



HiAI DDK V320

Operator Specifications

Issue **04**
Date **2020-02-18**

Copyright © Huawei Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei HiAI Application

Send an application email to developer@huawei.com.

Email subject: HUAWEI HiAI + Company name + Product name

Email body: Cooperation company + Contact person + Phone number + Email address

We will reply to you within 5 working days.

Official website: <https://developer.huawei.com/consumer/en/>

About This Document

Purpose

This document describes the operator specifications supported by Huawei HiAI DDK V320.

This document is used in conjunction with the following documents:

- Huawei HiAI DDK V320 Quick Start
- Huawei HiAI DDK V320 Model Inference and Integration Guide

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Date	Version	Change Description
2020-02-18	04	Added restrictions for operator stridedSlice. Added the description of CPU operators.
2019-12-31	03	Added the description of HiAI DDK V320.
2019-11-15	02	Added restrictions for operator AvgPool.
2019-09-06	01	Added the description of HiAI DDK V310.

Contents

About This Document i

1 Parameter Description 1

2 NPU Operator Restrictions 3

2.1 General Restrictions 3

2.2 Caffe Operator Boundaries 3

2.3 TensorFlow Operator Boundaries 22

2.4 AndroidNN Operator Boundaries 69

3 CPU Operator List 126

1

Parameter Description

Parameter	Description
ni	Batch size
ci/co	Channel count
hi/ho/	Height
wi/wo	Width
sh/sw	Stride
kh/kw	Size of the convolution filter
window_h(window_y)/ window_w(window_x)	Window size
dh(dilation_h)/ dw(dilation_w)	Convolution dilation coefficient
FilterHDilation/ FilterWDilation	H/W dimension of the dilated filter
FilterH/FilterW	H/W dimension of the convolution weight
padWHead/padHHead	Pad head of the H/W dimension
PadWTail/padHTail	Pad tail of the H/W dimension
dilationsize	User-defined dilation coefficient
FilterSize	User-defined filter count
INT32_MAX	Maximum value that can be represented by data type int32

Parameter	Description
ALIGN(X, N)	X is mapped to the nearest multiple of N For examples: ALIGN(1, 16)=16, ALIGN(16, 16)=16, ALIGN(17, 16)=32

2 NPU Operator Restrictions

2.1 General Restrictions

In NCHW and NHWC scenarios, $1 \leq N \leq 65535$.

For the Caffe framework, if the **axis** parameter is available, the input tensor rank must not be 1. When the input tensor rank is 2 or 3, **axis** must not be negative.

The HiAI DDK supports Caffe 1.0, TensorFlow 1.12, and AndroidNN API level 29.

2.2 Caffe Operator Boundaries

No	Operator	Description	Boundary
1	Absval	Computes the absolute value of the input.	[Inputs] One input [Arguments] engine : (optional) enum, default to 0 , CAFFE = 1, CUDNN = 2
2	Argmax	Computes the index of the maximum values.	[Inputs] One input [Arguments] • out_max_val : (optional) bool, default to false • top_k : (optional) unit32, default to 1 • axis : (optional) int32
3	BatchNorm	Normalizes the input: variance of $[(x - \text{avg}(x))/x]$	[Inputs] One input [Arguments] • use_global_stats : bool, must be true

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • moving_average_fraction: (optional) float, default to 0.999 • eps: (optional) float, default to 1e - 5 <p>[Restrictions]</p> <ul style="list-style-type: none"> • Only C dimension can be normalized. • Shape of scale, beta, mean, and gamma: [c]
4	ChannelA xpy (available since V310)	ChannelAxy (Squeeze-and-Excitation Networks)	<p>[Inputs]</p> <p>Three inputs</p> <ul style="list-style-type: none"> • Input 1: Tensor, with shape [N, C, 1, 1] • Input 2: Tensor, with the same C dimension as input 1 • Input 3: Tensor, with the identical shape as input 2
5	Concat	Concatenates the input along the given dimension.	<p>[Inputs]</p> <p>Multiple inputs</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • concat_dim: uint32, default to 1, greater than 0 (optional) • axis: (optional) int32, default to 1, exclusive with concat_dim. When axis is -1, four input dimensions are required. Otherwise, the result may be incorrect. <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of dimensions of the input tensors must match, and all dimensions except axis must be equal. • The range of the input Tensor count is [1, 32].
6	Convolution	Convolve the input.	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: (optional) uint32 • bias_term: (optional) bool, default to true • pad: uint32, default to 0, array • kernel_size: uint32, array • stride: uint32, default to 1, array • dilation: uint32, default to 1, array • pad_h: (optional) uint32, default to 0 (2D only) • pad_w: (optional) uint32, default to 0 (2D only) • kernel_h: (optional) uint32 (2D only) • kernel_w: (optional) uint32 (2D only) • stride_h: (optional) uint32 (2D only)

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • stride_w: (optional) uint32 (2D only) • group: (optional) uint32, default to 1 • weight_filler: (optional) FillerParameter • bias_filler: (optional) FillerParameter • engine: (optional) enum, default to 0, CAFE = 1, CUDNN = 2 • force_nd_im2col: (optional) bool, default to false • axis: (optional) int32, default to 1 <p>[Restrictions]</p> <ul style="list-style-type: none"> • filter must be 4D constants. • $(inputW + padWHead + padWTail) \geq (((FilterW - 1) * dilationW) + 1)$ • $(inputW + padWHead + padWTail)/StrideW + 1 \leq 2147483647$ • $(inputH + padHHead + padHTail) \geq (((FilterH - 1) * dilationH) + 1)$ • $(inputH + padHHead + padHTail)/StrideH + 1 \leq 2147483647$ • $0 \leq Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 \leq dilationsize < 256$ • $StrideW \leq (inputW + padW) - ((filterW - 1) * dilationW) + 1$ • The shape of bias must be (1, co, 1, 1), with the same co as num_output.
7	Crop	Crops the input.	<p>[Inputs] Two inputs</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • axis: (optional) int32, default to 2. When axis is -1, four input dimensions are required. • offset: uint32, array
8	Correlation (available since V310)	Correlation	<p>[Inputs] Two inputs, with the same C dimension</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: uint32, with the same N dimension as input 2 • kernel_size: uint32, with the same H and W dimensions as input 2 • stride: (optional) uint32, default to 1 • group: (optional) uint32, default to and must be 1 • pad: (optional) uint32, default to 0 • dilation: (optional) uint32, default to 1

No	Operator	Description	Boundary
.			<p>[Restrictions]</p> <ul style="list-style-type: none"> • $N \leq 3840$ • Other constraints are the same as those of convolution.
9	Deconvolution	Deconvolution	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: (optional) uint32 • bias_term: (optional) bool, default to true • pad: uint32, default to 0, array • kernel_size: uint32, array • stride: uint32, default to 1, array • dilation: uint32, default to 1, array • pad_h: (optional) uint32, default to 0 (2D only) • pad_w: (optional) uint32, default to 0 (2D only) • kernel_h: (optional) uint32 (2D only) • kernel_w: (optional) uint32 (2D only) • stride_h: (optional) uint32 (2D only) • stride_w: (optional) uint32 (2D only) • group: uint32, default to 1 • weight_filler: (optional) FillerParameter • bias_filler: (optional) FillerParameter • engine: (optional) enum, default to 0, Caffe = 1, CUDNN = 2 • force_nd_im2col: (optional) bool, default to false • axis: (optional) int32, default to 1 <p>[Restrictions]</p> <ul style="list-style-type: none"> • filter must be 4D constants. • $group < 1000$ • $dilation = 1$ • $filterH - padHHead - 1 \geq 0$ • $filterW - padWHead - 1 \geq 0$ • Restrictions involving intermediate variables: <ul style="list-style-type: none"> 1. $a = \text{ALIGN}(\text{filter_num}, 16) * \text{ALIGN}(\text{filter_c}, 16) * \text{filter_h} * \text{filter_w} * 2$ If $\text{ALIGN}(\text{filter_c}, 16) \% 32 = 0$, $a = a/2$ 2. $\text{conv_input_width} = (\text{deconvolution input } W - 1) * \text{strideW}$

No	Operator	Description	Boundary
			<p>+ 1</p> <p>3. $b = (\text{conv_input_width}) * \text{filter_h} * \text{ALIGN}(\text{filter_num}, 16) * 4$</p> <p>4. $a + b \leq 512 \text{ KB}$</p>
10	DepthwiseConvolution	Depthwise convolution	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: (optional) uint32 • bias_term: (optional) bool, default to true • pad: uint32, default to 0, array • kernel_size: uint32, array • stride: uint32, default to 1, array • dilation: uint32, default to 1, array • pad_h: (optional) uint32, default to 0 (2D only) • pad_w: (optional) uint32, default to 0 (2D only) • kernel_h: (optional) uint32 (2D only) • kernel_w: (optional) uint32 (2D only) • stride_h: (optional) uint32 (2D only) • stride_w: (optional) uint32 (2D only) • group: (optional) uint32, default to 1 • weight_filler: (optional) FillerParameter • bias_filler: (optional) FillerParameter • engine: (optional) enum, default to 0, Caffe = 1, CUDNN = 2 • force_nd_im2col: (optional) bool, default to false • axis: (optional) int32, default to 1 <p>[Restrictions]</p> <ul style="list-style-type: none"> • filter must be 4D constants. • $\text{filterN} = \text{inputC} = \text{group}$ • $\text{StrideW} \leq (\text{inputW} + \text{padW}) - ((\text{filterW} - 1) * \text{dilationW}) + 1$ • The shape of bias must be (1, co, 1, 1), with the same co as num_output.
11	Eltwise	Compute element-wise operations (PROD, MAX, and SUM).	<p>[Inputs] At least two inputs</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • operation: (optional) enum, (PROD = 0; SUM = 1; MAX = 2),

No	Operator	Description	Boundary
.			<p>default to SUM</p> <ul style="list-style-type: none"> • coeff: array, float • stable_prod_grad: (optional) bool, default to true <p>[Restrictions]</p> <ul style="list-style-type: none"> • Up to four inputs • Compared with the native operator, the stable_prod_grad parameter is not supported. • PROD, MAX, and SUM operations are supported.
12	Elu	Activation function	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <p>alpha: (optional) float, default to 1</p>
13	Exp	Exponential operation	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • base: (optional) float, default to -1.0 • scale: (optional) float, default to 1.0 • shift: (optional) float, default to 0.0
14	Flatten	<p>Flattens data along the first dimension.</p> <p>Converts an input of shape $N * C * H * W$ to a vector output of shape $N * (C * H * W)$.</p>	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • axis: (optional) int32, default to 1 • end_axis: (optional) int32, default to -1 <p>[Restrictions]</p> <p>axis < end axis</p>
15	InnerProduct	Computes an inner product.	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: (optional) uint32 • bias_term: (optional) bool, default to true • weight_filler: (optional) FillerParameter, 2D • bias_filler: (optional) FillerParameter, 1D

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • axis: (optional) int32, default to 1 • transpose: (optional) bool, default to false <p>[Restrictions]</p> <ul style="list-style-type: none"> • transpose = false, axis = 1 • $\text{bais_C} \leq 56832$ • The shape of bias must be (1, co, 1, 1), with the same co as num_output. <p>To quantify a model, the following dimension restrictions must be satisfied:</p> <ul style="list-style-type: none"> • When $N = 1$: $2 * \text{ALIGN}(C, 16) * xH * xW \leq 524288$; • When $N > 1$: $2 * 16 * \text{ALIGN}(C, 16) * xH * xW \leq 524288$.
16	Interp	Interpolation layer	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • height: (optional) int32, default to 0 • width: (optional) int32, default to 0 • zoom_factor: (optional) int32, default to 1 • shrink_factor: (optional) int32, default to 1 • pad_beg: (optional) int32, default to 0 • pad_end: (optional) int32, default to 0 <p>Note the following:</p> <ul style="list-style-type: none"> • zoom_factor and shrink_factor are exclusive. • height and zoom_factor are exclusive. • height and shrink_factor are exclusive. <p>[Restrictions]</p> <p>$(\text{outputH} * \text{outputW}) / (\text{inputH} * \text{inputW}) > 1/7$</p>
17	LeakyRelu	LeakyRelu activation function	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <p>Same as Relu</p>
18	Log	Performs logarithmic operation on the input.	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • base: (optional) float, default to -1.0 • scale: (optional) float, default to 1.0

No	Operator	Description	Boundary
			<ul style="list-style-type: none"> • shift: (optional) float, default to 0.0
19	LRN	Normalizes the input in a local region.	<p>[Inputs] One non-constant input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • local_size: (optional) uint32, default to 5 • alpha: (optional) float, default to 1 • beta: (optional) float, default to 0.75 • norm_region: (optional) enum, default to ACROSS_CHANNELS (ACROSS_CHANNELS = 0, WITHIN_CHANNEL = 1) • k: (optional) float, default to 1 • engine: (optional) enum, default to 0, Caffe = 1, CUDNN = 2 <p>[Restrictions]</p> <ul style="list-style-type: none"> • local_size is an odd number greater than 0. • Inter-channel: If local_size is within [1, 15]: k > 0.00001 and beta > 0.01; Otherwise, k and beta are any values. k and alpha are not 0 at the same time. When the C dimension is greater than 680, local_size < 640. • Intra-channel: k = 1, local_size is within [1,15], beta > 0.01
20	LSTM	Long and short term memory network (LSTM)	<p>[Inputs] Two or three inputs</p> <ul style="list-style-type: none"> • X: time sequence data (T * N * Xt) • Cont: sequence continuity flag (T * N) • Xs: (optional) static data (N * Xt) <p>[Arguments]</p> <ul style="list-style-type: none"> • num_output: (optional) uint32, default to 0 • weight_filler: (optional) FillerParameter • bias_filler: (optional) FillerParameter • debug_info: (optional) bool, default to false • expose_hidden: (optional) bool, default to false <p>[Restrictions]</p> <p>Restrictions involving intermediate variables:</p> <ul style="list-style-type: none"> • $a = (\text{ALIGN}(xt, 16) + \text{ALIGN}(\text{output}, 16)) * 64$ • $b = (\text{ALIGN}(xt, 16) + \text{ALIGN}(\text{output}, 16)) * 256$ • $c = \text{use_projection} ? \text{ALIGN}(ht, 16) * \text{ALIGN}(\text{output}, 16) * 2 : 0$ • $d = 16 * \text{ALIGN}(ht, 16) * 2$

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • $e = \text{batchNum} * 4$ <p>That is:</p> <ul style="list-style-type: none"> • $a + b + c \leq 524288$ • $d \leq 16384$ • $e \leq 4096$
21	Normalize	Normalize layer	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • across_spatial: (optional) bool, default to false • scale_filler: (optional) default to 1.0 • channel_shared: (optional) bool, default to true • eps: (optional) float, default to 1e - 10 <p>[Restrictions]</p> <ul style="list-style-type: none"> • $1e - 7 < \text{eps} \leq 0.1 + (1e - 6)$ • across_spatial must be false for Caffe, indicating normalization by channel
22	Permute	Reshapes the input.	<p>[Inputs] One input</p> <p>[Arguments] order: uint32, array</p>
23	Pooling	Pools the input.	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • pool: (optional) enum, indicating the pooling method, MAX = 0, AVE = 1, and STOCHASTIC = 2, default to MAX • pad: (optional) uint32, default to 0 • pad_h: (optional) uint32, default to 0 • pad_w: (optional) uint32, default to 0 • kernel_size: (optional) uint32, exclusive with kernel_h/kernel_w • kernel_h: (optional) uint32 • kernel_w: (optional) uint32, used in pair with kernel_h • stride: (optional) uint32, default to 1 • stride_h: (optional) uint32 • stride_w: (optional) uint32

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • engine: (optional) enum, default to 0, CAFFE = 1, CUDNN = 2 • global_pooling: (optional) bool, default to false • ceil_mode: (optional) bool, default to true • round_mode: (optional) enum, CEIL = 0, FLOOR = 1, default to CEIL <p>[Restrictions]</p> <ul style="list-style-type: none"> • $\text{KernelH} < 256, \text{kernelW} < 256;$ • $\text{stride} < 64; \text{stride} > \text{pad}; \text{stride} < \text{input} + \text{pad} - \text{kernel}$ • If H and W of the output tensor are 1: $\text{input H} * \text{input W} < 65536$ • $\text{kernelH} \leq \text{inputH} + \text{padTop} + \text{padBottom}$ • $\text{kernelW} \leq \text{inputW} + \text{padLeft} + \text{padRight}$ • $\text{padTop} < \text{windowH}$ • $\text{padBottom} < \text{windowH}$ • $\text{padLeft} < \text{windowW}$ • $\text{padRight} < \text{windowW}$ • Only the global pooling mode is supported. The following restrictions must be satisfied: <ul style="list-style-type: none"> 1) $\text{outputH} == 1 \ \&\& \ \text{outputW} == 1 \ \&\& \ \text{kernelH} \geq \text{inputH} \ \&\& \ \text{kernelW} \geq \text{inputW}$ 2) $\text{inputH} * \text{inputW} \leq 10000$
24	Power	Computes the output y as $(\text{scale} * x + \text{shift})^{\text{power}}$.	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • power: (optional) float, default to 1.0 • scale: (optional) float, default to 1.0 • shift: (optional) float, default to 0.0 <p>[Restrictions]</p> <p>$\text{scale} * x + \text{shift} > 0$</p>
25	Prelu	Activation function	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • filler: (optional). • channel_shared: (optional) bool, indicating whether to share slope parameters across channels, default to false

No	Operator	Description	Boundary
26	PriorBox	Obtains the real location of the target from the box proposals.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Input 0: (mandatory) Only the input shape is concerned. • Input 1: (optional) image description <p>[Arguments]</p> <ul style="list-style-type: none"> • min_size: (mandatory) indicating the minimum frame size (in pixels) • max_size: (mandatory) indicating the maximum frame size (in pixels) • aspect_ratio: array, float. A repeated ratio is ignored. If no aspect ratio is provided, the default ratio 1 is used. • flip: (optional) bool, default to true. The value true indicates that each aspect ratio is reversed. For example, for aspect ratio r, the aspect ratio 1.0/r is generated. • clip: (optional) bool, default to false. The value true indicates that the previous value is clipped to the range [0, 1]. • variance: array, used to adjust the variance of the BBoxes • img_size: (optional) uint32. exclusive with img_h/img_w • img_h: (optional) uint32 • img_w: (optional) uint32 • step: (optional) float. step_h and step_w are exclusive. • step_h: (optional) float • step_w: (optional) float • offset: (optional) float, default to 0.5 <p>[Restrictions]</p> <ul style="list-style-type: none"> • Used for the SSD network only • Output dimensions: [n, 2, detection frame * 4, 1]
27	Proposal	Sorts the box proposals by (proposal, score) and obtains the top N proposals by using the NMS.	<p>[Inputs]</p> <p>Three inputs: scores, bbox_pred, im_info</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • feat_stride: (optional) float • base_size: (optional) float • min_size: (optional) float • ratio: array (optional), float • scale: array (optional), float • pre_nms_topn: (optional) int32 • post_nms_topn: (optional) int32

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • nms_thresh: (optional) float [Restrictions] • Used only for Faster R-CNN • ProposalParameter and PythonParameter are exclusive. • Value range of preTopK: 1–6144 • Value range of postTopK: 1–1024 • $\text{scaleCnt} * \text{ratioCnt} \leq 64$ • $0 < \text{nmsTresh} \leq 1$, indicating the threshold for box filtering • minSize: minimum edge length of a proposal. A box with any side smaller than minSize is removed. • featStride: H/W stride between the two adjacent boxes used in default box generation • baseSize: base box size used in default box generation • ratio and scale: used in default box generation • imgH and imgW: height and width of the image input to the network. The values must be greater than 0. • Restrictions on the input dimensions: clsProb: $C = 2 * \text{scaleCnt} * \text{ratioCnt}$ bboxPred: $C = 4 * \text{scaleCnt} * \text{ratioCnt}$ bboxPrior: $N = \text{clsProb.N}$, $C = 4 * \text{scaleCnt} * \text{ratioCnt}$ imInfo: $N = \text{clsProb.N}$, $C = 3$
28	PSROIPooling	Position-sensitive region-of-interest pooling (PSROI Pooling)	[Inputs] Two inputs [Arguments] <ul style="list-style-type: none"> • spatial_scale: (mandatory) float • output_dim: (mandatory) int32, indicating the number of output channels • group_size: (mandatory) int32, indicating the number of groups to encode position-sensitive score maps [Restrictions] <ul style="list-style-type: none"> • Used for the Region-based Fully Convolutional Network (R-FCN) • ROI coordinates [roiN, roiC, roiH, roiW]: $1 \leq \text{roiN} \leq 65535$, $\text{roiC} == 5$, $\text{roiH} == 1$, and $\text{roiW} == 1$ • Dimensions of the input feature map: [xN, xC, xH, xW] $\text{pooledH} == \text{pooledW} == \text{group_size} \leq 128$

No	Operator	Description	Boundary
.			<p>pooledH and pooledW indicate the length and width of the pooled ROI.</p> <p>Output format: y [yN, yC, yH, yW]</p> <ul style="list-style-type: none"> poolingMode == avg pooling, pooledH == pooledW == group_size, pooledH ≤ 128, spatial_scale > 0, group_size > 0, and output_dim > 0 1 ≤ xN ≤ 65535, roiN % xN == 0 xHW = xH * xW, pooledHW = pooledH * pooledW, HW_LIMIT = 768, xH ≥ pooledH, xW ≥ pooledW, xHW ≥ pooledHW, xHW/pooledHW ≤ HW_LIMIT In multi-batch scenarios, the ROIs are allocated equally to the batches. In addition, the batch sequence of the ROIs is the same as the feature. yN == roiN, yH == pooledH, yW == pooledW, yC == output_dim; xC == yC * pooledH * pooledW
29	Relu	Activation function, including common ReLU and Leaky ReLU, which can be specified by parameters	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> negative_slope: (optional) float, default to 0 engine: (optional) enum, default to 0, Caffe = 1, CUDNN = 2
30	Reorg	Real-time object detection	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> stride: (optional) uint32, default to 2 reverse: (optional) bool, default to false <p>[Restrictions]</p> <p>Used only for YOLOv2</p>
31	Reshape	Reshapes the input.	<p>[Inputs]</p> <p>One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> shape: constant, int64 or int32 axis: (optional) int32, default to 0 <p>num_axes: (optional) int32, default to -1</p>

