
RKNN Compiler Support Operator List

发布 v2.2.0

NPU 团队

2024 年 09 月 13 日

| | | |
|----------|---------------------------------------|------------|
| 1 | 关于本文档 | 1 |
| 2 | 更新历史 | 3 |
| 2.1 | V2.2.0 | 3 |
| 2.2 | V2.1.0 | 3 |
| 2.3 | V2.0.0-beta | 4 |
| 3 | NPU OPs 规格介绍 | 5 |
| 3.1 | RK3566/RK3568 | 6 |
| 3.2 | RK3588 | 50 |
| 3.3 | RK3562 | 102 |
| 3.4 | RV1103/RV1106 | 148 |
| 3.5 | RK3576 | 191 |
| 3.6 | RK2118 | 241 |
| 3.7 | RV1103B | 277 |
| 4 | CPU OPs 规格介绍 | 309 |
| 4.1 | 注: | 309 |
| 4.2 | Add | 309 |
| 4.3 | AveragePool | 309 |
| 4.4 | ArgMin | 310 |
| 4.5 | ArgMax | 310 |
| 4.6 | BatchNormalization | 310 |
| 4.7 | Cast | 310 |
| 4.8 | Clip | 310 |
| 4.9 | Concat | 310 |
| 4.10 | Convolution | 310 |
| 4.11 | ConvTranspose/Deconvolution | 311 |
| 4.12 | ConvTransposePad | 311 |
| 4.13 | Cos | 311 |
| 4.14 | DataConvert | 311 |
| 4.15 | DepthToSpace | 311 |
| 4.16 | Div | 311 |
| 4.17 | Equal | 311 |
| 4.18 | Exp | 312 |
| 4.19 | Flatten | 312 |
| 4.20 | Gather | 312 |
| 4.21 | Greater | 312 |
| 4.22 | GreaterOrEqual | 312 |
| 4.23 | GRU | 312 |
| 4.24 | exGRU | 312 |

| | | |
|------|-----------------------------|-----|
| 4.25 | exHardSwish | 313 |
| 4.26 | InstanceNormalization | 313 |
| 4.27 | exLayerNorm | 313 |
| 4.28 | Less | 313 |
| 4.29 | LessOrEqual | 313 |
| 4.30 | LogSoftmax | 313 |
| 4.31 | LpNormalization | 313 |
| 4.32 | exLRN | 314 |
| 4.33 | MatMul | 314 |
| 4.34 | Max | 314 |
| 4.35 | MaxPool | 314 |
| 4.36 | MaxRoiPool | 314 |
| 4.37 | MaxUnpool | 314 |
| 4.38 | exMish | 314 |
| 4.39 | Min | 315 |
| 4.40 | Mul | 315 |
| 4.41 | Pad | 315 |
| 4.42 | Pow | 315 |
| 4.43 | exProposal | 315 |
| 4.44 | ReduceMax | 315 |
| 4.45 | ReduceMean | 315 |
| 4.46 | ReduceSum | 316 |
| 4.47 | ReduceMin | 316 |
| 4.48 | Reorg | 316 |
| 4.49 | Reshape | 316 |
| 4.50 | Resize | 316 |
| 4.51 | ReverseSequence | 316 |
| 4.52 | exRMSNorm | 316 |
| 4.53 | RoiAlign | 317 |
| 4.54 | ScatterND | 317 |
| 4.55 | Sin | 317 |
| 4.56 | Slice | 317 |
| 4.57 | Softmax | 317 |
| 4.58 | exSoftmax13 | 317 |
| 4.59 | SpaceToDepth | 317 |
| 4.60 | Split | 318 |
| 4.61 | Sqrt | 318 |
| 4.62 | Squeeze | 318 |
| 4.63 | Sub | 318 |
| 4.64 | Tanh | 318 |
| 4.65 | Tile | 318 |
| 4.66 | Transpose | 318 |
| 4.67 | Upsample | 319 |
| 4.68 | Not | 319 |
| 4.69 | where | 319 |
| 4.70 | Erf | 319 |
| 4.71 | Floor | 319 |
| 4.72 | Mod | 319 |
| 4.73 | exMeanVarianceNormalization | 319 |
| 4.74 | And | 320 |
| 4.75 | GatherElements | 320 |
| 4.76 | Log | 320 |
| 4.77 | exDataConvert | 320 |
| 4.78 | SpaceToDepth | 320 |
| 4.79 | SoftmaxMask | 320 |
| 4.80 | TransposeReshape | 320 |
| 4.81 | ReshapeTranspose | 321 |
| 4.82 | ReduceL2 | 321 |

| | | |
|----------|--------------------|------------|
| 4.83 | LayerNormalization | 321 |
| 4.84 | PassThrough | 321 |
| 4.85 | Expand | 321 |
| 4.86 | Sigmoid | 321 |
| 4.87 | Tanh | 321 |
| 4.88 | Cos | 322 |
| 4.89 | Sin | 322 |
| 4.90 | exSwish | 322 |
| 4.91 | exGlu | 322 |
| 4.92 | exGelu | 322 |
| 4.93 | exWindow | 322 |
| 4.94 | exNorm | 322 |
| 4.95 | Round | 323 |
| 4.96 | TopK | 323 |
| 4.97 | exSwooshR | 323 |
| 4.98 | exSwooshL | 323 |
| 4.99 | OneHot | 323 |
| 5 | 模型输入输出限制说明 | 325 |
| 5.1 | 模型输入限制说明 | 325 |
| 5.2 | 模型输出限制说明 | 329 |
| 6 | 特殊说明 | 333 |
| 6.1 | dilated_kernel_h | 333 |
| 6.2 | dilated_kernel_w | 333 |

CHAPTER 1

关于本文档

文件标识：RK-YH-YF-418

发布版本：V2.1.0

日期：2024-08-07

文件密级：☐ 绝密 ☐ 秘密 ☐ 内部资料 ☒ 公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2024 瑞芯微电子股份有限公司

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

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园 A 区 18 号

网址：www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

概述

RKNN Compiler Support Operator List

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

| 版本号 | 作者 | 修改日期 | 修改说明 |
|-------------|--------|------------|---------------------|
| 早期版本 | NPU 团队 | - | 详情页 |
| V2.0.0-beta | NPU 团队 | 2024-03-22 | 详情页 |
| V2.1.0 | NPU 团队 | 2024-08-07 | 详情页 |
| V2.2.0 | NPU 团队 | 2024-09-04 | 详情页 |

CHAPTER 2

更新历史

2.1 V2.2.0

修改人

NPU 团队

修改日期

2024-09-03

修改内容

1. 全平台 (除 rk3568) 修改 softmax 规格限制, 新增 softmax 切分
2. RK3576、RK3588、RK3562、RK3568 新增 RMSNorm 支持

2.2 V2.1.0

修改人

NPU 团队

修改日期

2024-08-07

修改内容

1. 新增 RK2118 文档说明
2. 新增 RV1103B 文档说明
3. 全平台算子规格更新
4. RK3576 算子规格新增 (包含 Hardmax、ConvLut 等)
5. RK2118 exConvStreaming 支持
6. RK3562/RK3576 exSDPA 算子添加 FlashAttentionV2 优化支持

2.3 V2.0.0-beta

修改人

NPU 团队

修改日期

2024-03-22

修改内容

1. 新增 RK3576 平台算子支持规格
2. 新增 RK3588/RK3562/RK3576 平台的 exSDPAttention、exMatMul 的硬件支持规格
3. 新增 Floor、Mod、And、GatherElements 以及 exMeanVarianceNormalization CPU 算子支持
4. 新增 RK2118 平台算子支持规格

CHAPTER 3

NPU OPs 规格介绍

3.1 RK3566/RK3568

3.1.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.1.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add
-

其他支持

- 多核联合: 支持
-

3.1.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- epsilon : float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum : float
 - 训练时的滑动平均参数

3.1.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.1.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - $stride_h \in [1, 7]$
 - $stride_w \in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - $pads_top \in [0, 15]$
 - $pads_left \in [0, 15]$
 - $pads_bottom \in [0, 15]$
 - $pads_right \in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - $dilations_h \in [1, 32]$

- `dilations_w` $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.6 Depthwise Convolution

深度可分离卷积

输入列表

- `input_tensor`: **T1**
 - **shape**: [batch, channel, height, width]
 - * `width`: 当 `dilated_kernel_h` > 1 时, `width` < 16383。此外, 对首层输入 `width` 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- `weight`: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * `kernel_h` $\in [1, 8]$
 - * `kernel_w` $\in [1, 8]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- `output`: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- `strides`: int(strides_h, strides_w)
 - `stride_h` $\in [1, 7]$
 - `stride_w` $\in [1, 7]$
- `pads`: int(pads_top, pads_left, pads_bottom, pads_right)
 - `pads_top` $\in [0, 15]$
 - `pads_left` $\in [0, 15]$
 - `pads_bottom` $\in [0, 15]$
 - `pads_right` $\in [0, 15]$
- `group`: int
- `dilations`: int(dilations_h, dilations_w)

- dilations_h $\in [1, 32]$
- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.1.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.1.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.1.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

3.1.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.1.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.1.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.1.19 ConvolutionAddRelu

Convolution 与 Add 及 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Add}(\text{Conv}(\text{Input0}), \text{Input1}))$

说明及规格限制同Convolution

3.1.20 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h 仅支持 1,2,4,8
 - stride_w 仅支持 1,2,4,8
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.21 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.22 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.23 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{ConvTranpose}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.24 ConvTranposeAdd

ConvTranpose 与 Add 融合计算, $\text{Output} = \text{Add}(\text{ConvTranpose}(\text{Input0}), \text{Input1})$

说明及规格限制同[ConvTranspose](#)

3.1.25 ConvTranposeSigmoid

ConvTranpose 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.26 ConvTranposeTanh

ConvTranpose 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.27 ConvTranposeSoftplus

ConvTranpose 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.28 ConvTranposeHardSigmoid

ConvTranpose 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{ConvTranpose}(\text{Input}))$

3.1.29 ConvTranposeHardSwish

ConvTranpose 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.30 ConvTranposeElu

ConvTranpose 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.31 ConvTranposeSwish

ConvTranpose 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.32 ConvTranposeMish

ConvTranpose 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.1.33 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.1.34 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`: 维度右对齐; 两个对应维度必须具有相同的值, 或者其中一个等于 1

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - shape: **T2**
 - **shape** : [batch, channel, height, width]
-

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]->[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

3.1.35 GRU

门控循环单元（GRU，Gated Recurrent Unit）

GRU 扩展以及变体命名为 exGRU 算子，参数项中指明

（extern）的项为 exGRU 独有的参数项。

计算公式如下：

Equations (Default: f=Sigmoid, g=Tanh):

$$r_t = f(x_t \cdot W_r + h_{t-1} \cdot R_r + Wb_r + Rb_r)$$

$$z_t = f(x_t \cdot W_z + h_{t-1} \cdot R_z + Wb_z + Rb_z)$$

$$h_t = g(x_t \cdot W_h + (r_t \odot h_{t-1}) \cdot R_h + Rb_h + Wb_h), (WRB = 0)$$

$$h_t = g(x_t \cdot W_h + r_t \odot (h_{t-1} \cdot R_h + Rb_h)) + Wb_h, (WRB = 1)$$

$$H_t = (1 - z_t) \odot h_t + z_t \odot h_{t-1}$$

输入列表

- X : **T1**
 - **shape** : [seq_length, batch_size, input_size]
 - * $seq_length \in (0, 8192]$
 - * $batch_size \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $input_size \in (0, 8192]$
 - **量化支持**
 - * per-layer
- W : **T1**
 - **shape** : [num_directions, 4*hidden_size, input_size]

- * $inputs_size \in (0, 8192]$
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
- 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2**
 - **shape**: [num_directions, 8*hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T3** (optional)
 - **shape**: [batch_size]
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]
 - * $seqlength \in (0, 8192]$
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

- **Y_h : T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - **量化支持**
 - * per-layer

属性列表

- linear_before_reset: int
 - LBR 变种的选择: 1(T) or 0(F)
- direction (extern): string
 - 指定 GRU 的运算方向
 - forward: 指定 GRU 的运算方向为前向
 - reverse: 指定 GRU 的运算方向为反向
 - bidirectional: 指定 GRU 的运算方向为双向
- sequence_size (extern): int
 - 指定 GRU 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - GRU 单元中的 hiddensize, 无限制, 建议 8 对齐
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size, 1, 1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size, 1, 1]

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

其他支持

- **多核联合**: 暂不支持

3.1.36 Gather

根据索引 Indices 获取输入 X 的指定 axis 维度的条目，并将它们拼接在一起。

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- Indices: **T2**
 - 要收集的元素的索引，秩 $rank = 1$

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int(Tensor)

属性列表

- axis: int(Tensor)
 - $axis \in \{0, 1, 2, 3\}$
 - 指定 Indices 获取输入的维度

其他支持

- **多核联合**: 尚不支持

3.1.37 LSTM

长短期记忆网络 (LSTM, Long Short-Term Memory)

LSTM 扩展以及变体命名为 exLSTM 算子，参数项中指明

(extern) 的项为 exLSTM 独有的参数项。

计算公式如下：

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C_{(t-1)} + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- X: **T1**
 - **shape**: [seq_length, batch_size, input_size]
 - * *seqlength* $\in (0, 8192]$
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *inputsize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- W: **T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * *inputsize* $\in (0, 8192]$
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2, T3**
 - **shape**: [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T4, T5** (optional)
 - **shape**: [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数

