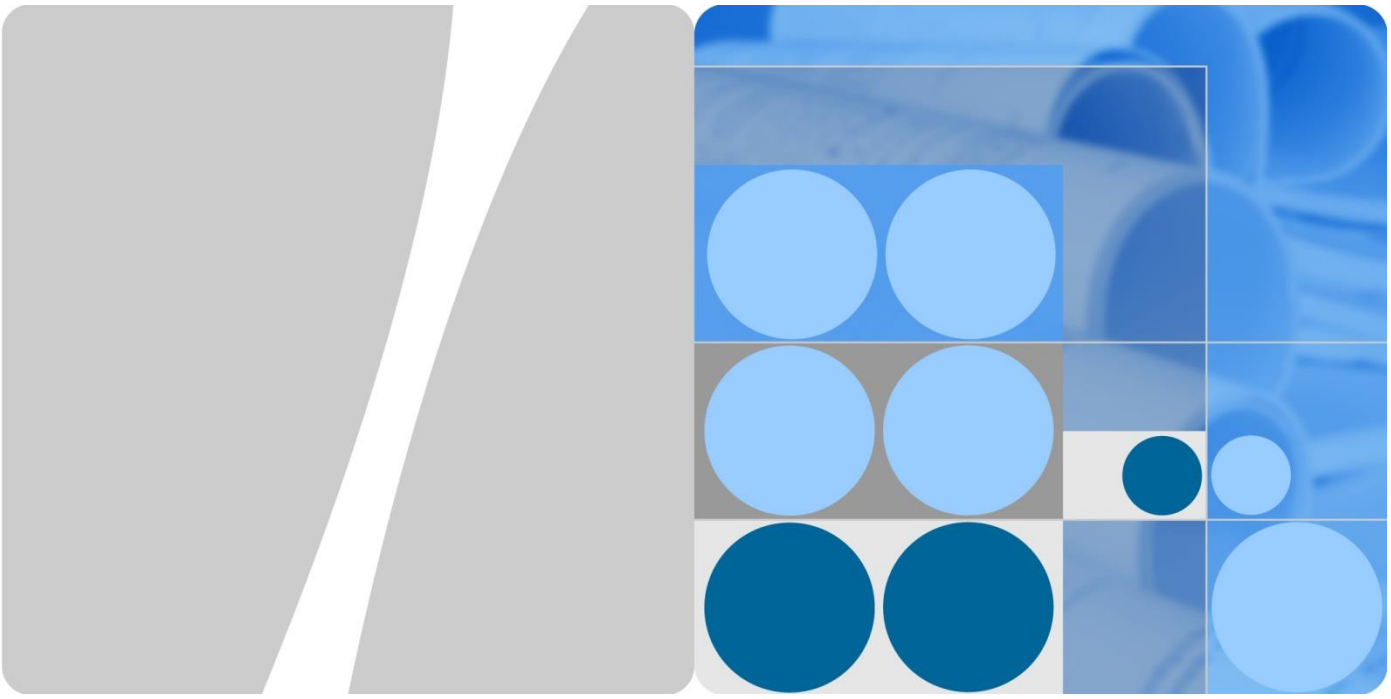

HiAI DDK V300

算子规格说明

文档版本 01

发布日期 2019-05-28

华为技术有限公司

华为 HiAI 申请方式

发送申请邮件到邮箱：developer@huawei.com

邮件名称：HUAWEI HiAI+公司名称+产品名称

邮件正文：合作公司+联系人+联系方式+联系邮箱

我们将在收到邮件的 5 个工作日内邮件给您反馈结果，请您注意查收。

官网地址 https://developer.huawei.com/consumer/cn/

前 言

概述

本文提供对华为 HIAI DDK V300 支持的算子规格的说明。

本文与以下文档配套使用：

- 华为 HIAI DDK V300 快速入门
- 华为 HIAI DDK V300 模型推理集成指导

修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	修订版本	修改描述
2019-05-28	01	初稿完成

目 录

1 参数说明

参数	含义说明
ni	批量的大小
ci/co	通道的数量
hi/ho/	高度
wi/wo	宽度
sh/sw	步长
kh/kw	卷积核的大小
window_h(window_y)/ window_w(window_x)	窗口的大小
dh(dilation_h)/ dw(dilation_w)	卷积膨胀系数
FilterHDilation/ FilterWDilation	卷积核膨胀完以后 H/W 维度的长度
FilterH/FilterW	卷积的权重参数的 H/W 维度的值
padWHead/padHHead	W/H 维度补 PAD 后前端多出的一截
PadWTail/padHTail	W/H 维度补 PAD 后尾端多出的一截
dilationsize	用户设置的膨胀系数的值
FilterSize	用户设置的卷积核的数目
INT32_MAX	数据类型 int32 能表示范围的最大值
ALIGN	向上对齐计算
CEIL	取整计算

2 算子边界

2.1 整体约束

对于 caffe 框架，当算子的输入维度不是 4 时，如果存在 axis 参数，不能使用负数。

- Caffe 支持版本 1.0。
- Tensorflow 支持版本 1.12。
- AndroidNN 支持版本 API 28。

2.2 Caffe 算子边界

序号	算子	含义	边界
1	Absval	对输入求绝对值	【输入】 个数为 1 【参数】 engine：枚举型，默认为 0（DEFAULT=0，CAFFE=1，CUDNN=2），可选 【约束】 无限制
2	Argmax	返回输入的最大值对应的索引序号	【输入】 个数为 1 【参数】 • out_max_val：布尔型，默认为 false，可选 • top_k：unit32，默认为 1，可选 • axis：int32，可选 【约束】 无限制
3	BatchNorm	对输入做标准化	【输入】

序号	算子	含义	边界
		（x - avg(x)）/x 的标准差	个数为 1 【参数】 • use_global_stats：布尔型，必须为 true • moving_average_fraction：float，默认为 0.999，可选 • eps：float，默认为 1e-5，可选 【约束】 仅支持对 C 维度方向做归一化
4	Concat	输入数据按维度拼接	【输入】 多个输入 【参数】 • concat_dim：uint32，默认为 1，取值：大于 0，可选 • axis：int32，默认 1，可选；与 concat_dim 功能相同，二者择其一；axis== -1 时，输入维度必须等于 4，否则结果可能错误 【约束】 • 输入的 tensor，除了进行 concat 的维度外，其他维度的 size 必须相等 • 输入 tensor 的个数范围不超过[1,1000]
5	Convolution	卷积	【输入】 个数为 1，且 filter 为常量，维度为 4 【参数】 • num_output: uint32，可选 • bias_term：布尔型，默认为 true，可选 • pad：uint32，默认为 0；数组 • kernel_size：uint32；数组 • stride：uint32，默认为 1；数组 • dilation：uint32，默认为 1；数组 • pad_h：uint32，默认为 0，可选（仅 2D） • pad_w：uint32，默认为 0，可选（仅 2D） • kernel_h：uint32，可选（仅 2D） • kernel_w：uint32，可选（仅 2D） • stride_h：uint32，可选（仅 2D） • stride_w：uint32，可选（仅 2D） • group：uint32，默认为 1，可选 • weight_filler：类型为 FillerParameter，可选

序号	算子	含义	边界
			• bias_filler：类型为 FillerParameter，可选 • engine：枚举型，默认为 0（DEFAULT=0，CAFFE=1，CUDNN=2），可选 • force_nd_im2col：布尔型，默认为 false，可选 • axis: int32，默认为 1，可选 【约束】 • (inputW + padWHead + padWTail) >= (((FilterW- 1) * dilationW) + 1) • (inputW + padWHead + padWTail) /StrideW + 1 <= 2147483647 • (inputH + padHHead + padHTail) >= (((FilterH- 1) * dilationH) + 1) • (inputH + padHHead + padHTail) /StrideH + 1 <= 2147483647 • 0 <= Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 <= dilationsize < 256 • StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1)
6	Crop	截取	【输入】 2 个输入 【参数】 • axis: int32，默认为 2，axis=-1，input dim 必须等于 4，可选 • offset：uint32，数组 【约束】 无限制
7	Deconvolution	反卷积	【输入】 个数为 1，且 filter 为常量，维度为 4 【参数】 • num_output：uint32，可选 • bias_term：布尔型，默认为 true，可选 • pad：uint32，默认为 0；数组 • kernel_size：uint32；数组 • stride：uint32，默认为 1；数组 • dilation：uint32，默认为 1；数组 • pad_h：uint32，默认为 0，可选（仅 2D） • pad_w：uint32，默认为 0，可选（仅 2D） • kernel_h：uint32，可选（仅 2D）

序号	算子	含义	边界
			• kernel_w：uint32，可选（仅 2D） • stride_h：uint32，可选（仅 2D） • stride_w：uint32，可选（仅 2D） • group：uint32，默认为 1，可选 • weight_filler：类型为 FillerParameter，可选 • bias_filler：类型为 FillerParameter，可选 • engine：枚举型，默认为 0（DEFAULT=0，CAFFE=1，CUDNN=2），可选 • force_nd_im2col：布尔型，默认为 false，可选 • axis: int32，默认为 1，可选 【约束】 group = 1 dilation = 1 filterH - padHHead - 1 >= 0 filterW - padWHead - 1 >= 0 还有一条约束涉及中间变量，公式如下： 1、a = ALIGN(filter_num,16) * ALIGN(filter_c,16) * filter_h * filter_w * 2 ； 如果 ALIGN(filter_c,16)%32 = 0，a = a/1； 2、conv_input_width=(反卷积输入 w - 1) * strideW + 1； 3、b = （conv_input_width） * filter_h * ALIGN(filter_num,16) * 2 * 2； 4、a + b <= 512KB。
8	DepthwiseConvolution	深度卷积	【输入】 个数为 1，且 filter 为常量，维度为 4 【参数】 • num_output: uint32，可选 • bias_term：布尔型，默认为 true，可选 • pad：uint32，默认为 0；数组 • kernel_size：uint32；数组 • stride：uint32，默认为 1；数组 • dilation：uint32，默认为 1；数组 • pad_h：uint32，默认为 0，可选（仅 2D） • pad_w：uint32，默认为 0，可选（仅 2D） • kernel_h：uint32，可选（仅 2D）

序号	算子	含义	边界
			• kernel_w：uint32，可选（仅 2D） • stride_h：uint32，可选（仅 2D） • stride_w：uint32，可选（仅 2D） • group：uint32，默认为 1，可选 • weight_filler：类型为 FillerParameter，可选 • bias_filler：类型为 FillerParameter，可选 • engine：枚举型，默认为 0（DEFAULT=0，CAFFE=1，CUDNN=2），可选 • force_nd_im2col：布尔型，默认为 false，可选 • axis: int32，默认为 1，可选 【约束】 • filterN=inputC=group； • StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1)。
9	DetectionOutput	检测结果输出 FSR	【输入】 个数为 3 【参数】 • num_classes: 必选，类型：int32，要预测的类数 • share_location: 可选，类型: bool，默认为 true（表示不同类间共享 bounding box） • background_label_id: 可选，类型：int32，默认为 0 • nms_param: 可选，非最大抑制 • save_output_param: 可选，用于保存检测结果 • code_type: 可选，默认为 CENTER_SIZE • variance_encoded_in_target: 可选，类型：bool，默认为 true；如果为 true，方差编码在目标中，否则需要相应地调整预测偏移量 • keep_top_k: 可选，类型：int32，在 nms 步骤后每个图像要保留的总 bbox 数； • confidence_threshold: 可选，类型：float，仅考虑置信度大于阈值的检测；如果没有设置，考虑所有的 box • nms_threshold: 可选，类型：float • top_k: 可选，类型：int32

序号	算子	含义	边界
			• boxes: 可选，类型：int32，默认为 1 • relative: 可选，类型：bool，默认为 true • objectness_threshold，可选，类型：float，默认为 0.5 • class_threshold: 可选，类型：float，默认为 0.5 • biases: 数组 • general_nms_param: 可选 【约束】 • 支持 fasterrcnn 网络 • 非极大值抑制比 nmsThreshold∈(0,1) • 概率阈值 postConfThreshold∈(0,1) • 类别>=2 • 最大支持 1024 个框输入 • 输出的 w 维度是 16
10	Eltwise	按元素操作层(求和、乘积、最大值)	【输入】 至少两个输入 【参数】 • operation：可选，类型：枚举型（PROD = 0; SUM = 1; MAX = 2），默认为 SUM • coeff：数组；类型：float • stable_prod_grad：可选，类型：bool，默认为 true 【约束】 • 最大支持四个输入； • 与原生算子相比，不支持 stable_prod_grad 参数； • 支持 prod、sum、max 三种模式。
11	Elu	激活函数	【输入】 个数为 1 【参数】 alpha (optional)：类型：float，默认为 1 【约束】 无限制
12	Exp	对输入求以 e 为底的 x 次方值	【输入】 个数为 1 【参数】