No	Operator	Description	Boundary
32	Reverse	Reversion	<p>[Inputs] One input</p> <p>[Arguments] axis: (optional) int32, default to 1. Controls the axis to be reversed. The content layout will not be reversed.</p>
33	ROIAlign	Aggregates features using ROIs.	<p>[Inputs] At least two inputs</p> <p>[Arguments] <ul style="list-style-type: none"> • pooled_h: (optional) uint32, default to 0 • pooled_w: (optional) uint32, default to 0 • spatial_scale: (optional) float, default to 1 • sampling_ratio: (optional) int32, default to -1 </p> <p>[Restrictions] Mainly used for Mask R-CNN Restrictions on the feature map: <ul style="list-style-type: none"> • $N < 65535$ • $H * W \leq 2464$ • $C \leq 1152$ • $((C - 1)/128 + 1) * \text{pooledW} \leq 92$ Restrictions on the ROI: <ul style="list-style-type: none"> • $N < 65535$ • $C = 5(\text{Caffe}), H = 1, W = 1$ • $\text{samplingRatio} * \text{pooledW} \leq 128, \text{samplingRatio} * \text{pooledH} \leq 128$ • $H \geq \text{pooledH}, W \geq \text{pooledW}$ </p>
34	ROIPooling	Maps ROI proposals to a feature map.	<p>[Inputs] At least two inputs</p> <p>[Arguments] <ul style="list-style-type: none"> • pooled_h: (optional) uint32, default to 0 • pooled_w: (optional) uint32, default to 0 • spatial_scale: (optional) float, default to 1. The multiplication spatial scale factor is used to convert ROI coordinates from the input scale to the pool scale. </p> <p>[Restrictions] Mainly used for Faster R-CNN</p>

No	Operator	Description	Boundary
			<ul style="list-style-type: none"> Feature map size (H * W): input up to 3888, output up to 256
35	Scale	out = alpha * Input + beta	<p>[Inputs] Two inputs</p> <p>[Arguments]</p> <ul style="list-style-type: none"> axis: (optional) int32, default to 1. Only 1 or -3 is supported. num_axes: (optional) int32, default to 1 filler: (optional) ignored unless only one bottom is given and scale is a learned parameter bias_term: (optional) bool, default to false, indicating whether to learn a bias (equivalent to ScaleLayer + BiasLayer, but may be more efficient). Initialized with bias_filler. bias_filler: (optional) default to 0 <p>[Restrictions] shape of scale and bias: (n, c, 1, 1), with the C dimension equal to that of the input</p>
36	ShuffleChannel	Shuffles information across the feature channels.	<p>[Inputs] One input</p> <p>[Arguments] group: (optional) uint32, default to 1</p>
37	Sigmoid	Activation function	<p>[Inputs] One input</p> <p>[Arguments] engine: (optional) enum, default to 0, CAFFE = 1, CUDNN = 2</p>
38	Slice	Slices an input into multiple outputs.	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> slice_dim: (optional) uint32, default to 1, exclusive with axis slice_point: array, uint32 axis: (optional) int32, default to 1, indicating concatenation along the channel dimension <p>[Returns]</p>
39	Softmax	Normalization logic function	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> engine: (optional) default to 0, CAFFE = 1, CUDNN = 2 axis: (optional) int32, default to 1, indicating the axis along

No	Operator	Description	Boundary
.			<p>which softmax is performed</p> <p>[Restrictions]</p> <p>axis is of the range $[-rank, rank)$.</p> <p>Softmax can be performed on each of the four input dimensions (NCHW).</p> <ul style="list-style-type: none"> axis = 0: $N \leq 28544$ axis = 1 (channel dimension): $C \leq 11136$ axis = 2 (height dimension): $W = 1, 0 < H < 16384$ axis = 3 (width dimension): $0 < W < 16384$ <p>If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968.</p>
40	SSDDetectionOutput	SSD network detection output	<p>[Inputs]</p> <p>Three inputs</p> <p>[Arguments]</p> <ul style="list-style-type: none"> num_classes: (mandatory) int32, indicating the number of classes to be predicted, including the background class share_location: (optional) bool, default to true, indicating that classes share one BBox background_label_id: (optional) int32, default to 0 nms_param: (optional) indicating non-maximum suppression (NMS) save_output_param: (optional) indicating whether to save the detection result code_type: (optional) default to CENTER_SIZE variance_encoded_in_target: (optional) bool, default to true. The value true indicates that the variance is encoded in the target, otherwise the prediction offset needs to be adjusted accordingly. keep_top_k: (optional) int32, indicating the total number of BBoxes to be reserved for each image after NMS confidence_threshold: (optional) float, indicating that only the detection whose confidence is above the threshold is considered. If this parameter is not set, all boxes are considered. nms_threshold: (optional) float top_k: (optional) int32 boxes: (optional) int32, default to 1

No	Operator	Description	Boundary
.			<ul style="list-style-type: none"> • relative: (optional) bool, default to true • objectness_threshold: (optional) float, default to 0.5 • class_threshold: (optional) float, default to 0.5 • biases: array • general_nms_param (optional) [Restrictions] • Used for the SSD network under Caffe • Value range of preTopK and postTopK: 1–1024 • shareLocation = true • nmsEta = 1 • Value range of numClasses: 1–2048 • code_type = CENTER_SIZE • Value range of nms_threshold and confidence_threshold: 0.0–1.0 • In multi-batch scenario, multi-batch priorbox results are also generated.
41	Tanh	Activation function	[Inputs] One input [Arguments] engine : (optional) enum, default to 0 , CAFFE = 1, CUDNN = 2
42	Upsample	Backward propagation of max pooling	[Inputs] Two inputs [Arguments] scale : (optional) int32, default to 1
43	SpatialTransform (available since V310)	Spatial transformation	[Inputs] One input [Arguments] <ul style="list-style-type: none"> • output_h: (mandatory) uint32, default to 0 • output_w: (mandatory) uint32, default to 0 • border_value: (optional) float, default to 0 • affine_transform: float • engine: (optional) enum, default to 0, CAFFE = 1, CUDNN = 2 [Restrictions] $(\text{outputH} * \text{outputW}) / (\text{inputH} * \text{inputW}) > 1/7$ [Returns]

No	Operator	Description	Boundary
44	Tile (available since V320)	Creates a Blob by replicating input multiples times along a given axis.	<p>[Inputs] One input</p> <p>[Arguments] <ul style="list-style-type: none"> • axis: int32, axis to tile. For details about the value, see section 2.1 General Restrictions. • tiles: int32, replication times </p>
45	Split (available since V320)	Splits an input Blob to multiple output Blobs.	<p>[Inputs] One input</p> <p>Supported data types: float, double, int8, uint8, int32, uint32, int64, uint64, bool</p> <p>Supported data format: NCHW</p>
46	BatchReindex (available since V320)	Selects, reorders, and replicates the input in a batch.	<p>[Inputs] Two inputs</p> <ul style="list-style-type: none"> • Input 0: float, input data • Input 1: uint32, input index <p>[Restrictions] $0 \leq \text{input1 batch size} < \text{input0 batch size}$ If the restriction is exceeded, the output result is unreliable. </p>
47	SPP (available since V320)	Performs spatial pyramid pooling (SPP) on the input by taking the max, average, and more within regions so that the result vector of different sized images are of the same size.	<p>[Inputs] One input</p> <p>[Arguments] <ul style="list-style-type: none"> • pyramid_height: (mandatory) int32, pyramid height • pool: (optional) int32, either 0 (MAX) or 1 (AVG) </p>
48	Threshold (available since V320)	Tests whether the input exceeds a threshold: outputs 1 for	<p>[Inputs] One input</p> <p>[Arguments] threshold: (optional) float32, default to 0.0 </p>

No	Operator	Description	Boundary
	V320)	inputs above threshold; 0 otherwise.	
49	MVN (available since V320)	The Mean-Variance Normalization (MVN) layer normalizes the input.	<p>[Inputs] One input</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • normalize_variance: (optional) bool. The value false indicates to normalize mean only and true indicates to normalize variance only. Default to true. • across_channels: (optional) bool. If false, batch * channel is used as the row of the matrix, and height * width is used as the column of the matrix. If true, batch is used as the row of the matrix, and channel * height * width is used as the column of the matrix. Default to false.
50	BNLL (available since V320)	Computes output as binomial normal log likelihood.	<p>[Inputs] One input</p>
51	Swish (available since V320)	Swish takes one input data (Tensor) and produces one output data (Tensor) where the swish function, $y = x * \sigma(\text{scale} * x)$, is applied to the tensor element-wise.	<p>[Inputs] One input</p> <p>[Arguments]</p> <p>beta: float32, scaling factor of the swish function</p>
52	Bias (available since V320)	Computes a sum of two inputs input 0 and bias , with the shape of bias broadcast to match the shape of input 0	<p>[Inputs] Two inputs</p> <p>[Arguments]</p> <p>axis: scalar of type int, specifying the broadcast start. The value range is [-4, +3].</p>

No	Operator	Description	Boundary
		(determined by axis).	
53	Dropout (available since V310)	Avoids overfitting.	[Inputs] One input [Arguments] <ul style="list-style-type: none"> • dropout_ratio: (optional) float, default to 0.5 • scale_train: (optional) bool, default to true
54	ReLU6 (available since V320)	Activation function	[Inputs] One input [Arguments] negative_slope : (optional) float, default to 0 [Restrictions] negative_slope must be 0 .

2.3 TensorFlow Operator Boundaries

No	Python API	C + + API	Boundary
1	tf.abs	Abs	[Arguments] <ul style="list-style-type: none"> • x: Tensor of type float32 or int32 • name: (optional) string [Returns] Returns the absolute value of x , Tensor. The size and type are the same as those of x .
2	tf.add	Add	[Arguments] <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of the identical type as x. For two constant inputs, one of them is a scalar. • name: (optional) string [Restrictions] If the two inputs have inconsistent dimensions, broadcasting (that is, dimension padding) is performed. Broadcasting is supported in the following scenarios: <ul style="list-style-type: none"> • NHWC + NHWC, NHWC + scalar

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • NHWC + 1 1 1 1 • NHWC + W, HWC + W, HW + W (broadcasting along W) • NCHW + NH1C, HWC + H1C, HW + H1 • HWC + 1 WC (broadcasting along H) <p>Note: The input sequence of the two Tensors is not fixed.</p> <p>[Returns]</p> <p>Tensor of the identical type as y</p>
3	tf.add_n (available since V310)	AddN	<p>[Arguments]</p> <ul style="list-style-type: none"> • inputs: list of Tensor or IndexedSlices objects of type float32, with the same shape • name: (optional) string <p>[Restrictions]</p> <p>A single input is not supported.</p> <p>[Returns]</p> <p>Tensor with the identical shape as inputs</p>
4	tf.batch_to_space_nd	BatchToSpaceND	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: n-D Tensor of type float32, with shape: input_shape = [batch] + spatial_shape + remaining_shape, where spatial_shape has M dimensions. • block_shape: 1D Tensor of type int32, with shape [M]. All values must be ≥ 1. • crops: 2D Tensor of type int32, with shape [M, 2]. All values must be ≥ 0. <p>[Restrictions]</p> <ul style="list-style-type: none"> • The element data type of block_shape and crops must be int32. When the dimension count of the Tensor is 4, the length of block_shape must be 2, and the length of crops must be 4. • Element value of block_shape ≥ 1; Element value of crops ≥ 0 • Crops array: $\text{crop_start}[i] + \text{crop_end}[i] < \text{block_shape}[i] * \text{input_shape}[i + 1]$ <p>[Returns]</p> <p>Tensor of the identical type as images</p>
5	tf.cast	Cast	<p>[Inputs]</p> <p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor, must be one of the following types: float32,

No	Python API	C + + API	Boundary
.			<p>int32, bool, int64, int16, int8, uint8, uint16, double</p> <ul style="list-style-type: none"> • dtype: destination type, same as the data type of x • name: (optional) string <p>[Returns]</p> <p>Tensor, SparseTensor, or IndexedSlices, same dtype and shape as the input</p>
6	tf.math.ceil (available since V310)	Ceil	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
7	tf.clip_by_value	ClipByValue	<p>[Arguments]</p> <ul style="list-style-type: none"> • t: Tensor • clip_value_min: minimum value to clip by • clip_value_max: maximum value to clip by • name: (optional) string <p>[Restrictions]</p> <p>The minimum value must be less than or equal to the maximum value.</p> <p>[Returns]</p> <p>Clipped Tensor. The return value range is [clip_value_min, clip_value_max].</p>
8	tf.concat	ConcatV2	<p>[Arguments]</p> <ul style="list-style-type: none"> • values: list of Tensor objects or a single Tensor. The values of dimensions must be the same except the dimensions to be concatenated. • axis: 0D Tensor of type int32, specifying the dimension to be concatenated. The value range is [-rank(values), rank(values)]. As in Python, indexing for axis is 0-based. Positive axis in the range [0, rank(values)) refers to axis-th dimension, while negative axis refers to [axis + rank(values)]-th dimension. <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of dimensions of the input tensors must match, and all dimensions except axis must be equal. • The range of the input Tensor count is [1, 32]. <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor, resulting from concatenation of the input Tensors
9	tf.constant	Const	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: constant value (or list) • dtype: data type of the resulting Tensor • shape: (optional) dimensions of the resulting Tensor • name: (optional) string <p>verify_shape: (optional) Boolean that enables verification of a shape of values, default to False</p> <p>[Returns]</p> <p>One constant Tensor</p>
10	tf.depth_to_space	DepthToSpace	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: Tensor of type float32 • block_size: integer scalar, ≥ 2 • data_format: string, either NHWC (default) or NCHW • name: (optional) string <p>[Restrictions]</p> <p>blockSize must be greater than or equal to 1, and $\text{blockSize} * \text{blockSize}$ must be exactly divided by C.</p> <p>[Returns]</p> <p>Tensor of the identical type as input</p>
11	tf.equal	Equal	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor, must be one of the following types: float32, uint8, int32, bool • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>Broadcasting is supported, so the shape of x and shape of y are compared. For a right-aligned dimension, if the values of xdim[i] and ydim[i] are not the same, one of them must be 1 or missing.</p> <p>[Returns]</p> <p>Tensor of type bool</p>
12	tf.exp	Exp	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 or double • name: (optional) string

No	Python API	C + + API	Boundary
			[Returns] Tensor of the identical type as x
13	tf.math.exp m1 (available since V310)	Exp m1	[Arguments] <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string [Returns] Tensor of the identical type as x
14	tf.expand_dim s	ExpandDims	[Arguments] <ul style="list-style-type: none"> • input: Tensor • axis: 0D scalar, specifying the dimension index of the extended input shape • name: name of the output Tensor dim : (deprecated) 0D scalar, equivalent to axis [Returns] Tensor with the same data as input , but its shape has an additional dimension of size 1 added
15	tf.extract_image_patches	ExtractImagePatches	[Arguments] <ul style="list-style-type: none"> • images: 4D Tensor of type float32 or uint8, with shape [batch, in_rows, in_cols, depth] • ksize: list of integers with length ≥ 4 • strides: list of integers, with shape [1, stride_rows, stride_cols, 1] • rate: list of integers, with shape [1, rate_rows, rate_cols, 1] • padding: string, either VALID or SAME. VALID indicates that the selected patch area must be completely included in the source image. SAME indicates that the part that exceeds the source image is padded with 0. name : (optional) string [Returns] Tensor of the identical type as images
16	tf.fake_quant_with_min_max_vars	FakeQuantWithMinMaxVars	[Arguments] <ul style="list-style-type: none"> • inputs: Tensor of type float32 • min: Tensor of type float32 • max: Tensor of type float32 • num_bits: integer scalar, default to 8 • narrow_range: (optional) bool, default to False

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • name: (optional) string <p>[Restrictions]</p> <p>$-65504 \leq \min \leq +65504, -65504 \leq \max \leq +65504$</p> <p>[Returns]</p> <p>Tensor of type float32</p>
17	tf.fill	Fill	<p>[Arguments]</p> <ul style="list-style-type: none"> • dims: 1D Tensor of type int32 • value: variable of type int32 or float32 • name: (optional) string <p>[Restrictions]</p> <p>The following padding modes are supported: Constant, GivenTensor, Range, Diagonal, Gaussian, MSRA, Uniform, UniformInt, UniqueUniform, and XavierFill. When the Uniform, UniformInt, UniqueUniform, and xavier padding modes are used, the value range of the generated value is [min, max).</p> <p>[Returns]</p> <p>Tensor of the identical type as value</p>
18	tf.floormod	FloorMod	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 or int32 • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>Broadcasting is supported, so the shape of x and shape of y are compared. For a right-aligned dimension, if the values of xdim[i] and ydim[i] are not the same, one of them must be 1 or missing.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
19	tf.gather	Gather GatherV2	<p>[Arguments]</p> <ul style="list-style-type: none"> • params: Tensor, must be at least rank axis + 1 • indices: Tensor of type float32 or int64, must be in range [0, params.shape[axis]) • axis: Tensor of type float32 or int64, specifying the axis in params to gather indices from, rank = 0 <p>name: (optional) string</p>

No	Python API	C + + API	Boundary
			<p>[Returns]</p> <p>Tensor of the identical type as params</p>
20	tf.gather_nd	GatherNd	<p>[Arguments]</p> <ul style="list-style-type: none"> • params: Tensor, must be at least rank axis + 1 • indices: Tensor of type int32 or int64 • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • indices: The last dimension of indices can be at most the rank of params. • The elements in the last dimension of indices correspond to the coordinates along a dimension of params. Therefore, the coordinate rules must be met. <p>The coordinates along the corresponding dimension of indices cannot exceed the dimension size.</p> <p>[Returns]</p> <p>Tensor of the identical type as params</p>
21	tf.greater	Greater	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 or int32 • y: Tensor of type float32 or int32 • name: (optional) string <p>[Restrictions]</p> <p>Only constant inputs are accepted.</p> <p>Broadcasting is supported.</p> <p>[Returns]</p> <p>Tensor of type bool</p>
22	tf.image.crop_and_resize	CropAndResize	<p>[Arguments]</p> <ul style="list-style-type: none"> • image: 4D Tensor, must be one of the following types: float32 and int8, int32, int64; with shape [batch, image_height, image_width, depth] • boxes: 2D Tensor of type float32, with shape [num_boxes, 4] • box_ind: 1D Tensor of type int32, with shape [num_boxes] • crop_size: 1D 2-element Tensor of type int32 • method: interpolation method string, options: bilinear (default) or nearest • extrapolation_value: (optional) float32, default to 0

No	Python API	C + + API	Boundary
			name: (optional) string [Returns] Tensor of type float32
23	tf.image.non_max_suppression (available since V310)	NonMaxSuppressionV2	[Arguments] <ul style="list-style-type: none"> • boxes: 2D Tensor of type float32, with shape [num_boxes, 4] • scores: 1D Tensor of type float32, with shape [num_boxes] • max_output_size: scalar of type int32, representing the maximum number of boxes to be selected • iou_threshold: scalar of type float32 • name: (optional) string [Returns] 1D Tensor of type int32, with shape [M], where $M \leq \text{max_output_size}$
24	tf.image.resize_bilinear	ResizeBilinear	[Arguments] <ul style="list-style-type: none"> • images: 4D Tensor with shape [batch, height, width, channels] of type float32 • size: 1D 2-element constant Tensor, indicating the new size for the images • align_corners: bool, default to False. The value true indicates that the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. [Restrictions] $(\text{outputH} * \text{outputW}) / (\text{inputH} * \text{inputW}) > 1/7$ [Returns] Tensor of type float, with the identical shape as the input
25	tf.image.resize_nearest_neighbor	ResizeNearestNeighbor	[Arguments] <ul style="list-style-type: none"> • images: 4D Tensor of type float32, with shape [batch, height, width, channels] • size: 1D 2-element constant Tensor, indicating the new size for the images align_corners: bool, default to False . The value true indicates that the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. [Returns]

No	Python API	C + + API	Boundary
			Tensor of type float, with the identical shape as the input
26	tf.invert_permutation	InvertPermutation	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: 1D Tensor of type int32 or int64 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
27	tf.keras.backend.hard_sigmoid	Hardsigmoid	<p>[Arguments]</p> <p>x: Tensor</p> <p>[Returns]</p> <p>Output Tensor</p> <ul style="list-style-type: none"> • If $x < -2.5$, returns 0. • If $x > 2.5$, returns 1. • If $-2.5 \leq x \leq 2.5$, returns $0.2 * x + 0.5$.
28	tf.keras.layers.ThresholdedReLU	ThresholdedReLU	<p>[Arguments]</p> <p>theta: scalar of type float32, ≥ 0</p> <p>[Restrictions]</p> <p>$0 \leq \text{theta} \leq 65504$</p> <p>[Returns]</p> <p>Tensor</p>
29	tf.log	Log	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>$x > 0$</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
30	tf.math.log1p (available since V310)	Log1p	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
31	tf.math.acos	Acos	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor, must be one of type float32

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • name: (optional) string <p>[Restrictions]</p> <p>The input data range is $(-1 \leq x \leq +1)$, and the output data range is $(0 \leq y \leq \pi)$.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
32	tf.math.acosh	Acosh	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>$x > 0$</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
33	tf.math.argmax ax	ArgMax	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: Tensor, must be one of the following types: int8, uint8, int16, uint16, int32, int64, float32 • axis: Tensor of type int32 or int64 • out_type: data type for the output Tensor, either int32 or int64 (default) <p>name: (optional) string</p> <p>[Returns]</p> <p>Tensor of the data type specified by out_type</p>
34	tf.math.asin	Asin	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>The input data range is $(-1 \leq x \leq +1)$, and the output data range is $(-\pi/2 \leq y \leq +\pi/2)$.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
35	tf.math.asinh	Asinh	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 <p>name: (optional) string</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>

No	Python API	C + + API	Boundary
36	tf.math.atan	Atan	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>The input data range is $(-65504 \leq x \leq +65504)$, and the output data range is $(-\pi/2 \leq y \leq +\pi/2)$.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
37	tf.math.atanh	Atanh	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>Input data range: x is within $(-1, 1)$</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
38	tf.math.cosh	Cosh	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
39	tf.math.floor (available since V310)	Floor	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>8-bit quantization is not supported.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
40	tf.math.greater_equal	GreaterEqual	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>Input data range: $-65504 \leq x \leq +65504$</p> <p>[Returns]</p> <p>Tensor of type bool</p>