- * $hiddensize \in (0, 8192]$
- 量化支持
 - * per-layer
- initial_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y : **T1**
 - **shape** : [seq_length, num_directions, batch_size, hidden_size]
 - * $seqlength \in (0, 8192]$
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向
 - bidirectional: 指定 LSTM 的运算方向为双向

- `sequence_size` (extern): int
 - 指定 LSTM 输入的 `seqsize`，无限制，建议 4 对齐
- `hidden_size` (extern): int
 - LSTM 单元中的 `hiddensize`，无限制，建议 8 对齐
- `proj_size` (extern): int
 - projection 时的 `proj_size`， $projsize \in [0, hiddensize]$
 - 目前限定 0，即尚不支持 projection 功能
- `input_forget` (extern): int
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0，即尚不支持
- `has_projection` (extern): int
 - projection 变种: 1(T) or 0(F) 目前限定 0，即尚不支持
- `input_layout` (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout，同时要求填写该 op 实际对应的 `batch_size`、`sequence_size`、`hidden_size`。
 - `snc`: 指定 layout 对应的输入 shape 为 `[seqs, batches, input_size]`
 - `(sn)c`: 指定 layout 对应的输入 shape 为 `[seqs*batches, input_size, 1, 1]`
- `output_layout` (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout，同时要求填写该 op 实际对应的 `batch_size`、`sequence_size`、`hidden_size`、`directions`。
 - `sbnc`: 指定 layout 对应的输出 shape 为 `[seqs, directions, batches, hidden_size]`
 - `(sn)c`: 指定 layout 对应的输出 shape 为 `[seqsbatches, directionsinput_size, 1, 1]`

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int32(Tensor)
- **T4**: float(Scalar)
- **T5**: int(Scalar)

其他支持

- **多核联合**: 暂不支持

3.1.38 Max

计算输入张量的元素级最大值

输入列表

- **A: T1**
 - **shape**: `[batch, channel, height, width]`
 - **量化支持**

- * per-layer/per-channel
- 广播支持
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W),B(N,C,H,W))$ ，即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting
- B: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel
 - 广播支持
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W),B(N,C,H,W))$ ，即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

3.1.39 Min

计算输入张量的元素级最小值

输入列表

- A: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel
 - 广播支持
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W),B(N,C,H,W))$ ，即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting

- B: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W)) , 即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1)), 即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar)), 即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

3.1.40 Mul

执行元素级二进制乘法 ($C = A * B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.1.41 MulRelu

其余说明同 Mul

- 支持的数据类型:
 - 同 Mul

3.1.42 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- pads: **T2**
 - **shape**: [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape**: [1]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)
- **T3**: float(Scalar), int8(Scalar), float16(Scalar)

属性列表

- mode : **string**(默认值 constant)
 - 支持模式: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$

其他支持

- 多核联合: 尚不支持

3.1.43 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * channel $\in (0, 8192]$
 - * height $\in (0, 8192]$
 - * width $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)

- $\text{pad_top} \in [0, 7]$
- $\text{pad_left} \in [0, 7]$
- $\text{pad_bottom} \in [0, 7]$
- $\text{pad_right} \in [0, 7]$
- $\text{storage_order} : \text{int64}$
 - 优先储存方式: 0

数据类型约束

- **T1**: $\text{int8}(\text{Tensor})$, $\text{float16}(\text{Tensor})$
- **T2**: $\text{float}(\text{Tensor})$
- **T3**: $\text{float}(\text{Scalar})$

3.1.44 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $\text{channel} \in (0, 8192]$
 - * $\text{height} \in (1, 7]$
 - * $\text{width} \in (1, 7]$

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

属性列表

- $\text{kernel_shape} : \text{int64}(\text{kernel_h}, \text{kernel_w})$
 - $\text{kernel_h} \in (0, 7]$
 - $\text{kernel_w} \in (0, 7]$
- $\text{auto_pad} : \text{string}$
 - pad 的方式: 仅支持 NOTSET
- $\text{ceil_mode} : \text{int64}$
 - 使用 ceil 或 floor 的方式计算输出的 shape: 不支持
- $\text{strides} : \text{int64}(\text{strides_h}, \text{strides_w})$
 - $\text{strides_h} \in (0, 8]$
 - $\text{strides_w} \in (0, 8]$
- $\text{count_include_pad} : \text{int64}[]$
 - count_include_pad 是否包含 pad 数值进行计算: 1

- **pad** : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.1.45 AveragePool

AveragePool 使用输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用平均池化。平均池化包括根据内核大小计算输入张量子集的所有值的平均值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- **input** : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (0, 8192]$
 - * *width* $\in (0, 8192]$

输出列表

- **output** : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- **kernel_shape**: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- **auto_pad**: string
 - pad 的方式: 仅支持 NOTSET
- **ceil_mode**: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- **strides** : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- **count_include_pad** : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1

- **pad** : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.1.46 GlobalAveragePool

GlobalAveragePool 使用输入张量 X 并对同一通道中的值应用平均池化。这相当于 AveragePool，其 kernel_shape 大小等于输入张量的空间维度。

输入列表 *

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (0, 7]$
 - * *width* $\in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- **pad** : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$

- `pad_bottom` $\in [0, 7]$
- `pad_right` $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.1.47 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- **Y: T2**
 - **shape**: [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- **Z: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)

其他支持

- **多核联合**: 尚不支持

3.1.48 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**

- * per-layer
- 广播支持
- * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.1.49 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.1.50 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.1.51 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
-

- * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.1.52 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- shape: int
 - 指定输出张量的形状

其他支持

- 多核联合: 尚不支持

3.1.53 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - roi (optional): **T2**
-

- 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

3.1.54 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴，支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 height * width == 1 && channel % subc == 0 && starts[1] % subc == 0 && ends[1] % subc == 0 && steps[1] <= subc 时, steps 可以不为 1，否则只有 steps == [1,1,1,1] 时，才可通过 NPU 运行 Slice OP

3.1.55 Split

将向量沿着 axis 方向分成 num_outputs 个向量，split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**

- 一或多个输出向量，由 num_outputs 属性决定
- **shape** : 根据切分情况决定
- **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis :int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs :int
- split : int(Tensor)

3.1.56 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播

输入列表

- A : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * [b, c, 1, 1]
 - * [scalar]
- B : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.1.57 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats : **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

3.1.58 Transpose

对输入张量进行转置

输入列表

- X : **T1**
 - **shape** : [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

输出列表

- Y : **T1, T2**
 - **shape** : [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

属性列表

- perm 转置的轴顺序:
 - 不同 perm 参数限制如下: (其中 p=8192, q=2048)
 - perm=[0,1,2,3]

- * 无限制
- perm=[0,1,3,2]
 - * 不支持
- perm=[0,2,3,1]
 - * 不支持
- perm=[0,2,1,3]
 - * 不支持
- perm=[0,3,1,2]
 - * 不支持
- perm=[0,3,2,1]
 - * 不支持
- perm=[1,0,2,3]
 - * 不支持
- perm=[1,0,3,2]
 - * 不支持
- perm=[1,2,0,3]
 - * 不支持
- perm=[1,2,3,0]
 - * 不支持
- perm=[1,3,0,2]
 - * 不支持
- perm=[1,3,2,0]
 - * 不支持
- perm=[2,1,0,3]
 - * **T1, T2:** $c1 < 8192, n < 8192, h1 * w1 < 8192, (h1=w1=1, n1 \% 4=0)$ or $(n1=h1=1, h1 \% 4=0)$
- perm=[2,1,3,0]
 - * **T1, T2:** $c1 < 8192, n < 8192, h1 * w1 < 8192, (h1=w1=1, n1 \% 4=0)$ or $(n1=h1=1, h1 \% 4=0)$
- perm=[2,0,3,1]
 - * 不支持
- perm=[2,0,1,3]
 - * 不支持
- perm=[2,3,1,0]
 - * 不支持
- perm=[2,3,0,1]
 - * 不支持
- perm=[3,0,2,1]
 - * 不支持
- perm=[3,0,1,2]
 - * 不支持

- perm=[3,2,0,1]
 - * 不支持
- perm=[3,2,1,0]
 - * 不支持
- perm=[3,1,0,2]
 - * **T1, T2**: $c1 < 8192, n < 8192, h1 * w1 < 8192, (h1 = w1 = 1, n1 \% 4 = 0)$ or $(n1 = h1 = 1, w1 \% 4 = 0)$
- perm=[3,1,2,0]
 - * **T1, T2**: $c1 < 8192, n < 8192, h1 * w1 < 8192, (h1 = w1 = 1, n1 \% 4 = 0)$ or $(n1 = h1 = 1, w1 \% 4 = 0)$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.1.59 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.60 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.61 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- **X: T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.62 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **input: T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.63 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.64 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.65 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.66 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.67 exGelu

给定输入张量, 函数 $y = x * \Phi(x)$ 按元素应用于输入张量, 得到输出张量, 当 $\Phi(x)$ 函数设置成 \tanh 时, $y = 0.5 * x * (1 + \text{Tanh}(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- **input: T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.68 exSwooshR

给定输入张量，按论文 <https://arxiv.org/abs/2310.11230> 给定的下列函数，按元素应用于输入张量，得到输出张量

$$SwooshR(x) = \log(1 + \exp(x-1)) - 0.08x - 0.313261687$$

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.69 exSwooshL

给定输入张量，按论文 <https://arxiv.org/abs/2310.11230> 给定的下列函数，按元素应用于输入张量，得到输出张量

$$SwooshL(x) = \log(1 + \exp(x-4)) - 0.08x - 0.035$$

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持

- * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.1.70 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor: *T2
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

输出列表

- output: **T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持

★ per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor), float16(Tensor)

3.1.71 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

3.1.72 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- A: **T1**
 - **shape**: [b, k, 1, n]
 - ★ $k \in (0, 8192]$
 - 量化支持
 - ★ per-layer
- B: **T1**

- **shape** : [b, k, 1, m]
- 量化支持
 - * per-layer
- C (optional) : **T1, T2**
 - **shape** : [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- Y: **T1**
 - **shape** : [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)

属性列表

- c_type : strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

3.1.73 exNorm

对输入数据进行各种归一化操作，目前支持 LayerNormlization、RMSNormlization 和 MeanVarianceNormalization 三种：

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot W + B$$

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\text{RMS}^2 + \epsilon}} \cdot W + B$$

$$\text{MeanVarianceNorm}(x) = \frac{x - \mu}{\sqrt{\text{Variance} + \epsilon}} \cdot W + B$$

其中：

x 是输入向量。

μ 和 σ^2 分别是输入向量的均值和方差：

$$\mu = \frac{1}{H} \sum_{i=1}^H x_i$$

$$\sigma^2 = \frac{1}{H} \sum_{i=1}^H (x_i - \mu)^2$$

RMS 是输入向量均方根：

$$RMS = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i)^2}$$

$Variance$ 是输入向量的等效方差：

$$Variance = \frac{1}{H} \sum_{i=1}^H (x_i)^2 - \mu^2$$

ϵ 是一个很小的数，用于防止除零操作。

W 和 B 是可选的参数。

输入列表

- X : **T1**
 - **shape**: [batch, channel, height, width]
- W (optional): **T1**
 - **shape**: [batch, channel, height, width]
- B (optional): **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y : **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- epsilon: **float**(默认值 1e-05)
 - 用来防止除 0 的 epsilon 数值
- mode: **string**
 - 支持 layer、rms 和 mean_var 三种 Normalization 模式可选
- axes: **int64[]**(默认值 [1])
 - 归一化轴，目前只支持 [1] 和 [1,2,3]

其他支持

- 多核联合: 尚不支持

3.1.74 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。

原论文中输入特征被划分为多个不重叠的窗口，每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape，在高和宽方向上都进行划分。

pytorch 实现的计算方法如下：

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
    windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
    return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
        window_size (int): Window size
        H (int): Height of image
        W (int): Width of image

    Returns:
        x: (B, C, H, W)
    """
    B, C, _, _ = windows.shape
    x = windows.view(B, C, H // window_size, window_size, W // window_size, window_size)
    x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)
```

(下页继续)

(续上页)