序号	算子	含义	边界
			• base：可选，类型：float，默认为-1.0 • scale：可选，类型：float，默认为 1.0 • shift：可选，类型：float，默认为 0.0 【约束】 无限制
13	Flatten	输入 n*c*h*w 变为向量 n*(c*h*w)	【输入】 个数为 1（top_size 与 bottom_size 必须不一致，且不为 1；当 axis=-1 时，输入的 dim 必须等于 4） 【参数】 • axis：可选，类型：int32，默认为 1 • end_axis：可选，类型：int32, 默认为-1 【约束】 axis 必须小于 end axis
14	InnerProduct	全连接	【输入】 个数为 1 【参数】 • num_output: 可选，类型：uint32 • bias_term: 可选，类型：bool，默认为 true • weight_filler: 可选，类型：FillerParameter，维度为 2 • bias_filler: 可选，类型：FillerParameter，维度为 1 • axis: 可选，类型：int32，默认为 1 • transpose: 可选，类型：bool，默认为 false 【约束】 仅支持 transpose=false，axis=1； Bais_C <= 56832 如果客户要量化模型时，需要满足下列维度： 当 N = 1，2 * CEIL(C,16) * 16 * xH * xW<= 512 * 1024； 当 N > 1，2 * 16 * CEIL(C,16) * 16 * xH * xW <= 512 * 1024。
15	Interp	插值层	【输入】 个数为 1 【参数】 • height：可选，类型 int32，默认为 0

序号	算子	含义	边界
			• width：可选，类型 int32，默认为 0 • zoom_factor：可选，类型 int32，默认为 1 • shrink_factor：可选，类型 int32，默认为 1 • pad_beg：可选，类型 int32，默认为 0 • pad_end：可选，类型 int32，默认为 0 说明： • zoom_factor 与 shrink_factor 不能同时存在 • height 与 zoom_factor 不能同时存在 • height 与 shrink_factor 不能同时存在 【约束】 （outputH*outputW）/（inputH*inputW） > 1/7
16	LeakyRelu	LeakyRelu 激活函数	【输入】 一个输入 【参数】 同 Relu 【约束】 无限制
17	Log	对输入进行 log 计算	【输入】 个数为 1 【参数】 • base：可选，类型：float，默认为-1.0 • scale：可选，类型：float，默认为 1.0 • shift：可选，类型：float，默认为 0.0 【约束】 无限制
18	LRN	局部响应归一化层	【输入】 个数为 1，不支持常量输入 【参数】 • local_size：可选，类型：uint32，默认为 5 • alpha：可选，类型：float，默认为 1. • beta：可选，类型：float，默认为 0.75 • norm_region：可选，类型：枚举型，默认为 ACROSS_CHANNELS，（ACROSS_CHANNELS=0;WITHIN_CHANNEL = 1） • lrnk：可选，类型：float，默认为 1. • engine：可选，类型：枚举型（DEFAULT

序号	算子	含义	边界
			= 0;CAFFE = 1; CUDNN = 2） 【约束】 local_size >0，且必须为奇数 通道间： 当 local_size∈[1,15]时，lrnK>0.00001 且 beta>0.01，否则 lrnK 和 beta 为任意值；LrnK 和 alpha 不同时为 0；当 C 维度大于 680 时，local_size <640； 通道内： LrnK=1，local_size∈[1,15]，beta>0.01；
19	LSTM	长短期记忆网络	【输入】 个数为 2 个或者 3 个 X: 时间序列数据(T*N*Xt) Cont：序列连续性标志(T*N) Xs: 静态数据(N*Xt)，可选 【参数】 • num_output：可选，类型：uint32，默认为 0 • weight_filler：可选，类型：FillerParameter • bias_filler：可选，类型：FillerParameter • debug_info：可选，类型：bool，默认为 false • expose_hidden：可选，类型：bool，默认为 false 【约束】 此约束涉及中间变量计算，公式如列： a = (ALIGN(xt,16) + ALIGN(output,16)) * 16 * 2 * 2 b = (ALIGN(xt,16) + ALIGN(output,16)) * 16 * 4 * 2 * 2 c = use_projection ? ALIGN(ht,16) * ALIGN(output,16) * 2) : 0 d = 16 * ALIGN(ht,16) * 2 e = batchNum* 4 则，约束为： a + b + c <= 512 * 1024； d <= 128*1024 / 8； e <= 128*1024 / 32；
20	Normalize	标准化层	【输入】

序号	算子	含义	边界
			个数为 1 【参数】 • across_spatial：可选，类型：bool，默认为 true • scale_filler：可选，默认为 1.0 • channel_shared：可选，类型：bool，默认为 true • eps：可选，类型：float，默认为 1e-10 【约束】 1、eps 应大于 1e-7,并且小于等于 0.1+(1e-6)； 2、caffe 框架中的参数 across_spatial 目前只支持 true，按 channel 进行 norm 操作。
21	Permute	将输入维度按照给定模式进行重排	【输入】 个数为 1 【参数】 order: uint32；数组 【约束】 无约束
22	Pooling	池化层	【输入】 个数为 1 【参数】 • pool: 池化方法，可选，类型：枚举，取值：MAX=0，AVE=1，STOCHASTIC = 2，默认为 MAX • pad: 可选，类型：uint32，默认为 0 • pad_h: 可选，类型：uint32，默认为 0 • pad_w: 可选，类型：uint32，默认为 0 • kernel_size: 可选，类型：uint32，kernel_size 和 kernel_h/kernel_w 不能同时出现 • kernel_h: 可选，类型：uint32 • kernel_w: 可选，类型：uint32，kernel_h/kernel_w 必须同时存在 • stride: 可选，类型：uint32，默认为 1 • stride_h: 可选，类型：uint32 • stride_w: 可选，类型：uint32 • engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2

序号	算子	含义	边界
			• global_pooling: 可选，类型：bool，默认值为 false • ceil_mode: 可选，类型：bool，默认为 true • round_mode: 可选，类型：枚举，取值：CEIL=0，FLOOR=1；默认为 CEIL 【约束】 kernelH<=inputH+padTop+padBottom kernelW<=inputW+padLeft+padRight padTop<windowH padBottom<windowH padLeft<windowW padRight<windowW 只支持 globalpool 模式，此模式下的约束条件是： 1) outputH==1 && outputW==1 && kernelH>=inputH && kernelW>=inputW 2) inputH*inputW<=10000
23	Power	计算 y = （scale*x+shift）^ power	【输入】 个数为 1 【参数】 • power: 可选，类型：float，默认为 1.0 • scale: 可选，类型：float，默认为 1.0 • shift: 可选，类型：float，默认为 0.0 【约束】 power!=1 scale*x+shift>0
24	Prelu	激活函数	【输入】 个数为 1 【参数】 • filler: 可选 • channel_shared: 可选，类型：bool，默认为 false，是否跨 channel 共享斜率参数 【约束】 无限制
25	PriorBox	从目标预选框获取真实目标的位置	【输入】 个数为 1 【参数】 • min_size: 必须设置，最小框大小（以像

序号	算子	含义	边界
			素为单位） • max_size: 必须设置，最大框大小（以像素为单位） • aspect_ratio: 数组；类型：float；各种宽高比，重复的比率将被忽略，如果没有提供，使用默认比率 1 • flip: 可选，类型：bool，默认为 true；如果为 true，将翻转每个宽高比，例如，如果有宽高比‘r’，也将生成宽高比‘1.0/r’ • clip: 可选，类型：bool，默认为 false；如果为 true，将剪切先前的值，使其在[0, 1]范围内 • variance: 数组；调整先前 bbox 的方差 • img_size: 可选，类型：uint32，img_size 与 img_h/img_w 不能同时存在 • img_h: 可选，类型：uint32 • img_w: 可选，类型：uint32 • step: 可选，类型：float，step 与 step_h/step_w 不能同时存在 • step_h: 可选，类型：float • step_w: 可选，类型：float • offset: 类型：float，默认为 0.5 【约束】 只支持 ssd 网络。 输出维度：[n,2,检测框*4,1]。
26	Proposal	将预选框通过（proposal, score）排序，通过 nms 获取 topN proposal	【输入】 3 个输入（scores，bbox_pred，im_info） 【参数】 • feat_stride: 可选，类型：float • base_size: 可选，类型：float • min_size: 可选，类型：float • ratio: 数组；类型：float • scale: 数组；类型：float • pre_nms_topn: 可选，类型：int32 • post_nms_topn: 可选，类型：int32 • nms_thresh: 可选，类型：float 【约束】 只用于 fastercnn • ProposalParameter、PythonParameter 不能

序号	算子	含义	边界
			同时存在 • preTopK 范围为 1~6144 • postTopK 范围为 1~1024 • scaleCnt*ratioCnt 的最大值支持到 64 • nms_thresh:过滤框使用的阈值，0<nms_thresh<=1 • min_size:框的边长的最小值，所有小于此最小值的框将会被过滤掉 • feat_stride:在生成默认框时，指定两个相邻的框延 H 或 W 的步长 • base_size:用来生成默认框用到的参数，表示框的基本大小 • ratio & scale:生成默认框用到的参数 • imgH&imgW:输入到网络的图片高和宽，其值必须大于 0 • input 维度约束: clsProb：C=2*scaleCnt*ratioCnt bboxPred：C=4*scaleCnt*ratioCnt bboxPrior：N=clsProb.N，C=4*scaleCnt*ratioCnt imInfo：N=clsProb.N，C=3
27	PSROIPooling	位置敏感的区域池化	【输入】 2 个输入 【参数】 • spatial_scale: 必须配置，类型: float • output_dim: 必须配置，类型: int32，输出通道数 • group_size: 必须配置，类型: int32，编码位置敏感分数图的组数 【约束】 用于 RFCN 网络 输入 Roi 框的坐标信息为[roiN, roiC, roiH, roiW]，格式范围是 1<= roiN <=65535，roiC == 5，roiH == 1，roiW == 1； 1）输入的 featuremap 维度为[xN,xC,xH,xW] pooledH==pooledW==groupSize <= 128 [pooledH pooledW]表示 pool 框的长宽 输出的格式 y[yN, yC, yH, yW] 2）poolingMode == avg pooling， pooledH ==

序号	算子	含义	边界
			pooledW == groupSize， pooledH <= 128，spatialScale > 0，groupSize > 0，outputDim > 0； 3）1<=xN<=65535， roisN % xN == 0； 4）xHW = xH * xW，pooledHW = pooledH * pooledW，HW_LIMIT = (32*1024－8*1024) / 32， xH >= pooledH， xW >= pooledW，xHW >= pooledHW， xHW / pooledHW <= HW_LIMIT； 5）多 batch 场景时，每个 batch 的 roi 框个数相同，且 roi 的 batch 排列顺序与 feature 相同。
28	Relu	激活函数，同时包含普通的 relu 和 leaky relu，可通过参数指定	【输入】 个数为 1 【参数】 • negative_slope: 可选，类型：float，默认为 0 • engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2 【约束】 无限制
29	Reorg	实时物体检测	【输入】 一个输入 【参数】 • stride: 可选，类型：uint32，默认为 2 • reverse: 可选，类型：bool，默认为 false 【约束】 只用于 YOLOV2
30	Reshape	改变输入维度	【输入】 个数为 1 【参数】 • shape: 常量，类型：int64 或 int • axis: 可选，类型：int32，默认为 0 • num_axes: 可选，类型：int32，默认为-1 【约束】 无限制
31	Reverse	逆转	【输入】 一个输入 【参数】

序号	算子	含义	边界
			axis: 可选，类型：int32，默认为 1；控制需翻转的数据轴，内容的布局不会被颠倒 【约束】 无限制
32	ROIAlign	一种区域特征聚集的方式	【输入】 至少有两个输入 【参数】 pooled_h: 可选，类型：uint32，默认为 0 pooled_w: 可选，类型：uint32，默认为 0 spatial_scale: 可选，类型：float，默认为 1 sampling_ratio: 可选，类型：int32，默认为-1 【约束】 主要用于 maskrcnn FeatureMap（特征图）约束： 1、H * W <=2464； 2、C <= 1152； 3、((C-1)/128+1)*pooledW <= 92； （感兴区域）约束： C=5(caffe) ，H=1，W=1； 4、samplingRatio*pooledW <= 128 且 samplingRatio*pooledH <= 128； 5、H >= pooledH, W >= pooledW。
33	ROIPooling	将“候选框”映射到特征图上	【输入】 至少有两个输入 【参数】 • pooled_h: 必须设置，类型：uint32，默认为 0 • pooled_w: 必须设置，类型：uint32，默认为 0 • spatial_scale: 必须设置，类型：float；默认为 1；乘法空间比例因子将 ROI 坐标从其输入比例转换为 pool 时使用的比例 【约束】 主要用于 fasterrcnn 1、只支持 PooledH 与 PooledW 相等的场景； 2、支持输入 feature map 最大 HW：54 * 72，输出最大 HW：16*16

