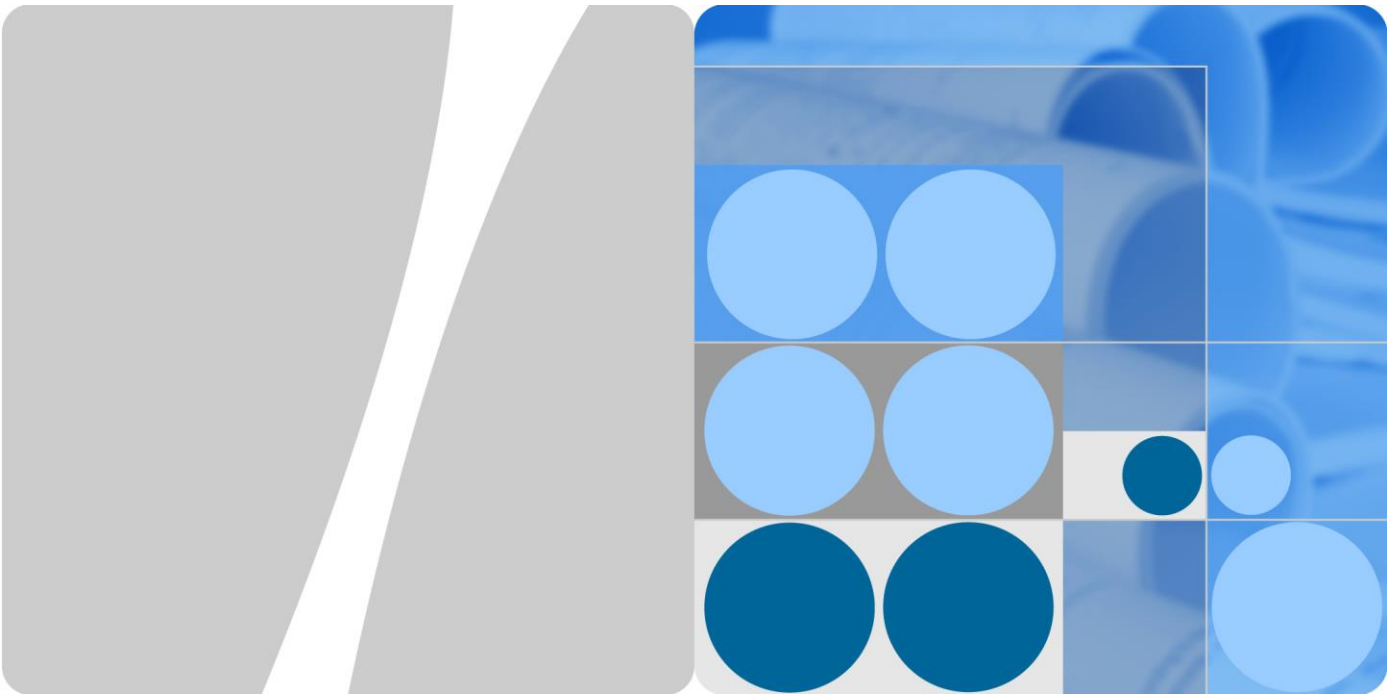

HiAI DDK V320

算子规格说明

文档版本 04

发布日期 2020-02-29

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 V320 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 V320 快速入门
- 华为 HiAI DDK V320 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
2020-02-29	04	增加 stridedSlice 约束，补充 CPU 算子
2019-12-31	03	新增 V320 版本内容
2019-11-15	02	增加 avgpool 约束
2019-09-06	01	新增 V310 版本内容

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(X，N)	输出 X 向上取整 到 N 的倍数（举例： ALIGN(1,16)=16，ALIGN(16,16)=16， ALIGN(17,16)=32）

2 NPU 算子约束

2.1 整体约束

NCHW 场景和 NHWC 场景， 1<=N<=65535；

对于 caffe 框架，如果存在 axis 参数，则输入维度不能为 1，且输入维度为 2 或 3 时 axis 不能为负数。

- Caffe 支持版本 1.0
- Tensorflow 支持版本 1.12
- AndroidNN 支持版本 API 29

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】 一个输入 【参数】 engine：枚举型，默认为 0（DEFAULT=0，CAFFE=1，CUDNN=2），可选
2	Argmax	返回输入的最大值对应的索引序号	【输入】 一个输入 【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
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	一个输入 【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 • scale, beta,mean,gamma 的 shape 为[c]
4	ChannelAxpy （V310 新增）	ChannelAxpy (Squeeze-and-Excitation Networks)	【输入】 三个输入 • 输入 1: tensor,shape 为[N, C, 1, 1] • 输入 2: tensor, C 通道与输入 1 的 C 通道一致 • 输入 3: tensors shape 同输入 2
5	Concat	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[1,32]
6	Convolution	卷积	【输入】 一个输入 【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

序号	算子	含义	边界
			• 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 • 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】 • filter 必须是常量 4D • (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) • 输入 bias 维度只支持(1,co,1,1), co 与 num_output 相等
7	Crop	截取	【输入】 两个输入 【参数】 • axis: int32，默认为 2，axis=-1，

序号	算子	含义	边界
			input dim 必须等于 4，可选 • offset：uint32，数组
8	Correlation（V310 新增）	相关性	【输入】 两个输入，要求输入在 C 通道必须相同 【参数】 • num_output: uint32，与输入 2 的 N 维相等 • kernel_size: uint32，与输入 2 的 H 维、W 维相等 • stride: uint32，默认为 1，可选 • group: uint32，默认为 1，可选，group 必须是 1 • pad: uint32，默认为 0，可选 • dilation: uint32，默认为 1，可选 【约束】 • N <= 3840 • 其他约束和卷积相同
9	Deconvolution	反卷积	【输入】 一个输入 【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 • weight_filler：类型为

序号	算子	含义	边界
			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】 filter 必须是常量 4D group < 1000 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/2； 2、conv_input_width=(反卷积输入 w - 1) *strideW + 1； 3、b = （conv_input_width） * filter_h * ALIGN(filter_num,16) * 4； 4、a + b <= 512KB
10	Convolution Depthwise	深度卷积	【输入】 一个输入 【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

序号	算子	含义	边界
			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 • 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】 • filter 必须是常量 4D • filterN=inputC=group； • StrideW<=(inputW + padW) - ((filterW - 1) * dilationW) + 1)； • 输入 bias 维度只支持(1,co,1,1), co 与 num_output 相等
11	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。

序号	算子	含义	边界
12	Elu	激活函数	【输入】 一个输入 【参数】 alpha (optional)：类型：float，默认为 1
13	Exp	指数运算	【输入】 一个输入 【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
14	Flatten	数据按第一维度展开 输入 n*c*h*w 变为向量 n* (c*h*w)	【输入】 一个输入 【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
15	InnerProduct	全连接	【输入】 一个输入 【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； 输入 bias 维度只支持(1,co,1,1), co 与

序号	算子	含义	边界
			num_output 相等； 如果客户要量化模型时，需要满足下列维度： 当 N = 1，2 * ALIGN(C,16) * xH * xW<= 524288； 当 N > 1，2 * 16 * ALIGN(C,16) * xH * xW <= 524288。
16	Interp	插值层	【输入】 一个输入 【参数】 • height：可选，类型 int32，默认为 0 • 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
17	LeakyRelu	LeakyRelu 激活函数	【输入】 一个输入 【参数】 同 Relu
18	Log	对数运算	【输入】 一个输入 【参数】 • base：可选，类型：float，默认为-1.0 • scale：可选，类型：float，默认为

序号	算子	含义	边界
			1.0 • shift：可选，类型：float，默认为 0.0 【约束】 scale*x + shift > 0
19	LRN	局部响应归一化层	【输入】 一个输入，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） • k：可选，类型：float，默认为 1. • engine：可选，类型：枚举型（DEFAULT = 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，beta>0.01，否则 k 和 beta 为任意值；k 和 alpha 不同时为 0；当 C 维度大于 680 时，local_size <640； 通道内： k=1，local_size∈[1,15]，beta>0.01；
20	LSTM	长短期记忆网络	【输入】 两个或三个输入 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，

序号	算子	含义	边界
			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)) * 64 b = (ALIGN(xt,16) + ALIGN(output,16)) * 256 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 <= 524288； d <= 16384； e <= 4096；
21	Normalize	标准化层	【输入】 一个输入 【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。
22	Permute	Reshape 操作	【输入】

序号	算子	含义	边界
			一个输入 【参数】 order: uint32；数组
23	Pooling	池化层	【输入】 一个输入 【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 • 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<256, kernelW<256; • stride < 64 ; stride > pad 并且 stride < input+pad-kernel; • 当输出 tensor shape H、W 为 1 时，要求 input H * input W < 65536

序号	算子	含义	边界
			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
24	Power	计算 y = （scale*x+shift）^ power	【输入】 一个输入 【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】 scale*x + shift > 0
25	Prelu	激活函数	【输入】 一个输入 【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
26	PriorBox	从目标预选框获取真实目标的位置	【输入】 • 输入 0，必选，对于此算子，只关注输入的 shape，忽略输入的值 • 输入 1，可选，输入图片信息 【参数】 • min_size: 必须设置，最小框大小（以像素为单位） • 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，

序号	算子	含义	边界
			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]。
27	Proposal	将预选框通过（proposal, score）排序，通过 nms 获取 topN proposal	【输入】 三个输入（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 不能同时存在

序号	算子	含义	边界
			• 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
28	PSROIPooling	位置敏感的区域池化	【输入】 两个输入 【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==group_size <= 128 [pooledH　pooledW]表示 pool 框的长宽

序号	算子	含义	边界
			输出的格式 y[yN, yC, yH, yW] 2）poolingMode == avg pooling，pooledH == pooledW == groupSize，pooledH <= 128， spatial_scale > 0，group_size > 0， output_dim > 0； 3）1<=xN<=65535， roiN % xN == 0； 4）xHW = xH * xW，pooledHW = pooledH * pooledW，HW_LIMIT = 768， 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； 6）yN == roiN，yH == pooledH，yW == pooledW，yC == output_dim； 7）xC == yC * pooledH * pooledW
29	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】 一个输入 【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
30	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
31	Reshape	改变输入维度	【输入】 一个输入 【参数】 • shape: 常量，类型：int64 或 int32 • axis: 可选，类型：int32，默认为 0

序号	算子	含义	边界
			• num_axes: 可选，类型：int32，默认为-1
32	Reverse	逆转	【输入】 一个输入 【参数】 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
33	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、N<65535 2、H * W <=2464； 3、C <= 1152； 4、((C-1)/128+1)*pooledW <= 92； RoI（感兴区域）约束： 1、N<65535 2、C=5(caffe) ，H=1，W=1； 3、samplingRatio*pooledW <= 128 且 samplingRatio*pooledH <= 128； 4、H >= pooledH, W >= pooledW。
34	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: 可选，类型：uint32，默认为 0 • pooled_w: 可选，类型：uint32，默

序号	算子	含义	边界
			认为 0 • spatial_scale: 可选，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 支持输入 feature map 最大 H*W：3888，输出最大 H*W：256
35	Scale	out=alpha*Input+beta	【输入】 两个输入 【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；
36	ShuffleChannel	帮助信息在特征通道交叉流动	【输入】 一个输入 【参数】 • group: 可选，类型：uint32，默认为 1
37	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
38	Slice	将输入分解成多个输出	【输入】 一个输入

序号	算子	含义	边界
			【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）；
39	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】 • axis 输入范围[-rank,rank) 输入 4 维(NCHW)时可以针对每一维做 softmax： axis=0，n<=28544； axis=1 即 channel 的话，c<=11136，h*w < 65536； axis=2 即 Height 场景下，W=1, 0<h<=16384； axis=3 即 Width 场景下，0<W<=16384； 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。
40	SSDDetectionOutput	SSD 网络检测输出	【输入】 三个输入 【参数】 • num_classes: 必选，类型：int32，要预测的分类（含背景分类） • 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

序号	算子	含义	边界
			• 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】 只支持 Caffe 框架的 SSD 网络结构场景 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； 多 batch 场景与标准 SSD 不同，priorbox 也会计算生成多 batch 结果
41	Tanh	激活函数	【输入】

序号	算子	含义	边界
			一个输入 【参数】 engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2
42	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
43	SpatialTransform （V310 新增）	空间变换	【输入】 一个输入 【参数】 • output_h: 必须设置，类型：uint32，默认为 0 • output_w: 必须设置，类型：uint32，默认为 0 • border_value: 可选，类型: float，默认为 0 • affine_transform: 类型：float • engine: 可选，类型：枚举，取值：DEFAULT=0，CAFFE=1，CUDNN=2 【约束】 (outputH*outputW) / (inputH*inputW) > 1/7
44	Tile （V320 新增）	将输入 Blob 的某个指定维度的数据进行复制扩展	【输入】 一个输入 【参数】 • axis：int32；指定扩展轴，取值见 2.1 整体约束 • tiles：int32；复制扩展的倍数
45	Split （V320 新增）	将输入数据复制为 1 份或者多份	【输入】 一个输入； 支持的数据类型：float,double,int8,uint8,int32,uint32,int64,uint64,bool； 支持的数据格式：NCHW
46	BatchReindex	将输入的数据按照不同 batch 的索引进行	【输入】 两个输入

序号	算子	含义	边界
	（V320 新增）	抽取、重排、复制	input0：输入数据：float； input1：输入的索引数据：uint32； 【约束】 0≤input1＜input0 的 batch 数量，如果超出约束，输出结果不可靠
47	SPP （V320 新增）	对输入图像按照空间金字塔结构在区域内进行最大、平均或者随机值池化操作，使不同大小的图像的结果向量具有相同的大小，从而在输入图像上汇聚在一起	【输入】 一个输入 【参数】 • pyramid_height：必选；类型：int32；空间金字塔的层数 • pool：可选；类型：int32；目前仅支持 MAX、AVG 两种方式，MAX、AVG、分别表示 0、1，默认为 MAX
48	Threshold （V320 新增）	遍历各数值与给定阈值进行比较，大于输出 1，否则输出 0	【输入】 一个输入 【参数】 • threshold：可选；float32；默认为 0.0
49	MVN （V320 新增）	将输入的数据进行归一化操作	【输入】 一个输入 【参数】 • normalize_variance：可选；bool 型；false 表示均值归一化，true 表示方差归一化，默认为 true • across_channels：可选；bool 型；若为 false：batch*channel 作为矩阵的行, height*width 作为矩阵的列；若为 true: batch 作为矩阵的行, channel*height*width 作为矩阵的列；默认为 false
50	BNLL （V320 新增）	实现二项正太对数似然函数（激活函数）	【输入】 一个输入
51	Swish （V320 新增）	计算输入 tensor 中每个元素的 swish 函数值 y=x* σ (scale*x)，其中 scale 为缩放因子	【输入】 一个输入 【参数】 • beta: swish 函数缩放因子，float32

序号	算子	含义	边界
52	Bias （V320 新增）	实现两个 tensor 输入 input0 与 bias 相加，如果 input0 与 bias 的 shape 不相同，需要将 bias 的维度进行扩展，具体的扩展方式根据入参 axis 来确定，axis 表示 bias 从哪一维开始匹配 input0	【输入】 两个输入 【参数】 • axis: 标量，int 型，指定 inputs 从哪个轴与 axis 对齐；范围为[-4,3]。
53	Dropout （V310 新增）	防止过拟合的层	【输入】 一个输入 【参数】 • dropout_ratio：可选，默认为 0.5，类型：float • scale_train：可选，bool，默认为 true
54	ReLU6 (V320 新增)	激活函数	【输入】 一个输入 【参数】 negative_slope：可选，默认为 0，类型：float 【约束】 negative_slope 只支持 0

2.3 Tensorflow 算子边界

序号	Python API	C++ API	边界
1	tf.abs	Abs	【参数】 • x: 输入 Tensor，类型：float32, int32 • name: optional, string 【输出】 返回 x 的绝对值，Tensor，尺寸与类型同 x
2	tf.add	Add	【参数】 • x: 输入 Tensor，类型：float32 • y: 输入 Tensor，类型同 x；

序号	Python API	C++ API	边界
			• name: optional, string 【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】 输出 Tensor，类型同 y
3	tf.add_n （V310 新增）	AddN	【参数】 • inputs: 输入 Tensor list 或者 IndexedSlices list，其中每个的 tensor 或者 IndexedSlices 具有相同的 type 与 shape，类型：float32 • name: optional, string 【约束】 无限制 【输出】 输出 Tensor，类型与 shape 与 inputs 元素相同
4	tf.batch_to_space_nd	BatchToSpaceND	【参数】 • input：输入 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；类型：float32 • block_shape：Tensor；类型：int32；1-D,shape 为[M],所有值必须>=1 • crops：Tensor；类型：int32；二维, shape 为[M, 2],所有值必须>=0. 【约束】 • block_shape 和 crops 的元素值数据类型必须是 int32，当 Tensor 维数为 4 时：block_shape 的长度必须等于 2，crops 的长度必须等于 4. • block_shape 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] <

