided.  
  // Explicitly provide the img_size.  
  optional uint32 img_size = 7;  
  // Either img_size or img_h/img_w should be specified; not both.  
  optional uint32 img_h = 8;  
  optional uint32 img_w = 9;  
  
  // Explicitly provide the step size.  
  optional float step = 10;  
  // Either step or step_h/step_w should be specified; not both.  
  optional float step_h = 11;  
  optional float step_w = 12;  
  
  // Offset to the top left corner of each cell.  
  optional float offset = 13 [default = 0.5];  
}
```

基于上述原则, PriorBox在prototxt中定义的代码样例如下:

```
layer {
  name: "layer_priorbox"
  type: "PriorBox"
  bottom: "some_input"
  bottom: "some_input"
  top: "some_output"
  prior_box_param {
    min_size: 30.0
    max_size: 60.0
    aspect_ratio: 2
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    step: 8
    offset: 0.5
  }
}
```

## SpatialTransformer

该算子计算过程其实做了一个仿射变换：仿射变换的参数，可以是在prototxt固定的，多个batch使用一份；也可以是层的第二个输入，每个batch使用不一样的参数。

计算步骤：

1. 对于输出坐标，通过如下公式将其变换成[-1,1]区间中的值。

$$x' = x \times \frac{1.0}{out_h} \times 2 - 1$$
$$y' = y \times \frac{1.0}{out_w} \times 2 - 1$$

计算代码如下：

```
Dtype* data = output_grid.mutable_cpu_data();
for(int i=0; i< output_H_ * output_W_; ++i) {
  data[3 * i] = (i / output_W_) * 1.0 / output_H_ * 2 - 1;
  data[3 * i + 1] = (i % output_W_) * 1.0 / output_W_ * 2 - 1;
  data[3 * i + 2] = 1;
}
```

2. 通过仿射变换，转换为输入的坐标，其中s为输入坐标，t为输出坐标，计算公式如下：

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

计算代码如下：

```
caffe_cpu_gemm<Dtype>(CblasNoTrans, CblasTrans, output_H_ * output_W_, 2, 3, (Dtype)1.,
  output_grid_data, full_theta_data + 6 * i, (Dtype)0., coordinates);
```

3. 通过输入坐标取对应位置的值，赋值给输出的对应位置。

因为在第一步对输出坐标做了一次转换，所以对应的输入坐标，使用同样的方式再缩放回去，代码样例如下：

```
Dtype x = (px + 1) / 2 * H;
Dtype y = (py + 1) / 2 * W;
if(debug) std::cout<<prefix<<"(x, y) = ("<<x<< ", "<<y<<")"<<std::endl;
```



```
for(int m = floor(x); m <= ceil(x); ++m)
for(int n = floor(y); n <= ceil(y); ++n) {
    if(debug) std::cout<<prefix<<"(m, n) = ("<<m<< ", "<<n<<)"<<std::endl;
    if(m >= 0 && m < H && n >= 0 && n < W) {
        res += (1 - abs(x - m)) * (1 - abs(y - n)) * pic[m * W + n];
        if(debug) std::cout<<prefix<<" pic[m * W + n]= "<<std::endl;
    }
}
```

算子定义如下：

1. 在LayerParameter中添加SpatialTransformParameter

```
message LayerParameter {
...
    optional SpatialTransformParameter spatial_transform_param = 153;
...
}
```

2. 定义SpatialTransformParameter类型以及属性参数

```
message SpatialTransformParameter {
    optional uint32 output_h = 1 [default = 0];
    optional uint32 output_w = 2 [default = 0];
    optional float border_value = 3 [default = 0];
    repeated float affine_transform = 4;
    enum Engine {
        DEFAULT = 0;
        CAFFE = 1;
        CUDNN = 2;
    }
    optional Engine engine = 15 [default = DEFAULT];
}
```

基于上述原则，SpatialTransform层在prototxt中定义的代码样例如下：

```
layer {
    name: "st_1"
    type: "SpatialTransformer"
    bottom: "data"
    bottom: "theta"
    top: "transformed"
    st_param {
        to_compute_dU: false
        theta_1_1: -0.129
        theta_1_2: 0.626
        theta_2_1: 0.344
        theta_2_2: 0.157
    }
}
```

## 8.1.4 样例参考

本章节介绍几种常用网络的修改方法。

### 8.1.4.1 FasterRCNN 网络模型 prototxt 修改

#### 说明

本章节所有的代码样例都不能直接复制到网络模型中使用，需要用户根据使用的网络模型，自行调整相应参数，比如bottom、top中的参数要和具体网络模型中的bottom、top一一对应，并且bottom和top对应的参数顺序不能更改。

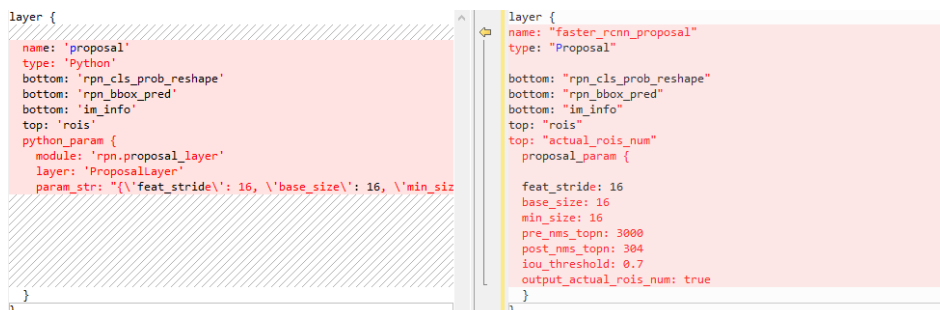
如下以FasterRCNN Resnet34网络模型为例进行说明。

1. proposal算子修改。

根据《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，该算子有3个输入，2个输出；根据上述原则，对原始proposal算子进行修改，

type修改为caffe.proto文件中的类型，增加actual\_rois\_num输出节点。参见caffe.proto文件中的属性定义增加相应属性信息。修改情况如图8-4所示，其中左边为原始算子prototxt，右边是适配昇腾AI处理器的prototxt。

图 8-4 原始算子与修改后的算子比对 1



代码样例如下：

```
layer {
name: "faster_rcnn_proposal"
type: "Proposal"           // 算子Type

bottom: "rpn_cls_prob_reshape"
bottom: "rpn_bbox_pred"
bottom: "im_info"
top: "rois"
top: "actual_rois_num"     // 增加的算子输出
proposal_param {

  feat_stride: 16
  base_size: 16
  min_size: 16
  pre_nms_topn: 3000
  post_nms_topn: 304
  iou_threshold: 0.7
  output_actual_rois_num: true
}
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

2. 最后一层增加FSRDetectionOutput算子，用于输出最终的检测结果。

对于FasterRCNN网络，参考[8.1.2 扩展算子列表](#)在原始prototxt文件的最后增加后处理算子层FSRDetectionOutput，参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，FSRDetectionOutput算子有五个输入，两个输出，此算子的Type及属性定义如下：

代码样例如下：

```
layer {
name: "FSRDetectionOutput_1"
type: "FSRDetectionOutput"

bottom: "rois"
bottom: "bbox_pred"
bottom: "cls_prob"
bottom: "im_info"
bottom: "actual_rois_num"
top: "actual_bbox_num1"
top: "box1"

fsrdetectionoutput_param {
```

```
num_classes:3
score_threshold:0.0
iou_threshold:0.7
batch_rois:1
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

### 8.1.4.2 YOLOv3 网络模型 prototxt 修改

#### 说明

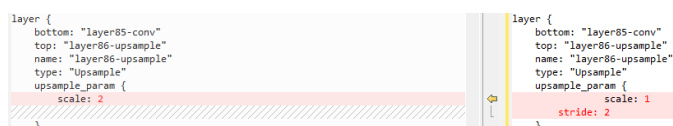
本章节所有的代码样例都不能直接复制到网络模型中使用，需要用户根据使用的网络模型，自行调整相应参数，比如bottom、top中的参数要和具体网络模型中的bottom、top——对应，并且bottom和top对应的参数顺序不能更改。

1. upsample算子upsample\_param属性参数修改。

参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，需要将原始算子prototxt中的scale:2修改为scale:1 stride:2。

对比情况如[图8-5](#)所示，其中左边为原始算子prototxt，右边是适配昇腾AI处理器的prototxt。

图 8-5 原始算子与修改后的算子比对 4



```
layer {
  bottom: "layer85-conv"
  top: "layer86-upsample"
  name: "layer86-upsample"
  type: "Upsample"
  upsample_param {
    scale: 2
  }
}

layer {
  bottom: "layer85-conv"
  top: "layer86-upsample"
  name: "layer86-upsample"
  type: "Upsample"
  upsample_param {
    scale: 1
    stride: 2
  }
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

2. 新增三个Yolo算子。

由于Yolo算子+DetectionOutput算子才是特征检测网络完整的后处理逻辑，根据原始算子prototxt所示，在增加YoloV3DetectionOutput算子之前需要先增加三个Yolo算子。

根据《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，该算子有1个输入，3个输出，根据上述原则，构造的Yolo算子代码样例如下：

- 算子1代码样例：

```
layer {
  bottom: "layer82-conv"
  top: "yolo1_coords"
  top: "yolo1_obj"
  top: "yolo1_classes"
  name: "yolo1"
  type: "Yolo"
  yolo_param {
    boxes: 3
    coords: 4
    classes: 80
    yolo_version: "V3"
    softmax: true
    background: false
  }
}
```

- 算子2代码样例：

```
layer {
  bottom: "layer94-conv"
```

```
top: "yolo2_coords"
top: "yolo2_obj"
top: "yolo2_classes"
name: "yolo2"
type: "Yolo"
yolo_param {
  boxes: 3
  coords: 4
  classes: 80
  yolo_version: "V3"
  softmax: true
  background: false
}
```

– 算子3代码样例:

```
layer {
  bottom: "layer106-conv"
  top: "yolo3_coords"
  top: "yolo3_obj"
  top: "yolo3_classes"
  name: "yolo3"
  type: "Yolo"
  yolo_param {
    boxes: 3
    coords: 4
    classes: 80
    yolo_version: "V3"
    softmax: true
    background: false
  }
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

3. 最后一层增加YoloV3DetectionOutput算子。

对于YOLOv3网络，参考[8.1.2 扩展算子列表](#)在原始prototxt文件的最后增加后处理算子层YoloV3DetectionOutput，参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，YoloV3DetectionOutput算子有十个输入，两个输出；根据该原则，构造的后处理算子代码样例如下：

```
layer {
  name: "detection_out3"
  type: "YoloV3DetectionOutput"
  bottom: "yolo1_coords"
  bottom: "yolo2_coords"
  bottom: "yolo3_coords"
  bottom: "yolo1_obj"
  bottom: "yolo2_obj"
  bottom: "yolo3_obj"
  bottom: "yolo1_classes"
  bottom: "yolo2_classes"
  bottom: "yolo3_classes"
  bottom: "img_info"
  top: "box_out"
  top: "box_out_num"
  yolov3_detection_output_param {
    boxes: 3
    classes: 80
    relative: true
    obj_threshold: 0.5
    score_threshold: 0.5
    iou_threshold: 0.45
    pre_nms_topn: 512
    post_nms_topn: 1024
    biases_high: 10
    biases_high: 13
    biases_high: 16
  }
}
```

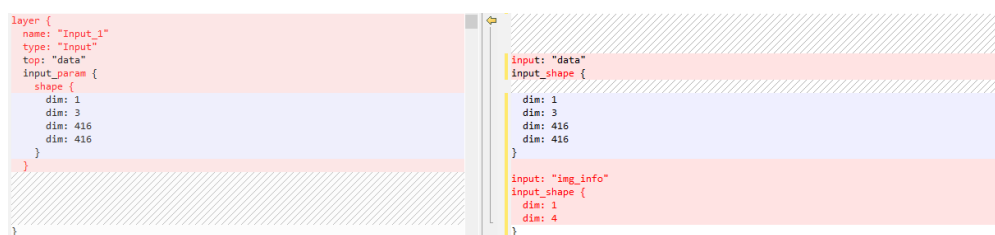
```
biases_high: 30
biases_high: 33
biases_high: 23
biases_mid: 30
biases_mid: 61
biases_mid: 62
biases_mid: 45
biases_mid: 59
biases_mid: 119
biases_low: 116
biases_low: 90
biases_low: 156
biases_low: 198
biases_low: 373
biases_low: 326
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

#### 4. 新增输入：

由于YoloV3DetectionOutput算子有img\_info输入，故模型输入时增加该输入。对比情况如图8-6所示，其中左边为原始算子prototxt，右边是适配昇腾AI处理器的prototxt。

图 8-6 原始算子与修改后的算子比对 3



```
layer {
  name: "Input_1"
  type: "Input"
  top: "data"
  input_param {
    shape {
      dim: 1
      dim: 3
      dim: 416
      dim: 416
    }
  }
}

input: "img_info"
input_shape {
  dim: 1
  dim: 4
}
```

代码样例如下，参数为[batch,4]，4表示netH、netW、scaleH、scaleW四个维度。其中netH,netW为网络模型输入的HW，scaleH,scaleW为原始图片的HW。

```
input: "img_info"
input_shape {
  dim: 1
  dim: 4
}
```

### 8.1.4.3 YOLOv2 网络模型 prototxt 修改

#### 说明

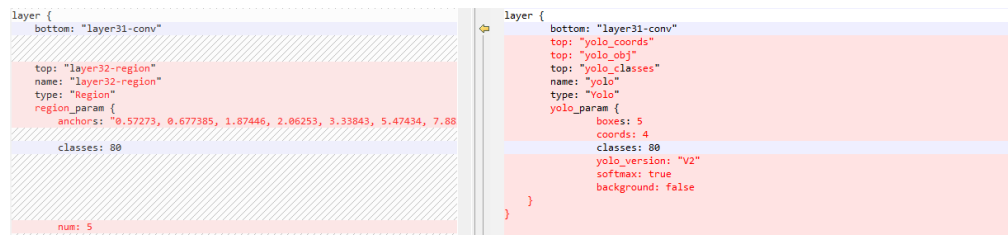
本章节所有的代码样例都不能直接复制到网络模型中使用，需要用户根据使用的网络模型，自行调整相应参数，比如bottom、top中的参数要和具体网络模型中的bottom、top一一对应，并且bottom和top对应的参数顺序不能更改。

#### 1. 修改Region算子。

由于Yolo算子+DetectionOutput算子才是特征检测网络完整的后处理逻辑，在增加YoloV2DetectionOutput算子之前需要将Region过程算子，替换为Yolo算子。

参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，Yolo算子有一个输入，三个输出；基于该原则，修改后的Yolo算子如下，对比情况如图8-7所示，其中左边为原始算子prototxt，右边是适配昇腾AI处理器的prototxt。

图 8-7 原始算子与修改后的算子比对 2



构造的代码样例如下：

```
layer {
  bottom: "layer31-conv"
  top: "yolo_coors"
  top: "yolo_obj"
  top: "yolo_classes"
  name: "yolo"
  type: "Yolo"
  yolo_param {
    boxes: 5
    coors: 4
    classes: 80
    yolo_version: "V2"
    softmax: true
    background: false
  }
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

## 2. 最后一层增加YoloV2DetectionOutput算子。

对于YoloV2网络，参考[8.1.2 扩展算子列表](#)在原始prototxt文件的最后增加后处理算子层YoloV2DetectionOutput，参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，YoloV2DetectionOutput算子有四个输入，两个输出；根据上述原则，构造的后处理算子代码样例如下：

```
layer {
  name: "detection_out2"
  type: "YoloV2DetectionOutput"
  bottom: "yolo_coors"
  bottom: "yolo_obj"
  bottom: "yolo_classes"
  bottom: "img_info"
  top: "box_out"
  top: "box_out_num"
  yolov2_detection_output_param {
    boxes: 5
    classes: 80
    relative: true
    obj_threshold: 0.5
    score_threshold: 0.5
    iou_threshold: 0.45
    pre_nms_topn: 512
    post_nms_topn: 1024
    biases: 0.572730
    biases: 0.677385
    biases: 1.874460
    biases: 2.062530
    biases: 3.338430
    biases: 5.474340
    biases: 7.882820
    biases: 3.527780
    biases: 9.770520
    biases: 9.168280
  }
}
```

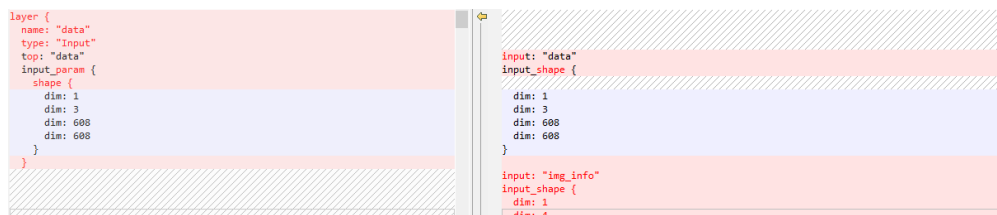
```
}  
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

### 3. 新增输入:

由于YoloV2DetectionOutput算子有img\_info输入, 故模型输入时增加该输入。对比情况如[图8-8](#)所示, 其中左边为原始算子prototxt, 右边是适配昇腾AI处理器的prototxt。

图 8-8 原始算子与修改后的算子比对 1



```
layer {  
  name: "data"  
  type: "Input"  
  top: "data"  
  input_param {  
    shape {  
      dim: 1  
      dim: 3  
      dim: 608  
      dim: 608  
    }  
  }  
}  
  
input: "data"  
input_shape {  
  dim: 1  
  dim: 3  
  dim: 608  
  dim: 608  
}  
  
input: "img_info"  
input_shape {  
  dim: 1  
  dim: 4  
}
```

代码样例如下, 参数为[batch,4], 4表示netH、netW、scaleH、scaleW四个维度。其中netH,netW为网络模型输入的HW, scaleH,scaleW为原始图片的HW。

```
input: "img_info"  
input_shape {  
  dim: 1  
  dim: 4  
}
```

## 8.1.4.4 SSD 网络模型 prototxt 修改

### 说明

本章节所有的代码样例都不能直接复制到网络模型中使用, 需要用户根据使用的网络模型, 自行调整相应参数, 比如bottom、top中的参数要和具体网络模型中的bottom、top一一对应, 并且bottom和top对应的参数顺序不能更改。

对于SSD网络, 参考[8.1.2 扩展算子列表](#)在原始prototxt文件的最后增加后处理算子层SSDDetectionOutput。

参见caffe.proto文件(该文件路径为\${INSTALL\_DIR}/include/proto), 先在LayerParameter message中添加自定义层参数的声明(如下自定义层在caffe.proto中已经声明, 用户无需再次添加):

```
message LayerParameter {  
  ...  
  optional SSDDetectionOutputParameter ssddetectionoutput_param = 232;  
  ...  
}
```

参见caffe.proto文件, 此算子的Type及属性定义如下:

```
message SSDDetectionOutputParameter {  
  optional int32 num_classes = 1 [default = 2];  
  optional bool share_location = 2 [default = true];  
  optional int32 background_label_id = 3 [default = 0];  
  optional float iou_threshold = 4 [default = 0.45];  
  optional int32 top_k = 5 [default = 400];  
  optional float eta = 6 [default = 1.0];  
  optional bool variance_encoded_in_target = 7 [default = false];  
  optional int32 code_type = 8 [default = 2];  
  optional int32 keep_top_k = 9 [default = 200];  
}
```

```
optional float confidence_threshold = 10 [default = 0.01];  
}
```