序号	算子	含义	边界
34	Scale	out=alpha*Input+beta	【输入】 两个输入，input 的 dim 为 4 【参数】 • axis: 可选，类型：int32，默认为 1，仅支持 axis 为 1 或者-3 • num_axes: 可选，类型：int32，默认为 1 • filler: 可选；filler 被忽略，除非只给一个 bottom 且 scale 是学习参数 • bias_term: 可选，类型：bool，默认为 false；是否也学习 bias（相当于 ScaleLayer + BiasLayer，但可能更有效率），使用 bias_filler 初始化 • bias_filler: 可选，默认为 0 【约束】 scale,bias 的 shape 只支持（n,c,1,1），且 c 维度与 input 的 c 维度相等；
35	ShuffleChannel	帮助信息在特征通道交叉流动	【输入】 1 个输入 【参数】 • group: 可选，类型：uint32，默认为 1 【约束】 无限制
36	Sigmoid	激活函数	【输入】 一个输入 【参数】 engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2 【约束】 无限制
37	Slice	将输入分解成多个输出	【输入】 一个输入 【参数】 • slice_dim: 可选，类型：uint32，默认为 1；axis 和 slice_dim 不能同时存在 • slice_point: 数组；类型：uint32 • axis: 可选，类型: int32，默认为 1（表示沿 channel 拼接）； 【约束】

序号	算子	含义	边界
			无限制 【输出】 无限制
38	Softmax	归一化逻辑函数	【输入】 一个输入 【参数】 • engine: 可选，取值：DEFAULT=0，CAFFE=1，CUDNN=2 • axis: 可选，类型：int32，默认为 1；表示沿哪个 axis 作 softmax 【约束】 根据分类的轴不同，计算边界分别为 • axis=1 时，c<=（(128*1024/4)-8*1024-256）/2； • axis=0 时，n<=(56*1024-256)/2； • axis=2 时，W=1, 0<h<（512*1024/32）； axis=3 时，0<W<（512*1024/32）
39	SSDDetection Output	SSD 网络检测输出	【输入】 三个输入 【参数】 • num_classes: 必选，类型：int32，要预测的类数 • share_location: 可选，类型: bool，默认为 true（表示不同类间共享 bounding box） • background_label_id: 可选，类型：int32，默认为 0 • nms_param: 可选，非最大抑制 • save_output_param: 可选，用于保存检测结果 • code_type: 可选，默认为 CENTER_SIZE • variance_encoded_in_target: 可选，类型：bool，默认为 true；如果为 true，方差编码在目标中，否则需要相应地调整预测偏移量 • keep_top_k: 可选，类型：int32，在 nms 步骤后每个图像要保留的总 bbox 数； • confidence_threshold: 可选，类型：float，仅考虑置信度大于阈值的检测；如果没有设置，考虑所有的 box

序号	算子	含义	边界
			• nms_threshold: 可选，类型：float • top_k: 可选，类型：int32 • boxes: 可选，类型：int32，默认为 1 • relative: 可选，类型：bool，默认为 true • objectness_threshold，可选，类型：float，默认为 0.5 • class_threshold: 可选，类型：float，默认为 0.5 • biases: 数组 • general_nms_param: 可选 【约束】 只支持 SSD 网络 preTopK 和 postTopK 的取值范围当前仅支持 1~1024； shareLocation 仅支持 true； nmsEta 仅支持 1； numClasses 支持的范围是 1~2048； code_type 仅支持 CENTER_SIZE； nms_threshold 和 confidence_threshold 的范围为 0.0~1.0
40	Tanh	激活函数	【输入】 一个输入 【参数】 engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2 【约束】 无限制
41	Upsample	maxpool 的反向传播过程	【输入】 两个输入 【参数】 scale: 可选，类型：int32，默认为 1 【约束】 无限制
42	YOLODetectionOutput	YOLO 网络检测输出	【输入】 四个输入 【参数】 • num_classes: 必选，类型：int32，要预测

序号	算子	含义	边界
			的类数 • share_location: 可选，类型: bool，默认为 true（表示不同类间共享 bounding box） • background_label_id: 可选，类型：int32，默认为 0 • nms_param: 可选，非最大抑制 • save_output_param: 可选，用于保存检测结果 • code_type: 可选，默认为 CENTER_SIZE • variance_encoded_in_target: 可选，类型：bool，默认为 true；如果为 true，方差编码在目标中，否则需要相应地调整预测偏移量 • keep_top_k: 可选，类型：int32，在 nms 步骤后每个图像要保留的总 bbox 数； • confidence_threshold: 可选，类型：float，仅考虑置信度大于阈值的检测；如果没有设置，考虑所有的 box • nms_threshold: 可选，类型：float • top_k: 可选，类型：int32 • boxes: 可选，类型：int32，默认为 1 • relative: 可选，类型：bool，默认为 true • objectness_threshold，可选，类型：float，默认为 0.5 • class_threshold: 可选，类型：float，默认为 0.5 • biases: 数组 • general_nms_param: 可选 【约束】 只用于 YOLOV2 classNUm<10240；anchorBox <= 8； W <= 768； yolodetectionoutput 的上层必须为 yoloregion 算子。

2.3 Tensorflow 算子边界

序号	Python API	C++ API	边界
1	tf.abs	Abs	【参数】 • x: Tensor 或稀疏 Tensor，类型： float32 • name: 操作的名称，可选参数 【约束】 无限制 【输出】 返回 x 的绝对值，Tensor 或稀疏 Tensor，尺寸与类型同 x
2	tf.add	Add	【参数】 • x: 输入，一个 Tensor，类型：float32、int32 • y: 输入，一个 Tensor，类型同 x；两个输入为常量时，一个输入为标量 • name: 此操作的名称（可选） 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐），目前支持以下几种广播场景： NHWC+ NHWC, NHWC+scalar 或者 NHWC +1 1 1 1， 或者 NHWC+W 和 HWC+W 和 HW+W(备注, W 维度做广播)， 或者 NCHW + NH1C 和 HWC + H1C 和 HW + H1， 还有 HWC + 1 WC(备注，H 维度做广播)； 说明：两个 Tensor 的输入顺序可以互换。 【输出】 一个 Tensor，类型同 y
3	tf.batch_to_space_nd	BatchToSpaceND	【参数】 • input：一个 Tensor，是 N 维的并且具有形状 input_shape = [batch] + spatial_shape + remaining_shape,其中 spatial_shape 有 M 维度；支持数据类型为：uint8，int8，int16，uint16，int32，int64，float32 • block_shape：一个 Tensor；必须是以下类型之一：int32,int64；1-D,shape 为[M],所有值必须>=1 • crops：一个 Tensor；必须是以下类型之一：int32,int64；二维, shape 为[M, 2],所有值必须>=0. 【约束】 • blockShape 和 crops 的元素值数据类型必须是 int32，当 Tensor 维数为 4 时：blockShape 的长度必须等于 2，crops

序号	Python API	C++ API	边界
			的长度必须等于 4. • blockShape 元素的大小必须要大于等于 1，crops 元素值的大小必须大于等于 0，crops 数组的大小必须满足 • crop_start[i] + crop_end[i] < block_shape[i] * input_shape[i+1]； 【输出】 一个 Tensor，与 images 具有相同的类型
4	tf.cast	Cast	【输入】 类型：float32, int32, bool, int64, int16, int8, uint8, uint16, double 【参数】 • x: Tensor 或 sparseTensor 或 indexedSlices • dtype: 目标类型，同 x 支持的数据类型 • name: 名称（可选） 【约束】 无限制 【输出】 Tensor 或 sparseTensor 或 indexedSlices，同输入的 dtype、shape
5	tf.clip_by_value	ClipByValue	【参数】 • t: Tensor • clip_value_min: clip 最小值 • clip_value_max：clip 最大值 • name: string;名称(可选) 【约束】 min 值要小于或者等于 max 值 【输出】 输出 Tensor，返回值范围【clip_value_min。clip_value_max】
6	tf.concat	Concat	【参数】 • values: 输入，包含 Tensor 对象的列表或单个 Tensor，除要拼接的维度外，其他维度上的值要一致 • axis: 0-D Tensor，类型：int32，指定要拼接的维度，范围在[-rank(values), rank(values)]；python 中，索引以 0 开始，axis 为正值时，表示对第 axis 维拼接；axis 为负值时，对第 axis+rank(values)维拼接； 【约束】 • 输入的 Tensor，除了进行 concat 的维度外，其他维度的 size 必须相等 • 输入的的 Tensor 个数范围属于[1,1000]

序号	Python API	C++ API	边界
			【输出】 一个 Tensor，为输入 Tensors 拼接后的结果
7	tf.constant	Const	【参数】 • value: 常量或常量数组 • dtype: 指定数据类型 • shape: 维度尺寸(可选)，用于输出 • name: string;名称(可选) • verify_shape: 布尔值(可选)，默认 False 【约束】 无限制 【输出】 1 个常量 Tensor
8	tf.depth_to_space	DepthToSpace	【参数】 • input： 输入 Tensor 数据类型: `float32`,`int64`,`int32`,`uint8`,`int8`, • block_size： 标量，整型，值>=2 • data_format：数据类型：string 值: `"NHWC", "NCHW", "NCHW_VECT_C"`. 默认值是 `"NHWC" • name: string;名称(可选) 【约束】 blockSize 必须大于等于 1，且 blockSize* blockSize 必须能被 C 整除 【输出】 Tensor，输出类型 input
9	tf.equal	Equal	【参数】 • x: Tensor • y: Tensor 数据类型同 x • name: string;名称(可选) 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上右对齐的情况下，xdim[i]和 ydim[i]只能相同或者一方为 1 或者一方缺失 【输出】 输出 Tensor，数据类型 bool
10	tf.exp	exp	【参数】 • x: 一个输入 Tensor，类型 float32, double；

序号	Python API	C++ API	边界
			• name: 此操作的名称（可选） 【约束】 无限制 【输出】 一个 Tensor，类型同 x
11	tf.expand_dims	ExpandDims	【参数】 • input: 输入，一个 Tensor • axis: 0-D（标量），指定扩展 input 形状的维度索引； • name: 输出 Tensor 的名称 • dim: 0-D（标量），相当于 axis，被弃用 【约束】 无限制 【输出】 一个 Tensor，与 input 数据相同，shape 增加了一维（值为 1）
12	tf.extract_image_patches	ExtractImagePatches	【参数】 • images: 一个 Tensor，支持数据类型：float32， int32，int64，uint8，int8，uint16，int16；4-D Tensor shape：[batch, in_rows, in_cols, depth] • ksizes: 一个整形 list，长度>=4 • strides: 一个整形 list，必须是：[1, stride_rows, stride_cols, 1] • rate: 一个整形 list，必须是：[1, rate_rows, rate_cols, 1] • padding: string，取值：“VALID”或者“SAME”，“VALID”表示所取的 patch 区域必须完全包含在原始图中."SAME"表示取超出原始图像的部分，以 0 填充该部分 • name: 名称(可选) 【约束】 无约束 【输出】 一个 Tensor，与 images 具有相同的类型
13	tf.fake_quant_with_min_max_vars	FakeQuantWithMinMaxVars	【参数】 • inputs : 输入 Tensor；数据类型：`float32`， • min: Tensor, 数据类型：`float32` • max: Tensor, 数据类型：`float32` • num_bits：标量，整型，默认值`8` • narrow_range：bool（可选），默认值`False` • name: string;名称(可选)