No	Python API	C + + API	Boundary
41	tf.math.less	Less	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of the identical type as x • name: (optional) string <p>[Returns]</p> <p>Tensor of type bool</p>
42	tf.math.logical_and	LogicalAnd	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: non-constant Tensor of type bool • y: non-constant Tensor of type bool • name: (optional) string <p>[Restrictions]</p> <p>Broadcasting is supported in the following dimension scenarios: NHWC and [1,1,1,1], [N,C,H,W], [N,1,H,W], [1,C,H,W], [N,C,1,1]</p> <p>[Returns]</p> <p>Tensor of type bool</p>
43	tf.math.logical_not	LogicalNot	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type bool • name: (optional) string <p>[Returns]</p> <p>Tensor of type bool</p>
44	tf.math.logical_or	LogicalOr	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: non-constant Tensor of type bool • y: non-constant Tensor of type bool • name: (optional) string <p>[Restrictions]</p> <p>Broadcasting is supported, so the shape of x and shape of y are compared. For a right-aligned dimension, if the values of xdim[i] and ydim[i] are not the same, one of them must be 1 or missing.</p> <p>[Returns]</p> <p>Tensor of type bool</p>
45	tf.math.maximum	Maximum	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type int32 or float32 • y: Tensor of the identical type as x

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • name: (optional) string <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Tensor. Returns the max of x and y ($x > y ? x : y$), of identical type as x</p>
46	tf.math.minimum	Minimum	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type int32 or float32 • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Tensor. Returns the min of x and y ($x < y ? x : y$), of identical type as x</p>
47	tf.math.negative	Neg	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>The input data range is $-65504 \leq x \leq +65504$, and the output data range is $-65504 \leq y \leq +65504$.</p> <p>[Returns]</p> <p>Tensor. Returns $-x$.</p>
48	tf.math.pow	Power	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Tensor</p>
49	tf.math.reciprocal	Reciprocal	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Restrictions]</p>

No	Python API	C + + API	Boundary
.			<p>The input data cannot contain 0.</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
50	tf.math.reduce_all	All	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor of type bool • axis: dimension to reduce • keepdims: scalar of type bool • name: (optional) string • reduction_indices: (deprecated) equivalent to axis <p>keep_dims: (deprecated) equivalent to keepdims</p> <p>[Returns]</p> <p>Tensor of the identical type as input_tensor</p>
51	tf.math.reduce_max	ReduceMax	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor, must be one of the following types: float32, int64, uint8, uint16, int8, int16 • axis: dimension to reduce • keepdims: scalar of type bool • name: (optional) string • reduction_indices: (deprecated) equivalent to axis • keep_dims: (deprecated) equivalent to keepdims <p>[Restrictions]</p> <p>axis is of the range $[-rank, rank)$.</p> <p>[Returns]</p> <p>Tensor of the identical type as input_tensor</p>
52	tf.math.reduce_mean (available since V310)	Mean	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor of type float32 • axis: dimension to reduce • keepdims: scalar of type bool. • name: (optional) string • reduction_indices: (deprecated) equivalent to axis • keep_dims: (deprecated) equivalent to keepdims <p>[Restrictions]</p> <ul style="list-style-type: none"> • axis is of the range $[-rank, rank)$. • Dimension reduction support of axis:

No	Python API	C + + API	Boundary
.			<p>1D: All scenarios are supported.</p> <p>2D: Only 1H and C1 scenarios are supported, where, 0 and 1 indicate C and H, respectively.</p> <p>3D: Only C11, CH1, C1W, 111, 1H1, and 1HW scenarios are supported, where, 0, 1, and 2 indicate C, H, and W, respectively.</p> <p>4D: Only NC11, NCH1, NC1W, N111, N1H1, and N1HW scenarios are supported, where, 0, 1, 2, and 3 indicate N, C, H, and W, respectively.</p> <p>[Returns]</p> <p>Reduced Tensor of the identical type as input_tensor</p>
53	tf.math.reduce_min	Min	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor, must be one of the following types: float32, int64, int32, uint8, uint16, int8, int16 • axis: dimension to reduce • keepdims: scalar of type bool • name: (optional) string • reduction_indices: (deprecated) equivalent to axis • keep_dims: (deprecated) equivalent to keepdims <p>[Restrictions]</p> <ul style="list-style-type: none"> • When the input Tensor has four dimensions: input axis = {3,{1,2,3}}, keepdims = true, $H * W \leq 512$ • When the input Tensor has two dimensions: input axis = {1,{1}}, keepdims = true, $H * W * \text{ALIGN}(C, 16) \leq 8192$ <p>[Returns]</p> <p>Reduced Tensor of the identical type as input_tensor</p>
54	tf.math.reduce_prod	Prod	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor, must be one of the following types: float32, int64, int32, uint8, uint16, int8, int16 • axis: dimension to reduce • keepdims: scalar of type bool • name: (optional) string • reduction_indices: (deprecated) equivalent to axis • keep_dims: (deprecated) equivalent to keepdims <p>[Restrictions]</p> <ul style="list-style-type: none"> • If the input tensor is 4-dimensional: axis = {3, {1,2,3}}, keepdims = true, $H \times W \leq 512$

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> If the input tensor is 2-dimensional: axis={1,{1}}, keepdims=true, $H*W*ALIGN(C, 16) \leq 8192$ [Returns] Tensor of the identical type as input_tensor
55	tf.math rint	Rint	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name : (optional) string [Returns] Tensor of the identical type and shape as x
56	tf.math round	Round	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name: (optional) string [Returns] Tensor of the identical type and shape as x
57	tf.math rsqrt	Rsqrt	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name: (optional) string [Returns] Tensor of the identical type as x
58	tf.math sinh	Sinh	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name: (optional) string [Returns] Tensor of the identical type as x
59	tf.math sin (available since V310)	Sin	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name: (optional) string [Returns] Tensor of the identical type as x
60	tf.math cos (available since V310)	Cos	[Arguments] <ul style="list-style-type: none"> x: Tensor of type float32 name: (optional) string [Returns] Tensor of the identical type as x

No	Python API	C + + API	Boundary
61	tf.math.sqrt	Sqrt	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
62	tf.math.squared_difference	SquaredDifference	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor, must be one of the following types: float32, int64, int32 • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>Broadcasting is supported only in the following scenarios: One NCHW Tensor and one Tensor of the following format: dim{} = [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], or [N,1,1,1]</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
63	tf.math.tan	Tan	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
64	tf.math.top_k	TopKV2	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: n-D Tensor ($N \geq 1$) with the last dimension at least k • k: scalar of type int32, ≥ 1 • sorted: bool • name: (optional) string <p>[Restrictions]</p> <p>k must be a constant.</p> <p>[Returns]</p> <ul style="list-style-type: none"> • values: Tensor, indicating k largest elements along each last dimensional slice • indices: Tensor, indicating the indices of values of input
65	tf.matmul	MatMul	<p>[Arguments]</p>

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • a: Tensor of type float32, $2 \leq \text{rank} \leq 4$ • b: Tensor with the same data type and rank as a • transpose_a: The value true indicates that a is transposed before multiplication. Must be false when rank > 2. • transpose_b: The value true indicates that b is transposed before multiplication. • adjoint_a: The value must be False. • adjoint_b: The value must be False. • a_is_sparse: The value must be False. • b_is_sparse: The value must be False. • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • When rank = 2, for the matrix operation $[m,n] * [n,k]$, n must be less than or equal to 1664. If transposing is required, n must be less than or equal to 1664 after transposing. • When rank > 2, shape [-1] of a must be less than or equal to 1024. <p>[Returns]</p> <p>Tensor of the identical type as a and b</p>
66	tf.multinomial	Multinomial	<p>[Arguments]</p> <ul style="list-style-type: none"> • logits: 2D Tensor with shape [batch_size, num_classes] • num_samples: scalar, indicating the number of samples to draw • seed: int32 or int64, used to create a random seed • name: (optional) string • output_dtype: integer, data type for the output Tensor, default to int64 <p>[Restrictions]</p> <p>When seed is 0, the generated random is dynamic.</p> <p>[Returns]</p> <p>Tensor with shape [batch_size, num_samples]</p>
67	tf.math.multiply	Multiply	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of the identical type as x • name: (optional) string

No	Python API	C + + API	Boundary
.			<p>[Restrictions]</p> <p>If the two inputs have inconsistent dimensions, broadcasting (that is, dimension padding) is performed.</p> <p>Broadcasting is supported only in the following scenarios:</p> <ul style="list-style-type: none"> • NHWC + NHWC, NHWC + scalar • NHWC + 1 1 1 1 • NHWC + W, HWC + W, HW + W (broadcasting along W) • NCHW + NH1C, HWC + H1C, HW + H1 • HWC + 1 WC (broadcasting along H) <p>Note: The input sequence of the two Tensors is not fixed.</p> <p>[Returns]</p> <p>Tensor</p>
68	tf.nn.avg_pool	AvgPool	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor of type float32, with shape [batch, height, width, channels] • ksize: list or tuple of four integers, each value corresponding to the window size for each dimension of the input tensor. • strides: list or tuple of four integers, each value corresponding to the stride of the sliding window for each dimension of the input tensor • padding: string, either VALID or SAME • data_format: string, either NHWC (default) or NCHW • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • $\text{KernelH} < 256$, $\text{kernelW} < 256$ • If H and W of the output tensor are 1: $\text{input H} * \text{input W} < 65536$ • $\text{kernelH} \leq \text{inputH} + \text{padTop} + \text{padBottom}$ • $\text{kernelW} \leq \text{inputW} + \text{padLeft} + \text{padRight}$ • $\text{padTop} < \text{windowH}$ • $\text{padBottom} < \text{windowH}$ • $\text{padLeft} < \text{windowW}$ • $\text{padRight} < \text{windowW}$ <p>[Returns]</p> <p>Tensor of the identical type as value</p>

No	Python API	C + + API	Boundary
69	tf.nn.bias_add	BiasAdd	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: Tensor • bias: 1D constant Tensor, with size matching the last dimension of value, of the same type as value unless value is a quantized type • data_format: string, either NHWC or NCHW • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • $C \leq 10000$ • input and bias must have the same data layout. • When bias is added to the C dimensions, the C dimensions of input and bias must be the same. <p>[Returns]</p> <p>Tensor of the identical type as value</p>
70	tf.nn.conv2d	Conv2D	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor of type float32, with shape [batch, height, width, channels] • filter: constant Tensor, with same data type and dimensions as value, with shape [filter_height, filter_width, in_channels, out_channels] • strides: non-null list or tuple of four integers, each value corresponding to the stride of the sliding window for each dimension of the input tensor • padding: non-null string, either VALID or SAME • use_cudnn_on_gpu: bool, default to True • data_format: non-null, string, either NHWC (default) or NCHW • dilations: (optional) list of four integers, default to [1,1,1,1], each value corresponding to a dimension. If $k > 1$, $k - 1$ units are skipped at the corresponding dimension in filtering. The dimension sequence is determined by data_format. The values of batch and depth of dilations must be 1. • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • $(inputW + padWHead + padWTail) \geq (((FilterW - 1) * dilationW) + 1)$ • $(inputW + padWHead + padWTail) / StrideW + 1 \leq$

No	Python API	C + + API	Boundary
.			<p>INT32_MAX</p> <ul style="list-style-type: none"> • $(\text{inputH} + \text{padHHead} + \text{padHTail}) \geq (((\text{FilterH} - 1) * \text{dilationH}) + 1)$ • $(\text{inputH} + \text{padHHead} + \text{padHTail}) / \text{StrideH} + 1 \leq \text{INT32_MAX}$ • $0 \leq \text{Pad} < 256, 0 < \text{FilterSize} < 256, 0 < \text{Stride} < 64, 1 \leq \text{dilationSize} < 256$ • $\text{StrideW} \leq (\text{inputW} + \text{padW}) - ((\text{filterW} - 1) * \text{dilationW}) + 1$ <p>[Returns]</p> <p>Tensor of the identical type as value</p>
71	tf.nn.conv2d_transpose	Conv2DBackpropInput	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor with shape [batch, height, width, in_channels] for NHWC data format or [batch, in_channels, height, width] for NCHW data format • filter: 4D constant Tensor with shape [height, width, output_channels, in_channels] • output_shape: 1D Tensor, indicating the output shape • strides: non-null list of integers, each value corresponding to the stride of the sliding window for each dimension of the input tensor • padding: non-null, string, either VALID or SAME • data_format: non-null string, either NHWC or NCHW • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • $\text{filterH} - \text{padHHead} - 1 \geq 0$ • $\text{filterW} - \text{padWHead} - 1 \geq 0$ <p>Restrictions involving intermediate variables:</p> <ul style="list-style-type: none"> • $a = \text{ALIGN}(\text{filter_num}, 16) * \text{ALIGN}(\text{filter_c}, 16) * \text{filter_h} * \text{filter_w} * 2$ • If $\text{ALIGN}(\text{filter_c}, 16) \% 32 = 0, a = a/2$ • $\text{conv_input_width} = (\text{deconvolution input W} - 1) * \text{strideW} + 1$ • $b = (\text{conv_input_width}) * \text{filter_h} * \text{ALIGN}(\text{filter_num}, 16) * 4$ • $a + b \leq 512 \text{ KB}$ <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor of the identical type as value
72	tf.nn.depthwise_conv2d	DepthwiseConv2dNative	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: 4D • filter: 4D constant, with shape [filter_height, filter_width, in_channels, channel_multiplier] • strides: non-null list of four integers, each value corresponding to the stride of the sliding window for each dimension of the input tensor • padding: string, either VALID or SAME • rate: 1D of size 2. The dilation rate in which we sample input values across the height and width dimensions in atrous convolution. If it is greater than 1, then all values of strides must be 1. • data_format: data format for input, either NHWC (default) or NCHW • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • filterN = inputC = group • $\text{StrideW} \leq (\text{inputW} + \text{padW}) - ((\text{filterW} - 1) * \text{dilationW}) + 1$ <p>[Returns]</p> <p>4D Tensor, with shape according to data_format. For example, for format NHWC, shape = [batch, out_height, out_width, in_channels * channel_multiplier] for the NHWC format</p>
73	tf.nn.elu	Elu	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as features</p>
74	tf.nn.fused_batch_norm	FusedBatchNorm	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: input, 4D Tensor of type float32 • scale: 1D Tensor for scaling • offset: 1D Tensor for bias • mean: 1D Tensor for population mean used for inference • variance: 1D Tensor for population variance used for inference

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • epsilon: small float number added to the variance of x • data_format: data format for x, either NHWC (default) or NCHW • is_training: bool, specifying whether the operation is used for training or inference • name: (optional) string <p>[Restrictions]</p> <p>The shape of scale, bias, mean, and var must be (1, C, 1, 1), with the same C dimension as input.</p> <p>[Returns]</p> <ul style="list-style-type: none"> • y: 4D Tensor for the normalized, scaled, offset x • batch_mean: 1D Tensor for the mean of x • batch_var: 1D Tensor for the variance of x
75	tf.nn.l2_normalize	L2Normalize	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: 4-D tensor of type float32 • axis: dimension along which to normalize For format NCHW, axis must be set to 1. For format NHWC, axis must be set to 3. • epsilon: lower bound value for the norm. If norm < sqrt(epsilon), sqrt(epsilon) is used as the divisor. • name: (optional) string • dim: (deprecated) equivalent to axis <p>[Restrictions]</p> <p>$H * W * 2 < 32768$</p> <p>[Returns]</p> <p>Tensor of the identical type as x</p>
76	tf.nn.leaky_relu	LeakyRelu	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor of type float32 • alpha: slope of the activation function at $x < 0$ • name: (optional) string <p>[Returns]</p> <p>Activation value</p>
77	tf.nn.log_softmax (available since V310)	LogSoftmax	<p>[Arguments]</p> <ul style="list-style-type: none"> • logits: non-null Tensor of type float32 • axis: dimension softmax would be performed on, default to

No	Python API	C + + API	Boundary
.			<p>-1, indicating the last dimension</p> <ul style="list-style-type: none"> • name: (optional) string • dim: (deprecated) equivalent to axis <p>[Restrictions]</p> <p>axis is of the range [-rank, rank).</p> <p>Softmax can be performed on each of the four input dimensions (NCHW).</p> <ul style="list-style-type: none"> • axis = 0: not supported • axis = 1 (channel dimension): $C \leq 11136$ • axis = 2 (height dimension): $W = 1, 0 < H < 16384$ • axis = 3 (width dimension): $0 < W < 16384$ <p>[Returns]</p> <p>Tensor of the identical type and shape as logits</p>
78	tf.nn.lrn	Local response normalization (LRN)	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: 4D Tensor of type float32 • depth_radius: 0D of type int, default to 5, indicating the half-width of the 1D normalization window • bias: (optional) float, default to 1, indicating the offset (usually positive to avoid dividing by 0) • alpha: (optional) float, default to 1, indicating the scale factor, usually positive • beta: (optional) float, default to 0.5, indicating an exponent • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • depth_radius is an odd number greater than 0. • When depth_radius is within [1,15], alpha > 0.00001 and beta > 0.01; Otherwise, alpha and beta are any values. When $C > 680$, depth_radius < 640. <p>[Returns]</p> <p>Tensor of the identical type as input</p>
79	tf.nn.max_pool	MaxPool	Same as tf.nn.avg_pool
80	tf.nn.relu	Relu	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor, must be one of the following types: float16, uint8, quint8

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • name: (optional) string <p>[Returns] Tensor of the identical type as features</p>
81	tf.nn.relu6	Relu6	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor of type float16 or float32 • name: (optional) string <p>[Returns] Tensor of the identical type as features</p>
82	tf.nn.selu	Selu	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor of type float32 • name: (optional) string <p>[Returns] Tensor of the identical type as features</p>
83	tf.nn.softmax	Softmax	<p>[Arguments]</p> <ul style="list-style-type: none"> • logits: non-null Tensor of type float32 • axis: dimension softmax would be performed on, default to -1, indicating the last dimension. The value cannot be greater than the rank of logits. • name: (optional) string • dim: (deprecated) equivalent to axis <p>[Restrictions] axis is of the range $[-rank, rank)$. Softmax can be performed on each of the four input dimensions (NCHW).</p> <ul style="list-style-type: none"> • axis = 0: $N \leq 28544$ • axis = 1 (channel dimension): $C \leq 11136, H * W < 65536$ • axis = 2 (height dimension): $W = 1, 0 < H < 16384$ • axis = 3 (width dimension): $0 < W < 16384$ <p>If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968.</p> <p>[Returns] Tensor of the identical type and shape as logits</p>
84	tf.nn.softplus	Softplus	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor of type float32

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • name: (optional) string <p>[Returns] Tensor of the identical type as features</p>
85	tf.nn.softsign	Softsign	<p>[Arguments]</p> <ul style="list-style-type: none"> • features: Tensor of type float32 • name: (optional) string <p>[Returns] Tensor of the identical type as features</p>
86	tf.pad	Pad PadV2 MirrorPad	<p>[Arguments]</p> <ul style="list-style-type: none"> • tensor: 4D Tensor of type float32 or int32 • paddings: constant Tensor of type int32 • mode: string, one of CONSTANT, REFLECT, or SYMMETRIC <p>If mode is CONSTANT and constant_values is 0, the C++ interface is a Pad. Otherwise, the C++ interface is PadV2. When mode is REFLECT or SYMMETRIC, the C++ interface is MirrorPad.</p> <ul style="list-style-type: none"> • name: (optional) string • constant_values: scalar pad value to use, of the identical type as tensor <p>[Restrictions] If mode is CONSTANT: $0 \leq \text{PAD} \leq 128$, $0 < W \leq 3000$</p> <p>[Returns] Tensor of the identical type as tensor</p>
87	tf.placeholder	Placeholder	<p>[Arguments]</p> <ul style="list-style-type: none"> • dtype: (mandatory) data type • shape: (mandatory) shape of the tensor • name: (optional) string <p>[Returns] Tensor</p>
88	tf.range	Range	<p>[Arguments]</p> <ul style="list-style-type: none"> • start: start constant scalar of type float32 or int32 • limit: end constant scalar of type float32 or int32 • delta: stride constant scalar of type float32 or int32 • dtype: data type of the resulting Tensor

No	Python API	C + + API	Boundary
			name: (optional) string [Returns] 1D Tensor
89	tf.realdiv	RealDiv	[Arguments] <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of type float32 • name: (optional) string [Returns] Tensor of the identical type as x
90	tf.math.reduce_sum	Sum	[Arguments] <ul style="list-style-type: none"> • input_tensor: Tensor • axis: dimensions to reduce, int32 • keepdims: bool, indicating whether to retain reduced dimensions • name: (optional) string • reduction_indices: (deprecated) string, equivalent to axis • keep_dims: (deprecated) equivalent to keepdims [Returns] Tensor of the identical type as tensor
91	tf.reshape	Reshape	[Arguments] <ul style="list-style-type: none"> • tensor: Tensor • shape: output shape, constant Tensor of type int64 or int32 name: (optional) string [Returns] Tensor of the identical type as input
92	tf.reverse (alias: tf.reverse_v2)	ReverseV2	[Arguments] <ul style="list-style-type: none"> • tensor: Tensor, must be one of the following types: int8, int16, int32, int64, float16, float32 • axis: dimensions to reverse, of type int32,int64 • name: (optional) string [Returns] Tensor of the identical type as tensor
93	tf.reverse_sequence	ReverseSequence	[Arguments]

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • input: Tensor • seq_lengths: 1D Tensor of type int32 or int64 • seq_axis: scalar of type integer • batch_axis: integer scalar, default to 0 • name: (optional) string <p>[Returns] Tensor of the identical type as input</p>
94	tf.scatter_nd (available since V310)	ScatterNd	<p>[Arguments]</p> <ul style="list-style-type: none"> • indices: Tensor of type int32 The value must not be negative, and the maximum value must be less than shape [0]. • updates: Tensor of type float32 • shape: 1D const of type int32 • name: (optional) string <p>[Returns] Tensor of the identical type as updates</p>
95	tf.shape	Shape	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: Tensor • name: (optional) string • out_type: data type for the output Tensor, either int32 (default) or int64 <p>[Returns] Tensor of the data type specified by out_type</p>
96	tf.sigmoid	Sigmoid	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor • name: (optional) string <p>[Returns] Tensor of the identical type as value</p>
97	tf.sign (available since V310)	Sign	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • name: (optional) string <p>[Returns] Tensor of the identical type as x</p>
98	tf.size	Size	<p>[Arguments]</p>