参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，SSDDetectionOutput算子有3个输入，2个输出，基于上述原则，构造的代码样例如下：

```
layer {  
  name: "detection_out"  
  type: "SSDDetectionOutput"  
  bottom: "bbox_delta"  
  bottom: "score"  
  bottom: "anchors"  
  top: "out_boxnum"  
  top: "y"  
  ssddetectionoutput_param {  
    num_classes: 2  
    share_location: true  
    background_label_id: 0  
    iou_threshold: 0.45  
    top_k: 400  
    eta: 1.0  
    variance_encoded_in_target: false  
    code_type: 2  
    keep_top_k: 200  
    confidence_threshold: 0.01  
  }  
}
```

- bottom输入中的bbox\_delta对应caffe原始网络中的mbox\_loc，score对应caffe原始网络中的mbox\_conf\_flatten，anchors对应caffe原始网络中的mbox\_priorbox；num\_classes取值需要与原始网络模型中的取值保持一致。
- top输出多batch场景下：
  - out\_boxnum输出shape是(batchnum,8)，每个batchnum的第一个值是实际框的个数。
  - y输出shape是(batchnum,len,8)，其中len是keep\_top\_k 128对齐后的取值（如batch为2，keep\_top\_k 为200，则最后输出shape为(2,256,8)），前256\*8个数据为第一个batch的结果。

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

#### 8.1.4.5 BatchedMatMul 算子 prototxt 修改

##### 说明

本章节所有的代码样例都不能直接复制到网络模型中使用，需要用户根据使用的网络模型，自行调整相应参数，比如bottom、top中的参数要和具体网络模型中的bottom、top一一对应，并且bottom和top对应的参数顺序不能更改。

该算子用户输出两个张量的乘积： $y=x1*x2$ （ $x1$ 和 $x2$ 的张量相乘， $x1$ 和 $x2$ 维数必须大于2而小于等于8）。如果网络模型使用该算子，则请参见该章节，修改相应prototxt后，再进行模型转换。

参见caffe.proto文件（该文件路径为\${INSTALL\_DIR}/include/proto），先在LayerParameter message中添加自定义层参数的声明（如下自定义层在caffe.proto中已经声明，用户无需再次添加）：

```
message LayerParameter {  
  ...  
  optional BatchMatMulParameter batch_matmul_param = 235;  
  ...  
}
```



参见caffe.proto文件，此算子的Type及属性定义如下：

```
message BatchMatMulParameter{
  optional bool adj_x1 = 1 [default = false];
  optional bool adj_x2 = 2 [default = false];
}
```

参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单，BatchedMatMul算子有2个输入，1个输出，基于上述原则，构造的代码样例如下：

```
layer {
  name: "batchmatmul"
  type: "BatchedMatMul"
  bottom: "matmul_data_1"
  bottom: "matmul_data_2"
  top: "batchmatmul_1"
  batch_matmul_param {
    adj_x1: false
    adj_x2: true
  }
}
```

参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

#### 8.1.4.6 SENet 网络模型 prototxt 修改

##### 📖 说明

本章节所有的代码样例都不能直接复制到网络模型中使用，需要用户根据使用的网络模型，自行调整相应参数，比如bottom、top中的参数要和具体网络模型中的bottom、top——对应，并且bottom和top对应的参数顺序不能更改。

该网络模型中的Axy算子，需要修改为Reshape、Scale、Eltwise三个算子，修改样例如下：

```
layer {
  name: "conv3_1"
  type: "Axy"
  bottom: "conv3_1_1x1_up"

  bottom: "conv3_1_1x1_increase"

  bottom: "conv3_1_1x1_proj"
  top: "conv3_1"
}

layer {
  name: "conv3_1_axpy_reshape"
  type: "Reshape"
  bottom: "conv3_1_1x1_up"
  top: "conv3_1_axpy_reshape"
  reshape_param {
    shape {
      dim: 0
      dim: -1
    }
  }
}
layer {
  name: "conv3_1_axpy_scale"
  type: "Scale"
  bottom: "conv3_1_1x1_increase"
  bottom: "conv3_1_axpy_reshape"
  top: "conv3_1_axpy_scale"
  scale_param {
    axis: 0
    bias_term: false
  }
}
layer {
  name: "conv3_1_axpy_eltwise"
  type: "Eltwise"
  bottom: "conv3_1_axpy_scale"
  bottom: "conv3_1_1x1_proj"
  top: "conv3_1"
}
```

修改后的代码样例如下：

```
layer {
  name: "conv3_1_axpy_reshape"
  type: "Reshape"
  bottom: "conv3_1_1x1_up"
  top: "conv3_1_axpy_reshape"
```

```

    reshape_param {
      shape {
        dim: 0
        dim: -1
      }
    }
  }
}
layer {
  name: "conv3_1_axpy_scale"
  type: "Scale"
  bottom: "conv3_1_1x1_increase"
  bottom: "conv3_1_axpy_reshape"
  top: "conv3_1_axpy_scale"
  scale_param {
    axis: 0
    bias_term: false
  }
}
}
layer {
  name: "conv3_1_axpy_eltwise"
  type: "Eltwise"
  bottom: "conv3_1_axpy_scale"
  bottom: "conv3_1_1x1_proj"
  top: "conv3_1"
}
}

```

Reshape、Scale、Eltwise算子相关参数解释请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

## 8.2 定制网络修改 ( TensorFlow )

### 8.2.1 概述

## 简介

本章节介绍如何将TensorFlow有控制流算子的网络模型（如图8-9所示）转成函数类算子的网络模型（如图8-10所示），然后利用ATC工具转换成适配昇腾AI处理器的离线模型。

图 8-9 有控制流算子的网络模型

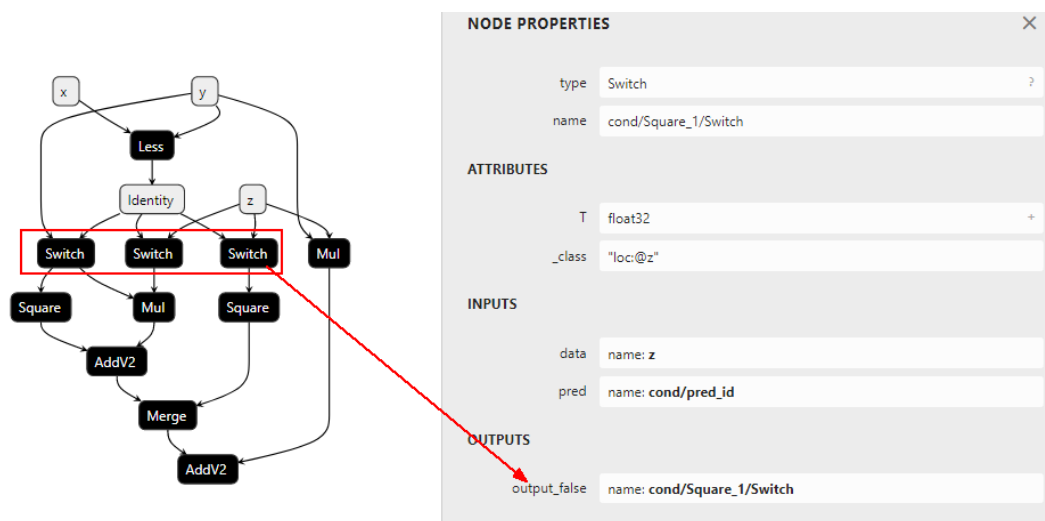
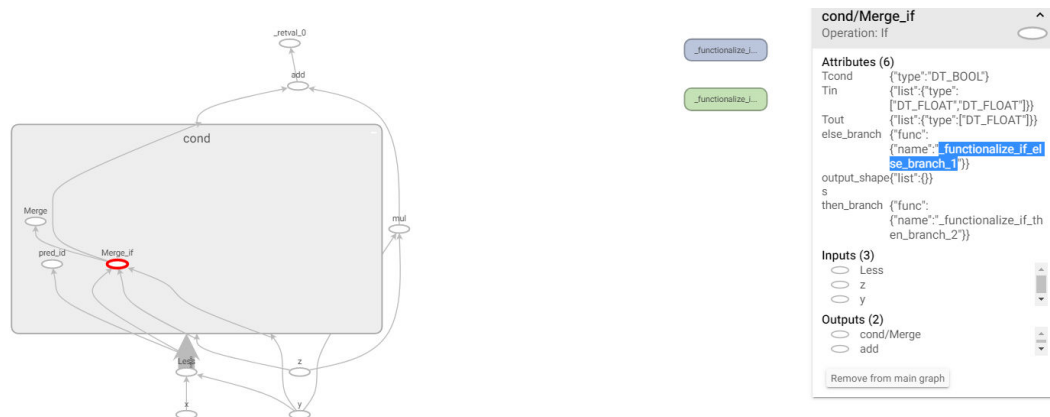


图 8-10 函数类算子的网络模型



## 使用前提

- 使用该工具时，请确保工具所在服务器能连接网络。
- 安装bazel编译工具。

请参见<https://docs.bazel.build/versions/master/install-ubuntu.html>官方地址安装bazel编译工具。

- 安装TensorFlow以及依赖future。

ATC安装路径下的func2graph.py脚本依赖TensorFlow，使用**pip3.7.5 list**查看列表中是否有TensorFlow依赖，若有则不用安装，否则执行如下命令安装：

```
pip3.7.5 install tensorflow==1.15 --user
```

bazel编译工具依赖python，请使用**pip3.7.5 list**命令查看是否安装future、patch，若有则不用安装，否则执行如下命令安装：

```
pip3.7.5 install future --user
pip3.7.5 install patch --user
pip3.7.5 install numpy --user
```

如果执行上述命令时无法连接网络，且提示“Could not find a version that satisfies the requirement xxx”，请参见[8.2.4.2 使用pip3.7.5 install软件时提示"Could not find a version that satisfies the requirement xxx"解决](#)。

## 8.2.2 编译可执行文件

请以ATC软件包安装用户进行如下操作：

**步骤1** 从<https://github.com/tensorflow/tensorflow/archive/v1.15.0.tar.gz>链接下载Tensorflow源码，然后将下载的源码上传到ATC工具所在Linux服务器任意目录。

**步骤2** 登录Linux服务器，切换到Tensorflow源码所在路径，执行如下命令解压源码包：

```
tar -zxvf tensorflow-1.15.0.tar.gz
```

**步骤3** 进入解压后的tensorflow-1.15.0，安装补丁。

参见[8.2.4.4 获取xlacompile.patch补丁文件](#)获取xlacompile.patch补丁，然后上传到Linux服务器tensorflow-1.15.0路径下，执行如下命令安装补丁：

```
patch -p1 < xlacompile.patch
```

**步骤4** 切换到tensorflow-1.15.0目录，执行如下命令编译xlacompile工具：

```
bazel build --config=monolithic //tensorflow/compiler/aot:xlacompile
```

若出现类似如下信息，则说明编译成功，编译大约需要10分钟左右；若编译失败，请参见[8.2.4.1 使用bazel编译工具编译时提示 “An error occurred during the fetch of repository 'io\\_bazel\\_rules\\_docker'”](#)，编译失败解决。

```
Target //tensorflow/compiler/aot:xlacompile up-to-date:
  bazel-bin/tensorflow/compiler/aot:xlacompile
INFO: Elapsed time: 214.550s, Critical Path: 73.38s
INFO: 1511 processes: 1511 local.
INFO: Build completed successfully, 1513 total actions
```

编译成功后，会在\$HOME/.cache/bazel/\_bazel\_test/abd37aaac8a380ca5a3f13938322fcb2/external/org\_tensorflow/bazel-out/k8-opt/bin/tensorflow/compiler/aot路径生成**xlacompile**可执行文件（该路径只是样例，请以用户实际编译后的为准）。

**xlacompile**可执行文件用于将控制流算子的网络模型转成函数类算子的网络模型。

#### 步骤5 切换到tensorflow-1.15.0目录，执行如下命令编译**summarize\_graph**工具：

```
bazel build --config=monolithic -c opt //tensorflow/tools/graph_transforms:summarize_graph
```

若出现类似如下信息，则说明编译成功：

```
Target //tensorflow/tools/graph_transforms:summarize_graph up-to-date:
  bazel-bin/tensorflow/tools/graph_transforms/summarize_graph
INFO: Elapsed time: 70.474s, Critical Path: 53.16s
INFO: 1028 processes: 1028 local.
INFO: Build completed successfully, 1053 total actions
```

编译成功后，会在\$HOME/.cache/bazel/\_bazel\_test/abd37aaac8a380ca5a3f13938322fcb2/execroot/org\_tensorflow/bazel-out/k8-opt/bin/tensorflow/tools/graph\_transforms路径生成**summarize\_graph**可执行文件（该路径只是样例，请以用户实际编译后的为准）。

**summarize\_graph**可执行文件用来查看有控制流算子网络模型的输入输出节点，方便用户构造config.pbtxt输入输出配置文件。

----结束

## 8.2.3 转换模型

下面以Switch\_v1.pb网络模型为例进行说明，演示如何将控制流算子网络模型转换成函数类算子网络模型，然后通过ATC工具转换成适配昇腾AI处理器的离线模型。请先参见[8.2.4.3 获取Switch\\_v1.pb网络模型](#)获取Switch\_v1.pb网络模型。

#### 步骤1 获取控制流算子网络模型的输出。

切换到**summarize\_graph**可执行文件所在路径，执行如下命令：

```
./summarize_graph --in_graph=/home/test/module/Switch_v1.pb
```

若返回如下信息，则说明执行成功：

```
Found 3 possible inputs: (name=x, type=float(1), shape=<unknown>) (name=y, type=float(1), shape=<unknown>) (name=z, type=float(1), shape=<unknown>)
No variables spotted.
Found 1 possible outputs: (name=add, op=AddV2)
Found 0 (0) const parameters, 0 (0) variable parameters, and 0 control_edges
Op types used: 3 Placeholder, 3 Switch, 2 AddV2, 2 Mul, 2 Square, 1 Identity, 1 Less, 1 Merge
To use with tensorflow/tools/benchmark:benchmark_model try these arguments:
bazel run tensorflow/tools/benchmark:benchmark_model -- --graph=/home/test/module/Switch_v1.pb --
show_flops --input_layer=x,y,z --input_layer_type=float,float,float --input_layer_shape=: --output_layer=add
```

#### 步骤2 构造config.pbtxt输出配置文件。

在任意路径执行**vim config.pbtxt**命令创建config.pbtxt文件，本示例以在\$HOME/module路径创建为例进行说明。

根据**步骤1**所示，该网络模型有一个输出，构造的config.pbtxt配置文件样例如下（如下配置文件只是样例，需要用户根据实际情况修改输出算子的name，其中fetch表示输出）：

```
fetch {  
  id { node_name: "add" }  
}
```

配置完成后，执行**wq!**命令退出。

### 步骤3 生成函数类算子网络模型的配置文件。

在任意路径执行如下命令设置**xlacompile**命令执行过程中的打屏日志信息：

```
export TF_CPP_MIN_LOG_LEVEL=0  
export TF_CPP_MIN_VLOG_LEVEL=1
```

切换到**xlacompile**可执行文件所在路径，执行如下命令：

```
./xlacompile --graph=/home/test/module/Switch_v1.pb --config=/home/test/module/config.pbtxt --output=/home/test/module/Switch_v1_v2
```

如果提示“Successfully convert ...”信息，则说明转换成功。切换到**--output**参数指定的路径，可以看到如下输出文件：

```
-rw-rw-r-- 1 test test 1236 Jun 20 17:13 Switch_v1_v2.pb  
-rw-rw-r-- 1 test test 4803 Jun 20 17:13 Switch_v1_v2.pbtxt
```

### 步骤4 函数类算子网络模型生成graph子图。后续使用ATC工具进行模型转换时，需要使用该文件生成子图。

在任意路径执行如下命令，将函数类算子网络模型生成子图：

```
python3.7.5 ${INSTALL_DIR}/compiler/python/func2graph/func2graph.py -m /home/test/module/  
Switch_v1_v2.pb
```

若提示如下信息，则说明生成成功。

```
graph_def_file: /home/test/module/graph_def_library.pbtxt  
INFO: Convert to subgraphs successfully.
```

### 步骤5 函数类算子网络模型转换成适配昇腾AI处理器的离线模型。

参见**设置环境变量**设置ATC工具执行时需设置的环境变量，然后执行如下命令进行模型转换：

```
atc --model=/home/test/module/Switch_v1_v2.pb --framework=3 --output=/home/test/module/out/  
Switch_v1_v2_to_om --soc_version=<soc_version>
```

若提示如下信息，则说明模型转换成功：

```
ATC run success
```

成功执行命令后，在**--output**参数指定的路径下，可查看模型文件。

----结束

## 8.2.4 FAQ

### 8.2.4.1 使用 bazel 编译工具编译时提示 “An error occurred during the fetch of repository 'io\_bazel\_rules\_docker'”，编译失败

#### 问题描述

使用 `bazel build --config=monolithic //tensorflow/compiler/aot:xlacompile` 命令编译过程中，提示 “ERROR: An error occurred during the fetch of repository 'io\_bazel\_rules\_docker'”，检查服务器，能够连接网络，但仍旧提示如下错误信息：

```
ERROR: An error occurred during the fetch of repository 'io_bazel_rules_docker':
  java.io.IOException: Error downloading [https://github.com/bazelbuild/rules_docker/releases/download/v0.14.3/rules_docker-v0.14.3.tar.gz] to /home/test/.cache/bazel/_bazel_test/abd37aac8a380ca5a3f13938322fcb2/external/io_bazel_rules_docker/rules_docker-v0.14.3.tar.gz: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
ERROR: no such package '@io_bazel_rules_docker//repositories': java.io.IOException: Error downloading [https://github.com/bazelbuild/rules_docker/releases/download/v0.14.3/rules_docker-v0.14.3.tar.gz] to /home/test/.cache/bazel/_bazel_test/abd37aac8a380ca5a3f13938322fcb2/external/io_bazel_rules_docker/rules_docker-v0.14.3.tar.gz: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
INFO: Elapsed time: 24.586s
INFO: 0 processes.
FAILED: Build did NOT complete successfully (0 packages loaded)
```

#### 解决方案

如果连网情况下，仍旧提示上述无法下载压缩包的问题，则请参见如下方法解决：

- 步骤1** 在本地Windows PC将如下链接中的附件下载到本地，然后上传到linux服务器任意路径，例如上传到\$HOME/bazel\_tools路径。

```
https://github.com/bazelbuild/rules_docker/releases/download/v0.14.3/rules_docker-v0.14.3.tar.gz
https://github.com/bazelbuild/bazel-skylib/releases/download/0.8.0/bazel-skylib-0.8.0.tar.gz
https://github.com/bazelbuild/rules_swift/releases/download/0.11.1/rules_swift-0.11.1.tar.gz
https://github.com/llvm-mirror/llvm/archive/7a7e03f906aada0cf4b749b51213fe5784eeff84.tar.gz
```

则上述文件在linux服务器绝对路径为：

```
/home/test/bazel_tools/rules_docker-v0.14.3.tar.gz
/home/test/bazel_tools/bazel-skylib-0.8.0.tar.gz
/home/test/bazel_tools/rules_swift-0.11.1.tar.gz
/home/test/bazel_tools/llvm-7a7e03f906aada0cf4b749b51213fe5784eeff84.tar.gz
```

