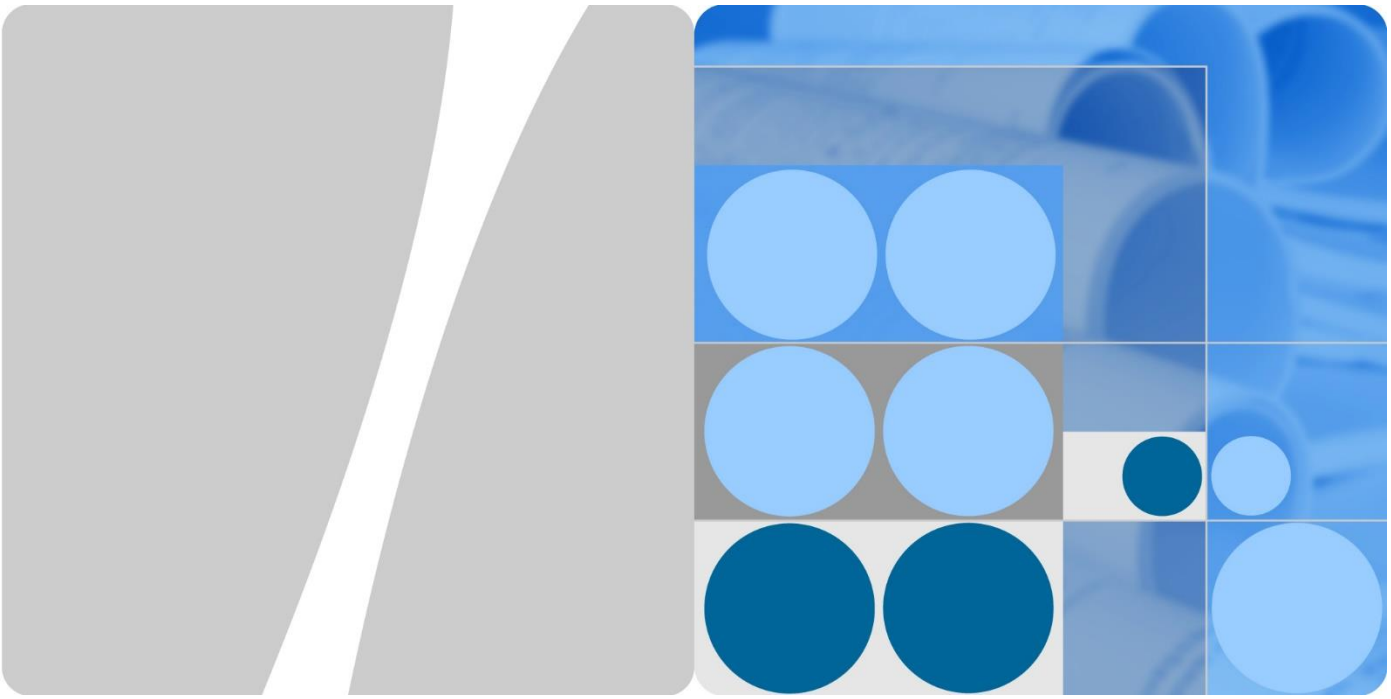

HiAI DDK V320

Operator Specifications

Issue 04

Date 2020-02-18

Huawei Technologies Co., Ltd.

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

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 ≤ N ≤ 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 – avg(x))/x]	[Inputs] One input [Arguments] • **use_global_stats**: bool, must be **true**

No.	Operator	Description	Boundary
			• **moving_average_fraction**: (optional) float, default to **0.999** • **eps**: (optional) float, default to **1e – 5** [Restrictions] • Only C dimension can be normalized. • Shape of **scale**, **beta**, **mean**, and **gamma**: [c]
4	ChannelAxpy (available since V310)	ChannelAxpy (Squeeze-and-Excitation Networks)	[Inputs] Three inputs • 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.	[Inputs] Multiple inputs [Arguments] • **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. [Restrictions] • 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	Convolves the input.	[Inputs] One input [Arguments] • **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
			• **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** [Restrictions] • **filter** must be 4D constants. • (inputW + padWHead + padWTail) ≥ (((FilterW – 1) * dilationW) + 1) • (inputW + padWHead + padWTail)/StrideW + 1 ≤ 2147483647 • (inputH + padHHead + padHTail) ≥ (((FilterH – 1) * dilationH) + 1) • (inputH + padHHead + padHTail)/StrideH + 1 ≤ 2147483647 • 0 ≤ Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 ≤ dilationsize < 256 • StrideW ≤ (inputW + padW) – ((filterW – 1) * dilationW) + 1) • The shape of **bias** must be (1, co, 1, 1), with the same **co** as **num_output**.
7	Crop	Crops the input.	[Inputs] Two inputs [Arguments] • **axis**: (optional) int32, default to **2**. When axis is **–1**, four input dimensions are required. • **offset**: uint32, array
8	Correlation (available since V310)	Correlation	[Inputs] Two inputs, with the same C dimension [Arguments] • **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
			[Restrictions] • N ≤ 3840 • Other constraints are the same as those of convolution.
9	Deconvolution	Deconvolution	[Inputs] One input [Arguments] • **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** [Restrictions] • **filter** must be 4D constants. • group < 1000 • dilation = 1 • filterH – padHHead – 1 ≥ 0 • filterW – padWHead – 1 ≥ 0 • Restrictions involving intermediate variables: 1. a = ALIGN(filter_num, 16) * ALIGN(filter_c, 16) * filter_h * filter_w * 2 If ALIGN(filter_c, 16)%32 = 0, a = a/2 2. conv_input_width = (deconvolution input W – 1) * strideW

No.	Operator	Description	Boundary
			+ 1 3. b = (conv_input_width) * filter_h * ALIGN(filter_num, 16) * 4 4. a + b ≤ 512 KB
10	DepthwiseConvolution	Depthwise convolution	[Inputs] One input [Arguments] • **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** [Restrictions] • **filter** must be 4D constants. • filterN = inputC = group • StrideW ≤ (inputW + padW) – ((filterW – 1) * 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).	[Inputs] At least two inputs [Arguments] • **operation**: (optional) enum, (PROD = 0; SUM = 1; MAX = 2),

No.	Operator	Description	Boundary
			default to **SUM** • **coeff**: array, float • **stable_prod_grad**: (optional) bool, default to **true** [Restrictions] • 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	[Inputs] One input [Arguments] **alpha**: (optional) float, default to **1**
13	Exp	Exponential operation	[Inputs] One input [Arguments] • **base**: (optional) float, default to **–1.0** • **scale**: (optional) float, default to **1.0** • **shift**: (optional) float, default to **0.0**
14	Flatten	Flattens data along the first dimension. Converts an input of shape N * C * H * W to a vector output of shape N * (C * H * W).	[Inputs] One input [Arguments] • **axis**: (optional) int32, default to **1** • **end_axis**: (optional) int32, default to **–1** [Restrictions] axis < end axis
15	InnerProduct	Computes an inner product.	[Inputs] One input [Arguments] • **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
			• **axis**: (optional) int32, default to **1** • **transpose**: (optional) bool, default to **false** [Restrictions] • transpose = false, axis = 1 • bais_C ≤ 56832 • The shape of **bias** must be (1, co, 1, 1), with the same **co** as **num_output**. To quantify a model, the following dimension restrictions must be satisfied: • When N = 1: 2 * ALIGN(C, 16)* xH * xW ≤524288; • When N > 1: 2 * 16 * ALIGN(C, 16) * xH * xW ≤524288.
16	Interp	Interpolation layer	[Inputs] One input [Arguments] • **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** Note the following: • **zoom_factor** and **shrink_factor** are exclusive. • **height** and **zoom_factor** are exclusive. • **height** and **shrink_factor** are exclusive. [Restrictions] (outputH * outputW)/(inputH * inputW) > 1/7
17	LeakyRelu	LeakyRelu activation function	[Inputs] One input [Arguments] Same as Relu
18	Log	Performs logarithmic operation on the input.	[Inputs] One input [Arguments] • **base**: (optional) float, default to **–1.0** • **scale**: (optional) float, default to **1.0**

No.	Operator	Description	Boundary
			• **shift**: (optional) float, default to **0.0**
19	LRN	Normalizes the input in a local region.	[Inputs] One non-constant input [Arguments] • **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 [Restrictions] • **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)	[Inputs] Two or three inputs • **X**: time sequence data (T * N * Xt) • **Cont**: sequence continuity flag (T * N) • **Xs**: (optional) static data (N * Xt) [Arguments] • **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** [Restrictions] Restrictions involving intermediate variables: • a = (ALIGN(xt,16) + ALIGN(output,16)) * 64 • b = (ALIGN(xt,16) + ALIGN(output,16)) * 256 • c = use_projection ? ALIGN(ht,16) * ALIGN(output,16) * 2) : 0 • d = 16 * ALIGN(ht,16) * 2

No.	Operator	Description	Boundary
			• e = batchNum * 4 That is: • a + b + c ≤ 524288 • d ≤ 16384 • e ≤ 4096
21	Normalize	Normalize layer	[Inputs] One input [Arguments] • **across_spatial**: (optional) bool, default to **true** • **scale_filler**: (optional) default to **1.0** • **channel_shared**: (optional) bool, default to **true** • **eps**: (optional) float, default to **1e – 10** [Restrictions] • 1e – 7 < eps ≤ 0.1 + (1e – 6) • **across_spatial** must be **true** for Caffe, indicating normalization by channel
22	Permute	Reshapes the input.	[Inputs] One input [Arguments] **order**: uint32, array
23	Pooling	Pools the input.	[Inputs] One input [Arguments] • **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
			• **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** [Restrictions] • KernelH < 256, kernelW < 256; • stride < 64; stride > pad; stride < input + pad – kernel • If H and W of the output tensor are **1**: input H * input W < 65536 • kernelH ≤ inputH + padTop + padBottom • kernelW ≤ inputW + padLeft + padRight • padTop < windowH • padBottom < windowH • padLeft < windowW • padRight < windowW • Only the global pooling mode is supported. The following restrictions must be satisfied: 1) outputH ==1 && outputW ==1 && kernelH ≥ inputH && kernelW ≥ inputW 2) inputH * inputW ≤ 10000
24	Power	Computes the output **y** as (scale * x + shift)^power.	[Inputs] One input [Arguments] • **power**: (optional) float, default to **1.0** • **scale**: (optional) float, default to **1.0** • **shift**: (optional) float, default to **0.0** [Restrictions] scale * x + shift > 0
25	Prelu	Activation function	[Inputs] One input [Arguments] • **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.	[Inputs] • Input 0: (mandatory) Only the input shape is concerned. • Input 1: (optional) image description [Arguments] • **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 [Restrictions] • 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.	[Inputs] Three inputs: **scores**, **bbox_pred**, **im_info** [Arguments] • **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
			• **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 • scaleCnt * ratioCnt ≤ 64 • 0 < **nmsTresh** ≤ 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 * scaleCnt * ratioCnt **bboxPred**: C = 4 * scaleCnt * ratioCnt **bboxPrior**: N = clsProb.N, C = 4 * scaleCnt * ratioCnt **imInfo**: N = clsProb.N, C = 3
28	PSROIPooling	Position-sensitive region-of-interest pooling (PSROIPooling)	[Inputs] Two inputs [Arguments] • **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] • Used for the Region-based Fully Convolutional Network (R-FCN) • ROI coordinates [roiN, roiC, roiH, roiW]: 1 ≤ roiN ≤ 65535, roiC == 5, roiH == 1, and roiW == 1 • Dimensions of the input feature map: [xN, xC, xH, xW] pooledH == pooledW == group_size ≤ 128