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • input: Tensor of type float32 • name: (optional) string out_type : data type for the output Tensor, default to int32 [Returns] Tensor of the data type specified by out_type
99	tf.slice	Slice	[Arguments] <ul style="list-style-type: none"> • input: Tensor • begin: Tensor of type int32 or int64 • size: Tensor of type int32 or int64 • name: (optional) string [Returns] Tensor of the identical type as input_
100	tf.space_to_batch_nd	SpaceToBatchND	[Arguments] <ul style="list-style-type: none"> • input: 4-D Tensor of type float32, with shape: input_shape = [batch] + spatial_shape + remaining_shape, where spatial_shape has M dimensions. • block_shape: 1D Tensor of type int32 or int64, with shape [M]. All values must be ≥ 1. • paddings: 2D Tensor of type int32 or int64, with shape [M, 2]. It is required that block_shape[i] divides the sum of (input_shape[i + 1] + pad_start + pad_end). [Restrictions] <ul style="list-style-type: none"> • The length of block_shape must be 2, and the length of paddings must be 4. • Element value of block_shape ≥ 1; Element value of paddings ≥ 0 • The padded H dimension is a multiple of block_shape[0], and the padded W dimension is a multiple of block_shape[1]. [Returns] Tensor of the identical type as input
101	tf.space_to_depth	SpaceToDepth	[Arguments] <ul style="list-style-type: none"> • input: Tensor of type float32 • block_size: scalar of type int32, ≥ 2 • data_format: (optional) string, NHWC (default) or NCHW • name: (optional) string

No	Python API	C + + API	Boundary
.			<p>[Returns]</p> <p>Tensor of the identical type as input</p>
102	tf.split	Split SplitV	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: Tensor • num_or_size_splits: If a scalar, then it must evenly divide value (the C++ interface is Split). If a 1D tensor, the value indicates the sum of sizes along the split axis (the C++ interface is SplitV). • axis: scalar of type int32, specifying the dimension along which to split <p>name: (optional) string</p> <p>[Returns]</p> <p>List of Tensor objects resulting from splitting</p>
103	tf.math.square	Square	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 or int32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
104	tf.squeeze	Squeeze	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: Tensor • axis: (optional) list of integers, specifying the dimensions to be squeezed, default to []. It is an error to squeeze a dimension that is not 1. • name: (optional) string • squeeze_dims: (deprecated) exclusive with axis <p>[Returns]</p> <p>Tensor, with the same data and type as input, but has one or more dimensions of size 1 removed.</p>
105	tf.stack	Pack	<p>[Arguments]</p> <ul style="list-style-type: none"> • values: list of Tensor objects with the same shape and type (float32 or int32) • axis: (mandatory) integer, indicating the axis to stack along, default to the first dimension • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as values</p>

No	Python API	C + + API	Boundary
106	tf.strided_slice	StridedSlice	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_: Tensor of type float32 • begin: 1D Tensor of type int32 • end: 1D Tensor of type int32 • strides: 1D Tensor of type int32 • begin_mask: scalar of type int32 • end_mask: scalar of type int32 • ellipsis_mask: scalar of type int32 • new_axis_mask: scalar of type int32 • shrink_axis_mask: scalar of type int32 • var: variable corresponding to input_ or None • name: (optional) string <p>[Restrictions]</p> <p>strides $\neq 0$</p> <p>[Restrictions]</p> <ul style="list-style-type: none"> • In the shrink_axis_mask scenario, only a positive mask is supported. • new_axis_mask is not supported. <p>[Returns]</p> <p>Tensor of the identical type as input_</p>
107	tf.subtract	Subtract	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of the identical type as x • name: (optional) string <p>[Restrictions]</p> <p>If the two inputs have inconsistent dimensions, broadcasting (that is, dimension padding) is performed.</p> <p>Broadcasting is supported only in the following scenarios:</p> <ul style="list-style-type: none"> • NHWC + NHWC, NHWC + scalar • NHWC + 1 1 1 1 • NHWC + W, HWC + W, HW + W (broadcasting along W) • NCHW + NH1C, HWC + H1C, HW + H1 • HWC + 1 WC (broadcasting along H) <p>Note: The input sequence of the two Tensors is not fixed.</p> <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor
108	tf.math.tanh	Tanh	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float16 or float32 • name: (optional) string <p>[Returns]</p> <p>Tensor of the identical type as x</p>
109	tf.tile	Tile	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: Tensor with at least one dimension • multiples: 1D constant Tensor of type int32. The length must be the same as that of input. • name: (optional) string <p>[Returns]</p> <p>Tensor</p>
110	tf.transpose	Transpose	<p>[Arguments]</p> <ul style="list-style-type: none"> • a: Tensor • perm: permutation of the dimensions of a • name: (optional) string • conjugate: (optional) bool, default to and must be False. <p>[Returns]</p> <p>Transposed Tensor</p>
111	tf.unstack	Unpack	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: Tensor to be unstacked, must be one of the following types: float32, int32, bool • num: integer of type int32, indicating the length of the dimension axis, default to None • axis: integer, indicating the axis to unstack along, default to 0 • name: (optional) string <p>[Returns]</p> <p>List of Tensor objects unstacked from value</p>
112	tf.where	Where	<p>[Arguments]</p> <ul style="list-style-type: none"> • condition: Tensor of type bool • x: None • y: None • name: (optional) string

No	Python API	C + + API	Boundary
			<p>[Returns]</p> <p>Tensor with 2-dimensional shape, where the first dimension represents the number of true elements</p>
113	tf.where	Select	<p>[Arguments]</p> <ul style="list-style-type: none"> • condition: Tensor of type bool • x: Tensor, must be one of the following types: float32, int32, uint8, bool • y: Tensor with the same shape and type as x • name: (optional) string <p>[Restrictions]</p> <ul style="list-style-type: none"> • condition, x, and y have the same shape. • If condition is 1-dimensional, x and y are with the same shape (rank ≥ 1), and the size of dimension 0 (shape[0]) of x and y is the same as the size of condition. <p>[Returns]</p> <p>Tensor of the identical shape and data type as x</p>
114	from tensorflow.pyt hon.ops import control_flow_ ops control_flow_ ops.switch (available since V310)	Switch	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor • pred: scalar of type bool • dtype: (optional) data type of the output Tensor • name: (optional) string <p>[Restrictions]</p> <p>This operator must be used in conjunction with operator Merge. For details, see operator Merge.</p> <p>[Returns]</p> <p>Tensor (output_false, output_true): If pred is true, data will be forwarded to output_true, otherwise it goes to output_false.</p>
115	from tensorflow.pyt hon.ops import control_flow_ ops control_flow_ ops.merge (available	Merge	<p>[Arguments]</p> <ul style="list-style-type: none"> • inputs: Tensors, at least one of which is valid • name: (optional) string <p>[Restrictions]</p> <p>This operator must be used in conjunction with Switch, and Merge cannot be the final output.</p> <p>[Returns]</p> <p>Tuple containing the selected tensors and their indexing</p>

No	Python API	C + + API	Boundary
	since V310)		
116	tf.matmul (available since V320)	BatchMatMul	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Left matrix, 3D–4D Tensor • y: Right matrix, 3D–4D Tensor, with same type and rank as x • adj_x: bool. If True, x is transposed before multiplication. • adj_y: bool. If True, y is transposed before multiplication. <p>[Restrictions]</p> <ul style="list-style-type: none"> • Tensor of type float32 • adj_x: Default to and must be false. • adj_y: Default to false. <p>[Returns]</p> <p>Tensor of the identical type as x and y</p>
117	tf.contrib.layers.layer_norm (available since V320)		<p>[Arguments]</p> <ul style="list-style-type: none"> • inputs: 2D–4D Tensor of type float in format NHWC • center: bool. If True, an offset is added to normalized inputs. Default to True. • scale: bool. If True, normalized inputs is multiplied by a scale factor. Default to True. • activation_fn: Default to None to skip the activation function and maintain a linear activation. • reuse: Whether or not the layer and its variables should be reused. Default to and must be None. • collection_collections: optional collections for the variables. Default to and must be None. • outputs_collections: Collections to add the outputs. Default to and must be None. • trainable: If True, variables are added to the graph collection GraphKeys. Default to and must be None. • begin_norm_axis: first normalization dimension. Only CHW data is normalized. For 4D input, only 1 is supported. For 2D or 3D input, only 0 is supported. • begin_params_axis: first parameter dimension. The scale and centering parameters apply to CHW data only. For 4D input, only 1 is supported. For 2D or 3D input, only 0 is supported. • scope: optional scope of variables. Default to and must be None.

No	Python API	C + + API	Boundary
.			<p>[Returns]</p> <p>Tensor of the identical shape and dtype as inputs, normalized CHW within the range of N of inputs.</p>
118	tf.stop_gradient (available since V320)	StopGradient	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: input Tensor <p>[Returns]</p> <p>Tensor of the identical type as input</p>
119	tf.contrib.layers.instance_norm (available since V320)		<p>[Arguments]</p> <ul style="list-style-type: none"> • inputs: 4D tensor of type float32 in format NCHW or NHWC • center: bool. If True, an offset is added to normalized inputs. Default to True. • scale: bool. If True, normalized inputs is multiplied by a scale factor. Default to True. • epsilon: small float added to variance to avoid dividing by zero. Default to 1e - 06. The minimum value is 1e-7. • activation_fn: Default to None to skip the activation function and maintain a linear activation. • param_initializers: optional initializers for beta, gamma, mean and variance. Default to None. • reuse: Whether or not the layer and its variables should be reused. Default to and must be None. • collection_collections: optional collections for the variables. Default to and must be None. • outputs_collections: Collections to add the outputs. Default to and must be None. • trainable: If True, variables are added to the graph collection GraphKeys. Default to True. • data_format: either NHWC (default) or NCHW • scope: optional scope of variables. Default to and must be None. <p>[Returns]</p> <p>Tensor of the identical shape and dtype as inputs, normalized HW within the range of NC of inputs.</p>
120	tf.random.normal (available since V320)	RandomNormal	<p>[Arguments]</p> <ul style="list-style-type: none"> • shape: constant, 1D tensor of type int32, shape of the output tensor • mean: 0D scalar of type float16, mean of the normal

No	Python API	C + + API	Boundary
.			<p>distribution.</p> <ul style="list-style-type: none"> • stddev: 0D scalar of type float16, standard deviation of the normal distribution. • seed: int32, random seed for the distribution. This parameter can be set by calling tf.random.set_random_seed. In this version, the seed value is ignored and 0 is used. • seed2: int32, random seed for the distribution. This parameter is derived from the seed parameter of tf.random.random_normal. If this parameter is not set or is set to 0, the computation result is different each time. In this version, the seed2 value is ignored and 0 is used. <p>[Returns] Tensor of type float16</p>
121	tf.random.shuffle (available since V320)	RandomShuffle	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: constant or non-constant, Tensor of type float16, float32, double, int8, int16, int32, int64, uint8, uint16, bool • seed: int32, random seed for the distribution. This parameter can be set by calling tf.random.set_random_seed. In this version, the seed value is ignored and 0 is used. • seed2: int32, random seed for the distribution. This parameter is derived from the seed parameter of tf.random.random_shuffle. If this parameter is not set or is set to 0, the computation result is different each time. In this version, the seed2 value is ignored and 0 is used. <p>[Returns] Tensor of the identical type as value</p>
122	tf.random.uniform (available since V320)	RandomUniformInt	<p>[Arguments]</p> <ul style="list-style-type: none"> • shape: 1D constant or Shape operator of type int32 • minval: 0D scalar of type int32, lower bound on the range of random values to generate (inclusive) • maxval: 0D scalar of type int32, upper bound on the range of random values to generate (inclusive) • dtype: int32, type of the output tensor • seed: int32, random seed for the distribution. This parameter can be set by calling tf.random.set_random_seed. In this version, the seed value is ignored and 0 is used.

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • seed2: int32, random seed for the distribution. This parameter is derived from the seed parameter of tf.random.uniform. If this parameter is not set or is set to 0, the computation result is different each time. In this version, the seed2 value is ignored and 0 is used. <p>[Returns] Tensor of type int32</p>
123	tf.random.uniform (available since V320)	RandomUniform	<p>[Arguments]</p> <ul style="list-style-type: none"> • shape: 1D constant or Shape operator of type int32 • minval: 0D scalar of type float32, lower bound on the range of random values to generate (inclusive) • maxval: 0D scalar of type float32, upper bound on the range of random values to generate (inclusive) • dtype: float32, type of the output tensor • seed: int32, random seed for the distribution. This parameter can be set by calling tf.random.set_random_seed. In this version, the seed value is ignored and 0 is used. • seed2: int32, random seed for the distribution. This parameter is derived from the seed parameter of tf.random.uniform. If this parameter is not set or is set to 0, the computation result is different each time. In this version, the seed2 value is ignored and 0 is used. <p>[Returns] Tensor of type float32</p>
124	tf.math.argmax (available since V320)	ArgMin	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: constant or non-constant, Tensor of type float32, uint8, or int32 • axis: constant, Tensor of type int32 • dimension: (deprecated) equivalent to axis • out_type: Tensor of type int32 • name: (optional) string <p>[Returns] Tensor of type out_type</p>
125	tf.rank (available since V320)	Rank	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: constant or non-constant, Tensor, must be one of the following data types: int32, float32, uint8, bool • name: (optional) string

No	Python API	C + + API	Boundary
.			[Returns] Tensor of type int32
126	tf.truncatemod (available since V320)	Truncatemod	[Arguments] <ul style="list-style-type: none"> • x: constant or non-constant, Tensor of type int32 or float32 • y: Tensor of type int32 or float32 • name: (optional) string [Returns] Tensor of the identical shape and type as x
127	tf.math.unsorted_segment_sum (available since V320)	UnsortedSegmentSum	[Arguments] <ul style="list-style-type: none"> • data: constant or non-constant, Tensor of type int32, float32, or uint8 • segment_ids: (mandatory) Tensor of type int32, with the identical shape as data, specifying the result as out[i] • num_segments: (mandatory) constant, 0D Tensor of type int32, specifying the length of segment_ids • name: (optional) string [Restrictions] num_segments is greater than or equal to the number of segment_id groups. [Returns] Tensor of the identical type as data
128	tf.math.cumsum (available since V320)	Cumsum	[Arguments] <ul style="list-style-type: none"> • x: constant or non-constant, Tensor of type float32, uint8, int32 • rank: Tensor of type int32, Must be in the range $[-rank(x), rank(x)]$. Default to 0. • exclusive: (optional) bool. If true, the first output element starts from 0. If false, the first element of the input is identical to the first element of the output. Default to false. • reverse: (optional) bool. If true, the cumsum is performed in the opposite direction. Default to false. • name: (optional) string [Returns] Tensor of the identical type as x
12	tf.math.cumpr	Cumprod	[Arguments]

No	Python API	C + + API	Boundary
9	od (available since V320)		<ul style="list-style-type: none"> • x: Tensor of type float32, uint32, or uint8 • rank: Tensor of type int32, Must be in the range $[-rank(x), rank(x)]$. Default to 0. • exclusive: (optional) bool. If true, the first output element starts from 0. If false, the first element of the input is identical to the first element of the output. Default to false. • reverse: (optional) bool. If true, the cumprod is performed in the opposite direction. Default to false. • name: (optional) string <p>[Returns] Tensor of the identical type as x</p>
130	tf.nn.conv1d (available since V320)		<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 3D Tensor of type float32 • filters: 3D Tensor of type float32 • stride: int • padding: either VALID or SAME • use_cudnn_on_gpu: (optional) bool, default to true • data_format: (optional) either NWC (default) or NCW • name: (optional) string <p>[Returns] Tensor of the identical type as value</p>
131	tf.nn.atrous_conv2d (available since V320)		<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor of type float32 in NHWC format • filters: 4D Tensor of type float32, with the same type as value and shape as value • rate: int32, stride with which we sample input values across the height and width dimensions • padding: either VALID or SAME • name: (optional) string <p>[Returns] Tensor of the identical type as value</p>
132	tf.math.reduce_any (available since V320)	Any	<p>[Arguments]</p> <ul style="list-style-type: none"> • input_tensor: Tensor of type bool • axis: dimensions to reduce. Must be in the range $[-rank(input_tensor), rank(input_tensor)]$.

No	Python API	C + + API	Boundary
.			<ul style="list-style-type: none"> • keepdims: scalar of type bool, default to false. • name: (optional) string • reduction_indices: (deprecated) equivalent to axis • keep_dims: (deprecated) equivalent to keepdims <p>[Returns] Tensor of type bool</p>
133	tf.math.logical_xor (available since V320)		<p>[Arguments]</p> <ul style="list-style-type: none"> • x: constant or non-constant, Tensor of type bool • y: constant or non-constant, Tensor of type bool • name: (optional) string <p>[Restrictions] Bidirectional broadcast is not supported.</p> <p>[Returns] Tensor of the identical type as the inputs</p>
134	tf.nn.fractional_max_pool (available since V320)	FractionalMaxPool	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor of type float32, int32, or int64, in the NHWC format • pooling_ratio: list of float32 that has length 4. Indicates the length/width ratio of the pooling window. The ratio must be greater than or equal to 1.0, and ratio[0] and ratio[3] must be 1.0. • pseudo_random: (optional) bool. If True, rowSeq and colSeq are generated in a pseudo-random fashion, otherwise, in a random fashion. Default to False. • overlapping: (optional) bool. If True, it means when pooling, the values at the boundary of adjacent pooling cells are used by both cells. Default to False. • deterministic: (optional) bool. If True, generated rowSeq and colSeq are determined. Default to False. • seed: int32, random number generator of the output tensor • seed2: int32, random number of the output tensor <p>[Restrictions] If deterministic is set to false, seed and seed2 must be both 0. In this case, a true random number is generated, and the results are different each time. If deterministic is set to true, seed and seed2 must not both be 0 at the same time. In this case, a pseudo-random number is generated, and the results</p>

No	Python API	C + + API	Boundary
.			<p>are the same each time.</p> <p>[Returns]</p> <ul style="list-style-type: none"> • y: Tensor of the identical type as value • row_pooling_sequence: Tensor of type int64 • col_pooling_sequence: Tensor of type int64
135	tf.nn.fractional_avg_pool (available since V320)	FractionalAvgPool	<p>[Arguments]</p> <ul style="list-style-type: none"> • value: 4D Tensor of type float32, int32, or int64, in the NHWC format • pooling_ratio: list of float32 that has length 4. Indicates the length/width ratio of the pooling window. The ratio must be greater than or equal to 1.0, and ratio[0] and ratio[3] must be 1.0. • pseudo_random: (optional) bool. If True, rowSeq and colSeq are generated in a pseudo-random fashion, otherwise, in a random fashion. Default to False. • overlapping: (optional) bool. If True, it means when pooling, the values at the boundary of adjacent pooling cells are used by both cells. Default to False. • deterministic: (optional) bool. If True, generated rowSeq and colSeq are determined. Default to False. • seed: int32, random number generator of the output tensor • seed2: int32, random number of the output tensor <p>[Restrictions]</p> <p>If deterministic is set to false, seed and seed2 must be both 0. In this case, a true random number is generated, and the results are different each time. If deterministic is set to true, seed and seed2 must not both be 0 at the same time. In this case, a pseudo-random number is generated, and the results are the same each time.</p> <p>[Returns]</p> <ul style="list-style-type: none"> • y: Tensor of the identical type as value • row_pooling_sequence: Tensor of type int64 • col_pooling_sequence: Tensor of type int64
136	tf.math.not_equal (available since V320)	NotEqual	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of type float32 <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor of type bool
137	tf.math.less_equal (available since V320)	LessEqual	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float32 • y: Tensor of type float32 <p>[Returns]</p> <p>Tensor of type bool</p>
138	tf.quantization.quantize (available since V320)	QuantizeV2	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: constant or non-constant, Tensor of type float32 • min_range: constant, Tensor of type float32, specifying the minimum value of the quantization range. The value must be less than or equal to 0. • max_rang: constant, Tensor of type float32, specifying the maximum value of the quantization range • T: uint8, destination data type • mode: (optional) string, either MIN_COMBINED, MIN_FIRST, or SCALED. Must be MIN_COMBINED. • round_mode: (optional) string, either HALF_AWAY_FROM_ZERO or HALF_TO_EVEN. Must be HALF_AWAY_FROM_ZERO. <p>[Returns]</p> <p>Tensor of type uint8</p>
139	tf.quantization.dequantize (available since V320)	Dequantize	<p>[Arguments]</p> <ul style="list-style-type: none"> • input: constant or non-constant, Tensor of type quint8 • min_range: constant, Tensor of type float32, specifying the minimum value of the quantization range. The value must be less than or equal to 0. • max_rang: constant, Tensor of type float32, specifying the maximum value of the quantization range • mode: (optional) string, either MIN_COMBINED, MIN_FIRST, or SCALED. Must be MIN_COMBINED. <p>[Returns]</p> <p>Tensor of type float32</p>
140	tf.math.floor_div (available since V320)	FloorDiv	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type int32, float, or uint8 • y: Tensor of type int32, float, or uint8 <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor of the identical shape and data type as x
14 1	tf.quantization.fake_quant_with_min_max_vars_per_channel (available since V320)	FakeQuantWithMinMaxVarsPerChannel	<p>[Arguments]</p> <ul style="list-style-type: none"> • x: constant or non-constant, 1D Tensor of type float32 • min: constant 1D Tensor of type float32, minimum input value • max: constant 1D Tensor of type float32, maximum input value • num_bits: (optional) int, bit width of quantization. Default to 8. • narrow_range: (optional) bool, quantization range identifier. Default to false. <p>[Returns]</p> <p>Tensor of the identical shape and data type as x</p>
14 2	tf.one_hot (available since V320)	OneHot	<p>[Arguments]</p> <ul style="list-style-type: none"> • indices: constant or non-constant, Tensor of indices of type uint8 or int32 • axis: constant, Tensor of type int32, depth of the one hot dimension • on_value: (optional) constant or non-constant, Tensor of type uint8, int32, float, or bool. Default to 1. • off_value: (optional) consistent with dtype of on_value. Default to 0. • axis: (optional) int, default to -1 • dtype: destination data type <p>[Returns]</p> <p>One-hot tensor, of one of the following data types: uint8, int32, float, double, bool</p>
14 3	tf.math.segment_max (available since V320)	SegmentMax	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor of type float32 • segment_ids: constant, 1D tensor of type int32, sorted by ID <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of segment_ids elements is equal to the size of data's dimension 0. • segment_ids does not support negative indexes and is sorted in ascending order starting from 0. <p>[Returns]</p>