- 步骤2** 修改bazel编译工具相关文件中的下载链接：

1. 切换到tensorflow-1.15.0目录，使用**vi WORKSPACE**命令打开WORKSPACE，修改该文件中的下载链接，将下载链接修改为linux服务器绝对路径地址：

```
bazel_toolchains_repositories()
http_archive(
  name = "io_bazel_rules_docker",
  sha256 = "6287241e033d247e9da5ff705dd6ef526bac39ae82f3d17de1b69f8cb313f9cd",
  strip_prefix = "rules_docker-0.14.3",
  urls = ["file:///home/test/bazel_tools/rules_docker-v0.14.3.tar.gz"],
)

load(
  "@io_bazel_rules_docker//repositories:repositories.bzl",
  container_repositories = "repositories",
)
container_repositories()

load("//third_party/toolchains/preconfig:generate_workspace.bzl",
  "remote_config_workspace")
remote_config_workspace()
```

```
# Apple and Swift rules.
http_archive(
  name = "build_bazel_rules_apple",
  sha256 = "6efdde60c91724a2be7f89b0c0a64f01138a45e63ba5add2dca2645d981d23a1",
  urls = ["https://github.com/bazelbuild/rules_apple/releases/download/0.17.2/rules_apple.0.17.2.tar.gz"],
) # https://github.com/bazelbuild/rules_apple/releases
http_archive(
  name = "build_bazel_rules_swift",
  sha256 = "96a86afcbdb215f8363e65a10cf023b752e90b23abf02272c4fc668fcb70311",
  urls = ["file:///home/test/bazel_tools/rules_swift.0.11.1.tar.gz"],
) # https://github.com/bazelbuild/rules_swift/releases
```

执行**wq!**保存退出。

2. 修改tensorflow-1.15.0/tensorflow路径下**workspace.bzl**文件中llvm对应的链接：

```
# TODO(phawkins): currently, this rule uses an unofficial LLVM mirror.
# Switch to an official source of snapshots if/when possible.
tf_http_archive(
  name = "llvm",
  build_file = clean_dep("//third_party/llvm:llvm.autogenerated.BUILD"),
  sha256 = "599b89411df88b9e2be40b019e7ab0f7c9c10dd5ab1c948cd22e678cc8f8f352",
  strip_prefix = "llvm-7a7e03f906aada0cf4b749b51213fe5784eeff84",
  urls = [
    "https://mirror.bazel.build/github.com/llvm-mirror/llvm/archive/7a7e03f906aada0cf4b749b51213fe5784eeff84.tar.gz",
    "file:///home/test/bazel_tools/llvm-7a7e03f906aada0cf4b749b51213fe5784eeff84.tar.gz",
  ],
)
```

3. 修改完成后，重新切换到tensorflow-1.15.0目录，执行如下命令编译**xlacompile**工具：

```
bazel build --config=monolithic //tensorflow/compiler/aot:xlacompile
```

若提示“ERROR: An error occurred during the fetch of repository 'bazel\_skylib'”，则进入下一步继续修改**bazel\_skylib**相关的文件。

4. 修改**.cache**目录中的相关链接：

切换到\$HOME目录，使用**grep -r bazel-skylib.0.8.0 .cache/**命令查看**.cache**目录哪个文件引用**bazel-skylib.0.8.0.tar.gz**的url，根据返回信息，进入引用该url文件所在的目录，例如.cache/bazel/\_bazel\_test/abd37aaac8a380ca5a3f13938322fcb2/external/io\_bazel\_rules\_closure/closure/repositories.bzl（该路径只是样例，请以用户实际编译后的为准）

打开该文件，将其中的url改为：

```
file:///home/test/bazel_tools/bazel-skylib.0.8.0.tar.gz
```

执行**wq!**保存退出。

----结束

### 8.2.4.2 使用 pip3.7.5 install 软件时提示" Could not find a version that satisfies the requirement xxx"

#### 问题描述

安装依赖时，使用**pip3.7.5 install xxx**命令安装相关软件时提示无法连接网络，且提示“Could not find a version that satisfies the requirement xxx”，使用**apt-get update**命令检查源可用。提示信息如下图所示。

图 8-11 使用 pip3.7.5 安装软件提示信息

```
ascend@dgghicpr32833:~$ pip3 install numpy --user
Collecting numpy
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725b9320>: Failed to establish a new connection: [Errno 101] Network is unreachable')': /simple/numpy/
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd908>: Failed to establish a new connection: [Errno 101] Network is unreachable')': /simple/numpy/
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd908>: Failed to establish a new connection: [Errno 101] Network is unreachable')': /simple/numpy/
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cdac8>: Failed to establish a new connection: [Errno 101] Network is unreachable')': /simple/numpy/
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd908>: Failed to establish a new connection: [Errno 101] Network is unreachable')': /simple/numpy/
Could not find a version that satisfies the requirement numpy (from versions: )
```

## 可能原因

没有配置pip源。

## 解决方法

配置pip源，配置方法如下：

**步骤1** 使用ATC软件包安装用户，执行如下命令：

```
cd ~/.pip
```

如果提示目录不存在，则执行如下命令创建：

```
mkdir ~/.pip
cd ~/.pip
```

在.pip目录下创建pip.conf 文件，命令为：

```
touch pip.conf
```

**步骤2** 编辑pip.conf文件。

使用**vi pip.conf**命令打开pip.conf文件，写入如下内容：

```
[global]
#可用的源，请根据实际情况进行替换。
index-url=http://xxx
[install]
#可信主机，请根据实际情况进行替换。
trusted-host=xxx
```

**步骤3** 执行**wq!**命令保存文件。

----结束

### 8.2.4.3 获取 Switch\_v1.pb 网络模型

**步骤1** 将如下文件中的脚本复制到.py文件中，例如复制到*Switch\_v1.py*文件中。

```
import os
import numpy as np
import tensorflow as tf

x = tf.compat.v1.placeholder(tf.float32, name='x')
y = tf.compat.v1.placeholder(tf.float32, name='y')
z = tf.compat.v1.placeholder(tf.float32, name='z')

def then_branch(x, y, z):
    m = tf.square(x)
    return m + tf.multiply(y, z)

def else_branch(x, y, z):
    m = tf.pow(x, y)
    return m - tf.div(y, z)

# 控制流算子使用入口，执行脚本之后，在图中生成对应的V1控制流算子
```



```
def testDefun(x, y, z):
    return tf.cond(pred=x < y, true_fn=lambda: then_branch(x, y, z), false_fn=lambda: else_branch(x, y, z)), y

def testCase(x, y, z):
    a, b = testDefun(x, y, z)
    return a + b * z

with tf.compat.v1.Session() as sess:
    result = sess.run(testCase(x, y, z), feed_dict={x: 1., y: .6, z: .2})

    with tf.io.gfile.GFile('./Switch_v1.pb', 'wb') as f:
        f.write(sess.graph_def.SerializeToString())
```

**步骤2** 切换到`Switch_v1.py`脚本所在目录，执行如下命令生成`Switch_v1.pb`网络模型：

```
python3.7.5 Switch_v1.py
```

命令执行完毕，在当前目录会生成`Switch_v1.pb`网络模型。

----结束

#### 8.2.4.4 获取 xlacompile.patch 补丁文件

用户安装完`xlacompile.patch`补丁，编译成`xlacompile`工具后，该工具可以将有控制流的V1网络模型转成函数类的V2网络模型。

将如下代码复制到文件中，并另存为`xlacompile.patch`，然后上传到Linux服务器`tensorflow-1.15.0`路径下：

```
---
WORKSPACE                                | 7 +
tensorflow/compiler/aot/BUILD              | 27 +++++
tensorflow/compiler/aot/xlacompile_main.cc | 170 +++++
tensorflow/compiler/tf2xla/tf2xla.cc      | 6 +
tensorflow/compiler/tf2xla/tf2xla.h       | 4 +
5 files changed, 195 insertions(+)
create mode 100644 tensorflow/compiler/aot/xlacompile_main.cc

diff --git a/WORKSPACE b/WORKSPACE
index 74ea14d..d2265f9 100644
--- a/WORKSPACE
+++ b/WORKSPACE
@@ -34,6 +34,13 @@ load(

    bazel_toolchains_repositories()

+http_archive(
+  name = "io_bazel_rules_docker",
+  sha256 = "6287241e033d247e9da5ff705dd6ef526bac39ae82f3d17de1b69f8cb313f9cd",
+  strip_prefix = "rules_docker-0.14.3",
+  urls = ["https://github.com/bazelbuild/rules_docker/releases/download/v0.14.3/rules_docker-
v0.14.3.tar.gz"],
+)
+
load(
  "@io_bazel_rules_docker//repositories:repositories.bzl",
  container_repositories = "repositories",

diff --git a/tensorflow/compiler/aot/BUILD b/tensorflow/compiler/aot/BUILD
index f871115..b2620db 100644
--- a/tensorflow/compiler/aot/BUILD
+++ b/tensorflow/compiler/aot/BUILD
@@ -106,6 +106,33 @@ cc_library(
  ],
)

+tf_cc_binary(
```

```
+ name = "xlacompile",
+ visibility = ["//visibility:public"],
+ deps = [":xlacompile_main"],
+)
+
+cc_library(
+ name = "xlacompile_main",
+ srcs = ["xlacompile_main.cc"],
+ visibility = ["//visibility:public"],
+ deps = [
+     ":tfcompile_lib",
+     "//tensorflow/compiler/tf2xla:tf2xla_proto",
+     "//tensorflow/compiler/tf2xla:tf2xla_util",
+     "//tensorflow/compiler/xla:debug_options_flags",
+     "//tensorflow/compiler/xla/service:compiler",
+     "//tensorflow/core:core_cpu",
+     "//tensorflow/core:core_cpu_internal",
+     "//tensorflow/core:framework",
+     "//tensorflow/core:framework_internal",
+     "//tensorflow/core:graph",
+     "//tensorflow/core:lib",
+     "//tensorflow/core:protos_all_cc",
+     "@com_google_absl//absl/strings",
+ ],
+)
+
+ # NOTE: Most end-to-end tests are in the "tests" subdirectory, to ensure that
+ # tfcompile.bzl correctly handles usage from outside of the package that it is
+ # defined in.

diff --git a/tensorflow/compiler/aot/xlacompile_main.cc b/tensorflow/compiler/aot/xlacompile_main.cc
new file mode 100644
index 0000000..bc795ef
--- /dev/null
+++ b/tensorflow/compiler/aot/xlacompile_main.cc
@@ -0,0 +1,170 @@
+/* Copyright 2017 The TensorFlow Authors. All Rights Reserved.
+
+Licensed under the Apache License, Version 2.0 (the "License");
+you may not use this file except in compliance with the License.
+You may obtain a copy of the License at
+
+    http://www.apache.org/licenses/LICENSE-2.0
+
+Unless required by applicable law or agreed to in writing, software
+distributed under the License is distributed on an "AS IS" BASIS,
+WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
+See the License for the specific language governing permissions and
+limitations under the License.
+=====*/
+
+#include <memory>
+#include <string>
+#include <utility>
+#include <vector>
+#include <map>
+
+#include "tensorflow/compiler/aot/flags.h"
+#include "tensorflow/compiler/tf2xla/tf2xla.h"
+#include "tensorflow/compiler/tf2xla/tf2xla_util.h"
+#include "tensorflow/core/platform/init_main.h"
+
+namespace tensorflow {
+namespace xlacompile {
+
+const char kUsageHeader[] =
+    "xlacompile performs ahead-of-time compilation of a TensorFlow graph,\n"
+    "resulting in an object file compiled for your target architecture, and a\n"
+    "header file that gives access to the functionality in the object file.\n"
```

```
+ "A typical invocation looks like this:\n"
+ "\n"
+ " $ xlacompile --graph=mygraph.pb --config=config.pbtxt --output=output.pbtxt\n"
+ "\n";
+
+void AppendMainFlags(std::vector<Flag>* flag_list, tfcompile::MainFlags* flags) {
+ const std::vector<Flag> tmp = {
+   {"graph", &flags->graph,
+    "Input GraphDef file. If the file ends in '.pbtxt' it is expected to "
+    "be in the human-readable proto text format, otherwise it is expected "
+    "to be in the proto binary format."},
+   {"config", &flags->config,
+    "Input file containing Config proto. If the file ends in '.pbtxt' it "
+    "is expected to be in the human-readable proto text format, otherwise "
+    "it is expected to be in the proto binary format."},
+   {"output", &flags->out_session_module,
+    "Output session module proto. Will generate '.pb' and '.pbtxt' file."},
+ };
+ flag_list->insert(flag_list->end(), tmp.begin(), tmp.end());
+}
+
+Status ReadProtoFile(const string& fname, protobuf::Message* proto) {
+ if (absl::EndsWith(fname, ".pbtxt")) {
+   return ReadTextProto(Env::Default(), fname, proto);
+ } else {
+   return ReadBinaryProto(Env::Default(), fname, proto);
+ }
+}
+
+Status Main(tfcompile::MainFlags& flags) {
+ // Process config.
+ tf2xla::Config config;
+ if (flags.config.empty()) {
+   return errors::InvalidArgument("Must specify --config");
+ }
+ TF_RETURN_IF_ERROR(ReadProtoFile(flags.config, &config));
+ TF_RETURN_IF_ERROR(ValidateConfig(config));
+ if (flags.dump_fetch_nodes) {
+   std::set<string> nodes;
+   for (const tf2xla::Fetch& fetch : config.fetch()) {
+     nodes.insert(fetch.id().node_name());
+   }
+   std::cout << absl::StrJoin(nodes, ",");
+   return Status::OK();
+ }
+
+ // Read and initialize the graph.
+ if (flags.graph.empty()) {
+   return errors::InvalidArgument("Must specify --graph");
+ }
+ if (flags.out_session_module.empty()) {
+   return errors::InvalidArgument("Must specify --output");
+ }
+
+ string output_pb_bin = flags.out_session_module + ".pb";
+ string output_pb_txt = flags.out_session_module + ".pbtxt";
+ if (output_pb_bin == flags.config || output_pb_bin == flags.graph ||
+     output_pb_txt == flags.config || output_pb_txt == flags.graph) {
+   return errors::InvalidArgument("Must different --config --graph --output");
+ }
+
+ GraphDef graph_def;
+ TF_RETURN_IF_ERROR(ReadProtoFile(flags.graph, &graph_def));
+ std::unique_ptr<Graph> graph;
+ TF_RETURN_IF_ERROR(ConvertGraphDefToXla(graph_def, config, graph));
+
+ std::map<string, string> arg_name_maps;
+ GraphDef new_graph_def;
+ graph->ToGraphDef(&new_graph_def);
```

```

+ // Delete _class attribute for: expects to be colocated with unknown node
+ for (int i = 0; i < new_graph_def.node_size(); ++i) {
+   NodeDef *node = new_graph_def.mutable_node(i);
+   node->mutable_attr()->erase("_class");
+   if (node->op() == "_Retval") {
+     node->set_name(absl::StrCat(node->attr().at("index").i(), "_Retval"));
+   }
+   if (node->op() == "_Arg") {
+     const string name = node->name();
+     node->set_name(absl::StrCat(node->attr().at("index").i(), "_Arg"));
+     arg_name_maps[name] = node->name();
+   }
+ }
+
+ for (int i = 0; i < new_graph_def.node_size() && !arg_name_maps.empty(); ++i) {
+   NodeDef *node = new_graph_def.mutable_node(i);
+   for (int j = 0; j < node->input_size(); ++j) {
+     auto it = arg_name_maps.find(node->input(j));
+     if (it != arg_name_maps.end()) {
+       *node->mutable_input(j) = it->second;
+     }
+   }
+ }
+
+ TF_RETURN_IF_ERROR(WriteBinaryProto(Env::Default(), output_pb_bin, new_graph_def));
+ std::cerr << "Successfully convert: " << output_pb_bin << "\n";
+ TF_RETURN_IF_ERROR(WriteTextProto(Env::Default(), output_pb_txt, new_graph_def));
+ std::cerr << "Successfully convert: " << output_pb_txt << "\n";
+ return Status::OK();
+}
+
+} // end namespace xlacompile
+} // end namespace tensorflow
+
+int main(int argc, char** argv) {
+  tensorflow::tfcompile::MainFlags flags;
+  flags.target_triple = "x86_64-pc-linux";
+  flags.out_function_object = "out_model.o";
+  flags.out_metadata_object = "out_helper.o";
+  flags.out_header = "out.h";
+  flags.entry_point = "entry";
+
+  std::vector<tensorflow::Flag> flag_list;
+  tensorflow::xlacompile::AppendMainFlags(&flag_list, &flags);
+
+  tensorflow::string usage = tensorflow::xlacompile::kUsageHeader;
+  usage += tensorflow::Flags::Usage(argv[0], flag_list);
+  if (argc > 1 && absl::string_view(argv[1]) == "--help") {
+    std::cerr << usage << "\n";
+    return 0;
+  }
+  bool parsed_flags_ok = tensorflow::Flags::Parse(&argc, argv, flag_list);
+  QCHECK(parsed_flags_ok) << "\n" << usage;
+
+  tensorflow::port::InitMain(usage.c_str(), &argc, &argv);
+  QCHECK(argc == 1) << "\nERROR: This command does not take any arguments "
+    "other than flags\n\n"
+    << usage;
+  tensorflow::Status status = tensorflow::xlacompile::Main(flags);
+  if (status.code() == tensorflow::error::INVALID_ARGUMENT) {
+    std::cerr << "INVALID ARGUMENTS: " << status.error_message() << "\n\n"
+      << usage;
+    return 1;
+  } else {
+    TF_QCHECK_OK(status);
+  }
+  return 0;
+}

```

```
diff --git a/tensorflow/compiler/tf2xla/tf2xla.cc b/tensorflow/compiler/tf2xla/tf2xla.cc
index 3c2b256..3872776 100644
--- a/tensorflow/compiler/tf2xla/tf2xla.cc
+++ b/tensorflow/compiler/tf2xla/tf2xla.cc
@@ -410,4 +410,10 @@ Status ConvertGraphDefToXla(const GraphDef& graph_def,
    return Status::OK();
  }

+Status ConvertGraphDefToXla(const GraphDef &graph_def,
+                             const tf2xla::Config &config,
+                             std::unique_ptr<Graph> &graph) {
+  return InitGraph(graph_def, config, &graph);
+}
+
  } // namespace tensorflow

diff --git a/tensorflow/compiler/tf2xla/tf2xla.h b/tensorflow/compiler/tf2xla/tf2xla.h
index 432a12a..969500c 100644
--- a/tensorflow/compiler/tf2xla/tf2xla.h
+++ b/tensorflow/compiler/tf2xla/tf2xla.h
@@ -20,6 +20,7 @@ limitations under the License.
#include "tensorflow/compiler/xla/client/client.h"
#include "tensorflow/compiler/xla/client/xla_computation.h"
#include "tensorflow/core/framework/graph.pb.h"
+#include "tensorflow/core/graph/graph.h"

namespace tensorflow {

@@ -34,6 +35,9 @@ Status ConvertGraphDefToXla(const GraphDef& graph_def,
                             const tf2xla::Config& config, xla::Client* client,
                             xla::XlaComputation* computation);

+Status ConvertGraphDefToXla(const GraphDef &graph_def,
+                             const tf2xla::Config &config,
+                             std::unique_ptr<Graph> &graph);
+} // namespace tensorflow

#ifdef TENSORFLOW_COMPILER_TF2XLA_TF2XLA_H_

--
1.8.3.1
```

# 9 参考

[算子规格参考](#)

[错误码参考](#)

[一键收集故障信息](#)

[开启Cast算子自动插入特性](#)

## 9.1 算子规格参考

### 9.1.1 Caffe 框架算子规格

该算子规格仅适用于Caffe框架原生IR定义的网络模型，算子详情请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持Caffe算子清单。

如果要查看基于Ascend IR定义的单算子信息，请参见《[算子清单（开放态）](#)》手册。

当前支持的Caffe版本为caffe-master分支commit id为9b891540183ddc834a02b2bd81b31afae71b2153。

### 9.1.2 TensorFlow 框架算子规格

该算子规格仅适用于TensorFlow框架原生IR定义的网络模型。算子详情请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持TensorFlow算子清单。

如果要查看基于Ascend IR定义的单算子信息，请参见《[算子清单（开放态）](#)》手册。

当前支持的TensorFlow版本为1.15.0。

### 9.1.3 ONNX 算子规格

该算子规格仅适用于ONNX原生IR定义的网络模型。算子详情请参见《[CANN Caffe/TensorFlow/ONNX算子规格清单](#)》>支持ONNX算子清单。

如果要查看基于Ascend IR定义的单算子信息，请参见《[算子清单（开放态）](#)》手册。

当前支持的ONNX版本为1.8.0、Opset版本为v9~v13、ONNX Runtime版本为1.6.0。

## 9.2 错误码参考

### 9.2.1 Command Line Errors

#### 9.2.1.1 E10001 Invalid Argument

##### Symptom

Value [%s] for [--%s] is invalid. Reason: %s.