No.	Operator	Description	Boundary
			pooledH and **pooledW** indicate the length and width of the pooled ROI. Output format: y [yN, yC, yH, yW] • 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	[Inputs] One input [Arguments] • **negative_slope**: (optional) float, default to **0** • **engine**: (optional) enum, default to **0**, CAFFE = 1, CUDNN = 2
30	Reorg	Real-time object detection	[Inputs] One input [Arguments] • **stride**: (optional) uint32, default to **2** • **reverse**: (optional) bool, default to false [Restrictions] Used only for YOLOv2
31	Reshape	Reshapes the input.	[Inputs] One input [Arguments] • **shape**: constant, int64 or int32 • **axis**: (optional) int32, default to **0** **num_axes**: (optional) int32, default to **–1**

No.	Operator	Description	Boundary
32	Reverse	Reversion	[Inputs] One input [Arguments] **axis**: (optional) int32, default to **1**. Controls the axis to be reversed. The content layout will not be reversed.
33	ROIAlign	Aggregates features using ROIs.	[Inputs] At least two inputs [Arguments] • **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** [Restrictions] Mainly used for Mask R-CNN Restrictions on the feature map: • N < 65535 • H * W ≤ 2464 • C ≤ 1152 • ((C – 1)/128 + 1) * pooledW ≤ 92 Restrictions on the ROI: • N < 65535 • C = 5(Caffe), H = 1, W = 1 • samplingRatio * pooledW ≤ 128, samplingRatio * pooledH ≤ 128 • H ≥ pooledH, W ≥ pooledW
34	ROIPooling	Maps ROI proposals to a feature map.	[Inputs] At least two inputs [Arguments] • **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. [Restrictions] Mainly used for Faster R-CNN

No.	Operator	Description	Boundary
			• Feature map size (H * W): input up to 3888, output up to 256
35	Scale	out = alpha * Input + beta	[Inputs] Two inputs [Arguments] • **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** [Restrictions] **shape** of **scale** and **bias**: (n, c, 1, 1), with the **C** dimension equal to that of the input
36	ShuffleChannel	Shuffles information across the feature channels.	[Inputs] One input [Arguments] **group**: (optional) uint32, default to **1**
37	Sigmoid	Activation function	[Inputs] One input [Arguments] **engine**: (optional) enum, default to **0**, CAFFE = 1, CUDNN = 2
38	Slice	Slices an input into multiple outputs.	[Inputs] One input [Arguments] • **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[Returns]
39	Softmax	Normalization logic function	[Inputs] One input [Arguments] • **engine**: (optional) default to **0**, CAFFE = 1, CUDNN = 2 • **axis**: (optional) int32, default to 1, indicating the axis along

No.	Operator	Description	Boundary
			which softmax is performed [Restrictions] **axis** is of the range [–rank, rank). Softmax can be performed on each of the four input dimensions (NCHW). • axis = 0: N ≤ 28544 • axis = 1 (channel dimension): C ≤ 11136 • axis = 2 (height dimension): W = 1, 0 < H < 16384 • axis = 3 (width dimension): 0 < W < 16384 If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968.
40	SSDDetectionOutput	SSD network detection output	[Inputs] Three inputs [Arguments] • **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
			• **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] • **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] (outputH * outputW)/(inputH * 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.	[Inputs] One input [Arguments] • **axis**: int32, axis to tile. For details about the value, see section 2.1 General Restrictions. • **tiles**: int32, replication times
45	Split (available since V320)	Splits an input Blob to multiple output Blobs.	[Inputs] One input Supported data types: float, double, int8, uint8, int32, uint32, int64, uint64, bool Supported data format: NCHW
46	BatchReindex (available since V320)	Selects, reorders, and replicates the input in a batch.	[Inputs] Two inputs • Input 0: float, input data • Input 1: uint32, input index [Restrictions] 0 ≤ input1 batch size < input0 batch size If the restriction is exceeded, the output result is unreliable.
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.	[Inputs] One input [Arguments] • **pyramid_height**: (mandatory) int32, pyramid height • **pool**: (optional) int32, either **0** (MAX) or **1** (AVG)
48	Threshold (available since	Tests whether the input exceeds a threshold: outputs **1** for	[Inputs] One input [Arguments] **threshold**: (optional) float32, default to **0.0**

No.	Operator	Description	Boundary
	V320)	inputs above threshold; **0** otherwise.	
49	MVN (available since V320)	The Mean-Variance Normalization (MVN) layer normalizes the input.	[Inputs] One input [Arguments] • **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.	[Inputs] One input
51	Swish (available since V320)	Swish takes one input data (Tensor) and produces one output data (Tensor) where the swish function, y = x * σ(scale * x), is applied to the tensor element-wise.	[Inputs] One input [Arguments] **beta**: float32, scaling factor of the swish function
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	[Inputs] Two inputs [Arguments] **axis**: scalar of type int, specifying the broadcast start. The value range is [–4, +3].

No.	Operator	Description	Boundary
		(determined by **axis**).	
53	Dropout (available since V310)	Avoids overfitting.	[Inputs] One input [Arguments] • **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] • **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] • **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: • NHWC + NHWC, NHWC + scalar

No.	Python API	C + + API	Boundary
			• 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) Note: The input sequence of the two Tensors is not fixed. [Returns] Tensor of the identical type as **y**
3	tf.add_n (available since V310)	AddN	[Arguments] • **inputs**: list of Tensor or IndexedSlices objects of type float32, with the same shape • **name**: (optional) string [Returns] Tensor with the identical shape as **inputs**
4	tf.batch_to_space_nd	BatchToSpaceND	[Arguments] • **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. [Restrictions] • 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: crop_start[i] + crop_end[i] < block_shape[i] * input_shape[i + 1] [Returns] Tensor of the identical type as **images**
5	tf.cast	Cast	[Inputs] [Arguments] • **x**: Tensor, must be one of the following types: float32, int32, bool, int64, int16, int8, uint8, uint16, double • **dtype**: destination type, same as the data type of **x**

No.	Python API	C + + API	Boundary
			• **name**: (optional) string [Returns] Tensor, SparseTensor, or IndexedSlices, same **dtype** and **shape** as the input
6	tf.math.ceil (available since V310)	Ceil	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
7	tf.clip_by_value	ClipByValue	[Arguments] • **t**: Tensor • **clip_value_min**: minimum value to clip by • **clip_value_max**: maximum value to clip by • **name**: (optional) string [Restrictions] The minimum value must be less than or equal to the maximum value. [Returns] Clipped Tensor. The return value range is [clip_value_min, clip_value_max].
8	tf.concat	ConcatV2	[Arguments] • **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. [Restrictions] • 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]. [Returns] Tensor, resulting from concatenation of the input Tensors

No.	Python API	C + + API	Boundary
9	tf.constant	Const	[Arguments] • **value**: constant value (or list) • **dtype**: data type of the resulting Tensor • **shape**: (optional) dimensions of the resulting Tensor • **name**: (optional) string **verify_shape**: (optional) Boolean that enables verification of a shape of values, default to **False** [Returns] One constant Tensor
10	tf.depth_to_space	DepthToSpace	[Arguments] • **input**: Tensor of type float32 • **block_size**: integer scalar, ≥ 2 • **data_format**: string, either **NHWC** (default) or **NCHW** • **name**: (optional) string [Restrictions] **blockSize** must be greater than or equal to **1**, and blockSize * blockSize must be exactly divided by **C**. [Returns] Tensor of the identical type as **input**
11	tf.equal	Equal	[Arguments] • **x**: Tensor, must be one of the following types: float32, uint8, int32, bool • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] 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. [Returns] Tensor of type bool
12	tf.exp	Exp	[Arguments] • **x**: Tensor of type float32 or double • **name**: (optional) string [Returns]

No.	Python API	C + + API	Boundary
			Tensor of the identical type as **x**
13	tf.math.expm1 (available since V310)	Expm1	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
14	tf.expand_dims	ExpandDims	[Arguments] • **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] • **images**: 4D Tensor of type float32 or uint8, with shape [batch, in_rows, in_cols, depth] • **ksizes**: 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] • **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** • **name**: (optional) string

No.	Python API	C + + API	Boundary
			[Restrictions] –65504 ≤ min ≤ +65504, –65504 ≤ max ≤ +65504 [Returns] Tensor of type float32
17	tf.fill	Fill	[Arguments] • **dims**: 1D Tensor of type int32 • **value**: variable of type int32 or float32 • **name**: (optional) string [Restrictions] 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). [Returns] Tensor of the identical type as **value**
18	tf.floormod	FloorMod	[Arguments] • **x**: Tensor of type float32 or int32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] 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. [Returns] Tensor of the identical type as **x**
19	tf.gather	Gather GatherV2	[Arguments] • **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 **name**: (optional) string [Returns]

No.	Python API	C + + API	Boundary
			Tensor of the identical type as **params**
20	tf.gather_nd	GatherNd	[Arguments] • **params**: Tensor, must be at least rank **axis + 1** • **indices**: Tensor of type int32 or int64 • **name**: (optional) string [Restrictions] • **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. The coordinates along the corresponding dimension of **indices** cannot exceed the dimension size. [Returns] Tensor of the identical type as **params**
21	tf.greater	Greater	[Arguments] • **x**: Tensor of type float32 or int32 • **y**: Tensor of type float32 or int32 • **name**: (optional) string [Restrictions] Only constant inputs are accepted. Broadcasting is supported. [Returns] Tensor of type bool
22	tf.image.crop_and_resize	CropAndResize	[Arguments] • **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** **name**: (optional) string