序号	Python API	C++ API	边界
			block_shape[i] * input_shape[i+1]； 【输出】 输出 Tensor，与 images 具有相同的类型
5	tf.cast	Cast	【参数】 • x: 输入 Tensor，类型：float32, int32, bool, int64, int16, int8, uint8, uint16, double • dtype: 目标类型，同 x 支持的数据类型 • name: optional, string 【约束】 无限制 【输出】 Tensor 或 sparseTensor 或 indexedSlices， 同输入的 dtype、shape
6	tf.math.ceil （V310 新增）	Ceil	【参数】 • x: 输入 tensor，类型： float32 • name: optional, string 【输出】 输出 tensor，类型 同 x
7	tf.clip_by_value	ClipByValue	【参数】 • t: 输入 Tensor • clip_value_min: clip 最小值 • clip_value_max：clip 最大值 • name: optional, string 【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]
8	tf.concat	ConcatV2	【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；

序号	Python API	C++ API	边界
			【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[1,32] 【输出】 输出 Tensor，为输入 Tensors 拼接后的结果
9	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(optional)，用于输出 • name: optional, string • verify_shape: 布尔值(optional)，默认 False 【约束】 无限制 【输出】 1 个常量 Tensor
10	tf.depth_to_space	DepthToSpace	【参数】 • input：输入 Tensor，类型: float32 • block_size：scalar，整型，值>=2 • data_format：数据类型：string 值: NHWC, NCHW, 默认值是 NHWC • name: optional, string 【输出】 输出 Tensor，输出类型 input
11	tf.equal	Equal	【参数】 • x: 输入 Tensor, 类型：float32, uint8, int32, bool • y: 输入 Tensor, 类型同 x • name: optional, string 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一维度上(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
12	tf.exp	Exp	【参数】 • x: 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】 输出 Tensor，类型同 x
13	tf.math.expm1 （V310 新增）	Expm1	【参数】 • x: 输入 tensor，类型: float32 • name: optional, string 【输出】 输出 tensor，类型 同 x
14	tf.expand_dims	ExpandDims	【参数】 • input: 输入 tensor • axis: 0-D（scalar），指定扩展 input 形状的维度索引； • name: optional, string • dim: 0-D（scalar），相当于 axis，被弃用 【约束】 无限制 【输出】 输出 Tensor，与 input 数据相同，shape 增加了一维（值为 1）
15	tf.extract_image_patches	ExtractImagePatches	【参数】 • images: 输入 Tensor，类型：float32, uint8；4-D Tensor shape：[batch, in_rows, in_cols, depth] • ksizes: 一个整型 list，长度>=4 • strides: 一个整型 list，必须是：[1, stride_rows, stride_cols, 1] • rate: 一个整型 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: optional, string 【约束】 无约束 【输出】

序号	Python API	C++ API	边界
			输出 Tensor，与 images 具有相同的类型
16	tf.fake_quant_with_min_max_vars	FakeQuantWithMinMaxVars	【参数】 • inputs : 输入 Tensor，类型：float32 • min: Tensor, 类型：float32 • max: Tensor, 类型：float32 • num_bits：scalar，整型，默认为 8 • narrow_range：bool（optional），默认值 False • name: optional, string 【约束】 -65504<=min<=65504，-65504<=max<=65504。 【输出】 输出 Tensor，类型：float32
17	tf.fill	Fill	【参数】 • dims: 1-D Tensor, 类型：int32 • value: 变量，类型：int32, float32 • name: optional, string 【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】 输出 Tensor，类型同 value 数据类型
18	tf.floormod	FloorMod	【参数】 • x: 输入 Tensor，类型：float32，int32 • y: 输入 Tensor，与 x 具有相同类型 • name: optional, string 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一维度上(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，与 x 具有相同的类型
19	tf.gather	Gather GatherV2	【参数】

序号	Python API	C++ API	边界
			• params: 输入 Tensor ,维度必须大于 `axis + 1` • indices: Tensor, 类型：int32, int64, 范围[0, params.shape[axis]) • axis: Tensor，类型：int32, int64，指定 indices 选取的维度，rank=0 • name: optional, string 【约束】 无限制 【输出】 输出 Tensor,输出数据类型于 params 相同
20	tf.gather_nd	GatherNd	【参数】 • params: 输入 Tensor ,维度必须大于 `axis + 1` • indices: 输入 Tensor, 类型：int32, int64 • name: optional, string 【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】 输出 Tensor,输出数据类型于 params 相同
21	tf.greater	Greater	【参数】 • x: 输入 Tensor，类型：float32, int32 • y: 输入 Tensor，类型：float32, int32 • name: optional, string 【约束】 仅支持常量输入 支持广播 broadcast 【输出】 输出 Tensor，类型：bool
22	tf.image.crop_and_resize	CropAndResize	【参数】 • image: 4-D Tensor，类型：float32, int8, int32, int64; shape: [batch, image_height, image_width, depth] • boxes：2-D Tensor，类型: float32; shape: [num_boxes, 4] • box_ind：1-D Tensor，类型：int32; shape:

序号	Python API	C++ API	边界
			[num_boxes] • crop_size：1-D Tensor ,包含 2 个元素, 类型：int32 • method: string, 表示插值方法, 值为 "bilinear"(默认), "nearest" • extrapolation_value：可选，数据类型：float32，默认为 0 • name: optional, string 【约束】 无限制 【输出】 输出 Tensor，类型: float32
23	tf.image.non_max_suppression （V310 新增）	NonMaxSuppressionV2	【参数】 • boxes: 2-D Tensor，类型：float32；shape : [num_boxes, 4] • scores：1-D Tensor， 类型：float32；shape: [num_boxes] • max_output_size：scalar，类型：int32, 表示最大输出框个数 • iou_threshold：scalar，类型：float32 • name: optional, string 【输出】 1-D Tensor，int32，shape:[M]，M<=max_output_size
24	tf.image.resize_bilinear	ResizeBilinear	【参数】 • images: 4-D tensor, [batch, height, width, channels]；类型：float32 • size: 1-D Tensor，常量，2 个元素（新的高宽） • 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】 输出 tensor，shape 同输入，类型：float
25	tf.image.resize_nearest_neighbor	ResizeNearestNeighbor	【参数】 • images: 4-D Tensor，[batch, height, width, channels]；类型：float32

序号	Python API	C++ API	边界
			• size: 1-D Tensor，常量，2 个元素（新的高宽） • 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】 输出 tensor，shape 同输入，类型：float
26	tf.invert_permutation	InvertPermutation	【参数】 • x: 1-D Tensor，类型：int32, int64 • name: optional, string 【约束】 无限制 【输出】 输出 Tensor，类型同 x
27	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：当 x < -2.5 时，返回 0；当 x > 2.5 时，返回 1 当-2.5 <= x <= 2.5, 返回 0.2 x + 0.5.
28	tf.keras.layers.ThresholdedReLU	ThresholdedReLU	【参数】 theta：类型：float32 【约束】 0 <= theta <= 65504 【输出】 输出 Tensor
29	tf.log	Log	【参数】 • x: 输入 Tensor，类型：float32 • name: optional, string 【约束】 输入数据范围（x>0）

序号	Python API	C++ API	边界
			【输出】 输出 Tensor，类型同 x
30	tf.math.log1p （V310 新增）	Log1p	【参数】 • x: 输入 tensor，类型: float32 • name: optional, string 【输出】 输出 tensor，类型 同 x
31	tf.math.acos	Acos	【参数】 • x：输入 Tensor，类型：float32 • name: optional, string 【约束】 输入数据范围（-1<=x<=1），输出数据范围（0<=y<= π ） 【输出】 输出 Tensor，类型同 x
32	tf.math.acosh	Acosh	【参数】 • x：输入 Tensor，类型：float32 • name: optional, string 【约束】 输入数据范围（x>=1） 【输出】 输出 Tensor，类型同 x
33	tf.math.argmax	ArgMax	【参数】 • input: 输入 Tensor, 类型：int8, uint8, int16, uint16, int32, int64, float32 • axis: Tensor, 类型：int32, int64 • out_type 输出 Tensor，类型：int32, int64 (optional, 默认为 int64) • name: optional, string 【约束】 无限制 【输出】 输出 Tensor, 输出类型为 out_type
34	tf.math.asin	Asin	【参数】 • x：输入 Tensor，类型：float32

序号	Python API	C++ API	边界
			• name: optional, string 【约束】 输入数据范围（-1<=x<=1），输出数据范围（-π/2<=y<=π/2）。 【输出】 输出 Tensor，类型同 x
35	tf.math.asinh	Asinh	【参数】 • x：输入 Tensor，类型: float32 • name: optional, string 【约束】 无约束 【输出】 输出 Tensor，类型同 x
36	tf.math.atan	Atan	【参数】 • x：输入 Tensor，类型：float32 • name: optional, string 【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，类型同 x
37	tf.math.atanh	Atanh	【参数】 • x：输入 Tensor，类型：float32 • name: optional, string 【约束】 输入数据范围：x∈(-1,1) 【输出】 输出 Tensor，类型同 x
38	tf.math.cosh	Cosh	【参数】 • x：输入 Tensor，类型：float32 • name: optional, string 【约束】 无限制 【输出】 输出 Tensor，类型同 x

序号	Python API	C++ API	边界
39	tf.math.floor（V310 新增）	Floor	【参数】 • x: 输入 Tensor，类型：float32 • name: optional, string 【约束】 不支持 8-bit 量化 【输出】 输出 Tensor, 类型同 x
40	tf.math.greater_equal	GreaterEqual	【参数】 • x：输入 Tensor，类型：float32 • y：输入 Tensor， 类型同 x • name: optional, string 【约束】 输入数据范围（-65504<=x<=65504） 【输出】 输出 Tensor，类型：bool
41	tf.math.less	Less	【参数】 • x：输入 tensor，类型：float32 • y：输入 tensor，类型同 x • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型：bool
42	tf.math.logical_and	LogicalAnd	【参数】 • x: tensor，类型：bool（不支持常量输入） • y: tensor，类型：bool（不支持常量输入） • name: optional, string 【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，类型: bool
43	tf.math.logical_not	LogicalNot	【参数】 • x: tensor，类型：bool • name: optional, string

序号	Python API	C++ API	边界
			【约束】 无限制 【输出】 输出 Tensor，类型：bool
44	tf.math.logical_or	LogicalOr	【参数】 • x：输入 tensor，类型: bool（不支持常量输入） • y：输入 tensor，类型: bool（不支持常量输入） • name: optional, string 【约束】 由于支持广播 broadcast，对比 x 和 y 的 shape，在同一维度上(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，类型: bool
45	tf.math.maximum	Maximum	【参数】 • x: tensor，类型：int32，float32 • y: tensor，类型同 x • name: optional, string 【约束】 无 【输出】 输出 tensor，返回值（x > y ? x : y） 类型同 x
46	tf.math.minimum	Minimum	【参数】 • x: 输入 tensor，类型：int32，float32 • y: 输入 tensor，类型同 x • name: optional, string 【约束】 无 【输出】 输出 tensor，返回值（x < y ? x : y） 类型同 x
47	tf.math.negative	Neg	【参数】 • x：输入 tensor，类型: float32 • name: optional, string

序号	Python API	C++ API	边界
			【约束】 输入数据范围（-65504<=x<=65504），输出数据范围（-65504<=y<=65504） 【输出】 输出 tensor，输出= -x
48	tf.math.pow	Pow	【参数】 • x: 输入 tensor，类型：float32 • y: 输入 tensor，类型：float32 • name: optional, string 【约束】 无约束 【输出】 输出 tensor
49	tf.math.reciprocal	Reciprocal	【参数】 • x：输入 tensor, 类型：float32 • name: optional, string 【约束】 不支持输入数据中包含 0 【输出】 输出 tensor，类型同 x
50	tf.math.reduce_all	All	【参数】 • input_tensor：输入 tensor, 类型：bool • axis：reduce 的维度轴向 • keepdims：标量，类型：bool • name: optional, string • 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】 输出 tensor，类型同 input_tensor
51	tf.math.reduce_max	ReduceMax	【参数】 • input_tensor：输入 tensor, 类型：float32, int64, uint8, uint16, int8, int16 • axis: reduce 的维度轴向

序号	Python API	C++ API	边界
			• keepdims：标量，类型：bool • name: optional, string • reduction_indices：axis 旧的别名(不推荐) • keep_dims：keepdims 别名（不推荐） 【约束】 • axis 输入范围[-rank,rank) 【输出】 输出 tensor，类型同 input_tensor
52	tf.math.reduce_mean （V310 新增）	Mean	【参数】 • input_tensor：输入 tensor, 类型: float32 • axis：reduce 的维度轴向 • keepdims：标量，类型：bool • name：optional, string • reduction_indices：axis 旧的别名(不推荐) • keep_dims：keepdims 别名（不推荐） 【约束】 • axis 输入范围[-rank,rank) • axis 降维方式支持情况： 1D：支持全场景 2D：仅支持 1H、C1 两种，其中 0、1 分别代表 C、H 3D：仅支持 C11、CH1、C1W、111、1H1、1HW 六种，其中 0、1、2 分别代表 C、H、W 4D：仅支持 NC11、NCH1、NC1W、N111、N1H1、N1HW 六种，其中 0、1、2、3 分别代表 N、C、H、W 【输出】 输出 tensor，类型同 input_tensor
53	tf.math.reduce_min	Min	【参数】 • input_tensor：输入 tensor，类型: float32, int64, int32, uint8, uint16, int8, int16 • axis: reduce 的维度轴向 • keepdims：标量，类型：bool • name: optional, string • reduction_indices：axis 旧的别名(不推荐) • keep_dims：keepdims 别名（不推荐）

序号	Python API	C++ API	边界
			【约束】 当输入的 Tensor 维数等于 4 时：输入 axis={3,{1,2,3}}，keepdims=true，H*W <= 512； 当输入的 Tensor 维数等于 2 时，输入 axis={1,{1}}，keepdims=true，H*W*ALIGN(C,16) <= 8192 【输出】 输出 tensor，类型同 input_tensor
54	tf.math.reduce_prod	Prod	【参数】 • input_tensor：输入 tensor，类型：float32, int64, int32, uint8, uint16, int8, int16 • axis: reduce 的维度轴向 • keepdims：标量，类型：bool • name: optional, string • 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 <= 512； 当输入的 Tensor 维数等于 2 时，输入 axis={1,{1}}，keepdims=true，H*W*ALIGN(C,16) <= 8192 【输出】 输出 tensor，类型同 input_tensor
55	tf.math.rint	Rint	【参数】 • x：输入 tensor，类型：float32 • name: optional, string 【约束】 无限制 【输出】 输出类型同 x，输出 shape 同 x
56	tf.math.round	Round	【参数】 • x：输入 tensor，类型：float32 • name: optional, string 【输出】 输出 tensor，类型同 x，输出 shape 同 x
57	tf.math.rsqrt	Rsqrt	【参数】

序号	Python API	C++ API	边界
			• x: 输入 tensor，类型：float32 • name: optional, string 【输出】 输出 tensor，类型 同 x
58	tf.math.sinh	Sinh	【参数】 • x：输入 tensor，类型: float32 • name: optional, string 【输出】 输出 tensor，类型同 x
59	tf.math.sin （V310 新增）	Sin	【参数】 • x：输入 tensor，类型: float32 • name: optional, string 【输出】 输出 Tensor，类型同 x
60	tf.math.cos （V310 新增）	Cos	【参数】 • x：输入 tensor，类型: float32 • name: optional, string 【输出】 输出 tensor，类型同 x
61	tf.math.sqrt	Sqrt	【参数】 • x：输入 tensor，类型：float32 • name: optional, string 【约束】 无约束 【输出】 输出 tensor，类型同 x
62	tf.math.squared_difference	SquaredDifference	【参数】 • x：输入 tensor，类型：float32, int64, int32 • y：输入 tensor，类型同 x • name: optional, string 【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

序号	Python API	C++ API	边界
			几种情况。 【输出】 输出 tensor，类型同 x
63	tf.math.tan	Tan	【参数】 • x：输入 tensor，类型: float32 • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 x
64	tf.math.top_k	TopKV2	【参数】 • input: n-D Tensor，n>=1, 最后一个维度大小必须大于 k • k：scalar >=1，类型：int32 • sorted：bool • name: optional, string 【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
65	tf.matmul	MatMul	【参数】 • a: 输入 tensor，2<=rank<=4，类型：float32 • b: 输入 tensor，类型与 rank 同 a • transpose_a: 如果属性为 True，a 在乘法前做转置；当 rank>2 时，属性为 false • transpose_b: 如果属性为 True，b 在乘法前做转置; • adjoint_a: 只支持 False; • adjoint_b: 只支持 False; • a_is_sparse: 只支持 False • b_is_sparse: 只支持 False， • name: optional, string 【约束】 • rank=2 时，对于矩阵运算[m,n] *[n,k]， 要求 n 不大于 1664，如需转置，要求转置后对应的 n