序号	Python API	C++ API	边界
			【约束】 -65504<=min<=65504，-65504<=max<=65504。 【输出】 输出 Tensor，数据类型：`float32`
14	tf.fill	Fill	【参数】 • dims: 1-D Tensor, 数据类型：`int32`, • value: 变量，数据类型：`int32`,`float32` • name: string;名称(可选) 【约束】 支持 Constant、GivenTensor、Range、Diagonal、Gaussian、MSRA、Uniform、UniformInt、UniqueUniform、XavierFill 这些填充模式,在 Uniform 填充、UniformInt 填充、UniqueUniform 填充、xavier 填充时，生成的数值区间最大范围介于[min,max)之间 【输出】 1 个 Tensor，类型同 value 数据类型
15	tf.floormod	FloorMod	【参数】 • x: 一个 Tensor，支持数据类型：float32，int32 • y: 一个 Tensor，与 x 具有相同类型 • name: 名称(可选) 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上，dim[x]只能相同或者一方为 1 或者一方缺失。 【输出】 一个 Tensor，与 x 具有相同的类型
16	tf.gather	Gather GatherV2	【参数】 • params: 一个 Tensor ,维度必须大于 `axis + 1` • indices: 一个 Tensor, 数据类型`int32`,`int64`范围[0, params.shape[axis]) • axis: 输出 Tensor 数据类型, `int32` 或 `int64`，指定 indices 选取的维度，rank=0 • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 Tensor,输出数据类型于 params 相同
17	tf.gather_nd	GatherNd	【参数】

序号	Python API	C++ API	边界
			• params: 一个 Tensor ,维度必须大于 `axis + 1` • indices: 一个 Tensor, 数据类型`int32`,`int64` • name: string;名称(可选) 【约束】 indices 最后一维的大小不能超过 params 的维数 indices 最后一维中的元素对应着 params 中的一个维度上的坐标，必须满足坐标规则 indices 中对应维度上的坐标不能超过维度的大小 【输出】 1 个 Tensor,输出数据类型于 params 相同
18	tf.greater	Greater	【参数】 • x: 输入，一个 Tensor • y: 标量 • name: 此操作的名称（可选） 【约束】 支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上，dim[x]只能相同或者一方为 1 或者一方缺失。 【输出】 一个 Tensor，类型：bool
19	tf.image.crop_and_resize	CropAndResize	【参数】 • image: 4-D Tensor; 数据类型： `float32`， `int8`, `int32`,`int64`,shape: `[num_boxes, 4]` • boxes： 2-D Tensor;，数据类型: `float32` shape：`[num_boxes]` • box_ind：1-D Tensor，数据类型：`int32` • crop_size：1-D Tensor ,包含 2 个元素 数据类型：`int32`， • method: string,表示插值方法, 值为"bilinear"(默认), "nearest" ： • name: string;名称(可选) 【约束】 无限制 【输出】 • Tensor，数据类型: `float32`
20	tf.image.resize_bilinear	ResizeBilinear	【参数】 • images: 4 维非常量 Tensor，[batch, height, width, channels]；类型：float32 • size: 1 维 Tensor，常量，2 个元素（新的高宽）

序号	Python API	C++ API	边界
			• method: ResizeMethod.BILINEAR • align_corners: bool 值，默认为 False；如果为 True，输入和输出的 4 个角像素的中心对齐，保留角像素处的值 【约束】 （outputH*outputW）/（inputH*inputW） > 1/7 【输出】 shape 同输入，类型：float
21	tf.image.resize_nearest_neighbor	ResizeNearestNeighbor	【参数】 • images: 4 维 Tensor，[batch, height, width, channels]；3 维 Tensor，[height, width, channels]；类型：float32 • size: 1 维 Tensor，常量，2 个元素（新的高宽） • method: ResizeMethod.NEARESTNEIGHBOR • align_corners: bool 值，默认为 False；如果为 True，输入和输出的 4 个角像素的中心对齐，保留角像素处的值 【约束】 无限制 【输出】 shape 同输入，类型：float
22	tf.invert_permutation	InvertPermutation	【参数】 • x: 1-D Tensor 类型：`int32`,`int64`, • name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，类型同 x
23	tf.keras.backend.hard_sigmoid	Hardsigmoid	【参数】 x： 输入 Tensor 【约束】 无限制 【输出】 输出 Tensor，返回值：`0` if `x < -2.5`, `1` if `x > 2.5`. 当 `-2.5 <= x <= 2.5`，返回 `0.2 * x + 0.5`.
24	tf.keras.layers.ThresholdedReLU	ThresholdedReLU	【参数】 theta：标量>=0，类型:float32 【约束】 无限制

序号	Python API	C++ API	边界
			【输出】 Tensor
25	tf.log	Log	【参数】 • x: 输入 Tensor，类型：float32 • name: 名称（可选） 【约束】 无限制 【输出】 输出 Tensor，类型同 x
26	tf.math.acos	Acos	【参数】 • x： 输入 Tensor 类型`float32`,`int32`,`int64` • name: string;名称(可选) 【约束】 输入数据范围（-1<=x<=1），输出数据范围（0<=y<= π ） 【输出】 Tensor，类型同 x
27	tf.math.acosh	Acosh	【参数】 • x： 输入 Tensor 类型`float32` • name: string;名称(可选) 【约束】 无约束 【输出】 Tensor，数据类型同 x
28	tf.math.argmax	ArgMax	【参数】 • input: 一个 Tensor,支持类型:`int8`,`uint8`,`int16`, `uint16`, `int32`,`int64`,`float32` • axis: 一个 Tensor, 类型`int32`,`int64` • out_type 输出 Tensor 类型 `int32` 或 `int64` (可选,默认为`int64`) • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 Tensor,输出类型为 out_type
29	tf.math.asin	Asin	【参数】 • x： 输入 Tensor 类型`float32`,`int32`,`int64`

序号	Python API	C++ API	边界
			• name: string;名称(可选) 【约束】 输入数据范围（-1<=x<=1），输出数据范围（-π/2<=y<=π/2）。 【输出】 Tensor，类型同 x
30	tf.math.asinh	Asinh	【参数】 • x： 输入 Tensor 类型`float32` • name: string;名称(可选) 【约束】 无约束 【输出】 Tensor，类型同 x
31	tf.math.atan	Arctan	【参数】 • x： 输入 Tensor 数据类型`float32`,`int32`,`int64` • name: string;名称(可选) 【约束】 输入数据范围（-65504<=x<=65504），输出数据范围（-π/2<y<π/2）。 【输出】 Tensor，数据类型同 x
32	tf.math.atanh	Atanh	【参数】 • x： 输入 Tensor 数据类型`float32`,`int32`,`int64` • name: string;名称(可选) 【约束】 输入数据范围：x∈(-1,1) 【输出】 Tensor，数据类型同 x
33	tf.math.cosh	Cosh	【参数】 • x： 输入 Tensor 数据类型: `float32` • name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，输出类型同 x
34	tf.math.floordiv	FloorDiv	【参数】

序号	Python API	C++ API	边界
			• x: 一个 Tensor ,数据类型：`float32`,`int32`, • y: 一个 Tensor, 分母，数据类型：` float32`,` int32` • name: string;名称(可选) 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上，dim[x]只能相同或者一方为 1 或者一方缺失 【输出】 1 个 Tensor ， floor（x/y）
35	tf.math.greater_equal	Greaterequal	【参数】 • x： 输入 Tensor 数据类型: `float32`,`int64`,`int32`,`uint8`,`uint16`,`int8`,`int16`, • y： 输入 Tensor 数据类型同 x, • name: string;名称(可选) 【约束】 输入数据范围（-65504<=x<=65504） 【输出】 Tensor，输出类型 bool
36	tf.math.less	Less	【参数】 • x： 输入 Tensor 数据类型: `float32`,`int64`,`int32`,`uint8`,`uint16`,`int8`,`int16`, • y： 输入 Tensor 数据类型同 x, • name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，输出类型 bool
37	tf.math.logical_and	LogicalAnd	【参数】 • x: Tensor 数据类型 bool • y: Tensor 数据类型 bool • name: string;名称(可选) 【约束】 BroadCast 只支持如下几种维度的广播，NHWC 和 [1,1,1,1],[N,H,W,C],[N,H,W,1],[1,H,W,C],[N,1,1,C] 【输出】 输出 Tensor，数据类型 bool
38	tf.math.logical_not	LogicalNot	【参数】

序号	Python API	C++ API	边界
			• x: Tensor 数据类型 bool • name: string;名称(可选) 【约束】 无限制 【输出】 输出 Tensor，数据类型 bool
39	tf.math.logical_or	LogicalOr	【参数】 • x： 输入 Tensor 数据类型:`bool` • y： 输入 Tensor 数据类型:`bool` • name: string;名称(可选) 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上右对齐的情况下，xdim[i]和 ydim[i]只能相同或者一方为 1 或者一方缺失。 【输出】 Tensor，数据类型: `bool`
40	tf.math.maximum	Maximum	【参数】 • x: Tensor 数据类型：`int32`，`int64`，`float32` • y: Tensor 数据类型同 x • name: string;名称(可选) 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一纬度上，dim[x]只能相同或者一方为 1 或者一方缺失 【输出】 输出 Tensor，返回值（x > y ? x : y） 数据类型同 x
41	tf.math.minimum	Minimum	【参数】 • x: Tensor 数据类型：`int32`，`int64`，`float32` • y: Tensor 数据类型同 x • name: 名称（可选） 【约束】 广播场景只支持下面两种 NHWC+scaler,NHWC+NHWC 【输出】 输出 Tensor，返回值（x < y ? x : y） 数据类型同 x

序号	Python API	C++ API	边界
42	tf.math.negative	Neg	【参数】 • x： 输入 Tensor 数据类型:`float32`,`int64`,`int32`, • name: string;名称(可选) 【约束】 输入数据范围（-65504<=x<=65504），输出数据范围（-65504<=y<=65504） 【输出】 Tensor，输出= -x
43	tf.math.pow	Power	【参数】 • x: 一个 Tensor，支持数据类型：float32 • y: 一个 Tensor，支持数据类型：float32 • name: 名称(可选) 【约束】 power!=1 scale*x+shift>0。 【输出】 一个 Tensor
44	tf.math.reciprocal	Reciprocal	【参数】 • x： 输入 Tensor 数据类型`float32` • name: string;名称(可选) 【约束】 不支持输入数据中包含 0 【输出】 Tensor，数据类型同 x
45	tf.math.reduce_all	All	【参数】 • input_tensor： 输入 Tensor；数据类型: bool • axis： 指定 reduce 的维度 • keepdims：bool 值 • name: string;名称(可选) • reduction_indices：axis 旧的别名(不推荐) • keep_dims：keepdims 别名（不推荐） 【约束】 无约束 【输出】 Tensor，数据类型同 input_tensor

序号	Python API	C++ API	边界
46	tf.math.reduce_min	ReduceMin	【参数】 • input_tensor： 输入 Tensor 数据类型:`float32`,`int64`,`int32`,`uint8`,`uint16`,`int8`,`int16`, • axis: reduce 的维度轴向 • keepdims：标量，bool 类型， • name: string;名称(可选) • reduction_indices：axis 旧的别名 • keep_dims：keepdims 别名（不推荐） 【约束】 当输入的 Tensor 维数等于 4 时：输入 axis={3,{1,2,3}}，keepDims=true，H*W*16*2 <= 16*1024； 当输入的 Tensor 维数等于 2 时，输入 axis={1,{1}}，keepDims=true，H*W*CEIL(C,16)*16*2 <= 16*1024 【输出】 Tensor，数据类型同 input_tensor
47	tf.math.reduce_prod	Prod	【参数】 • input_tensor： 输入 Tensor • axis : reduce 维度 • keepdims： bool 是否需要保存 reduce 维度 • name: string;名称(可选) • reduction_indices： 参数 axis 旧的别名（不推荐） • keep_dims：参数 keepdims 别名（不推荐） 【约束】 无约束 【输出】 Tensor，raduce 后的 Tensor
48	tf.math.rint	Rint	【参数】 • x： 输入 Tensor；数据类型: `float32`,`int64`,`int32`, `uint8`,`int8` • name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，输出类型同 x，输出 shape 同 x
49	tf.math.round	Round	【参数】 • x： 输入 Tensor；数据类型: `float32`,`int64`,`int32`

序号	Python API	C++ API	边界
			• name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，输出类型同 x，输出 shape 同 x
50	tf.math.sinh	Sinh	【参数】 • x： 输入 Tensor 数据类型: `float32` • name: string;名称(可选) 【约束】 无限制 【输出】 Tensor，输出类型同 x
51	tf.math.sqrt	Sqrt	【参数】 • x： 输入 Tensor；数据类型: `float32` • name: string;名称(可选) 【约束】 无约束 【输出】 Tensor，数据类型同 x
52	tf.math.squared_difference	Squared_difference	【参数】 • x： 输入 Tensor; 数据类型: `float32`,`int64`,`int32` , • y： 输入 Tensor 数据类型同 x, • name: string;名称(可选) 【约束】 广播模式只支持下列场景： 第一个的 tensor_format 是 NCHW，另一个的 dim{}可以是 [1,1,1,1]，[N,C,H,W]，[N,1,H,W]，[1,C,H,W]，[N,C,1,1],[1,C,1,1],[1,1,H,W],[N,1,1,1]这几种情况。 【输出】 Tensor，数据类型同 x
53	tf.math.tan	Tan	【参数】 • x： 输入 Tensor 数据类型`float32`,`int32`,`int64` • name: string;名称(可选) 【约束】 无限制 【输出】