No.	Python API	C + + API	Boundary
			[Returns] Tensor of type float32
23	tf.image.non_max_suppression (available since V310)	NonMaxSuppressionV2	[Arguments] • **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 ≤ max_output_size
24	tf.image.resize_bilinear	ResizeBilinear	[Arguments] • **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] (outputH * outputW)/(inputH * inputW) > 1/7 [Returns] Tensor of type float, with the identical shape as the input
25	tf.image.resize_nearest_neighbor	ResizeNearestNeighbor	[Arguments] • **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] Tensor of type float, with the identical shape as the input

No.	Python API	C + + API	Boundary
26	tf.invert_permutation	InvertPermutation	[Arguments] • **x**: 1D Tensor of type int32 or int64 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
27	tf.keras.backend.hard_sigmoid	Hardsigmoid	[Arguments] **x**: Tensor [Returns] Output Tensor • If x < –2.5, returns **0**. • If x > 2.5, returns **1**. • If –2.5 ≤ x ≤ 2.5, returns **0.2 * x + 0.5**.
28	tf.keras.layers.ThresholdedReLU	ThresholdedReLU	[Arguments] **theta**: scalar of type float32, ≥ 0 [Restrictions] 0 ≤ theta ≤ 65504 [Returns] Tensor
29	tf.log	Log	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] x > 0 [Returns] Tensor of the identical type as **x**
30	tf.math.log1p (available since V310)	Log1p	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
31	tf.math.acos	Acos	[Arguments] • **x**: Tensor, must be one of type float32 • **name**: (optional) string

No.	Python API	C + + API	Boundary
			[Restrictions] The input data range is (–1 ≤ x ≤ +1), and the output data range is (0 ≤ y ≤ π). [Returns] Tensor of the identical type as **x**
32	tf.math.acosh	Acosh	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] x > 0 [Returns] Tensor of the identical type as **x**
33	tf.math.argmax	ArgMax	[Arguments] • **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) **name**: (optional) string [Returns] Tensor of the data type specified by **out_type**
34	tf.math.asin	Asin	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] The input data range is (–1 ≤ x ≤ +1), and the output data range is (–π/2 ≤ y ≤ +π/2). [Returns] Tensor of the identical type as **x**
35	tf.math.asinh	Asinh	[Arguments] • **x**: Tensor of type float32 **name**: (optional) string [Returns] Tensor of the identical type as **x**
36	tf.math.atan	Atan	[Arguments]

No.	Python API	C + + API	Boundary
			• **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] The input data range is (–65504 ≤ x ≤ +65504), and the output data range is (–π/2 ≤ y ≤ +π/2). [Returns] Tensor of the identical type as **x**
37	tf.math.atanh	Atanh	[Arguments] • **x**: Tensor of type float32**name**: (optional) string [Restrictions] Input data range: x is within (–1, 1) [Returns] Tensor of the identical type as **x**
38	tf.math.cosh	Cosh	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
39	tf.math.floor (available since V310)	Floor	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] 8-bit quantization is not supported. [Returns] Tensor of the identical type as **x**
40	tf.math.greater_equal	GreaterEqual	[Arguments] • **x**: Tensor of type float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] Input data range: –65504 ≤ x ≤ +65504 [Returns] Tensor of type bool
41	tf.math.less	Less	[Arguments]

No.	Python API	C + + API	Boundary
			• **x**: Tensor of type float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Returns] Tensor of type bool
42	tf.math.logical_and	LogicalAnd	[Arguments] • **x**: non-constant Tensor of type bool • **y**: non-constant Tensor of type bool • **name**: (optional) string [Restrictions] 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] [Returns] Tensor of type bool
43	tf.math.logical_not	LogicalNot	[Arguments] • **x**: Tensor of type bool • **name**: (optional) string [Returns] Tensor of type bool
44	tf.math.logical_or	LogicalOr	[Arguments] • **x**: non-constant Tensor of type bool • **y**: non-constant Tensor of type bool • **name**: (optional) string [Restrictions] 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. [Returns] Tensor of type bool
45	tf.math.maximum	Maximum	[Arguments] • **x**: Tensor of type int32 or float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string

No.	Python API	C + + API	Boundary
			[Restrictions] None [Returns] Tensor. Returns the max of **x** and **y** (x > y ? x: y), of identical type as **x**
46	tf.math.minimum	Minimum	[Arguments] • **x**: Tensor of type int32 or float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] None [Returns] Tensor. Returns the min of x and y (x < y ? x : y), of identical type as **x**
47	tf.math.negative	Neg	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] The input data range is –65504 ≤ x ≤ +65504, and the output data range is –65504 ≤ y ≤ +65504. [Returns] Tensor. Returns **–x**.
48	tf.math.pow	Power	[Arguments] • **x**: Tensor of type float32 • **y**: Tensor of type float32 • **name**: (optional) string [Restrictions] None [Returns] Tensor
49	tf.math.reciprocal	Reciprocal	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Restrictions] The input data cannot contain **0**.

No.	Python API	C + + API	Boundary
			[Returns] Tensor of the identical type as **x**
50	tf.math.reduce_all	All	[Arguments] • **input_tensor**: Tensor of type bool • **axis**: dimension to reduce • **keepdims**: scalar of type bool • **name**: (optional) string • **reduction_indices**: (deprecated) equivalent to **axis** **keep_dims**: (deprecated) equivalent to **keepdims** [Returns] Tensor of the identical type as **input_tensor**
51	tf.math.reduce_max	ReduceMax	[Arguments] • **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** [Restrictions] **axis** is of the range [–rank, rank). [Returns] Tensor of the identical type as **input_tensor**
52	tf.math.reduce_mean (available since V310)	Mean	[Arguments] • **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** [Restrictions] • **axis** is of the range [–rank, rank). • Dimension reduction support of **axis**: 1D: All scenarios are supported. 2D: Only 1H and C1 scenarios are supported, where, 0 and

No.	Python API	C + + API	Boundary
			1 indicate C and H, respectively. 3D: Only C11, CH1, C1W, 111, 1H1, and 1HW scenarios are supported, where, 0, 1, and 2 indicate C, H, and W, respectively. 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. [Returns] Reduced Tensor of the identical type as **input_tensor**
53	tf.math.reduce_min	Min	[Arguments] • **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** [Restrictions] • When the input Tensor has four dimensions: input axis = {3,{1,2,3}}, keepdims = true, H * W ≤ 512 • When the input Tensor has two dimensions: input axis = {1,{1}}, keepdims = true, H * W * ALIGN(C, 16)≤ 8192 [Returns] Reduced Tensor of the identical type as **input_tensor**
54	tf.math.reduce_prod	Prod	[Arguments] • **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** [Restrictions] • If the input tensor is 4-dimensional: axis = {3, {1,2,3}}, keepdims = true, H x W ≤ 512 • If the input tensor is 2-dimensional: axis={1,{1}}, keepdims=true, H*W*ALIGN(C, 16) ≤ 8192

No.	Python API	C + + API	Boundary
			[Returns] Tensor of the identical type as **input_tensor**
55	tf.math.rint	Rint	[Arguments] • **x**: Tensor of type float32 **name**: (optional) string [Returns] Tensor of the identical type and shape as **x**
56	tf.math.round	Round	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type and shape as **x**
57	tf.math.rsqrt	Rsqrt	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
58	tf.math.sinh	Sinh	[Arguments] • **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] • **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] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
61	tf.math.sqrt	Sqrt	[Arguments]

No.	Python API	C + + API	Boundary
			• **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
62	tf.math.squared_difference	SquaredDifference	[Arguments] • **x**: Tensor, must be one of the following types: float32, int64, int32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] 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] [Returns] Tensor of the identical type as **x**
63	tf.math.tan	Tan	[Arguments] • **x**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
64	tf.math.top_k	TopKV2	[Arguments] • **input**: n-D Tensor (N ≥ 1) with the last dimension at least **k** • **k**: scalar of type int32, ≥ 1 • **sorted**: bool • **name**: (optional) string [Restrictions] **k** must be a constant. [Returns] • **values**: Tensor, indicating **k** largest elements along each last dimensional slice • **indices**: Tensor, indicating the indices of **values** of **input**
65	tf.matmul	MatMul	[Arguments] • **a**: Tensor of type float32, 2 ≤ rank ≤ 4

No.	Python API	C + + API	Boundary
			• **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 [Restrictions] • 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**. [Returns] Tensor of the identical type as **a** and **b**
66	tf.multinomial	Multinomial	[Arguments] • **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 [Restrictions] When **seed** is **0**, the generated random is dynamic. [Returns] Tensor with shape [batch_size, num_samples]
67	tf.math.multiply	Multiply	[Arguments] • **x**: Tensor of type float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions]

No.	Python API	C + + API	Boundary
			If the two inputs have inconsistent dimensions, broadcasting (that is, dimension padding) is performed. Broadcasting is supported only in the following scenarios: • 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) Note: The input sequence of the two Tensors is not fixed. [Returns] Tensor
68	tf.nn.avg_pool	AvgPool	[Arguments] • **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 [Restrictions] • KernelH < 256, kernelW < 256 • If H and W of the output tensor are 1: input H * input W < 65536 • kernelH ≤ inputH + padTop + padBottom • kernelW ≤ inputW + padLeft + padRight • padTop < windowH • padBottom < windowH • padLeft < windowW • padRight < windowW [Returns] Tensor of the identical type as **value**
69	tf.nn.bias_add	BiasAdd	[Arguments]

No.	Python API	C + + API	Boundary
			• **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 [Restrictions] • C ≤ 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. [Returns] Tensor of the identical type as **value**
70	tf.nn.conv2d	Conv2D	[Arguments] • **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 [Restrictions] • (inputW + padWHead + padWTail) ≥ (((FilterW – 1) * dilationW) + 1) • (inputW + padWHead + padWTail)/StrideW + 1 ≤ INT32_MAX • (inputH + padHHead + padHTail) ≥ (((FilterH – 1) *