No	Python API	C + + API	Boundary
			Tensor
144	tf.math.segment_min (available since V320)	SegmentMin	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor of type float32 • segment_ids: constant, 1D tensor of type int32, sorted by ID <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of segment_ids elements is equal to the size of data's dimension 0. • segment_ids does not support negative indexes and is sorted in ascending order starting from 0. <p>[Returns]</p> <p>Tensor</p>
145	tf.math.segment_mean (available since V320)	SegmentMean	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor of type float32 • segment_ids: constant, 1D tensor of type int32, sorted by ID <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of segment_ids elements is equal to the size of data's dimension 0. • segment_ids does not support negative indexes and is sorted in ascending order starting from 0. <p>[Returns]</p> <p>Tensor</p>
146	tf.math.segment_prod (available since V320)	SegmentProd	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor of type float32 • segment_ids: constant, 1D tensor of type int32, sorted by ID <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of segment_ids elements is equal to the size of data's dimension 0. • segment_ids does not support negative indexes and is sorted in ascending order starting from 0. <p>[Returns]</p> <p>Tensor</p>
147	tf.math.segment_sum	SegmentSum	<p>[Arguments]</p> <ul style="list-style-type: none"> • data: Tensor of type float32

No	Python API	C + + API	Boundary
.	(available since V320)		<ul style="list-style-type: none"> • segment_ids: constant, 1D tensor of type int32, sorted by ID <p>[Restrictions]</p> <ul style="list-style-type: none"> • The number of segment_ids elements is equal to the size of data's dimension 0. • segment_ids does not support negative indexes and is sorted in ascending order starting from 0. <p>[Returns]</p> <p>Tensor</p>
148	tf.zeros_like (available since V310)	ZerosLike	<p>[Arguments]</p> <p>x: constant or 0D–4D Tensor of type float32</p> <p>[Returns]</p> <p>Tensor of the identical shape and data type as x</p>
149	tf.identity (available since V310)	Identity	<p>[Arguments]</p> <p>x: Tensor</p> <p>[Returns]</p> <p>Tensor of the identical shape and data type as x</p>
150	tf.Assert (available since V310)	Assert	<p>[Arguments]</p> <ul style="list-style-type: none"> • condition: condition to evaluate • data: list of tensors, indicating the tensors to print out when condition is false • summarize: (optional) indicating this many entries of each tensor (data) are printed. Default to None. <p>[Restrictions]</p> <p>When building a model, a dependency is required to ensure the execution of this operator, which usually used in conjunction with tf.control_dependencies([assert_op]).</p> <p>[Returns]</p> <p>Operation that, when executed, raises a tf.errors.InvalidArgumentError if condition is false.</p>
151	tf.keras.layers.PReLU (available since V310)		<p>[Arguments]</p> <ul style="list-style-type: none"> • x: Tensor of type float • slope: trained slope <p>[Returns]</p> <p>Tensor of the identical shape and data type as x</p>

No	Python API	C + + API	Boundary
152	$GELU(x) = 0.5x(1 + \tanh[\frac{2}{\pi}((\frac{x}{0.044715x^3})^2 + 1)])$ (available since V320)		[Arguments] <ul style="list-style-type: none"> • features: Tensor of type float32 • name: (optional) string [Returns] <p>Tensor of the identical type as features</p>
153	tf.nn.bidirectional_dynamic_rnn (available since V320)		[Inputs] <ul style="list-style-type: none"> • cell_fw: forward computation cell, generated in tf.nn.rnn_cell.BasicLSTMCell mode • cell_bw: backward computation cell, generated in tf.nn.rnn_cell.BasicLSTMCell mode • inputs: RNN inputs. If time_major is False (default), this must be a tensor with shape [batch_size, max_time, depth], or a nested tuple of such elements. If time_major is True, this must be a tensor of shape [max_time, batch_size, depth], or a nested tuple of such elements. • sequence_length: (optional) non-constant, vector of type int32, with shape [batch_size], containing the actual lengths for each of the sequences in the batch. If not provided, all batch entries are assumed to be full sequences (max_time). Use this parameter when the sequences have different lengths. • initial_state_fw: (optional) initial state for the forward RNN. Must be none. • initial_state_bw: (optional) initial state for the backward RNN. Must be none. • dtype: (optional) data type for the initial states and expected output. Must be fp32. • time_major: shape format of the inputs and outputs Tensors. If False, these Tensors must be shaped [batch_size, max_time, depth] or a nested tuple of such elements. If True, these Tensors must be shaped TBX. • scope: name of the created subgraph, default to bidirectional_rnn [Restrictions] <ul style="list-style-type: none"> • All the parameters of the basis forward and backward RNN cells must consistent. That is, all the parameters including the cell names, activation functions, and num_units must be consistent. • Only BasicLSTMCell is supported. The activation functions

No	Python API	C + + API	Boundary
.			<p>include Tanh, Sigmoid, ReLU, and ReLU6.</p> <ul style="list-style-type: none"> The initial_state_fw and initial_state_bw parameters for dynamic RNN must be none. state_is_tuple must be True. The value of num_units of BasicLSTMCell must be an integer multiple of 16. <p>[Returns]</p> <ul style="list-style-type: none"> Forward and the backward RNN output Tensors The forward cell state, forward hidden state, and backward cell state, backward hidden state are output separately.
154	tf.nn.static_bidirectional_rnn (available since V320)		<p>[Inputs]</p> <ul style="list-style-type: none"> cell_fw: (mandatory) forward computation cell, generated in tf.nn.rnn_cell.BasicLSTMCell mode. cell_bw: (mandatory) backward computation cell, generated in tf.nn.rnn_cell.BasicLSTMCell mode. inputs: (mandatory) length T (max_time) list of RNN inputs, each a tensor of shape [batch_size, input_size], or a nested tuple of such elements. initial_state_fw: (optional) initial state for the forward RNN. Must be none. initial_state_bw: (optional) initial state for the backward RNN. Must be none. dtype: (Optional) data type for the initial state. Must be fp32. sequence_length: (optional) non-constant, vector of type int32, with shape [batch_size], containing the actual lengths for each of the sequences in the batch. If not provided, all batch entries are assumed to be full sequences (max_time). Use this parameter when the sequences have different lengths. scope: (Optional) name of the created subgraph, default to bidirectional_rnn <p>[Restrictions]</p> <ul style="list-style-type: none"> All the parameters of the basis forward and backward RNN cells must consistent. That is, all the parameters including the cell names, activation functions, and num_units must be consistent. Only BasicLSTMCell is supported. The activation functions include Tanh, Sigmoid, ReLU, and ReLU6.

No.	Python API	C + + API	Boundary
			<ul style="list-style-type: none"> The initial_state_fw and initial_state_bw parameters for static RNN must be none. state_is_tuple must be True. The value of num_units of BasicLSTMCell must be an integer multiple of 16. <p>[Returns]</p> <ul style="list-style-type: none"> Length T list of stacked outputs The forward cell state, forward hidden state, and backward cell state, backward hidden state are output separately.

2.4 AndroidNN Operator Boundaries

No.	Operation	Description	Boundary
1	ANEURALNE TWORKS_ABS (available since V310)	Computes the absolute value of a tensor, element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor <p>[Restrictions]</p> <p>The last dimension of input 0 must not exceed 400.</p> <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
2	ANEURALNE TWORKS_ADD	Adds two tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_QUANT8_ASYMM TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor Input 1: Tensor of identical OperandCode as input 0 Input 2: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values

No.	Operation	Description	Boundary
			<p>FuseCode{ ANEURALNETWORKS_FUSED_NONE = 0, ANEURALNETWORKS_FUSED_RELU = 1, ANEURALNETWORKS_FUSED_RELU1 = 2, ANEURALNETWORKS_FUSED_RELU6 = 3 }</p> <p>[Restrictions] None</p> <p>[Returns] Output 0: Tensor of identical OperandCode as input 0</p>
3	ANEURALNETWORKS_ARGMAX (available since V310)	Returns the index of the largest element along an axis.	<p>[Inputs] • Supported tensor OperandCode: Tensor_FLOAT32 (only in the relaxed scenario) Tensor_FLOAT16 (available since API 29) Supported tensor rank: up to 4 • Input 0: n-D tensor, where N is within [1, 4] • Input 1: scalar of type Tensor_INT32</p> <p>[Restrictions] None</p> <p>[Returns] Output 0: (N - 1)D tensor of type Tensor_INT32</p>
4	ANEURALNETWORKS_ARGMIN (available since V310)	Returns the index of the smallest element along an axis.	<p>[Inputs] • Supported tensor OperandCode: Tensor_FLOAT32 (only in the relaxed scenario) Tensor_FLOAT16 (available since API 29) Tensor_QUANT8_ASYMM Supported tensor rank: up to 4 • Input 0: n-D tensor, where N is within [1, 4] • Input 1: scalar of type Tensor_INT32</p> <p>[Restrictions] None</p> <p>[Returns] Output 0: (N - 1)D tensor of type Tensor_INT32</p>
5	ANEURALNETWORKS_AVERAGE_POOL_2D	Average pooling	<p>[Inputs] • Supported tensor OperandCode: Tensor_FLOAT32 (only in the relaxed scenario)</p>

No.	Operation	Description	Boundary
			<p>TENSOR_FLOAT16 (available since API 29)</p> <p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: 4</p> <p>NHWC data layout supported</p> <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D Tensor, with shape [batches, height, width, depth_in] • Input 1: scalar of type INT32, specifying the padding on the left, in the width dimension, $0 \leq \text{Pad} < 256$ • Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, $0 \leq \text{Pad} < 256$ • Input 3: scalar of type INT32, specifying the padding on the top, in the height dimension, $0 \leq \text{Pad} < 256$ • Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, $0 \leq \text{Pad} < 256$ • Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, $0 < \text{Stride} < 64$ • Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, $0 < \text{Stride} < 64$ • Input 7: scalar of type INT32, specifying the filter width • Input 8: scalar of type INT32, specifying the filter height • Input 9: scalar of type INT32, specifying the activation to invoke on the result • Input 10: (optional) scalar of type BOOL, specifying the input and output data formats. Only the default format NHWC is supported. (available since API 29) <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D Tensor, with shape [batches, height, width, depth], specifying the input • Input 1: scalar of type INT32, specifying the padding scheme, which must be one of the PaddingCode values (SAME or VALID) <p style="padding-left: 40px;">PaddingCode{ ANEURALNETWORKS_PADDING_SAME = 0, ANEURALNETWORKS_PADDING_VALID = 1 }</p> <ul style="list-style-type: none"> • Input 2: scalar of type INT32, specifying the stride when walking through input in the width dimension, $\text{strideW} < 64$

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 3: scalar of type INT32, specifying the stride when walking through input in the height dimension, strideH < 64 Input 4: scalar of type INT32, specifying the filter width Input 5: scalar of type INT32, specifying the filter height Input 6: scalar of type INT32, specifying the activation to invoke on the result Input 7: (optional) scalar of type BOOL, specifying the input and output data formats. Only the default format NHWC is supported. (available since API 29) <p>[Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth]</p> <p>[Restrictions] <ul style="list-style-type: none"> KernelH < 256, kernelW < 256 If H and W of the output tensor are 1: input H * input W < 65536 </p>
6	ANEURALNE WORKS_BATCH_TO_SPACE_ND	BatchToSpace for n-D tensors	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 Input 0: 4D Tensor Input 1: 1D Tensor of type TENSOR_INT32, specifying the block sizes for each spatial dimension of the input Tensor. All values must be ≥ 1. Input 2: (optional) BOOL. Only the default format NHWC is supported. (available since API 29) <p>[Restrictions] The product of the dimensions of input 0 must be less than 500.</p> <p>[Returns] Output 0: Tensor of identical OperandCode as input 0</p>
7	ANEURALNE WORKS_CAST (available since V310)	Casts a tensor to a new type.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29)

No.	Operation	Description	Boundary
			<p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: n-D tensor, where N is within [1, 4] <p>[Returns]</p> <p>Output 0: n-D tensor</p>
8	ANEURALNE WORKS_CO NCATENATI ON	Concatenates the input along the given dimension.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Inputs 0 to n-1: one list of n input Tensors, with shape [D0, D1, ..., Daxis(i), ..., Dm]. TENSOR_QUANT8_ASYMM input Tensors must have the same scale and zeroPoint as the output Tensor. Input n: scalar of type INT32, specifying the concatenation axis <p>[Restrictions]</p> <p>The number of dimensions of the input tensors must match, and all dimensions except axis must be equal.</p> <p>The range of the input Tensor count is [2, 32].</p> <p>Inputs 0-(n - 1) do not accept constant inputs.</p> <p>Only QUANT8 inputs are accepted. axis ≠ 2</p> <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as the input Tensors, with shape [D0, D1, ..., sum(Daxis(i)), ..., Dm]</p>
9	ANEURALNE WORKS_CO NV_2D	Convolves the input.	<p>[Inputs]</p> <p>Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 NHWC data layout supported</p> <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, with shape [batches, height, width, depth_in] Input 1: 4D Tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter, 0 < FilterSize <

No.	Operation	Description	Boundary
			<p>256</p> <ul style="list-style-type: none"> • Input 2: 1D Tensor, with shape [depth_out], specifying the bias. For a TENSOR_FLOAT32 Tensor, the bias must be of the same type. For a TENSOR_QUANT8_ASYMM Tensor, the bias should also be of TENSOR_INT32, with zeroPoint = 0 and bias_scale == input_scale * filter_scale. • Input 3: scalar of type INT32, specifying the padding on the left, in the width dimension, $0 \leq \text{Pad} < 256$ • Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension, $0 \leq \text{Pad} < 256$ • Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension, $0 \leq \text{Pad} < 256$ • Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension, $0 \leq \text{Pad} < 256$ • Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension, $0 < \text{Stride} < 64$ • Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension, $0 < \text{Stride} < 64$ • Input 9: scalar of type INT32, specifying the activation to invoke on the result • Input 10: (optional) scalar of type BOOL, default to false. Set to true to specify NCHW data layout for input 0 and output 0. • Input 11: (optional) scalar of type INT32, specifying the dilation factor for width. Default to and must be 1. If this input is set, input 12 must be specified as well. • Input 12: (optional) scalar of type INT32, specifying the dilation factor for height. Default to and must be 1. If this input is set, input 11 must be specified as well. <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D Tensor, with shape [batches, height, width, depth_in] • Input 1: 4D Tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter, $0 < \text{FilterSize} < 256$ • Input 2: 1D Tensor, with shape [depth_out], specifying the bias. For a TENSOR_FLOAT32 Tensor, the bias must be of the same type. For a TENSOR_QUANT8_ASYMM Tensor, the bias should be of TENSOR_INT32, with zeroPoint = 0 and bias_scale == input_scale * filter_scale.

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> • Input 3: scalar of type INT32, specifying the padding scheme, which must be one of the PaddingCode values (SAME or VALID) • Input 4: scalar of type INT32, specifying the stride when walking through input in the width dimension, $0 < \text{Stride} < 64$ • Input 5: scalar of type INT32, specifying the stride when walking through input in the height dimension, $0 < \text{Stride} < 64$ • Input 6: scalar of type INT32, specifying the activation to invoke on the result • Input 7: (optional) scalar of type BOOL, default to false. Set to true to specify NCHW data layout for input 0 and output 0. • Input 8: (optional) scalar of type INT32, specifying the dilation factor for width. Default to and must be 1. If this input is set, input 9 must be specified as well. • Input 9: (optional) scalar of type INT32, specifying the dilation factor for height. Default to and must be 1. If this input is set, input 8 must be specified as well. <p>[Restrictions]</p> <p>When the input type is QUANT8, the data layout of input 0 and output 0 must be NHWC.</p> <p>[Returns]</p> <p>Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth_out]</p>
10	ANEURALNETWORKS_DEPTH_TO_SPACE	Rearranges data from depth into blocks of spatial data.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 NHWC data layout supported • Input 0: 4D Tensor, with shape [batches, height, width, depth_in], specifying the input • Input 1: scalar of type int32, block_size, specifying block_size. block_size ≥ 1 and must be a divisor of the height and width of the input Tensor. • Input 2: (optional), BOOL. Only the default format NHWC is supported. (available since API 29) <p>[Restrictions]</p>

No.	Operation	Description	Boundary
			<p>None</p> <p>[Returns]</p> <p>Output 0: 4D Tensor, with shape [batch, height * block_size, width * block_size, depth/(block_size * block_size)]</p>
11	ANEURALNETWORKS_DEPTHWISE_CONV_2D	Depthwise convolution	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 NHWC data layout supported <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, with shape [batches, height, width, depth_in] Input 1: 4D Tensor, with shape [1, filter_height, filter_width, depth_out], filter, $0 < \text{FilterSize} < 256$ Input 2: 1D Tensor, with shape [depth_out], specifying the bias. For a TENSOR_FLOAT32 Tensor, the bias must be of the same type. For a TENSOR_QUANT8_ASYMM Tensor, the bias should also be of TENSOR_INT32, with zeroPoint = 0 and bias_scale == input_scale * filter_scale. Input 3: scalar of type INT32, specifying the padding on the left, in the width dimension, $0 \leq \text{Pad} < 256$ Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension, $0 \leq \text{Pad} < 256$ Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension, $0 \leq \text{Pad} < 256$ Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension, $0 \leq \text{Pad} < 256$ Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension, $0 < \text{Stride} < 64$ Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension, $0 < \text{Stride} < 64$ Input 9: scalar of type INT32, specifying the depthwise multiplier Input 10: scalar of type INT32, specifying the activation to invoke on the result Input 11: (optional) scalar of type BOOL, default to false.

No.	Operation	Description	Boundary
			<p>Set to true to specify NCHW data layout for input 0 and output 0.</p> <ul style="list-style-type: none"> • Input 12: (optional) scalar of type INT32, specifying the dilation factor for width. Default to and must be 1. If this input is set, input 13 must be specified as well. • Input 13: (optional) scalar of type INT32, specifying the dilation factor for height. Default to and must be 1. If this input is set, input 12 must be specified as well. <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D Tensor, with shape [batches, height, width, depth_in], specifying the input • Input 1: 4D Tensor, with shape [1, filter_height, filter_width, depth_out], filter, $0 < \text{FilterSize} < 256$ • Input 2: 1D Tensor, with shape [depth_out], specifying the bias. For a TENSOR_FLOAT32 Tensor, the bias must be of the same type. For a TENSOR_QUANT8_ASYMM Tensor, the bias should be of TENSOR_INT32, with zeroPoint = 0 and bias_scale == input_scale * filter_scale. • Input 3: scalar of type INT32, specifying the padding scheme, which must be one of the PaddingCode values (SAME or VALID) • Input 4: scalar of type INT32, specifying the stride when walking through input in the width dimension, $0 < \text{Stride} < 64$ • Input 5: scalar of type INT32, specifying the stride when walking through input in the height dimension, $0 < \text{Stride} < 64$ • Input 6: scalar of type INT32, specifying the depthwise multiplier • Input 7: scalar of type INT32, specifying the activation to invoke on the result • Input 8: (optional) scalar of type BOOL, default to false. Set to true to specify NCHW data layout for input 0 and output 0. • Input 9: (optional) scalar of type INT32, specifying the dilation factor for width. Default to and must be 1. If this input is set, input 10 must be specified as well. • Input 10: (optional) scalar of type INT32, specifying the dilation factor for height. Default to and must be 1. If this input is set, input 9 must be specified as well. <p>[Restrictions]</p>

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> • filterN = inputC = group • $\text{StrideW} \leq (\text{inputW} + \text{padW}) - ((\text{filterW} - 1) * \text{dilationW}) + 1$ • When the input type is QUANT8, the data layout of input 0 and output 0 must be NHWC. <p>[Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth_out].</p>
12	ANEURALNE TWORKS_DE QUANTIZE	Dequantizes the input tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_QUANT8_ASYMM • Supported output tensor OperandCode: TENSOR_FLOAT32 TENSOR_FLOAT16 Supported tensor rank: up to 4 • Input 0: tensor <p>[Returns] Output 0: Tensor of type TENSOR_FLOAT32 and the identical shape as input 0</p>
13	ANEURALNE TWORKS_DIV	Divides tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 • Input 0: n-D Tensor, where N is within [1, 4] • Input 1: Tensor of identical OperandCode as input 0 • Input 2: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values <p>[Restrictions] None</p> <p>[Returns] Output 0: Tensor of identical OperandCode as input 0</p>
14	ANEURALNE TWORKS_QUANTIZED_16BIT_LSTM	A version of quantized LSTM, using 16 bit	<p>[Inputs]</p> <ul style="list-style-type: none"> • Input 0: 2D tensor with shape [batch_size, input_size]. • Input 1: constant, 2D tensor with shape [nums_units, input_size], specifying the input-to-input weights.