序号	Python API	C++ API	边界
			不大于 1664 • rank>2 时，输入 a 的 shape[-1] <=1024 【输出】 输出 Tensor，类型同 a 与 b
66	tf.multinomial	Multinomial	【参数】 • logits：2-D tensor，shape: [batch_size, num_classes] • num_samples：scalar，抽样个数 • seed：随机数种子，类型：int32, int64 • name: optional, string • output_dtype：输出 tensor，类型：整型，默认 int64 【约束】 seed 为 0 时产生随机数是动态的 【输出】 Tensor, shape: [batch_size, num_samples]
67	tf.math.multiply	Multiply	【参数】 • x：输入 tensor，类型：float32 • y：输入 tensor，类型同 x • name: optional, string 【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】 输出 tensor
68	tf.nn.avg_pool	AvgPool	【参数】 • value: 4-D tensor，格式：[batch, height, width, channels]，类型：float32

序号	Python API	C++ API	边界
			• 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: optional, string 【约束】 • KernelH<256, kernelW<256; • 当输出 tensor shape H、W 为 1 时，要求 input H * input W < 65536 kernelH<=inputH+padTop+padBottom kernelW<=inputW+padLeft+padRight padTop<windowH padBottom<windowH padLeft<windowW padRight<windowW 【输出】 输出 tensor，类型同 value
69	tf.nn.bias_add	BiasAdd	【参数】 • value: 输入 tensor • 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: optional, string 【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】 输出 Tensor，类型同 value
70	tf.nn.conv2d	Conv2D	【参数】 • value: 4-D tensor，格式：[batch, height, width, channels]，类型：float32

序号	Python API	C++ API	边界
			• 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: optional, string 【约束】 • (inputW + padWHead + 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_MAX • 0 <= Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 <= dilationsize < 256 StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1) 【输出】 tensor，类型同 Value
71	tf.nn.conv2d_transpose	Conv2DBackpropInput	【参数】 • value: 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

序号	Python API	C++ API	边界
			SAME，非空 • data_format: 类型: string，NHWC 或 NCHW，非空 • name: optional, string 【约束】 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/2； 2、conv_input_width=(反卷积输入 w - 1) *strideW + 1； 3、b = （conv_input_width） * filter_h * ALIGN(filter_num,16) * 4； 4、a + b <= 512KB 【输出】 输出 Tensor，类型同 value
72	tf.nn.depthwise_conv2d	DepthwiseConv2dNative	【参数】 • input: 4-D tensor • 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: optional, string 【约束】 • filterN=inputC=group • StrideW<=(inputW + padW)-((filterW - 1) * dilationW) + 1) 【输出】 4-D Tensor，形状与 data_format 一致，例如，对于 NHWC 格式，形状是[batch, out_height, out_width,

序号	Python API	C++ API	边界
			in_channels * channel_multiplier]
73	tf.nn.elu	Elu	【参数】 • features: 输入 tensor • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 features
74	tf.nn.fused_batch_norm	FusedBatchNorm	【参数】 • x: 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理 • name: optional, string 【约束】 scale, bias,mean,var 的 shape 只支持(1,C,1,1)，且 c 维度与 input 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
75	tf.nn.l2_normalize	L2Normalize	【参数】 • x：输入 tensor，类型：float32 • 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: optional, string • dim：axis 旧的别名(不推荐)

序号	Python API	C++ API	边界
			【约束】 H*W*2 < 32768； 【输出】 tensor，类型同 x
76	tf.nn.leaky_relu	LeakyRelu	【参数】 • features: 输入 tensor，类型：float32 • alpha: x <0 时激活函数的斜率. • name: optional, string 【约束】 无限制 【输出】 激活值
77	tf.nn.log_softmax （V310 新增）	LogSoftmax	【参数】 • logits: 非空 tensor，类型：float32 • axis: softmax 的维度，默认-1，表示最后一个维度 • name: optional, string • dim: axis 的弃用名 【约束】 • axis 输入范围[-rank,rank) 输入 4 维(NCHW)时可以针对每一维做 softmax： axis=0， 不支持 axis=1 即 channel 的话，c<=11136， h*w < 65536； axis=2 即 Height 场景下，W=1, 0<h<=16384； axis=3 即 Width 场景下，0<W<=16384； 【输出】 输出 tensor，类型与 shape 同 logits
78	tf.nn.lrn	LRN	【参数】 • input: 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

序号	Python API	C++ API	边界
			• beta: 可选参数，float 型，默认为 0.5；一个指数 • name: optional, string 【约束】 depth_radius >0，且必须为奇数； 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 tensor，类型同 input
79	tf.nn.max_pool	MaxPool	同 tf.nn.avg_pool
80	tf.nn.relu	Relu	【参数】 • features：输入 tensor，类型：float16，float32 • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 features
81	tf.nn.relu6	Relu6	【参数】 • features: 输入 tensor，类型：float16, float32 • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 features
82	tf.nn.selu	Selu	【参数】 • features： 输入 tensor，类型：float32 • name: optional, string 【约束】 无约束 【输出】 输出 tensor，类型同 features
83	tf.nn.softmax	Softmax	【参数】 • logits: 非空 tensor，类型: float32 • axis: 在该维度上执行 softmax; 默认值为-1，表

序号	Python API	C++ API	边界
			示最后一个维度，不超过 logits 维度 • name: optional, string • dim: axis 的已弃用的别名 【约束】 • axis 输入范围[-rank,rank) 输入 4 维(NCHW)时可以针对每一维做 softmax： axis=0，n<=28544； axis=1 即 channel 的话，c<=11136， h*w < 65536； axis=2 即 Height 场景下，W=1, 0<h<=16384； axis=3 即 Width 场景下，0<W<=16384； 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 tensor，类型和 shape 同 logits
84	tf.nn.softplus	Softplus	【参数】 • features: 输入 tensor，类型：float32 • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 features
85	tf.nn.softsign	Softsign	【参数】 • features: 输入 tensor，类型：float32 • name: optional, string 【约束】 无限制 【输出】 输出 tensor，类型同 features
86	tf.pad	Pad PadV2 MirrorPad	【参数】 • tensor: 4-D tensor, 类型：float32 • paddings: tensor, 常量，类型：int32 • mode: string, 值为 CONSTANT, 或 REFLECT, 或 SYMMETRIC。当 mode=CONSTANT，若 constant_values 等于 0 时，c++接口为 Pad，否则，c++接口为 PadV2；当 mode=REFLECT

序号	Python API	C++ API	边界
			或 SYMMETRIC，c++接口是 MirrorPad。 • name: optional, string • 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，类型与 tensor 相同
87	tf.placeholder	Placeholder	【参数】 • dtype: 类型(必须) • shape: Tensor 的维度和大小 • name: optional, string 【约束】 无限制 【输出】 输出 tensor
88	tf.range	Range	【参数】 • start: 开始, scalar，类型：float32, int32, 必须为常量 • limit: 结束, scalar，类型：float32, int32, 必须为常量 • delta: 步长, scalar，类型：float32, int32, 必须为常量 • dtype：返回 Tensor 的类型 • name: optional, string 【约束】 无限制 【输出】 输出 1-D tensor
89	tf.realdiv	RealDiv	【参数】 • x: 输入 tensor, 类型: float32 • y: 输入 tensor, 类型: float32 • name: optional, string 【约束】 无限制

序号	Python API	C++ API	边界
			【输出】 输出 tensor，类型同 x
90	tf.math.reduce_sum	Sum	【参数】 • input_tensor: 输入 tensor • axis: reduce 的维度，类型：int32 • keepdims：bool 值，是否保留维度 • name: optional, string • reduction_indices：axis 的旧名（弃用） • keep_dims: 不推荐使用，参数 keepdims 别名 【输出】 输出 Tensor，类型同 tensor
91	tf.reshape	Reshape	【参数】 • tensor: 输入 Tensor； • shape: 定义输出的 shape，常量 Tensor，类型：int64, int32 • name: optional, string 【输出】 输出 Tensor，类型同输入
92	tf.reverse（别名：tf.reverse_v2）	ReverseV2	【参数】 • tensor: 输入 tensor；类型：int8，int16，int32，int64，float16，float32 • axis: tensor, reverse 的维度，类型：int32 • name: optional, string 【输出】 输出 Tensor，类型同 tensor
93	tf.reverse_sequence	ReverseSequence	【参数】 • input：输入 tensor • seq_lengths：1-D tensor；类型：int32, int64 • seq_axis: scalar, 类型：整型 • batch_axis: scalar（optional），默认为 0, 类型：整型 • name: optional, string 【输出】 输出 tensor，类型同 input
94	tf.scatter_nd（V310 新	ScatterNd	【参数】 • indices: tensor, 类型: int32，不支持负数，且最

序号	Python API	C++ API	边界
	增）		大值要求小于 shape[0] • updates: tensor，类型：float32 • shape: 1-D const，类型: int32 • name: optional, string 【输出】 输出 tensor，类型同 updates
95	tf.shape	Shape	【参数】 • input: 输入 tensor • name: optional, string • out_type: 指定输出类型：int32, int64 (optional, 默认为 int32) 【输出】 输出 tensor，输出数据类型为 out_type
96	tf.sigmoid	Sigmoid	【参数】 • x: 输入 tensor • name: optional, string 【输出】 输出 tensor，类型同 value
97	tf.math.sign（V310 新增）	Sign	【参数】 • x: 输入 tensor，类型: float32 • name: optional, string 【输出】 输出 tensor，类型同 x
98	tf.size	Size	【参数】 • input: 输入 tensor，类型：float32 • name: optional, string • out_type: optional, 指定输出类型，默认 int32 【输出】 输出 tensor，类型由 out_type 指定
99	tf.slice	Slice	【参数】 • input: 输入 tensor • begin: tensor，类型：int32, int64 • size: tensor，类型：int32, int64 • name: optional, string 【输出】

序号	Python API	C++ API	边界
			输出 tensor，类型同 input
100	tf.space_to_batch_nd	SpaceToBatchND	【参数】 • input：n-D Tensor， input_shape = [batch] + spatial_shape + remaining_shape,其中 spatial_shape 有 M 维度；类型：float32 • block_shape： 1-D Tensor，类型：int32；shape 为[M], 所有值必须>=1 • paddings： Tensor，类型：int32；shape 为[M, 2], 且要求`input_shape[i + 1] + pad_start + pad_end. 能够被 block_shape[i]整除 【约束】 • 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 Tensor，类型同 input
101	tf.space_to_depth	SpaceToDepth	【参数】 • input：输入 Tensor, 类型：float32 • block_size：scalar，类型：int32，值>=2 • data_format：optional, string, NHWC 或 NCHW, 默认值是 NHWC • name: optional，string 【输出】 输出 Tensor，类型同 input
102	tf.split	Split SplitV	【参数】 • value: 输入 Tensor，类型：float32, bool, int32 • num_or_size_splits：若为 scalar，表示指定分割个数，c++接口为 Split；若为 1-D Tensor，表示指定分割大小，c++接口为 SplitV • axis：scalar，类型：int32，指定分割维度 • name: optional，string 【输出】 List，包含 split 后的各 Tensor

序号	Python API	C++ API	边界
103	tf.math.square	Square	【参数】 • x: 输入 Tensor，类型： float32 • name: optional，string 【输出】 输出 Tensor，类型同 x
104	tf.squeeze	Squeeze	【参数】 • input: 输入 Tensor • axis: 可选 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: optional，string • squeeze_dims: 不推荐使用的参数，axis 和 dim 不能同时存在 【输出】 输出 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
105	tf.stack	Pack	【参数】 • values: tensor list，要求每个 tensor 形状与类型相同，类型：float32，int32 • axis：整数，沿 axis 维度堆叠，默认是第一维 • name: optional，string 【输出】 输出 Tensor，类型同 values
106	tf.strided_slice	StridedSlice	【参数】 • input_: 输入 Tensor，类型：float32 • begin: 1-D Tensor，类型: int32 • end: 1-D Tensor，类型: int32 • strides: 1-D Tensor，类型: int32 • begin_mask: 标量，类型: int32 • end_mask: 标量，类型: int32 • ellipsis_mask: 标量，类型: int32 • new_axis_mask:: 标量，类型: int32 • shrink_axis_mask: 标量，类型: int32 • var：与 input_ 或 None 对应的变量 • name: optional,string 【约束】 在 shrink_axis_mask 掩码场景下，只支持正序 mask

序号	Python API	C++ API	边界
			不支持 new_axis_mask 【输出】 输出 Tensor，类型同 input_
107	tf.subtract	Subtract	【参数】 • x: 输入 tensor，类型：float32 • y: 输入 tensor，类型同 x； • name: optional,string 【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】 输出 Tensor
108	tf.math.tanh	Tanh	【参数】 • x: 输入 Tensor，类型：float16, float32 • name: optional，string 【输出】 输出 Tensor，类型同 x
109	tf.tile	Tile	【参数】 • input: 输入 Tensor，维度大于等于 1 • multiples: 1-D Tensor，长度须和 input 的秩相同，数据类型：`int32`，必须为常量 • name: optional，string; 【输出】 1 个 Tensor
110	tf.transpose	Transpose	【参数】 • a: 输入 Tensor • perm: a 的维数的排列

序号	Python API	C++ API	边界
			• name: 名称（可选） • conjugate: optional，类型: bool，默认为 False；不支持 True 【输出】 被转置后的 Tensor
111	tf.unstack	Unpack	【参数】 • value: 输入 Tensor，类型: float32, int32, bool • num: 整数，表示 axis 维度的长度，类型：int32, 默认为 None • axis: 整数，沿 axis 拆分，默认为 0 • name: optional,string 【输出】 从 value 中拆分出的包含 Tensor 对象的列表
112	tf.where	Where	【参数】 • condition: tensor，类型：bool • x: None • y: None • name: optional，string 【输出】 tensor，其 shape 为二维，第一维的大小和输入中元素值为 true 的个数相同
113	tf.where	Select	【参数】 • condition: tensor，类型：bool • x: tensor，类型：float32, int32, uint8, bool • y: tensor，shape 和类型同 x； • name: optional，string 【约束】 1：condition,x,y 三者 shape 相同 2：condition 为 1-D, x 和 y 的 shape 相同（rank >=1），且 x,y 的第 0 维的大小（shape[0]）和 condition 的大小相同 【输出】 输出 tensor, shape 和类型同 x
114	from tensorflow.python.ops import	Switch	【参数】 • data: 输入 Tensor • pred：scalar，类型：bool

序号	Python API	C++ API	边界
	control_flow_ops control_flow_ops.switch （V310 新增）		• dtype: optional 指定输出 Tensor 类型 • name: optional ,string; 【约束】 算子不能独立支持，需和 merge 一起使用,参考 Merge 【输出】 输出 Tensor，(output_false, output_true): 若 pred 为 true, forward 返回 output_true，否则，返回 output_false
115	from tensorflow.python.ops import control_flow_ops control_flow_ops.merge （V310 新增）	Merge	【参数】 • inputs: 输入 tensors，至少有一个是可用的 • name: optional, string 【约束】 算子不能独立支持，需和 Switch 一起使用，并且 Merge 不能作为最后的输出 【输出】 一个元组，包含所选 tensor 以及其索引
116	tf. matmul （V320 新增）	BatchMatMul	【参数】 • x: 输入 3-4D Tensor 左矩阵 • y: 输入 3-4D Tensor 右矩阵，类型与 rank 同 x • adj_x: Bool，输入 x tensor 的最后两维是否转置 • adj_y: Bool，输入 y tensor 的最后两维是否转置 【约束】 • 输入 tensor，类型：float32 • adj_x: 默认为 false，目前仅支持 false（暂不支持 true） • adj_y: 默认为 false，如果属性为 True，y 在乘法前做转置 【输出】 一个 Tensor，类型同 x 与 y
117	tf.contrib.layers.layer_norm （V320 新增）		【参数】 • inputs：2D-4D tensor，类型：float，支持的 format: NHWC • center：bool 型，给 inputs 归一化添加偏移，默认为 True • scale：bool 型，给 inputs 归一化乘以比例系数，默认为 True