```
return x
```

输入列表

- X : **T1**
 - **shape** : [batch, channel, height, width]

输出列表

- Y : **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- mode : **string**
 - **支持模式**: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;
 - * partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
- window_sizes : **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
- num_windows : **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。

其他支持

- **多核联合**: 尚不支持

3.2 RK3588

3.2.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(const Tensor)
-

其他支持

- **多核联合**: 支持

3.2.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add
-

其他支持

- 多核联合: 支持

3.2.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- epsilon: float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum: float
 - 训练时的滑动平均参数

其他支持

- 多核联合: 尚不支持

3.2.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.2.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**

- **shape** : [batch, channel, height, width]
- **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 支持

3.2.6 Depthwise Convolution

深度可分离卷积

输入列表

- input_tensor: **T1**
 - **shape** : [batch, channel, height, width]
 - * width : 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape** : [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 8]$
 - * kernel_w $\in [1, 8]$
 - **量化支持**

- * per-layer
- * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 支持

3.2.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.2.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.2.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.2.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.2.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 暂不支持

3.2.19 ConvolutionAddRelu

Convolution 与 Add 及 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Add}(\text{Conv}(\text{Input0}), \text{Input1}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.2.20 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h 仅支持 1,2,4,8
 - stride_w 仅支持 1,2,4,8
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.21 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, Output = Relu(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.2.22 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, Output = Clip(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.2.23 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, Output = PRelu(ConvTranpose(Input)) 或者 Output = LeakyRelu(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.2.24 ConvTranposeAdd

ConvTranpose 与 Add 融合计算, Output = Add(ConvTranpose(Input0), Input1)

说明及规格限制同[ConvTranspose](#)

3.2.25 ConvTranposeSigmoid

ConvTranpose 与 Sigmoid 融合计算, Output = Sigmoid(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.2.26 ConvTranposeTanh

ConvTranpose 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.27 ConvTranposeSoftplus

ConvTranpose 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.28 ConvTranposeHardSigmoid

ConvTranpose 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{ConvTranpose}(\text{Input}))$

3.2.29 ConvTranposeHardSwish

ConvTranpose 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.30 ConvTranposeElu

ConvTranpose 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.31 ConvTranposeSwish

ConvTranpose 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.32 ConvTranposeMish

ConvTranpose 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.2.33 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

其他支持

- 多核联合: 不支持

3.2.34 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`: 维度右对齐; 两个对应维度必须具有相同的值, 或者其中一个等于 1

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持

- ★ per-layer
- shape : **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - ★ per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]->[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

其他支持

- 多核联合: 不支持

3.2.35 GRU

门控循环单元（GRU，Gated Recurrent Unit）

GRU 扩展以及变体命名为 exGRU 算子，参数项中指明

（extern）的项为 exGRU 独有的参数项。

计算公式如下：

Equations (Default: f=Sigmoid, g=Tanh):

$$r_t = f(x_t \cdot W_r + h_{t-1} \cdot R_r + Wb_r + Rb_r)$$

$$z_t = f(x_t \cdot W_z + h_{t-1} \cdot R_z + Wb_z + Rb_z)$$

$$h_t = g(x_t \cdot W_h + (r_t \odot h_{t-1}) \cdot R_h + Rb_h + Wb_h), (WRB = 0)$$

$$h_t = g(x_t \cdot W_h + r_t \odot (h_{t-1} \cdot R_h + Rb_h)) + Wb_h, (WRB = 1)$$

$$H_t = (1 - z_t) \odot h_t + z_t \odot h_{t-1}$$

输入列表

- X : **T1**
 - **shape** : [seq_length, batch_size, input_size]
 - ★ seq_length $\in (0, 8192]$

- * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $inputsize \in (0, 8192]$
- 量化支持
 - * per-layer
- W: **T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * $inputsize \in (0, 8192]$
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2**
 - **shape**: [num_directions, 8*hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T3** (optional)
 - **shape**: [batch_size]
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]

- * $seqlength \in (0, 8192]$
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
- 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- linear_before_reset: int
 - LBR 变种的选择: 1(T) or 0(F)
- direction (extern): string
 - 指定 GRU 的运算方向
 - forward: 指定 GRU 的运算方向为前向
 - reverse: 指定 GRU 的运算方向为反向
 - bidirectional: 指定 GRU 的运算方向为双向
- sequence_size (extern): int
 - 指定 GRU 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - GRU 单元中的 hiddensize, 无限制, 建议 8 对齐
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size, 1, 1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size, 1, 1]

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)

- **T3**: float(Scalar)

其他支持

- 多核联合: 暂不支持

3.2.36 Gather

根据索引 Indices 获取输入 X 的指定 axis 维度的条目，并将它们拼接在一起。

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- Indices: **T2**
 - 要收集的元素的索引，秩 $rank = 1$

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int(Tensor)

属性列表

- axis: int(Tensor)
 - $axis \in \{0, 1, 2, 3\}$
 - 指定 Indices 获取输入的维度

其他支持

- 多核联合: 尚不支持

3.2.37 Hardmax

输出列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * axis=1, $height * width < 16384$;
 - * axis=3/-1, $channel * height < 16384$, 且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 hardmax 的轴: 1,3/-1, 即 channel 和 width 方向

其他支持

- **多核联合**: 尚不支持

3.2.38 LSTM

长短期记忆网络 (LSTM, Long Short-Term Memory)

LSTM 扩展以及变体命名为 exLSTM 算子, 参数项中指明

(extern) 的项为 exLSTM 独有的参数项。

计算公式如下:

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C(t-1) + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- **X: T1**
 - **shape**: [seq_length, batch_size, input_size]
 - * *seqlength* $\in (0, 8192]$
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *inputsize* $\in (0, 8192]$
 - **量化支持**
 - * per-layer
- **W: T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * *inputsize* $\in (0, 8192]$
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2

- 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2, T3**
 - **shape**: [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T4, T5** (optional)
 - **shape**: [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- initial_c: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]
 - * *seqlength* $\in (0, 8192]$
 - * num directions: 1 or 2

- * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
- 量化支持
 - * per-layer
- Y_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_c: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向
 - bidirectional: 指定 LSTM 的运算方向为双向
- sequence_size (extern): int
 - 指定 LSTM 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - LSTM 单元中的 hiddensize, 无限制, 建议 8 对齐
- proj_size (extern): int
 - projection 时的 proj_size, $projsize \in [0, hiddensize]$
 - 目前限定 0, 即尚不支持 projection 功能
- input_forget (extern): int
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0, 即尚不支持
- has_projection (extern): int
 - projection 变种: 1(T) or 0(F) 目前限定 0, 即尚不支持
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。

- snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
- (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size,1,1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size,1,1]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int32(Tensor)
- **T4**: float(Scalar)
- **T5**: int(Scalar)

其他支持

- **多核联合**: 暂不支持

3.2.39 Max

计算输入张量的元素级最大值

输入列表

- **A: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W)), 即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1)), 即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar)), 即以单个标量做 broadcasting
- **B: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W)), 即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1)), 即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar)), 即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.2.40 Min

计算输入张量的元素级最小值

输入列表

- A: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W)) , 即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1)) , 即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar)) , 即以单个标量做 broadcasting
- B: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W)) , 即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1)) , 即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar)) , 即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
-

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.41 Mul

执行元素级二进制乘法 ($C = A * B$), 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(const Tensor)
-

其他支持

- 多核联合: 支持

3.2.42 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul
-

其他支持

- 多核联合: 支持

3.2.43 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value),

生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]

width_end]

- constant_value (optional): **T3**
 - **shape** : [1]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: int64(Tensor)
 - **T3**: float(Scalar), int8(Scalar), float16(Scalar)
-

属性列表

- mode : **string**(默认值 constant)
 - 支持模式: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$

其他支持

- 多核联合: 尚不支持

3.2.44 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * channel $\in (0, 8192]$
 - * height $\in (0, 8192]$
 - * width $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)

- $\text{pad_top} \in [0, 7]$
- $\text{pad_left} \in [0, 7]$
- $\text{pad_bottom} \in [0, 7]$
- $\text{pad_right} \in [0, 7]$
- $\text{storage_order} : \text{int64}$
 - 优先储存方式: 0

数据类型约束

- **T1**: $\text{int8}(\text{Tensor})$, $\text{float16}(\text{Tensor})$
- **T2**: $\text{float}(\text{Tensor})$
- **T3**: $\text{float}(\text{Scalar})$

3.2.45 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $\text{channel} \in (0, 8192]$
 - * $\text{height} \in (0, 7]$
 - * $\text{width} \in (0, 7]$

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

属性列表

- $\text{kernel_shape} : \text{int64}(\text{kernel_h}, \text{kernel_w})$
 - $\text{kernel_h} \in (0, 7]$
 - $\text{kernel_w} \in (0, 7]$
- $\text{auto_pad} : \text{string}$
 - pad 的方式: 仅支持 NOTSET
- $\text{ceil_mode} : \text{int64}$
 - 使用 ceil 或 floor 的方式计算输出的 shape: 不支持
- $\text{strides} : \text{int64}(\text{strides_h}, \text{strides_w})$
 - $\text{strides_h} \in (0, 8]$
 - $\text{strides_w} \in (0, 8]$
- $\text{count_include_pad} : \text{int64}[]$
 - count_include_pad 是否包含 pad 数值进行计算: 1

- `pad : int64(pad_top, pad_left, pad_bottom, pad_right)`
 - `pad_top` $\in [0, 7]$
 - `pad_left` $\in [0, 7]$
 - `pad_bottom` $\in [0, 7]$
 - `pad_right` $\in [0, 7]$

数据类型约束

- **T1**: `int8(Tensor)`, `float16(Tensor)`
- **T2**: `float(Tensor)`
- **T3**: `float(Scalar)`

3.2.46 AveragePool

AveragePool 使用输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用平均池化。平均池化包括根据内核大小计算输入张量子集的所有值的平均值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- **input : T1**
 - **shape** : `[batch, channel, height, width]`
 - * `channel` $\in (0, 8192]$
 - * `height` $\in (0, 8192]$
 - * `width` $\in (0, 8192]$

输出列表

- **output : T1**
 - **shape** : `[batch, channel, height, width]`

属性列表

- `kernel_shape: int64(kernel_h, kernel_w)`
 - `kernel_h` $\in (0, 7]$
 - `kernel_w` $\in (0, 7]$
- `auto_pad: string`
 - pad 的方式: 仅支持 NOTSET
- `ceil_mode: int64`
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- `strides : int64(strides_h, strides_w)`
 - `strides_h` $\in (0, 8]$
 - `strides_w` $\in (0, 8]$
- `count_include_pad : int64[]`
 - `count_include_pad` 是否包含 pad 数值进行计算: 1
- `pad : int64(pad_top, pad_left, pad_bottom, pad_right)`

- $\text{pad_top} \in [0, 7]$
- $\text{pad_left} \in [0, 7]$
- $\text{pad_bottom} \in [0, 7]$
- $\text{pad_right} \in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.2.47 GlobalAveragePool

GlobalAveragePool 使用输入张量 X 并对同一通道中的值应用平均池化。这相当于 AveragePool，其 kernel_shape 大小等于输入张量的空间维度。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $\text{channel} \in (0, 8192]$
 - * $\text{height} \in (0, 7]$
 - * $\text{width} \in (0, 7]$

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - $\text{kernel_h} \in (0, 7]$
 - $\text{kernel_w} \in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape: 不支持
- strides: int64(strides_h, strides_w)
 - $\text{strides_h} \in (0, 8]$
 - $\text{strides_w} \in (0, 8]$
- count_include_pad: int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad: int64(pad_top, pad_left, pad_bottom, pad_right)
 - $\text{pad_top} \in [0, 7]$
 - $\text{pad_left} \in [0, 7]$
 - $\text{pad_bottom} \in [0, 7]$

- $\text{pad_right} \in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.2.48 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- **Y: T2**
 - **shape**: [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- **Z: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)
-

其他支持

- 多核联合: 尚不支持

3.2.49 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
-

- 广播支持

- * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.50 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.51 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.52 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
-

- * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.2.53 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- shape: int
 - 指定输出张量的形状

其他支持

- 多核联合: 尚不支持

3.2.54 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - roi (optional): **T2**
-

- 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

其他支持

- **多核联合**: 不支持

3.2.55 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴，支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 height * width == 1 && channel % subc == 0 && starts[1] % subc == 0 && ends[1] % subc == 0 && steps[1] <= subc 时, steps 可以不为 1，否则只有 steps == [1,1,1,1] 时，才可通过 NPU 运行 Slice OP

3.2.56 Softmax

该运算符计算给定输入的归一化指数值：

$$\text{Softmax}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

其他支持

- **多核联合**: 尚不支持

3.2.57 exSoftmax13

该运算符计算给定输入的归一化指数值:

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
- 受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

其他支持

- **多核联合**: 尚不支持

3.2.58 exSoftmaxMask

该运算符计算给定输入的归一化指数值：

$$input(x, mast_t, mask_value) = \begin{cases} x + (mast_t - 1) * inf & \text{if } mask_value = 0 \\ x + mast_t * (-inf) & \\ \text{elif } mask_value = 1 \end{cases}$$

$$exSoftmaxMask(input, axis) = \frac{\exp(input - \max(input, axis))}{\sum_{axis} \exp(input - \max(input, axis))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
受限于 tranpose 的规格限制
- mask (optional): **T1**
 - **shape**: [1, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $width$:
 - axis=1, 无限制;
 - axis=3/-1, $width \in (0, 8192]$

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向
- mask_value: int64
 - 需要 mask 的值: 0,1

其他支持

- 多核联合: 尚不支持

3.2.59 Split

将向量沿着 axis 方向分成 num_outputs 个向量，split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - 一或多个输出向量，由 num_outputs 属性决定
 - **shape**: 根据切分情况决定
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs: int
- split: int(Tensor)

3.2.60 Sub

- 执行元素级二进制减法 ($C = A - B$)，支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持

- * [b, c, 1, 1]
- * [scalar]

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(const Tensor)
-

其他支持

- 多核联合: 不支持

3.2.61 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats: **T2**
 - **shape**: [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: int64(Tensor)
-

其他支持

- 多核联合: 不支持

3.2.62 Transpose

对输入张量进行转置

输入列表

- X: **T1**
 - **shape**: [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

输出列表

- Y: **T1, T2**
 - **shape**: [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

属性列表

- perm 转置的轴顺序:
不同 perm 参数限制如下: (其中 p=8192, q=2048)
 - perm=[0,1,2,3]
 - * 无限制
 - perm=[0,1,3,2]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
 - perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 16$
 - * **T2**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
 - perm=[0,2,1,3]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - perm=[0,3,1,2]
 - * **T1, T2**: $h1 * w1 < p, w1 < 8192, h1 * c1 < p$
 - perm=[0,3,2,1]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 < p * 16$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 < p * 8$
 - perm=[1,0,2,3]
 - * **T1**: $h1 * w1 < 16 * p, n1 < 8192, h1 * w1 < p$
 - * **T2**: $h1 * w1 < 8 * p, n1 < 8192, h1 * w1 < p$
 - perm=[1,0,3,2]

- * **T1**: $h1 \% 16 = 0, c1 * h1 * w1 < 16 * p, h1 * w1 < 2 * q$
- * **T2**: $h1 \% 8 = 0, c1 * h1 * w1 < 8 * p, h1 * w1 < 2 * q$
- perm=[1,2,0,3]
 - * **T1**: $n1 * w1 < p * 16, c1 * h1 * w1 < p, w1 \% 16 = 0$
 - * **T2**: $n1 * w1 < p * 8, c1 * h1 * w1 < p, w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1**: $h1 * w1 < p, n1 < p * 16, c1 * h1 * w1 < p * 16, n1 * w1 < p * 16$
 - * **T2**: $h1 * w1 < p, n1 < p * 8, c1 * h1 * w1 < p * 8, n1 * w1 < p * 8$
- perm=[1,3,0,2]
 - * **T1**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 16 = 0$
 - * **T2**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p, n1 * w1 < p$
- perm=[2,1,3,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * w1 < p * 16, c1 < 8192, n1 * w1 \% 4 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * w1 < p * 8, c1 < 8192, n1 * w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p$
- perm=[2,0,1,3]
 - * **T1**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 16 = 0$
 - * **T2**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1, T2**: $h1 * w1 < 8192, n1 * c1 < p$
- perm=[3,0,2,1]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 8 = 0$
- perm=[3,2,0,1]
 - * **T1, T2**: $h1 * w1 < p, n1 * c1 < 8192, n1 * c1 * w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1 * h1 * w1 < p, h1 * w1 < 8192, 2 * h1 * w1 < q, n1 * c1 < p * 16, h1 \% 16 = 0$

- **T2**: $c1 \cdot h1 \cdot w1 < p$, $h1 \cdot w1 < 8192$, $2 \cdot h1 \cdot w1 < q$, $n1 \cdot c1 < p \cdot 8$, $h1 \% 8 = 0$
- perm=[3,1,0,2]
- **T1, T2**: $h1 \cdot w1 < p$, $c1 \cdot w1 < 8192$, $n1 \cdot h1 < p$
- perm=[3,1,2,0]
- **T1, T2**: $h1 \cdot w1 < p$, $n1 \cdot c1 < 8192$, $n1 \cdot h1 < p$, $n1 \cdot c1 \cdot w1 < p$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.2.63 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.64 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**

- **shape** : [batch, channel, height, width]
- 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.65 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.66 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.67 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.68 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.69 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- **多核联合**: 尚不支持

3.2.70 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- **多核联合**: 尚不支持

3.2.71 exGelu

给定输入张量，函数 $y = x * \Phi(x)$ 按元素应用于输入张量，得到输出张量，当 $\Phi(x)$ 函数设置成 \tanh 时， $y = 0.5 * x * (1 + \tanh(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.72 exSwooshR

给定输入张量，按论文 <https://arxiv.org/abs/2310.11230> 给定的下列函数，按元素应用于输入张量，得到输出张量

$$SwooshR(x) = \log(1 + \exp(x-1)) - 0.08x - 0.313261687$$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.73 exSwooshL

给定输入张量，按论文 <https://arxiv.org/abs/2310.11230> 给定的下列函数，按元素应用于输入张量，得到输出张量

$$SwooshL(x) = \log(1 + \exp(x - 4)) - 0.08x - 0.035$$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.2.74 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor: *T2
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

- **y_tensor: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 支持任意维度广播操作
 - **广播约束**
 - * 广播约束同Expand算子

输出列表

- **output: T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor), float16(Tensor)

3.2.75 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- **Input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Output: T1**
 - **shape**: [batch, channel / 2, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- **axis**: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

其他支持

- **多核联合**: 尚不支持

3.2.76 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- A: **T1**
 - **shape**: [b, k, 1, n]
 - * $k \in (0, 8192]$
 - 量化支持
 - * per-layer
- B: **T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- C (optional): **T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- Y: **T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)

属性列表

- c_type: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

其他支持

- 多核联合: 不支持

3.2.77 exNorm

对输入数据进行各种归一化操作，目前支持 LayerNormlization、RMSNormlization 和 MeanVarianceNormalization 三种：

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot W + B$$

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{RMS^2 + \epsilon}} \cdot W + B$$

$$\text{MeanVarianceNorm}(x) = \frac{x - \mu}{\sqrt{Variance + \epsilon}} \cdot W + B$$

其中：

x 是输入向量。

μ 和 σ^2 分别是输入向量的均值和方差：

$$\mu = \frac{1}{H} \sum_{i=1}^H x_i$$

$$\sigma^2 = \frac{1}{H} \sum_{i=1}^H (x_i - \mu)^2$$

RMS 是输入向量均方根：

$$RMS = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i)^2}$$

$Variance$ 是输入向量的等效方差：

$$Variance = \frac{1}{H} \sum_{i=1}^H (x_i)^2 - \mu^2$$

ϵ 是一个很小的数，用于防止除零操作。

W 和 B 是可选择的参数。

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
- **W (optional): T1**
 - **shape**: [batch, channel, height, width]
- **B (optional): T1**
 - **shape**: [batch, channel, height, width]

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- epsilon : **float**(默认值 1e-05)
 - 用来防止除 0 的 epsilon 数值
- mode : **string**
 - 支持 layer、rms 和 mean_var 三种 Normalization 模式可选
- axes : **int64[]**(默认值 [1])
 - 归一化轴, 目前只支持 [1] 和 [1,2,3]

其他支持

- 多核联合: 尚不支持

3.2.78 exSDPAttention

计算查询 (query)、键 (key) 和值 (value) 之间的缩放点积注意力 (SDPA)

输入列表

- query : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- key : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- value : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- mask (optional) : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
- scale (optional) : **T3**

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

其他支持

- 多核联合: 不支持

3.2.79 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。原论文中输入特征被划分为多个不重叠的窗口，每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape，在高和宽方向上都进行划分。pytorch 实现的计算方法如下：

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
    windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
    return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
```