No.	Operation	Description	Boundary
	(available since V320)	quantization for internal state.	<p>nums_units corresponds to the number of cell units.</p> <ul style="list-style-type: none"> • Input 2: constant, 2D tensor with shape [nums_units, input_size], specifying the input-to-forget weights • Input 3: constant, 2D tensor with shape [nums_units, input_size], specifying the input-to-cell weights • Input 4: constant, 2D tensor with shape [nums_units, input_size], specifying the input-to-output weights • Input 5: constant, 2D tensor with shape [nums_units, output_size], specifying the recurrent-to-input weights. <p>output_size corresponds to either the number of cell units (num_units), or the second dimension of projection_weights, if defined.</p> <ul style="list-style-type: none"> • Input 6: constant, 2D tensor with shape [nums_units, output_size], specifying the recurrent-to-forget weights • Input 7: constant, 2D tensor with shape [nums_units, output_size], specifying the recurrent-to-cell weights • Input 8: constant, 2D tensor with shape [nums_units, output_size], specifying the recurrent-to-output weights • Input 9: constant, 1D tensor with shape [nums_units], specifying the input gate bias • Input 10: constant, 1D tensor with shape [nums_units], specifying the forget gate bias • Input 11: constant, 1D tensor with shape [nums_units], specifying the cell gate bias • Input 12: constant, 1D tensor output linear inputs, with shape [num_units] • Input 13: 2D tensor with shape [numBatches, outputSize], specifying the cell state from the previous time step of the LSTM cell • Input 14: 2D tensor with shape [numBatches, outputSize], specifying the output from the previous time step of the LSTM cell <p>[Restrictions]</p> <ul style="list-style-type: none"> • Inputs 1–12 accept constants only. • Input 13 must be of type UINT16, and the residual inputs must be of type UINT8. Output 0 must be of type UINT16, and output 1 must be of type UINT8. <p>[Returns]</p> <ul style="list-style-type: none"> • Output 0: 2D tensor with shape [numBatches, outputSize], specifying the cell state • Output 1: 2D tensor with shape [numBatches, outputSize],

No.	Operation	Description	Boundary
			specifying the output value
15	ANEURALNE TWORKS_EQ UAL (available since V310)	For input tensors x and y, computes x == y element- wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode and dimensions compatible with input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8</p>
16	ANEURALNE TWORKS_EX P (available since V310)	Computes exponential of x element- wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
17	ANEURALNE TWORKS_FL OOR	Computes floor() on the input tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor <p>[Restrictions]</p> <p>Quantization is not supported.</p> <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode and dimensions as input 0</p>

No.	Operation	Description	Boundary
18	ANEURALNETWORKS_FULLY_CONNECTED	Computes an inner product.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 2 or 4 Input 0: Tensor of rank 2 or 4, specifying the input. If rank is greater than 2, then it is flattened to a 2D Tensor, reshaped to [batch_size, input_size]. Input 1: 2D Tensor, with shape [num_units, input_size], where num_units indicates the number of output nodes. Specifying the weights. Input 2: 1D Tensor, with shape [num_units]. For a TENSOR_FLOAT32 Tensor, the bias should also be of TENSOR_FLOAT32. For a TENSOR_QUANT8_ASYMM Tensor, the bias should also be of TENSOR_INT32, with zeroPoint = 0 and bias_scale == input_scale * filter_scale. Specifying the bias. Input 3: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values <p>[Returns]</p> <p>Output 0: tensor, with shape [batch_size, num_units]. On a platform earlier than API 29, for a TENSOR_QUANT8_ASYMM tensor, output_scale > input_scale * filter_scale.</p>
19	ANEURALNETWORKS_GREATER (available since V310)	For input tensors x and y, computes x > y element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D Tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode as input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: TENSOR_BOOL8 Tensor</p>

No.	Operation	Description	Boundary
20	ANEURALNE TWORKS_GR EATER_EQU AL (available since V310)	For input tensors x and y, computes x ≥ y element- wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode as input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8</p>
21	ANEURALNE TWORKS_HA SHTABLE_LO OKUP	Looks up sub- tensors in the input tensor using a key- value map.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_QUANT8_ASYMM Supported tensor rank: 2–4 Input 0: Lookups, 1D tensor of type TENSOR_INT32, with shape [k] Input 1: Keys, 1D tensor of type TENSOR_INT32, with shape [n]. The Keys and Values pair represent a map. The <i>l</i>th element in Keys (Keys[<i>l</i>]) is the key to select the <i>l</i>th sub-tensor in Values (Values[<i>l</i>]), where 0 ≤ i ≤ n-1. The Keys tensor must be sorted in ascending order. Input 2: Values, tensor with shape [n, ...], where, the first dimension must be n. <p>[Returns]</p> <ul style="list-style-type: none"> Output 0: Output, tensor with shape [k ...]. For a TENSOR_QUANT8_ASYMM tensor, scale and zeroPoint must be the same as those of input 2. Output 1: Hits, boolean tensor with shape [k], indicating whether the lookup hits (True) or not (False). A non-zero byte represents True, a hit. A zero indicates otherwise. <p>[Restrictions]</p> <p>ALIGN(Lookups, 32) * 2 + ALIGN(Keys, 32) + ALIGN(Values, 256) * 32 ≤ 49152</p>

No.	Operation	Description	Boundary
22	ANEURALNE WORKS_L2 _NORMALIZ ATION	Applies L2 normalization along the depth dimension.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: (optional), scalar of type IINT32, specifying the dimension normalization would be performed on. (available since API 29) <p>[Returns]</p> <p>Output 0: tensor</p>
23	ANEURALNE WORKS_L2 _POOL_2D	Performs L2 pooling.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 NHWC data layout supported <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, with shape [batches, height, width, depth] Input 1: scalar of type INT32, specifying the padding on the left, in the width dimension, $0 \leq \text{Pad} < 256$ Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, $0 \leq \text{Pad} < 256$ Input 3: scalar of type INT32, specifying the padding on the top, in the height dimension, $0 \leq \text{Pad} < 256$ Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, $0 \leq \text{Pad} < 256$ Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, $\text{strideW} < 64$ Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, $\text{strideH} < 64$ Input 7: scalar of type INT32, specifying the filter width Input 8: scalar of type INT32, specifying the filter height Input 9: scalar of type INT32, specifying the activation to invoke on the result

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 10: (optional), scalar of type BOOL. Only the default format NHWC is supported. (available since API 29) <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, shape [batches, height, width, depth], specifying the input Input 1: scalar of type INT32, specifying the padding scheme, which must be one of the PaddingCode values (SAME or VALID) Input 2: scalar of type INT32, specifying the stride when walking through input in the width dimension, strideW < 64 Input 3: scalar of type INT32, specifying the stride when walking through input in the height dimension, strideH < 64 Input 4: scalar of type INT32, specifying the filter width Input 5: scalar of type INT32, specifying the filter height Input 6: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values Input 7: (optional), scalar of type BOOL. Only the default format NHWC is supported. (available since API 29) <p>[Returns]</p> <p>Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth].</p>
24	ANEURALNE WORKS_LE SS (available since V310)	For input tensors x and y, computes x < y element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode as input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8</p>
25	ANEURALNE WORKS_LE	For input tensors x and	<p>[Inputs]</p>

No.	Operation	Description	Boundary
	SS_EQUAL (available since V310)	y, computes $x \leq y$ element-wise.	<ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode as input 0 [Restrictions] <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. [Returns] Output 0: Tensor of type TENSOR_BOOL8
26	ANEURALNETWORKS_LOCAL_RESPONSE_NORMALIZATION	Applies Local Response Normalization (LRN) along the depth dimension.	[Inputs] <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 Input 0: 4D Tensor, with shape [batches, height, width, depth], specifying the input Input 1: scalar of type INT32, specifying the radius of the normalization window Input 2: scalar of type FLOAT32, specifying bias, which must not be 0 For FLOAT16 input 0, bias is of type FLOAT16. For FLOAT32 input 0, bias is of type FLOAT32. Input 3: scalar of type FLOAT32, specifying alpha For FLOAT16 input 0, alpha is of type FLOAT16. For FLOAT32 input 0, alpha is of type FLOAT32. Input 4: scalar of type FLOAT32, specifying beta For FLOAT16 input 0, beta is of type FLOAT16. For FLOAT32 input 0, beta is of type FLOAT32. Input 5: (optional), scalar of type INT32, specifying the dimension normalization would be performed on. (available since API 29) [Restrictions] If there are six inputs, the last INT32 input is -1 or the dimension of input 0 minus 1 (that is, 3).

No.	Operation	Description	Boundary
			<p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
27	ANEURALNE WORKS_LO G (available since V310)	Computes natural logarithm of x element- wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
28	ANEURALNE WORKS_LO GICAL_AND (available since V310)	Returns the truth value of x AND y element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_BOOL8 Supported tensor rank: up to 4 Input 0: tensor Input 1: Tensor with the identical shape as input 0 <p>[Returns]</p> <p>Output 0: tensor</p>
29	ANEURALNE WORKS_LO GICAL_NOT (available since V310)	Computes the truth value of NOT x element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_BOOL8 Supported tensor rank: up to 4 Input 0: tensor <p>[Returns]</p> <p>Output 0: tensor</p>
30	ANEURALNE WORKS_LO GICAL_OR (available since V310)	Computes sigmoid activation on the input tensor element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_BOOL8 Supported tensor rank: up to 4 Input 0: tensor Input 1: Tensor with the identical shape as input 0 <p>[Returns]</p> <p>Output 0: tensor</p>
31	ANEURALNE WORKS_LO	Logistic activation	<p>[Inputs]</p>

No.	Operation	Description	Boundary
	GISTIC		<ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: tensor <p>[Restrictions] None</p> <p>[Returns] Output 0: Tensor with the identical shape as input 0. For a TENSOR_QUANT8_ASYMM Tensor, scale = 1.f/256 and zeroPoint = 0.</p>
32	ANEURALNE TWORKS_LO G_SOFTMAX (available since V310)	Computes the log softmax activations given logits.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor Input 1: scalar of type TENSOR_FLOAT16 or TENSOR_FLOAT32, specifying the scaling factor for the operation Input 2: scalar of type INT32, specifying the dimension (axis) to reduce across <p>[Restrictions] axis is of the range [-rank, rank). Softmax can be performed on each of the four input dimensions (NCHW).</p> <ul style="list-style-type: none"> axis = 0: not supported axis = 1 (channel scenario): $C \leq 11136$ axis = 2 (height dimension): $W = 1, 0 < H < 16384$ axis = 3 (width dimension): $0 < W < 16384$ <p>[Returns] Output 0: Tensor of identical OperandCode as input 0</p>
33	ANEURALNE TWORKS_LS H_PROJECTI ON	Projects an input to a bit vector via locality sensitive	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29)

No.	Operation	Description	Boundary
		hashing.	<p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: Hash functions, 2D Tensor, FLOAT. tensor [0].Dim [0] specifies the number of hash functions. tensor [0].Dim [1] specifies the number of projected output bits generated by each hash function. If the projection type is Sparse: $\text{Tensor}[0].\text{Dim}[1] \leq 32$ Input 1: tensor, $\text{Dim.size} \geq 1$, no restriction on DataType Input 2: (optional) Weight Tensor, $\text{Dim.size} == 1$, $\text{DataType} == \text{Float}$. If this parameter is not set, each input element is considered to have the same weight of 1.0. $\text{Tensor}[1].\text{Dim}[0] == \text{Tensor}[2].\text{Dim}[0]$ Input 3: scalar of type int32 <p>Type:Sparse Value LSHProjectionType_SPARSE(=3) (available since API 29). Each output element is made up of multiple bits computed from hash functions.</p> <p>Type:Dense Value LSHProjectionType_DENSE(=2). The computed bit vector is considered to be dense. Each output element represents a bit and can take the value of either 0 or 1.</p> <p>[Returns]</p> <p>Output 0: tensor</p> <ul style="list-style-type: none"> If the projection type is Sparse: $\text{Output.Dim} == \{ \text{Tensor}[0].\text{Dim}[0] \}$ If the projection type is Dense: $\text{Output.Dim} == \{ \text{Tensor}[0].\text{Dim}[0] * \text{Tensor}[0].\text{Dim}[1] \}$
34	ANEURALNE WORKS_M AX_POOL_2 D	Performs max pooling	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM <p>Supported tensor rank: 4</p> <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, with shape [batches, height, width, depth] Input 1: scalar of type INT32, specifying the padding on the left, in the width dimension, $0 \leq \text{Pad} < 256$ Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, $0 \leq \text{Pad} < 256$ Input 3: scalar of type INT32, specifying the padding on

No.	Operation	Description	Boundary
			<p>the top, in the height dimension, $0 \leq \text{Pad} < 256$</p> <ul style="list-style-type: none"> • Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, $0 \leq \text{Pad} < 256$ • Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, $\text{strideW} < 64$ • Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, $\text{strideH} < 64$ • Input 7: scalar of type INT32, specifying the filter width • Input 8: scalar of type INT32, specifying the filter height • Input 9: scalar of type INT32, specifying the activation to invoke on the result • Input 10: (optional), scalar of type BOOL, specifying the data format. Only the default format NHWC is supported. (available since API 29) <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D Tensor, with shape [batches, height, width, depth], specifying the input • Input 1: scalar of type INT32, specifying the padding scheme, which must be one of the PaddingCode values (SAME or VALID) • Input 2: scalar of type INT32, specifying the stride when walking through input in the width dimension, $\text{strideW} < 64$ • Input 3: scalar of type INT32, specifying the stride when walking through input in the height dimension, $\text{strideH} < 64$ • Input 4: scalar of type INT32, specifying the filter width • Input 5: scalar of type INT32, specifying the filter height • Input 6: scalar of type INT32, specifying the activation to invoke on the result • Input 7: (optional), scalar of type BOOL, specifying the data format. Only the default format NHWC is supported. (available since API 29) <p>[Returns]</p> <p>Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth]</p> <p>[Restrictions]</p> <ul style="list-style-type: none"> • $\text{KernelH} < 256$, $\text{kernelW} < 256$

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> If H and W of the output tensor are 1: input H * input W < 65536
35	ANEURALNETWORKS_MAXIMUM (available since V310)	Returns the element-wise maximum of two tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor Input 1: Tensor of the same OperandCode and compatible dimensions with input 0. <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8; For a QUANT8_ASYMM tensor, scale and zeroPoint can be different from the input.</p>
36	ANEURALNETWORKS_MEAN	Computes the mean of elements across dimensions of a tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: tensor Input 1: 1D Tensor of type TENSOR_INT32, must be in the range [-rank(input_tensor), rank(input_tensor)). Specifying the dimension to be reduced. Input 2: scalar of type int32. If the value is positive, the reduced dimensions are retained with length 1. Specifying keep_dims. <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as input 0</p>
37	ANEURALNETWORKS_MINIMUM (available since V310)	Returns the element-wise minimum of two tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29)

No.	Operation	Description	Boundary
			<p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: tensor Input 1: Tensor of the same OperandCode and compatible dimensions with input 0. For a QUANT8_ASYMM tensor, its scale and zeroPoint can be different from those of input 0. <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8; For a QUANT8_ASYMM tensor, scale and zeroPoint can be different from the input.</p>
38	ANEURALNE TWORKS_M UL	Multiplies two tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: tensor Input 1: Tensor of identical OperandCode as input 0 Input 2: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as input 0. For the TENSOR_QUANT8_ASYMM tensor, output_scale > input1_scale * input2_scale</p>
39	ANEURALNE TWORKS_N E G (available since V310)	Computes numerical negative value element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>

No.	Operation	Description	Boundary
40	ANEURALNE TWORKS_N OT_EQUAL (available since V310)	For input tensors x and y, computes $x \neq y$ element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: Tensor of identical OperandCode and dimensions compatible with input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 0 and input 1 have identical dimensions. The product of dimension sizes of input 0 or input 1 must be less than 3000. <p>[Returns]</p> <p>Output 0: Tensor of type TENSOR_BOOL8</p>
41	ANEURALNE TWORKS_PA D	Pads a tensor with zeros.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] Input 1: 2D Tensor of type TENSOR_INT32, with shape {rank(input0), 2}. padding[i, 0] specifies the number of elements to be padded in the front of dimension <i>i</i>. padding[i, 1] specifies the number of elements to be padded after the end of dimension <i>i</i>. Specifying the number of elements to be padded in each space dimensions of input 0. <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode and dimensions compatible with input 0</p>
42	ANEURALNE TWORKS_PA D_V2 (available since V310)	Pads a tensor with the given constant value according to the specified paddings.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 Input 0: 4D Tensor

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 1: 2D Tensor of type TENSOR_INT32, with shape {rank(input0), 2}. padding[i, 0] specifies the number of elements to be padded in the front of dimension <i>i</i>. padding[i, 1] specifies the number of elements to be padded after the end of dimension <i>i</i>. Specifying the number of elements to be padded in each space dimensions of input 0. Input 2: scalar specifying the value to use for padding For TENSOR_FLOAT32 input 0, input 2 is of type FLOAT32. For TENSOR_FLOAT16 input 0, input 2 is of type FLOAT16. <p>[Returns] Output 0: Tensor of identical OperandCode and dimensions as input 0</p>
43	ANEURALNE WORKS_P RELU (available since V310)	Activation function Parametric Rectified Linear Unit (PRELU)	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor Input 1: tensor, specifying alpha <p>[Returns] Output 0: tensor</p>
44	ANEURALNE WORKS_Q UANTIZE (available since V310)	Quantizes the input tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported input tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: tensor <p>[Returns] Output 0: Tensor of type TENSOR_QUANT8_ASYMM, with the identical shape as input 0</p>
45	ANEURALNE WORKS_R ELU	Activation function ReLU	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29)

No.	Operation	Description	Boundary
			<p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: tensor <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
46	ANEURALNE TWORKS_RE LU1	Activation function ReLU1	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <p>TENSOR_FLOAT32 (only in the relaxed scenario)</p> <p>TENSOR_FLOAT16 (available since API 29)</p> <p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: up to 4</p> Input 0: tensor <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: tensor with the identical shape as input 0</p>
47	ANEURALNE TWORKS_RE LU6	Activation function ReLU6	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <p>TENSOR_FLOAT32 (only in the relaxed scenario)</p> <p>TENSOR_FLOAT16 (available since API 29)</p> <p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: up to 4</p> Input 0: Tensor <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
48	ANEURALNE TWORKS_RE SHAPE	Reshapes the input.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <p>TENSOR_FLOAT32 (only in the relaxed scenario)</p> <p>TENSOR_FLOAT16 (available since API 29)</p> <p>TENSOR_QUANT8_ASYMM</p> <p>Supported tensor rank: up to 4</p>

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 0: Tensor Input 1: 1D Tensor of type TENSOR_INT32, specifying the shape of the output Tensor <p>[Restrictions] None</p> <p>[Returns] Output 0: Tensor, with shape specified by the input</p>
49	ANEURALNE WORKS_RE SIZE_BILINE AR	Resizes images.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) Supported tensor rank: 4 <p>[Inputs (resizing by shape)]</p> <ul style="list-style-type: none"> Input 0: 4D Tensor, with shape [batches, height, width, depth], specifying the input. Input 1: scalar of type TENSOR_INT32, specifying the width of the output tensor Input 2: scalar of type int32 type, specifying the height of the output tensor Input 3: BOOL, specifying the data format. Only the default format NHWC is supported. (available since API 29) <p>[Input (resizing by scale, since API level 29)]</p> <ul style="list-style-type: none"> Input 0: 4D tensor, with shape [batches, height, width, depth], specifying the input. Input 1: scalar of type TENSOR_INT32 or TENSOR_INT16, specifying width_scale. The output width is calculated as $\text{new_width} = \text{floor}(\text{width} * \text{width_scale})$. Input 2: scalar of type TENSOR_INT32 or TENSOR_INT16, specifying height_scale. The output height is calculated as $\text{new_height} = \text{floor}(\text{height} * \text{height_scale})$. Input 3: (optional) scale of type BOOL, specifying the data format. Only the default format NHWC is supported. <p>[Returns] Output 0: 4D Tensor, with shape [batches, new_height, new_width, depth]</p>
50	ANEURALNE WORKS_RS QRT (available	Computes reciprocal of square root of x element-	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario)

No.	Operation	Description	Boundary
	since V310)	wise.	<p>TENSOR_FLOAT16 (available since API 29)</p> <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: n-D tensor, where N is within [1, 4] <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
51	ANEURALNE TWORKS_SIN (available since V310)	Computes sin of x element- wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> Input 0: n-D tensor, where N is within [1, 4] <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0</p>
52	ANEURALNE TWORKS_SOFTMAX	Normalizatio n logic function	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM <p>Supported tensor rank: 2 or 4</p> <ul style="list-style-type: none"> Input 0: 2D or 4D Tensor, with shape [batches, height, width, depth], specifying the input Input 1: scalar of type FLOAT32, specifying the positive scaling factor for beta For TENSOR_FLOAT32 or TENSOR_QUANT8_ASYMM input 0, scale must be of type TENSOR_FLOAT32. For TENSOR_FLOAT16 input 0, scale must be of type TENSOR_FLOAT16. Input 2: scalar of type INT32, specifying the dimension the activation would be performed on. (available since API 29) <p>[Restrictions]</p> <p>axis is of the range [-rank, rank).</p> <p>Softmax can be performed on each of the four input dimensions (NCHW).</p> <ul style="list-style-type: none"> axis = 0: $N \leq 28544$ axis = 1 (channel dimension): $C \leq 11136$ axis = 2 (height dimension): $W = 1, 0 < H < 16384$ axis = 3 (width dimension): $0 < W < 16384$

No.	Operation	Description	Boundary
			<p>If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968.</p> <p>[Returns]</p> <p>Output 0: Tensor with the identical shape as input 0. For a TENSOR_QUANT8_ASYMM Tensor, scale = 1.f/256 and zeroPoint = 0.</p>
53	ANEURALNE TWORKS_SP ACE_TO_BAT CH_ND	SpaceToBatch for n-D tensors.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 Input 0: n-D Tensor, specifying the input Input 1: 1D Tensor of type TENSOR_INT32, specifying the block sizes for each spatial dimension of the input Tensor. All values must be ≥ 1. Input 2: 2D Tensor of type TENSOR_INT32, with shape {M, or 2}, where M is the number of spatial dimensions. padding[i, 0] specifies the number of elements to be padded in the front of dimension <i>i</i>. padding[i, 1] specifies the number of elements to be padded after the end of dimension <i>i</i>. Specifying paddings for each spatial dimension of the input Tensor. All values must be ≥ 0. Input 3: (optional) BOOL, specifying the data format. Only the default format NHWC is supported. (available since API 29) <p>[Restrictions]</p> <p>When the tensor rank is 4: the length of block_shape must be 2, and the length of paddings must be 4.</p> <p>Element value of block_shape ≥ 1; Element value of paddings ≥ 0</p> <p>The padded H dimension is a multiple of block_shape[0], and the padded W dimension is a multiple of block_shape[1].</p> <p>The product of the dimensions of input 0 must be less than 500.</p> <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as input 0</p>
54	ANEURALNE	Rearranges	<p>[Inputs]</p>