序号	Python API	C++ API	边界
			• activation_fn：对结果进行激活，默认为 None • reuse：表示是否重用该层及其变量，默认为 None，且当前仅支持 None • variables_collections：可选变量的集合，默认为 None，且当前仅支持 None • outputs_collections： 添加输出的集合，默认为 None，且当前仅支持 None • trainable：如果为 True,将变量添加到图集合 GraphKeys，默认为 None，且当前仅支持 None • begin_norm_axis：归一化起始轴，当前固定为对 CHW 的数据做归一化，输入为 4D 时仅支持 1，输入为 2D、3D 时仅支持 0 • begin_params_axis：设置偏移以及比例因子数据的维度起始轴信息，当前固定为对 CHW 的数据添加偏置或乘比例因子，输入为 4D 时仅支持 1，输入为 2D、3D 时仅支持 0 • scope：变量的可选范围，默认为 None，且当前仅支持 None 【输出】 一个 Tensor，与输入 inputs 的 shape 和 dtype 相同，对 inputs 输入 N 范围内的数据 CHW 做归一化。
118	tf.stop_gradient （V320 新增）	StopGradient	【参数】 • input：输入 tensor 【输出】 一个 Tensor，输出和输入的 tensor 一样
119	tf.contrib.layers.instance_norm （V320 新增）		【参数】 • inputs：4D tensor，类型：float32，支持的 format: NCHW 和 NHWC • center：bool 型，给 inputs 归一化添加偏移，默认为 True • scale：bool 型，给 inputs 归一化乘以比例系数，默认为 True • epsilon：添加一个小的浮点数到方差作为除数，避免除数为 0；支持的数据类型：float；默认值为 1e-06，最小值为 1e-7 • activation_fn：对结果进行激活，默认为 None • param_initializers：用于 beta、gamma 以及方差和均值的初始化，默认为 None

序号	Python API	C++ API	边界
			• reuse：表示是否重用该层及其变量，默认为 None，且当前仅支持 None • variables_collections：可选变量的集合，默认为 None，且当前仅支持 None • outputs_collections： 添加输出的集合，默认为 None，且当前仅支持 None • trainable：如果为 True,将变量添加到图集合 GraphKeys，默认为 True • data_format：数据格式，支持 NHWC 和 NCHW，默认为 NHWC • scope：变量的可选范围，默认为 None，且当前仅支持 None 【输出】 一个 Tensor，与输入 inputs 的 shape 和 dtype 相同，对 inputs 输入 NC 范围内的 HW 一个面做归一化。
120	tf.random.normal （V320 新增）	RandomNormal	【参数】 • shape：1-D tensor，常量，表示输出 tensor 的 shape，类型：int32 • mean：0-D 标量，正态分布的平均值；类型：float16 • stddev：0-D 标量，正态分布的标准偏差，类型：float16 • seed：输出 tensor 的随机数种子，类型：int32，可以通过 tf.random.set_random_seed 接口设置该参数，当前会忽略设置的 seed 值，使用 0。 • seed2：表示生成的 tensor 随机数种子，类型：int32，该值是 tf.random.random_normal 接口中的 seed 参数值，如果该参数不设置或者设置为 0，则每次计算的结果是不同的，当前会忽略设置的 seed2 值，使用 0。 【输出】 Tensor，类型：float16
121	tf.random.shuffle （V320 新增）	RandomShuffle	【参数】 • value：Tensor（支持常量和非常量输入），类型：float16, float32, double, int8, int16, int32, int64, uint8, uint16, bool • seed：输出 tensor 的随机数种子，类型：int32，可以通过 tf.random.set_random_seed 接口

序号	Python API	C++ API	边界
			设置该参数，当前会忽略设置的 seed 值，使用 0。 • seed2：表示生成的 tensor 随机数种子，类型：int32，该值是 tf.random.random_shuffle 接口中的 seed 参数值，如果该参数不设置或者设置为 0，则每次计算的结果是不同的，当前会忽略设置的 seed2 值，使用 0。 【输出】 Tensor，类型同 value
122	tf.random.uniform （V320 新增）	RandomUniformInt	【参数】 • shape：1-D 常量 或者输入是 Shape 算子，类型：int32 • minval：0-D （标量），该值表示生成的 tensor 的最小值（包括该值），类型：int32 • maxval：0-D （标量），该值表示生成的 tensor 的最大值（不包括该值），类型：int32 • dtype：指定输出 tensor 的 dtype，类型：int32 • seed：输出 tensor 的随机数种子，类型：int32，可以通过 tf.random.set_random_seed 接口设置该参数，当前会忽略设置的 seed 值，使用 0。 • seed2：表示生成的 tensor 随机数种子，类型：int32，该值是 tf.random.uniform 接口中的 seed 参数值，如果该参数不设置或者设置为 0，则每次计算的结果是不同的，当前会忽略设置的 seed2 值，使用 0。 【输出】 Tensor，类型：int32
123	tf.random.uniform （V320 新增）	RandomUniform	【参数】 • shape：1-D 常量 或者输入是 Shape 算子，类型：int32 • minval：0-D （标量），该值表示生成的 tensor 的最小值（包括该值），类型：float32 • maxval：0-D （标量），该值表示生成的 tensor 的最大值（不包括该值），类型：float32 • dtype：指定输出 tensor 的 dtype，类型：float32 • seed：输出 tensor 的随机数种子，类型：int32，可以通过 tf.random.set_random_seed 接口设置该参数，当前会忽略设置的 seed 值，使用 0。

序号	Python API	C++ API	边界
			• seed2：表示生成的 tensor 随机数种子，类型：int32，该值是 tf.random.uniform 接口中的 seed 参数值，如果该参数不设置或者设置为 0，则每次计算的结果是不同的，当前会忽略设置的 seed2 值，使用 0。 【输出】 Tensor，类型：float32
124	tf.math.argmin （V320 新增）	ArgMin	【参数】 • input： Tensor（支持常量和非常量输入），类型：float32, uint8, int32 • axis： Tensor, 常量，类型：int32 • dimension：被废弃，使用 axis • out_type ：输出 Tensor 的类型：int32 • name：optional, string 【输出】 1 个 Tensor，输出类型为 out_type
125	tf.rank （V320 新增）	Rank	【参数】 • input：Tensor（支持常量和非常量输入），类型：int32, float32, uint8, bool • name：optional, string 【输出】 Tensor，类型：int32
126	tf.truncatemod （V320 新增）	Truncatemod	【参数】 • x：Tensor（支持常量和非常量输入），类型：int32, float32 • y：Tensor，类型：int32, float32 • name：optional, string 【输出】 Tensor，shape 与类型同 x
127	tf.math.unsorted_segment_sum （V320 新增）	UnsortedSegmentSum	【参数】 • data：Tensor（支持常量和非常量输入），类型：int32, float32, uint8 • segment_ids：必选，Tensor（支持常量和非常量），shape 与 data 一致，用于指定结果为 out[i]，类型：int32 • num_segments：必选，0-D 常量输入，取值为 segment_ids 的长度大小，类型：int32

序号	Python API	C++ API	边界
			• name：optional, string 【约束】 num_segments 大于等于 segment_id 分组的个数 【输出】 Tensor，类型同 data
128	tf.math.cumsum （V320 新增）	Cumsum	【参数】 • x：Tensor（支持常量和非常量输入），类型: float32, uint8, int32 • axis：Tensor，范围为 [-rank(x), rank(x)]，类型：int32，默认为 0 • exclusive：可选，bool 型，若为 true，输出的第一个值从 0 开始，若为 false，采用第一个元素进行初始化；默认为 false • reverse：可选，bool 型；设置为 true 时，cumsum 将以反方向执行；默认为 false • name：optional, string 【输出】 Tensor，类型同 x
129	tf.math.cumprod （V320 新增）	Cumprod	【参数】 • x：Tensor，类型：float32, uint32, uint8 • axis：Tensor，范围为 [-rank(x), rank(x)]，类型：int32，默认为 0 • exclusive：可选，bool 型，若为 true，输出的第一个值从 0 开始，若为 false，采用第一个元素进行初始化；默认为 false • reverse：可选，bool 型；设置为 true 时，cumprod 将以反方向执行；默认为 false • name：optional, string 【输出】 Tensor，类型同 x
130	tf.nn.conv1d （V320 新增）		【参数】 • value：输入，3-D Tensor ，类型：float32 • filters：输入，3-D Tensor，常量，类型：float32 • stride：int 型 • padding：填充类型，'SAME' or 'VALID' • use_cudnn_on_gpu：可选，bool 型，默认为 true • data_format：可选，NWC 或者 NCW，默认

序号	Python API	C++ API	边界
			NWC • name：optional, string 【输出】 Tensor，类型同 value
131	tf.nn.atrous_conv2d （V320 新增）		【参数】 • value：输入，4-D Tensor ，类型：float32，数据格式：NHWC • filters：输入，4-D Tensor，常量，类型：float32，类型和 shape 与 value 一致 • rate：在 H 和 W 上对输入值进行采样的步长，类型：int32 • padding：填充类型，'SAME' or 'VALID' • name：optional, string 【输出】 Tensor，类型同 value
132	tf.math.reduce_any （V320 新增）	Any	【参数】 • input_tensor: Tensor，类型：bool • axis: reduce 的维度轴向，范围：[-rank(input_tensor),rank(input_tensor)] • keepdims：标量，类型：bool，默认值 false • name: optional, string • reduction_indices：axis 的旧名字 • keep_dims: keepdims 的弃用名 【输出】 Tensor，类型：bool
133	tf.math.logical_xor （V320 新增）		【参数】 • x: Tensor（支持常量和非常量输入），类型：bool • y: Tensor（支持常量和非常量输入），类型：bool • name: optional, string 【约束】 不支持双向广播 【输出】 Tensor，类型同输入
134	tf.nn.fractional_max_po	Fractional MaxPool	【参数】 • value: 输入，4-D Tensor，类型：flaot32, int32,

序号	Python API	C++ API	边界
	ol （V320 新增）		int64；仅支持 NHWC • pooling_ratio: list，类型：float32 ，表示池化窗口的长宽比率，长度必须等于 4，ratio 值必须不小于 1.0，且 ratio[0]和 ratio[3]必须等于 1.0 • pseudo_random：可选，bool 型，表示生成的 rowSeq 和 colSeq 是否伪随机，默认为 false • overlapping：可选，bool 型，表示 pooling 的窗口之间是否允许重叠，默认为 false • deterministic：可选，bool 型，表示生成的 rowSeq 和 colSeq 是否确定，默认为 false • seed：输出 tensor 的随机数种子，类型：int32 • seed2：表示生成的 tensor 随机数种子，类型：int32 【约束】 如果 deterministic 设置为 false，则 seed 和 seed2 必须同时设置为 0，此时表示生成真随机数，每次运行的结果是不一样的；如果 deterministic 设置为 true，则 seed 和 seed2 不能同时设置为 0，此时表示生成伪随机数，每次运行的结果是一样的。 【输出】 y：Tensor，类型同 value row_pooling_sequence：Tensor，类型：int64 col_pooling_sequence：Tensor，类型：int64
135	tf.nn.fractional_ avg _pool （V320 新增）	FractionalAvgPool	【参数】 • value: 输入，4-D Tensor，类型：float32, int32, int64；仅支持 NHWC • pooling_ratio: list，类型：float32 ，表示池化窗口的长宽比率，长度必须等于 4，ratio 值必须不小于 1.0，且 ratio[0]和 ratio[3]必须等于 1.0 • pseudo_random：可选，bool 型，表示生成的 rowSeq 和 colSeq 是否伪随机，默认为 false • overlapping：可选，bool 型，表示 pooling 的窗口之间是否允许重叠，默认为 false • deterministic：可选，bool 型，表示生成的 rowSeq 和 colSeq 是否确定，默认为 false • seed：输出 tensor 的随机数种子，类型：int32 • seed2：表示生成的 tensor 随机数种子，类型：int32 【约束】 如果 deterministic 设置为 false，则 seed 和 seed2 必

序号	Python API	C++ API	边界
			须同时设置为 0，此时表示生成真随机数，每次运行的结果是不一样的；如果 deterministic 设置为 true，则 seed 和 seed2 不能同时设置为 0，此时表示生成伪随机数，每次运行的结果是一样的。 【输出】 y：Tensor，类型同 value row_pooling_sequence：Tensor，类型：int64 col_pooling_sequence：Tensor，类型：int64
136	tf.math.not_equal （V320 新增）	NotEqual	【参数】 • x：输入 Tensor，类型：float32 • y：输入 Tensor，类型：float32 【输出】 Tensor，类型：bool
137	tf.math.less_equal （V320 新增）	LessEqual	【参数】 • x：输入 Tensor，类型：float32 • y：输入 Tensor，类型：float32 【输出】 Tensor，类型：bool
138	tf.quantization.quantize （V320 新增）	QuantizeV2	【参数】 • input：Tensor（支持常量和非常量输入），类型：float32 • min_range：Tensor，常量，input 数据可能的最小值，需小于等于 0，类型：float32 • max_rang：Tensor，常量，input 数据可能的最大值，类型：float32 • T：指定量化的数据类型，类型：uint8 • mode：可选，类型：string；值可为："MIN_COMBINED", "MIN_FIRST", "SCALED"，目前仅支持 MIN_COMBINED • round_mode：可选，类型：string；值可为："HALF_AWAY_FROM_ZERO", "HALF_TO_EVEN"，目前仅支持 HALF_AWAY_FROM_ZERO 【输出】 Tensor，类型：uint8
139	tf.quantization.dequantize	Dequantize	【参数】 • input：Tensor（支持常量和非常量输入），类

序号	Python API	C++ API	边界
	（V320 新增）		型：quint8 • min_range：Tensor，常量，input 数据可能的最小值，需小于等于 0，类型：float32 • max_rang：Tensor，常量，input 数据可能的最大值，类型：float32 • mode：可选，类型：string；值可为："MIN_COMBINED", "MIN_FIRST", "SCALED"，目前仅支持 MIN_COMBINED 【输出】 Tensor，类型：float32
140	tf.math.floor_div （V320 新增）	FloorDiv	【参数】 • x：Tensor，类型：int32, float, uint8 • y：Tensor，类型：int32, float, uint8 【输出】 Tensor，shape 与类型同 x
141	tf.quantization.fake_quant_with_min_max_vars_per_channel （V320 新增）	FakeQuantWithMinMaxVarsPerChannel	【参数】 • x：1-D Tensor（支持常量和非常量输入），类型：float32 • min：1-D，常量，最小值，类型：float32 • max：1-D，常量，最大值，类型：float32 • num_bits：可选，量化范围的位宽；int 型， 默认值为 8 • narrow_range：可选，量化范围标识；bool 型，默认值为 false 【输出】 Tensor，shape 与类型同 x
142	tf.one_hot （V320 新增）	OneHot	【参数】 • indices：Tensor（支持常量和非常量输入），表示输入的多个数值，类型：uint8, int32 • depth：Tensor，常量，输出的尺寸，类型：int32 • on_value：可选，Tensor（支持常量和非常量输入），类型：uint8, int32, float, bool，默认为 1 • off_value：可选，与 on_value 的 dtype 保持一致，默认为 0 • axis：可选，int 型， 默认值为-1 • dtype：类型

序号	Python API	C++ API	边界
			【输出】 返回一个 tensor，在 axis 轴上添加维度，类型: uint8, int32, float, double, bool
143	tf.math.segment_max （V320 新增）	Segment Max	【参数】 • data：Tensor，表示输入的 tensor，类型：float32 • segment_ids：1-D 常量；表示分段，相同的 id 为一段；类型：int32 【约束】 1、输入 data 的第 0 维 等于输入 segment_ids 的元素个数； 2、segment_ids 不支持负数索引，且从 0 开始并且升序排列； 【输出】 Tensor
144	tf.math.segment_min （V320 新增）	Segment Min	【参数】 • data：Tensor，表示输入的 tensor，类型：float32 • segment_ids：1-D 常量；表示分段，相同的 id 为一段；类型：int32 【约束】 1、输入 data 的第 0 维 等于输入 segment_ids 的元素个数； 2、segment_ids 不支持负数索引，且从 0 开始并且升序排列； 【输出】 Tensor
145	tf.math.segment_mean （V320 新增）	Segment Mean	【参数】 • data：Tensor，表示输入的 tensor，类型：float32 • segment_ids：1-D 常量；表示分段，相同的 id 为一段；类型：int32 【约束】 1、输入 data 的第 0 维 等于输入 segment_ids 的元素个数； 2、segment_ids 不支持负数索引，且从 0 开始并且升序排列； 【输出】

序号	Python API	C++ API	边界
			Tensor
146	tf.math.segment_ prod（V320 新增）	SegmentProd	【参数】 • data：Tensor，表示输入的 tensor，类型：float32 • segment_ids：1-D 常量；表示分段，相同的 id 为一段；类型：int32 【约束】 1、输入 data 的第 0 维 等于输入 segment_ids 的元素个数； 2、segment_ids 不支持负数索引，且从 0 开始并且升序排列； 【输出】 Tensor
147	tf.math.segment_sum（V320 新增）	SegmentSum	【参数】 • data：Tensor，表示输入的 tensor，类型：float32 • segment_ids：1-D 常量；表示分段，相同的 id 为一段；类型：int32 【约束】 1、输入 data 的第 0 维等于输入 segment_ids 的元素个数； 2、segment_ids 不支持负数索引，且从 0 开始并且升序排列； 【输出】 Tensor
148	tf.zeros_like（V310 新增）	ZerosLike	【参数】 • x：0-4D Tensor，支持常量和 tensor，类型：float32 【输出】 Tensor，shape 与类型同 x
149	tf.identity（V310 新增）	Identity	【参数】 • x：Tensor 【输出】 Tensor，shape 与类型同 x
150	tf.Assert（V310 新增）	Assert	【参数】 • condition：评估条件 • data： list of tensors，当 condition 为 false 时输