(下页继续)

(续上页)

```

windows: (B, C, num_windows*num_windows, window_size*window_size)

window_size (int): Window size

H (int): Height of image

W (int): Width of image

Returns:

x: (B, C, H, W)

"""

B,C,_,_ = windows.shape

x = windows.view(B, C, H // window_size, W // window_size, window_size, window_size)

x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)

return x

```

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- mode: **string**
 - **支持模式**: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;
 - * partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
- window_sizes: **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
- num_windows: **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。

其他支持

- **多核联合:** 尚不支持

3.3 RK3562

3.3.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.3.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add

其他支持

- 多核联合: 支持

3.3.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- epsilon : float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum : float
 - 训练时的滑动平均参数

3.3.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.3.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - $stride_h \in [1, 7]$
 - $stride_w \in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - $pads_top \in [0, 15]$
 - $pads_left \in [0, 15]$
 - $pads_bottom \in [0, 15]$
 - $pads_right \in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - $dilations_h \in [1, 32]$

- `dilations_w` $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.6 Depthwise Convolution

深度可分离卷积

输入列表

- `input_tensor`: **T1**
 - **shape**: [batch, channel, height, width]
 - * `width`: 当`dilated_kernel_h` > 1 时, `width` < 16383。此外, 对首层输入 `width` 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- `weight`: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * `kernel_h` $\in [1, 8]$
 - * `kernel_w` $\in [1, 8]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- `output`: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

属性列表

- `strides`: int(strides_h, strides_w)
 - `stride_h` $\in [1, 7]$
 - `stride_w` $\in [1, 7]$
- `pads`: int(pads_top, pads_left, pads_bottom, pads_right)
 - `pads_top` $\in [0, 15]$
 - `pads_left` $\in [0, 15]$
 - `pads_bottom` $\in [0, 15]$
 - `pads_right` $\in [0, 15]$
- `group`: int
- `dilations`: int(dilations_h, dilations_w)

- dilations_h $\in [1, 32]$
- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.3.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.3.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.3.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

3.3.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.3.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.3.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同Convolution

3.3.19 ConvolutionAddRelu

Convolution 与 Add 及 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Add}(\text{Conv}(\text{Input0}), \text{Input1}))$

说明及规格限制同Convolution

3.3.20 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 8]$
 - stride_w $\in [1, 8]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.21 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.22 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.23 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{ConvTranpose}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.24 ConvTranposeAdd

ConvTranpose 与 Add 融合计算, $\text{Output} = \text{Add}(\text{ConvTranpose}(\text{Input0}), \text{Input1})$

说明及规格限制同[ConvTranspose](#)

3.3.25 ConvTranposeSigmoid

ConvTranpose 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.26 ConvTranposeTanh

ConvTranpose 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.27 ConvTranposeSoftplus

ConvTranpose 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.28 ConvTranposeHardSigmoid

ConvTranpose 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{ConvTranpose}(\text{Input}))$

3.3.29 ConvTranposeHardSwish

ConvTranpose 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.30 ConvTranposeElu

ConvTranpose 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.31 ConvTranposeSwish

ConvTranpose 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.32 ConvTranposeMish

ConvTranpose 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.3.33 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.3.34 Expand

输入列表

- input1: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

广播约束

- $[b, c, 1, w] \rightarrow [b, c, h, w], h \in [1, 8192]$
- $[b, 1, 1, w] \rightarrow [b, c, h, w], h \in [1, 8192]$
- $[b, c, h, 1] \rightarrow [b, c, h, w], w \in [1, 8192]$
- $[b, 1, h, 1] \rightarrow [b, c, h, w], w \in [1, 8192]$
- $[b, c, 1, 1] \rightarrow [b, c, h, w], h \in [1, 8192], w \in [1, 8192]$

3.3.35 GRU

门控循环单元（GRU，Gated Recurrent Unit）

GRU 扩展以及变体命名为 exGRU 算子，参数项中指明

（extern）的项为 exGRU 独有的参数项。

计算公式如下：

Equations (Default: f=Sigmoid, g=Tanh):

$$r_t = f(x_t \cdot W_r + h_{t-1} \cdot R_r + Wb_r + Rb_r)$$

$$z_t = f(x_t \cdot W_z + h_{t-1} \cdot R_z + Wb_z + Rb_z)$$

$$h_t = g(x_t \cdot W_h + (r_t \odot h_{t-1}) \cdot R_h + Rb_h + Wb_h), (WRB = 0)$$

$$h_t = g(x_t \cdot W_h + r_t \odot (h_{t-1} \cdot R_h + Rb_h)) + Wb_h, (WRB = 1)$$

$$H_t = (1 - z_t) \odot h_t + z_t \odot h_{t-1}$$

输入列表

- **X: T1**
 - **shape**: [seq_length, batch_size, input_size]
 - * $seqlength \in (0, 8192]$
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $inputsize \in (0, 8192]$
 - **量化支持**
 - * per-layer
- **W: T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * $inputsize \in (0, 8192]$
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - **量化支持**
 - * per-layer
 - * per-channel

- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2**
 - **shape**: [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T3** (optional)
 - **shape**: [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]
 - * *seqlength* $\in (0, 8192]$
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持

* per-layer

属性列表

- linear_before_reset: int
 - LBR 变种的选择: 1(T) or 0(F)
- direction (extern): string
 - 指定 GRU 的运算方向
 - forward: 指定 GRU 的运算方向为前向
 - reverse: 指定 GRU 的运算方向为反向
 - bidirectional: 指定 GRU 的运算方向为双向
- sequence_size (extern): int
 - 指定 GRU 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - GRU 单元中的 hiddensize, 无限制, 建议 8 对齐
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size,1,1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size,1,1]

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

其他支持

- **多核联合**: 暂不支持

3.3.36 Gather

根据索引 Indices 获取输入 X 的指定 axis 维度的条目, 并将它们拼接在一起。

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**

- * per-layer
- Indices: **T2**
 - 要收集的元素的索引, 秩 $rank = 1$

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int(Tensor)

属性列表

- axis: int(Tensor)
 - $axis \in \{0, 1, 2, 3\}$
 - 指定 Indices 获取输入的维度

其他支持

- 多核联合: 尚不支持

3.3.37 Hardmax

输出列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $axis=1, height * width < 16384$;
 - * $axis=3/-1, channel * height < 16384$, 且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 hardmax 的轴: 1,3/-1, 即 channel 和 width 方向

其他支持

- 多核联合: 尚不支持

3.3.38 LSTM

长短期记忆网络 (LSTM, Long Short-Term Memory)

LSTM 扩展以及变体命名为 exLSTM 算子, 参数项中指明

(extern) 的项为 exLSTM 独有的参数项。

计算公式如下:

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C_{(t-1)} + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- X: **T1**

- **shape**: [seq_length, batch_size, input_size]

- * *seqlength* $\in (0, 8192]$

- * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数

- * *inputsize* $\in (0, 8192]$

- 量化支持

- * per-layer

- W: **T2**

- **shape**: [num_directions, 4*hidden_size, input_size]

- * *inputsize* $\in (0, 8192]$

- * *hiddensize* $\in (0, 8192]$

- * num directions: 1 or 2

- 量化支持

- * per-layer

- * per-channel

- R: **T2**

- **shape**: [num_directions, 4*hidden_size, hidden_size]