No.	Operation	Description	Boundary
	TWORKS_SP ACE_TO_DEPTH	blocks of spatial data, into depth.	<ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 Input 0: 4D Tensor, with shape [batches, height, width, depth_in], specifying the input Input 1: scalar of type int32, block_size, specifying block_size. block_size ≥ 1 and must be a divisor of the height and width of the input Tensor. Input 2: (optional) BOOL, specifying the data format. Only the default format NHWC is supported. (available since API 29) <p>[Restrictions] blockSize ≥ 1 and blockSize must be a divisor of both the input height and width.</p> <p>[Returns] Output 0: 4D Tensor, with shape [batches, height/block_size, width/block_size, depth_in * block_size * block_size]</p>
55	ANEURALNE TWORKS_SQ RT (available since V310)	Computes square root of x element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: n-D tensor, where N is within [1, 4] <p>[Returns] Output 0: Tensor with the identical shape as input 0</p>
56	ANEURALNE TWORKS_SQ UEEZE	Squeeze	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: n-D Tensor, where N is within [1, 4] Input 1: (optional) 1D Tensor of type TENSOR_INT32. If not specified, all dimensions are squeezed. The dimension index starts at 0. An error is reported if a dimension that is

No.	Operation	Description	Boundary
			<p>not 1 is squeezed.</p> <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as input 0</p>
57	ANEURALNE TWORKS_ST RIDED_SLICE	Extracts a strided slice of a tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: n-D Tensor, specifying the input, where N is within [1, 4] Input 1: begin, 1D Tensor of type TENSOR_INT32. The length must be of rank(input0). Input 2: end, 1D Tensor of type TENSOR_INT32. The length must be of rank(input0). Input 3: strides, 1D Tensor of type TENSOR_INT32. The length must be of rank(input0). Input 4: begin_mask, scalar of type int32. If the λth bit of begin_mask is set, begin[λ] is ignored and the fullest possible range in that dimension is used instead. Input 5: end_mask, scalar of type int32. If the λth bit of end_mask is set, end [λ] is ignored and the fullest possible range in that dimension is used instead. Input 6: shrink_axis_mask, scalar of type int32. If the λth bit of shrink_axis_mask is set, the λth dimension is shrunk by 1, taking on the value at index begin [λ]. <p>[Restrictions]</p> <p>strides \neq 0</p> <p>[Returns]</p> <p>Output 0: Tensor of identical OperandCode as input 0</p>
58	ANEURALNE TWORKS_SU B	Subtraction of two tensors	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: n-D Tensor, specifying the input, where N is within [1, 4]

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 1: Tensor of identical OperandCode as input 0 Input 2: scalar of type INT32, specifying the activation to invoke on the result, which must be one of the FuseCode values <p>[Restrictions] None [Returns] Output 0: Tensor of identical OperandCode as input 0</p>
59	ANEURALNE TWORKS_TANH	Activation function Tanh	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 Input 0: Tensor <p>[Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0</p>
60	ANEURALNE TWORKS_TRANSPOSE	Transposes the input tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: up to 4 Input 0: n-D Tensor, specifying the input, where N is within [1, 4] Input 1: (optional) 1D Tensor of type TENSOR_INT32, specifying the dimension of the input Tensor to determine the transposition mode <p>[Restrictions] None [Returns] Output 0: Tensor of identical OperandCode as input 0</p>
61	ANEURALNE TWORKS_SPLIT (available	Splits a tensor along a given axis into	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario)

No.	Operation	Description	Boundary
	since V320)	num_splits subtensors.	<p>TENSOR_FLOAT16</p> <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> • Input 0: n-D tensor • Input 1: scalar of type TENSOR_INT32, specifying the dimension along which to split • Input 2: scalar of type TENSOR_INT32, specifying the num_split subtensors split into <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Outputs 0 to (num_split – 1): resulting subtensors of the identical type as input 0</p>
62	ANEURALNE WORKS_SLICE (available since V320)	Extracts a slice from a tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 <p>Supported tensor rank: up to 4</p> <ul style="list-style-type: none"> • Input 0: n-D tensor • Input 1: 1D tensor of type TENSOR_INT32, specifying the beginning indices of the slice in each dimension. • Input 2: 1D tensor of type TENSOR_INT32, specifying the size of the slice in each dimension <p>[Restrictions]</p> <ul style="list-style-type: none"> • A sized 0 tensor is not supported. • Input 1 and input 2 accept constants only. <p>[Returns]</p> <p>Output 0: n-D tensor of the identical type as input 0</p>
63	ANEURALNE WORKS_RESIZE_NEAREST (available since V320)	Resizes images to given size using the nearest neighbor interpretation.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 <ul style="list-style-type: none"> • Input 0: 4D tensor • Input 1: scalar. If the data type is TENSOR_INT32, this indicates the specifying the output width of the output 0 tensor. If the data type is TENSOR_FLOAT16 or TENSOR_FLOAT32, this indicates the scaling factor of the width dimension: $\text{new_width} = \text{floor}(\text{width} * \text{width_scale})$

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 2: scalar. If the data type is TENSOR_INT32, this indicates the specifying the output height of the output 0 tensor. If the data type is TENSOR_FLOAT16 or TENSOR_FLOAT32, this indicates the scaling factor of the height dimension: $\text{new_height} = \text{floor}(\text{height} * \text{height_scale})$ Input 3: scalar of type bool. Set to true to specify NCHW data layout for input 0 and output 0. Set to false for NHWC. <p>[Restrictions] Zero batches is not supported for the input tensor.</p> <p>[Returns] Output 0: 4D tensor of the identical type as input 0</p>
64	ANEURALNE TWORKS_HE ATMAP_MAX _KEYPOINT (available since V320)	Localizes the maximum keypoints from heatmaps.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 4D tensor with shape [num_boxes, heatmap_size, heatmap_size, num_keypoints], specifying the heatmaps, where, the height and width of heatmaps should be the same, and must be greater than or equal to 2 Input 1: 2D tensor with shape [num_boxes, 4], specifying the bounding boxes, each with format [x1, y1, x2, y2] Input 2: scalar of type bool. Set to true to specify NCHW data layout for input 0. Set to false for NHWC. <p>[Restrictions] The NCHW format is not supported.</p> <p>[Returns]</p> <ul style="list-style-type: none"> Output 0: 2D tensor of the identical type as input 0, with shape [num_boxes, num_keypoints], specifying the score of keypoints Output 1: 3D tensor of the identical type as input 1, with shape [num_boxes, num_keypoints, 2], specifying the location of the keypoints
65	ANEURALNE TWORKS_GA THER (available since V320)	Gathers values along an axis.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 <p>Supported tensor rank: up to 4</p>

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 0: n-D tensor Input 1: scalar of type INT32, specifying the axis to gather from. Must in range $[-n, n)$. Input 2: constant, k-D vector, specifying the indices of the axis dimension of input 0 <p>[Restrictions]</p> <ul style="list-style-type: none"> The value range of input 1 is $[-n, n)$. The values of input 2 must be in the bounds of the corresponding dimensions of input 0. Indices of input 2 must be constants. <p>[Returns]</p> <p>Output 0: $(n + k - 1)$-D tensor of the identical type as input 0</p>
66	ANEURALNE WORKS_PO W (available since V320)	Given a tensor base and a tensor exponent, computes $\text{base}^{\text{exponent}}$ element-wise.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor, specifying the base Input 1: n-D tensor, specifying the exponent <p>[Restrictions]</p> <ul style="list-style-type: none"> This operator does not support broadcast. When the base or power is constant, TENSOR_FLOAT16 is not supported. <p>[Returns]</p> <p>Output 0: n-D tensor of the identical type as input 0</p>
67	ANEURALNE WORKS_TIL E (available since V320)	Constructs a tensor by tiling a given tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor to be tiled Input 1: 1D tensor, tile multiples <p>[Restrictions]</p> <ul style="list-style-type: none"> The tile multiples must be constants. <p>[Returns]</p> <p>Output 0: n-D tensor of the identical type and rank as input</p>

No.	Operation	Description	Boundary
			0
68	ANEURALNE TWORKS_CH ANNEL_SHU FFLE (available since V320)	Shuffles the channels of the input tensor by dividing the channel dimension into numGroup groups, and reorganize the channels by grouping channels with the same index in each group.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor to be shuffled Input 1: scalar, specifying the number of groups Input 2: scalar, specifying the dimension channel shuffle would be performed on <p>[Restrictions]</p> <p>The value range of input 2 is [-n, n).</p> <p>[Returns]</p> <p>Output 0: n-D tensor of the identical type and rank as input 0</p>
69	ANEURALNE TWORKS_SE LECT (available since V320)	Selects elements from input 1 (if input0[i] = true) or input 2 (if input0[i] = false), depending on condition input 0.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor of type TENSOR_BOOL8, specifying the condition for selecting from input 1 (if true) or input 2 (if false) Input 1: n-D tensor, with the same shape as input 0 Input 2: n-D tensor, with the same type and shape as input 1 <p>[Restrictions]</p> <p>Each dimension size must be within 256.</p> <p>[Returns]</p> <p>Output 0: n-D tensor, with the same type and shape as input 1 and input 2.</p>
70	ANEURALNE TWORKS_TO PK_V2 (available since V320)	Finds values and indices of the k largest entries for the last dimension. Resulting	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4

No.	Operation	Description	Boundary
		values in each dimension are sorted in descending order.	<ul style="list-style-type: none"> Input 0: n-D tensor Input 1: scalar, specifying the number of top elements to look for along the last dimension <p>[Restrictions]</p> <p>Input 1 must be within the dimension size of the last dimension.</p> <p>[Returns]</p> <ul style="list-style-type: none"> Output 0: n-D tensor of the identical type as input 0 Output 1: n-D tensor of type TENSOR_INT32
71	ANEURALNETWORKS_EXPAND_DIMS (available since V320)	Inserts a dimension into a tensor's shape.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: scalar of type TENSOR_INT32, specifying the dimension index to expand <p>[Restrictions]</p> <p>Given an input of n dimensions, axis must be in range $[-(n + 1), (n + 1))$</p> <p>[Returns]</p> <p>Output 0: (n + 1)-D tensor of the identical type as input 0</p>
72	ANEURALNETWORKS_REDUCE_ALL (available since V320)	Reduces a tensor by computing the "logical and" of elements along given dimensions.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_BOOL8 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: scalar of type TENSOR_INT32, dimensions to reduce Input 2: scalar of type BOOL. If true, the reduced dimensions are retained with length 1. If false, the rank of the input 0 tensor is reduced by 1. <p>[Restrictions]</p> <p>Broadcast is not supported.</p> <p>[Returns]</p> <p>Output 0: Tensor of the identical type as input 0</p>
73	ANEURALNE	Reduces a	[Inputs]

No.	Operation	Description	Boundary
	TWORKS_REDUCE_ANY (available since V320)	tensor by computing the "logical or" of elements along given dimensions.	<ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_BOOL8 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: scalar of type TENSOR_INT32, dimensions to reduce Input 2: scalar of type BOOL. If true, the reduced dimensions are retained with length 1. If false, the rank of the input 0 tensor is reduced by 1. <p>[Restrictions] Broadcast is not supported.</p> <p>[Returns] Output 0: Tensor of the identical type as input 0</p>
74	ANEURALNETWORKS_REDUCE_PROD (available since V320)	Reduces a tensor by multiplying elements along given dimensions.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: 1D tensor of type TENSOR_INT32, dimensions to reduce Input 2: BOOL, keep_dims. If true, the reduced dimensions are retained with length 1. If false, the rank of the tensor is reduced by 1. <p>[Restrictions] Broadcast is not supported.</p> <p>[Returns] Output 0: Tensor of the identical type as input 0</p>
75	ANEURALNETWORKS_REDUCE_MAX (available since V320)	Reduces a tensor by computing the maximum of elements along given dimensions.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: 1D tensor of type TENSOR_INT32, dimensions to reduce Input 2: BOOL, keep_dims. If true, the reduced

No.	Operation	Description	Boundary
			<p>dimensions are retained with length 1. If false, the rank of the tensor is reduced by 1.</p> <p>[Restrictions]</p> <p>Broadcast is not supported.</p> <p>[Returns]</p> <p>Output 0: Tensor of the identical type as input 0</p>
76	ANEURALNE WORKS_RE DUCE_MIN (available since V320)	Reduces a tensor by computing the minimum of elements along given dimensions.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: 1D tensor of type TENSOR_INT32, dimensions to reduce Input 2: BOOL, keep_dims. If true, the reduced dimensions are retained with length 1. If false, the rank of the tensor is reduced by 1. <p>[Restrictions]</p> <p>Broadcast is not supported.</p> <p>[Returns]</p> <p>Output 0: Tensor of the identical type as input 0</p>
77	ANEURALNE WORKS_RE DUCE_SUM (available since V320)	Reduces a tensor by summing elements along given dimensions.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Supported tensor rank: up to 4 Input 0: n-D tensor Input 1: 1D tensor of type TENSOR_INT32, dimensions to reduce Input 2: BOOL, keep_dims. If true, the reduced dimensions are retained with length 1. If false, the rank of the tensor is reduced by 1. <p>[Restrictions]</p> <p>Broadcast is not supported.</p> <p>[Returns]</p> <p>Output 0: Tensor of the identical type as input 0</p>
78	ANEURALNE	Draws	[Inputs]

No.	Operation	Description	Boundary
	TWORKS_RANDOM_MULTINOMIAL (available since V320)	samples from a multinomial distribution.	<ul style="list-style-type: none"> Supported tensor OperandCode: <div> <div>TENSOR_FLOAT32 (only in the relaxed scenario)</div> <div>TENSOR_FLOAT16</div> </div> Input 0: 2D tensor, specifying the unnormalized log-probabilities for all classes Input 1: scalar, specifying the number of independent samples to draw for each row slice Input 2: 1D tensor with shape [2], specifying seeds used to initialize the random distribution <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 1 must be positive. Input 2 accepts constants only. <p>[Returns]</p> <p>Output 0: 2D tensor, drawn samples</p>
79	ANEURALNETWORKS_DETECTION_POSTPROCESSING (available since V320)	Apply post-processing steps to bounding box detections. Bounding box detections are generated by applying transformation on a set of predefined anchors with the bounding box deltas from bounding box regression. A final step of NMS is applied to limit the number of returned boxes.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <div> <div>TENSOR_FLOAT32 (only in the relaxed scenario)</div> <div>TENSOR_FLOAT16</div> </div> Input 0: 3D tensor, score, with shape [batches, num_anchors, num_classes] Input 1: 3D tensor, delta, with shape [batches, num_anchors, length_box_encoding] Input 2: 2D tensor, anchor, with shape [num_anchors, 4]. Input 3: scalar, scaleY, specifying the scaling factor for dy in bounding box deltas Input 4: scalar, scaleX, specifying the scaling factor for dx in bounding box deltas Input 5: scalar, scaleH, specifying the scaling factor for dh in bounding box deltas Input 6: scalar, scaleW, specifying the scaling factor for dw in bounding box deltas Input 7: scalar, specifying the NMS algorithm Input 8: scalar, specifying the maximum number of boxes for the output Input 9: scalar, only used when input 7 is set to false, specifying the maximum number of classes per detection Input 10: scalar, only used when input7 is set to true, specifying the maximum number of detections for each

No.	Operation	Description	Boundary
			<p>single class</p> <ul style="list-style-type: none"> • Input 11: scalar, score threshold. Boxes with scores lower than the threshold are filtered before sending to the non-maximal suppression (NMS) algorithm. • Input 12: scalar, intersection-over-union (IOU) threshold for NMS • Input 13: scalar. If true, includes background class in the list of label map for the output. If false, excludes the background. When the background class is included, it has label 0 and the output classes start at 1 in the label map, otherwise, the output classes start at 0. <p>[Restrictions]</p> <p>$0 < \text{Inputs } 8 \text{ and } 9 \leq 1024$ $\text{Input } 10 = 1$ $2 \leq \text{num_classes} < 1024$ $0 < \text{num_anchors} < 65536$ $\text{Inputs } 11 \text{ and } 12 \geq 0$ $\text{scaleX, scaleY, scaleH, and scaleW} > 1e - 5$</p> <p>[Returns]</p> <ul style="list-style-type: none"> • Output 0: 2D tensor with shape [batches, max_num_detections], specifying the score of each output detection • Output 1: 3D tensor with shape [batches, max_num_detections, 4], specifying the coordinates of each output ROI • Output 2: 2D tensor with shape [batches, max_num_detections], specifying the class label for each output detection • Output 3: 1D tensor, with shape [batches], specifying the number of valid output detections for each batch
80	ANEURALNE TWORKS_GR OUPED_CON V_2D (available since V320)	Performs a grouped 2D convolution operation.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D tensor with shape [batches, height, width, depth] • Input 1: 4D tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter • Input 2: 1D tensor, with shape [depth_out], specifying the

No.	Operation	Description	Boundary
			<p>bias</p> <ul style="list-style-type: none"> • Input 3: scalar of type INT32, specifying the padding on the left, in the width dimension • Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension • Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension • Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension • Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension • Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension • Input 9: scalar of type INT32, specifying the number of groups • Input 10: scalar of type INT32, specifying the activation to invoke on the result • Input 11: (optional) scalar of type BOOL. Set to true to specify the NCHW data layout. Set to false for NHWC. <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> • Input 0: 4D tensor with shape [batches, height, width, depth], specifying the input • Input 1: 4D tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter • Input 2: 1D tensor, with shape [depth_out], specifying the bias • Input 3: scalar of type INT32, specifying the implicit padding scheme. Must be one of the PaddingCode values (either SAME or VALID). • Input 4: scalar of type INT32, specifying the stride when walking through input in the width dimension • Input 5: scalar of type INT32, specifying the stride when walking through input in the height dimension • Input 6: scalar of type INT32, specifying the number of groups • Input 7: scalar of type INT32, specifying the activation to invoke on the result • Input 8: (optional) scalar of type BOOL. Set to true to specify the NCHW data layout. Set to false for NHWC. <p>[Restrictions]</p>

No.	Operation	Description	Boundary
			<p>inputC = group * filterC; filterN%group = 0</p> <p>[Returns]</p> <p>Output 0: 4D tensor with shape [batches, out_height, out_width, depth_out]</p>
81	ANEURALNETWORKS_TRANSPOSE_CONV_2D (available since V320)	Performs the transpose of 2D convolution operation.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 <p>[Inputs (explicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D tensor with shape [batches, height, width, depth] Input 1: 4D tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter Input 2: 1D tensor, with shape [depth_out], specifying the bias Input 3: scalar of type INT32, specifying the padding on the left, in the width dimension Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension Input 9: scalar of type INT32, specifying the activation to invoke on the result Input 10: (optional) scalar of type BOOL. Set to true to specify the NCHW data layout. Set to false for NHWC. <p>[Inputs (implicit padding)]</p> <ul style="list-style-type: none"> Input 0: 4D tensor with shape [batches, height, width, depth], specifying the input Input 1: 4D tensor, with shape [depth_out, filter_height, filter_width, depth_in], specifying the filter Input 2: 1D tensor, with shape [depth_out], specifying the bias Input 3: Tensor of type INT32, specifying the shape of the

No.	Operation	Description	Boundary
			<p>output tensor</p> <ul style="list-style-type: none"> Input 4: scalar of type INT32, specifying the implicit padding scheme. Must be one of the PaddingCode values (either SAME or VALID). Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension Input 7: scalar of type INT32, specifying the activation to invoke on the result Input 8: (optional) scalar of type BOOL. Set to true to specify the NCHW data layout. Set to false for NHWC. <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 1 and input 2 accept constants only. $(\text{inputH} - 1) * \text{strideH} + 1 + \text{aH} \leq 4000$; $(\text{inputW} - 1) * \text{strideW} + 1 + \text{aW} \leq 4000$; $\text{group} == 1$; $\text{dilationH} == \text{dilationW} == 1$; $\text{filterH} \leq 15 \ \&\& \ \text{filterW} \leq 15$; $\text{filterH} - \text{padHHead} - 1 \geq 0$ $\text{filterW} - \text{padWHead} - 1 \geq 0$ where, $\text{aH} = (\text{inputH} + \text{padHHead} + \text{padHTail} - \text{filterH}) \% \text{strideH}$ $\text{aW} = (\text{inputW} + \text{padHHead} + \text{padHTail} - \text{filterW}) \% \text{strideW}$ $(\text{inputH} - 1) * \text{strideH} - \text{padHHead} - \text{padHTail} \leq \text{outputH} \leq \text{inputH} * \text{strideH} - \text{padHHead} - \text{padHTail}$; $(\text{inputW} - 1) * \text{strideW} - \text{padWHead} - \text{padWTail} \leq \text{outputW} \leq \text{inputW} * \text{strideW} - \text{padWHead} - \text{padWTail}$ <p>[Returns]</p> <p>Output 0: 4D tensor</p>
82	ANEURALNETWORKS_LSTM (available since V320)	LSTM cell	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: non-constant 2D tensor, with shape [batch_size, input_size] Input 1: (optional) 2D tensor with shape [num_units,

No.	Operation	Description	Boundary
			<p>input_size], specifying the input-to-input weights</p> <ul style="list-style-type: none"> • Input 2: 2D tensor with shape [num_units, input_size], specifying the input-to-forgot weights • Input 3: 2D tensor with shape [num_units, input_size], specifying the input-to-cell weights • Input 4: 2D tensor with shape [num_units, input_size], specifying the input-to-output weights • Input 5: (optional) 2D tensor with shape [num_units, output_size], specifying the recurrent-to-input weights • Input 6: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-forgot weights • Input 7: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-cell weights • Input 8: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-output weights • Input 9: (optional) 1D tensor with shape [num_units], specifying the cell-to-input weights • Input 10: (optional) 1D tensor with shape [num_units], specifying the cell-to-forgot weights • Input 11: (optional) 1D tensor with shape [num_units], specifying the cell-to-output weights • Input 12: (optional) 1D tensor with shape [num_units], specifying the input gate bias • Input 13: 1D tensor with shape [num_units], specifying the forget gate bias • Input 14: 1D tensor with shape [num_units], specifying the cell gate bias • Input 15: 1D tensor with shape [num_units], specifying the output gate bias • Input 16: (optional) 2D tensor with shape [output_size, num_units], specifying the project weights • Input 17: 1D tensor with shape [output_size], specifying the project bias • Input 18: 2D tensor with shape [batch_size, output_size], specifying the output state (in) • Input 19: 2D tensor with shape [batch_size, num_units], specifying the cell state (in) • Input 20: scalar, indicating the activation function. 1: Relu; 2: Relu6; 3: Tanh; 4: Sigmoid. • Input 21: scalar, clipping threshold for the cell state. The

No.	Operation	Description	Boundary
			<p>value range is [-cell_clip, cell_clip]. If set to 0, then clipping is disabled.</p> <ul style="list-style-type: none"> Input 22: scalar, clipping threshold for the output from the projection layer. The value range is [-proj_clip, proj_clip]. If set to 0, then clipping is disabled. Input 23: 1D tensor with shape [num_units], specifying the input layer normalization weights Input 24: 1D tensor with shape [num_units], specifying the forget layer normalization weights Input 25: 1D tensor with shape [num_units], specifying the cell layer normalization weights Input 26: 1D tensor with shape [num_units], specifying the output layer normalization weights <p>[Restrictions]</p> <ul style="list-style-type: none"> Inputs 23–26 for layer normalization are not used in computation. Inputs 1–17 accept constants only. Outputting the scratch buffer is not supported. <p>[Returns]</p> <ul style="list-style-type: none"> Output 0: 2D tensor with shape [batch_size, num_units * 3] with CIFG, or [batch_size, num_units * 4] without CIFG. Output 1: 2D tensor with shape [batch_size, output_size], specifying the output state (out) Output 2: 2D tensor with shape [batch_size, num_units], specifying the cell state (out) Output 3: 2D tensor with shape [batch_size, output_size]
83	ANEURALNE TWORKS_UN IDIRECTION AL_SEQUEN CES_LSTM (available since V320)	Unidirectional LSTM cell	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: non-constant 3D tensor with shape [max_time, batch_size, input_size] if time-major, or [batch_size, max_time, input_size] if batch-major. Input 1: (optional) 2D tensor with shape [num_units, input_size], specifying the input-to-input weights Input 2: 2D tensor with shape [num_units, input_size], specifying the input-to-forgot weights Input 3: 2D tensor with shape [num_units, input_size], specifying the input-to-cell weights

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> • Input 4: 2D tensor with shape [num_units, input_size], specifying the input-to-output weights • Input 5: (optional) 2D tensor with shape [num_units, output_size], specifying the recurrent-to-input weights • Input 6: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-forgot weights • Input 7: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-cell weights • Input 8: 2D tensor with shape [num_units, output_size], specifying the recurrent-to-output weights • Input 9: (optional) 1D tensor with shape [num_units], specifying the cell-to-input weights • Input 10: (optional) 1D tensor with shape [num_units], specifying the cell-to-forgot weights • Input 11: (optional) 1D tensor with shape [num_units], specifying the cell-to-output weights • Input 12: (optional) 1D tensor with shape [num_units], specifying the input gate bias • Input 13: 1D tensor with shape [num_units], specifying the forget gate bias • Input 14: 1D tensor with shape [num_units], specifying the cell gate bias • Input 15: 1D tensor with shape [num_units], specifying the output gate bias • Input 16: (optional) 2D tensor with shape [output_size, num_units], specifying the project weights • Input 17: 1D tensor with shape [output_size], specifying the project bias • Input 18: 2D tensor with shape [batch_size, output_size], specifying the output state (in) • Input 19: 2D tensor with shape [batch_size, num_units], specifying the cell state (in) • Input 20: scalar, indicating the activation function. 1: Relu; 2: Relu6; 3: Tanh; 4: Sigmoid. • Input 21: scalar, clipping threshold for the cell state. The value range is [-cell_clip, cell_clip]. If set to 0, then clipping is disabled. • Input 22: scalar, clipping threshold for the output from the projection layer. The value range is [-proj_clip, proj_clip]. If set to 0, then clipping is disabled.