##### Possible Cause

Invalid Argument.

##### Solution

Try again with a valid argument.

#### 9.2.1.2 E10002 Invalid --input\_shape Argument

##### Symptom

Value [%s] for [--input\_shape] is invalid. Reason: %s. Must be formatted as [%s].

##### Possible Cause

The [--input\_shape] argument in the atc command line is invalid.

##### Solution

The valid format is [input\_name1:n1,c1,h1,w1;input\_name2:n2,c2,h2,w2]. Replace [input\_name#] with node names. Ensure that the shape values are integers.

#### 9.2.1.3 E10003 Invalid Argument

##### Symptom

Value [%s] for the [--%s] argument is invalid. Reason: %s.

##### Possible Cause

Invalid Argument.

##### Solution

Try again with a valid argument.

#### 9.2.1.4 E10004 Invalid Argument

##### Symptom

Value for [--%s] is empty.

##### Possible Cause

The argument must not be empty.

##### Solution

Try again with a valid argument.

#### 9.2.1.5 E10005 Invalid Argument

##### Symptom

Value [%s] for [--%s] is invalid. Must be either [true] or [false].

##### Possible Cause

The argument is invalid. Must be either [true] or [false].

##### Solution

Try again with a valid argument.

#### 9.2.1.6 E10006 Invalid Argument

##### Symptom

Value [%s] for [--%s] is invalid. Must be either 1 or 0.

##### Possible Cause

The argument is invalid. Must be either 1 or 0.

##### Solution

Try again with a valid argument.

#### 9.2.1.7 E10007 Invalid Required Argument

##### Symptom

[--%s] is required. Must be [%s].

##### Possible Cause

The argument is invalid.



## Solution

Try again with a valid argument.

### 9.2.1.8 E10008 Invalid Argument

#### Symptom

[--weight] must not be empty when [--framework] is set to 0 (Caffe).

#### Possible Cause

[--weight] must not be empty when [--framework] is set to 0 (Caffe).

## Solution

1. If the source model framework is Caffe, try again with a valid [--weight] argument.
2. If the source model framework is not Caffe, try again with a valid [--framework] argument.

### 9.2.1.9 E10009 Invalid Dynamic Shape Argument

#### Symptom

[--dynamic\_batch\_size], [--dynamic\_image\_size], and [--dynamic\_dims] are mutually exclusive.

#### Possible Cause

[--dynamic\_batch\_size], [--dynamic\_image\_size], and [--dynamic\_dims] are mutually exclusive.

## Solution

1. In dynamic shape scenarios, include only one of these options in your command line.
2. In static shape scenarios, remove these options from your command line.

### 9.2.1.10 E10010 Invalid --log Argument

#### Symptom

Value [%s] for [--log] is invalid. Must be selected from [debug], [info], [warning], [error], and [null].

#### Possible Cause

The [--log] argument is invalid.

## Solution

Valid values include [debug], [info], [warning], [error], and [null].

### 9.2.1.11 E10011 Invalid --input\_shape Argument

#### Symptom

Argument [%s] for [--input\_shape] is invalid. Shape values must be positive integers. The error value is: %s.

#### Possible Cause

In static shape scenarios, set the shape values of each input node in the [--input\_shape] to positive integers.

## Solution

1. In static shape scenarios, set the shape values in [--input\_shape] to positive integers.
2. In dynamic shape scenarios, add the related dynamic-input option in your command line, such as [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims].

### 9.2.1.12 E10012 Invalid --input\_shape Argument

#### Symptom

[--dynamic\_batch\_size] is included, but the dimension count of the dynamic-shape input configured in [--input\_shape] is less than 1.

#### Possible Cause

As [--dynamic\_batch\_size] is included, the dimension count of the dynamic-shape input configured in [--input\_shape] must not be 0.

## Solution

1. In static shape scenarios, remove the [--dynamic\_batch\_size] option from your command line.
2. In dynamic shape scenarios, set the corresponding axis of the dynamic-shape input in [--input\_shape] to -1.

### 9.2.1.13 E10013 Invalid Argument

#### Symptom

Value [%s] for [--%s] is out of range.

#### Possible Cause

The argument is out of range.

## Solution

Try again with a valid argument.

### 9.2.1.14 E10014 Invalid Argument

#### Symptom

Value [%s] for [--%s] is invalid.

#### Possible Cause

The argument is invalid.

## Solution

Try again with a valid argument. For details, see [ATC Instructions].

### 9.2.1.15 E10015 Invalid Argument

#### Symptom

Value [%s] for [--%s] is invalid.

#### Possible Cause

Try again with a valid argument.

## Solution

Try again with a valid argument. For details, see [ATC Instructions].

### 9.2.1.16 E10016 Invalid Argument, Node Not Found

#### Symptom

Input Op [%s] specified in [--%s] is not found in the model.

#### Possible Cause

The node specified in the argument does not exist in the model.

## Solution

Try again with a valid node name.

### 9.2.1.17 E10017 Invalid Argument, Node Not Found

#### Symptom

Input Op [%s] specified in [--%s] is invalid.

## Possible Cause

The input node specified in the argument is invalid.

## Solution

Try again with a valid input node name.

### 9.2.1.18 E10018 Invalid Dynamic Shape Argument

## Symptom

Value [%s] for shape [%s] is invalid. When [--dynamic\_batch\_size] is included, only batch size N can be -1 in [--input\_shape].

## Possible Cause

When [--dynamic\_batch\_size] is included, only batch size N can be -1 in the shape.

## Solution

Try again with a valid [--input\_shape] argument. Make sure that non-batch size axes are not -1.

### 9.2.1.19 E10019 Invalid --input\_shape Argument

## Symptom

When [--dynamic\_image\_size] is included, only the height and width axes can be -1 in in [--input\_shape].

## Possible Cause

When [--dynamic\_image\_size] is included, only the height and width axes can be -1 in the shape.

## Solution

Try again with a valid [--input\_shape] argument. Make sure that axes other than height and width are not -1.

### 9.2.1.20 E10020 Invalid --dynamic\_image\_size Argument

## Symptom

Value [%s] for [--dynamic\_image\_size] is invalid. Must be formatted as [imagesize1\_height,imagesize1\_width;imagesize2\_height,imagesize2\_width].

## Possible Cause

In the [--dynamic\_image\_size] argument, each profile must have two dimensions.

## Solution

Try again with a valid [--dynamic\_image\_size] argument. Make sure that each profile has two dimensions.

### 9.2.1.21 E10021 Invalid Argument

#### Symptom

Directory for [--%s] is too long. Keep the length within %s.

#### Possible Cause

The specified directory is too long.

#### Solution

Try again with a valid directory.

### 9.2.1.22 E10022 Invalid Argument

#### Symptom

Directory [%s] for [--%s] does not include the file name

#### Possible Cause

The specified directory does not include the file name.

#### Solution

Add the file name to the directory.

### 9.2.1.23 E10023 Invalid --singleop Argument

#### Symptom

Value [%s] for [--singleop] is invalid. Must be a proper directory.

#### Possible Cause

The [--singleop] argument is not a proper directory.

#### Solution

Try again with a valid argument.

### 9.2.1.24 E10024 Invalid --singleop Argument

#### Symptom

Failed to open file [%s] specified by the [--singleop] argument.

## Possible Cause

Failed to open the file specified by the [--singleop] argument.

## Solution

Check the owner group and permission settings and ensure that the user who runs the atc command has enough permission to open the file.

### 9.2.1.25 E10025 Invalid --singleop Argument

## Symptom

File [%s] specified by [--singleop] is not a valid JSON file. Reason: %s.

## Possible Cause

Failed to parse the file specified by the [--singleop] argument in JSON format.

## Solution

Check that the file is in valid JSON format.

### 9.2.1.26 E10026 Invalid --singleop Argument

## Symptom

Empty Op name in the file specified by the [--singleop] argument.

## Possible Cause

The file specified by the [--singleop] argument contains an unnamed Op.

## Solution

Check that no Op name is empty in the file.

### 9.2.1.27 E10027 Invalid --singleop Argument

## Symptom

[%s] and [%s] for tensor indexed [%s] of Op [%s] are invalid in the file specified by the [--singleop] argument.

## Possible Cause

The file specified by the [--singleop] argument contains invalid tensor attributes.

## Solution

Try again with valid tensor dtype and format.

### 9.2.1.28 E10029 Invalid --singleop Argument

#### Symptom

Attribute name of Op [%s] is empty in the file specified by the [--singleop] argument.

#### Possible Cause

The file specified by the [--singleop] argument contains an unnamed attribute.

#### Solution

Check that no Op attribute name is empty in the file.

### 9.2.1.29 E10030 Invalid --singleop Argument

#### Symptom

Attribute name [%s] of Op [%s] is invalid in the file specified by the [--singleop] argument.

#### Possible Cause

The file specified by the [--singleop] argument contains an empty attribute.

#### Solution

Check that no Op attribute value is empty in the file.

### 9.2.1.30 E10031 Invalid --input\_shape Argument

#### Symptom

[--dynamic\_batch\_size] is included, but none of the nodes specified in [--input\_shape] have a batch size equaling -1.

#### Possible Cause

As [--dynamic\_batch\_size] is included, ensure that at least one of the nodes specified in [--input\_shape] has a batch size equaling -1.

#### Solution

1. In static shape scenarios, remove the [--dynamic\_batch\_size] option from your command line.
2. In dynamic shape scenarios, set the corresponding axis of the dynamic-shape input in [--input\_shape] to -1.

### 9.2.1.31 E10034 Invalid --input\_fp16\_nodes Argument

#### Symptom

Nodes (for example, [%s]) connected to aipp must not be of type fp16. Check the [--insert\_op\_conf] and [--input\_fp16\_nodes] options in your atc command line.

#### Possible Cause

Do not set dtype to fp16 for a node connected to aipp.

#### Solution

1. To enable AIPP, remove the nodes connected to aipp from the [--input\_fp16\_nodes] argument.
2. If AIPP is not required, remove the [--insert\_op\_conf] option from your atc command line.

### 9.2.1.32 E10035 Invalid Dynamic Shape Argument

#### Symptom

The [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument has [%s] profiles, which is less than the minimum [%s].

#### Possible Cause

The number of configured profiles is less than the minimum.

#### Solution

Ensure that the number of profiles configured in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument is at least the minimum.

### 9.2.1.33 E10036 Invalid Dynamic Shape Argument

#### Symptom

[--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] has [%s] profiles, which exceeds the maximum [%s].

#### Possible Cause

The number of configured profiles exceeds the maximum.

#### Solution

Ensure that the number of profiles configured in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument does not exceed the maximum.



### 9.2.1.34 E10037 Invalid Dynamic Shape Argument

#### Symptom

The profiles configured in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument have inconsistent dimension counts. A profile has %s dimensions while another has %s dimensions.

#### Possible Cause

The configured profiles have inconsistent dimension counts.

#### Solution

Ensure that the profiles configured in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument have the same dimension count.

### 9.2.1.35 E10038 Invalid Dynamic Shape Argument

#### Symptom

Dimension size [%s] is invalid. Must be greater than 0.

#### Possible Cause

The shape values of each profile must be positive.

#### Solution

Set the shape values of each profile to positive in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument.

### 9.2.1.36 E10039 Invalid Dynamic Shape Argument

#### Symptom

The [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument have duplicate profiles.

#### Possible Cause

The configured profiles must be unique.

#### Solution

Check that the profiles configured in the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument are unique.

### 9.2.1.37 E10040 Invalid --input\_shape Argument

#### Symptom

As the [--dynamic\_batch\_size], [--dynamic\_image\_size], or [--dynamic\_dims] argument is included, the corresponding nodes specified in [--input\_shape] must have -1 axes.

#### Possible Cause

In dynamic shape scenarios, the corresponding shape value in the input node must be set to -1.

#### Solution

1. In static shape scenarios, remove the [--dynamic\_batch\_size] option from your command line.
2. In dynamic shape scenarios, set the corresponding axis of the dynamic-shape input in [--input\_shape] to -1.

### 9.2.1.38 E10041 Invalid --framework or --model Argument

#### Symptom

Failed to load the model from [%s]. Check the model file or the [--framework] argument.

#### Possible Cause

The model file is invalid.

#### Solution

1. Check that the model file is valid.
2. Check that the [--framework] argument matches the actual framework of the model file.

### 9.2.1.39 E10045 Invalid Dynamic Shape Argument

#### Symptom

The number of -1 axes overflow for data [%s] with shape [%s].

#### Possible Cause

[--dynamic\_dims] is included for more than one input, but the number of -1 axes in the [--input\_shape] argument does not match the dimension count per profile in [--dynamic\_dims].

## Solution

Ensure that the number of -1 axes in the [--input\_shape] argument matches the dimension count per profile in [--dynamic\_dims].

### 9.2.1.40 E10046 Invalid Dynamic Shape Argument

#### Symptom

The number of -1 axes of data [%s] with shape [%s] is greater than the dimension count per profile.

#### Possible Cause

The number of -1 axes in the [--input\_shape] argument is greater than the dimension count per profile in [--dynamic\_dims].

## Solution

Ensure that the total number of -1 axes in the [--input\_shape] argument is less than the dimension count per profile in [--dynamic\_dims].

### 9.2.1.41 E10047 Invalid Argument

#### Symptom

[--%s] and [--%s] are mutually exclusive.

#### Possible Cause

The options are mutually exclusive.

## Solution

Remove either of them and try again.

### 9.2.1.42 E10048 Invalid --input\_shape\_range Argument

#### Symptom

Value [%s] for [--input\_shape\_range] is invalid. Reason: %s. Must be formatted as [%s].

#### Possible Cause

Invalid --input\_shape\_range Argument

## Solution

Try again with a valid argument.

### 9.2.1.43 E10049 Invalid --input\_shape\_range Argument

#### Symptom

Dimension count [%s] configured in the [--input\_shape\_range] argument does not match dimension count [%s] of the node.

#### Possible Cause

The dimension count in the [--input\_shape\_range] argument does not match the dimension count of the node.

#### Solution

Set the dimension count in the [--input\_shape\_range] argument according to the dimension count of the node.

### 9.2.1.44 E10050 Invalid --input\_shape\_range Argument

#### Symptom

Current dimension size [%s] is out of range [%s-%s] specified by [--input\_shape\_range].

#### Possible Cause

The dimension size is out of the range specified by [--input\_shape\_range].

#### Solution

Set the dimension size according to [--input\_shape\_range].

### 9.2.1.45 E10052 Invalid AIPP Configuration

#### Symptom

AIPP configuration is invalid. Reason: %s.

#### Possible Cause

The AIPP configuration is invalid.

#### Solution

Try again with valid AIPP configuration.

### 9.2.1.46 E10410 Invalid Argument

#### Symptom

File [%s] does not exist.

## Possible Cause

The file specified by the [--keep\_dtype] or [--compress\_weight\_conf] argument does not exist.

## Solution

Try again with a valid file directory.

### 9.2.1.47 E11001 Caffe Model Data Error

## Symptom

[input\_dim] and [input\_shape] are mutually exclusive in [NetParameter] for Caffe model conversion.

## Possible Cause

--input\_dim] and --input\_shape] are mutually exclusive in NetParameter of the source Caffe model.

## Solution

Remove either of [--input\_dim] and [--input\_shape] from your atc command line.

### 9.2.1.48 E11002 Caffe Model Data Error

## Symptom

Caffe model has no input.

## Possible Cause

The source Caffe model has no input.

## Solution

Modify your Caffe model and try again.

### 9.2.1.49 E11003 Caffe Model Data Error

## Symptom

The number of [input\_dim] fields in the model is [%s], which is not 4x the input count [%s].

## Possible Cause

The number of [input\_dim] fields in the source Caffe model is not 4x the input count.

## Solution

Modify your Caffe model and try again.

### 9.2.1.50 E11004 Caffe Model Data Error

## Symptom

The number of input shapes is [%s], which does not match the number of inputs [%s].

## Possible Cause

The number of input shapes of the source Caffe model does not match the number of inputs.

## Solution

Modify your Caffe model and try again.

### 9.2.1.51 E11005 Invalid --input\_shape Argument

## Symptom

Shape is not defined by using [--input\_shape] for input [%s].

## Possible Cause

The shape of the input in the source Caffe model is not defined.

## Solution

Modify your Caffe model, or add the shape of the input to the [--input\_shape] argument in your atc command line.

### 9.2.1.52 E11008 Caffe Model Data Error

## Symptom

Optype DetectionOutput is unsupported. Modify the model file and use an explicit type, such as FSRDetectionOutput or SSDDetectionOutput.

## Possible Cause

The source Caffe model contains an unsupported optype, DetectionOutput.

## Solution

Modify your Caffe model and replace DetectionOutput operators with FSRDetectionOutput or SSDDetectionOutput.

### 9.2.1.53 E11009 Unsupported Caffe Operator

#### Symptom

No Caffe parser is registered for Op [%s, optype [%s]].

#### Possible Cause

No parser is registered for the optype in Caffe model conversion.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.1.54 E11012 Caffe Model Data Error

#### Symptom

Unknown bottom blob [%s] at layer [%s]. The bottom blob is indexed [%s].

#### Possible Cause

The bottom blob in the source Caffe model has no connected top blob.

#### Solution

Modify your Caffe model and try again.

### 9.2.1.55 E11014 Caffe Model Data Error

#### Symptom

Failed to find the top blob for layer [%s].

#### Possible Cause

The top blob has no corresponding node in the source Caffe model.

#### Solution

Modify your Caffe model and try again.

### 9.2.1.56 E11015 Caffe Model Data Error

#### Symptom

Failed to find the bottom blob for layer [%s].

#### Possible Cause

The bottom blob has no corresponding node in the source Caffe model.

## Solution

Modify your Caffe model and try again.

### 9.2.1.57 E11016 Invalid --out\_node Argument

#### Symptom

Failed to add Op [%s] to NetOutput. Op output index [%s] is not less than [%s].  
NetOutput input\_index [%s] is not less than [%s].

#### Possible Cause

The node index specified by the [--out\_node] argument is out of the range of outputs of the corresponding node in the source Caffe model.

## Solution

Try again with a valid [--out\_node] argument.

### 9.2.1.58 E11017 Invalid --out\_node Argument

#### Symptom

Failed to find node [%s] specified by [--out\_node].

#### Possible Cause

The node specified by the [--out\_node] argument does not exist in the source Caffe model.

## Solution

Try again with a valid [--out\_node] argument.

### 9.2.1.59 E11018 Caffe Model Data Error

#### Symptom

Op name string [%s] is invalid. Allowed characters include: letters, digits, hyphens (-), periods (.), underscores (\_), and slashes (/).

#### Possible Cause

The Op name string in the source Caffe model is invalid. Allowed characters include: letters, digits, hyphens (-), periods (.), underscores (\_), and slashes (/).

## Solution

Modify the Op name string and try again.



### 9.2.1.60 E11021 Caffe Model Data Error

#### Symptom

Model file [%s] contains [layers] structures, which have been deprecated in Caffe and unsupported by ATC. Replace [layers] with [layer].

#### Possible Cause

The Caffe model contains [layers] structures, which have been deprecated in Caffe and unsupported by ATC. Replace [layers] with [layer].