No.	Python API	C + + API	Boundary
			dilationH) + 1) • (inputH + padHHead + padHTail)/StrideH + 1 ≤ INT32_MAX • 0 ≤ Pad < 256, 0 < FilterSize < 256, 0 < Stride < 64, 1 ≤ dilationsize < 256 • StrideW ≤ (inputW + padW) – ((filterW – 1) * dilationW) + 1) [Returns] Tensor of the identical type as **value**
71	tf.nn.conv2d_transpose	Conv2DBackpropInput	[Arguments] • **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 [Restrictions] • filterH – padHHead – 1 ≥ 0 • filterW – padWHead – 1 ≥ 0 Restrictions involving intermediate variables: • a = ALIGN(filter_num, 16) * ALIGN(filter_c, 16) * filter_h * filter_w * 2 • If ALIGN(filter_c, 16)%32 = 0, a = a/2 • conv_input_width = (deconvolution input W – 1) * strideW + 1 • b = (conv_input_width) * filter_h * ALIGN(filter_num, 16) * 4 • a + b ≤ 512 KB [Returns] Tensor of the identical type as **value**
72	tf.nn.depthwis	DepthwiseC	[Arguments]

No.	Python API	C + + API	Boundary
	e_conv2d	onv2dNative	• **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 [Restrictions] • filterN = inputC = group • StrideW ≤ (inputW + padW) – ((filterW – 1) * dilationW) + 1) [Returns] 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
73	tf.nn.elu	Elu	[Arguments] • **features**: Tensor • **name**: (optional) string [Returns] Tensor of the identical type as **features**
74	tf.nn.fused_batch_norm	FusedBatchNorm	[Arguments] • **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 • **epsilon**: small float number added to the variance of **x** • **data_format**: data format for **x**, either **NHWC** (default) or

No.	Python API	C + + API	Boundary
			NCHW • **is_training**: bool, specifying whether the operation is used for training or inference • **name**: (optional) string [Restrictions] The shape of **scale**, **bias**, **mean**, and **var** must be (1, C, 1, 1), with the same C dimension as input. [Returns] • **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	[Arguments] • **x**: 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** [Restrictions] H * W * 2 < 32768 [Returns] Tensor of the identical type as **x**
76	tf.nn.leaky_relu	LeakyRelu	[Arguments] • **features**: Tensor of type float32 • **alpha**: slope of the activation function at x < 0 • **name**: (optional) string [Returns] Activation value
77	tf.nn.log_softmax (available since V310)	LogSoftmax	[Arguments] • **logits**: non-null Tensor of type float32 • **axis**: dimension softmax would be performed on, default to **–1**, indicating the last dimension • **name**: (optional) string

No.	Python API	C + + API	Boundary
			• **dim**: (deprecated) equivalent to axis [Restrictions] **axis** is of the range [–rank, rank). Softmax can be performed on each of the four input dimensions (NCHW). • axis = 0: not supported • axis = 1 (channel dimension): C ≤ 11136 • axis = 2 (height dimension): W = 1, 0 < H < 16384 • axis = 3 (width dimension): 0 < W < 16384 [Returns] Tensor of the identical type and shape as **logits**
78	tf.nn.lrn	Local response normalization (LRN)	[Arguments] • **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 [Restrictions] • **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. [Returns] Tensor of the identical type as **input**
79	tf.nn.max_pool	MaxPool	Same as **tf.nn.avg_pool**
80	tf.nn.relu	Relu	[Arguments] • **features**: Tensor, must be one of the following types: float16, uint8, quint8 • **name**: (optional) string [Returns]

No.	Python API	C + + API	Boundary
			Tensor of the identical type as **features**
81	tf.nn.relu6	Relu6	[Arguments] • **features**: Tensor of type float16 or float32 • **name**: (optional) string [Returns] Tensor of the identical type as **features**
82	tf.nn.selu	Selu	[Arguments] • **features**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **features**
83	tf.nn.softmax	Softmax	[Arguments] • **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** [Restrictions] **axis** is of the range [–rank, rank). Softmax can be performed on each of the four input dimensions (NCHW). • axis = 0: N ≤ 28544 • axis = 1 (channel dimension): C ≤ 11136, H * W < 65536 • axis = 2 (height dimension): W = 1, 0 < H < 16384 • axis = 3 (width dimension): 0 < W < 16384 If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968. [Returns] Tensor of the identical type and shape as **logits**
84	tf.nn.softplus	Softplus	[Arguments] • **features**: Tensor of type float32 • **name**: (optional) string [Returns]

No.	Python API	C + + API	Boundary
			Tensor of the identical type as **features**
85	tf.nn.softsign	Softsign	[Arguments] • **features**: Tensor of type float32 • **name**: (optional) string [Returns] Tensor of the identical type as **features**
86	tf.pad	Pad PadV2 MirrorPad	[Arguments] • **tensor**: 4D Tensor of type float32 or int32 • **paddings**: constant Tensor of type int32 • **mode**: string, one of **CONSTANT**, **REFLECT**, or **SYMMETRIC** 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. • **name**: (optional) string • **constant_values**: scalar pad value to use, of the identical type as **tensor** [Restrictions] If **mode** is **CONSTANT**: 0 ≤ PAD ≤ 128, 0 < W ≤ 3000 [Returns] Tensor of the identical type as **tensor**
87	tf.placeholder	Placeholder	[Arguments] • **dype**: (mandatory) data type • **shape**: (mandatory) shape of the tensor • **name**: (optional) string [Returns] Tensor
88	tf.range	Range	[Arguments] • **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 **name**: (optional) string [Returns]

No.	Python API	C + + API	Boundary
			1D Tensor
89	tf.realdiv	RealDiv	[Arguments] • **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] • **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] • **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] • **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] • **input**: Tensor • **seq_lengths**: 1D Tensor of type int32 or int64

No.	Python API	C + + API	Boundary
			• **seq_axis**: scalar of type integer • **batch_axis**: integer scalar, default to **0** • **name**: (optional) string [Returns] Tensor of the identical type as **input**
94	tf.scatter_nd (available since V310)	ScatterNd	[Arguments] • **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 [Returns] Tensor of the identical type as **updates**
95	tf.shape	Shape	[Arguments] • **input**: Tensor • **name**: (optional) string • **out_type**: data type for the output Tensor, either int32 (default) or int64 [Returns] Tensor of the data type specified by **out_type**
96	tf.sigmoid	Sigmoid	[Arguments] • **x**: Tensor • **name**: (optional) string [Returns] Tensor of the identical type as **value**
97	tf.sign (available since V310)	Sign	[Arguments] • x: Tensor of type float32 • name: (optional) string [Returns] Tensor of the identical type as **x**
98	tf.size	Size	[Arguments] • **input**: Tensor of type float32 • **name**: (optional) string

No.	Python API	C + + API	Boundary
			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] • **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] • **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 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] • 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] • **input**: Tensor of type float32 • **block_size**: scalar of type int32, ≥ 2 • **data_format**: (optional) string, **NHWC** (default) or **NCHW** • **name**: (optional) string [Returns] Tensor of the identical type as **input**

No.	Python API	C + + API	Boundary
102	tf.split	Split SplitV	[Arguments] • **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 **name**: (optional) string [Returns] List of Tensor objects resulting from splitting
103	tf.math.square	Square	[Arguments] • **x**: Tensor of type float32 or int32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
104	tf.squeeze	Squeeze	[Arguments] • **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** [Returns] Tensor, with the same data and type as **input**, but has one or more dimensions of size **1** removed.
105	tf.stack	Pack	[Arguments] • **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 [Returns] Tensor of the identical type as **values**
106	tf.strided_slice	StridedSlice	[Arguments] • **input_**: Tensor of type float32

No.	Python API	C + + API	Boundary
			• **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 [Restrictions] strides ≠ 0 [Restrictions] • In the **shrink_axis_mask** scenario, only a positive mask is supported. • **new_axis_mask** is not supported. [Returns] Tensor of the identical type as **input_**
107	tf.subtract	Subtract	[Arguments] • **x**: Tensor of type float32 • **y**: Tensor of the identical type as **x** • **name**: (optional) string [Restrictions] If the two inputs have inconsistent dimensions, broadcasting (that is, dimension padding) is performed. Broadcasting is supported only in the following scenarios: • 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) Note: The input sequence of the two Tensors is not fixed. [Returns] Tensor
108	tf.math.tanh	Tanh	[Arguments]

No.	Python API	C + + API	Boundary
			• **x**: Tensor of type float16 or float32 • **name**: (optional) string [Returns] Tensor of the identical type as **x**
109	tf.tile	Tile	[Arguments] • **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 [Returns] Tensor
110	tf.transpose	Transpose	[Arguments] • **a**: Tensor • **perm**: permutation of the dimensions of **a** • **name**: (optional) string • **conjugate**: (optional) bool, default to and must be **False**. [Returns] Transposed Tensor
111	tf.unstack	Unpack	[Arguments] • **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 [Returns] List of Tensor objects unstacked from **value**
112	tf.where	Where	[Arguments] • **condition**: Tensor of type bool • **x**: None • **y**: None • **name**: (optional) string [Returns] Tensor with 2-dimensional shape, where the first dimension

No.	Python API	C + + API	Boundary
			represents the number of true elements
113	tf.where	Select	[Arguments] • **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 [Restrictions] • **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**. [Returns] Tensor of the identical shape and data type as **x**
114	from tensorflow.python.ops import control_flow_ops control_flow_ops.switch (available since V310)	Switch	[Arguments] • **data**: Tensor • **pred**: scalar of type bool • **dtype**: (optional) data type of the output Tensor • **name**: (optional) string [Restrictions] This operator must be used in conjunction with operator Merge. For details, see operator Merge. [Returns] Tensor (output_false, output_true): If **pred** is **true**, data will be forwarded to **output_true**, otherwise it goes to **output_false**.
115	from tensorflow.python.ops import control_flow_ops control_flow_ops.merge (available since V310)	Merge	[Arguments] • **inputs**: Tensors, at least one of which is valid • **name**: (optional) string [Restrictions] This operator must be used in conjunction with Switch, and Merge cannot be the final output. [Returns] Tuple containing the selected tensors and their indexing
11	tf.matmul	BatchMatM	[Arguments]

No.	Python API	C + + API	Boundary
6	(available since V320)	ul	• 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. [Restrictions] • Tensor of type float32 • **adj_x**: Default to and must be **false**. • **adj_y**: Default to **false**. [Returns] Tensor of the identical type as **x** and **y**
117	tf.contrib.layers.layer_norm (available since V320)		[Arguments] • **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**. [Returns] Tensor of the identical **shape** and **dtype** as **inputs**,

No.	Python API	C + + API	Boundary
			normalized CHW within the range of N of inputs.
118	tf.stop_gradient (available since V320)	StopGradient	[Arguments] • **input**: input Tensor [Returns] Tensor of the identical type as **input**
119	tf.contrib.layers.instance_norm (available since V320)		[Arguments] • **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**. [Returns] Tensor of the identical **shape** and **dtype** as **inputs**, normalized HW within the range of NC of **inputs**.
120	tf.random.normal (available since V320)	RandomNormal	[Arguments] • **shape**: constant, 1D tensor of type int32, shape of the output tensor • **mean**: 0D scalar of type float16, mean of the normal distribution. • **stddev**: 0D scalar of type float16, standard deviation of the