序号	Python API	C++ API	边界
			出的数据 • summarize：可选，默认为 None，决定输出 data 为几位数 【约束】 构建模型时，为确保 assert 执行，需附加一个依赖关系，通常与 tf.control_dependencies([assert_op])连用 【输出】 返回一个 operation，如果条件为 false，报 tf.errors.InvalidArgumentError
151	tf.keras.layers.PReLU （V310 新增）		【参数】 • x：Tensor，类型：float • slope：训练出来的系数 【输出】 Tensor，shape 与类型同 x
152	*GELU(x)=0.5x(1+tanh[((2/π)^0.5)*(x+0.044715x^3)])* (V320 新增)		【参数】 • features： 输入 tensor, 类型: float32 • name: optional, string 【输出】 输出 tensor，类型同 features
153	tf.nn.bidirectional_dynamic_rnn （V320 新增）		【输入】 • cell_fw: 前向计算单元，可通过 tf.nn.rnn_cell.BasicLSTMCell 方式生成 • cell_bw:反向计算单元，可通过 tf.nn.rnn_cell.BasicLSTMCell 方式生成 • inputs：RNN 输入，与 time_major 的设置有关，默认 time_major 为 False，此时 inputs 必须形状是 BTX（[batch_size,max_time,depth]）的张量或此类元素的嵌套元祖 ,time_major 为 True 时，其形状需要是 TBX（[max_time,batch_size,depth]）的张量或此类元素的嵌套元组。 • sequence_length: （可选）一个 int32 的向量，尺寸为[batch_size]若指定，则包含单个 batch 里要设定的各个序列的有效长度，如果未指定，使用最大长度（max_time），如果序列的长度不同，请使用此参数，它是各个序列长度构成的数组，该参数不支持常量输入。 • initial_state_fw: （可选）前向的初始化状态，

序号	Python API	C++ API	边界
			当前只支持为 none 的场景。 • initial_state_bw: （可选）后向的初始化状态，当前只支持为 none 的场景。 • dtype: （可选）初始状态和预期输出的的数据类型，当前支持 fp32 的格式。 • time_major:设置输入输出张量的形状格式，默认 time_major 为 False，此时 inputs 必须形状是 BTX（[batch_size,max_time,depth]）的张量或此类元素的嵌套元祖 ,time_major 为 True 时，其形状需要是 TBX • scope: 指定生成的 rnn 子图名称，默认为 bidirectional_rnn，用户可自行设置 【约束】 fw、bw 的基础 Cell 的所有参数需要一致，即 cell 的名称，激活函数，num_units 等 API 中所有需要输入的参数必须全部一致。 当前仅支持 BasicLSTMCell，激活函数包含：Tanh、sigmoid、relu、relu6。 动态 rnn 的参数 initial_state_fw 与 initial_state_bw 只支持为 none。 state_is_tuple 只支持为 True 的场景。 BasicLSTMCell 的 num_units 当前仅支持为 16 整数倍。 【输出】 支持分别输出前向 output 和后向 output； 支持前向 Cell state，Hidden state 和后向 Cell state，Hidden state 分成 4 路输出。
154	tf.nn.static_bidirectional_rnn （V320 新增）		【输入】 • cell_fw: (必填)前向计算单元，可通过 tf.nn.rnn_cell.BasicLSTMCell 方式生成; • cell_bw: (必填)反向计算单元，可通过 tf.nn.rnn_cell.BasicLSTMCell 方式生成; • inputs: (必填) RNN 输入，长度为 T 的列表，每个元素是一个 tensor(shape[batch_size,input_size]), T 代表 max_time； • initial_state_fw: (可选)前向 RNN 的初始状态，当前只支持为 none 的场景; • initial_state_bw: (可选)反向 RNN 的初始状态，当前只支持为 none 的场景；

序号	Python API	C++ API	边界
			• dtype: (可选)初始状态的数据类型，当前只支持 fp32 类型。 • sequence_length: （可选）一个 int32 的向量，尺寸为[batch_size]若指定，则包含单个 batch 里要设定的各个序列的有效长度，如果未指定，使用最大长度（max_time），如果序列的长度不同，请使用此参数，它是各个序列长度构成的数组，该参数不支持常量输入; • scope: (可选)指定生成的 rnn 子图名称，默认为 bidirectional_rnn，用户可自行设置。 【约束】 fw、bw 的基础 Cell 的所有参数需要一致，即 cell 的名称，激活函数，num_units 等 API 中所有需要输入的参数必须全部一致。 当前仅支持 BasicLSTMCell，激活函数包含：Tanh、sigmoid、relu、relu6。 静态 rnn 的参数 initial_state_fw 与 initial_state_bw 只支持为 none。 state_is_tuple 只支持为 True 的场景。 BasicLSTMCell 的 num_units 当前仅支持为 16 整数倍。 【输出】 支持 T 个输出通过 stack 之后输出; 支持前向 Cell state，Hidden state 和后向 Cell state，Hidden state 分成 4 路输出。

2.4 AndroidNN 算子边界

序号	Operation	含义	边界
1	ANEURALNETWORKS_ABS （V310 新增）	绝对值	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor 【约束】

序号	Operation	含义	边界
			输入 0 的最后一个维度不超过 400 【输出】 输出 0：输出 tensor ， shape 同输入 0
2	ANEURALNETWORKS_ADD	加法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_QUANT8_ASYMM TENSOR_FLOAT16（API29） 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: FuseCode{ ANEURALNETWORKS_FUSED_NONE = 0, ANEURALNETWORKS_FUSED_RELU = 1, ANEURALNETWORKS_FUSED_RELU1 = 2, ANEURALNETWORKS_FUSED_RELU6 = 3 } 【约束】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
3	ANEURALNETWORKS_ARGMAX （V310 新增）	查找最大值索引	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：scalar，TENSOR_INT32 【约束】 无限制 【输出】 输出 0：(n-1)D tensor ， type ： TENSOR_INT32
4	ANEURALNETWORKS_ARGMIN	查找最小值索引	【输入】 • 支持的 tensor OperandCode：

序号	Operation	含义	边界
	（V310 新增）		TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：scalar，TENSOR_INT32 【约束】 无限制 【输出】 输出 0：(n-1)D tensor ， type ： TENSOR_INT32
5	ANEURALNETWORKS_AVERAGE_POOL_2D	平均池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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，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 • 输入 10：scalar ， optional ， BOOL，指定输入输出数据格式，只支持默认格式“NHWC"

序号	Operation	含义	边界
			（API29） 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 值，为 SAME 或 VALID 之一： PaddingCode{ ANEURALNETWORKS_PADDING_SAME = 0, ANEURALNETWORKS_PADDING_VALID = 1 } • 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 • 输入 7：scalar ，optional ，BOOL，指定输入输出数据格式，只支持默认格式“NHWC"（API29） 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth] 【约束】 • KernelH<256, kernelW<256; • 当输出 tensor shape H、W 为 1 时，要求 input H * input W < 65536
6	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） TENSOR_FLOAT16(API29) TENSOR_QUANT8_ASYMM 支持的 tensor 维数：4 • 输入 0：4-D tensor • 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： optional ， BOOL 类型，只支持默认

序号	Operation	含义	边界
			格式“NHWC"（API29） 【约束】 输入 0 的 shape 乘积需小于 500 【输出】 输出 0：tensor，OperandCode 与输入 0 相同
7	ANEURALNETWORKS_CAST （V310 新增）	类型转换	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：n-D tensor
8	ANEURALNETWORKS_CONCATENATION	数据按指定维度拼接	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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 TENSOR_QUANT8_ASYMM 类型，所有输入 tensors 的 scale 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[2,32] • 输入 0~n-1：不支持常量输入 • QUANT8 输入，不支持 axis=2 【输出】 输出 0：tensor, OperandCode 与输入相同，输出 shape 为 [D0, D1, ..., sum(Daxis(i)), ..., Dm]
9	ANEURALNETWORKS_CONV_2D	卷积	【输入】 • 支持的 tensor OperandCode：

序号	Operation	含义	边界
			TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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 TENSOR_QUANT8_ASYMM 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 • 输入 9：scalar，INT32，指定 activation • 输入 10：optional，scalar，BOOL，默认为 false，设置为 true 时表示输入 0 和输出 0 的数据格式为 NCHW • 输入 11：optional，scalar，INT32，指定 width 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 12 • 输入 12：optional，scalar，INT32，指定 height 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 11 隐式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height,

序号	Operation	含义	边界
			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 TENSOR_QUANT8_ASYMM 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 维度，0 < Stride < 64 • 输入 5：scalar，INT32，指定 stride 的 height 维度，0 < Stride < 64 • 输入 6：scalar，INT32，指定 activation • 输入 7：optional，scalar，BOOL，默认为 false，设置为 true 时表示输入 0 和输出 0 的数据格式为 NCHW • 输入 8：optional，scalar，INT32，指定 width 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 9 • 输入 9：optional，scalar，INT32，指定 height 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 8 【约束】 • 输入类型为 QUANT8 时，仅支持输入 0 和输出 0 为 NHWC 格式 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth_out]；
10	ANEURALNETWORKS_DEPTH_TO_SPACE	数据重新排列从深度转为空间数据块	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：4 支持 "NHWC"数据布局 • 输入 0：4-D tensor，shape [batches, height,

序号	Operation	含义	边界
			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 • 输入 2：optional ， BOOL，只支持默认格式“NHWC"（API29） 【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)]
11	ANEURALNETWORKS_DEPTHWISE_CONV_2D	深度卷积	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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 TENSOR_QUANT8_ASYMM 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

序号	Operation	含义	边界
			• 输入 8：scalar，INT32，指定 stride 的 height 维度，0 < Stride < 64 • 输入 9：scalar，INT32，指定 depthwise multiplier • 输入 10：scalar，INT32，指定 activation • 输入 11：optional，scalar，BOOL，默认为 false，设置为 true 时表示输入 0 和输出 0 的数据格式为 NCHW • 输入 12：optional，scalar，INT32，指定 width 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 13 • 输入 13：optional，scalar，INT32，指定 height 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 12 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 TENSOR_QUANT8_ASYMM 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 维度，0 < Stride < 64 • 输入 5：scalar，INT32，指定 stride 的 height 维度，0 < Stride < 64 • 输入 6：scalar，INT32，指定 depthwise multiplier • 输入 7：scalar，INT32，指定 activation • 输入 8：optional，scalar，BOOL，默认为 false，设置为 true 时表示输入 0 和输出 0 的数据格式为 NCHW • 输入 9：optional，scalar，INT32，指定 width 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 10

序号	Operation	含义	边界
			• 输入 10：optional，scalar，INT32，指定 height 的膨胀因子，只支持默认值 1，若设置该输入也必须指定输入 9 【约束】 • filterN=inputC=group • StrideW<=(inputW + padW) -((filterW - 1) * dilationW) + 1) • 输入类型为 QUANT8 时，仅支持输入 0 和输出 0 为 NHWC 格式 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth_out];
12	ANEURALNETWORKS_DEQUANTIZE	反量化	【输入】 • 支持的输入 tensor OperandCode： TENSOR_QUANT8_ASYMM • 支持的输出 tensor OperandCode： TENSOR_FLOAT32 TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor 【输出】 输出 0：输出 tensor； shape 与输入 0 相同
13	ANEURALNETWORKS_DIV	除法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 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】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
14	ANEURALNETWORKS_EMBEDDING_L	映射查找	【输入】 • 支持的 tensor OperandCode：

序号	Operation	含义	边界
	OOKUP		TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：from 2 up to 4 • 输入 0：Lookups，1-D tensor，TENSOR_INT32 • 输入 1：Values，n-D tensor，n>=2,从中提取子张量。 【约束】 当数据类型为 quint8 时，输入输出的量化系数在合法范围内(offset<256, scale>0)，且应一致。 【输出】 • 输出 0：n-D tensor，rank 与 shape 与 Values 相同，第一个维度与 Lookups 唯一维度大小相同；类型为 TENSOR_QUANT8_ASYMM 时，scale 与 zeroPoint 必须与 input1 相同
15	ANEURALNETWORKS_EQUAL（V310 新增）	等于	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：OperandCode、维度与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8
16	ANEURALNETWORKS_EXP （V310 新增）	指数运算	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【约束】 无限制 【输出】 输出 0：输出 tensor，shape 同输入 0

序号	Operation	含义	边界
17	ANEURALNETWORKS_FLOOR	向下取整	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：tensor 【约束】 不支持量化 【输出】 输出 0：tensor, OperandCode 和 dimensions 输入 0 相同
18	ANEURALNETWORKS_FULLY_CONNECTED	全连接	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：2,4 • 输入 0：tensor，维度为 2 或 4，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 TENSOR_QUANT8_ASYMM 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 【输出】 输出 0：tensor, shape [batch_size, num_units]；API29 之前，若为 TENSOR_QUANT8_ASYMM 类型，需满足 output_scale > input_scale * filter_scale.
19	ANEURALNETWORKS_GREATER （V310 新增）	大于	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）

序号	Operation	含义	边界
			TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：tensor，OperandCode 与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8
20	ANEURALNETWORKS_GREATER_EQUAL （V310 新增）	大于等于	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16(API29) 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：tensor，OperandCode 与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8
21	ANEURALNETWORKS_HASHTABLE_LOOKUP	查找	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：from 2 up to 4 • 输入 0：Lookups,1-D tensor，类型为 TENSOR_INT32，shape[k] • 输入 1：Keys,1-D tensor，类型为 TENSOR_INT32，shape[n]; Keys 与 Values 构成一个 map，i.e., Keys（Keys [i]）中的第 i 个元素是在 Values（Values [i]）中选择第 i 个子张量的关键,其中 0<=i<=n-1,Keys 必须按照升序排序 • 输入 2：Values，tensor，shape[n,…],i.e.,第一个维度必须是 n 【输出】

序号	Operation	含义	边界
			输出 0：Output，tensor, shape[k…],对于类型为 TENSOR_QUANT8_ASYMM 时，scale 和 zeroPoint 必须与输入 2 相同 输出 1：Hits，boolean tensor，shape[k],表示是否查找到 True/False 非零字节表示 True，即 hits，否则为 0 【约束】 ALIGN(Lookups, 32)*2 + ALIGN(Keys, 32) + ALIGN(Values, 256) * 32 <= 49152
22	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） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：optional，scalar，IINT32，指定 normalization 的维度（API29） 【输出】 输出 0：tensor
23	ANEURALNETWORKS_L2_POOL_2D	L2 池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 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

序号	Operation	含义	边界
			• 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 • 输入 10：scalar，optional ，BOOL ，只支持默认格式“NHWC"（API29） 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 scheme，必须为 PaddingCode 值，为 SAME 或 VALID 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 • 输入 7：scalear，optional，BOOL ，只支持默认格式“NHWC"（API29） 【输出】 输出 0：4-D tensor，shape [batches, out_height, out_width, depth].
24	ANEURALNETWORKS_LESS （V310 新增）	小于	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数:up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：tensor，OperandCode 与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】

序号	Operation	含义	边界
			输出 0：输出 tensor，TENSOR_BOOL8
25	ANEURALNETWORKS_LESS_EQUAL （V310 新增）	小于等于	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16(API29) 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：tensor，OperandCode 与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8
26	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） TENSOR_FLOAT16（API29） 支持的 tensor 维数：only 4 • 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，不能为 0，指定 bias 输入 0 类型为 FLOAT16，bias 类型为 FLOAT16 输入 0 类型为 FLOAT32，bias 类型为 FLOAT32 • 输入 3：scalar，指定 alpha 输入 0 类型为 FLOAT16，alpha 类型为 FLOAT16 输入 0 类型为 FLOAT32，alpha 类型为 FLOAT32 • 输入 4：scalar，指定 beta 输入 0 类型为 FLOAT16，beta 类型为 FLOAT16 输入 0 类型为 FLOAT32，beta 类型为 FLOAT32 • 输入 5：scalar，optional，INT32，指定

序号	Operation	含义	边界
			normalization 的维度（API29） 【约束】 如果 input 有 6 个，则最后一个 INT32 类型的输入是-1 或 input0 的维度减 1（即 3） 【输出】 输出 0：tensor, shape 与输入 0 相同
27	ANEURALNETWORKS_LOG （V310 新增）	对数运算	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：输出 tensor ，shape 同输入 0
28	ANEURALNETWORKS_LOGICAL_AND （V310 新增）	逻辑与	【输入】 • 支持的 tensor OperandCode： TENSOR BOOL8 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor • 输入 1：输入 tensor，shape 同输入 0 【输出】 输出 0：输出 tensor
29	ANEURALNETWORKS_LOGICAL_NOT （V310 新增）	逻辑非	【输入】 • 支持的 tensor OperandCode： TENSOR BOOL8 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor 【输出】 输出 0：输出 tensor
30	ANEURALNETWORKS_LOGICAL_OR （V310 新增）	逻辑或	【输入】 • 支持的 tensor OperandCode： TENSOR BOOL8 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor • 输入 1：输入 tensor，shape 同输入 0