- * *hiddensize* $\in (0, 8192]$

- * num directions: 1 or 2

- 量化支持

- * per-layer

- * per-channel

- B: **T2**

- **shape** : [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
- 量化支持
 - * per-layer
 - * per-channel
- sequence_lens : **T2** (optional)
 - **shape** : [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- initial_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y : **T1**
 - **shape** : [seq_length, num_directions, batch_size, hidden_size]
 - * *seqlength* $\in (0, 8192]$
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

- **Y_c: T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - **量化支持**
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向
 - bidirectional: 指定 LSTM 的运算方向为双向
- sequence_size (extern): int
 - 指定 LSTM 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - LSTM 单元中的 hiddensize, 无限制, 建议 8 对齐
- proj_size (extern): int
 - projection 时的 proj_size, *projsize* $\in [0, hiddensize]$
 - 目前限定 0, 即尚不支持 projection 功能
- input_forget (extern): int
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0, 即尚不支持
- has_projection (extern): int
 - projection 变种: 1(T) or 0(F) 目前限定 0, 即尚不支持
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size, 1, 1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size, 1, 1]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int32(Tensor)

- **T4**: float(Scalar)

其他支持

- **多核联合**: 暂不支持

3.3.39 Max

计算输入张量的元素级最大值

输入列表

- **A: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W))，即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1))，即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar))，即以单个标量做 broadcasting
- **B: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W))，即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1))，即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar))，即以单个标量做 broadcasting

输出列表

- **C: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

3.3.40 Mul

执行元素级二进制乘法 ($C = A * B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.3.41 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul
-

其他支持

- 多核联合: 支持
-

3.3.42 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape** : [1]
 - **量化支持**
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)
- **T3**: float(Scalar), int8(Scalar), float16(Scalar)

属性列表

- mode : **string**(默认值 constant)
 - **支持模式**: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$

其他支持

- **多核联合**: 尚不支持

3.3.43 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (0, 8192]$
 - * *width* $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$
- storage_order :int64[]
 - 优先储存方式: 0

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.3.44 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.3.45 AveragePool

AveragePool 使用输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用平均池化。平均池化包括根据内核大小计算输入张量子集的所有值的平均值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (0, 8192]$
 - * *width* $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.3.46 GlobalAveragePool

GlobalAveragePool 使用输入张量 X 并对同一通道中的值应用平均池化。这相当于 AveragePool，其 kernel_shape 大小等于输入张量的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.3.47 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- Y: **T2**
 - **shape**: [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- Z: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)

其他支持

- 多核联合: 尚不支持

3.3.48 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持

- * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.3.49 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.3.50 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel
 - 广播支持
-

- * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.3.51 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.3.52 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- shape : int
 - 指定输出张量的形状

其他支持

- **多核联合**: 尚不支持

3.3.53 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- roi (optional) : **T2**
 - 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

3.3.54 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴, 支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 $height * width == 1 \ \&\& \ channel \% subc == 0 \ \&\& \ starts[1] \% subc == 0 \ \&\& \ ends[1] \% subc == 0 \ \&\& \ steps[1] \leq subc$ 时, steps 可以不为 1, 否则只有 steps == [1,1,1,1] 时, 才可通过 NPU 运行 Slice OP

3.3.55 Softmax

该运算符计算给定输入的归一化指数值:

$$\text{Softmax}(input, axis) = \frac{\exp(input - \max(input, axis))}{\sum_{axis} \exp(input - \max(input, axis))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 受限与 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.3.56 exSoftmax13

该运算符计算给定输入的归一化指数值：

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 受限干 tranpose 的规格限制

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.3.57 exSoftmaxMask

该运算符计算给定输入的归一化指数值：

$$\text{input}(x, \text{mast}_t, \text{mask_value}) = \begin{cases} x + (\text{mast}_t - 1) * \text{inf} & \text{if } \text{mask_value} = 0 \\ x + \text{mast}_t * (-\text{inf}) & \text{elif } \text{mask_value} = 1 \end{cases}$$

$$\text{exSoftmaxMask}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 受限干 tranpose 的规格限制
- mask (optional) : **T1**
 - **shape** : [1, channel, height, width]

- * $channel \in (0, 8192]$
- * $width$:
 - axis=1, 无限制;
 - axis=3/-1, $width \in (0, 8192]$

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向
- mask_value: int64
 - 需要 mask 的值: 0,1

3.3.58 Split

将向量沿着 axis 方向分成 num_outputs 个向量, split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T**
 - 一或多个输出向量, 由 num_outputs 属性决定
 - **shape**: 根据切分情况决定
 - **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs: int
- split: int(Tensor)

3.3.59 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播
-

输入列表

- A : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(const Tensor)

3.3.60 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats : **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

3.3.61 Transpose

对输入张量进行转置

输入列表

- X: **T1**
 - **shape**: [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

输出列表

- Y: **T1, T2**
 - **shape**: [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

属性列表

- perm 转置的轴顺序:
不同 perm 参数限制如下: (其中 p=268435456, q=65536)
 - perm=[0,1,2,3]
 - * 无限制
 - perm=[0,1,3,2]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
 - perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 16$
 - * **T2**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
 - perm=[0,2,1,3]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - perm=[0,3,1,2]

- * **T1, T2:** $h1*w1 < p$, $w1 < 8192$, $h1*c1 < p$
- perm=[0,3,2,1]
 - * **T1:** $h1 \% 16 = 0$, $h1*w1 < 16*p$, $h1*w1 < 2*q$, $c1 < p*16$
 - * **T2:** $h1 \% 8 = 0$, $h1*w1 < 8*p$, $h1*w1 < 2*q$, $c1 < p*8$
- perm=[1,0,2,3]
 - * **T1:** $h1*w1 < 16*p$, $n1 < 8192$, $h1*w1 < p$
 - * **T2:** $h1*w1 < 8*p$, $n1 < 8192$, $h1*w1 < p$
- perm=[1,0,3,2]
 - * **T1:** $h1 \% 16 = 0$, $c1*h1*w1 < 16*p$, $h1*w1 < 2*q$
 - * **T2:** $h1 \% 8 = 0$, $c1*h1*w1 < 8*p$, $h1*w1 < 2*q$
- perm=[1,2,0,3]
 - * **T1:** $n1*w1 < p*16$, $c1*h1*w1 < p$, $w1 \% 16 = 0$
 - * **T2:** $n1*w1 < p*8$, $c1*h1*w1 < p$, $w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1:** $h1*w1 < p$, $n1 < p*16$, $c1*h1*w1 < p*16$, $n1*w1 < p*16$
 - * **T2:** $h1*w1 < p$, $n1 < p*8$, $c1*h1*w1 < p*8$, $n1*w1 < p*8$
- perm=[1,3,0,2]
 - * **T1:** $c1*w1 < 8192$, $n1*h1 < p$, $w1 \% 16 = 0$
 - * **T2:** $c1*w1 < 8192$, $n1*h1 < p$, $w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1:** $c1*h1*w1 < p$, $n1*h1 < p$, $w1 < 8192$, $w1 \% 16 = 0$
 - * **T2:** $c1*h1*w1 < p$, $n1*h1 < p$, $w1 < 8192$, $w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1, T2:** $c1 < 8192$, $c1*h1*w1 < p$, $n1*w1 < p$
- perm=[2,1,3,0]
 - * **T1:** $c1*h1*w1 < p$, $n1*w1 < p*16$, $c1 < 8192$, $n1*w1 \% 4 = 0$
 - * **T2:** $c1*h1*w1 < p$, $n1*w1 < p*8$, $c1 < 8192$, $n1*w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1, T2:** $c1 < 8192$, $c1*h1*w1 < p$
- perm=[2,0,1,3]
 - * **T1:** $n1*c1*h1*w1 < p$, $h1*w1 \% 16 = 0$
 - * **T2:** $n1*c1*h1*w1 < p$, $h1*w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1:** $c1*h1*w1 < p$, $n1*c1 < p$, $h1*w1 < 8192$, $h1*w1 \% 16 = 0$
 - * **T2:** $c1*h1*w1 < p$, $n1*c1 < p$, $h1*w1 < 8192$, $h1*w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1, T2:** $h1*w1 < 8192$, $n1*c1 < p$
- perm=[3,0,2,1]
 - * **T1:** $h1*w1 < p$, $n1*c1 < 8192$, $c1*h1 < p$, $n1*w1 < 8192$, $n1*w1 \% 16 = 0$

- * **T2**: $h1*w1 < p, n1*c1 < 8192, c1*h1 < p, n1*w1 < 8192, n1*w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1*w1 < p, n1*c1 < 8192, c1*h1 < p, n1*w1 < 8192, n1*h1*w1 \% 16 = 0$
 - * **T2**: $h1*w1 < p, n1*c1 < 8192, c1*h1 < p, n1*w1 < 8192, n1*h1*w1 \% 8 = 0$
- perm=[3,2,0,1]
 - * **T1, T2**: $h1*w1 < p, n1*c1 < 8192, n1*c1*w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1*h1*w1 < p, h1*w1 < 8192, 2*h1*w1 < q, n1*c1 < p*16, h1 \% 16 = 0$
 - * **T2**: $c1*h1*w1 < p, h1*w1 < 8192, 2*h1*w1 < q, n1*c1 < p*8, h1 \% 8 = 0$
- perm=[3,1,0,2]
 - * **T1, T2**: $h1*w1 < p, c1*w1 < 8192, n1*h1 < p$
- perm=[3,1,2,0]
 - * **T1, T2**: $h1*w1 < p, n1*c1 < 8192, n1*h1 < p, n1*c1*w1 < p$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.3.62 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.63 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.64 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.65 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.66 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.67 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.68 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.69 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.70 exGelu

给定输入张量, 函数 $y = x * \Phi(x)$ 按元素应用于输入张量, 得到输出张量, 当 $\Phi(x)$ 函数设置成 \tanh 时, $y = 0.5 * x * (1 + \text{Tanh}(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- **input: T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.3.71 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor : *T2
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

输出列表

- output : **T2**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor), float16(Tensor)

3.3.72 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

3.3.73 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- A: **T1**
 - **shape** : [b, k, 1, n]
 - * $k \in (0, 8192]$
 - 量化支持

- * per-layer
- B: **T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- C (optional): **T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- Y: **T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)

属性列表

- c_type: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

3.3.74 exNorm

对输入数据进行各种归一化操作，目前支持 LayerNormlization、RMSNormlization 和 MeanVarianceNormalization 三种：

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot W + B$$

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\text{RMS}^2 + \epsilon}} \cdot W + B$$

$$\text{MeanVarianceNorm}(x) = \frac{x - \mu}{\sqrt{\text{Variance} + \epsilon}} \cdot W + B$$

其中：

x 是输入向量。

μ 和 σ^2 分别是输入向量的均值和方差：

$$\mu = \frac{1}{H} \sum_{i=1}^H x_i$$

$$\sigma^2 = \frac{1}{H} \sum_{i=1}^H (x_i - \mu)^2$$

RMS 是输入向量均方根：

$$RMS = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i)^2}$$

$Variance$ 是输入向量的等效方差：

$$Variance = \frac{1}{H} \sum_{i=1}^H (x_i)^2 - \mu^2$$

ϵ 是一个很小的数，用于防止除零操作。

W 和 B 是可选的参数。

输入列表

- X : **T1**
 - **shape**: [batch, channel, height, width]
- W (optional): **T1**
 - **shape**: [batch, channel, height, width]
- B (optional): **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y : **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- epsilon: **float**(默认值 1e-05)
 - 用来防止除 0 的 epsilon 数值
- mode: **string**
 - 支持 layer、rms 和 mean_var 三种 Normalization 模式可选
- axes: **int64[]**(默认值 [1])
 - 归一化轴，目前只支持 [1] 和 [1,2,3]

其他支持

- 多核联合: 尚不支持

3.3.75 exSDPAttention

计算查询 (query)、键 (key) 和值 (value) 之间的缩放点积注意力 (SDPA)

输入列表

- query : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- key : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- value : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- mask (optional) : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 默认支持任意维度广播操作
 - * 开启 FA 支持后, 只支持 [b,c,1,1], [1,c,1,1] 和 [1,c,h,w] 三种规格
- scale (optional) : **T3**

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(Tensor)
 - **T3**: float(Scalar)
-

其他支持

- **FlashAttentionV2(FA)** : 支持
-

- 基于 <https://arxiv.org/abs/2307.08691> 实现, 通过高速缓存内循环实现加速以及减少宽带使用, 但是会导致模型增大, 请根据具体场景和模型选择是否开启使用
- RKNPU 驱动版本 $\geq 0.8.8$
- RKNPU Runtime 库 (librknrt.so) 版本 $\geq 2.0.0$
- 支持数据类型: float16

3.3.76 exWindow

exWindow 来源于 `swin_transformer` 的 `window_partition` 和 `window_reverse` 操作。

原论文中输入特征被划分为多个不重叠的窗口, 每个窗口的大小为 `window_size` \times `window_size`。要求输入和输出特征都为 4 维 shape, 在高和宽方向上都进行划分。

pytorch 实现的计算方法如下:

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
    windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
    return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
        window_size (int): Window size
        H (int): Height of image
        W (int): Width of image

    Returns:
```

(下页继续)

(续上页)

```

x: (B, C, H, W)

"""