No.	Python API	C + + API	Boundary
			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. [Returns] Tensor of type float16
121	tf.random.shuffle (available since V320)	RandomShuffle	[Arguments] • **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. [Returns] Tensor of the identical type as **value**
122	tf.random.uniform (available since V320)	RandomUniformInt	[Arguments] • **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. • **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

No.	Python API	C + + API	Boundary
			0, the computation result is different each time. In this version, the **seed2** value is ignored and **0** is used. [Returns] Tensor of type int32
123	tf.random.uniform (available since V320)	RandomUniform	[Arguments] • **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. [Returns] Tensor of type float32
124	tf.math.argmin (available since V320)	ArgMin	[Arguments] • **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 [Returns] Tensor of type **out_type**
125	tf.rank (available since V320)	Rank	[Arguments] • **input**: constant or non-constant, Tensor, must be one of the following data types: int32, float32, uint8, bool • **name**: (optional) string [Returns] Tensor of type int32

No.	Python API	C + + API	Boundary
126	tf.truncatemod (available since V320)	Truncatemod	[Arguments] • **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] • **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] • **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**
129	tf.math.cumprod (available since V320)	Cumprod	[Arguments] • **x**: Tensor of type float32, uint32, or uint8 • **rank**: Tensor of type int32, Must be in the range [–rank(x),

No.	Python API	C + + API	Boundary
			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 [Returns] Tensor of the identical type as **x**
130	tf.nn.conv1d (available since V320)		[Arguments] • **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 [Returns] Tensor of the identical type as **value**
131	tf.nn.atrous_conv2d (available since V320)		[Arguments] • **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 [Returns] Tensor of the identical type as **value**
132	tf.math.reduce_any (available since V320)	Any	[Arguments] • **input_tensor**: Tensor of type bool • **axis**: dimensions to reduce. Must be in the range [–rank(input_tensor), rank(input_tensor)]. • **keepdims**: scalar of type bool, default to **false**. • **name**: (optional) string

No.	Python API	C + + API	Boundary
			• **reduction_indices**: (deprecated) equivalent to **axis** • keep_dims: (deprecated) equivalent to **keepdims** [Returns] Tensor of type bool
133	tf.math.logical_xor (available since V320)		[Arguments] • **x**: constant or non-constant, Tensor of type bool • **y**: constant or non-constant, Tensor of type bool • **name**: (optional) string [Restrictions] Bidirectional broadcast is not supported. [Returns] Tensor of the identical type as the inputs
134	tf.nn.fractional_max_pool (available since V320)	FractionalMaxPool	[Arguments] • **value**: 4D Tensor of type flaot32, 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 [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 are the same each time. [Returns]

No.	Python API	C + + API	Boundary
			• 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	[Arguments] • **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 [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 are the same each time. [Returns] • 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	[Arguments] • **x**: Tensor of type float32 • **y**: Tensor of type float32 [Returns] Tensor of type bool
13	tf.math.less_e	LessEqual	[Arguments]

No.	Python API	C + + API	Boundary
7	qual (available since V320)		• **x**: Tensor of type float32 • **y**: Tensor of type float32 [Returns] Tensor of type bool
138	tf.quantization.quantize (available since V320)	QuantizeV2	[Arguments] • **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**. [Returns] Tensor of type uint8
139	tf.quantization.dequantize (available since V320)	Dequantize	[Arguments] • **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**. [Returns] Tensor of type float32
140	tf.math.floor_div (available since V320)	FloorDiv	[Arguments] • **x**: Tensor of type int32, float, or uint8 • **y**: Tensor of type int32, float, or uint8 [Returns] Tensor of the identical shape and data type as **x**
14	tf.quantization.fake_quant_	FakeQuantWithMinMa	[Arguments]

No.	Python API	C + + API	Boundary
1	with_min_max_vars_per_channel (available since V320)	xVarsPerChannel	• **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**. [Returns] Tensor of the identical shape and data type as **x**
142	tf.one_hot (available since V320)	OneHot	[Arguments] • **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 [Returns] One-hot tensor, of one of the following data types: uint8, int32, float, double, bool
143	tf.math.segment_max (available since V320)	SegmentMax	[Arguments] • **data**: Tensor of type float32 • **segment_ids**: constant, 1D tensor of type int32, sorted by ID [Restrictions] • 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. [Returns] Tensor
14	tf.math.segm	SegmentMi	[Arguments]

No.	Python API	C + + API	Boundary
4	ent_min (available since V320)	n	• **data**: Tensor of type float32 • **segment_ids**: constant, 1D tensor of type int32, sorted by ID [Restrictions] • 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. [Returns] Tensor
145	tf.math.segment_mean (available since V320)	SegmentMean	[Arguments] • **data**: Tensor of type float32 • **segment_ids**: constant, 1D tensor of type int32, sorted by ID [Restrictions] • 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. [Returns] Tensor
146	tf.math.segment_ prod (available since V320)	SegmentProd	[Arguments] • **data**: Tensor of type float32 • **segment_ids**: constant, 1D tensor of type int32, sorted by ID [Restrictions] • 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. [Returns] Tensor
147	tf.math.segment_sum (available since V320)	SegmentSum	[Arguments] • **data**: Tensor of type float32 • **segment_ids**: constant, 1D tensor of type int32, sorted by ID

No.	Python API	C + + API	Boundary
			[Restrictions] • 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. [Returns] Tensor
148	tf.zeros_like (available since V310)	ZerosLike	[Arguments] **x**: constant or 0D–4D Tensor of type float32 [Returns] Tensor of the identical shape and data type as **x**
149	tf.identity (available since V310)	Identity	[Arguments] **x**: Tensor [Returns] Tensor of the identical shape and data type as **x**
150	tf.Assert (available since V310)	Assert	[Arguments] • **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**. [Restrictions] 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])**. [Returns] Operation that, when executed, raises a **tf.errors.InvalidArgumentError** if **condition** is **false**.
151	tf.keras.layers.PReLU (available since V310)		[Arguments] • **x**: Tensor of type float • **slope**: trained slope [Returns] Tensor of the identical shape and data type as **x**
152	GELU(x)=0.5x(1+tanh[((2/π)^0.5)*(x+0.0		[Arguments] • **features**: Tensor of type float32

No.	Python API	C + + API	Boundary
	44715x^3)]) (available since V320)		• **name**: (optional) string [Returns] Tensor of the identical type as **features**
153	tf.nn.bidirectional_dynamic_rnn (available since V320)		[Inputs] • 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] • 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. • The **initial_state_fw** and **initial_state_bw** parameters for dynamic RNN must be **none**.

No.	Python API	C + + API	Boundary
			• **state_is_tuple** must be **True**. • The value of **num_units** of BasicLSTMCell must be an integer multiple of 16. [Returns] • 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)		[Inputs] • **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** [Restrictions] • 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. • The **initial_state_fw** and **initial_state_bw** parameters for static RNN must be **none**. • **state_is_tuple** must be **True**.

No.	Python API	C + + API	Boundary
			• The value of **num_units** of BasicLSTMCell must be an integer multiple of 16. [Returns] • 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	ANEURALNETWORKS_ABS (available since V310)	Computes the absolute value of a tensor, element-wise.	[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: tensor [Restrictions] The last dimension of input 0 must not exceed 400. [Returns] Output 0: Tensor with the identical shape as input 0
2	ANEURALNETWORKS_ADD	Adds two tensors.	[Inputs] • 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 FuseCode{ ANEURALNETWORKS_FUSED_NONE = 0,

No.	Operation	Description	Boundary
			ANEURALNETWORKS_FUSED_RELU = 1, ANEURALNETWORKS_FUSED_RELU1 = 2, ANEURALNETWORKS_FUSED_RELU6 = 3 } [Restrictions] None [Returns] Output 0: Tensor of identical **OperandCode** as input 0
3	ANEURALNETWORKS_ARGMAX (available since V310)	Returns the index of the largest element along an axis.	[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 [Restrictions] None [Returns] Output 0: (N – 1)D tensor of type TENSOR_INT32
4	ANEURALNETWORKS_ARGMIN (available since V310)	Returns the index of the smallest element along an axis.	[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 [Restrictions] None [Returns] Output 0: (N – 1)D tensor of type TENSOR_INT32
5	ANEURALNETWORKS_AVERAGE_POOL_2D	Average pooling	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM

No.	Operation	Description	Boundary
			Supported tensor rank: 4 NHWC data layout supported [Inputs (explicit padding)] • 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 ≤ Pad < 256 • Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, 0 ≤ Pad < 256 • Input 3: scalar of type INT32, specifying the padding on the top, in the height dimension, 0 ≤ Pad < 256 • Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, 0 ≤ Pad < 256 • Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, 0 < Stride < 64 • Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, 0 < 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) [Inputs (implicit padding)] • 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**) PaddingCode{ ANEURALNETWORKS_PADDING_SAME = 0, ANEURALNETWORKS_PADDING_VALID = 1 } • 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 <

No.	Operation	Description	Boundary
			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) [Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth] [Restrictions] • KernelH < 256, kernelW < 256 • If H and W of the output tensor are 1: input H * input W < 65536
6	ANEURALNETWORKS_BATCH_TO_SPACE_ND	BatchToSpace for n-D tensors	[Inputs] • 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) [Restrictions] The product of the dimensions of input 0 must be less than **500**. [Returns] Output 0: Tensor of identical **OperandCode** as input 0
7	ANEURALNETWORKS_CAST (available since V310)	Casts a tensor to a new type.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4

No.	Operation	Description	Boundary
			• Input 0: n-D tensor, where N is within [1, 4] [Returns] Output 0: n-D tensor
8	ANEURALNETWORKS_CONCATENATION	Concatenates the input along the given dimension.	[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 • 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 [Restrictions] 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 [2, 32]. Inputs 0–(n – 1) do not accept constant inputs. Only QUANT8 inputs are accepted. axis ≠ 2 [Returns] Output 0: Tensor of identical **OperandCode** as the input Tensors, with shape [D0, D1, ..., sum(Daxis(i)), ..., Dm]
9	ANEURALNETWORKS_CONV_2D	Convolves the input.	[Inputs] 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 [Inputs (explicit padding)] • 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 < 256