序号	Operation	含义	边界
			【输出】 输出 0：输出 tensor
31	ANEURALNETWORKS_LOGISTIC	LOGISTIC 激活	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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 TENSOR_QUANT8_ASYMM 类型，scale 为 1.f / 256，zeroPoint 为 0
32	ANEURALNETWORKS_LOG_SOFTMAX （V310 新增）	激活函数 softmax 取对数运算	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor • 输入 1：scalar，数据类型 TENSOR_FLOAT16，TENSOR_FLOAT32，指定运算中的系数因子 • 输入 2：scalar, INT32, 指定 reduce 的维度（axis）。 【约束】 • axis 输入范围[-rank,rank) 输入 4 维(NCHW)时可以针对每一维做 softmax： axis=0， 不支持 axis=1 即 channel 的话，c<=11136， h*w < 65536； axis=2 即 Height 场景下，W=1, 0<h<=16384； axis=3 即 Width 场景下，0<W<=16384； 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】

序号	Operation	含义	边界
			输出 0：tensor, OperandCode 与输入 0 相同
33	ANEURALNETWORKS_LSH_PROJECTION	局部敏感哈希（LSH）投影变换	【输入】 支持的 tensor OperandCode TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持 tensor 维度：up to 4 • 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 • 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)（API29），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。 【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] }
34	ANEURALNETWORKS_MAX_POOL_2D	最大池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：4 显式 padding 输入：

序号	Operation	含义	边界
			• 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 • 输入 10：scalar，optional ，BOOL，指定数据格式，只支持默认格式“NHWC"（API29） 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 值，为 SAME 或 VALID 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 • 输入 7：scalar，optional ，BOOL，指定数据格式，只支持默认格式“NHWC"（API29） 【输出】 输出 0：4-D tensor, shape [batches, out_height, out_width, depth] 【约束】 • KernelH<256, kernelW<256;

序号	Operation	含义	边界
			• 当输出 tensor shape H、W 为 1 时，要求 input H * input W < 65536
35	ANEURALNETWORKS_MAXIMUM （V310 新增）	对应元素取最大	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持 tensor 维度：up to 4 • 输入 0：tensor • 输入 1：tensor，OperandCode、维度与输入 0 相同； 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8；对于 QUANT8_ASYMM，scale 和 zeroPoint 可以不同于输入
36	ANEURALNETWORKS_MEAN	平均值	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
37	ANEURALNETWORKS_MINIMUM （V310 新增）	对应元素取最小	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持 tensor 维度：up to 4

序号	Operation	含义	边界
			• 输入 0：tensor • 输入 1：tensor，OperandCode 与输入 0 相同；对于 QUANT8_ASYMM，scale 和 zeroPoint 可以不同于输入 0 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8；对于 QUANT8_ASYMM，scale 和 zeroPoint 可以不同于输入
38	ANEURALNETWORKS_MUL	乘法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同；若为 TENSOR_QUANT8_ASYMM 类型，需满足 output_scale > input1_scale * input2_scale
39	ANEURALNETWORKS_NEG （V310 新增）	取反运算	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：输出 tensor ,shape 同输入 0。
40	ANEURALNETWORKS_NO	不等于	【输入】

序号	Operation	含义	边界
	T_EQUAL （V310 新增）		• 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16(API29) 输入 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1：tensor，OperandCode、维度与输入 0 相同 【约束】 输入 0 与输入 1 的维度相同 输入 0 与输入 1 的各维度上数的乘积需小于 3000 【输出】 输出 0：输出 tensor，TENSOR_BOOL8
41	ANEURALNETWORKS_PAD	填充 0	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 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 【输出】 输出 0：tensor, OperandCode 和维度与输入 0 相同
42	ANEURALNETWORKS_PAD_V2 （V310 新增）	支持指定常数填充	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：only 4 • 输入 0：4-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

序号	Operation	含义	边界
			• 输入 2：scalar 指定 padding 使用的数 输入 0 为 TENSOR_FLOAT32 时，输入 2 类型为 FLOAT32 输入 0 为 TENSOR_FLOAT16 时，输入 2 类型为 FLOAT16 【输出】 输出 0：tensor, OperandCode 和维度与输入 0 相同
43	ANEURALNETWORKS_PRELU （V310 新增）	激活函数 PRELU	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor • 输入 1：输入 tensor，用于指定 alpha 【输出】 输出 0：输出 tensor
44	ANEURALNETWORKS_QUANTIZE （V310 新增）	量化	【输入】 • 支持的输入 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：输入 tensor 【输出】 输出 0：输出 tensor, shape 与输入 0 相同，类型 TENSOR_QUANT8_ASYMM.
45	ANEURALNETWORKS_RELU	激活函数 relu	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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
46	ANEURALNETWORKS_RELU1	激活函数 relu1	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16(API 29) TENSOR_QUANT8_ASYMM； 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
47	ANEURALNETWORKS_RELU6	激活函数 relu6	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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
48	ANEURALNETWORKS_RESHAPE	改变输入维度	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 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
49	ANEURALNETWORKS_RESIZE_BILINEAR	调整图像大小	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：only 4 输入（通过 shape 指定缩放）： • 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 width • 输入 2：scalar，类型为 INT32。指定输出 tensor 的 height • 输入 3： BOOL，指定输出数据格式，只支持默认格式“NHWC"（API29） 输入（通过 scale 指定缩放）（API29）： • 输入 0：4-D tensor，shape [batches, height, width, depth]。指定 input • 输入 1：scalar，类型为 TENSOR_INT32，TENSOR_INT16。指定缩放 width 因子，new_width = floor(width * width_scale) • 输入 2：scalar，类型为 TENSOR_INT32，TENSOR_INT16。指定缩放 height 因子，new_height = floor(height * height_scale) • 输入 3： scale，optional ，BOOL，指定输出数据格式，只支持默认格式“NHWC" 【输出】 输出 0：4-D tensor, shape [batches, new_height, new_width, depth]
50	ANEURALNETWORKS_RSQRT （V310 新增）	平方根倒数	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：输出 tensor ,shape 同输入 0。
51	ANEURALNE	正弦函	【输入】

序号	Operation	含义	边界
	TWORKS_SIN （V310 新增）	数	• 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景) TENSOR_FLOAT16(API29) 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：输出 tensor ,shape 同输入 0。
52	ANEURALNETWORKS_SOFTMAX	归一化逻辑函数	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16(API29) TENSOR_QUANT8_ASYMM 支持的 tensor 维数：2 or 4 • 输入 0：2-D 或 4-D tensor • 输入 1：scalar，指定 beta 的正比例因子；如果 input0 的类型为 TENSOR_FLOAT32 或者 TENSOR_QUANT8_ASYMM，scalar 必须是 TENSOR_FLOAT32；如果为 TENSOR_FLOAT16，scalar 必须是 TENSOR_FLOAT16 • 输入 2：scalar，INT32，指定运算的维度（API29） 【约束】 axis 输入范围[-rank,rank); 输入 4 维(NCHW)时可以针对每一维做 softmax： axis=0，n<=28544； axis=1 即 channel 的话，c<=11136， h*w < 65536； axis=2 即 Height 场景下，W=1, 0<h<=16384； axis=3 即 Width 场景下，0<W<=16384； 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 TENSOR_QUANT8_ASYMM, scale=1.f / 256，zeroPoint = 0.
53	ANEURALNETWORKS_SPACE_TO_BA	用于 N 维张量	【输入】 • 支持的 tensor OperandCode：

序号	Operation	含义	边界
	TCH_ND	的SpaceToBatch	TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：4 • 输入 0：n-D tensor，指定 input，n 范围[1,4] • 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 • 输入 3：BOOL，可选，指定数据格式，只支持默认格式“NHWC"（API29） 【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. 输入 0 的 shape 乘积需小于 500. 【输出】 输出 0：tensor，OperandCode 与输入 0 相同
54	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） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：4 • 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 • 输入 2: BOOL ,optional，指定数据格式，只支持默认格式“NHWC"（API29）

序号	Operation	含义	边界
			【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]
55	ANEURALNETWORKS_SQRT （V310 新增）	平方根	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] 【输出】 输出 0：输出 tensor ,shape 同输入 0。
56	ANEURALNETWORKS_SQUEEZE	压缩	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 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 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
57	ANEURALNETWORKS_STRIDED_SLICE	切片	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，指定 input，n 范围[1,4] • 输入 1：begin，1-D tensor，类型为 TENSOR_INT32，长度为输入 0 的 rank

序号	Operation	含义	边界
			• 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 不为 0 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
58	ANEURALNETWORKS_SUB	减法	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，指定 input，n 范围[1,4] • 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】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
59	ANEURALNETWORKS_TANH	激活函数 tanh	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） 支持的 tensor 维数：up to 4

序号	Operation	含义	边界
			• 输入 0：tensor 【约束】 无限制 【输出】 输出 0：tensor, shape 与输入 0 相同
60	ANEURALNETWORKS_TRANSPOSE	转置	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16（API29） TENSOR_QUANT8_ASYMM 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，n 范围[1,4] • 输入 1（optional）：1-D tensor，TENSOR_INT32，指定输入 tensor 的维度确定转置方式 【约束】 无限制 【输出】 输出 0：tensor, OperandCode 与输入 0 相同
61	ANEURALNETWORKS_SPLIT （V320 新增）	将输入 tensor 按照指定维度进行均匀分割。	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量，TENSOR_INT32，指分割的轴 axis • 输入 2：标量，TENSOR_INT32，分割的数目 num_split 【约束】 无限制 【输出】 输出 0~(num_split-1)：tensor, 类型与输入 0 一致。
62	ANEURALNETWORKS_SLICE （V320 新增）	从指定位置开始从输入	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） ，

序号	Operation	含义	边界
		Tensor 中截取指定大小的切片	TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：1-D tensor，TENSOR_INT32，对应维度的开始切片的索引 • 输入 2：1-D tensor， TENSOR_INT32，切片对应维度的大小 【约束】 不支持 size 为 0 的 tensor。 只支持输入 1 和输入 2 为常量输入。 【输出】 输出 0：n-D tensor, 类型与输入 0 一致。
63	ANEURALNETWORKS_RESIZE_NEAREST_NEIGHBOR （V320 新增）	使用最近邻插值来调整图像大小	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：4-D tensor • 输入 1：标量，类型为 TENSOR_INT32 时表示输出 0 的 width;类型为 TENSOR_FLOAT16 或 TENSOR_FLOAT32 时，表示 width 的缩放系数 new_width = floor(width * width_scale)。 • 输入 2：标量，类型为 TENSOR_INT32 时表示输出 0 的 height;类型为 TENSOR_FLOAT16 或 TENSOR_FLOAT32 时，表示 height 的缩放 0 系数 new_height = floor(height * height_scale) • 输入 3：标量，Bool, true:输入 0 和输出 0 的数据格式为 NCHW，false 为 NHWC。 【约束】 不支持 batch 为 0 的 tensor 【输出】 输出 0：4-D tensor, 类型与输入 0 一致。
64	ANEURALNETWORKS_HEATMAP_MAX_KEYPOINT （V320 新增）	从热图中找出最大值的点，并计算这个点的关键点分数	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：4-D tensor，shape [num_boxes, heatmap_size, heatmap_size, num_keypoints]，heatmaps 的宽、高相同，且大于等于 2

序号	Operation	含义	边界
		以及坐标值	• 输入 1：2-D tensor，[num_boxes, 4]，指定边界框，格式 [x1, y1, x2, y2]. • 输入 2：标量，Bool, true:指定输入 0 的数据格式为 NCHW，false 为 NHWC 【约束】 不支持 NCHW 格式。 【输出】 输出 0：2-D tensor, 类型与输入 0 一致，shape [num_boxes, num_keypoints]，指定 keypoints 的 score。 输出 1：3-D tensor 类型与输入 1 一致，shape [num_boxes, num_keypoints, 2]，指定 keypoints 的位置。
65	ANEURALNETWORKS_GATHER （V320 新增）	按照指定索引和指定轴对输入数据进行数据收集	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量 axis，指定进行数据收集的轴，数据类型为 INT32，范围是 [-n, n) • 输入 2：k 维向量 indices，指定 input0 对应 axis 维度上的索引，该输入仅支持常量输入 【约束】 • 输入 1 的取值范围为[-n,n) • 输入 2 中的元素值需满足 0<=indices 的元素值 <=input0 的 dim[axis]-1 • 输入 2 中 indices 仅支持常量 【输出】 输出 0：(n + k - 1)-D tensor，数据类型与输入 0 一致
66	ANEURALNETWORKS_POW （V320 新增）	对两个输入 tensor 进行幂运算	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，幂运算的底

序号	Operation	含义	边界
			• 输入 1：n-D tensor，幂运算的幂 【约束】 • 该算子不支持广播 • 当底或幂为常量输入时，不支持 TENSOR_FLOAT16 【输出】 输出 0：n-D tensor，数据类型与输入 0 一致
67	ANEURALNETWORKS_TILE （V320 新增）	该算子实现对一个 tensor 在多个维度的平铺操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，用于平铺的向量 • 输入 1：1-D tensor，平铺系数 【约束】 • 平铺系数仅支持常量输入 【输出】 输出 0：n-D tensor，数据类型和维数与输入 0 一致
68	ANEURALNETWORKS_CHANNEL_SHUFFLE （V320 新增）	将输入 tensor 中的某一维按照参数 numGroup 进行分组，组间进行均匀混合	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 ， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，用于分组的向量 • 输入 1：标量，指定分组的组数 • 输入 2：标量，用于指定进行分组的轴 【约束】 输入 2 的取值范围为[-n,n) 【输出】 输出 0：n-D tensor，数据类型和维数与输入 0 一致
69	ANEURALNETWORKS_SELECT （V320 新增）	该算子的功能：将 input0 作为条件，若 input0[i]	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor，数据类型：

序号	Operation	含义	边界
		为 true，则 outpu0[i] 的值为 input1，反之 ouput[0] 为 input2[i]	TENSOR_BOOL8；设置每个元素的值选择是从输入 1（如果为 true）还是从输入 2（如果为 false）中获取输出中的对应元素。 • 输入 1：n-D tensor，与输入 0 的 shape 一致 • 输入 2：n-D tensor，与输入 1 的 shape、type 一致 【约束】 每个维度的维度值不超过 256. 【输出】 输出 0：n-D tensor, 类型和维数与输入 1 和输入 2 一致。
70	ANEURALNETWORKS_TOPK_V2 （V320 新增）	该算子用于获取输入最后一个维度的前 K 个最大值（以降序输出）以及他们的索引	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量，用于表明需要获取的前多少个最大的数。 【约束】 输入 1 的值不超过输入最后一个维度的维度值。 【输出】 输出 0：n-D tensor, 类型与输入 0 一致。 输出 1：n-D tensor，类型为 TENSOR_INT32。
71	ANEURALNETWORKS_EXPAND_DIMS （V320 新增）	对给定的 tensor，增加 1 个维度。	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） ， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量，数据类型：TENSOR_INT32，指定需要扩维的维度。 【约束】 Axis 值不超过输入的维度数, 范围：[-(n + 1), (n + 1)) 【输出】 输出 0：n+1-D tensor, 类型与输入 0 一致。

序号	Operation	含义	边界
72	ANEURALNETWORKS_REDUCE_ALL （V320 新增）	表示在给定的维度上进行 logical_and 操作	【输入】 • 支持的 tensor OperandCode： TENSOR_BOOL8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量，TENSOR_INT32 类型表示在哪个维度上进行 reduce。 • 输入 2：BOOL 类型的标量，如果是 true，则输入 input0 的维度不变，x 维度还是 x 维度，只是将 input1 中轴所表示的那个维度的维度值置为 1。如果是 false，则 input0 由 x 维变为 x-1 维。 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
73	ANEURALNETWORKS_REDUCE_ANY （V320 新增）	表示在给定的维度上进行 logical_or 操作	【输入】 • 支持的 tensor OperandCode： TENSOR_BOOL8 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：标量，TENSOR_INT32 类型表示在哪个维度上进行 reduce。 • Input2：BOOL 类型的标量，如果是 true，则输入 input0 的维度不变，x 维度还是 x 维度，只是将 input1 中轴所表示的那个维度的维度值置为 1。如果是 false，则 input0 由 x 维变为 x-1 维。 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
74	ANEURALNETWORKS_REDUCE_PROD （V320 新增）	表示在给定的维度上进行累乘操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4