B,C,_,_ = windows.shape

x = windows.view(B, C, H // window_size, W // window_size, window_size, window_size)

x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)

return x

```

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]

输出列表

- Y: **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- mode: **string**
 - 支持模式: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;
 - * partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
- window_sizes: **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
- num_windows: **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。

其他支持

- 多核联合: 尚不支持

3.4 RV1103/RV1106

3.4.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.4.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add
-

其他支持

- 多核联合: 支持

3.4.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- epsilon : float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum : float
 - 训练时的滑动平均参数

3.4.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.4.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - $stride_h \in [1, 7]$
 - $stride_w \in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - $pads_top \in [0, 15]$
 - $pads_left \in [0, 15]$
 - $pads_bottom \in [0, 15]$
 - $pads_right \in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - $dilations_h \in [1, 32]$

- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.6 Depthwise Convolution

深度可分离卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 8]$
 - * kernel_w $\in [1, 8]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)

- dilations_h $\in [1, 32]$
- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

3.4.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.4.19 ConvolutionAddRelu

Convolution 与 Add 及 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Add}(\text{Conv}(\text{Input0}), \text{Input1}))$

说明及规格限制同[Convolution](#)

3.4.20 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 8]$
 - stride_w $\in [1, 8]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.21 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.22 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.23 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{ConvTranpose}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.24 ConvTranposeAdd

ConvTranpose 与 Add 融合计算, $\text{Output} = \text{Add}(\text{ConvTranpose}(\text{Input0}), \text{Input1})$

说明及规格限制同[ConvTranspose](#)

3.4.25 ConvTranposeSigmoid

ConvTranpose 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.26 ConvTranposeTanh

ConvTranpose 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.27 ConvTranposeSoftplus

ConvTranpose 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.28 ConvTranposeHardSigmoid

ConvTranpose 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{ConvTranpose}(\text{Input}))$

3.4.29 ConvTranposeHardSwish

ConvTranpose 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.30 ConvTranposeElu

ConvTranpose 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.31 ConvTranposeSwish

ConvTranpose 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.32 ConvTranposeMish

ConvTranpose 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.4.33 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.4.34 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`: 维度右对齐; 两个对应维度必须具有相同的值, 或者其中一个等于 1

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - shape: **T2**
 - **shape** : [batch, channel, height, width]
-

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]→[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

3.4.35 Hardmax**输出列表**

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $channel * height * width \leq 8192$
 - * $axis=1, height * width < 16384$ 且 $height * width$ 等于 1 或大于 8;
 - * $axis=3/-1, channel * height < 16384$, 且 $channel * height$ 等于 1 或大于 8, 且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 hardmax 的轴: 1,3/-1, 即 channel 和 width 方向

其他支持

- 多核联合: 尚不支持

3.4.36 LSTM

长短期记忆网络 (LSTM, Long Short-Term Memory)

LSTM 扩展以及变体命名为 exLSTM 算子, 参数项中指明

(extern) 的项为 exLSTM 独有的参数项。

计算公式如下:

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C_{(t-1)} + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- X: **T1**

- **shape**: [seq_length, batch_size, input_size]

- * *seqlength* $\in (0, 8192]$

- * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数

- * *inputsize* $\in (0, 8192]$

- 量化支持

- * per-layer

- W: **T1**

- **shape**: [num_directions, 4*hidden_size, input_size]

- * *inputsize* $\in (0, 8192]$

- * *hiddensize* $\in (0, 8192]$

- * num directions: 1 or 2

- 量化支持

- * per-layer

- * per-channel

- R: **T1**

- **shape**: [num_directions, 4*hidden_size, hidden_size]

- * *hiddensize* $\in (0, 8192]$

- * num directions: 1 or 2

- 量化支持

- * per-layer

- * per-channel

- B: **T2**

- **shape** : [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
- 量化支持
 - * per-layer
 - * per-channel
- sequence_lens : **T3** (optional)
 - **shape** : [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- initial_c : **T3** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y : **T1**
 - **shape** : [seq_length, num_directions, batch_size, hidden_size]
 - * *seqlength* $\in (0, 8192]$
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

- Y_c: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - **量化支持**
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向
 - bidirectional: 指定 LSTM 的运算方向为双向
- sequence_size (extern): int
 - 指定 LSTM 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - LSTM 单元中的 hiddensize, 无限制, 建议 8 对齐
- proj_size (extern): int
 - projection 时的 proj_size, *projsize* $\in [0, hiddensize]$
 - 目前限定 0, 即尚不支持 projection 功能
- input_forget (extern): int
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0, 即尚不支持
- has_projection (extern): int
 - projection 变种: 1(T) or 0(F) 目前限定 0, 即尚不支持
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size, 1, 1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size, 1, 1]

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int32(Tensor)
- **T3**: int(Scalar)

其他支持

- 多核联合: 暂不支持

3.4.37 Max

计算输入张量的元素级最大值

输入列表

- A: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W), B(N,C,H,W))$, 即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W), B(C,1,1))$, 即 C 维度做 broadcasting
 - $OP(A(N,C,H,W), B(scalar))$, 即以单个标量做 broadcasting
- B: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W), B(N,C,H,W))$, 即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W), B(C,1,1))$, 即 C 维度做 broadcasting
 - $OP(A(N,C,H,W), B(scalar))$, 即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.4.38 Min

计算输入张量的元素级最小值

输入列表

- A: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W),B(N,C,H,W))$ ，即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting
- B: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - $OP(A(N,C,H,W),B(N,C,H,W))$ ，即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- **多核联合**: 暂不支持

3.4.39 Mul

执行元素级二进制乘法 ($C = A * B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.4.40 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul
-

其他支持

- 多核联合: 支持
-

3.4.41 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape** : [1]
 - **量化支持**
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
 - **T2**: int64(Tensor)
 - **T3**: float(Scalar), int8(Scalar), float16(Scalar)
-

属性列表

- mode : **string**(默认值 constant)
 - **支持模式**: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$
-

其他支持

- **多核联合**: 尚不支持

3.4.42 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 8192]$
 - * $width \in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$
- storage_order :int64[]
 - 优先储存方式: 0

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.4.43 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.4.44 AveragePool

AveragePool 使用输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用平均池化。平均池化包括根据内核大小计算输入张量子集的所有值的平均值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 8192]$
 - * $width \in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.4.45 GlobalAveragePool

GlobalAveragePool 使用输入张量 X 并对同一通道中的值应用平均池化。这相当于 AveragePool，其 kernel_shape 大小等于输入张量的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.4.46 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- Y: **T2**
 - **shape**: [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- Z: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)

其他支持

- 多核联合: 尚不支持

3.4.47 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持

- * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.4.48 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.4.49 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel
 - 广播支持
-

- * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.4.50 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.4.51 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- shape : int
 - 指定输出张量的形状

其他支持

- **多核联合**: 尚不支持

3.4.52 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- roi (optional) : **T2**
 - 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

3.4.53 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴, 支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 height * width == 1 && channel % subc == 0 && starts[1] % subc == 0 && ends[1] % subc == 0 && steps[1] <= subc 时, steps 可以不为 1, 否则只有 steps == [1,1,1,1] 时, 才可通过 NPU 运行 Slice OP

3.4.54 Softmax

该运算符计算给定输入的归一化指数值:

$$\text{Softmax}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.4.55 exSoftmax13

该运算符计算给定输入的归一化指数值：

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
- 受限于 tranpose 的规格限制

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.4.56 exSoftmaxMask

该运算符计算给定输入的归一化指数值：

$$\text{input}(x, \text{mast}_t, \text{mask_value}) = \begin{cases} x + (\text{mast}_t - 1) * \text{inf} & \text{if } \text{mask_value} = 0 \\ x + \text{mast}_t * (-\text{inf}) & \text{if } \text{mask_value} = 1 \end{cases}$$

$$\text{exSoftmaxMask}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - mask (optional) : **T1**
 - **shape** : [1, channel, height, width]
- 受限于 tranpose 的规格限制

- * $channel \in (0, 8192]$
- * $width$:
 - axis=1, 无限制;
 - axis=3/-1, $width \in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向
- mask_value: int64
 - 需要 mask 的值: 0,1

3.4.57 Split

将向量沿着 axis 方向分成 num_outputs 个向量, split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output : **T**
 - 一或多个输出向量, 由 num_outputs 属性决定
 - **shape** : 根据切分情况决定
 - **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis :int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs :int
- split : int(Tensor)

3.4.58 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播
-

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.4.59 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats: **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

3.4.60 Transpose

对输入张量进行转置

输入列表

- X: **T1**
 - **shape**: [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

输出列表

- Y: **T1, T2**
 - **shape**: [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

属性列表

- perm 转置的轴顺序:
不同 perm 参数限制如下: (其中 p=8192, q=2048)
 - perm=[0,1,2,3]
 - * 无限制
 - perm=[0,1,3,2]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
 - perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 16$
 - * **T2**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
 - perm=[0,2,1,3]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - perm=[0,3,1,2]

- * **T1, T2:** $h1 \cdot w1 < p$, $w1 < 8192$, $h1 \cdot c1 < p$
- perm=[0,3,2,1]
 - * **T1:** $h1 \% 16 = 0$, $h1 \cdot w1 < 16 \cdot p$, $h1 \cdot w1 < 2 \cdot q$, $c1 < p \cdot 16$
 - * **T2:** $h1 \% 8 = 0$, $h1 \cdot w1 < 8 \cdot p$, $h1 \cdot w1 < 2 \cdot q$, $c1 < p \cdot 8$
- perm=[1,0,2,3]
 - * **T1:** $h1 \cdot w1 < 16 \cdot p$, $n1 < 8192$, $h1 \cdot w1 < p$
 - * **T2:** $h1 \cdot w1 < 8 \cdot p$, $n1 < 8192$, $h1 \cdot w1 < p$
- perm=[1,0,3,2]
 - * **T1:** $h1 \% 16 = 0$, $c1 \cdot h1 \cdot w1 < 16 \cdot p$, $h1 \cdot w1 < 2 \cdot q$
 - * **T2:** $h1 \% 8 = 0$, $c1 \cdot h1 \cdot w1 < 8 \cdot p$, $h1 \cdot w1 < 2 \cdot q$
- perm=[1,2,0,3]
 - * **T1:** $n1 \cdot w1 < p \cdot 16$, $c1 \cdot h1 \cdot w1 < p$, $w1 \% 16 = 0$
 - * **T2:** $n1 \cdot w1 < p \cdot 8$, $c1 \cdot h1 \cdot w1 < p$, $w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1:** $h1 \cdot w1 < p$, $n1 < p \cdot 16$, $c1 \cdot h1 \cdot w1 < p \cdot 16$, $n1 \cdot w1 < p \cdot 16$
 - * **T2:** $h1 \cdot w1 < p$, $n1 < p \cdot 8$, $c1 \cdot h1 \cdot w1 < p \cdot 8$, $n1 \cdot w1 < p \cdot 8$
- perm=[1,3,0,2]
 - * **T1:** $c1 \cdot w1 < 8192$, $n1 \cdot h1 < p$, $w1 \% 16 = 0$
 - * **T2:** $c1 \cdot w1 < 8192$, $n1 \cdot h1 < p$, $w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot h1 < p$, $w1 < 8192$, $w1 \% 16 = 0$
 - * **T2:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot h1 < p$, $w1 < 8192$, $w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1, T2:** $c1 < 8192$, $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot w1 < p$
- perm=[2,1,3,0]
 - * **T1:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot w1 < p \cdot 16$, $c1 < 8192$, $n1 \cdot w1 \% 4 = 0$
 - * **T2:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot w1 < p \cdot 8$, $c1 < 8192$, $n1 \cdot w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1, T2:** $c1 < 8192$, $c1 \cdot h1 \cdot w1 < p$
- perm=[2,0,1,3]
 - * **T1:** $n1 \cdot c1 \cdot h1 \cdot w1 < p$, $h1 \cdot w1 \% 16 = 0$
 - * **T2:** $n1 \cdot c1 \cdot h1 \cdot w1 < p$, $h1 \cdot w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot c1 < p$, $h1 \cdot w1 < 8192$, $h1 \cdot w1 \% 16 = 0$
 - * **T2:** $c1 \cdot h1 \cdot w1 < p$, $n1 \cdot c1 < p$, $h1 \cdot w1 < 8192$, $h1 \cdot w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1, T2:** $h1 \cdot w1 < 8192$, $n1 \cdot c1 < p$
- perm=[3,0,2,1]
 - * **T1:** $h1 \cdot w1 < p$, $n1 \cdot c1 < 8192$, $c1 \cdot h1 < p$, $n1 \cdot w1 < 8192$, $n1 \cdot w1 \% 16 = 0$

- * **T2**: $h1*w1 < p$, $n1*c1 < 8192$, $c1*h1 < p$, $n1*w1 < 8192$, $n1*w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1*w1 < p$, $n1*c1 < 8192$, $c1*h1 < p$, $n1*w1 < 8192$, $n1*h1*w1 \% 16 = 0$
 - * **T2**: $h1*w1 < p$, $n1*c1 < 8192$, $c1*h1 < p$, $n1*w1 < 8192$, $n1*h1*w1 \% 8 = 0$
- perm=[3,2,0,1]
 - * **T1, T2**: $h1*w1 < p$, $n1*c1 < 8192$, $n1*c1*w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1*h1*w1 < p$, $h1*w1 < 8192$, $2*h1*w1 < q$, $n1*c1 < p*16$, $h1 \% 16 = 0$
 - * **T2**: $c1*h1*w1 < p$, $h1*w1 < 8192$, $2*h1*w1 < q$, $n1*c1 < p*8$, $h1 \% 8 = 0$
- perm=[3,1,0,2]
 - * **T1, T2**: $h1*w1 < p$, $c1*w1 < 8192$, $n1*h1 < p$
- perm=[3,1,2,0]
 - * **T1, T2**: $h1*w1 < p$, $n1*c1 < 8192$, $n1*h1 < p$, $n1*c1*w1 < p$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.4.61 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.62 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.63 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.64 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.65 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.66 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.67 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.68 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.69 exGelu

给定输入张量, 函数 $y = x * \Phi(x)$ 按元素应用于输入张量, 得到输出张量, 当 $\Phi(x)$ 函数设置成 \tanh 时, $y = 0.5 * x * (1 + \text{Tanh}(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- **input: T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

3.4.70 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor : *T2
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

输出列表

- output : **T2**

- **shape** : [batch, channel, height, width]
- **量化支持**
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor), float16(Tensor)

3.4.71 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

3.4.72 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- A: **T1**
 - **shape** : [b, k, 1, n]
 - * $k \in (0, 8192]$
 - **量化支持**

- * per-layer
- B: **T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- C (optional): **T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- Y: **T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)

属性列表

- c_type: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

3.4.73 exSDPAttention

计算查询 (query)、键 (key) 和值 (value) 之间的缩放点积注意力 (SDPA)

输入列表

- query: **T1**
 - **shape**: [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- key: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- value: **T1**

- **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
- 量化支持
 - * per-layer
- mask (optional) : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 默认支持任意维度广播操作
- scale (optional) : **T3**

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.4.74 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。

原论文中输入特征被划分为多个不重叠的窗口，每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape，在高和宽方向上都进行划分。

pytorch 实现的计算方法如下：

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
```