No.	Operation	Description	Boundary
			• 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 ≤ Pad < 256 • Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension, 0 ≤ Pad < 256 • Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension, 0 ≤ Pad < 256 • Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension, 0 ≤ Pad < 256 • Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension, 0 < Stride < 64 • Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension, 0 < 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. [Inputs (implicit padding)] • 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 < 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

No.	Operation	Description	Boundary
			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 < Stride < 64 • Input 5: scalar of type INT32, specifying the stride when walking through input in the height dimension, 0 < 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. [Restrictions] When the input type is QUANT8, the data layout of input 0 and output 0 must be NHWC. [Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth_out]
10	ANEURALNETWORKS_DEPTH_TO_SPACE	Rearranges data from depth into blocks of spatial data.	[Inputs] • 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) [Restrictions] None

No.	Operation	Description	Boundary
			[Returns] Output 0: 4D Tensor, with shape [batch, height * block_size, width * block_size, depth/(block_size * block_size)]
11	ANEURALNETWORKS_DEPTHWISE_CONV_2D	Depthwise convolution	[Inputs] • 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 [Inputs (explicit padding)] • 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 < 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 ≤ Pad < 256 • Input 4: scalar of type INT32, specifying the padding on the right, in the width dimension, 0 ≤ Pad < 256 • Input 5: scalar of type INT32, specifying the padding on the top, in the height dimension, 0 ≤ Pad < 256 • Input 6: scalar of type INT32, specifying the padding on the bottom, in the height dimension, 0 ≤ Pad < 256 • Input 7: scalar of type INT32, specifying the stride when walking through input in the width dimension, 0 < Stride < 64 • Input 8: scalar of type INT32, specifying the stride when walking through input in the height dimension, 0 < 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**. Set to **true** to specify NCHW data layout for input 0 and

No.	Operation	Description	Boundary
			output 0. • 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. [Inputs (implicit padding)] • 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 < 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 < Stride < 64 • Input 5: scalar of type INT32, specifying the stride when walking through input in the height dimension, 0 < 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. [Restrictions] • filterN = inputC = group

No.	Operation	Description	Boundary
			• StrideW ≤ (inputW + padW) – ((filterW – 1) * dilationW) + 1) • When the input type is QUANT8, the data layout of input 0 and output 0 must be NHWC. [Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth_out].
12	ANEURALNETWORKS_DEQUANTIZE	Dequantizes the input tensor.	[Inputs] • Supported tensor **OperandCode**: TENSOR_QUANT8_ASYMM • Supported output tensor **OperandCode**: TENSOR_FLOAT32 TENSOR_FLOAT16 Supported tensor rank: up to 4 • Input 0: tensor [Returns] Output 0: Tensor of type TENSOR_FLOAT32 and the identical shape as input 0
13	ANEURALNETWORKS_DIV	Divides tensors.	[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: 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 [Restrictions] None [Returns] Output 0: Tensor of identical **OperandCode** as input 0
14	ANEURALNETWORKS_EMBEDDING_LOOKUP	Looks up sub-tensors in the input tensor.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_QUANT8_ASYMM

No.	Operation	Description	Boundary
			Supported value tensor rank: from 2 • Input 0: **Lookups**, 1D Tensor of type TENSOR_INT32, with shape [k] • Input 1: **Values**, n-D Tensor (N ≥ 2), from which sub-tensors are extracted [Returns] Output 0: n-D tensor with the same rank and shape as the **Values** tensor, except for the first dimension which has the same size as Lookups' only dimension. For a TENSOR_QUANT8_ASYMM tensor, its **scale** and **zeroPoint** must be the same as those of input 1.
15	ANEURALNETWORKS_EQUAL (available since V310)	For input tensors x and y, computes x == y element-wise.	[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: Tensor of identical **OperandCode** and dimensions compatible with input 0 [Restrictions] • 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
16	ANEURALNETWORKS_EXP (available since V310)	Computes exponential of x element-wise.	[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] [Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0
17	ANEURALNETWORKS_FL	Computes floor() on the	[Inputs] • Supported tensor **OperandCode**:

No.	Operation	Description	Boundary
	OOR	input tensor.	TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 • Input 0: tensor [Restrictions] Quantization is not supported. [Returns] Output 0: Tensor of identical **OperandCode** and dimensions as input 0
18	ANEURALNETWORKS_FULLY_CONNECTED	Computes an inner product.	[Inputs] • 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 [Returns] 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.
19	ANEURALNETWORKS_GREATER (available	For input tensors x and y, computes x > y element-wise.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29)

No.	Operation	Description	Boundary
	since V310)		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] • 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_BOOL8 Tensor
20	ANEURALNETWORKS_GREATER_EQUAL (available since V310)	For input tensors x and y, computes x ≥ y element-wise.	[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: Tensor of identical **OperandCode** as input 0 [Restrictions] • 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
21	ANEURALNETWORKS_HASHTABLE_LOOKUP	Looks up sub-tensors in the input tensor using a key-value map.	[Inputs] • 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*th element in **Keys** (**Keys[*i*]**) is the key to select the *i*th sub-tensor in Values (**Values[*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**.

No.	Operation	Description	Boundary
			[Returns] • 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. [Restrictions] ALIGN(Lookups, 32) * 2 + ALIGN(Keys, 32) + ALIGN(Values, 256) * 32 ≤ 49152
22	ANEURALNETWORKS_L2_NORMALIZATION	Applies L2 normalization along the depth dimension.	[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: (optional), scalar of type IINT32, specifying the dimension normalization would be performed on. (available since API 29) [Returns] Output 0: tensor
23	ANEURALNETWORKS_L2_POOL_2D	Performs L2 pooling.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 NHWC data layout supported [Inputs (explicit padding)] • 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 ≤ Pad < 256 • Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, 0 ≤ Pad < 256 • Input 3: scalar of type INT32, specifying the padding on the top, in the height dimension, 0 ≤ Pad < 256 • Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, 0 ≤ Pad < 256

No.	Operation	Description	Boundary
			• Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, strideW < 64 • Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, 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. Only the default format **NHWC** is supported. (available since API 29) [Inputs (implicit padding)] • 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) [Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth].
24	ANEURALNETWORKS_LESS (available since V310)	For input tensors x and y, computes x < y element-wise.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4

No.	Operation	Description	Boundary
			• Input 0: n-D tensor, where N is within [1, 4] • Input 1: Tensor of identical **OperandCode** as input 0 [Restrictions] • 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
25	ANEURALNETWORKS_LESS_EQUAL (available since V310)	For input tensors x and y, computes x ≤ y element-wise.	[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: Tensor of identical **OperandCode** as input 0 [Restrictions] • 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] • 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.

No.	Operation	Description	Boundary
			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**). [Returns] Output 0: Tensor with the identical shape as input 0
27	ANEURALNETWORKS_LOG (available since V310)	Computes natural logarithm of x element-wise.	[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] [Returns] Output 0: Tensor with the identical shape as input 0
28	ANEURALNETWORKS_LOGICAL_AND (available since V310)	Returns the truth value of x AND y element-wise.	[Inputs] • Supported tensor **OperandCode**: TENSOR BOOL8 Supported tensor rank: up to 4 • Input 0: tensor • Input 1: Tensor with the identical shape as input 0 [Returns] Output 0: tensor
29	ANEURALNETWORKS_LOGICAL_NOT (available since V310)	Computes the truth value of NOT x element-wise.	[Inputs] • Supported tensor **OperandCode**: TENSOR BOOL8 Supported tensor rank: up to 4 • Input 0: tensor [Returns] Output 0: tensor

No.	Operation	Description	Boundary
30	ANEURALNETWORKS_LOGICAL_OR (available since V310)	Computes sigmoid activation on the input tensor element-wise.	[Inputs] • Supported tensor **OperandCode**: TENSOR BOOL8 Supported tensor rank: up to 4 • Input 0: tensor • Input 1: Tensor with the identical shape as input 0 [Returns] Output 0: tensor
31	ANEURALNETWORKS_LOGISTIC	Logistic activation	[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: tensor [Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0. For a TENSOR_QUANT8_ASYMM Tensor, scale = 1.f/256 and zeroPoint = 0.
32	ANEURALNETWORKS_LOG_SOFTMAX (available since V310)	Computes the log softmax activations given logits.	[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: 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 [Restrictions] **axis** is of the range [–rank, rank). Softmax can be performed on each of the four input dimensions (NCHW).

No.	Operation	Description	Boundary
			• axis = 0: not supported • axis = 1 (channel scenario): C ≤ 11136 • axis = 2 (height dimension): W = 1, 0 < H < 16384 • axis = 3 (width dimension): 0 < W < 16384 [Returns] Output 0: Tensor of identical **OperandCode** as input 0
33	ANEURALNETWORKS_LSH_PROJECTION	Projects an input to a bit vector via locality sensitive hashing.	[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: 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: Tensor [0].Dim [1] ≤ 32 • Input 1: tensor, Dim.size ≥ 1, no restriction on **DataType** • Input 2: (optional) Weight Tensor, Dim.size == 1, DataType == Float. If this parameter is not set, each input element is considered to have the same weight of **1.0**. Tensor[1].Dim[0] == Tensor[2].Dim[0] • Input 3: scalar of type int32 **Type:Sparse Value LSHProjectionType_SPARSE(=3)** (available since API 29). Each output element is made up of multiple bits computed from hash functions. **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**. [Returns] Output 0: tensor • If the projection type is **Sparse**: Output.Dim == { Tensor[0].Dim[0] } • If the projection type is **Dense**: Output.Dim == { Tensor[0].Dim[0] * Tensor[0].Dim[1] }
34	ANEURALNETWORKS_MAX_POOL_2	Performs max pooling	[Inputs] • Supported tensor **OperandCode**:

No.	Operation	Description	Boundary
	D		TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) TENSOR_QUANT8_ASYMM Supported tensor rank: 4 [Inputs (explicit padding)] • 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 ≤ Pad < 256 • Input 2: scalar of type INT32, specifying the padding on the right, in the width dimension, 0 ≤ Pad < 256 • Input 3: scalar of type INT32, specifying the padding on the top, in the height dimension, 0 ≤ Pad < 256 • Input 4: scalar of type INT32, specifying the padding on the bottom, in the height dimension, 0 ≤ Pad < 256 • Input 5: scalar of type INT32, specifying the stride when walking through input in the width dimension, strideW < 64 • Input 6: scalar of type INT32, specifying the stride when walking through input in the height dimension, 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) [Inputs (implicit padding)] • 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, strideW < 64 • Input 3: scalar of type INT32, specifying the stride when walking through input in the height dimension, strideH < 64