#### Solution

Try again with a valid Caffe model.

### 9.2.1.61 E11022 Caffe Model Data Error

#### Symptom

Invalid prototxt file. No [layer] structures are found in the model.

#### Possible Cause

No [layer] structures are found in the Caffe model.

#### Solution

Try again with a valid Caffe model.

### 9.2.1.62 E11023 Caffe Model Data Error

#### Symptom

Weight file [%s] contains [layers] structures, which have been deprecated in Caffe and unsupported by ATC. Replace [layers] with [layer].

#### Possible Cause

The Caffe weight file contains [layers] structures, which have been deprecated in Caffe and unsupported by ATC. Replace [layers] with [layer].

#### Solution

Try again with a valid Caffe weight file.

### 9.2.1.63 E11024 Caffe Model Data Error

#### Symptom

Invalid Caffe weight file. No [layer] structures are found in the weight file.

## Possible Cause

No [layer] structures are found in the Caffe weight file.

## Solution

Try again with a valid Caffe weight file.

### 9.2.1.64 E11027 Caffe Model Data Error

## Symptom

Op [%s, optype [%s]] in the Caffe model has an input node with shape size 0.

## Possible Cause

The source Caffe model has an input node with shape size 0.

## Solution

Try again with a valid Caffe model.

### 9.2.1.65 E11029 Caffe Model Data Error

## Symptom

Op [%s] exists in model file but not found in weight file.

## Possible Cause

The node exists in the Caffe model file but is not found in weight file.

## Solution

Try again with a valid Caffe model or weight file. Ensure that the two files match with each other.

### 9.2.1.66 E11032 Caffe File Error

## Symptom

Failed to parse message [%s]. The error field is %s. Reason: %s.

## Possible Cause

The Caffe .proto file is invalid.

## Solution

Try again with a valid .proto file.

### 9.2.1.67 E11033 The Caffe weight file is invalid.

#### Symptom

Failed to convert the weight file. Blob [%s] of size [%s] is invalid. Reason: %s.

#### Possible Cause

The blob size of the node in the Caffe weight file does not match the number of elements calculated based on its shape.

#### Solution

Try again with a valid Caffe model or weight file. Ensure that the two files match with each other.

### 9.2.1.68 E11035 Caffe Model Data Error

#### Symptom

The top size of data node [%s] is not 1 but [%s].

#### Possible Cause

The top size of the data node in the source Caffe model is not 1.

#### Solution

Try again with a valid Caffe model.

### 9.2.1.69 E11036 Caffe Model Data Error

#### Symptom

Data nodes have duplicate top blobs [%s].

#### Possible Cause

The data nodes in the source Caffe model have duplicate top blobs.

#### Solution

Try again with a valid Caffe model.

### 9.2.1.70 E11037 Caffe Model Data Error

#### Symptom

Op [%s] has zero outputs.

## Possible Cause

Nodes in the Caffe model must have at least one output.

## Solution

Try again with a valid Caffe model.

### 9.2.1.71 E12009 TensorFlow Model Data Error

## Symptom

Op [%s]'s input [%s] is not found in graph\_def.

## Possible Cause

The input name of the node is not found in the graph.

## Solution

Try again with a valid TensorFlow model.

### 9.2.1.72 E12010 TensorFlow Model Data Error

## Symptom

Model has no Placeholder or \_Arg node.

## Possible Cause

No node of type Placeholder or \_Arg is found in the model.

## Solution

Try again with a valid TensorFlow model.

### 9.2.1.73 E12013 TensorFlow Model Data Error

## Symptom

Failed to find a subgraph by the name [%s].

## Possible Cause

No subgraph .proto description is found based on the subgraph name.

## Solution

1. To use function subgraphs to convert a TensorFlow model, place the subgraph .proto description file in the same directory as the model file and name it [graph\_def\_library.pbtxt].

2. Run the [func2graph.py] script in the ATC installation directory to save the subgraphs to [graph\_def\_library.pbtxt].

#### 9.2.1.74 E12029 TensorFlow Model Data Error

##### Symptom

Failed to find the subgraph library. As the model contains function operators, run the [func2graph.py] script in the ATC installation directory to save the subgraphs to [graph\_def\_library.pbtxt].

##### Possible Cause

If the model to convert contains function subgraphs, the [graph\_def\_library.pbtxt] file is required.

##### Solution

1. To use function subgraphs to convert a TensorFlow model, place the subgraph .proto description file in the same directory as the model file and name it [graph\_def\_library.pbtxt].
2. Run the [func2graph.py] script in the ATC installation directory to save the subgraphs to [graph\_def\_library.pbtxt].

#### 9.2.1.75 E16001 ONNX Model Data Error

##### Symptom

Model has no [%s] node.

##### Possible Cause

The node is not found in the ONNX model.

##### Solution

Try again with a valid ONNX model.

#### 9.2.1.76 E16002 ONNX Model Data Error

##### Symptom

No ONNX parser is registered for optype [%s].

##### Possible Cause

No parser is registered for the optype in ONNX model conversion.

##### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.1.77 E16004 ONNX Model Data Error

#### Symptom

ONNX model has no graph.

#### Possible Cause

The ONNX model has no graph.

#### Solution

Try again with a valid ONNX model.

### 9.2.1.78 E16005 ONNX Model Data Error

#### Symptom

The model has [%s] [--domain\_version] fields, but only one is allowed.

#### Possible Cause

The source ONNX model has more than one [--domain\_version] argument.

#### Solution

Try again with a valid ONNX model.

### 9.2.1.79 E20100 Invalid GE Initialization Argument

#### Symptom

Value [%s] is invalid for option [ge.opSelectImplmode]. Must be either [high\_precision] or [high\_performance].

#### Possible Cause

The operator implementation mode specified for GE initialization is invalid. Must be either [high\_precision] or [high\_performance].

#### Solution

Try again with a valid [opSelectImplmode] argument, which must match the [--op\_select\_implmode] argument in the ATC command line. Alternatively, the error is due to a GE translation failure.

### 9.2.1.80 E20101 Invalid GE Initialization Argument

#### Symptom

Value [%s] is invalid for option [%s].

## Possible Cause

See the error message for details.

## Solution

Try again with a valid argument.

### 9.2.1.81 E20102 Invalid GE Initialization Argument

#### Symptom

Value [%s] is invalid for option [ge.engineType]. Must be either [AiCore] or [VectorCore].

#### Possible Cause

The engine type argument for GE initialization is invalid. Only [AiCore] or [VectorCore] is supported.

#### Solution

Contact Huawei technical support to check the engine type specified for GE initialization.

### 9.2.1.82 E20103 Invalid Platform Initialization Argument

#### Symptom

Value [%s] for option [ge.aicoreNum] is out of range (0, %s).

#### Possible Cause

The number of AI Cores configured for platform initialization is out of the valid range. This is a Huawei internal parameter and is user-irrelevant.

#### Solution

Contact Huawei technical support to check the value of [aicoreNum] in the [platform.ini] file.

### 9.2.1.83 W21000 Empty Path

#### Symptom

Path [%s] is empty. Reason: %s.

#### Possible Cause

This error does not terminate the process. The path is not specified or the specified path is empty.

## Solution

Try again with a valid path.

### 9.2.1.84 E21001 Failure to Open File

#### Symptom

Failed to open file [%s]. Reason: %s.

#### Possible Cause

The file does not exist or cannot be opened.

#### Solution

Check the error message and error log.

### 9.2.1.85 E21002 Failure to Read File

#### Symptom

Failed to read file [%s]. Reason: %s.

#### Possible Cause

The file path is empty or the file does not exist.

#### Solution

Check the error message and error log.

### 9.2.1.86 E21003 Empty Path

#### Symptom

The file path of node [%s] is empty. Reason: %s.

#### Possible Cause

The compilation result of this node is empty, possibly due to a compilation failure.

#### Solution

1. Check the error message to see if a compilation failure has occurred.
2. If no compilation failure has occurred, contact Huawei technical support for fault locating.



### 9.2.1.87 W40001 Invalid Path

#### Symptom

Path [%s] for [%s] is invalid. Reason: %s.

#### Possible Cause

The path is empty or does not exist.

#### Solution

Try again with a valid path.

### 9.2.1.88 W40002 Failure to Create Directory

#### Symptom

Failed to create directory [%s]. Reason: %s.

#### Possible Cause

You do not have the permission on the directory or the directory name is invalid.

#### Solution

Modify the permission or directory name and try again.

### 9.2.1.89 E40000 Failure to Import te.platform.log\_util

#### Symptom

Failed to import te.platform.log\_util.

#### Possible Cause

A component installation is missing or its installed version does not match.  
Alternatively, the specified Python path is incorrect.

#### Solution

1. Check that all required components are properly installed, Python is properly installed, and the specified Python path matches the Python installation directory.
2. Check the configured environment variables such as [PYTHONPATH] and [ASCEND\_OPP\_PATH].

### 9.2.1.90 E40001 Failure to Import Python Module

#### Symptom

Failed to import the Python module: [%s]

## Possible Cause

A component installation is missing or its installed version does not match.  
Alternatively, the specified Python path is incorrect.

## Solution

Check that all required components are properly installed and the specified Python path matches the Python installation directory.

### 9.2.1.91 E40002 Failure to Call Python Function

## Symptom

Failed to call function [%s] with arguments [%s].

## Possible Cause

An invalid argument is passed or the Python function has a bug.

## Solution

This error is caused by an internal problem. Check the code of the Python function call.

### 9.2.1.92 E40003 Failure to Import Auto Tiling Manager of Python Module

## Symptom

Failed to import Auto Tiling Manager of Python module: [%s]

## Possible Cause

A component installation is missing or its installed version does not match.  
Alternatively, the specified [PYTHONPATH] is invalid.

## Solution

Check that all required components are properly installed and the specified [PYTHONPATH] matches the Python installation directory.

### 9.2.1.93 E40004 Failure to Import RL Tiling Manager of Python Module

## Symptom

Failed to import RL Tiling Manager of Python module: [%s]

## Possible Cause

A component installation is missing or its installed version does not match.  
Alternatively, the specified [PYTHONPATH] is invalid.

## Solution

Check that the ATC and FwkACCLib components are properly installed and the specified [PYTHONPATH] matches the Python installation directory.

### 9.2.1.94 E40008 Failure to Import Python Module

#### Symptom

Failed to import the Python module from [%s].

#### Possible Cause

A component installation is missing or its installed version does not match. Alternatively, the specified [PYTHONPATH] is invalid. Check that Python is properly installed, and the specified Python path matches the Python installation directory.

## Solution

Check that the ATC and FwkACCLib components are properly installed and the specified [PYTHONPATH] matches the Python installation directory.

### 9.2.1.95 E40010 Invalid Auto Tune Mode

#### Symptom

Auto Tune mode [%s] is invalid.

#### Possible Cause

The specified Auto Tune mode is invalid.

## Solution

Try again with a valid argument.

### 9.2.1.96 E40011 Invalid Argument

#### Symptom

Argument [%s] is invalid. Reason: %s.

#### Possible Cause

An invalid argument is passed.

## Solution

Try again with a valid argument. If the error persists, check the argument verification logic in the code.

### 9.2.1.97 EE1000 Invalid SoC Version

#### Symptom

Value %s is invalid for the SoC version. Valid values are: %s

#### Possible Cause

The SoC version argument is invalid.

#### Solution

Try again with a valid argument.

### 9.2.1.98 E76002 ONNX Model Data Error

#### Symptom

No ONNX parser is registered for optype [%s].

#### Possible Cause

No parser is registered for the optype in ONNX model conversion.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

## 9.2.2 Resource Errors

### 9.2.2.1 E10044 Insufficient Memory

#### Symptom

The available memory is [%s KB], which is less than the required minimum [%s KB].

#### Possible Cause

Auto Tune requires at least 2 GB available memory.

#### Solution

Stop unnecessary processes and ensure that at least 2 GB system memory is available. Alternatively, change a machine with 2 GB or more of available memory.

### 9.2.2.2 E19022 Insufficient Memory

#### Symptom

Model %s requires [%s] memory, which exceeds system limit [%s].

#### Possible Cause

The memory size required by the model exceeds the system limit.

#### Solution

1. Retrain the model with a reduced batch size.
2. Reduce the model size.

### 9.2.2.3 E19023 Too Large OM Model

#### Symptom

Model %s has size [%s], which exceeds system limit [%s].

#### Possible Cause

The generated OM model is too large and therefore cannot be dumped to the disk.

#### Solution

Try again with reduced model size.

## 9.2.3 Invalid Argument

### 9.2.3.1 E10051 Invalid --job\_id Argument

#### Symptom

Value [%s] for [--job\_id] is too long, exceeding the allowed maximum [%s].

#### Possible Cause

The length of the [--job\_id] argument is too long.

#### Solution

Try again with a valid argument.

### 9.2.3.2 E10401 Invalid Operator Input Count

#### Symptom

The operator takes [%s] input(s) according to specification, but [%s] input(s) are configured.

#### Possible Cause

The number of inputs configured for operator execution does not match the operator specification.

#### Solution

Try again with a valid number of inputs.

### 9.2.3.3 E10402 Invalid Input Buffer Allocation for Operator Execution

#### Symptom

Input indexed [%s] requires a %s buffer, but %s (aligned) are allocated.

#### Possible Cause

The input buffer allocation for operator execution does not match that required.

#### Solution

Try again with a valid buffer allocation.

### 9.2.3.4 E10403 Invalid Operator Output Count

#### Symptom

The operator returns [%s] output(s) according to specification, but [%s] output(s) are configured.

#### Possible Cause

The number of outputs configured for operator execution does not match the operator specification.

#### Solution

Try again with a valid number of outputs.

### 9.2.3.5 E10404 Invalid Output Buffer Allocation for Operator Execution

#### Symptom

Output indexed [%s] requires a %s buffer, but %s (aligned) are allocated.

## Possible Cause

The output buffer allocation for operator execution does not match that required.

## Solution

Try again with a valid buffer allocation.

### 9.2.3.6 E10405 Inconsistent Input Buffer Count and Input Tensor Count for Operator Execution

## Symptom

The number of input buffers is [%s], which does not match the number of input tensors [%s].

## Possible Cause

The number of input buffers for operator execution does not match the number of input tensors.

## Solution

Ensure that the number of input buffers for operator execution matches the number of input tensors.

### 9.2.3.7 E10406 Inconsistent Output Buffer Count and Output Tensor Count for Operator Execution

## Symptom

The number of output buffers is [%s], which does not match the number of output tensors [%s].

## Possible Cause

The number of output buffers for operator execution does not match the number of output tensors.

## Solution

Ensure that the number of output buffers for operator execution matches the number of output tensors.

### 9.2.3.8 E14001 Invalid Argument for Operator Compilation

## Symptom

Argument [%s] for Op [%s, optype [%s]], is invalid. Reason: %s.

## Possible Cause

The argument for operator compilation is invalid.

## Solution

Try again with a valid argument.

### 9.2.3.9 E19009 Operator Name Conflict

## Symptom

Op [%s] has a name conflict in the graph.

## Possible Cause

The graph contains operators with duplicate names.

## Solution

Ensure that the operators in the graph have unique names.

### 9.2.3.10 E19014 Operator Data Verification Failure

## Symptom

Value [%s] for Op [%s] is invalid. Reason: %s.

## Possible Cause

Verification of operator data fails.

## Solution

Fix the error according to the error message.

### 9.2.3.11 E19018 Failure to Parse File

## Symptom

Failed to parse file [%s] through [google::protobuf::TextFormat::Parse]. Try again with a valid protobuf file.

## Possible Cause

Failed to parse the file using the protobuf library function. The file may not be in valid protobuf format.

## Solution

Try again with a valid file.



### 9.2.3.12 E19025 Input Tensor Error

#### Symptom

Input tensor is invalid. Reason: %s.

#### Possible Cause

The input tensor does not match the graph is invalid, resulting in an execution failure of the training or inference job.

#### Solution

Fix the error according to the error message.

## 9.2.4 System Support Errors

### 9.2.4.1 E10501 Unsupported Operator

#### Symptom

IR for Op [%s, optype [%s]], is not registered.

#### Possible Cause

IR for the operator type is not registered.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.4.2 E13002 Unsupported Operator

#### Symptom

Optype [%s] of Ops kernel [%s] is unsupported. Reason: %s.

#### Possible Cause

The operator type is unsupported in the operator information library due to specification mismatch.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.4.3 E13003 Unsupported Operator

#### Symptom

No supported Ops kernel and engine are found for [%s], optype [%s].

#### Possible Cause

The operator is not supported by the system. Therefore, no hit is found in any operator information library.

#### Solution

1. Check that the OPP component is installed properly.
2. Submit an issue to request for the support of this operator type.

### 9.2.4.4 E19010 Unsupported Operator

#### Symptom

No parser is registered for Op [%s, optype [%s]].

#### Possible Cause

No parser is registered for the operator type.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>.

### 9.2.4.5 EZ0501 Unsupported Operator

#### Symptom

IR for Op [%s, optype [%s]], is not registered.

#### Possible Cause

IR for the operator type is not registered.

#### Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.4.6 EZ3002 Unsupported Operator

#### Symptom

Optype [%s] of Ops kernel [%s] is unsupported. Reason: %s.

## Possible Cause

The operator type is unsupported in the operator information library due to specification mismatch.

## Solution

Submit an issue to request for support at <https://gitee.com/ascend>, or remove this type of operators from your model.

### 9.2.4.7 EZ3003 Unsupported Operator

## Symptom

No supported Ops kernel and engine are found for [%s], optype [%s].,

## Possible Cause

The operator is not supported by the system. Therefore, no hit is found in any operator information library.,

## Solution

1. Check that the OPP component is installed properly.
2. Submit an issue to request for the support of this operator type.

### 9.2.4.8 EZ9010 Unsupported Operator

## Symptom

No parser is registered for Op [%s, optype [%s]].

## Possible Cause

No parser is registered for the operator type.

## Solution

Submit an issue to request for support at <https://gitee.com/ascend>.

## 9.2.5 Configuration Errors

### 9.2.5.1 E12004 Operator Prototype Registration Error

## Symptom

Failed to register the prototype of Op [%s]. If input index is less than 0, then input index [-%s] (the abs value) must be less than the input count [%s]

## Possible Cause

To optimize an input to an attribute during operator registration, if the input index is less than 0, the absolute value of the index must be less than the actual number of inputs of the operator.

## Solution

Try again with a valid operator prototype.

### 9.2.5.2 E19024 Invalid Environment Variable

## Symptom

Value [%s] for environment variable [%s] is invalid when %s.

## Possible Cause

The configured environment variable is invalid.

## Solution

Fix the error according to the error message.

## 9.2.6 File Errors

### 9.2.6.1 E19000 Invalid Directory

## Symptom

Path[%s] is empty. Reason: %s.

## Possible Cause

The file does not exist.

## Solution

Try again with a valid directory.

### 9.2.6.2 E19001 Failure to Open File

## Symptom

Failed to open file[%s]. Reason: %s.

## Possible Cause

Failed to open the file.

## Solution

Fix the error according to the error message.

### 9.2.6.3 E19002 Too Long File Directory

#### Symptom

Directory [%s] is too long. Keep the length within [%s] characters.

#### Possible Cause

The file directory is too long.

## Solution

Try again with a valid file directory.

### 9.2.6.4 E19003 Failure to Read File

#### Symptom

Failed to read file [%s]. Reason: %s.

#### Possible Cause

Failed to read the file.

## Solution

Fix the error according to the error message.

### 9.2.6.5 E19004 Failure to Write File

#### Symptom

Failed to write file [%s]. Reason: %s.

#### Possible Cause

Failed to write the file.

## Solution

Fix the error according to the error message.

### 9.2.6.6 E19005 Failure to Parse File

#### Symptom

Failed to parse file [%s]. Try again with a valid protobuf file or check the version of your protobuf installation.