序号	Python API	C++ API	边界
			Tensor，数据类型同 x
54	tf.math.top_k	TopKV2	【参数】 • input: Tensor，>=1-D,最后一个维度大小必须大于 k; 数据类型： `float32`k： 标量，>=1; 数据类型：`int32` • sorted：bool • name: string;名称(可选) 【约束】 K 一定要以常量传入 【输出】 • values: Tensor 返回最后维度上的 K 个最大向量 • indices: Tensor，values 在 input 中的索引位置
55	tf.matmul	MatMul	【参数】 • a: 非常量 Tensor，rank>=2，类型：float32 • b: 常量 Tensor，类型与 rank 同 a • transpose_a: 如果属性为 True，a 在乘法前做转置 • transpose_b: 如果属性为 True，b 在乘法前做转置；transpose_a 属性为 False，transpose_b 属性也为 False • adjoint_a: 如果属性为 True，a 在乘法前被共轭和转置 • adjoint_b: 如果属性为 True，b 在乘法前被共轭和转置 • a_is_sparse: 如果属性为 True，a 被看做是稀疏矩阵 • b_is_sparse: 如果属性为 True，b 被看做是稀疏矩阵 • name: 可选参数 【约束】 • weight 的转置属性为 false • 不能支持两个 Tensor 相乘，只能支持一个 Tensor 乘以一个常量 【输出】 一个 Tensor，类型同 a 与 b
56	tf.multinomial	Multinomial	【参数】 • logits：2-D Tensor，shape `[batch_size, num_classes]` • num_samples： 标量 抽样个数 • seed: 随机数种子：数据类型： `int32`,`int64`, • name: string;名称(可选) • output_dtype：输出 Tensor 数据类型：整型默认`int64` 【约束】 seed 为 0 时产生随机数是动态的

序号	Python API	C++ API	边界
			输出数据行数等于输出数据的行，输出数据的列数等 num_samples 【输出】 Tensor，数据类型同 output_dtype
57	tf.multiply	Multiply	【参数】 • x: 输入，一个 Tensor，类型：float32、int32 • y: 输入，一个 Tensor，类型同 x；两个输入为常量时，维度必须一致，为标量或者 1-D Tensor • name: 此操作的名称（可选） 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐）， 目前支持以下几种广播场景： NHWC+ NHWC, NHWC+scalar 或者 NHWC +1 1 1 1， 或者 NHWC+W 和 HWC+W 和 HW+W(备注, W 维度做广播)， 或者 NCHW + NH1C 和 HWC + H1C 和 HW + H1， 还有 HWC + 1 WC(备注，H 维度做广播)； 说明：两个 Tensor 的输入顺序可以互换。 【输出】 一个 Tensor
58	tf.nn.avg_pool	AvgPool	【参数】 • value: 4-D Tensor，格式：[batch, height, width, channels]，数据类型：float32 • ksize: 包含四个 int 的列表或元组，其中每个值对应相应维度的窗口大小 • strides: 包含四个 int 的列表或元组，其中每个值对应相应维度的滑动步长 • padding: 类型：string，值必须为’VALID’ 或’SAME’ • data_format: 类型：string，值为'NHWC' （默认值）或 'NCHW' • name: 可选参数，类型：string 【约束】 kernelH<=inputH+padTop+padBottom kernelW<=inputW+padLeft+padRight padTop<windowH padBottom<windowH

序号	Python API	C++ API	边界
			padLeft<windowW padRight<windowW 只支持 globalpool 模式，此模式下的约束条件是： 1) outputH==1 && outputW==1 && kernelH>=inputH && kernelW>=inputW 2) inputH*inputW<=10000 【输出】 一个 Tensor，数据类型与 value 相同
59	tf.nn.bias_add	BiasAdd	【参数】 • value: 输入，一个 Tensor，非常量 • bias: 1-D Tensor，常量，尺寸与 value 的最后一维一致；除非 value 为量化类型，否则类型需同 value • data_format: 类型：string，’NHWC’或’NCHW’ • name: 此操作的名称（可选） 【约束】 1、C < 10000； 2、input 和 bias 的数据排布要一致； 3、当在 c 通道上加 bias 时，input 和 bias 的 C 维度大小要一致。 【输出】 一个 Tensor，类型同 value
60	tf.nn.conv2d	Conv2D	【参数】 • value: 4-D Tensor，格式：[batch, height, width, channels]，数据类型：float32 • filter: 一个常量 Tensor，数据类型与维度与 value 相同，[filter_height, filter_width, in_channels, out_channels] • strides: 非空，包含四个 int 的列表或元组，其中每个值对应相应维度的滑动步长 • padding: 非空，类型：string，值必须为’VALID’ 或’SAME’ • use_cudnn_on_gpu: 类型: bool，默认为’True’ • data_format: 非空，类型：string，值为'NHWC'（默认值）或 'NCHW' • dilations: 可选参数，一个 int 列表（长度为 4）；默认为 [1,1,1,1]，对应输入的每一维；如果 k>1，相应维度做 filter 间跳过 k-1 个单元；维度顺序由 data_format 决定；dilations 的 batch 与 depth 维度上的值必须是 1 • name: 可选参数，类型：string 【约束】 • (inputW + padWHead + padWTail) >= (((FilterW- 1) * dilationW) + 1)

序号	Python API	C++ API	边界
			• (inputW + padWHead + padWTail) /StrideW + 1 <= INT32_MAX • (inputH + padHHead + padHTail) >= (((FilterH- 1) * dilationH) + 1) • (inputH + padHHead + padHTail) /StrideH + 1 <= INT32_MAX • 0 <= Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 <= dilationsize < 256 StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1) 【输出】 Tensor，数据类型与 Value 相同
61	tf.nn.conv2d_transpose	Conv2DBackpropInput	【参数】 • value: 输入，4-D Tensor，数据格式: NHWC（[batch, height, width, in_channels]）或 NCHW（[batch, in_channel, height, width]） • filter: 4-D Tensor，常量，shape: [height, width, output_channels, in_channels] • output_shape: 1-D Tensor，表示输出的 shape • strides: int 型列表，非空，输入的每个维度的滑动窗口的步长 • padding: 类型：string，值为’VALID’或’SAME’，非空 • data_format: 类型: string，’NHWC’或’NCHW’，非空 • name: 输出名（可选） 【约束】 group = 1 dilation = 1 filterH - padHHead - 1 >= 0 filterW - padWHead - 1 >= 0 还有一条约束涉及中间变量，公式如下： 1、a = ALIGN(filter_num,16) * ALIGN(filter_c,16) * filter_h * filter_w * 2 ； 如果 ALIGN(filter_c,16)%32 = 0，a = a/1； 2、conv_input_width=(反卷积输入 w - 1) * strideW + 1； 3、b = （conv_input_width） * filter_h * ALIGN(filter_num,16) * 2 * 2； 4、a + b <= 512KB。 【输出】 一个 Tensor，类型同 value
62	tf.nn.depthwise	DepthwiseCo	【参数】

序号	Python API	C++ API	边界
	_conv2d	nv2dNative	• input: 4 维 • filter: 4 维，常量，数据格式：[filter_height, filter_width, in_channels, channel_multiplier] • strides: 输入的每个维度的滑动窗口的步长，属性必须是 list(int)，且大小为 4 • padding: 类型：string，值为’VALID’或’SAME’ • rate：1 维，大小为 2；空洞卷积中在 height 和 width 维度上对输入值进行采样的扩张率；如果值大于 1，则步长的所有值必须为 1 • data_format: 输入的数据格式，可以是 NHWC（默认）或 NCHW • name: 名称（可选） 【约束】 • filterN=inputC=group • StrideW<=(inputW + padW)-((filterW - 1) * dilationW) + 1) 【输出】 4 维 Tensor，形状与 data_format 一致，例如，对于 NHWC 格式，形状是[batch, out_height, out_width, in_channels * channel_multiplier]
63	tf.nn.elu	Elu	【参数】 • features: 输入 Tensor，非常量 • name: 名称（可选） 【约束】 无限制 【输出】 输出 Tensor，类型同 features
64	tf.nn.fused_batch_norm	FusedBatchNorm	【参数】 • x: 输入，4-D Tensor，类型：float32 • scale: 1-D Tensor，用于缩放 • offset: 1-D Tensor，偏差 • mean: 1-D Tensor，用于推理总体均值 • variance: 1-D Tensor，用于推理总体方差 • epsilon: 在 x 的方差中添加的一个小的浮点数 • data_format: x 的数据格式，值为'NHWC'（默认）或 'NCHW' • is_training: bool 值，用于指定操作是用于训练还是推断 • name: 操作的名称，可选参数 【约束】 scale, bias,mean,var 的 shape 只支持(1,C,1,1)，且 c 维度与 input

序号	Python API	C++ API	边界
			的 c 维度相等。 【输出】 • y: 标准化、缩放、偏移 x 的 4-D Tensor • batch_mean: 1-D Tensor，表示 x 的均值 • batch_var: 1-D Tensor，表示 x 的方差
65	tf.nn.l2_normalize	L2Normalize	【参数】 • x： 输入 Tensor；数据类型: boolean • axis： 指定 normalize 的维度轴向； 如果 format 为 NCHW，则 axis 必须为 1； 如果 format 为 NHWC，则 axis 必须为 3 • epsilon：规范化的下限值.如果 norm <sqrt(epsilon),将使用 sqrt(epsilon)作为除数. • name: string;名称(可选) • dim：axis 旧的别名(不推荐) 【约束】 H*W*2 < 128*1024/4； 【输出】 Tensor，数据类型同 x
66	tf.nn.leaky_relu	/	【参数】 • features: 非常量输入，Tensor，表示预激活的值 • alpha: x<0 时激活函数的斜率 • name: 此操作的名称（可选） 【约束】 无限制 【输出】 激活值
67	tf.nn.leaky_relu	LeakyRelu	【参数】 • features: 一个 Tensor，支持数据类型： float32 • alpha: x <0 时激活函数的斜率. • name: 名称(可选) 【约束】 无限制 【输出】 激活值
68	tf.nn.lrn	LRN	【参数】

序号	Python API	C++ API	边界
			• input: 输入，4-D Tensor，类型：float32 • depth_radius: 0-D int 型，默认值为 5，1-D 标准化窗口的半宽 • bias: 可选参数，float 型，默认为 1；偏移（通常为正值，以避免除以 0） • alpha: 可选参数，float 型，默认为 1；比例因子，通常为正值 • beta: 可选参数，float 型，默认为 0.5；一个指数 • name: 此操作的名称（可选） 【约束】 depth_radius >0，且必须为奇数； 通道间： 当 depth_radius∈[1,15]时，alpha >0.00001 且 beta>0.01，否则 alpha 和 beta 为任意值；当 C 维度大于 680 时，depth_radius <640 【输出】 一个 Tensor，类型同 input
69	tf.nn.max_pool	MaxPool	同 tf.nn.avg_pool
70	tf.nn.relu	Relu	【参数】 • features: 非常量输入，Tensor • name: 此操作的名称（可选） 【约束】 无限制 【输出】 一个 Tensor，类型同 features
71	tf.nn.relu6	Relu6	【参数】 • features: 非常量输入，Tensor • name: 此操作的名称（可选） 【约束】 无限制 【输出】 一个 Tensor，类型同 features

序号	Python API	C++ API	边界
72	tf.nn.selu	Selu	【参数】 • features： 输入 Tensor 数据类型`float32`, • name: string;名称(可选) 【约束】 无约束 【输出】 Tensor，数据类型同 features
73	tf.nn.softmax	Softmax	【参数】 • logits: 一个非空的 Tensor，支持数据类型： float32 • axis:在其上执行维度 softmax。默认值为-1，表示最后一个维度，不超过 logits 维度 • name: 名称(可选) • dim: axis 的已弃用的别名 【约束】 输入 4 维时可以针对每一维做 softmax： axis=1 即 channel 的话，c<=（(128*1024/4)-8*1024-256）/2； axis=0，n<=(56*1024-256)/2； axis=2 即 Height 场景下，W=1, 0<h<（512*1024/32）； axis=3 即 Width 场景下，0<W<（512*1024/32）； 输入维度不足 4 维时，仅支持对最后一维做 softmax 计算，并且最后一维不超过 19968。 【输出】 一个 Tensor，与 logits 具有相同的类型和 shape
74	tf.nn.softplus	Softplus	【参数】 • features: 输入 Tensor；数据类型：`float32` • name: string;名称(可选) 【约束】 无限制 【输出】 输出 Tensor，数据类型与 features 相同