No.	Operation	Description	Boundary
			• 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) [Returns] Output 0: 4D Tensor, with shape [batches, out_height, out_width, depth] [Restrictions] • KernelH < 256, kernelW < 256 • 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.	[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: tensor • Input 1: Tensor of the same **OperandCode** and compatible dimensions with input 0. [Restrictions] • 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; For a QUANT8_ASYMM tensor, **scale** and **zeroPoint** can be different from the input.
36	ANEURALNETWORKS_MEAN	Computes the mean of elements across dimensions of a tensor.	[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: tensor

No.	Operation	Description	Boundary
			• 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**. [Returns] Output 0: Tensor of identical **OperandCode** as input 0
37	ANEURALNETWORKS_MINIMUM (available since V310)	Returns the element-wise minimum of two tensors.	[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: 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. [Restrictions] • 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; For a QUANT8_ASYMM tensor, **scale** and **zeroPoint** can be different from the input.
38	ANEURALNETWORKS_MUL	Multiplies two tensors.	[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: 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 [Returns]

No.	Operation	Description	Boundary
			Output 0: Tensor of identical **OperandCode** as input 0. For the TENSOR_QUANT8_ASYMM tensor, output_scale > input1_scale * input2_scale
39	ANEURALNETWORKS_NEG (available since V310)	Computes numerical negative value element-wise.	[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] [Returns] Output 0: Tensor with the identical shape as input 0
40	ANEURALNETWORKS_NOT_EQUAL (available since V310)	For input tensors x and y, computes x != y element-wise.	[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: Tensor of identical **OperandCode** and dimensions compatible with input 0 [Restrictions] • 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
41	ANEURALNETWORKS_PAD	Pads a tensor with zeros.	[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: 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*. **padding[*i*, 1]** specifies the number of elements to be padded after the end of dimension *i*. Specifying the

No.	Operation	Description	Boundary
			number of elements to be padded in each space dimensions of input 0. [Returns] Output 0: Tensor of identical **OperandCode** and dimensions compatible with input 0
42	ANEURALNETWORKS_PAD_V2 (available since V310)	Pads a tensor with the given constant value according to the specified paddings.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: 4 • Input 0: 4D Tensor • 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*. **padding[i, 1]** specifies the number of elements to be padded after the end of dimension *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. [Returns] Output 0: Tensor of identical **OperandCode** and dimensions as input 0
43	ANEURALNETWORKS_PRELU (available since V310)	Activation function Parametric Rectified Linear Unit (PRELU)	[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: tensor • Input 1: tensor, specifying **alpha** [Returns] Output 0: tensor
44	ANEURALNETWORKS_QUANTIZE	Quantizes the input tensor.	[Inputs] • Supported input tensor **OperandCode**:

No.	Operation	Description	Boundary
	(available since V310)		TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 (available since API 29) Supported tensor rank: up to 4 • Input 0: tensor [Returns] Output 0: Tensor of type TENSOR_QUANT8_ASYMM, with the identical shape as input 0
45	ANEURALNETWORKS_RELU	Activation function ReLU	[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: tensor [Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0
46	ANEURALNETWORKS_RELU1	Activation function ReLU1	[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: tensor [Restrictions] None [Returns] Output 0: tensor with the identical shape as input 0
47	ANEURALNETWORKS_RELU6	Activation function ReLU6	[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

No.	Operation	Description	Boundary
			• Input 0: Tensor [Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0
48	ANEURALNETWORKS_RESHAPE	Reshapes the input.	[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: Tensor • Input 1: 1D Tensor of type TENSOR_INT32, specifying the shape of the output Tensor [Restrictions] None [Returns] Output 0: Tensor, with shape specified by the input
49	ANEURALNETWORKS_RESIZE_BILINEAR	Resizes images.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) Supported tensor rank: 4 [Inputs (resizing by shape)] • 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) [Input (resizing by scale, since API level 29)] • 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

No.	Operation	Description	Boundary
			new_width = floor(width * width_scale). • Input 2: scalar of type TENSOR_INT32 or TENSOR_INT16, specifying **height_scale**. The output height is calculated as new_height = floor(height * height_scale). • Input 3: (optional) scale of type BOOL, specifying the data format. Only the default format **NHWC** is supported. [Returns] Output 0: 4D Tensor, with shape [batches, new_height, new_width, depth]
50	ANEURALNETWORKS_RSQRT (available since V310)	Computes reciprocal of square root of x element-wise.	[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] [Returns] Output 0: Tensor with the identical shape as input 0
51	ANEURALNETWORKS_SIN (available since V310)	Computes sin of x element-wise.	[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] [Returns] Output 0: Tensor with the identical shape as input 0
52	ANEURALNETWORKS_SOFTMAX	Normalization logic function	[Inputs] • 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: 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

No.	Operation	Description	Boundary
			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) [Restrictions] **axis** is of the range [–rank, rank). Softmax can be performed on each of the four input dimensions (NCHW). • axis = 0: N ≤ 28544 • axis = 1 (channel dimension): C ≤ 11136 • axis = 2 (height dimension): W = 1, 0 < H < 16384 • axis = 3 (width dimension): 0 < W < 16384 If fewer than four dimensions are input, softmax can be performed only on the last dimension, with the last dimension ≤ 19968. [Returns] Output 0: Tensor with the identical shape as input 0. For a TENSOR_QUANT8_ASYMM Tensor, scale = 1.f/256 and zeroPoint = 0.
53	ANEURALNETWORKS_SPACE_TO_BATCH_ND	SpaceToBatch for n-D tensors.	[Inputs] • 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: 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*. **padding[i, 1]** specifies the number of elements to be padded after the end of dimension *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)

No.	Operation	Description	Boundary
			[Restrictions] When the tensor rank is 4: 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]**. The product of the dimensions of input 0 must be less than **500**. [Returns] Output 0: Tensor of identical **OperandCode** as input 0
54	ANEURALNETWORKS_SPACE_TO_DEPTH	Rearranges blocks of spatial data, into depth.	[Inputs] • 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) [Restrictions] blockSize ≥ 1 and **blockSize** must be a divisor of both the input height and width. [Returns] Output 0: 4D Tensor, with shape [batches, height/block_size, width/block_size, depth_in * block_size * block_size]
55	ANEURALNETWORKS_SQRT (available since V310)	Computes square root of x element-wise.	[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]

No.	Operation	Description	Boundary
			[Returns] Output 0: Tensor with the identical shape as input 0
56	ANEURALNETWORKS_SQUEEZE	Squeeze	[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: (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 not 1 is squeezed. [Returns] Output 0: Tensor of identical **OperandCode** as input 0
57	ANEURALNETWORKS_STRIDED_SLICE	Extracts a strided slice of a tensor.	[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, 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 *i*th bit of **begin_mask** is set, **begin[*i*]** is ignored and the fullest possible range in that dimension is used instead. • Input 5: **end_mask**, scalar of type int32. If the *i*th bit of **end_mask** is set, **end [*i*]** is ignored and the fullest possible range in that dimension is used instead. • Input 6: **shrink_axis_mask**, scalar of type int32. If the *i*th bit of **shrink_axis_mask** is set, the *i*th dimension is shrunk by **1**, taking on the value at index **begin [*i*]**.

No.	Operation	Description	Boundary
			[Restrictions] strides ≠ 0 [Returns] Output 0: Tensor of identical **OperandCode** as input 0
58	ANEURALNETWORKS_SUB	Subtraction of two tensors	[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, specifying the input, 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 [Restrictions] None [Returns] Output 0: Tensor of identical **OperandCode** as input 0
59	ANEURALNETWORKS_TANH	Activation function Tanh	[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: Tensor [Restrictions] None [Returns] Output 0: Tensor with the identical shape as input 0
60	ANEURALNETWORKS_TRANSPOSE	Transposes the input tensor.	[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

No.	Operation	Description	Boundary
			• 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 [Restrictions] None [Returns] Output 0: Tensor of identical **OperandCode** as input 0
61	ANEURALNETWORKS_SPLIT (available since V320)	Splits a tensor along a given axis into **num_splits** subtensors.	[Inputs] • 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 along which to split • Input 2: scalar of type TENSOR_INT32, specifying the **num_split** subtensors split into [Restrictions] None [Returns] Outputs 0 to (**num_split** – 1): resulting subtensors of the identical type as input 0
62	ANEURALNETWORKS_SLICE (available since V320)	Extracts a slice from a tensor.	[Inputs] • 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, 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 [Restrictions] • A sized 0 tensor is not supported. • Input 1 and input 2 accept constants only.

No.	Operation	Description	Boundary
			[Returns] Output 0: n-D tensor of the identical type as input 0
63	ANEURALNETWORKS_RESIZE_NEAREST_NEIGHBOR (available since V320)	Resizes images to given size using the nearest neighbor interpretation.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 • 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: new_width = floor(width * width_scale) • 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: new_height = floor(height * 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. [Restrictions] Zero batches is not supported for the input tensor. [Returns] Output 0: 4D tensor of the identical type as input 0
64	ANEURALNETWORKS_HEATMAP_MAX_KEYPOINT (available since V320)	Localizes the maximum keypoints from heatmaps.	[Inputs] • 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. [Restrictions] The NCHW format is not supported.

No.	Operation	Description	Boundary
			[Returns] • 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	ANEURALNETWORKS_GATHER (available since V320)	Gathers values along an axis.	[Inputs] • 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 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 [Restrictions] • 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. [Returns] Output 0: (n + k – 1)-D tensor of the identical type as input 0
66	ANEURALNETWORKS_POW (available since V320)	Given a tensor base and a tensor exponent, computes base^exponent element-wise.	[Inputs] • 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 [Restrictions] • This operator does not support broadcast. • When the base or power is constant, TENSOR_FLOAT16 is not supported. [Returns]

No.	Operation	Description	Boundary
			Output 0: n-D tensor of the identical type as input 0
67	ANEURALNETWORKS_TILE (available since V320)	Constructs a tensor by tiling a given tensor.	[Inputs] • 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 [Restrictions] • The tile multiples must be constants. [Returns] Output 0: n-D tensor of the identical type and rank as input 0
68	ANEURALNETWORKS_CHANNEL_SHUFFLE (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.	[Inputs] • 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 [Restrictions] The value range of input 2 is [–n, n). [Returns] Output 0: n-D tensor of the identical type and rank as input 0
69	ANEURALNETWORKS_SELECT (available since V320)	Selects elements from input 1 (if input0[i] = true) or input 2 (if input0[i] = false), depending on condition input 0.	[Inputs] • 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

No.	Operation	Description	Boundary
			• Input 2: n-D tensor, with the same type and shape as input 1 [Restrictions] Each dimension size must be within **256**. [Returns] Output 0: n-D tensor, with the same type and shape as input 1 and input 2.
70	ANEURALNETWORKS_TOPK_V2 (available since V320)	Finds values and indices of the k largest entries for the last dimension. Resulting values in each dimension are sorted in descending order.	[Inputs] • 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,specifying the number of top elements to look for along the last dimension [Restrictions] Input 1 must be within the dimension size of the last dimension. [Returns] • 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.	[Inputs] • 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 [Restrictions] Given an input of n dimensions, **axis** must be in range [–(n + 1), (n + 1)) [Returns] Output 0: (n + 1)-D tensor of the identical type as input 0
72	ANEURALNETWORKS_REDUCE_ALL	Reduces a tensor by computing	[Inputs] • Supported tensor **OperandCode**:

No.	Operation	Description	Boundary
	(available since V320)	the "logical and" of elements along given dimensions.	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. [Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
73	ANEURALNETWORKS_REDUCE_ANY (available since V320)	Reduces a tensor by computing the "logical or" of elements along given dimensions.	[Inputs] • 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. [Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
74	ANEURALNETWORKS_REDUCE_PROD (available since V320)	Reduces a tensor by multiplying elements along given dimensions.	[Inputs] • 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.