## Possible Cause

Failed to parse the file.

## Solution

Check that a matched protobuf version is installed and try again with a valid file.

### 9.2.6.7 E19015 File size invalid

## Symptom

File[%s] size %s is out of valid range[0, %s].

## Possible Cause

File size is not valid.

## Solution

Try again with a valid file.

## 9.2.7 Invalid Path Name

### 9.2.7.1 E19026 Input Path Name Error

## Symptom

Input path name [%s] is invalid. Reason: %s

## Possible Cause

The input path does not exist or is invalid.

## Solution

Try again with a valid path.

## 9.2.8 Minor Error

### 9.2.8.1 W11001 Operator Missing High-Priority Performance

## Symptom

Op [%s] does not hit the high-priority operator information library, which might result in compromised performance.

## Possible Cause

The operator does not hit the high-priority operator information library, which might result in compromised performance.

## Solution

Submit an issue to request for support at <https://gitee.com/ascend>.

## 9.2.9 ACL API Errors

### 9.2.9.1 EH0001 Invalid Argument

#### Symptom

Value [%s] for [%s] is invalid. Reason: %s.

#### Possible Cause

The argument is invalid.

#### Solution

Try again with a valid argument.

### 9.2.9.2 EH0002 Null Pointer

#### Symptom

Argument [%s] must not be null.

#### Possible Cause

A null pointer is passed to the parameter.

#### Solution

Check that memory is properly allocated to the pointer and try again with a valid pointer.

### 9.2.9.3 EH0003 Invalid Path

#### Symptom

Path [%s] is invalid. Reason: %s.

#### Possible Cause

The input path does not exist or is invalid. Alternatively, you do not have enough permission to access the path.

## Solution

Try again with a valid path.

### 9.2.9.4 EH0004 Invalid File

## Symptom

File [%s] is invalid. Reason: %s.

## Possible Cause

The file is invalid.

## Solution

Try again with a valid file.

### 9.2.9.5 EH0005 Invalid AIPP Argument

## Symptom

AIPP argument [%s] is invalid. Reason: %s.

## Possible Cause

The AIPP argument is invalid.

## Solution

Try again with a valid AIPP argument.

### 9.2.9.6 EH0006 Feature Unsupported

## Symptom

%s is not supported. Reason: %s.

## Possible Cause

The feature is not supported.

## Solution

Check if parameters of unsupported features are used according to your environment setup.

## 9.2.10 Environment Errors



### 9.2.10.1 E30000 Invalid Configuration File

#### Symptom

System error occurred. Failed to load AI CPU Op library information from file [%s].  
Reason: %s.

#### Possible Cause

The configuration file does not exist or the file is invalid.

#### Solution

Reinstall the FwkACCLib or ACLlib component or edit the file.

### 9.2.10.2 E30001 Invalid Configuration File

#### Symptom

System error occurred. Failed to load AI CPU kernel info from JSON file [%s].  
Reason: %s.

#### Possible Cause

The configuration file does not exist or the file is invalid.

#### Solution

Reinstall the OPP component or edit the file.

### 9.2.10.3 E30002 Invalid Configuration File

#### Symptom

System error occurred. Failed to load IR configuration file [%s]. Reason: %s.

#### Possible Cause

The configuration file does not exist or the file is invalid.

#### Solution

Reinstall the FwkACCLib or ACLlib component or edit the file.

### 9.2.10.4 EE2000 Device Error

#### Symptom

Device [%s] is not running properly.

## Possible Cause

Failed to connect to the device, possibly due to a bus error or device error.

## Solution

Send the log to Huawei technical support for fault locating.

### 9.2.10.5 EE2001 Failure to Allocate Memory

## Symptom

Failed to allocate memory for [%s] due to insufficient memory. The requested allocation is %s.

## Possible Cause

System memory is insufficient for the network.

## Solution

1. Free the system memory and try again.
2. If no more system memory can be freed, reduce the network size and try again.

### 9.2.10.6 EE3001 Driver Failure

## Symptom

The process has lost the connection between the host and device.

## Possible Cause

The execution of a particular operator times out or the host and device connection is unstable.

## Solution

Fix the timeout operator or reconnect the host and device, and then restart the app.

### 9.2.10.7 EI0001 Invalid Environment Variable Configuration

## Symptom

Environment variable [%s] is invalid. Tips: %s.

## Possible Cause

The environment variable configuration is invalid.

## Solution

Try again with valid environment variable configuration.

### 9.2.10.8 EI0002 SO File Not Found

#### Symptom

SO file [%s] is not found.

#### Possible Cause

The file does not exist or is damaged.

## Solution

Try again with a valid FwkACLib installation.

### 9.2.10.9 EI0004 Invalid Ranktable Configuration

#### Symptom

Invalid ranktable, with rankID [%s] and local devID [%s]. Check that ranktable [%s] is valid and the environment setup matches the ranktable.

#### Possible Cause

The cluster configuration in the ranktable file does not match that of the actual operating environment, or the configuration format is invalid

## Solution

Try again with a valid cluster configuration in the ranktable file. Ensure that the configuration matches the operating environment.

### 9.2.10.10 EI0007 Allocation Failure

#### Symptom

Failed to allocate resource [%s], with info [%s].

#### Possible Cause

Failed to allocate memory or notify due to resource insufficiency.

## Solution

Clean up unnecessary allocations.

## 9.2.11 Operator Compilation Errors

### 9.2.11.1 E20001 Failure to Precompile Op

#### Symptom

Failed to precompile Op [%s, optype [%s]] of graph\_id [%s].

#### Possible Cause

The operator has an invalid argument.

#### Solution

See the error message for details, and then check the Python stack where the error log is reported.

### 9.2.11.2 E20003 Failure to Compile Op

#### Symptom

Failed to compile Op [%s, optype [%s]] of graph\_id [%s].

#### Possible Cause

The operator has an invalid argument.

#### Solution

See the error message for details, and then check the Python stack where the error log is reported.

### 9.2.11.3 E20004 Failure to Compile Op

#### Symptom

Failed to compile Op [%s, optype [%s]] of graph\_id [%s] in thread [%s] of task [%s].

#### Possible Cause

The operator has an invalid argument.

#### Solution

See the error message for details, and then check the Python stack where the error log is reported.

### 9.2.11.4 E40009 Failure to Compile Op

#### Symptom

Failed to compile Op [%s, oppath [%s], optype [%s]] of taskID [%s].

## Possible Cause

See the error message for details.

## Solution

See the error message for details, and then check the Python stack where the error log is reported.

## 9.2.12 Invalid Operator Arguments

### 9.2.12.1 E20002 Failure to Precompile Op

#### Symptom

Failed to precompile Op [%s, optype [%s]] of graph\_id [%s] in thread [%s] of task [%s].

#### Possible Cause

The operator has an invalid argument.

#### Solution

See the error message for details, and then check the Python stack where the error log is reported.

### 9.2.12.2 E20006 Failure to Calculate Data Edge Size

#### Symptom

Failed to calculate the tensor size of Op [%s, optype %s].

#### Possible Cause

Failed to calculate the data edge size, possibly due to an invalid shape or format.

#### Solution

Inspect the error log, perform a partial dump by using [npucollect.sh](#) or the [export DUMP\_GE\_GRAPH=2] command, and then send the partial dump to Huawei technical support for fault locating.

### 9.2.12.3 E40005 Failure to Execute Function check\_supported

#### Symptom

Failed to execute the [check\_supported] function of Op [%s, oppath [%s], optype [%s]].

## Possible Cause

An invalid argument is passed or the Python function [check\_supported] has a bug. This error is most likely caused by an internal problem.

## Solution

This error is caused by an internal problem. Contact Huawei technical support to check the Python function call in the code and the implementation of the [check\_supported] function.

### 9.2.12.4 E40006 Failure to Obtain Op Format

## Symptom

Failed to obtain Op [%s]'s format. Reason: %s.

## Possible Cause

An invalid argument is passed or the Python function has a bug.

## Solution

This error is caused by an internal problem. Contact Huawei technical support to check the Python function call in the code and the implementation of the [op\_select\_format] function.

### 9.2.12.5 E40007 Failure to Precompile Op

## Symptom

Failed to precompile Op [%s, oppath [%s], optype [%s]].

## Possible Cause

See the error message for details.

## Solution

See the error message for details, and then check the Python stack where the error log is reported.

### 9.2.12.6 EI0003 Invalid Collective Communication Op Argument

## Symptom

In [%s], value [%s] of parameter [%s] is invalid. Tips: %s.

## Possible Cause

The collective communication operator has an invalid argument.

## Solution

Try again with a valid argument.

### 9.2.12.7 EI0005 Inconsistent Collective Communication Arguments Between Ranks

#### Symptom

The arguments for collective communication are inconsistent between ranks: tag [%s], parameter [%s], local [%s], remote [%s]

#### Possible Cause

The arguments for collective communication are inconsistent between ranks.

#### Solution

Try again with valid arguments. Ensure that the arguments are consistent between ranks.

### 9.2.12.8 EI0006 Invalid Collective Communication Allocation

#### Symptom

[%s] failed with info [%s]. Please check the passed [%s] resource.

#### Possible Cause

The memory or notify allocation passed to collective communication is invalid.

#### Solution

Try again with a valid memory or notify allocation.

## 9.2.13 Fusion Pass Errors

### 9.2.13.1 E20007 Failure to Execute Fusion Pass

#### Symptom

Failed to run the following graph fusion pass: [%s]. The pass type is [%s].

#### Possible Cause

The fusion pass code has a bug or the source graph does not meet the fusion pass requirements.

## Solution

1. If the pass code is user-defined, check the error message and the verification logic.
2. If the pass code is not user-defined, perform a complete or partial dump by using [npucollect.sh](#) and then send the dump to Huawei technical support for fault locating.

### 9.2.13.2 E20008 Failure to Execute Fusion Pass

#### Symptom

Failed to run pass [%s]. Reason: %s.

#### Possible Cause

The fusion pass code has a bug or the source graph does not meet the fusion pass requirements.

#### Solution

Check the pass code.

### 9.2.13.3 E20009 Failure to Execute Fusion Pass

#### Symptom

Failed to do topological sorting after graph fusion, graph is cyclic, graph name: %s, pass name: %s.

#### Possible Cause

The fusion pass code has a bug.

#### Solution

Check the pass code.

## 9.2.14 Task Generation Errors

### 9.2.14.1 E20005 Failure to Generate Task

#### Symptom

Failed to generate a task for Op [%s, optype [%s]] of graph\_id [%s].

#### Possible Cause

A compilation argument is invalid, the operator is not compiled, or the GE memory allocation is invalid.



## Solution

Check the error message and error log.

## 9.2.15 TBE Compiler Errors

### 9.2.15.1 EB0000 Invalid IR

#### Symptom

Failed to compile the operator. Reason:%s, %s.

#### Possible Cause

The IR is invalid.

#### Solution

Check that the Compute and Schedule APIs are used correctly. For details, see the TBE Custom Operator Developer Guide. If the error persists, contact Huawei technical support.

## 9.2.16 Task Schedule Execution Errors

### 9.2.16.1 EE4001 Model Execution Error

#### Symptom

This stream has been bound to a model. Repeated binding is not allowed. %s

#### Possible Cause

This stream has been bound to a model. Repeated binding is not allowed.

#### Solution

Remove redundant bindings from your code.

### 9.2.16.2 EE4002 Model Execution Error

#### Symptom

Failed to bind the stream to the model. %s

#### Possible Cause

The model ID does not match the input arguments or the specific stream is busy.

## Solution

See the error message for details and modify the rtModelBindStream arguments.

### 9.2.16.3 EE4003 Profiling Execution Error

#### Symptom

Failed to profile the HWTS log, as online profiling is in process. Please stop online profiling first. %s

#### Possible Cause

Failed to profile the HWTS log, as online profiling is in process.

#### Solution

Stop online profiling and try again.

### 9.2.16.4 EE4004 Profiling Execution Error

#### Symptom

Failed to enable profiling. %s

#### Possible Cause

Profiling is enabled. See the error message for details.

#### Solution

Disable profiling and try again.

### 9.2.16.5 EE5001 Task Schedule Resource Error

#### Symptom

Insufficient HWTS submit queue(SQ) resource. %s

#### Possible Cause

The HWTS SQ resource are not sufficient to launch any new tasks.

#### Solution

Wait until sufficient HWTS SQ resource is available.

### 9.2.16.6 EE5002 Task Schedule Resource Error

#### Symptom

Insufficient task schedule memory. %s

## Possible Cause

The task schedule memory are not sufficient to allocate new memory.

## Solution

Reduce the number of tasks of your model and try again.

## 9.2.17 Default Error

### 9.2.17.1 E19999 System Terminated

## Symptom

Unknown error occurred. Please check the log.

## Possible Cause

System terminated abnormally without valid error messages.

## Solution

In this scenario, use [npucollect.sh](#) to collect the logs generated when the fault occurs and locate the fault based on the logs.

## 9.3 一键收集故障信息

### 工具介绍

为提高系统故障维测效率，提供一键式故障信息收集脚本工具：一次性收集进程故障现场信息、为故障定位效率提升提供有效输入。工具可以收集以下信息：

- Host侧CANN软件栈日志
- Host侧驱动日志（需root权限）
- Host coredump文件
- 黑匣子、Device侧日志、Stackcore文件（需root权限）
- 训练打屏日志
- 包安装日志（需包安装账号与用例执行账号一致才可收集）
- GE dump图
- TF Adapter dump图
- 算子编译.o文件
- 机器环境信息，包括执行用例前机器的内存状态、硬盘状态、进程状态，机器的操作系统信息，内核版本信息。

## 说明

本工具会使用环境变量NPU\_COLLECT\_PATH控制维测信息的落盘路径，如dump图及算子.o文件的生成路径。

因版本不同，该工具收集的信息范围可能变化。

## 使用约束

1. 不支持原有执行脚本内部直接后台执行的方式。  
如：原有用例通过命令行执行 `sh cmd.sh`来拉起用例，而`cmd.sh`的实现里执行 `python3 test.py &`，用后台的方式执行，此种用例由于无法感知结束点，暂不支持使用。
2. 相同用户、相同时间段内，同机器同时作业时，收集到的数据会有交叉。
3. 非root用户，获取到的数据范围会受限，具体限制参见[工具介绍](#)中权限要求。
4. core文件的收集会对硬盘空间有较大要求，建议非coredump场景关闭core文件的收集。关闭方式如下：  
在`npucollect.sh`脚本开头的`modules`数组里，删除`core`的模块名，即不包括`core`。  
示例：`modules=(ge log ops environment)`
5. 云场景不支持一键式工具收集故障信息。

## 工具使用

请从码云tools仓（<https://gitee.com/ascend/tools/tree/master/npucollector>）获取脚本工具，然后将脚本工具存放到发生故障的环境上，例如存放到`/home/npucollector`目录下，接着在[原用例执行目录](#)下执行故障信息收集命令，命令行格式如下：

```
bash /home/npucollector/npucollect.sh "sh ../app_run.sh" /home/npucollector/target.tar.gz
```

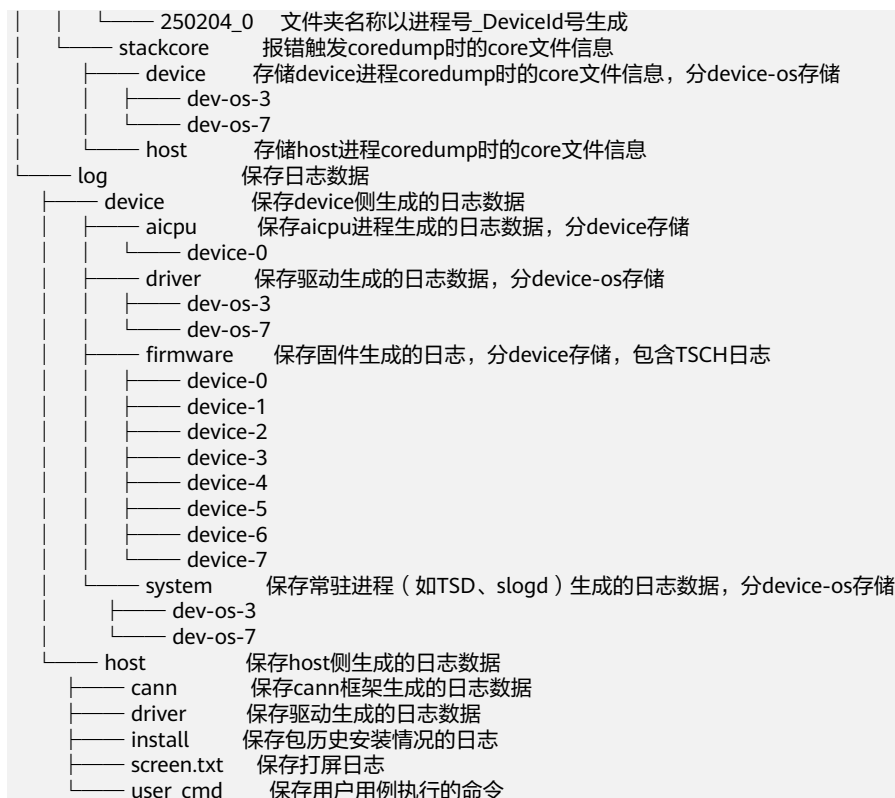
## 说明

命令行中，第一个参数字段为发生故障时执行的任务，填写完整的命令；第二个参数字段为收集的目标压缩文件名称，目前必须以`.tar.gz`结尾。

更多更新说明介绍，请参考码云[Readme.md](#)。

执行完命令后，生成本次收集的故障维测所需信息，`*.tar.gz`文件解压后内容如下：

extra-info	保存非日志数据
bbox	保存device侧的黑匣子信息，分device存储
device-0	
device-1	
device-2	
device-3	
device-4	
device-5	
device-6	
device-7	
environment	保存机器环境信息
cache_memory_status	保存用例执行前，机器内存占用情况
disk_memory_status	保存用例执行前，机器硬盘占用情况
kernel_info	保存机器内核版本信息
os_info	保存机器操作系统信息
process_status	保存用例执行前，机器进程使用情况
graph	保存用例执行时，生成的dump图信息，包含GE与TF Adapter的dump图
250204_0	文件夹名称以进程号_DeviceId号生成
ops	保存用例执行时，生成的算子.o，.json文件信息



## 进阶功能-集成自定义数据收集

在某些场景下，用户可能会有额外数据一起收集的诉求，一键式收集脚本支持自定义数据收集模块的集成。

使用方式如下：

**步骤1** 复制一份ops.sh文件，命名成自己模块的名字，如plugin.sh。

**步骤2** 在复制的文件中，自定义实现pre\_process、running\_process\_once、post\_process方法。

- pre\_process：执行用例任务之前进行的操作。
- running\_process\_once：执行用例任务过程中，循环进行的操作（对循环周期有诉求可以修改running\_process方法里的sleep 时长）。
- post\_process：执行用例任务结束后进行的操作。

脚本内容示例：

```
#!/bin/bash
ops_path=/extra-info/ops      // 此处可以定义脚本全局可以用的到的常量

pre_process()                 // 此函数，用例脚本执行前会被调用一次
{
    base_path=$1              // $1变量里存储的是用户指定的数据收集存储的根目录，自定义的数据收集可以基
    于此目录选择相对路径存储
    mkdir -p $base_path$ops_path // 此处可以实现用户自定义的预处理行为
}

running_process_once()        // 此函数，用例脚本执行过程中会被循环调用
{
    base_path=$1              // $1变量里存储的是用户指定的数据收集存储的根目录，自定义的数据收集可以基
    于此目录选择相对路径存储
    return                    // 此处可以实现用户自定义的，用例执行中的数据收集行为
```

```
}  
  
running_process()          // 此函数，用例脚本执行前会触发后台执行，用于循环调用running_process_once  
函数  
{  
  while true  
  do  
    running_process_once $1  
    sleep 60                // 循环周期可在这里控制  
  done  
}  
  
post_process()             // 此函数，用例脚本执行结束后会被调用一次  
{  
  base_path=$1             // $1变量里存储的是用户指定的数据收集存储的根目录，自定义的数据收集可以基  
于此目录选择相对路径存储  
  return                   // 此处可以实现用户自定义的后处理行为  
}  
  
$1 $2                      // 外部调用时，用于拉起不同函数，保留即可，不用动
```