序号	Operation	含义	边界
			• 输入 0：n-D tensor • 输入 1：1-D 的 tensor，数据类型：TENSOR_INT32;指定需要进行 reduce 操作的维度，且维度值的范围 [-n, n); • 输入 2：BOOL，keep_dims，若为 true，保持 reduce 操作的维度值为 1，false 不保存 reduce 后的维度. 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
75	ANEURALNETWORKS_REDUCE_MAX （V320 新增）	表示在给定的维度上进行取最大值操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：1-D 的 tensor，数据类型：TENSOR_INT32;指定需要进行 reduce 操作的维度，且维度值的范围 [-n, n); • 输入 2：BOOL，keep_dims，若为 true，保持 reduce 操作的维度值为 1，false 不保存 reduce 后的维度. 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
76	ANEURALNETWORKS_REDUCE_MIN （V320 新增）	表示在给定的维度上进行取最小值操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数：up to 4 • 输入 0：n-D tensor • 输入 1：1-D 的 tensor，数据类型：TENSOR_INT32;指定需要进行 reduce 操作的维度，且维度值的范围 [-n, n); • 输入 2：BOOL，keep_dims，若为 true，保持 reduce 操作的维度值为 1，false 不保存 reduce

序号	Operation	含义	边界
			后的维度. 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
77	ANEURALNETWORKS_REDUCE_SUM （V320 新增）	表示在给定的维度上进行累加操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 支持的 tensor 维数: up to 4 • 输入 0：n-D tensor • 输入 1：1-D 的 tensor，数据类型：TENSOR_INT32;指定需要进行 reduce 操作的维度，且维度值的范围 [-n, n); • 输入 2：BOOL，keep_dims，若为 true，保持 reduce 操作的维度值为 1，false 不保存 reduce 后的维度. 【约束】 不支持广播。 【输出】 输出 0：tensor, 类型与输入 0 一致。
78	ANEURALNETWORKS_RANDOM_MULTINOMIAL （V320 新增）	从多项式分布中抽取样本	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：2-D tensor，指定所有类的未归一化对数概率 • 输入 1：标量，用于指定从每个行切片抽取的独立样本数。 • 输入 2：1-D tensor，dim 值为 2，指定用于初始化随机分布的种子。 【约束】 • 输入 1 需为正整数 • 输入 2 需为常量输入 【输出】 输出 0：2-D tensor，抽取得到的样本

序号	Operation	含义	边界
79	ANEURALNETWORKS_DETECTION_POSTPROCESSING （V320 新增）	NN 网络后处理算子，得到 ROI 并进行排序过滤及 NMS 等操作，得到最终检测结果	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：3-D tensor，score，维度为[batches, num_anchors, num_classes] • 输入 1：3-D tensor，delta，维度为[batches, num_anchors, length_box_encoding] • 输入 2：2-D tensor，anchor，维度为[num_anchors, 4]。 • 输入 3：标量，scaleY，指定 dy 的缩放系数 • 输入 4：标量，scaleX，指定 dx 的缩放系数 • 输入 5：标量，scaleH，指定 dh 的缩放系数 • 输入 6：标量，scaleW，指定 dw 的缩放系数 • 输入 7：标量，指定 nms 算法 • 输入 8：标量，指定最大 BOX 输出个数 • 输入 9：标量，input7 为 false 时有效，指定每个 class 的最大输出检测 anchor 数 • 输入 10：标量，input7 为 true 时有效，指定每个 class 的最大输出检测 anchor 数 • 输入 11：标量，score 的阈值，低于该值会在 NMS 前被过滤 • 输入 12：标量，iou 阈值，在 NMS 时，大于该阈值且 score 较小的 anchor 将会在 NMS 过程中被抑制（过滤） • 输入 13：标量，指定是否包括背景类，当为 true 时，包括背景概率，输出的 class 的 label 从 1 开始，否则从 0 开始计数 【约束】 输入 8、输入 9 需大于 0 且不超过 1024，输入 10 只能为 1； 2 <= num_classes < 1024； 0 < num_anchors < 65536； 输入 11、输入 12 需大于等于 0； scaleX、scaleY、scaleH、scaleW 需大于 1e-5； 【输出】 • 输出 0：2-D tensor，检测到的 score，[batches, max_num_detections] • 输出 1：3-D tensor，检测到的目标的 roi， 维

序号	Operation	含义	边界
			度为[batches, max_num_detections, 4] • 输出 2：2-D tensor，检测到的 class label，[batches, max_num_detections] • 输出 3：1-D tensor，有效检测结果个数，[batches]
80	ANEURALNETWORKS_GROUPED_CONV_2D （V320 新增）	分组卷积	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景），TENSOR_FLOAT16 显式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height, width, depth] • 输入 1：4-D tensor, shape [depth_out, filter_height, filter_width, depth_in]，指定 filter • 输入 2：1-D tensor，shape [depth_out]，指定 bias • 输入 3：scalar，INT32，指定 padding 的左边‘ width’ 维度 • 输入 4：scalar，INT32，指定 padding 的右边‘ width’ 维度 • 输入 5：scalar，INT32，指定 padding 的顶部‘height’ 维度 • 输入 6：scalar，INT32，指定 padding 的底部‘height’ 维度 • 输入 7：scalar，INT32，指定 stride 的‘width’维度 • 输入 8：scalar，INT32，指定 stride 的‘height’ 维度 • 输入 9：scalar，INT32，指定 groups 的数量 • 输入 10：scalar，INT32，指定 activation • 输入 11：scalar, optional ,BOOL，指定数据格式，true 为格式“NCHW”,false 为格式“NHWC" 隐式 padding 输入： • 输入 0：4-D tensor，shape 为 [batches, height, width, depth]，指定 input • 输入 1：4-D tensor, shape [depth_out, filter_height, filter_width, depth_in]，指定 filter • 输入 2：1-D tensor，shape [depth_out]，指定

序号	Operation	含义	边界
			bias • 输入 3：scalar，INT32，指定隐式 padding scheme，必须为 PaddingCode 值，为 SAME 或 VALID 之一 • 输入 4：scalar，INT32，指定 stride 的‘width’ dimension • 输入 5：scalar，INT32，指定 stride 的‘height’ dimension • 输入 6：scalar，INT32，指定 groups 的数量 • 输入 7：scalar，INT32，指定 activation • 输入 8：scalar，optional ,BOOL，指定数据格式，true 为格式“NCHW”,false 为格式“NHWC" 【约束】 inputC=group*filterC；filterN%group=0 【输出】 输出 0：4-D 输出 tensor，shape [batches, out_height, out_width, depth_out]。
81	ANEURALNETWORKS_TRANSPOSE_CONV_2D （V320 新增）	反卷积	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 显式 padding 输入： • 输入 0：4-D tensor，shape 为[batches, height, width, depth] • 输入 1：4-D tensor, shape [depth_out, filter_height, filter_width, depth_in]，指定 filter • 输入 2：1-D tensor，shape [depth_out]，指定 bias • 输入 3：scalar，INT32，指定 padding 的左边‘ width’ 维度 • 输入 4：scalar，INT32，指定 padding 的右边‘ width’ 维度 • 输入 5：scalar，INT32，指定 padding 的顶部‘height’ 维度 • 输入 6：scalar，INT32，指定 padding 的底部‘height’ 维度 • 输入 7：scalar，INT32，指定 stride 的‘width’ 维度

序号	Operation	含义	边界

			• 输入 8：scalar，INT32，指定 stride 的‘height’ 维度
			• 输入 9：scalar，INT32，指定 activation
			• 输入 10：scalar, optional ,BOOL，指定数据格式，true 为格式“NCHW”,false 为格式“NHWC"
			隐式 padding 输入：
			• 输入 0：4-D tensor，shape 为 [batches, height, width, depth]，指定 input
			• 输入 1：4-D tensor, shape [depth_out, filter_height, filter_width, depth_in]，指定 filter
			• 输入 2：1-D tensor，shape [depth_out]，指定 bias
			• 输入 3：tensor，INT32，指定输出 tensor 的 shape
			• 输入 4：scalar，INT32，指定隐式 padding scheme，必须为 PaddingCode 值，为 SAME 或 VALID 之一
			• 输入 5：scalar，INT32，指定 stride 的‘width’ dimension
			• 输入 6：scalar，INT32，指定 stride 的‘height’ dimension
			• 输入 7：scalar，INT32，指定 activation
			• 输入 8：scalar，optional ,BOOL，指定数据格式，true 为格式“NCHW”,false 为格式“NHWC"
			【约束】
			• 输入 1,2 只支持常量输入。
			• (inputH - 1) * strideH + 1 + aH <= 4000;
			(inputW - 1) * strideW + 1 + aW<= 4000;
			group ==1; dilationH == dilationW == 1;
			filterH <= 15 && filterW <= 15;
			filterH - padHHead - 1 >= 0;
			filterW - padWHead - 1 >= 0;
			其中：
			aH = (inputH + padHHead + padHTail - filterH) % strideH
			aW = (inputW + padHHead + padHTail - filterW) % strideW
			• (inputH - 1) * strideH - padHHead - padHTail <= outputH <= inputH * strideH - padHHead –

序号	Operation	含义	边界
			padHTail；(inputW - 1) * strideW - padWHead - padWTail <= outputW <= inputW * strideW - padWHead - padWTail 【输出】 输出 0：4-D 输出 tensor
82	ANEURALNETWORKS_LSTM （V320 新增）	LSTM 单元	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：2-D tensor,非常量输入，shape 是[batch_size,input_size] • 输入 1: 2-D tensor,input-to-input weights 可选输入，shape 为[num_units,input_size] • 输入 2: 2-D tensor,input-to-forgot weights，shape 为[num_units,input_size] • 输入 3: 2-D tensor,input-to-cell weights，shape 为[num_units,input_size] • 输入 4: 2-D tensor,input-to-output weights，shape 为[num_units,input_size] • 输入 5: 2-D tensor,recurrent-to-input weights 可选输入，shape 为[num_units,output_size] • 输入 6: 2-D tensor,recurrent-to-forgot weights，shape 为[num_units,output_size] • 输入 7: 2-D tensor,recurrent-to-cell weights，shape 为[num_units,output_size] • 输入 8: 2-D tensor,recurrent-to-output weights，shape 为[num_units,output_size] • 输入 9: 1-D tensor,cell-to-input weights 可选输入，shape 为[num_units] • 输入 10: 1-D tensor,cell-to-forgot weights 可选输入，shape 为[num_units] • 输入 11: 1-D tensor,cell-to-output weights 可选输入，shape 为[num_units] • 输入 12: 1-D tensor,input gate bias 可选输入，shape 为[num_units] • 输入 13: 1-D tensor,forget gate bias，shape 为[num_units] • 输入 14: 1-D tensor,cell gate bias，shape 为[num_units] • 输入 15: 1-D tensor,output gate bias，shape 为

序号	Operation	含义	边界
			[num_units] • 输入 16: 2-D tensor,project weights 可选输入，shape 为[output_size,num_units] • 输入 17: 1-D tensor,projcet bias 可选输入，shape 为[output_size] • 输入 18: 2-D tensor,output state(in)，shape 为[batch_size,output_size] • 输入 19: 2-D tensor,cell state(in)，shape 为[batch_size,num_units] • 输入 20: 标量，标识激活函数, 支持 1:Relu, 2:Relu6, 3:Tanh,4:Sigmoid • 输入 21: 标量，cell_clip 门限，取值范围[-cell_clip,cell_clip],设置值为 0 时，忽略 cell_clip 门限。 • 输入 22: 标量，proj_clipp 门限，取值范围[-proj_clip,proj_clip],设置值为 0 时，忽略 proj_clip 门限。 • 输入 23: 1-D tensor,input layer normalization weights,shape 是[num_units] • 输入 24: 1-D tensor,forget layer normalization weights,shape 是[num_units] • 输入 25: 1-D tensor,cell layer normalization weights,shape 是[num_units] • 输入 26: 1-D tensor,output layer normalization weights,shape 是[num_units] 【约束】 • 不支持输入 23-26 layer normalization 参数参与计算。 • 输入 1 到输入 17 仅支持常量输入 • 不支持输出中间值状态，即不支持输出 scratch buffer。 【输出】 • 输出 0：2-D 输出 tensor，使能 CIFG 时 shape 是[batch_size,num_units*3]。不使能 CIFG 时，shape 是[batch_size,num_units*4] • 输出 1：2-D 输出 tensor，output state(out) shape 是[batche_size,output_size] • 输出 2：2-D 输出 tensor, cell state(out) shape 是[batche_size, num_units] • 输出 3：2-D 输出 tensor, shape 是

序号	Operation	含义	边界
			[batch_size,output_size]。
83	ANEURALNETWORKS_UNIDIRECTIONAL_SEQUENCES_LSTM （V320 新增）	单向 LSTM 单元	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：3-D tensor,非常量输入，time-major 时 shape 是[max_time,batch_size,input_size]，batch-major 时 shape 是 [batch_size,max_time,input_size] • 输入 1: 2-D tensor,input-to-input weights 可选输入，shape 为[num_units,input_size] • 输入 2: 2-D tensor,input-to-forgot weights，shape 为[num_units,input_size] • 输入 3: 2-D tensor,input-to-cell weights，shape 为 [num_units,input_size] • 输入 4: 2-D tensor,input-to-output weights，shape 为[num_units,input_size] • 输入 5: 2-D tensor,recurrent-to-input weights 可选输入，shape 为[num_units,output_size] • 输入 6: 2-D tensor,recurrent-to-forgot weights，shape 为[num_units,output_size] • 输入 7: 2-D tensor,recurrent-to-cell weights，shape 为[num_units,output_size] • 输入 8: 2-D tensor,recurrent-to-output weights，shape 为[num_units,output_size] • 输入 9: 1-D tensor,cell-to-input weights 可选输入，shape 为[num_units] • 输入 10: 1-D tensor,cell-to-forgot weights 可选输入，shape 为[num_units] • 输入 11: 1-D tensor,cell-to-output weights 可选输入，shape 为[num_units] • 输入 12: 1-D tensor,input gate bias 可选输入，shape 为[num_units] • 输入 13: 1-D tensor,forget gate bias，shape 为 [num_units] • 输入 14: 1-D tensor,cell gate bias，shape 为 [num_units] • 输入 15: 1-D tensor,output gate bias，shape 为 [num_units] • 输入 16: 2-D tensor,project weights 可选输入，

序号	Operation	含义	边界
			shape 为[output_size,num_units] • 输入 17: 1-D tensor,projcet bias 可选输入，shape 为[output_size] • 输入 18: 2-D tensor,output state(in)，shape 为[batch_size,output_size] • 输入 19: 2-D tensor,cell state(in)，shape 为[batch_size,num_units] • 输入 20: 标量，标识激活函数, 支持 1:Relu, 2:Relu6, 3:Tanh,4:Sigmoid • 输入 21: 标量，cell_clip 门限，取值范围[-cell_clip,cell_clip],设置值为 0 时，忽略 cell_clip 门限。 • 输入 22: 标量，proj_clipp 门限，取值范围[-proj_clip,proj_clip],设置值为 0 时，忽略 proj_clip 门限。 • 输入 23: 标量，true:time_major false:batch_major。 • 输入 24: 1-D tensor,input layer normalization weights,shape 是[num_units] • 输入 25: 1-D tensor,forget layer normalization weights,shape 是[num_units] • 输入 26: 1-D tensor,cell layer normalization weights,shape 是[num_units] • 输入 27: 1-D tensor,output layer normalization weights,shape 是[num_units] 【约束】 • 不支持输入 24-27 layer normalization 参数参与计算。 • 输入 1 到输入 17 仅支持常量输入 • 不支持输出中间值状态，即不支持输出 scratch buffer。 【输出】 输出 0：3-D 输出 tensor time-major 时 shape 是[max_time,batch_size,output_size]，batch-major 时 shape 是[batch_size,max_time,output_size]
84	ANEURALNETWORKS_ROI_POOLING （V320 新增）	对选取输入的特征图进行最大池化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16

序号	Operation	含义	边界
			• 输入 0：4-D tensor feature map. • 输入 1：2-D tensor,shape 是[nums_rois,4] • 输入 2: 1-D tensor,shape 是[nums_rois] • 输入 3: 标量，最大池化的 H 维度 • 输入 4：标量，最大池化的 W 维度 • 输入 5：标量，原始图像的高度到特征图的高度的比率 • 输入 6：标量，原始图像的宽度到特征图的宽度的比率 • 输入 7：标量，标识输入 0 和输出 0 数据排布。true 为 NCHW，false 为 NHWC 【约束】 • 输入 1 中不支持空区域计算，且需为常量 • 输入 2 需为常量 【输出】 输出 0：4-D 输出 tensor。
85	ANEURALNETWORKS_SVDF （V320 新增）	通过对每个节点进行奇异值分解来近似处理一系列的密集连接层	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：2-D tensor,shape 为 [batch_size,input_size] • 输入 1: 2-D tensor，shape 为 [num_units,input_size] • 输入 2：2-D tensor,shape 为 [num_units,memory_size] • 输入 3: 1-D tensor，可选输入，shape 为 [num_units] • 输入 4: 2-D tensor，shape 为 [batch_size,memory_size*num_units*rank] • 输入 5：标量，rank 值 • 输入 6：标量，激活函数，不支持 None 【约束】 输入 1 到输入 4 仅支持常量输入。 【输出】 • 输出 0：2-D 输出 tensor,shape 是 [batch_size,memory_size*num_units*rank] • 输出 1：2-D 输出 tensor，shape 是