No.	Operation	Description	Boundary
			[Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
75	ANEURALNETWORKS_REDUCE_MAX (available since V320)	Reduces a tensor by computing the maximum of elements along given dimensions.	[Inputs] • 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. [Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
76	ANEURALNETWORKS_REDUCE_MIN (available since V320)	Reduces a tensor by computing the minimum of elements along given dimensions.	[Inputs] • 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. [Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
77	ANEURALNETWORKS_REDUCE_SUM	Reduces a tensor by summing	[Inputs] • Supported tensor **OperandCode**:

No.	Operation	Description	Boundary
	(available since V320)	elements along given dimensions.	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. [Restrictions] Broadcast is not supported. [Returns] Output 0: Tensor of the identical type as input 0
78	ANEURALNETWORKS_RANDOM_MULTINOMIAL (available since V320)	Draws samples from a multinomial distribution.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 • 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 [Restrictions] • Input 1 must be positive. • Input 2 accepts constants only. [Returns] Output 0: 2D tensor, drawn samples
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	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 • 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].

No.	Operation	Description	Boundary
		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.	• 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 single class • 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. [Restrictions] 0 < Inputs 8 and 9 ≤ 1024 Input 10 = 1 2 ≤ num_classes < 1024 0 < num_anchors < 65536 Inputs 11 and 12 ≥ 0 **scaleX**, **scaleY**, **scaleH**, and **scaleW** > 1e – 5 [Returns] • Output 0: 2D tensor with shape [batches, max_num_detections], specifying the score of each output detections • Output 1: 3D tensor with shape [batches, max_num_detections, 4], specifying the coordinates of each output ROI

No.	Operation	Description	Boundary
			• 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	ANEURALNETWORKS_GROUPED_CONV_2D (available since V320)	Performs a grouped 2D convolution operation.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 [Inputs (explicit padding)] • 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 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. [Inputs (implicit padding)] • 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

No.	Operation	Description	Boundary
			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. [Restrictions] inputC = group * filterC; filterN%group = 0 [Returns] Output 0: 4D tensor with shape [batches, out_height, out_width, depth_out]
81	ANEURALNETWORKS_TRANSPOSE_CONV_2D (available since V320)	Performs the transpose of 2D convolution operation.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 [Inputs (explicit padding)] • 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

No.	Operation	Description	Boundary
			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. [Inputs (implicit padding)] • 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 output tensor • 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. [Restrictions] • Input 1 and input 2 accept constants only. • (inputH - 1) * strideH + 1 + aH ≤ 4000; (inputW - 1) * strideW + 1 + aW ≤ 4000; group ==1; dilationH == dilationW == 1; filterH ≤ 15 && filterW ≤ 15; filterH – padHHead – 1 ≥ 0 filterW – padWHead – 1 ≥ 0 where, aH = (inputH + padHHead + padHTail – filterH) % strideH aW = (inputW + padHHead + padHTail – filterW) %

No.	Operation	Description	Boundary
			strideW • (inputH – 1) * strideH – padHHead – padHTail ≤ outputH ≤ inputH * strideH – padHHead – padHTail; (inputW – 1) * strideW – padWHead – padWTail ≤ outputW ≤ inputW * strideW – padWHead – padWTail [Returns] Output 0: 4D tensor
82	ANEURALNETWORKS_LSTM (available since V320)	LSTM cell	[Inputs] • 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, 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 • 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

No.	Operation	Description	Boundary
			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 th 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. • 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 [Restrictions] • Inputs 23–26 for layer normalization are not used in computation. • Inputs 1–17 accept constants only. • Outputting the scratch buffer is not supported. [Returns] • 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 [batche_size, output_size], specifying the output state (out) • Output 2: 2D tensor with shape [batche_size, num_units], specifying the cell state (out) • Output 3: 2D tensor with shape [batch_size, output_size]

No.	Operation	Description	Boundary
83	ANEURALNETWORKS_UNIDIRECTIONAL_SEQUENCES_LSTM (available since V320)	Unidirectional LSTM cell	[Inputs] • Supported tensor **OperandCode**: 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 • 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 th project weights • Input 17: 1D tensor with shape [output_size], specifying

No.	Operation	Description	Boundary
			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. • 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 [Restrictions] • Inputs 24-27 for layer normalization are not used in computation. • Inputs 1–17 accept constants only. • Outputting the scratch buffer is not supported. [Returns] 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.
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.	[Inputs] • 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] • Input 3: scalar, specifying the output height of the output

No.	Operation	Description	Boundary
			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. [Restrictions] • Input 1 accepts constants only and does not support time-major computation. • Input 2 accepts constants only [Returns] Output 0: 4D tensor.
85	ANEURALNETWORKS_SVDF (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.	[Inputs] • Supported tensor **OperandCode**: 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**. [Restrictions] Inputs 1–4 accept constants only [Returns] • Output 0: 2D tensor with shape [batch_size, memory_size * num_units * rank] • Output 1: 2D tensor with shape [batch_size, num_units]
86	ANEURALNETWORKS_INSTANCE_NORMALIZATIO	Applies instance normalization to the input	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario)

No.	Operation	Description	Boundary
	N (available since V320)	tensor.	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 [Restrictions] None [Returns] Output 0: n-D tensor
87	ANEURALNETWORKS_BIDIRECTIONAL_SEQUENCE_LSTM (available since V320)	Performs a forward LSTM on the input followed by a backward LSTM.	[Inputs] • 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-

No.	Operation	Description	Boundary
			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 [bw_output_size, bw_num_units], specifying the forward projection weight • 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

No.	Operation	Description	Boundary
			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 [Restrictions] • Only constant inputs are accepted except input 0 and inputs 35–38. • Auxiliary inputs 39–47 and normalized weights 53–60 are in valid. • **merge_output** is not supported. That is, input 51 must be **false**. [Returns] • Output 0: 3D tensor, forward LSTM result • Output 1: 3D tensor, backward LSTM result
88	ANEURALNETWORKS_UNIDIRECTIONAL_SEQUENCE_RNN (available since V320)	Applies a basic RNN cell to a sequence of inputs.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 The input tensors must all be the same type. • 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

No.	Operation	Description	Boundary
			is [batchSize, maxTime, numUnits]. If set to **1**, the input and output shape is [maxTime, batchSize, numUnits]. [Returns] 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].
89	ANEURALNETWORKS_BIDIRECTIONAL_SEQUENCE_RNN (available since V320)	Applies a basic RNN cell to a sequence of inputs in forward and backward directions.	[Inputs] • Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 The input tensors must all be the same type. • 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

No.	Operation	Description	Boundary
			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**) [Restrictions] **mergeOutputs** must be **false**. [Returns] • Output 0: **fwOutput**, 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]. • 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	ANEURALNETWORKS_RNN (available since V320)	A basic recurrent neural network layer.	[Inputs] • 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**. [Returns]

No.	Operation	Description	Boundary
			Output 0: 2D tensor with shape [batch_size, num_units], specifying the hidden state (out) Output 1: 2D tensor with shape [batch_size, num_units], specifying the output
91	ANEURALNETWORKS_ROI_ALIGN (available since V320)	Selects and scales the feature map of each region of interest to a unified output size by average pooling sampling points from bilinear interpolation.	[Inputs] • 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 • 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. [Restrictions] • Input 1 accepts constants only and does not support time-major computation. • Input 2 accepts constants only [Returns] Output 0: 4D tensor.
92	ANEURALNETWORKS_GENERATE_PROPOSALS (available since V320)	Generates axis-aligned bounding box proposals.	[Inputs] Supported tensor **OperandCode**: TENSOR_FLOAT32 (only in the relaxed scenario) TENSOR_FLOAT16 • Input 0: 4D tensor with shape [batches, num_anchors, height, width]

No.	Operation	Description	Boundary
			• 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 • 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. [Restrictions] • Only NHWC is supported. [Returns] • 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	ANEURALNETWORKS_AXIS_ALIGNED_BBOX_TRANSFORM (available since V320)	Transform axis-aligned bounding box proposals using bounding box deltas.	[Inputs] • Supported tensor **OperandCode**: 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] [Restrictions]

No.	Operation	Description	Boundary
			None [Returns] Output 0: 2D tensor with shape [num_rois, num_classes * 4]
94	ANEURALNETWORKS_BOX_WITH_NMS_LIMIT (available since V320)	Greedily selects a subset of bounding boxes in descending order of score.	[Inputs] • Supported tensor **OperandCode**: 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** • 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** [Restrictions] Only the hard NMS algorithm is supported. [Returns] • 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]
95	ANEURALNETWORKS_QUANTIZED_16BIT_LSTM (available since V320)	A version of quantized LSTM, using 16 bit quantization for internal state.	[Inputs] • 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. **nums_units** corresponds to the number of cell units. • 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,

No.	Operation	Description	Boundary
			output_size], specifying the recurrent-to-input weights. **output_size** corresponds to either the number of cell units (num_units), or the second dimension of projection_weights, if defined. • 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 [Restrictions] • 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. [Returns] • Output 0: 2D tensor with shape [numBatches, outputSize], specifying the cell state • Output 1: 2D tensor with shape [numBatches, outputSize], specifying the output value

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 = (scale * x + shift)^power
14	Argmax	Computes the index of the maximum values.
15	Interp	Interpolation layer
16	LeakyRelu	Activation function
17	ConvolutionDepthwise	Depthwise convolution