序号	Python API	C++ API	边界
75	tf.nn.softsign	Softsign	【参数】 • features: 输入 Tensor；数据类型：`float32` • name: string;名称(可选) 【约束】 无限制 【输出】 输出 Tensor，数据类型与 features 相同
76	tf.pad	Pad/MirrorPad/ PadV2	【参数】 • tensor: 4-D Tensor；数据类型：`float32`，`int32` • paddings: Tensor, 常量，数据类型：`int32` • mode: string , 值为 "CONSTANT",或"REFLECT",或"SYMMETRIC" • name: string;名称(可选) • constant_values：Pad 默认填充的值标量，数据类型与 tensor 相同 【约束】 CONSTANT 模式时，0=<PAD<=128，0<W<=3000， 【输出】 输出 Tensor，数据类型与 tensor 相同
77	tf.placeholder		【参数】 • dype: 数据类型,(必须) • shape: Tensor 的维度和大小(必须) 【约束】 无限制 【输出】 1 个 Tensor
78	tf.range	Range	【参数】 • start: 开始标量，数据类型`float32`,`int32`，必须为常量 • limit: 结束标量，数据类型`float32`,`int32`，必须为常量 • delta: 步长标量，数据类型`float32`,`int32`，必须为常量 • dtype： 返回 Tensor 的数据类型 • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 1-D Tensor

序号	Python API	C++ API	边界
79	tf.realdiv	RealDiv	【参数】 • x: 输入 Tensor ; 数据类型: float32` • y: 输入 Tensor ; 数据类型: float32` • name: string;名称(可选) 【约束】 无限制 【输出】 输出 Tensor，数据类型同 x
80	tf.reduce_max	Max	【参数】 • input_tensor : 输入 Tensor；数据类型：`float32`，`int64`,`int32`,`uint8`,`uint16`，`int16`，`int8` • axis: 1-D list 或 标量，数据类型整型 • keepdims: bool, 是否保留维度为 1 • name: string;名称(可选) • reduction_indices: axis 旧的别名(不推荐) • keep_dims：keepdims 别名（不推荐） 【约束】 当输入的 Tensor 维数等于 4 时：输入 axis={3,{1,2,3}}，keepDims=true，H*W*16*2 <= 16*1024； 当输入的 Tensor 维数等于 2 时，输入 axis={1,{1}}，keepDims=true，H*W*CEIL(C,16)*16*2 <= 16*1024。 【输出】 输出 reduce 后的 Tensor，数据类型同 input_tensor
81	tf.reduce_sum	sum	【参数】 • input_tensor: 输入 Tensor， • axis: Tensor sum 轴向，数据类型：`int32` • keepdims：bool 值，是否保留维度 • name: string;名称(可选) • reduction_indices：axis 旧的 string;名称 • keep_dims：不推荐使用，参数 keepdims 别名 【约束】 无约束 【输出】 输出 Tensor，数据类型同 tensor
82	tf.reshape	Reshape	【参数】 • tensor: 输入，一个 Tensor

序号	Python API	C++ API	边界
			• shape: 定义输出的 shape，常量 Tensor，数据类型：int64, int • name: 此操作的名称（可选） 【约束】 无限制 【输出】 一个 Tensor，类型同输入
83	tf.reverse	Reverse	【参数】 • tensor: 1 个包含 Tensor 的列表 • axis: Tensor reverse 轴向，数据类型：`int32` • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 Tensor，数据类型同 tensor
84	tf.reverse V2	Reverse	【参数】 • tensor: 1 个包含 Tensor 的列表 • axis: Tensor reverse 轴向，数据类型：`int32`,`int64` • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 Tensor，数据类型同 tensor
85	tf.reverse_sequence	ReverseSequence	【参数】 • input： 输入 Tensor • seq_lengths： 1-D Tensor； 数据类型： `int32`,`int64` • seq_axis: 标量，,数据类型：整型 • batch_axis: 标量（可选），数据类型：整型 • name: string;名称(可选) 【约束】 • seq_lengths 的长度必须等于 input 在 batchAxis 的元素数。 • seq_lengths 的最大元素必须要小于等于 seq_dim 的元素数。 • seqAxis、batchAxis、seqDim、batchDim 必须是 int64 类型 • seqAxis 与 seqDim 不可同时指定，batchAxis 与 batchDim 不可同时指定 • batchAxis 和 batchDim 是可选参数，缺省值为 0 • 权值个数需要为 1

序号	Python API	C++ API	边界
			【输出】 Tensor，数据类型同 input
86	tf.rsqrt	Rsqrt	【参数】 • x: 输入 Tensor • name: 名称（可选） 【约束】 无限制 【输出】 输出 Tensor，类型同 x
87	tf.shape	Shape	【参数】 • input: 一个 Tensor 或 `SparseTensor` • name: string;名称(可选) • out_type 输出 Tensor 数据类型, `int32` 或 `int64` (可选,默认为`int32`) 【约束】 无限制 【输出】 1 个 Tensor,输出数据类型为 out_type
88	tf.sigmoid	Sigmoid	【参数】 • x: 一个 Tensor，输入 • name: 此操作的名称（可选） 【约束】 无限制 【输出】 一个 Tensor，类型同 value
89	tf.size	Size	【参数】 • input: 输入 Tensor，数据类型：float32` • name: string;名称(可选) • out_type: 输出 Tensor，数据类型，默认`int32` 【约束】 无限制 【输出】 1 个 Tensor，类型由 out_type 指定
90	tf.slice	Slice	【参数】 • input: 输入 Tensor

序号	Python API	C++ API	边界
			• begin: tensor，类型：int32 或 int64 • size: tensor，类型：int32 或 int64 • name: 名称（可选） 【约束】 无限制 【输出】 输出 Tensor，类型同 input_
91	tf.space_to_batch_nd	SpaceToBatchND	【参数】 • input：一个 Tensor，是 N 维的并且具有形状 input_shape = [batch] + spatial_shape + remaining_shape,其中 spatial_shape 有 M 维度；支持数据类型为：uint8，int8，int16，uint16，int32，int64，float32 • block_shape：一个 Tensor，必须是以下类型之一：int32,int64；1-D,shape 为[M],所有值必须>=1 • paddings：一个 Tensor，必须是以下类型之一：int32,int64；二维, shape 为[M, 2],所有值必须>=0. 【约束】 • 当 Tensor 维数为 4 时：blockShape 的长度必须等于 2，paddings 的长度必须等于 4. • blockShape 元素的大小必须要大于等于 1，paddings 元素值的大小必须大于等于 0. • padding 后的 h 维度要能够被 blockShape[0]整除，padding 后的 w 维度要能够被 blockShape[1]整除。 【输出】 一个 Tensor，与 input 具有相同的类型
92	tf.space_to_depth	SpaceToDepth	【参数】 • input： 输入 Tensor 数据类型: `float32`,`int64`,`int32`,`uint8`,`int8`, • block_size：标量，整型，值>=2 • data_format：数据类型：string 值: `"NHWC", "NCHW", "NCHW_VECT_C"`. 默认值是 `"NHWC" • name: string;名称(可选) 【约束】 blockSize 的大小必须大于等于 1，且能被 H 和 W 整除 【输出】 Tensor，输出类型 input
93	tf.split	Split/SplitV	【参数】 • value: 输入 Tensor；

序号	Python API	C++ API	边界
			• num_or_size_splits：标量 或 1-D Tensor，指定分割大小 • axis：标准，整型，指定输出维度 • name: string;名称(可选) 【约束】 无限制 【输出】 List，包含 split 后的各 Tensor
94	tf.square	Square	【参数】 • x: Tensor • name: string;名称(可选) 【约束】 无限制 【输出】 输出 Tensor，数据类型同 x
95	tf.squeeze	Squeeze	【参数】 • input: 一个 Tensor，非常量输入 • axis: 一个 int 型列表，指定要移除的维度，默认为[]；不能指定非 1 的维度； • name: 此操作的名称（可选） • squeeze_dims: 不推荐使用的参数，axis 和 dim 不能同时存在 【约束】 无限制 【输出】 一个 Tensor，与 input 的类型、数据相同，但删除了一个或多个值为 1 的维度
96	tf.stack	Stack	【参数】 • values: 包含 Tensor 对象（形状与类型相同，类型：float32，int32）的列表 • axis：整数，沿 axis 维度堆叠，默认是第一维，必须设置 • name: 名称（可选） 【约束】 无限制 【输出】 堆叠后的 Tensor，类型同 values 属性 T、N 和 axis 必须存在

序号	Python API	C++ API	边界
97	tf.strided_slice	StridedSlice	【参数】 • input_: 1 个 Tensor • begin: 1-D Tensor，数据类型: `int32` • end: 1-D Tensor，数据类型: `int32` • strides: 1-D Tensor，数据类型: `int32`; • begin_mask: 标量，数据类型: `int32` • end_mask: 标量，数据类型: `int32` • ellipsis_mask: 标量，数据类型: `int32` • new_axis_mask:：标量，数据类型: `int32` • shrink_axis_mask: 标量，数据类型: `int32` • var：与 input_ 或 None 对应的变量 • name: string;名称(可选) 【约束】 strides 不允许为 0 【输出】 输出 Tensor，数据类型同 input_
98	tf.subtract	Subtract	【参数】 • x: 输入，一个 Tensor • y: 输入，一个 Tensor，类型同 x；两个输入，可支持常量或非常量 • name: 此操作的名称（可选） 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐）。 目前支持以下几种广播场景： NHWC+ NHWC, NHWC+scalar 或者 NHWC +1 1 1 1， 或者 NHWC+W 和 HWC+W 和 HW+W(备注, W 维度做广播)， 或者 NCHW + NH1C 和 HWC + H1C 和 HW + H1， 还有 HWC + 1 WC(备注，H 维度做广播)； 说明：两个 Tensor 的输入顺序可以互换。 【输出】 一个 Tensor

序号	Python API	C++ API	边界
99	tf.tanh	Tanh	【参数】 • x: 输入 Tensor，非常量 • name: 名称（可选） 【约束】 无限制 【输出】 输出 Tensor，类型同 x
100	tf.tile	Tile	【参数】 • input: 输入 Tensor，数据类型：`int8`,`uint8`,`int16`, `uint16`, `int32`,`int64`,`float32` • multiples: 1-D Tensor，长度须和 input 的秩相同，数据类型：`int32`，必须为常量 • name: string;名称(可选) 【约束】 无限制 【输出】 1 个 Tensor
101	tf.transpose	Transpose	【参数】 • a: 输入 Tensor • perm: a 的维数的排列 • name: 名称（可选） • conjugate: 可选，类型: bool，默认为 False；设置为 True 时，等同于 tf.conj(tf.transpose(input)) 【约束】 无限制 【输出】 被转置后的 Tensor
102	tf.unstack	Unpack	【参数】 • value: 输入 Tensor，秩>0，类型: float32、int32 • num: 整数，表示 axis 维度的长度， 默认为 None • axis: 整数，沿 axis 拆分，默认是第一维 • name: 名称（可选） 【约束】 无限制 【输出】 从 value 中拆分出的包含 Tensor 对象的列表

2.4 AndroidNN 算子边界

序号	Operation	含义	边界
1	ANEURALNETWORKS_ADD	加法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor • 输入 1：tensor，OperandCode 与输入 0 相同 • 输入 2：scalar，类型为 INT32，必须是 FuseCode 值之一，指定 activation 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐），目前支持以下几种广播场景： NCHW+NCHW NCHW+1 1 1 1 NCHW+W 和 CHW+W 和 HW+W（W 维度做广播） NCHW + NCH1 和 CHW + CH1 和 HW + H1 CHW + C1W（H 维度做广播） 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
2	ANEURALNETWORKS_AVERAGE_POOL_2D	平均池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：4 支持"NHWC"数据布局 显式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth_in] • 输入 1：scalar，INT32，指定 padding 的左边‘ width’ 维度，0 <= Pad < 256 • 输入 2：scalar，INT32，指定 padding 的右边‘ width’ 维度，0 <= Pad < 256 • 输入 3：scalar，INT32，指定 padding 的顶部‘height’ 维度，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 padding 的底部‘height’ 维度，