(下页继续)

(续上页)

```

x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)

windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)

return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
        window_size (int): Window size
        H (int): Height of image
        W (int): Width of image

    Returns:
        x: (B, C, H, W)
    """
    B,C,_ = windows.shape
    x = windows.view(B, C, H // window_size, W // window_size, window_size, window_size)
    x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)
    return x

```

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- mode: **string**
 - **支持模式**: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;

- ★ reverse: 参考 window_reverse 实现;
 - ★ partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
 - window_sizes : **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
 - num_windows : **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。
-

其他支持

- 多核联合: 尚不支持

3.5 RK3576

3.5.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(const Tensor)
-

其他支持

- 多核联合: 支持

3.5.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add
-

其他支持

- 多核联合: 支持

3.5.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- epsilon: float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum: float
 - 训练时的滑动平均参数

其他支持

- 多核联合: 尚不支持

3.5.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.5.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- output: **T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 支持

3.5.6 Depthwise Convolution

深度可分离卷积

输入列表

- input_tensor: **T1**
 - **shape** : [batch, channel, height, width]
 - * width : 当 `dilated_kernel_h` > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- weight: **T1**
 - **shape** : [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 8]$
 - * kernel_w $\in [1, 8]$
 - 量化支持

- * per-layer
- * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 支持

3.5.7 ConvolutionRelu

Convolution 与 Relu 融合计算, Output = Relu(Conv(Input))

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.19 ConvolutionAddRelu

Convolution 与 Add 及 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Add}(\text{Conv}(\text{Input0}), \text{Input1}))$

说明及规格限制同[Convolution](#)

其他支持

- 多核联合: 支持

3.5.20 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 8]$
 - stride_w $\in [1, 8]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.5.21 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, Output = Relu(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.5.22 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, Output = Clip(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.5.23 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, Output = PRelu(ConvTranpose(Input)) 或者 Output = LeakyRelu(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.5.24 ConvTranposeAdd

ConvTranpose 与 Add 融合计算, Output = Add(ConvTranpose(Input0), Input1)

说明及规格限制同[ConvTranspose](#)

3.5.25 ConvTranposeSigmoid

ConvTranpose 与 Sigmoid 融合计算, Output = Sigmoid(ConvTranpose(Input))

说明及规格限制同[ConvTranspose](#)

3.5.26 ConvTranposeTanh

ConvTranpose 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.27 ConvTranposeSoftplus

ConvTranpose 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.28 ConvTranposeHardSigmoid

ConvTranpose 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.29 ConvTranposeHardSwish

ConvTranpose 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.30 ConvTranposeElu

ConvTranpose 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.31 ConvTranposeSwish

ConvTranpose 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.32 ConvTranposeMish

ConvTranpose 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.5.33 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

其他支持

- 多核联合: 不支持

3.5.34 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`: 维度右对齐; 两个对应维度必须具有相同的值, 或者其中一个等于 1

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持

- ★ per-layer
- shape: **T2**
 - **shape**: [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - ★ per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]->[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]->[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]->[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

其他支持

- 多核联合: 不支持

3.5.35 GRU

门控循环单元 (GRU, Gated Recurrent Unit)

GRU 扩展以及变体命名为 exGRU 算子, 参数项中指明

(extern) 的项为 exGRU 独有的参数项。

计算公式如下:

Equations (Default: f=Sigmoid, g=Tanh):

$$r_t = f(x_t \cdot W_r + h_{t-1} \cdot R_r + Wb_r + Rb_r)$$

$$z_t = f(x_t \cdot W_z + h_{t-1} \cdot R_z + Wb_z + Rb_z)$$

$$h_t = g(x_t \cdot W_h + (r_t \odot h_{t-1}) \cdot R_h + Rb_h + Wb_h), (WRB = 0)$$

$$h_t = g(x_t \cdot W_h + r_t \odot (h_{t-1} \cdot R_h + Rb_h)) + Wb_h, (WRB = 1)$$

$$H_t = (1 - z_t) \odot h_t + z_t \odot h_{t-1}$$

输入列表

- X: **T1**
 - **shape**: [seq_length, batch_size, input_size]
 - ★ $seqlength \in (0, 8192]$

- * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $inputsize \in (0, 8192]$
- 量化支持
 - * per-layer
- W: **T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * $inputsize \in (0, 8192]$
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2**
 - **shape**: [num_directions, 8*hidden_size]
 - * $hiddensize \in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T3** (optional)
 - **shape**: [batch_size]
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]

- * $seqlength \in (0, 8192]$
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
- 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- linear_before_reset: int
 - LBR 变种的选择: 1(T) or 0(F)
- direction (extern): string
 - 指定 GRU 的运算方向
 - forward: 指定 GRU 的运算方向为前向
 - reverse: 指定 GRU 的运算方向为反向
 - bidirectional: 指定 GRU 的运算方向为双向
- sequence_size (extern): int
 - 指定 GRU 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - GRU 单元中的 hiddensize, 无限制, 建议 8 对齐
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size, 1, 1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size, 1, 1]

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)

- **T3**: float(Scalar)

其他支持

- 多核联合: 暂不支持

3.5.36 Gather

根据索引 Indices 获取输入 X 的指定 axis 维度的条目，并将它们拼接在一起。

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- Indices: **T2**
 - 要收集的元素的索引，秩 $rank = 1$

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int(Tensor)

属性列表

- axis: int(Tensor)
 - $axis \in \{0, 1, 2, 3\}$
 - 指定 Indices 获取输入的维度

其他支持

- 多核联合: 尚不支持

3.5.37 Hardmax

该 OP 计算给定输入的 hardmax 激活函数，实现参照 onnx 中的[Hardmax-13](#)

$$\text{Hardmax}(\text{input}, \text{axis}) = \begin{cases} 1 & \text{if } i = \arg \max(\text{input}, \text{axis}) \\ 0 & \text{otherwise} \end{cases}$$

输入列表

- input: **T**

- **shape** : [batch, channel, height, width]
 - * axis=1 时, $height * width \in [1, 16384]$;
 - * axis=3/-1 时, $channel * height \in [1, 16384]$, 且受限于 tranpose 的规格限制

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - 做 hardmax 的轴: $axis \in \{-1, 1, 3\}$ 即 channel 和 width 方向

3.5.38 LSTM

长短期记忆网络 (LSTM, Long Short-Term Memory)

LSTM 扩展以及变体命名为 exLSTM 算子, 参数项中指明

(extern) 的项为 exLSTM 独有的参数项。

计算公式如下:

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C(t-1) + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- **X: T1**
 - **shape** : [seq_length, batch_size, input_size]
 - * $seqlength \in (0, 8192]$
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $inputsize \in (0, 8192]$
 - 量化支持
 - * per-layer
- **W: T1**
 - **shape** : [num_directions, 4*hidden_size, input_size]
 - * $inputsize \in (0, 8192]$
 - * $hiddensize \in (0, 8192]$

- * num directions: 1 or 2
- 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2, T3**
 - **shape**: [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T4, T5** (optional)
 - **shape**: [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- initial_c: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y: **T1**
 - **shape**: [seq_length, num_directions, batch_size, hidden_size]
 - * *seqlength* $\in (0, 8192]$

- * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
- 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *hiddensize* $\in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向
 - bidirectional: 指定 LSTM 的运算方向为双向
- sequence_size (extern): int
 - 指定 LSTM 输入的 seqsize, 无限制, 建议 4 对齐
- hidden_size (extern): int
 - LSTM 单元中的 hiddensize, 无限制, 建议 8 对齐
- proj_size (extern): int
 - projection 时的 proj_size, $projsize \in [0, hiddensize]$
 - 目前限定 0, 即尚不支持 projection 功能
- input_forget (extern): int
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0, 即尚不支持
- has_projection (extern): int
 - projection 变种: 1(T) or 0(F) 目前限定 0, 即尚不支持
- input_layout (extern): int
 - 指定与对应输入 shape 含义一致的 layout

- 要求填写指定的 layout，同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
- snc: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
- (sn)c: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size,1,1]
- output_layout (extern): int
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout，同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - sbnc: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - (sn)c: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size,1,1]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int32(Tensor)
- **T4**: float(Scalar)
- **T5**: int(Scalar)

其他支持

- **多核联合**: 暂不支持

3.5.39 Max

计算输入张量的元素级最大值

输入列表

- **A: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W))，即两个维度相同的 tensor 进行操作
 - OP(A(N,C,H,W),B(C,1,1))，即 C 维度做 broadcasting
 - OP(A(N,C,H,W),B(scalar))，即以单个标量做 broadcasting
- **B: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 支持两个 tensor 的广播操作，以 ONNX 默认排列 NCHW 做说明:
 - OP(A(N,C,H,W),B(N,C,H,W))，即两个维度相同的 tensor 进行操作

- $OP(A(N,C,H,W),B(C,1,1))$ ，即 C 维度做 broadcasting
- $OP(A(N,C,H,W),B(\text{scalar}))$ ，即以单个标量做 broadcasting

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.5.40 Mul

执行元素级二进制乘法 ($C = A * B$)，支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

- **T2**: float(const Tensor)

其他支持

- 多核联合: 支持

3.5.41 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul

其他支持

- 多核联合: 支持

3.5.42 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape** : [1]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)
- **T3**: float(Scalar), int8(Scalar), float16(Scalar)

属性列表

- mode : **string**(默认值 constant)
 - 支持模式: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$

其他支持

- 多核联合: 尚不支持

3.5.43 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * channel $\in (0, 8192]$
 - * height $\in (0, 8192]$
 - * width $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$

- `count_include_pad`: `int64[]`
 - `count_include_pad` 是否包含 `pad` 数值进行计算: 1
- `pad`: `int64(pad_top, pad_left, pad_bottom, pad_right)`
 - `pad_top` $\in [0, 7]$
 - `pad_left` $\in [0, 7]$
 - `pad_bottom` $\in [0, 7]$
 - `pad_right` $\in [0, 7]$
- `storage_order`: `int64[]`
 - 优先储存方式: 0

数据类型约束

- **T1**: `int8(Tensor)`, `float16(Tensor)`
- **T2**: `float(Tensor)`
- **T3**: `float(Scalar)`

3.5.44 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 `kernel_shape` 大小等于 `input` 的空间维度。

输入列表

- `input`: **T1**
 - **shape**: `[batch, channel, height, width]`
 - * `channel` $\in (0, 8192]$
 - * `height` $\in (0, 8192]$
 - * `width` $\in (0, 8192]$

输出列表

- `output`: **T1**
 - **shape**: `[batch, channel, height, width]`

属性列表

- `kernel_shape`: `int64(kernel_h, kernel_w)`
 - `kernel_h` $\in (0, 7]$
 - `kernel_w` $\in (0, 7]$
- `auto_pad`: `string`
 - `pad` 的方式: 仅支持 NOTSET
- `ceil_mode`: `int64`
 - 使用 `ceil` 或 `floor` 的方式计算输出的 `shape`: 不支持
- `strides`: `int64(strides_h, strides_w)`
 - `strides_h` $\in (0, 8]$
 - `strides_w` $\in (0, 8]$
- `count_include_pad`: `int64[]`

- count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.5.45 AveragePool

AveragePool 使用输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用平均池化。平均池化包括根据内核大小计算输入张量子集的所有值的平均值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (0, 8192]$
 - * *width* $\in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides: int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1

- **pad** : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.5.46 GlobalAveragePool

GlobalAveragePool 使用输入张量 X 并对同一通道中的值应用平均池化。这相当于 AveragePool，其 kernel_shape 大小等于输入张量的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *height* $\in (1, 7]$
 - * *width* $\in (1, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- **pad** : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$

- `pad_bottom` $\in [0, 7]$
- `pad_right` $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.5.47 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- **Y: T2**
 - **shape**: [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- **Z: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.48 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持

- * per-layer
- 广播支持
- * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.5.49 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.5.50 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.5.51 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
-

- * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.5.52 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- shape: int
 - 指定输出张量的形状

其他支持

- 多核联合: 尚不支持

3.5.53 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - roi (optional): **T2**
-