**步骤3** 在npucollect.sh脚本开头的modules数组里，增加自定义的sh脚本模块名称。

示例：modules=(core ge log ops environment plugin)

----结束

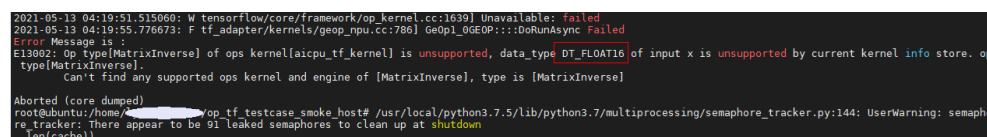
## 9.4 开启 Cast 算子自动插入特性

### 简介

模型编译时，若遇到AI CPU算子不支持某种数据类型导致编译失败的场景，可通过启用Cast算子自动插入特性快速将输入转换为算子支持的数据类型，从而实现网络的快速打通。

如图9-1，表示MatrixInverse算子的输入x不支持float16的数据类型。

图 9-1 报错示例



```
2021-05-13 04:19:51.515060: W tensorflow/core/framework/op_kernel.cc:1639] Unavailable: failed  
2021-05-13 04:19:55.776673: F tf_adapter/kernels/geop_npu.cc:786] GeOp1_0GEOP::DoRunAsync failed  
Error Message is :  
E13002: Op type[MatrixInverse] of ops kernel[aicpu_tf_kernel] is unsupported, data_type DT_FLOAT16 of input x is unsupported by current kernel info store. op  
type[MatrixInverse].  
Can't find any supported ops kernel and engine of [MatrixInverse], type is [MatrixInverse]  
  
Aborted (core dumped)  
root@ubuntu:/home/.../op_tf_testcase_smoke_host# /usr/local/python3.7.5/lib/python3.7/multiprocessing/semaphore_tracker.py:144: UserWarning: semapho  
re_tracker: There appear to be 91 leaked semaphores to clean up at shutdown  
  ten(cache))
```

此种场景下，即可开启Cast算子自动插入特性，详细操作方法见[操作步骤](#)。

### 操作步骤

**步骤1** 打开AutoCast开关。

修改“Ascend-cann-toolkit安装目录/ascend-toolkit/latest”目录中“lib64/plugin/opskernel/config/init.conf”文件，将“AutoCastMode”参数的值修改为1，如下所示：

```
...  
AutoCastMode = 1
```

**步骤2** 修改对应的算子信息库，在需要修改的算子中插入Cast转换规则。

如下所示，MatrixInverse算子的输入x不支持float16，算子信息库配置如下：

```
"MatrixInverse":{
  "input0":{
    "name":"x",
    "type":"DT_FLOAT,DT_DOUBLE,DT_COMPLEX128,DT_COMPLEX64"
  },
  "opInfo":{
    "computeCost":"100",
    "engine":"DNN_VM_AICPU",
    "flagAsync":"False",
    "flagPartial":"False",
    "formatAgnostic":"False",
    "opKernelLib":"TFKernel",
    "opsFlag":"OPS_FLAG_OPEN",
    "subTypeOfInferShape":"1"
  },
  "output0":{
    "name":"y",
    "type":"DT_FLOAT,DT_DOUBLE,DT_COMPLEX128,DT_COMPLEX64"
  }
},
```

为了让其支持float16，需要做如下修改：

1. 对输入信息进行修改，增加支持的数据类型，并增加数据类型转换规则。

例如，对MatrixInverse算子，输入增加对float16类型的支持，并增加cast规则，将float16转换为float32，代表在此输入前会插入一个float16到float32的cast算子。

```
"input0":{
  "name":"x",
  "type":"DT_FLOAT,DT_DOUBLE,DT_COMPLEX128,DT_COMPLEX64,DT_FLOAT16",
  "srcAutoCastType":"DT_FLOAT16",
  "dstAutoCastType":"DT_FLOAT"
},
```

- 支持的“type”中增加“DT\_FLOAT16”数据类型，支持的数据类型可参见对应的算子信息库中Cast算子的定义。
- 增加配置“srcAutoCastType”，代表输入数据的类型。
- 增加配置“dstAutoCastType”，代表需要转换成的目标数据类型。

2. 对输出信息进行修改，增加支持的数据类型，并增加数据类型转换规则。

例如，对MatrixInverse算子，输出增加对float16类型的支持，并增加cast规则，将float32转换为float16，代表在此输出后插入一个float32到float16的cast算子。

```
"output0":{
  "name":"y",
  "type":"DT_FLOAT,DT_DOUBLE,DT_COMPLEX128,DT_COMPLEX64,DT_FLOAT16"
  "srcAutoCastType":"DT_FLOAT",
  "dstAutoCastType":"DT_FLOAT16"
}
```

- 支持的“type”中增加“DT\_FLOAT16”数据类型，支持的数据类型可参见对应的算子信息库中Cast算子的定义。
- 增加配置“srcAutoCastType”，代表输入数据的类型。
- 增加配置“dstAutoCastType”，代表需要转换成的目标数据类型。

---

#### 须知

- 若算子的多个输入、多个输出要求具有相同的数据类型，则每个输入、输出都需要按照上述规则进行修改。
  - 由于插入Cast算子，精度会有一定程度的损失，具体损失大小与转换的数据类型有关。
- 

----结束



# 10 FAQ

开发环境架构为Arm ( aarch64 )，模型转换耗时较长

使用AIPP色域转换模型时如何判断视频流的格式标准

如何确定原始框架网络模型中的算子与昇腾AI处理器或BS9SX1A AI处理器支持的算子的对应关系

目标模型中包含算子原型库未注册的算子

目标模型中因为未找到算子归属引擎导致转换失败

缺省input\_shape参数导致转换失败

out\_nodes参数指定的输出节点不存在

out\_nodes参数输入校验失败导致转换失败

input\_fp16\_nodes参数指定的节点名字不存在

算子inferred\_shape失败导致模型转换失败

算子找不到算子信息库导致模型转换失败

insert\_op\_conf参数文件路径不存在导致转换失败

insert\_op\_conf参数指定的aipp配置文件校验失败导致转换失败

dynamic\_image\_size或dynamic\_batch\_size设置分辨率数值过大或档位过多导致运行环境异常缓慢

算子信息库中缺少算子

## 10.1 开发环境架构为 Arm ( aarch64 )，模型转换耗时较长

若开发环境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ATC安装用户登录开发环境，执行**su root**命令切换到root用户。
2. 确保开发环境已连接网络后，执行以下命令安装numactl工具。  
yum -y install numactl
3. 切换到ATC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 <args>
```

<args>请替换为具体使用ATC模型转换命令。

## 10.2 使用 AIPP 色域转换模型时如何判断视频流的格式标准

### 现象描述

使用AIPP色域转换模型时无法判断视频流的格式标准。

### 解决措施

此处以第三方ffprobe工具为例，演示如何进行判断。

**步骤1** 从官方路径下载工具及相关文档：<https://www.ffmpeg.org/ffprobe-all.html#Description>。

**步骤2** 通过 **ffprobe -show\_frames filename**参数获取对应视频信息。

**参数功能：** Show information about each frame and subtitle contained in the input multimedia stream. The information for each single frame is printed within a dedicated section with name "FRAME" or "SUBTITLE".

**步骤3** 通过结果中如下信息确认是哪种视频标准。

"color\_range": "tv" 或者 "pc"

"color\_space": "bt709" 或者 "bt601"

其中tv代表：tv表示 limited，即narrow range。pc代表：pc表示full，即wide range。

例如查询结果为color\_range=tv，color\_space=bt709，则代表BT-709，NARROW。

#### 说明

如命令变化，请以工具官方说明为准。

----结束

## 10.3 如何确定原始框架网络模型中的算子与昇腾 AI 处理器或 BS9SX1A AI 处理器支持的算子的对应关系

### 问题描述

用户使用精度比对工具或者性能比对工具进行算子精度或者性能分析时，若发现某些算子精度或者性能有问题，可能会考虑使用ATC工具中的某些参数调整算子的计算精度后，重新进行模型转换然后推理，比如通过**7.3.3.3 --modify\_mixlist**参数将有问题的算子配置为黑名单等等，该场景下，ATC中的参数要求配置的必须为基于Ascend IR定义的算子的OpType。

那如何获取此类算子的OpType？或者如果通过原始框架网络模型中的算子，来获取我们昇腾AI处理器或BS9SX1A AI处理器对应支持的算子的OpType呢？

## 解决方案

- 如果用户正在使用Profiling工具进行算子性能分析，该场景下直接获取昇腾AI处理器或BS9SX1A AI处理器支持的算子类型即可，参见《[CANN 开发工具指南（开放态）](#)》中的“Profiling工具使用指南”章节手册：
  - 导出summary数据中的“AI Core和AI CPU算子数据”，文件名为“op\_summary\_\*.csv”格式。
  - 该文件中的“OP Type”列即为昇腾AI处理器或BS9SX1A AI处理器支持的算子的OpType，从该列中找到有问题的算子即可。
- 如果用户正在使用精度比对工具进行算子精度分析：
  - 参见《[CANN 开发工具指南（开放态）](#)》中的“精度比对工具使用指南”章节手册获取精度比对结果文件result\_\*.csv。
  - 根据该文件中的“NPUDump”列找到有问题的算子名，然后到对应dump数据文件中检索对应的OpType。

dump数据的第一段即为昇腾AI处理器或BS9SX1A AI处理器支持的算子的OpType，例如下图dump数据中标红部分的算子信息：

```
Pooling.MaxPool2D_1.5.3.1656495909948325
Pooling.MaxPool2D_2.14.3.1656495909974359
Pooling.MaxPool2D_3.23.3.1656495909993443
SoftmaxV2_softmax.42.3.1656495910027798
TransData.trans_TransData_1.3.3.1656495909934641
```

## 10.4 目标模型中包含算子原型库未注册的算子

### 现象描述

模型转换失败，日志中有类似no OpProto of [XXX] registered的日志，如[图10-1](#)所示。

图 10-1 日志信息：no OpProto of [XXX] registered

```
INFO GE:2020-03-17-01:59:30.359.576 DFSTopologicalSorting:common/graph/.compute_graph.cpp:413:"node_vec.push_back fc1000"
INFO GE:2020-03-17-01:59:30.359.610 DFSTopologicalSorting:common/graph/.compute_graph.cpp:413:"node_vec.push_back prob"
INFO GE:2020-03-17-01:59:30.359.643 DFSTopologicalSorting:common/graph/.compute_graph.cpp:413:"node_vec.push_back Node_Output"
INFO GE:2020-03-17-01:59:30.359.737 DumpGraph:common/graph/.utils/graph_utils.cpp:410:"Start to dump on txt: 4"
WARNING GE:2020-03-17-01:59:30.359.873 SerializedEdge:common/graph/.model/serialize.cpp:188:"Input Anchor PeerOutAnchor Is Empty nodeName data nodeType Data"
WARNING GE:2020-03-17-01:59:31.166.120 EncodeModelLink:common/graph/.utils/ge_ir_utils.cpp:508:"Input Data Anchor PeerOutAnchor Is Empty nodeName data nodeType Data"
INFO GE:2020-03-17-01:59:31.209.384 DumpGraphToDnnx:common/graph/.utils/graph_utils.cpp:497:"Start to dump ge onnx file: 4"
INFO GE:2020-03-17-01:59:32.030.831 SetDefaultParams:framework/dnnx/graph/preprocess/insert_op/ge_aipp_op.cc:141:"parse aipp params:input_format:1, csc_switch:1."
INFO GE:2020-03-17-01:59:32.030.857 SetDefaultParams:framework/dnnx/graph/preprocess/insert_op/ge_aipp_op.cc:144:"parse aipp params:mean_chn_0:104, mean_chn_1:117, mean_chn_2:123."
INFO GE:2020-03-17-01:59:32.030.874 SetDefaultParams:framework/dnnx/graph/preprocess/insert_op/ge_aipp_op.cc:147:"parse aipp params:min_chn_0:0.000000, min_chn_1:0.000000, min_chn_2:0.000000."
WARNING GE:2020-03-17-01:59:32.031.446 CreateOperator:common/graph/.operator_factory_impl.cpp:33:"no OpProto of [aipp] registered"
ERROR GE:2020-03-17-01:59:32.031.472 InsertAippToGraph:framework/dnnx/graph/preprocess/insert_op/base_insert_op.cc:60:"aipp is not registered Error Code:0x50100001()"
ERROR GE:2020-03-17-01:59:32.031.542 InsertAippOps:framework/dnnx/graph/preprocess/insert_op/util_insert_aipp_op.cc:83:"insert op to graph failed"
ERROR GE:2020-03-17-01:59:32.031.563 TryDoAipp:framework/dnnx/graph/preprocess/graph_preprocess.cpp:521:"ErrorNo: 1343242252(Graph which insert dynamic op failed.) *TryDoAipp: insert aipp op ret failed, ret:1343225857"
ERROR GE:2020-03-17-01:59:32.031.580 Prepare:framework/dnnx/graph/preprocess/graph_preprocess.cpp:658:"ErrorNo: 1343242252(Graph which insert dynamic op failed.) *Run graph prepare fail, ret:1343242252"
ERROR GE:2020-03-17-01:59:32.031.600 PreRun:framework/dnnx/graph/manager/graph_manager.cpp:259:"ErrorNo: 1343242252(Graph which insert dynamic op failed.) *OMS RunGraph insert compute graph is NULL"
ERROR GE:2020-03-17-01:59:32.031.619 StartForRunGraph:framework/dnnx/graph/manager/graph_manager.cpp:384:"ErrorNo: 1343242252(Graph which insert dynamic op failed.) *PreRun failed."
ERROR GE:2020-03-17-01:59:32.031.639 BuildGraph:framework/dnnx/graph/manager/graph_manager.cpp:577:"ErrorNo: 1343242268(PreRun failed.) *BuildGraph StartForRunGraph failed"
INFO GE:2020-03-17-01:59:32.031.655 BuildModel:framework/dnnx/graph/manager/graph_manager.cpp:370:"ErrorNo: 1343242268(Graph manager build graph failed.) *BuildModel failed"
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：

aipp算子没有原型库注册。

## 解决措施

**步骤1** 确认报错算子，如aipp是否是新增算子，是否有算子原型库，并编译进libopsproto.so。

**步骤2** 确认是否是版本时间差异。

----结束

## 10.5 目标模型中因为未找到算子归属引擎导致转换失败

### 现象描述

模型转换失败，报错的日志中有类似Can not find engine of op type Data信息，如图10-2所示。

图 10-2 日志信息：Can not find engine of op type Data

```
is TFKernel
[ERROR] GE:2020-03-26-07:51:40.581.887 GetOpsKernelInfo: ErrorNo: -1(failed) Failed to get opsKernelInfo object by type: Data.
[ERROR] GE:2020-03-26-07:51:40.581.902 Run: ErrorNo: 1343229963(GE is not yet initialized or is finalized.) Can not find engine of op type Data
[ERROR] GE:2020-03-26-07:51:40.581.914 Initialize: ErrorNo: -1(failed) Engine placer run failed.
[ERROR] GE:2020-03-26-07:51:40.581.926 Partition: ErrorNo: 1343242240(failed to initialize graph.) [GraphPartitioner]: initialize failed
[ERROR] GE:2020-03-26-07:51:40.581.937 PreRun: ErrorNo: -1(failed) Graph partition failed
[ERROR] GE:2020-03-26-07:51:40.581.951 StartForRunGraph: ErrorNo: -1(failed) PreRun Failed.
[ERROR] GE:2020-03-26-07:51:40.581.962 BuildGraph: ErrorNo: 1343242268(PreRun failed.) [BuildGraph] StartForRunGraph failed!
[ERROR] GE:2020-03-26-07:51:40.581.974 BuildModel: ErrorNo: 1343266819(Graph manager build graph failed.) graphManager BuildGraph failed, id: 0
[ERROR] GE:2020-03-26-07:51:40.582.022 GenerateModel: ErrorNo: 1343266819(Graph manager build graph failed.) Build model failed
[ERROR] GE:2020-03-26-07:51:40.582.022 GenerateModel: "GE GenerateOfflineModel execute failed"
[ERROR] GE:2020-03-26-07:51:40.582.033 GenerateModel: "ATC Generate execute failed"
[DEBUG] FE:2020-03-26-07:51:40.582.050 Finalize: "Finalizing the FEOpsKernelStore..."
[DEBUG] FE:2020-03-26-07:51:40.582.064 FinalizeSubStore: "Finalizing the SubOpsKernelInfoStore, begin..."
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：

这个算子的引擎so没有被加载：需要确认这个算子是否有归属引擎，如果有归属引擎，则确认引擎是否没有被正确加载。

## 解决措施

**步骤1** 确认报错算子的归属引擎。例如报错日志中的Data是归属于libge\_local\_engine.so。

**步骤2** 搜索info日志关键字OPTION\_EXEC\_EXTERN\_PLUGIN\_PATH，后面的内容就是加载引擎的路径，查看环境上是否有这个路径；或者搜WARNING日志是否出现Failed to get realpath of日志。

**步骤3** 如果路径加载不正确，确认是否配置了ASCEND\_ENGINE\_PATH环境变量，若配置了则代码中会取该环境变量中的路径下去加载，需确保环境变量配置的路径正确。

----结束

## 10.6 缺省 input\_shape 参数导致转换失败

### 现象描述

模型转换失败，日志中有“input x shape is empty”的内容信息。如图10-3所示。

图 10-3 日志信息：“input x shape is empty”

```
are::InferOriginFormat1 is (9047) micro second.
[ERROR] GE(9478,atc):2020-07-02-00:10:08.261.524 [ops/built-in/op_proto/nn_calculation_ops.cpp:2612][OP_PROTO] Conv2dVerify:2612 OpName:[InceptionV3/InceptionV3/Mixed_Sc/Br
ch_0/Conv2d_0a_1x1/Conv2d] "input x shape is empty."
[ERROR] GE(9478,atc):2020-07-02-00:10:08.261.559 [common/graph/../shape_refiner.cc:338]9478 InferShapeAndType: ErrorNo: -1(failed) Verifying InceptionV3/InceptionV3/Mixed_Sc/B
ranch_0/Conv2d_0a_1x1/Conv2d failed.
[ERROR] GE(9478,atc):2020-07-02-00:10:08.261.577 [framework/domi/graph/passes/infer_shape_pass.cc:27]9478 Run: ErrorNo: 1343242270(Prepare Graph infer_shape failed) infer_shape
failed, node: InceptionV3/InceptionV3/Mixed_Sc/Branch_0/Conv2d_0a_1x1/Conv2d
[ERROR] GE(9478,atc):2020-07-02-00:10:08.261.591 [framework/domi/graph/passes/base_pass.cc:88]9478 RunPasses: ErrorNo: 1343225860(Internal errors) Failed to process pass Infe
rShapePass on node InceptionV3/InceptionV3/Mixed_Sc/Branch_0/Conv2d_0a_1x1/Conv2d, result 1343242270, the passes will be terminated immediately.
[ERROR] GE(9478,atc):2020-07-02-00:10:08.261.608 [framework/domi/graph/passes/base_pass.cc:216]9478 RunPassesOneGraph: ErrorNo: 1343242270(Prepare Graph infer_shape failed) Fa
```

## 可能原因

分析上述日志信息，可能存在以下故障原因：

原始的TensorFlow模型输入是动态shape，如“ $? \times 299 \times 299 \times 3$ ”，转换模型的时候缺省了input\_shape选项导致报错。

## 解决措施

**步骤1** 使用Netron工具查看模型，确认模型输入的shape。

**步骤2** 对于动态shape的模型，转换时必须加上input\_shape参数。命令行举例：