序号	Operation	含义	边界
			0 <= Pad < 256 • 输入 5：scalar，INT32，指定 stride 的‘width’维度，0 < Stride < 64 • 输入 6：scalar，INT32，指定 stride 的‘height’维度，0 < Stride < 64 • 输入 7：scalar，INT32，指定 filter width • 输入 8：scalar，INT32，指定 filter height • 输入 9：scalar，INT32，指定 activation 隐式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height, width, depth]，指定 input • 输入 1：scalar，INT32，指定 padding scheme，必须为 PaddingCode 之一 • 输入 2：scalar，INT32，指定 stride 的‘width’ dimension，strideW < 64 • 输入 3：scalar，INT32，指定 stride 的‘height’ dimension，strideH < 64 • 输入 4：scalar，INT32，指定 filter width • 输入 5：scalar，INT32，指定 filter height • 输入 6：scalar，INT32，指定 activation 【约束】 kernelH<=inputH+padTop+padBottom kernelW<=inputW+padLeft+padRight padTop<windowH padBottom<windowH padLeft<windowW padRight<windowW 只支持 globalpool 模式，此模式下的约束条件是： 1) outputH==1 && outputW==1 && kernelH>=inputH && kernelW>=inputW 2) inputH*inputW<=0x10000 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth]
3	ANEURALNETWORKS_BATCH_TO_SPACE_ND	用于 N 维张量的 BatchToSpace	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：4 • 输入 0：4-D tensor，指定 input • 输入 1：1-D tensor，类型为 TENSOR_INT32，指定输出 tensor

序号	Operation	含义	边界
			的每个空间维度的 block sizes，所有值必须> = 1 【约束】 无约束 【输出】 输出 0：tensor，OperandCode 与输入 0 相同
4	ANEURALNETWORKS_CONCATENATION	输入数据按维度拼接	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0~n-1：输入 n 个 tensors 的列表，shape 为 [D0, D1, ..., Daxis(i), ..., Dm]，若为 QUANT8 类型，所有输入 tensors 的 scales 与 zeroPoint 与输出 tensor 的相同 • 输入 n：scalar，类型为 INT32，指定拼接轴 【约束】 输入的 tensor，除了进行 concat 的维度外，其他维度的 size 必须相等 输入的 tensor 个数范围属于[1,1000] 【输出】 输出 0：tensor, OperandCode 与输入相同，输出 shape 为 [D0, D1, ..., sum(Daxis(i)), ..., Dm]
5	ANEURALNETWORKS_CONV_2D	卷积	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：4 支持“NHWC”数据布局 显式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth_in] • 输入 1：4-D tensor，shape [depth_out, filter_height, filter_width, depth_in]，指定 filter，0 < FilterSize < 256 • 输入 2：1-D tensor, shape [depth_out]，指定 bias，输入 tensor 类型为 TENSOR_FLOAT32 时 bias 必须为 TENSOR_FLOAT32；类型为 QUANT8 时，bias 为 TENSOR_INT32，并且 zeroPoint = 0，bias_scale == input_scale * filter_scale • 输入 3：scalar，INT32，指定 padding 的左边‘ width’ 维度，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 padding 的右边‘ width’ 维度，

序号	Operation	含义	边界
			0 <= Pad < 256 • 输入 5：scalar，INT32，指定 padding 的顶部‘height’ 维度，0 <= Pad < 256 • 输入 6：scalar，INT32，指定 padding 的底部‘height’ 维度，0 <= Pad < 256 • 输入 7：scalar，INT32，指定 stride 的‘width’ 维度，0 < Stride < 64 • 输入 8：scalar，INT32，指定 stride 的‘height’ 维度，0 < Stride < 64 • 输入 9：scalar，INT32，指定 activation 隐式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height, width, depth_in] • 输入 1：4-D tensor，shape [depth_out, filter_height, filter_width, depth_in]，指定 filter，0 < FilterSize < 256 • 输入 2：1-D tensor, shape [depth_out]，指定 bias，输入 tensor 类型为 TENSOR_FLOAT32 时 bias 必须为 TENSOR_FLOAT32；类型为 QUANT8 时，bias 为 TENSOR_INT32, 并且 zeroint = 0，bias_scale == input_scale * filter_scale • 输入 3：scalar，INT32，指定 padding scheme，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 stride 的‘width’ dimension，0 < Stride < 64 • 输入 5：scalar，INT32，指定 stride 的‘height’ dimension，0 < Stride < 64 • 输入 6：scalar，INT32，指定 activation 【约束】 • (inputW + padWHe ad + padWTail) >= (((FilterW- 1) * dilationW) + 1) • (inputW + padWHead + padWTail) /StrideW + 1 <= INT32_MAX • (inputH + padHHead + padHTail) >= (((FilterH- 1) * dilationH) + 1) • (inputH + padHHead + padHTail) /StrideH + 1 <= INT32_MAX0 <= Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 <= dilationsize < 256 StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1) 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth_out]，若数据类型为 QUANT8，必须满足 output_scale > input_scale * filter_scale（仅支持 filter 和 bias 为常量的场景）
6	ANEURALNETWORKS_DEPTH_TO_SPACE	数据重新排列从深度转为空间数据	【输入】 • 支持的 tensor OperandCode：

序号	Operation	含义	边界
		块	TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：4 支持 "NHWC"数据布局 • 输入 0：4-D tensor，shape [batches, height, width, depth_in]，指定 input • 输入 1：scalar，类型为 INT32。指定 block_size，block_size >=1 且为输入 tensor 高度和宽度的除数 【约束】 无约束 【输出】 输出 0：4-D tensor，shape [batch, height*block_size, width*block_size, depth/(block_size*block_size)]
7	ANEURALNETWORKS_DEPTHWISE_CONV_2D	深度卷积	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：4 支持"NHWC"数据布局 显式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth_in] • 输入 1：4-D tensor，shape [1, filter_height, filter_width, depth_out]，指定 filter，0 < FilterSize < 256 • 输入 2：1-D tensor, shape [depth_out]，指定 bias，输入 tensor 类型为 TENSOR_FLOAT32 时 bias 必须为 TENSOR_FLOAT32；类型为 QUANT8 时，bias 为 TENSOR_INT32, 并且 zeroPoint = 0，bias_scale == input_scale * filter_scale • 输入 3：scalar，INT32，指定 padding 的左边‘ width’ 维度，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 padding 的右边‘ width’ 维度，0 <= Pad < 256 • 输入 5：scalar，INT32，指定 padding 的顶部‘height’ 维度，0 <= Pad < 256 • 输入 6：scalar，INT32，指定 padding 的底部‘height’ 维度，0 <= Pad < 256 • 输入 7：scalar，INT32，指定 stride 的‘width’ 维度，0 < Stride < 64 • 输入 8：scalar，INT32，指定 stride 的‘height’ 维度，0 < Stride < 64

序号	Operation	含义	边界
			• 输入 9：scalar，INT32，指定 depthwise multiplier • 输入 10：scalar，INT32，指定 activation 隐式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth_in]，指定输入 • 输入 1：4-D tensor，shape [1, filter_height, filter_width, depth_out]，指定 filter，0 < FilterSize < 256 • 输入 2：1-D tensor, shape [depth_out]，指定 bias，输入 tensor 类型为 TENSOR_FLOAT32 时 bias 必须为相同类型；filter tensor 为 QUANT8 时，bias 为 TENSOR_INT32，并且 zeroint = 0，bias_scale == input_scale * filter_scale • 输入 3：scalar，INT32，指定 padding scheme，必须为 PaddingCode 值，为 SAME 或 VALID 之一 • 输入 4：scalar，INT32，指定 stride 的‘width’ dimension，0 < Stride < 64 • 输入 5：scalar，INT32，指定 stride 的‘height’ dimension，0 < Stride < 64 • 输入 6：scalar，INT32，指定 depthwise multiplier • 输入 7：scalar，INT32，指定 activation 【约束】 • filterN=inputC=group • StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1) 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth_out]，若数据类型为 QUANT8，必须满足 output_scale > input_scale * filter_scale（仅支持 filter 和 bias 为常量的场景）
8	ANEURALNETWORKS_DEQUANTIZE	反量化	【输入】 • 支持的 tensor OperandCode： QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, TENSOR_FLOAT32，shape 与输入 0 相同
9	ANEURALNETWORKS_DIV	相除	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） 支持的 tensor 维数：up to 4

序号	Operation	含义	边界
			• 输入 0：n-D tensor • 输入 1：tensor，OperandCode 与输入 0 相同 • 输入 2：scalar，类型为 INT32，必须是 FuseCode 值之一，指定对结果调用的 activation 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐），目前支持以下几种广播场景： NCHW+NCHW NCHW+1 1 1 1 NCHW+W 和 CHW+W 和 HW+W（W 维度做广播） NCHW + NCH1 和 CHW + CH1 和 HW + H1 CHW + C1W（H 维度做广播） 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
10	ANEURALNETWORKS_EMBEDDING_LOOKUP	使用 keys 值映射在输入张量中查找子张量	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：from 2 • 输入 0：Lookups，1-D tensor，TENSOR_INT32，shape [k] • 输入 1：Keys，1-D tensor，TENSOR_INT32，shape [n]，Keys 和 Values 对表示一个映射，即 Keys（Keys [i]）中的第 i 个元素是在 Values（Values [i]）中选择第 i 个子张量的 Key，其中 0 <= i <= n-1。Keys 张量必须按升序排序。 • 输入 2：Values，tensor，shape [n, …]，第一个维度必须是 n。 【约束】 CEIL(lookups,32)*32*2 + CEIL(keys,32)*32 + CEIL(feature,256) * 256* 4 <= 104*1024 【输出】 • 输出 0：tensor，shape [k …]，若为 QUANT8 类型，scale 与 zeroPoint 必须与输入 2 相同 • 输出 1：Hits，tensor，shape [k]，QUANT8，offset= 0，scale=1.0f，指定 lookup 是否 hits，是（非零，True）或否（0，False）
11	ANEURALNETWORKS_FLOOR	向下取整	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） 支持的 tensor 维数：up to 4

序号	Operation	含义	边界
			• 输入 0：tensor 【约束】 无约束 【输出】 输出 0：tensor, OperandCode 和 dimensions 输入 0 相同
12	ANEURALNETWORKS_FULLY_CONNECTED	全连接	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor，维度至少为 2，指定 input。若大于 2，则会被压缩为 2-D Tensor，shape 为[batch_size, input_size] • 输入 1：2-D tensor ，shape [num_units, input_size]，其中“num_units”对应于输出节点的数量。指定 weights • 输入 2：1-D tensor ，shape [num_units]，若为 TENSOR_FLOAT32 的输入张量，bias 相同；若为 QUANT8 的输入张量，bias 应为 TENSOR_INT32，并且 zeroPoint=0，bias_scale == input_scale * filter_scale。指定 bias • 输入 3：scalar，类型为 INT32，必须是 FuseCode 值之一，指定 activation 【约束】 仅支持 transpose=false，axis=1； Bais_C <= 56832 如果输入是 QUANT8： 当 N = 1，2 * CEIL(C,16) * 16 * xH * xW<= 512 * 1024； 当 N > 1，2 * 16 * CEIL(C,16) * 16 * xH * xW <= 512 * 1024。 【输出】 输出 0：tensor, shape [batch_size, num_units]；若为 QUANT8 类型，需满足 output_scale > input_scale * filter_scale.
13	ANEURALNETWORKS_L2_NORMALIZATION	沿深度维度的 L2 归一化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） 支持的 tensor 维数：up to 4 支持“NHWC”数据布局 • 输入 0：n-D tensor，shape [batches, height, width, depth]，定归一化的 tensor 【约束】 H*W*2 < 128*1024/4；