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 23: scalar. Time-major if true, batch-major if false. Input 24: 1D tensor with shape [num_units], specifying the input layer normalization weights Input 25: 1D tensor with shape [num_units], specifying the forget layer normalization weights Input 26: 1D tensor with shape [num_units], specifying the cell layer normalization weights Input 27: 1D tensor with shape [num_units], specifying the output layer normalization weights <p>[Restrictions]</p> <ul style="list-style-type: none"> Inputs 24-27 for layer normalization are not used in computation. Inputs 1-17 accept constants only. Outputting the scratch buffer is not supported. <p>[Returns]</p> <p>Output 0: 3D tensor with shape [max_time, batch_size, output_size] if time-major, or [batch_size, max_time, output_size] if batch-major.</p>
84	ANEURALNETWORKS_ROI_POOLING (available since V320)	Selects and scales the feature map of each region of interest to a unified output size by max-pooling.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 4D tensor, specifying the feature map Input 1: 2D tensor with shape [nums_rois, 4] Input 2: 1D tensor with shape [nums_rois] Input 3: scalar, specifying the output height of the output tensor Input 4: scalar, specifying the output width of the output tensor Input 5: scalar, specifying the ratio from the height of original image to the height of feature map Input 6: scalar, specifying the ratio from the width of original image to the width of feature map Input 7: scalar. Set to true to specify NCHW data layout for input 0 and output 0. Set to false for NHWC. <p>[Restrictions]</p> <ul style="list-style-type: none"> Input 1 accepts constants only and does not support time-major computation.

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 2 accepts constants only <p>[Returns]</p> <p>Output 0: 4D tensor.</p>
85	ANEURALNE TWORKS_SV DF (available since V320)	A densely connected layer that's processing a sequence of input frames can be approximated by using a singular value decomposition of each of its nodes.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 2D tensor with shape [batch_size, input_size] Input 1: 2D tensor with shape [num_units, input_size] Input 2: 2D tensor with shape [num_units, memory_size] Input 3: (optional) 1D tensor with shape [num_units] Input 4: 2D tensor with shape [batch_size, memory_size * num_units * rank] Input 5: scalar, specifying the rank of SVD approximation Input 6: scalar, indicating the activation function. Must not be NONE. <p>[Restrictions]</p> <p>Inputs 1–4 accept constants only</p> <p>[Returns]</p> <ul style="list-style-type: none"> Output 0: 2D tensor with shape [batch_size, memory_size * num_units * rank] Output 1: 2D tensor with shape [batch_size, num_units]
86	ANEURALNE TWORKS_IN STANCE_NO RMALIZATIO N (available since V320)	Applies instance normalization to the input tensor.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: non-constant, n-D tensor Input 1: scalar, specifying gamma, the scale applied to the normalized tensor Input 2: scalar, specifying beta, the offset applied to the normalized tensor Input 3: scalar, specifying epsilon, the small value added to variance to avoid dividing by zero <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: n-D tensor</p>

No.	Operation	Description	Boundary
87	ANEURALNETWORKS_BIDIRECTIONAL_SEQUENCE_LSTM (available since V320)	Performs a forward LSTM on the input followed by a backward LSTM.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 3D tensor with shape [max_time, batch_size, input_size] or [batch_size, max_time, input_size] Inputs 1–4: 2D tensor with shape [fw_num_units, input_size], specifying the forward input-to-input, input-to-forget, input-to-cell, and input-to-output weights, respectively. The input-to-input weights input is optional. Inputs 5–8: 2D tensor with shape [fw_num_units, fw_output_size], specifying the forward recurrent-to-input, recurrent-to-forget, recurrent-to-cell, and recurrent-to-output weights, respectively. The recurrent-to-input weights input is optional. Inputs 9–11: (optional) 1D tensor with shape [fw_num_units, input_size], specifying the forward cell-to-input, cell-to-forget, and cell-to-output weights, respectively. Inputs 12–15: 1D tensor with shape [fw_num_units], specifying the forward input, forget, cell, and output bias, respectively Input 16: (optional) 2D tensor with shape [fw_output_size, fw_num_units], specifying the forward projection weights Input 17: (optional) 1D tensor with shape [fw_output_size], specifying the forward projection bias Input: 18–21: 2D tensor with shape [bw_num_units, input_size], specifying the backward input-to-input, input-to-forget, input-to-cell, and input-to-output weights, respectively. The input-to-input weights input is optional. Inputs 22–25: 2D tensor with shape [bw_num_units, bw_output_size], specifying the backward recurrent-to-input, recurrent-to-forget, recurrent-to-cell, and recurrent-to-output weights, respectively. The recurrent-to-input weights input is optional. Inputs 26–28: (optional) 1D tensor with shape [bw_num_units], specifying the forward cell-to-input, cell-to-forget, and cell-to-output weights, respectively Inputs 29–32: 1D tensor with shape [bw_num_units], specifying the forward input, forget, cell, and output bias, respectively Input 33: (optional) 2D tensor with shape

No.	Operation	Description	Boundary
			<p>[bw_output_size, bw_num_units], specifying the forward projection weight</p> <ul style="list-style-type: none"> • Input 34: (optional) 1D tensor with shape [bw_output_size], specifying the forward projection bias • Input 35: 2D tensor with shape [batch_size, bw_output_size], specifying the forward input activation state • Input 36: 2D tensor with shape [batch_size, bw_num_units], specifying the forward input cell state • Input 37: 2D tensor with shape [batch_size, bw_output_size], specifying the backward input activation state • Input 38: 2D tensor with shape [batch_size, bw_num_units], specifying the backward input cell state • Input 39: (optional) 3D tensor with shape [max_time, batch_size, input_size], specifying the auxiliary input • Inputs 40–43: (optional) 2D tensor with shape [fw_num_units, input_size], specifying the auxiliary forward input-to-input, input-to-forget, input-to-cell, and input-to-output weights, respectively • Inputs 44–47: (optional) 2D tensor with shape [bw_num_units, input_size], specifying the backward auxiliary input-to-input, input-to-forget, input-to-cell, and input-to-output weights, respectively • Input 48: activation function. 1: Relu; 2: Relu6; 3: Tanh; 4: Sigmoid. • Inputs 49–50: clipping thresholds • Input 51: scalar, specifying if the outputs from forward and backward cells should be merged • Input 52: scalar, specifying the shape format of input and output tensors • Inputs 53–56: (optional) 1D tensor, specifying the forward normalization weights for the input, forget, cell, and output layers • Inputs 57–60: (optional) 1D tensor, specifying the backward normalization weights for the input, forget, cell, and output layers <p>[Restrictions]</p> <ul style="list-style-type: none"> • Only constant inputs are accepted except input 0 and inputs 35–38. • Auxiliary inputs 39–47 and normalized weights 53–60 are

No.	Operation	Description	Boundary
			<p>in valid.</p> <ul style="list-style-type: none"> • merge_output is not supported. That is, input 51 must be false. <p>[Returns]</p> <ul style="list-style-type: none"> • Output 0: 3D tensor, forward LSTM result • Output 1: 3D tensor, backward LSTM result
88	ANEURALNE TWORKS_UN IDIRECTION AL_SEQUEN CE_RNN (available since V320)	Applies a basic RNN cell to a sequence of inputs.	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: <p>TENSOR_FLOAT32 (only in the relaxed scenario)</p> <p>TENSOR_FLOAT16</p> <p>The input tensors must all be the same type.</p> • Input 0: 3D tensor. The shape is defined by input 6 (timeMajor). If it is set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If it is set to 1, the input and output shape is [maxTime, batchSize, numUnits]. • Input 1: constant, 2D tensor with shape [num_units, input_size], specifying the weights • Input 2: constant, 2D tensor with shape [num_units, num_units], specifying the recurrent weights • Input 3: constant, 1D tensor with shape [num_units], specifying the bias • Input 4: 2D tensor with shape [batch_size, num_units], specifying the hidden state (in) • Input 5: (optional) fused_activation_function, a FuseCode value indicating the activation function. Must not be None. • Input 6: timeMajor. If set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If set to 1, the input and output shape is [maxTime, batchSize, numUnits]. <p>[Returns]</p> <p>Output 0: 3D tensor. The shape is defined by input 6 (timeMajor). If it is set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If it is set to 1, the input and output shape is [maxTime, batchSize, numUnits].</p>
89	ANEURALNE TWORKS_BI DIRECTIONA L_SEQUENCE _RNN (available	Applies a basic RNN cell to a sequence of inputs in forward and	<p>[Inputs]</p> <ul style="list-style-type: none"> • Supported tensor OperandCode: <p>TENSOR_FLOAT32 (only in the relaxed scenario)</p> <p>TENSOR_FLOAT16</p> <p>The input tensors must all be the same type.</p>

No.	Operation	Description	Boundary
	since V320)	backward directions.	<ul style="list-style-type: none"> Input 0: 3D tensor. The shape is defined by input 6 (timeMajor). If it is set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If it is set to 1, the input and output shape is [maxTime, batchSize, numUnits]. Input 1: constant, 2D tensor with shape [fwNumUnits, inputSize], specifying fwWeights Input 2: constant, 2D tensor with shape [fwNumUnits, fwNumUnits], specifying fwRecurrentWeights Input 3: constant, 1D tensor with shape [fwNumUnits, inputSize], specifying fwBias Input 4: 2D tensor with shape [batchSize, fwNumUnits], specifying fwHiddenState, a hidden state input for the first time step of the computation. Input 5: constant, 2D tensor with shape [bwNumUnitsNumUnits, inputSize], specifying bwWeights Input 6: constant, 2D tensor with shape [bwNumUnits, fwNumUnits], specifying bwRecurrentWeights Input 7: constant, 1D tensor with shape [bwNumUnits, inputSize], specifying bwBias Input 8: 2D tensor with shape [batchSize, bwNumUnits], specifying bwHiddenState Input 9: 3D tensor with the identical shape as input 0, specifying auxInput. This parameter is invalid. Input 10: 2D tensor with shape [fwNumUnits, inputSize], specifying fwAuxWeights. This parameter is invalid. Input 11: 2D tensor with shape [bwNumUnits, inputSize], specifying bwAuxWeights Input 12: (optional) fused_activation_function, a FuseCode value indicating the activation function. Must not be None. Input 13: timeMajor. If set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If set to 1, the input and output shape is [maxTime, batchSize, numUnits]. Input 14: BOOL, mergeOutputs specifying if the outputs from forward and backward cells are separate (if set to 0) or concatenated (if set to 1) <p>[Restrictions] mergeOutputs must be false.</p> <p>[Returns] <ul style="list-style-type: none"> Output 0: fwOutput, a 3D tensor. The shape is defined by </p>

No.	Operation	Description	Boundary
			<p>input 6 (timeMajor). If it is set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If it is set to 1, the input and output shape is [maxTime, batchSize, numUnits].</p> <ul style="list-style-type: none"> Output 1: bwOutput, a 3D tensor. The shape is defined by input 6 (timeMajor). If it is set to 0, the input and output shape is [batchSize, maxTime, numUnits]. If it is set to 1, the input and output shape is [maxTime, batchSize, numUnits].
90	ANEURALNE WORKS_RN N (available since V320)	A basic recurrent neural network layer.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 The input tensors must all be the same type. Input 0: 2D tensor with shape [batch_size, input_size] Input 1: constant, 2D tensor with shape [num_units, input_size], specifying the weights Input 2: constant, 2D tensor with shape [num_units, num_units], specifying the recurrent weights Input 3: constant, 1D tensor with shape [num_units], specifying the bias Input 4: 2D tensor with shape [batch_size, num_units], specifying the hidden state (in) Input 5: (optional) fused_activation_function, a FuseCode value indicating the activation function. Must not be None. <p>[Returns]</p> <p>Output 0: 2D tensor with shape [batch_size, num_units], specifying the hidden state (out)</p> <p>Output 1: 2D tensor with shape [batch_size, num_units], specifying the output</p>
91	ANEURALNE WORKS_ROI_ALIGN (available since V320)	Selects and scales the feature map of each region of interest to a unified output size by average pooling	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 4D tensor, specifying the feature map Input 1: 2D tensor with shape [nums_rois, 4] Input 2: 1D tensor with shape [nums_rois], specifying the batch index of each box

No.	Operation	Description	Boundary
		sampling points from bilinear interpolation.	<ul style="list-style-type: none"> • Input 3: scalar, specifying the output height of the output tensor • Input 4: scalar, specifying the output width of the output tensor • Input 5: scalar, specifying the ratio from the height of original image to the height of feature map • Input 6: scalar, specifying the ratio from the width of original image to the height of feature map • Input 7: scalar, specifying the number of sampling points in height dimension used to compute the output • Input 8: scalar, specifying the number of sampling points in width dimension used to compute the output • Input 9: scalar. Set to true to specify NCHW data layout for input 0 and output 0. Set to false for NHWC. <p>[Restrictions]</p> <ul style="list-style-type: none"> • Input 1 accepts constants only and does not support time-major computation. • Input 2 accepts constants only <p>[Returns]</p> <p>Output 0: 4D tensor.</p>
92	ANEURALNE WORKS_GENERATE_PROPOSALS (available since V320)	Generates axis-aligned bounding box proposals.	<p>[Inputs]</p> <p>Supported tensor OperandCode: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16</p> <ul style="list-style-type: none"> • Input 0: 4D tensor with shape [batches, num_anchors, height, width] • Input 1: 4D Tensor with shape [batches, num_anchors * 4, height, width], specifying the bounding box • Input 2: 2D Tensor with shape [num_anchors, 4] • Input 3: 2D Tensor with shape [batches, 2], specifying the size of each image in the batch • Input 4: scalar of type float32, specifying the ratio from the height of original image to the height of feature map • Input 5: scalar of type float32, specifying the ratio from the width of original image to the width of feature map • Input 6: scalar of type int32, specifying the maximum number of boxes before going into the NMS algorithm • Input 7: scalar of type int32, specifying the maximum number of boxes returning from the NMS algorithm

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> Input 8: scalar of type float32, specifying the IoU threshold for NMS Input 9: scalar of type float32, specifying the minimum box size Input 10: BOOL. Set to true to specify the NCHW data layout for inputs 0 and 1. Set to false for NHWC. <p>[Restrictions]</p> <ul style="list-style-type: none"> Only NHWC is supported. <p>[Returns]</p> <ul style="list-style-type: none"> Output 1: 1D tensor with shape [num_output_rois], specifying the score of each output box Output 2: 2D tensor with shape [num_output_rois, 4], specifying the coordinates of each output bounding box for each class Output 3: 1D tensor of type int32, with shape [num_output_rois], specifying the batch index of each box
93	ANEURALNE TWORKS_AXIS_ALIGNED_ BBOX_TRANSFORM (available since V320)	Transform axis-aligned bounding box proposals using bounding box deltas.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 2D tensor with shape [num_rois, 4] Input 1: 2D tensor with shape [num_rois, num_classes x 4] Input 2: 1D tensor with shape [num_rois], in NCHW format Input 3: 2D tensor with shape [batches, 2] <p>[Restrictions]</p> <p>None</p> <p>[Returns]</p> <p>Output 0: 2D tensor with shape [num_rois, num_classes * 4]</p>
94	ANEURALNE TWORKS_BOX_WITH_NMS_LIMIT (available since V320)	Greedily selects a subset of bounding boxes in descending order of score.	<p>[Inputs]</p> <ul style="list-style-type: none"> Supported tensor OperandCode: <ul style="list-style-type: none"> TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 Input 0: 2D tensor with shape [num_rois, num_classes] Input 1: 2D tensor with shape [num_rois, num_classes x 4] Input 2: 1D tensor with shape [num_rois] Input 3: scalar, specifying score_threshold

No.	Operation	Description	Boundary
			<ul style="list-style-type: none"> • Input 4: scalar, specifying the maximum number of selected bounding boxes for each image • Input 5: scalar, specifying the NMS kernel method • Input 6: scalar, specifying the IoU threshold • Input 7: scalar, specifying the sigma in gaussian NMS kernel This field is invalid in this version. • Input 8: scalar, specifying nms_score_threshold <p>[Restrictions] Only the hard NMS algorithm is supported.</p> <p>[Returns]</p> <ul style="list-style-type: none"> • Output 0: 1D tensor with shape [num_output_rois] • Output 1: 2D tensor with shape [num_output_rois, 4] • Output 2: 1D tensor with shape [num_output_rois] • Output 3: 1D tensor with shape [num_output_rois]

3 CPU Operator List

No.	Operator	Remarks
1	Convolution	-
2	Scale	out = alpha x input + beta
3	Relu	Activation function
4	Pooling	Pooling layer
5	Eltwise	Computes element-wise operations (PROD, MAX, and SUM).
6	FullConnection	Fully connected
7	Softmax	Normalization logic function
8	Deconvolution	-
9	Crop	-
10	Concat	Stitches tensors by dimension.
11	Reshape	Reshapes the input.
12	Sigmoid	Activation function
13	Power	$y = (\text{scale} * x + \text{shift})^{\text{power}}$
14	Argmax	Computes the index of the maximum values.
15	Interp	Interpolation layer
16	LeakyRelu	Activation function
17	ConvolutionDepthwise	Depthwise convolution
18	FractionalPooling	Fractional pooling

No.	Operator	Remarks
19	Pow	Power
20	Multinomial	Generates random data of polynomial distribution.
21	RandomNormal	Generates random data with normal distribution.
22	RandomShuffle	Shuffles data along the first dimension.
23	RandomUniformNoSeed	Generates evenly distributed random data.
24	Cos	Performs cos computation.
25	Sin	Performs sin computation.
26	Tan	Performs tan computation.
27	TopK	Extracts the k maximum values of the last dimension.
28	Select	Selects the output based on conditions.
29	Permute	Permutes the input dimensions.
30	Prelu	Activation function
31	BatchNorm	Normalizes the input in batches.
32	Pad	Pads the input shape.
33	ExpandDims	Expands dimensions for the input shape.
34	InstanceNorm	Normalizes the input by pixels.
35	Add	Adds the two inputs.
36	Sub	Subtracts the two inputs.
37	Mul	Multiplies the two inputs.
38	Maximum	Returns the larger value between the two inputs.
39	BiasAdd	Adds the input tensor and a constant input.
40	StrideSlice	Extracts a strided slice of a tensor according to the specified start and end positions.
41	Pack	Stacks multiple inputs into one.
42	Abs	Obtains the absolute value of the input.
43	Matmul	Performs matrix multiplication on the two

No.	Operator	Remarks
		inputs.
44	GreaterEqual	Returns the truth value of $(x1 \geq x2)$ element-wise.
45	Less	Returns the truth value of $(x1 < x2)$ element-wise.
46	RealDiv	Returns $x1/x2$ element-wise for real types.
47	Tile	Tiles a given matrix.
48	PriorBox	Generates a prior box at each location of the feature map.
49	ReduceMean	Calculates the mean value of elements on each dimension of a tensor.
50	SSDDetectionOutput	An operator of the SSD network. Outputs the number of detection objects and candidate boxes based on the prior box, corresponding offset, and score.
51	Cast	Converts the input data to a new type.
52	Fill	Creates a tensor filled with a scalar value.
53	Flatten	Flattens each sub tensor of the first dimension of the input into a row vector.
54	ResizeBilinear	Resizes images to a specified size.