序号	Operation	含义	边界
			[batch_size,num_units]
86	ANEURALNETWORKS_INSTANCE_NORMALIZATION （V320 新增）	对输入 tensor 进行 instance normalization 归一化	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：N-D tensor,非常量输入 • 输入 1：标量，gamma 标识缩放 • 输入 2：标量，beta 标识偏移 • 输入 3：标量，epsilon 防止除 0 【约束】 无 【输出】 输出 0：N-D 输出 tensor。
87	ANEURALNETWORKS_BIDIRECTIONAL_SEQUENCE_LSTM （V320 新增）	双向 LSTM 算子	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：3-D tensor，维度为[max_time, batch_size, input_size]或[batch_size, max_time, input_size]。 • 输入 1~4：2-D tensor，前向 input-to-input、input-to-forget、input-to-cell、input-to-output 权重。大小为[fw_num_units, input_size]，input-to-input 输入为可选输入 • 输入 5~8：2-D tensor，前向 recurrent-to-input、recurrent-to-forget、recurrent-to-cell、recurrent-to-output 权重。大小为[fw_num_units, fw_output_size]。recurrent-to-input 输入为可选输入 • 输入 9~11：1-D tensor，前向 cell-to-input、cell-to-forget、cell-to-output 权重。大小为[fw_num_units, input_size]。这 3 个输入均为可选输入 • 输入 12~15：1-D tensor，前向 input、forget、cell、output 偏置。大小为[fw_num_units]。 • 输入 16：2-D tensor，前向 投影权重。大小为[fw_output_size, fw_num_units]。该输入为可选输入 • 输入 17：1-D tensor，前向 投影偏置。大小为

序号	Operation	含义	边界
			[fw_output_size]。该输入为可选输入 • 输入 18~21：2-D tensor，后向 input-to-input、input-to-forget、input-to-cell、input-to-output 权重。大小为[bw_num_units, input_size]，input-to-input 输入为可选输入 • 输入 22~25： 2-D tensor，后向 recurrent-to-input、recurrent-to-forget、recurrent-to-cell、recurrent-to-output 权重。大小为[bw_num_units, bw_output_size]。recurrent-to-input 输入为可选输入 • 输入 26~28：1-D 输入 tensor，前向 cell-to-input、 cell-to-forget、cell-to-output 权重。大小为[bw_num_units]。这 3 个输入均为可选输入 • 输入 29~32： 1-D tensor，前向 input、forget、cell、output 偏置。大小为[bw_num_units]。 • 输入 33： 2-D tensor，前向 投影权重。大小为[bw_output_size, bw_num_units]。该输入为可选输入 • 输入 34： 1-D tensor，前向 投影偏置。大小为[bw_output_size]，可选输入 • 输入 35： 2-D tensor，前向输入的激活状态。大小为 [batch_size, bw_output_size] • 输入 36： 2-D tensor，前向输入的 cell 状态。大小为 [batch_size, bw_num_units] • 输入 37 ：2-D tensor，后向输入的激活状态。大小为 [batch_size, bw_output_size] • 输入 38：2-D tensor，后向输入的 cell 状态。大小为 [batch_size, bw_num_units] • 输入 39： 3-D tensor，辅助输入。大小为[max_time, batch_size, input_size]。可选 • 输入 40~43： 2-D tensor，前向辅助 input-to-input、input-to-forget、input-to-cell、input-to-output 权重。大小为[fw_num_units, input_size]，均为可选输入 • 输入 44~47：2-D tensor，后向辅助 input-to-input、input-to-forget、input-to-cell、input-to-output 权重。大小为[bw_num_units, input_size]，均为可选输入 • 输入 48：激活函数，支持 1:Relu, 2:Relu6, 3:Tanh, 4:Sigmoid • 输入 49~50： 限幅阈值。 • 输入 51：标量。指定是否应合并前向和后向单

序号	Operation	含义	边界
			元的输出。 • 输入 52：标量。指定输入和输出张量的形状格式 • 输入 53~56：1-D tensor，前向 input、forget、cell、output 层的归一化权重。可选 • 输入 57~60：1-D tensor，后向 input、forget、cell、output 层的归一化权重。可选 【约束】 • 除输入 0、输入 35~38 外，其他输入仅支持常量输入 • 不支持辅助输入（39~47），不支持归一化权重（53~60） • 不支持 merge_output（即 Concat，输入 51 需设置为 false） 【输出】 • 输出 0：3-D tensor，前向 LSTM 计算结果 • 输出 1：3-D tensor，后向 LSTM 计算结果
88	ANEURALNETWORKS_UNIDIRECTIONAL_SEQUENCE_RNN （V320 新增）	单向 RNN	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16 输入 tensor 数据类型必须相同 • 输入 0：3-D tensor , 如果 input 6（timeMajor）参数设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1, 输入输出 shape： [maxTime, batchSize, numUnits]. • 输入 1：weights，2-D tensor，shape [num_units, input_size],只支持常量输入 • 输入 2：recurrent_weights， 2-D tensor ， shape [num_units, num_units], 只支持常量输入 • 输入 3：bias，1-D tensor ， shape [num_units]，只支持常量输入 • 输入 4: hidden state (in). 2-D tensor , shape [batch_size, num_units]. • 输入 5: fused_activation_function. optional FuseCode value , 不支持“NONE”. • 输入 6：timeMajor，只能取值为 0 或者 1. 设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1, 输入输出 shape：

序号	Operation	含义	边界
			[maxTime, batchSize, numUnits]. 【输出】 输出 0: 3-D tensor，如果 input 6（timeMajor）timeMajor 参数设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1，输入输出 shape： [maxTime, batchSize, numUnits].
89	ANEURALNETWORKS_BIDIRECTIONAL_SEQUENCE_RNN （V320 新增）	双向 RNN	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16 输入 tensor 数据类型必须相同 • 输入 0：3-D tensor，如果 input 6（timeMajor）参数设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1，输入输出 shape： [maxTime, batchSize, numUnits]. • 输入 1： fwWeights. A 2-D tensor of shape [fwNumUnits, inputSize],只支持常量输入. • 输入 2：fwRecurrentWeights. A 2-D tensor of shape [fwNumUnits, fwNumUnits],只支持常量输入. • 输入 3：fwBias. A 1-D tensor of shape [fwNumUnits],只支持常量输入. • 输入 4: fwHiddenState. A 2-D tensor of shape [batchSize, fwNumUnits]. Specifies a hidden state input for the first time step of the computation. • 输入 5: bwWeights. A 2-D tensor of shape [bwNumUnits, inputSize]，只支持常量输入. • 输入 6：bwRecurrentWeights. A 2-D tensor of shape [bwNumUnits, bwNumUnits]，只支持常量输入. • 输入 7：bwBias. A 1-D tensor of shape [bwNumUnits]. • 输入 8：bwHiddenState A 2-D tensor of shape [batchSize, bwNumUnits]. Specifies a hidden state input for the first time step of the computation. • 输入 9：auxInput. A 3-D tensor. The shape is the same as of the input 0,不支持该参数. • 输入 10：fwAuxWeights. A 2-D tensor of shape [fwNumUnits, inputSize],不支持该参数. • 输入 11：bwAuxWeights. A 2-D tensor of shape [bwNumUnits, inputSize],不支持该参数.

序号	Operation	含义	边界
			• 输入 12：fused_activation_function. optional FuseCode value , 不支持 “NONE”. • 输入 13：timeMajor，只能取值为 0 或者 1. 设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1，输入输出 shape：[maxTime, batchSize, numUnits]. • 输入 14: mergeOutputs.BOOL 指定 fwOutput 和 bwOutput 的输出是否分开,如果设置为 0 表示分开输出；如果设置为 1 表示合并输出; 【约束】： mergeOutputs 只支持 false 【输出】 • 输出 0: fwOutput. A 3-D tensor. 如果 input 6（timeMajor）timeMajor 参数设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1，输入输出 shape： [maxTime, batchSize, numUnits]. • 输出 1: bwOutput. A 3-D tensor. 如果 input 6（timeMajor）timeMajor 参数设置为 0，输入输出 shape ：[batchSize, maxTime, numUnits],设置为 1，输入输出 shape： [maxTime, batchSize, numUnits].
90	ANEURALNETWORKS_RNN （V320 新增）	RNN 网络	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16 输入 tensor 数据类型必须相同 • 输入 0：input ，2-D tensor，shape [batch_size, input_size] • 输入 1：weights，2-D tensor，shape [num_units, input_size]，只支持常量输入 • 输入 2：recurrent_weights， 2-D tensor ，shape [num_units, num_units]，只支持常量输入 • 输入 3：bias，1-D tensor ，shape [num_units]，只支持常量输入 • 输入 4: hidden state (in). 2-D tensor , shape [batch_size, num_units]. • 输入 5: fused_activation_function. optional FuseCode value,不支持“NONE” 【输出】 输出 0: hidden state (out). 2-D tensor , shape

序号	Operation	含义	边界
			[batch_size, num_units]. 输出 1: output. 2-D tensor , shape [batch_size, num_units].
91	ANEURALNETWORKS_ROI_ALIGN （V320 新增）	通过平均合并来自双线性插值的采样点，选择每个感兴趣区域的特征图并将其缩放到统一的输出大小	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：4-D tensor feature map. • 输入 1：2-D tensor,shape 是[nums_rois,4] • 输入 2: 1-D tensor,每个框的 batch 索引，shape 是[nums_rois] • 输入 3: 标量，平均池化的 H 维度 • 输入 4：标量，平均池化的 W 维度 • 输入 5：标量，原始图像的高度到特征图的高度的比率 • 输入 6：标量，原始图像的宽度到特征图的宽度的比率 • 输入 7：标量，H 方向上的采样点数。 • 输入 8：标量，W 方向上的采样点数 • 输入 9：标量，标识输入 0 和输出 0 数据排布。true 为 NCHW，false 为 NHWC 【约束】 • 输入 1 中不支持空区域计算，且需为常量 • 输入 2 需为常量 【输出】 输出 0：4-D 输出 tensor。
92	ANEURALNETWORKS_GENERATE_PROPOSALS （V320 新增）	GENERATE_PROPOSALS	【输入】 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景） TENSOR_FLOAT16 • 输入 0: 4-D Tensor shape 是[batches, num_anchors, height, width]. • 输入 1: 4-D Tensor 标识边界框，shape 是[batches, num_anchors * 4, height, width]. • 输入 2: 2-D Tensor，shape 是[num_anchors, 4], • 输入 3: 2-D Tensor，标识每个 batch 的 size，shape 是[batches, 2] • 输入 4: scalar，float32，指定原始图像到特征图

序号	Operation	含义	边界
			的高度比 • 输入 5: scalar，float32，指定原始图像到特征图的宽度比 • 输入 6: scalar，int32，指定用于 NMS 处理的最大输入 Box 数量，设置非正数指定无上限 • 输入 7: scalar，int32，指定返回 NMS 处理的最大输入 Box 数量，设置非正数指定无上限 • 输入 8: scalar，float32，指定 NMS 处理参数 IoU（交并比） • 输入 9: scalar，float32，指定 Boxes 的最小尺寸 • 输入 10: BOOL ，表示输入 0 和输入 1 的格式，true 表示 NCHW，false 表示 NHWC 【约束】 • 仅支持 NHWC 【输出】 • 输出 1: 1-D of shape [num_output_rois], 标识每个输出框的得分 • 输出 2: 2-D tensor.标识每个分类的每个框的坐标 shape 是 [num_output_rois, 4]. • 输出 3: 1-D of shape [num_output_rois], 每个框的索引,数据类型 int32
93	ANEURALNETWORKS_AXIS_ALIGNED_BBOX_TRANSFORM （V320 新增）	该算子的功能即对坐标按照一定的规则进行对齐变换，只需要对坐标进行操作	【输入】 • 支持的 tensor OperandCode： TENSOR_FLOAT32（仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：2-D 输入 tensor，维度为[num_rois, 4]。 • 输入 1：2-D 输入 tensor，维度为[num_rois, num_classes * 4] • 输入 2：1-D 输入 tensor，维度为[num_rois]，该输入格式为 NCHW • 输入 3：2-D 输入 tensor，维度为[batches, 2] 【约束】 无 【输出】 输出 0：2-D tensor，维度为[num_rois, num_classes * 4]
94	ANEURALNETWORKS_BO	基于贪	【输入】

序号	Operation	含义	边界
	X_WITH_NMS_LIMIT （V320 新增）	婪策略，按分数降序选择边界框的子集。	• 支持的 tensor OperandCode： TENSOR_FLOAT32 （仅支持 relaxed 场景）， TENSOR_FLOAT16 • 输入 0：2-D 输入 tensor，维度为[num_rois, num_classes]。 • 输入 1：2-D 输入 tensor，维度为[num_rois, num_classes * 4] • 输入 2：1-D 输入 tensor，维度为[num_rois] • 输入 3：标量，score_threshold • 输入 4：标量，每张图片最多选择的框个数 • 输入 5：标量，NMS 方式 • 输入 6：标量，IOU 门限 • 输入 7：标量，当 NMS 方式为高斯时，该值表示高斯核的 SIGMA 值。在本次迭代中该值不生效。 • 输入 8：标量，nms_score_threshold 【约束】 NMS 方式仅支持 HARD 模式 【输出】 • 输出 0：1-D 输出 tensor，维度为[num_output_rois] • 输出 1：2-D 输出 tensor，维度为[num_output_rois， 4] • 输出 2：1-D 输出 tensor，维度为[num_output_rois] • 输出 3：1-D 输出 tensor，维度为[num_output_rois]
95	ANEURALNETWORKS_QUANTIZED_16BIT_LSTM （V320 新增）	使用 16bit 量化内部状态的 LSTM	【输入】 • 输入 0：2-D tensor，shape 是[batch_size, input_size]。 • 输入 1：input-to-input weights，2-D tensor，shape 是[nums_units,input_size],nums_units 对应单元位的数量，只支持常量。 • 输入 2：2-D tensor， input-to-forget weights，shape 是[nums_units,input_size]，只支持常量。 • 输入 3：2-D tensor，input-to-cell weights，shape 是[nums_units,input_size] ，只支持常量。 • 输入 4：2-D tensor，input-to-output weights，shape 是[nums_units,input_size] ，只支持常量。

序号	Operation	含义	边界
			• 输入 5：2-D tensor，recurrent-to-input weights，shape 是[nums_units,output_size],output_size 对应于单元位的数量（num_units）;或者如果定义了 projection_weights，对应它的第 2 个维度，只支持常量。 • 输入 6：2-D tensor，recurrent-to-forget weights ，shape 是[nums_units,output_size] ，只支持常量。 • 输入 7：2-D tensor，recurrent-to-cell weights ，shape 是[nums_units,output_size]，只支持常量。 • 输入 8：2-D tensor，recurrent-to-output weights ，shape 是[nums_units,output_size] ，只支持常量。 • 输入 9：1-D tensor input gate bias，shape 是[num_units] ，只支持常量。 • 输入 10：1-D tensor forget gate bias，shape 是[num_units] ，只支持常量。 • 输入 11：1-D tensor cell bias，shape 是[num_units] ，只支持常量。 • 输入 12：1-D tensor output gate bias，shape 是[num_units] ，只支持常量。 • 输入 13：2-D tensor, pre_cell_state，前一个单元的 cell 状态, shape 为[numBatches, outputSize] • 输入 14：2-D tensor， pre output state, shape 为[numBatches, outputSize] 【约束】 • Input1~Input12 需为常量 • Input 13 需为 UINT16 格式，其他输入为 UINT8 格式，对应 Output0 为 UINT16 格式，Output1 为 UINT8 格式。 【输出】 • 输出 0：2-D tensor, cell 状态, shape 为[numBatches, outputSize]; • 输出 1：2-D tensor， 输出值, shape 为[numBatches, outputSize]

3 CPU 算子列表

序号	算子	含义
1	Convolution	卷积
2	Scale	out=alpha*Input+beta
3	Relu	激活函数
4	Pooling	池化层
5	Eltwise	按元素操作层(求和、乘积、最大值)
6	FullConnection	全连接
7	Softmax	归一化逻辑函数
8	Deconvolution	反卷积
9	Crop	截取
10	Concat	数据按维度拼接
11	Reshape	改变输入维度
12	Sigmoid	激活函数
13	Power	计算 y = （scale*x+shift）^ power
14	Argmax	返回输入的最大值对应的索引序号
15	Interp	插值层
16	LeakyRelu	激活函数
17	ConvolutionDepthwise	深度卷积