序号	Operation	含义	边界
			【输出】 输出 0：tensor，shape [batches, out_height, out_width, depth]
14	ANEURALNETWORKS_L2_POOL_2D	L2 池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：4 支持 "NHWC"数据布局 显式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth] • 输入 1：scalar，INT32，指定 padding 的左边‘ width’ 维度，0 <= Pad < 256 • 输入 2：scalar，INT32，指定 padding 的右边‘ width’ 维度，0 <= Pad < 256 • 输入 3：scalar，INT32，指定 padding 的顶部‘height’ 维度，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 padding 的底部‘height’ 维度，0 <= Pad < 256 • 输入 5：scalar，INT32，指定 stride 的‘width’ 维度，strideW < 64 • 输入 6：scalar，INT32，指定 stride 的‘height’ 维度，strideH < 64 • 输入 7：scalar，INT32，指定 filter width • 输入 8：scalar，INT32，指定 filter height • 输入 9：scalar，INT32，指定 activation 隐式 padding 输入： • 输入 0：4-D tensor，shape [batches, height, width, depth]，指定 input • 输入 1：scalar，类型为 INT32。指定 padding 填充方案，必须为 PaddingCode 之一。 • 输入 2：scalar，类型为 INT32。指定 stride 的‘width’ 维度，strideW < 64 • 输入 3：scalar，类型为 INT32。指定 stride 的‘height’ 维度，strideH < 64 • 输入 4：scalar，类型为 INT32。指定 filter width • 输入 5：scalar，类型为 INT32。指定 filter height • 输入 6： scalar，类型为 INT32，必须是 FuseCode 值之一，指定对结果调用的 activation 【约束】 1、kernelH <= inputH

序号	Operation	含义	边界
			2、kernelW <= inputW 【输出】 输出 0：4-D tensor，shape [batches, out_height, out_width, depth].
15	ANEURALNETWORKS_LOCAL_RESPONSE_NORMALIZATION	局部响应归一化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） 支持的 tensor 维数：up to 4 支持“NHWC”数据布局 • 输入 0：4-D tensor, shape [batches, height, width, depth]，指定 input • 输入 1：scalar，INT32，指定归一化窗口的 radius • 输入 2：scalar，FLOAT32，不能为 0，指定 bias • 输入 3：scalar，FLOAT32，指定 alpha • 输入 4：scalar，FLOAT32，指定 beta 【约束】 depth_radius >0，且必须为奇数； 通道间： 当 depth_radius∈[1,15]时，alpha >0.00001 且 beta>0.01，否则 alpha 和 beta 为任意值；当 C 维度大于 680 时，depth_radius <640 【输出】 输出 0：tensor, shape 与输入 0 相同
16	ANEURALNETWORKS_LOGISTIC	LOGISTIC 激活	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同，若为 QUANT8 类型，scale 为 1.f / 256，zeroPoint 为 0
17	ANEURALNETWORKS_LSH_PROJECTION	局部敏感哈希（LSH）投影变换	【输入】 • 输入 0：Hash functions，2-D tensor，类型：FLOAT ，tensor [0] .Dim [0]：哈希函数的数量，tensor [0] .Dim [1]：每个散列函数生成的投影输出位数。如果投影类型为 Sparse：Tensor [0] .Dim [1] <= 32

序号	Operation	含义	边界
			• 输入 1：tensor，Dim.size >= 1，类型没有限制 • 输入 2（Optional）：Weight，tensor，Dim.size == 1, DataType: Float。若没有设置，则认为每个输入元素具有 1.0 的相同权重，Tensor [1] .Dim [0] == Tensor [2] .Dim [0] • 输入 3：scalar，类型为 INT32， Type: Sparse：Value LSHProjectionType_SPARSE(=3)，每个输出元素都是一个由散列函数计算出的多位组成。 Type:Dense: Value LSHProjectionType_DENSE(=2)，计算的位向量被认为是密集的。每个输出元素代表一个位，可以取 0 或 1 的值。 【约束】 无限制 【输出】 输出 0：tensor， 若投射类型为 Sparse: Output.Dim == { Tensor[0].Dim[0] }； 若为 Dense：Output.Dim == { Tensor[0].Dim[0] * Tensor[0].Dim[1] }
18	ANEURALNETWORKS_MAX_POOL_2D	最大池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：4 支持 "NHWC"数据布局. 显式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height, width, depth] • 输入 1：scalar，INT32，指定 padding 的左边‘ width’ 维度，0 <= Pad < 256 • 输入 2：scalar，INT32，指定 padding 的右边‘ width’ 维度，0 <= Pad < 256 • 输入 3：scalar，INT32，指定 padding 的顶部‘height’ 维度，0 <= Pad < 256 • 输入 4：scalar，INT32，指定 padding 的底部‘height’ 维度，0 <= Pad < 256 • 输入 5：scalar，INT32，指定 stride 的‘width’ 维度，strideW < 64 • 输入 6：scalar，INT32，指定 stride 的‘height’ 维度，strideH < 64 • 输入 7：scalar，INT32，指定 filter width • 输入 8：scalar，INT32，指定 filter height • 输入 9：scalar，INT32，指定 activation

序号	Operation	含义	边界
			隐式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth]，指定 input • 输入 1：scalar，INT32，指定 padding scheme • 输入 2：scalar，INT32，指定 stride 的‘width’ dimension，strideW < 64 • 输入 3：scalar，INT32，指定 stride 的‘height’ dimension，strideH < 64 • 输入 4：scalar，INT32，指定 filter width • 输入 5：scalar，INT32，指定 filter height • 输入 6：scalar，INT32，指定 activation 【约束】 1、kernelH <= inputH 2、kernelW <= inputW 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth]
19	ANEURALNETWORKS_MEAN	平均值	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor • 输入 1：1-D tensor，类型为 TENSOR_INT32，range [-rank(input_tensor), rank(input_tensor))。指定削减的维度 • 输入 2：scalar，类型为 INT32，若为正，则保持维度为 1 削减。指定 keep_dims 【约束】 无约束 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
20	ANEURALNETWORKS_MUL	乘法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor • 输入 1：tensor，OperandCode 与输入 0 相同 • 输入 2：scalar，类型为 INT32，必须是 FuseCode 值之一，

序号	Operation	含义	边界
			指定 activation 【约束】 支持两组输入的维度不一致，进行广播操作（广播即维度补齐），目前支持以下几种广播场景： NCHW+NCHW NCHW+1 1 1 1 NCHW+W 和 CHW+W 和 HW+W（W 维度做广播） NCHW + NCH1 和 CHW + CH1 和 HW + H1 CHW + C1W（H 维度做广播） 【输出】 输出 0：tensor, OperandCode 与输入 0 相同；若为 QUANT8 类型，需满足 output_scale > input1_scale * input2_scale
21	ANEURALNETWORKS_PAD	填充 0	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：2-D tensor，类型为 TENSOR_INT32，shape {rank(input0), 2}，padding[i, 0]指定相应 i 维前填充的数量，padding[i, 1]指定相应 i 维后填充的数量。指定输入 0 的每个空间维度填充数 【约束】 CONSTANT 模式下 0=<PAD<=128，0<W<=3000，C0=16 【输出】 输出 0：tensor, OperandCode 和维度与输入 0 相同
22	ANEURALNETWORKS_RELU	激活函数 relu	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同

序号	Operation	含义	边界
23	ANEURALNETWORKS_RELU1	激活函数 relu1	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8； 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同
24	ANEURALNETWORKS_RELU6	激活函数 relu6	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同
25	ANEURALNETWORKS_RESHAPE	改变输入维度	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8； 支持的 tensor 维数：up to 4 • 输入 0：tensor • 输入 1：1-D tensor，TENSOR_INT32，指定输出张量的 shape 【约束】 无约束 【输出】 输出 0：tensor，shape 由输入指定

序号	Operation	含义	边界
26	ANEURALNETWORKS_RESIZE_BILINEAR	调整图像大小	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） 支持的 tensor 维数：up to 4 支持“NHWC”数据布局 • 输入 0：4-D tensor，shape [batches, height, width, depth]。指定 input • 输入 1：scalar，类型为 TENSOR_INT32。指定输出 tensor 的 height • 输入 2：scalar，类型为 INT32。指定输出 tensor 的 width 【约束】 （outputH*outputW）/（inputH*inputW） > 1/7 【输出】 输出 0：4-D tensor, shape [batches, new_height, new_width, depth]
27	ANEURALNETWORKS_SOFTMAX	归一化逻辑函数	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：2 or 4 • 输入 0：2-D 或 4-D tensor, shape [batches, height, width, depth]，指定 input • 输入 1：scalar，FLOAT32，指定 beta 的正比例因子 【约束】 输入 4 维时可以针对每一维做 softmax： axis=1 即 channel 的话，c<=（(128*1024/4)-8*1024-256）/2； axis=0，n<=(56*1024-256)/2； axis=2 即 Height 场景下，W=1, 0<h<（512*1024/32）； axis=3 即 Width 场景下，0<W<（512*1024/32）； 输入维度不足 4 维时，仅支持对最后一维做 softmax 计算，并且最后一维不超过 19968。 【输出】 输出 0：tensor, shape 与 input0 相同，若为 QUANT8, scale=1.f / 256，zeroPoint = 0.
28	ANEURALNETWORKS_SPACE_TO_BATCH_ND	用于 N 维张量的 SpaceToBatch	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8

序号	Operation	含义	边界
			支持的 tensor 维数：4 • 输入 0：n-D tensor，指定 input • 输入 1：1-D tensor，类型为 TENSOR_INT32。指定输出 tensor 的每个空间维度的 block sizes，所有值必须> = 1 • 输入 2：2-D tensor，类型为 TENSOR_INT32，shape{M，2}，其中 M 是空间维数，padding [i，0]指定在维度 i 的前面填充的数量，padding [i，1]指定在维度 i 结束之后要填充的数量。指定输出 tensor 的每个空间维度的 paddings，所有值必须> = 0 【约束】 当 tensor 维数为 4 时：blockShape 的长度必须等于 2，paddings 的长度必须等于 4. blockShape 元素的大小必须要大于等于 1，paddings 元素值的大小必须大于等于 0. padding 后的 h 维度要能够被 blockShape[0]整除，padding 后的 w 维度要能够被 blockShape[1]整除。 【输出】 输出 0：tensor，OperandCode 与输入 0 相同
29	ANEURALNETWORKS_SPACE_TO_DEPTH	数据重新排列从空间转为深度	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） QUANT8 支持的 tensor 维数：4 支持 "NHWC"数据布局 • 输入 0：4-D tensor，shape [batches, height, width, depth_in]，指定 input • 输入 1：scalar，类型为 INT32，指定 block_size，block_size >=1 且为输入 tensor 高度和宽度的除数 【约束】 blockSize 的大小必须大于等于 1，且能被 H 和 W 整除 【输出】 输出 0：4-D tensor，shape [batches, height/block_size, width/block_size, depth_in*block_size*block_size]

序号	Operation	含义	边界
30	ANEURALNETWORKS_SQUEEZE	压缩	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1（optional）：1-D tensor，类型为 TENSOR_INT32，若不指定，则压缩所有维度，维度索引从 0 开始，若不是单维度，则报错。指定压缩维度 【约束】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
31	ANEURALNETWORKS_STRIDED_SLICE	切分	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，指定 input • 输入 1：begin，1-D tensor，类型为 TENSOR_INT32，长度为输入 0 的 rank • 输入 2：end，1-D tensor，类型为 TENSOR_INT32，长度为输入 0 的 rank • 输入 3：strides，1-D tensor，类型为 TENSOR_INT32，长度为输入 0 的 rank • 输入 4：begin_mask，scalar，类型为 INT32，若设置了 begin_mask 的第 i 位，则忽略 begin [i]并使用该维度中的最大可能范围 • 输入 5：end_mask，scalar，类型为 INT32，若设置了 end_mask 的第 i 位，则忽略 end[i]并使用该维度中的最大可能范围 • 输入 6：shrink_axis_mask，scalar，类型为 INT32，若设置了 shrink_axis_mask 的第 i 位，则第 i 个维度指定 shrinks 维度为 1，值为索引 begin [i]处的值。 【约束】 strides 不允许为 0 【输出】 输出 0：tensor, OperandCode 与输入 0 相同

序号	Operation	含义	边界
32	ANEURALNETWORKS_SUB	减法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，指定第一个 input • 输入 1：tensor，OperandCode 与输入 0 相同 • 输入 2：scalar，类型为 INT32，必须是 FuseCode 值之一，指定 activation 【约束】 • 支持两组输入的维度不一致，进行广播操作（广播即维度补齐），目前支持以下几种广播场景： NCHW+NCHW NCHW+1 1 1 1 NCHW+W 和 CHW+W 和 HW+W（W 维度做广播） NCHW + NCH1 和 CHW + CH1 和 HW + H1 CHW + C1W（H 维度做广播） 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
33	ANEURALNETWORKS_TANH	激活函数 tanh	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同
34	ANEURALNETWORKS_TRANSPOSE	转置	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， QUANT8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，指定 input • 输入 0（optional）：1-D tensor，TENSOR_INT32，指定输入 tensor 的维度确定转置方式 【约束】

序号	Operation	含义	边界
			无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同（默认在 2-D 输入张量上执行常规矩阵转置）