```
atc --model=./resnetv2.pb --framework=3 --input_shape="input_tensor:8,299,299,3" --input_format="NHWC" --output=./resnetv2 --soc_version=Ascend310
```

----结束

## 10.7 out\_nodes 参数指定的输出节点不存在

### 现象描述

模型转换失败，报错日志中有“can not find node: \*”或者“node \* not found”的内容信息。如图10-4所示。

图 10-4 日志信息：can not find node

```
[DEBUG] GE:2020-04-03-03:21:34.848.601 AddEdges:Start add edge: From data1:0 to conv1:0.  
[ERROR] GE:2020-04-03-03:21:34.848.619 AddEdgeForUserOutNodes: ErrorNo: 50331649() Can not find out node:conv2, you should check --out_nodes  
[ERROR] GE:2020-04-03-03:21:34.848.632 Parse:"Caffe parser add edges for user out nodes failed."  
[ERROR] GE:2020-04-03-03:21:34.848.669 ParseGraph:"ATC model parse ret fail. Error Code:0x3000001(Parameter's invalid!)"  
[ERROR] GE:2020-04-03-03:21:34.848.700 GenerateModel:"ATC Parse graph dom1::FAILED"  
[ERROR] GE:2020-04-03-03:21:34.848.712 GenerateModel:"ATC Generate execute failed"
```

## 可能原因

分析上述日志信息，可能存在以下故障原因：

- 转模型使用了out\_nodes参数指定输出节点（算子名称），但是指定的输出节点在Graph中不存在。
- 指定的输出节点在Graph中存在，但是依然报找不到，可能是这个节点在转换过程中被融合了。

## 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

**步骤1** 检查Graph中是否存在out\_nodes参数指定的输出节点。

如果指定的节点在Graph中不存在，需要重新指定正确的节点。

**步骤2** 如果Graph中有指定的输出节点，依然报找不到错误，可以通过以下方法确认该节点在转换过程中是否被融合了：

尝试先不带out\_nodes参数进行转换，然后最后通过查看最终build的图是否有这个节点，若没有则说明已经被融合掉，这种情况需要重新指定输出节点。

----结束



## 10.8 out\_nodes 参数输入校验失败导致转换失败

### 现象描述

模型转换失败，报错日志中有 “The input format of --out\_nodes is invalid”，如图 10-5 所示。

图 10-5 日志信息：The input format of --out\_nodes is invalid

```
ATC start working now, please wait for a moment.
[ERROR] GE(2020-03-26-08:13:34.634.276 ParseOutNodes: ErrorNo: 50331649) The input format of out_nodes is invalid, the correct format is [opname:index], while the actual input is conv1.
[ERROR] GE(2020-03-26-08:13:34.634.303 ParseGraph: "ATC Generate parse out nodes fail"
[ERROR] GE(2020-03-26-08:13:34.634.320 GenerateModel: "ATC Parse graph domi::FAILED"
[ERROR] GE(2020-03-26-08:13:34.634.333 GenerateModel: "ATC Generate execute failed"
ATC run failed, please check the detail log
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：

转模型使用了out\_nodes参数指定输出节点（算子名称），但未指定第几个输出，输入格式校验错误。

### 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

确认out\_nodes入参的格式，需要指定第几个输出，例如--out\_nodes="node\_name1:0;node\_name1:1;node\_name2:0"

## 10.9 input\_fp16\_nodes 参数指定的节点名字不存在

### 现象描述

模型转换失败，报错日志中有 “Input parameter[--input\_fp16\_nodes]'s opname[\*\*\*] is not exist in model”，如图 10-6 所示。

图 10-6 日志信息：“Input parameter[--input\_fp16\_nodes]'s opname[\*\*\*] is not exist in model”

```
[INFO] GE(4564,atc):2020-09-02-10:53:30.534.105 [framework/domi/offline/./session/omg.cc:146]4564 CheckInputFp16Nodes:The input_fp16_nodes is set dat
[ERROR] GE(4564,atc):2020-09-02-10:53:30.534.283 [framework/domi/offline/./session/omg.cc:154]4564 CheckInputFp16Nodes: ErrorNo: 1343225857(Parameter's invalid) Input paramet
or (--input_fp16_nodes)'s opname[dat] is not exist in model
[ERROR] GE(4564,atc):2020-09-02-10:53:30.534.604 [framework/domi/offline/main.cc:946]4564 GenerateModel: ErrorNo: -1(failed) ATC Parse graph domi::FAILED
[ERROR] GE(4564,atc):2020-09-02-10:53:30.534.622 [framework/domi/offline/main.cc:947]4564 GenerateModel: ErrorNo: -1(failed) ATC Generate execute failed
[INFO] GE(4564,atc):2020-09-02-10:53:30.534.639 [framework/domi/graph/execute/graph_execute.cc:110]4564 FreeInOutBuffer:[GraphManager] not malloc buffer.
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：

转模型使用了input\_fp16\_nodes参数指定支持输入数据类型为FP16，该参数后面带的是输入节点的名字，该处报错是指这个节点名字没有找到。

#### 说明

如果日志报类似这样的信息：input\_fp16\_nodes: res5c\_relu is not a input node name，表示input\_fp16\_nodes参数指定的节点不是输入节点。

## 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

打开原始model确认输入节点的名字与input\_fp16\_nodes后面的是否一致。

## 10.10 算子 infershape 失败导致模型转换失败

### 现象描述

目标模型转换失败，报错日志中有以下类似信息：Run ge\_passes infershape for preprocess failed，如图10-7所示。

图 10-7 日志信息：Run ge\_passes infershape for preprocess failed

```
root@ubuntu:/var/dlog# grep ERROR host-0/host-0_20200229170651057.log
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.040 RandomShapeInitCateType:Layer1.Dropout/random_uniform/RandomUniform: get dtype attr failed.
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.049 CallInterface:Layer1.Dropout/random_uniform/RandomUniform call infer func, ret: 434967296
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.053 InferShapeAndType:Layer1.Dropout/random_uniform/RandomUniform call infer function failed.
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.057 Run: ErrorNo: 1343242270(Prepare Graph infershape failed) infershape failed, node: Layer1.Dropout/random_uniform/RandomUniform
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.061 RunPasses: ErrorNo: 1343225869(Internal errors) failed to process pass InferShapePass on node Layer1.Dropout/random_uniform/RandomUniform, result 1343242270, the passes will be terminated immediately.
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.065 Run: ErrorNo: 1343225869(Internal errors) failed to process passes on node Layer1.Dropout/random_uniform/RandomUniform type RandomUniform, error code: 1343242270
[ERROR] GE(1537,mg):2020-02-29-17:07:00.868.082 InferShapeForPreprocess: ErrorNo: 1343225869(Internal errors) Run ge_passes infershape for preprocess failed, ret:1343225869.
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.879 FormatAndShapeProcess: ErrorNo: 1343242270(Prepare Graph infershape failed) Prepare Graph infershape failed
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.902 Preprocess: ErrorNo: 1343242270(Prepare Graph infershape failed) FormatAndShape process failed
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.908 Prepare: ErrorNo: 1343242270(Prepare Graph infershape failed) Run graph prepare fail, ret:1343242270
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.911 PreRun: ErrorNo: 1343242270(Prepare Graph infershape failed) ATC RunGraph input compute graph is NULL
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.914 StartForRunGraph: ErrorNo: 1343242270(Prepare Graph infershape failed) PreRun failed.
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.917 BuildGraph: ErrorNo: 1343242268(PreRun failed) [BuildGraph] StartForRunGraph failed
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.921 BuildModel: ErrorNo: 1343266819(Graph manager build graph failed.) graphManager BuildGraph failed, id: 0
[ERROR] GE(1537,mg):2020-02-29-17:07:02.643.924 GenerateOfflineModel: ErrorNo: 1343266819(Graph manager build graph failed.) Build model failed
[ERROR] GE(1537,mg):2020-02-29-17:07:02.644.023 GenerateModel:framework/dm/offline/main.cpp:917: GE GenerateOfflineModel execute failed
[ERROR] GE(1537,mg):2020-02-29-17:07:02.644.033 GenerateModel:framework/dm/offline/main.cpp:918: ATC Generate execute failed
root@ubuntu:/var/dlog# ll
total 24
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：报错算子的输入有问题。

### 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

- 步骤1 定位报错算子实现的infershape失败的原因，确认异常输入的上个算子infershape实现是否正确。
  - 步骤2 不断向上回溯到infershape实现异常的问题算子。
- 结束

## 10.11 算子找不到算子信息库导致模型转换失败

### 现象描述

模型转换失败，报错日志中有“Can't find any supported ops kernel and engine of %s, type is %”，如图10-8所示。

图 10-8 日志信息：Can't find any supported ops kernel and engine of %s, type is %

```
t failed, kernel_name is AICoreEngine, op type is RoiPooling, op name is roi_pooling
[ERROR] GE(26576,atc):2020-03-16-20:40:32.744.432 [framework/dm/engine_manager/dnnengine_manager.cc:223]GetDNNEngineName: ErrorNo: 1343242282(assign engine failed) GetDNNEngineName:Op type RoiPooling of ops kernel AICoreEngine is unsupported, reason:Op roi_pooling not supported reason: The dtype, format or shape of input in op desc is not supported in op store, check the dtype, format or shape of input between the op store and the graph. Op store name is tbe-builtin.
The precompileFunc is nullptr, check whether this type of op does not exist in the-builtin and then go to the tbe-plugin. Op store name is tbe-plugin. The type of this op is not found in op store, check whether the op store has this type of op. Op store name is cce-builtin.
[ERROR] GE(26576,atc):2020-03-16-20:40:32.744.458 [framework/dm/engine_manager/dnnengine_manager.cc:226]GetDNNEngineName: ErrorNo: 1343242282(assign engine failed) Can't find any supported ops kernel and engine of roi_pooling, type is RoiPooling
[ERROR] GE(26576,atc):2020-03-16-20:40:32.744.483 [framework/dm/graph/partition/engine_place.cc:54]Run: ErrorNo: 1343229963(GE is not yet initialized or is finalized.) Can not find engine of op type RoiPooling
[ERROR] GE(26576,atc):2020-03-16-20:40:32.744.507 [framework/dm/graph/partition/graph_partition.cc:523]Initialize: ErrorNo: -1(failed) Engine placer
```

## 可能原因

分析上述日志信息，可能存在以下故障原因：

该算子在算子信息库中没找到。

## 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

**步骤1** 确认这个算子是否在算子信息库中支持。

**步骤2** 如果这个算子的dtype、format、shape不匹配算子信息库中的信息，则需要具体查看该算子推导出来的过程，排查是否是由于输入的节点异常导致的。

----结束

## 10.12 insert\_op\_conf 参数文件路径不存在导致转换失败

### 现象描述

模型转换失败，报错日志中有“insert op config file not found”，如图10-9所示。

图 10-9 日志信息：insert op config file not found

```
[ERROR] GE(9851,atc):2020-07-02-00:41:08.831.829 [framework/domi/offline/././ir_build/atc_ir_common.cc:326]9851 CheckInsertOpConfParamValid: ErrorNo: 1343225857(Parameter's is not valid) insert op config file not found: /home/test/aipp.cfg  
[ERROR] GE(9851,atc):2020-07-02-00:41:08.831.858 [framework/domi/offline/main.cc:370]9851 CheckFlags: ErrorNo: -1(failed) check insert op conf failed!
```

### 可能原因

分析上述日志信息，可能存在以下故障原因：

insert\_op\_conf入参是在aipp场景下才需要的，后面是aipp配置文件的路径，路径访问不到。

### 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：

**步骤1** 确认是否是aipp场景，若不是，则不需要此入参。

**步骤2** 如果是aipp场景，确定路径是否正确，若路径正确则查看属组权限是否正常。

----结束

## 10.13 insert\_op\_conf 参数指定的 aipp 配置文件校验失败导致转换失败

### 现象描述

模型转换失败，报错日志中有“Read AIPP conf file error”，如图10-10所示。



图 10-10 日志信息：Read AIPP conf file error

```
[libprotobuf ERROR google/protobuf/text_format.cc:317] Error parsing text-format domi.InsertNewOps: 3:5: Unknown enumeration value of "crop" for field "aipp_mode".
[ERROR] GE(9913,atc):2020-07-02-00:48:05.525.876 [framework/domi/common/util.cc:286]9913 ReadProtoFromText: ErrorNo: 0(success) Parse file[./bak.cfg] through [google::protobuf::TextFormat::Parse] failed, please check whether the file is a valid protobuf format file.
[ERROR] GE(9913,atc):2020-07-02-00:48:05.525.952 [framework/domi/graph/preprocess/insert_op/util_insert_aipp_op.cc:72]9913 Parse: ErrorNo: -1(failed) Read AIPP conf file error: ./bak.cfg Error Code:0xffffffff(failed)
[ERROR] GE(9913,atc):2020-07-02-00:48:05.525.976 [framework/domi/graph/preprocess/graph_preprocess.cc:1786]9913 Try0oAipp: ErrorNo: 1343242285(OMG parse dynamic node config file failed.) Try0oAipp: parse config file ./bak.cfg failed
[EVENT] GE(9913,atc):2020-07-02-00:48:05.526.000 [framework/domi/graph/preprocess/graph_preprocess.cc:2087]9913 PrepareDynShape:[GEPERFTRACE] The time cost of Prepare::Try0oAipp is [273] micro second.
[ERROR] GE(9913,atc):2020-07-02-00:48:05.526.024 [framework/domi/graph/preprocess/graph_preprocess.cc:2087]9913 PrepareDynShape: ErrorNo: 1343242285(OMG parse dynamic node config file failed.) failed to process Prepare_Try0oAipp
[EVENT] GE(9913,atc):2020-07-02-00:48:05.526.045 [framework/domi/graph/manager/graph_manager.cc:371]9913 PreRun:[GEPERFTRACE] The time cost of GraphManager::graph_preparer_.PrepareDynShape is [7534] micro second.
[ERROR] GE(9913,atc):2020-07-02-00:48:05.526.103 [framework/domi/graph/manager/graph_manager.cc:371]9913 PreRun: ErrorNo: 1343242285(OMG parse dynamic node config file failed.) failed to process GraphManager_graph_preparer_.PrepareDynShape
```

## 可能原因

分析上述日志信息，可能存在以下故障原因：  
insert\_op\_conf入参指定的aipp文件解析失败。

## 解决措施

针对分析的故障可能原因，可以参考下面步骤处理：  
确认aipp配置文件是否正确，是否是protobuf文件格式。

## 10.14 dynamic\_image\_size 或 dynamic\_batch\_size 设置分辨率数值过大或档位过多导致运行环境异常缓慢

### 现象描述

执行动态shape模型转换时，Host侧运行环境缓慢，无法执行其他操作，需重新启动昇腾处理器才能恢复正常，查看日志有如下类似信息，如图10-11所示。

图 10-11 一直打印等待日志

```
[INFO] TEUFUSION:2020-04-17-11:52:30.311.097 WaitAllFinished Get finished compilation task list size[0], graphId[140048978401024]
[INFO] TEUFUSION:2020-04-17-11:52:30.577.297 WaitAllFinished Get finished compilation task list size[0], graphId[140048961615616]
[INFO] TEUFUSION:2020-04-17-11:52:30.833.047 WaitAllFinished Get finished compilation task list size[0], graphId[140048953222912]
[INFO] TEUFUSION:2020-04-17-11:52:31.106.662 WaitAllFinished Get finished compilation task list size[0], graphId[140048970008320]
[INFO] TEUFUSION:2020-04-17-11:52:31.315.255 WaitAllFinished Get finished compilation task list size[0], graphId[140048928044800]
[INFO] TEUFUSION:2020-04-17-11:52:31.415.745 WaitAllFinished Get finished compilation task list size[0], graphId[140048886091280]
[INFO] TEUFUSION:2020-04-17-11:52:31.520.568 WaitAllFinished Get finished compilation task list size[0], graphId[140048877688576]
[INFO] TEUFUSION:2020-04-17-11:57:41.538.210 WaitAllFinished Get finished compilation task list size[0], graphId[140048953222912]
```

## 可能原因

动态shape模型转换时，算子编译会占用较大内存，如果Host主机内存配置不够，会出现Host侧内存占用光，且将swap交换区间作为内存继续处理，造成IO写满，导致Host运行环境变慢。

## 解决措施

针对执行动态shape模型转换场景下，尤其是dynamic\_image\_size或dynamic\_batch\_size设置分辨率数值过大或档位过多，建议执行**swapoff -a**命令关闭swap交换区间作为内存的功能，防止出现由于内存不足，将swap交换空间作为内存继续调用，导致运行环境异常缓慢的情况。

## 10.15 算子信息库中缺少算子

### 现象描述

执行模型转换时，报错日志信息：“The type of this op is not found in op store”，如图10-12所示。

图 10-12 日志信息：The type of this op is not found in op store

```
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.767 [framework/domi/engine_manager/dnnengine_manager.cc:241]18817 GetDNNEngineName: ErrorNo: 1343242282(assign engine failed) GetDNNEngineName:Op type Range of ops kernel AicoreEngine is unsupported, reason:Op range not supported reason: The type of this op is not found in op store, check whether the op store has this type of op. Op store name is tbe-builtin. The preCompileFunc is nullptr, check whether this type of op does not exist in the-builtin and then go to the tbe-plugin. Op store name is the-plugin.
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.788 [framework/domi/engine_manager/dnnengine_manager.cc:246]18817 GetDNNEngineName: ErrorNo: 1343242282(assign engine failed) Can't find any supported ops kernel and engine of range, type is Range
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.800 [framework/domi/graph/partition/engine_place.cc:55]18817 Run: ErrorNo: 1343229963(GE is not yet initialized or is finalized.) can not find engine of op type Range
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.812 [framework/domi/graph/partition/graph_partition.cc:504]18817 Initialize: ErrorNo: -1(failed) Engine placer run failed.
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.847 [framework/domi/graph/partition/graph_partition.cc:855]18817 PartitionSubGraph: ErrorNo: 1343242240(failed to initialize graph.) [graphPartitioner]: initialize failed
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.870 [framework/domi/graph/partition/graph_partition.cc:826]18817 Partition: ErrorNo: -1(failed) Sub graph partition failed
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.880 [framework/domi/graph/manager/graph_manager.cc:2326]18817 OptimizeSubgraph: ErrorNo: -1(failed) Graph partition failed
ERROR] GE(18669,python3):2020-07-03-03:52:51.235.900 [framework/domi/graph/manager/graph_manager.cc:392]18817 PreRun: ErrorNo: -1(failed) Failed to process GraphManager::OptimizeSubgraph
```

### 可能原因

分析上述日志信息，可能原因：

aicore或aicpu的算子信息库中没有该算子，当前产品不支持该算子。

### 处理步骤

针对分析的故障可能原因，当前版本并不支持该算子，可以通过以下方式确认：

检查\${INSTALL\_DIR}/opp/op\_impl/built-in/aicpu/tf\_kernel/config/tf\_kernel.json或\${INSTALL\_DIR}/opp/op\_impl/built-in/ai\_core/tbe/config/<soc\_version>文件，确认执行的算子是否在aicpu或aicore算子信息库中，如果不存在，说明当前不支持该算子，不可以使用。

#### 说明

\${INSTALL\_DIR}表示安装路径，请根据实际环境替换。