- 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

其他支持

- 多核联合: 不支持

3.5.54 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴，支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 $height * width == 1 \ \&\& \ channel \% subc == 0 \ \&\& \ starts[1] \% subc == 0 \ \&\& \ ends[1] \% subc == 0 \ \&\& \ steps[1] \leq subc$ 时, steps 可以不为 1，否则只有 steps == [1,1,1,1] 时，才可通过 NPU 运行 Slice OP

3.5.55 Softmax

该运算符计算给定输入的归一化指数值：

$$\text{Softmax}(input, axis) = \frac{\exp(input - \max(input, axis))}{\sum_{axis} \exp(input - \max(input, axis))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $axis=1, height * width \in (0, 16383]$

* $\text{axis}=3/-1, \text{channel} * \text{height} \in (0, 8192]$

且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

其他支持

- 多核联合: 尚不支持

3.5.56 exSoftmax13

该运算符计算给定输入的归一化指数值:

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * $\text{axis}=1, \text{height} * \text{width} \in (0, 16383]$
 - * $\text{axis}=3/-1, \text{channel} * \text{height} \in (0, 8192]$

且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

其他支持

- 多核联合: 尚不支持

3.5.57 exSoftmaxMask

该运算符计算给定输入的归一化指数值：

$$input(x, mast_t, mask_value) = \begin{cases} x + (mast_t - 1) * inf & \text{if } mask_value = 0 \\ x + mast_t * (-inf) & \\ \text{elif } mask_value = 1 \end{cases}$$

$$exSoftmaxMask(input, axis) = \frac{\exp(input - \max(input, axis))}{\sum_{axis} \exp(input - \max(input, axis))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * axis=1,height * width ∈ (0, 16383]
 - * axis=3/-1,channel * height ∈ (0, 8192]

且受限于 tranpose 的规格限制

- mask (optional): **T1**
 - **shape**: [1, channel, height, width]
 - * channel ∈ (0, 8192]
 - * width:
 - axis=1, 无限制;
 - axis=3/-1,width ∈ (0, 8192]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向
- mask_value: int64
 - 需要 mask 的值: 0,1

其他支持

- 多核联合: 尚不支持

3.5.58 Split

将向量沿着 axis 方向分成 num_outputs 个向量，split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T**
 - 一或多个输出向量，由 num_outputs 属性决定
 - **shape**: 根据切分情况决定
 - **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor), float16(Tensor)

属性列表

- axis: int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs: int
- split: int(Tensor)

3.5.59 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播
-

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * [b, c, 1, 1]
 - * [scalar]
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**

- * [b, c, 1, 1]
- * [scalar]

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: float(const Tensor)
-

其他支持

- 多核联合: 不支持

3.5.60 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats: **T2**
 - **shape**: [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
 - **T2**: int64(Tensor)
-

其他支持

- 多核联合: 不支持

3.5.61 Transpose

对输入张量进行转置

输入列表

- X: **T1**
 - **shape**: [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

输出列表

- Y: **T1, T2**
 - **shape**: [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

属性列表

- perm 转置的轴顺序:
不同 perm 参数限制如下: (其中 p=268435456, q=65536)
 - perm=[0,1,2,3]
 - * 无限制
 - perm=[0,1,3,2]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
 - perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 16$
 - * **T2**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
 - perm=[0,2,1,3]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - perm=[0,3,1,2]
 - * **T1, T2**: $h1 * w1 < p, w1 < 8192, h1 * c1 < p$
 - perm=[0,3,2,1]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 < p * 16$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 < p * 8$
 - perm=[1,0,2,3]
 - * **T1**: $h1 * w1 < 16 * p, n1 < 8192, h1 * w1 < p$
 - * **T2**: $h1 * w1 < 8 * p, n1 < 8192, h1 * w1 < p$
 - perm=[1,0,3,2]

- * **T1**: $h1 \% 16 = 0, c1 * h1 * w1 < 16 * p, h1 * w1 < 2 * q$
- * **T2**: $h1 \% 8 = 0, c1 * h1 * w1 < 8 * p, h1 * w1 < 2 * q$
- perm=[1,2,0,3]
 - * **T1**: $n1 * w1 < p * 16, c1 * h1 * w1 < p, w1 \% 16 = 0$
 - * **T2**: $n1 * w1 < p * 8, c1 * h1 * w1 < p, w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1**: $h1 * w1 < p, n1 < p * 16, c1 * h1 * w1 < p * 16, n1 * w1 < p * 16$
 - * **T2**: $h1 * w1 < p, n1 < p * 8, c1 * h1 * w1 < p * 8, n1 * w1 < p * 8$
- perm=[1,3,0,2]
 - * **T1**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 16 = 0$
 - * **T2**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p, n1 * w1 < p$
- perm=[2,1,3,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * w1 < p * 16, c1 < 8192, n1 * w1 \% 4 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * w1 < p * 8, c1 < 8192, n1 * w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p$
- perm=[2,0,1,3]
 - * **T1**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 16 = 0$
 - * **T2**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1, T2**: $h1 * w1 < 8192, n1 * c1 < p$
- perm=[3,0,2,1]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 8 = 0$
- perm=[3,2,0,1]
 - * **T1, T2**: $h1 * w1 < p, n1 * c1 < 8192, n1 * c1 * w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1 * h1 * w1 < p, h1 * w1 < 8192, 2 * h1 * w1 < q, n1 * c1 < p * 16, h1 \% 16 = 0$

- **T2**: $c1 \cdot h1 \cdot w1 < p$, $h1 \cdot w1 < 8192$, $2 \cdot h1 \cdot w1 < q$, $n1 \cdot c1 < p \cdot 8$, $h1 \% 8 = 0$
- perm=[3,1,0,2]
- **T1, T2**: $h1 \cdot w1 < p$, $c1 \cdot w1 < 8192$, $n1 \cdot h1 < p$
- perm=[3,1,2,0]
- **T1, T2**: $h1 \cdot w1 < p$, $n1 \cdot c1 < 8192$, $n1 \cdot h1 < p$, $n1 \cdot c1 \cdot w1 < p$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.5.62 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.63 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**

- **shape** : [batch, channel, height, width]
- 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.64 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.65 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.66 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.67 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.68 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- **多核联合**: 尚不支持

3.5.69 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- **多核联合**: 尚不支持

3.5.70 exGelu

给定输入张量，函数 $y = x * \Phi(x)$ 按元素应用于输入张量，得到输出张量，当 $\Phi(x)$ 函数设置成 \tanh 时， $y = 0.5 * x * (1 + \tanh(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.5.71 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor: **T2**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor: **T2**

- **shape** : [batch, channel, height, width]
- **量化支持**
 - * per-layer
- **广播支持**
 - * 支持任意维度广播操作
- **广播约束**
 - * 广播约束同Expand算子

输出列表

- output: **T2**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor), float16(Tensor)

3.5.72 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

其他支持

- **多核联合**: 尚不支持

3.5.73 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- **A: T1**
 - **shape**: [b, k, 1, n]
 - * $k \in (0, 8192]$
 - 量化支持
 - * per-layer
- **B: T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- **C (optional): T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- **Y: T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)

属性列表

- **c_type**: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

其他支持

- 多核联合: 不支持

3.5.74 exNorm

对输入数据进行各种归一化操作，目前支持 LayerNormlization、RMSNormlization 和 MeanVarianceNormalization 三种：

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot W + B$$

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\text{RMS}^2 + \epsilon}} \cdot W + B$$

$$\text{MeanVarianceNorm}(x) = \frac{x - \mu}{\sqrt{\text{Variance} + \epsilon}} \cdot W + B$$

其中：

x 是输入向量。

μ 和 σ^2 分别是输入向量的均值和方差：

$$\mu = \frac{1}{H} \sum_{i=1}^H x_i$$

$$\sigma^2 = \frac{1}{H} \sum_{i=1}^H (x_i - \mu)^2$$

RMS 是输入向量均方根：

$$\text{RMS} = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i)^2}$$

Variance 是输入向量的等效方差：

$$\text{Variance} = \frac{1}{H} \sum_{i=1}^H (x_i)^2 - \mu^2$$

ϵ 是一个很小的数，用于防止除零操作。

W 和 B 是可选择的参数。

输入列表

- X : **T1**
 - **shape**: [batch, channel, height, width]
- W (optional): **T1**
 - **shape**: [batch, channel, height, width]
- B (optional): **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y : **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- epsilon : **float**(默认值 1e-05)
 - 用来防止除 0 的 epsilon 数值
- mode : **string**
 - 支持 layer、rms 和 mean_var 三种 Normalization 模式可选
- axes : **int64[]**(默认值 [1])
 - 归一化轴, 目前只支持 [1] 和 [1,2,3]

其他支持

- 多核联合: 尚不支持

3.5.75 exSDPAttention

计算查询 (query)、键 (key) 和值 (value) 之间的缩放点积注意力 (SDPA)

输入列表

- query : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- key : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- value : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - 量化支持
 - * per-layer
- mask (optional) : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 默认支持任意维度广播操作
 - * 开启 FA 支持后, 只支持 [b,c,1,1], [1,c,1,1] 和 [1,c,h,w] 三种规格
- scale (optional) : **T3**

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

其他支持

- **多核联合**: 支持
- **FlashAttentionV2(FA)** : 支持
 - 基于 <https://arxiv.org/abs/2307.08691> 实现, 通过高速缓存内循环实现加速以及减少宽带使用, 但是会导致模型增大, 请根据具体场景和模型选择是否开启使用
 - RKNPU 驱动版本 >= 0.9.7
 - RKNPU Runtime 库 (librknnrt.so) 版本 >= 2.1.0
 - 支持数据类型: float16

3.5.76 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。

原论文中输入特征被划分为多个不重叠的窗口, 每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape, 在高和宽方向上都进行划分。

pytorch 实现的计算方法如下:

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
```

(下页继续)

(续上页)

```
windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
```

```
return windows
```

```
def window_reverse(windows, window_size, H, W):
```

```
    """
```

Args:

windows: (B, C, num_windows*num_windows, window_size*window_size)

window_size (int): Window size

H (int): Height of image

W (int): Width of image

Returns:

x: (B, C, H, W)

```
    """
```

```
B,C,_,_ = windows.shape
```

```
x = windows.view(B, C, H // window_size, W // window_size, window_size, window_size)
```

```
x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)
```

```
return x
```

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)

属性列表

- mode: **string**
 - **支持模式**: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;

- ★ `partition_num_first`: 与 `partition` 的区别在于, `partition_num_first` 的 H,W 按照 `num_windows` 进行划分, 而 `partition` 的 H,W 按照 `window_sizes` 进行划分。
 - `window_sizes`: **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
 - `num_windows`: **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。
-

其他支持

- **多核联合**: 尚不支持

3.6 RK2118

3.6.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.6.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add
-

3.6.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

属性列表

- epsilon : float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum : float
 - 训练时的滑动平均参数

3.6.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: float16(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.6.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - $stride_h \in [1, 7]$
 - $stride_w \in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - $pads_top \in [0, 15]$
 - $pads_left \in [0, 15]$
 - $pads_bottom \in [0, 15]$
 - $pads_right \in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - $dilations_h \in [1, 32]$

- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: float16(Tensor)

3.6.6 Depthwise Convolution

深度可分离卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 8]$
 - * kernel_w $\in [1, 8]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)

- dilations_h $\in [1, 32]$
- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: float16(Tensor)

3.6.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

3.6.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.6.19 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 8]$
 - stride_w $\in [1, 8]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group : 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: float16(Tensor)

3.6.20 Div

执行元素级二进制除法 ($C = A / B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.6.21 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`：维度右对齐；两个对应维度必须具有相同的值，或者其中一个等于 1

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- shape : **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]→[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

3.6.22 LSTM

长短期记忆网络（LSTM，Long Short-Term Memory）

LSTM 扩展以及变体命名为 exLSTM 算子，参数项中指明

（extern）的项为 exLSTM 独有的参数项。

计算公式如下：

Equations (Default: f=Sigmoid, g=Tanh, h=Tanh):

$$i_t = f(x_t \cdot W_i + h_{t-1} \cdot R_i + Wb_i + Rb_i)$$

$$f_t = f(x_t \cdot W_f + h_{t-1} \cdot R_f + Wb_f + Rb_f)$$

$$c_t = f(x_t \cdot W_c + h_{t-1} \cdot R_c + Wb_c + Rb_c)$$

$$o_t = f(x_t \cdot W_o + h_{t-1} \cdot R_o + Wb_o + Rb_o)$$

$$C_t = f_t \odot C(t-1) + i_t \odot c_t$$

$$h_t = o_t \odot h(C_t)$$

输入列表

- X: **T1**
 - **shape**: [seq_length, batch_size, input_size]
 - * *seqlength* $\in (0, 8192]$
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * *inputsize* $\in (0, 8192]$
 - 量化支持
 - * per-layer
- W: **T1**
 - **shape**: [num_directions, 4*hidden_size, input_size]
 - * *inputsize* $\in (0, 8192]$
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- R: **T1**
 - **shape**: [num_directions, 4*hidden_size, hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- B: **T2**
 - **shape**: [num_directions, 8*hidden_size]
 - * *hiddensize* $\in (0, 8192]$
 - * num directions: 1 or 2
 - 量化支持
 - * per-layer
 - * per-channel
- sequence_lens: **T3** (optional)
 - **shape**: [batch_size]
 - * *batchsize* $\in (0, 8192]$, 大于 1 时仅支持 4 的倍数
- initial_h: **T1** (optional)
 - **shape**: [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2

- * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
- 量化支持
 - * per-layer
- initial_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

输出列表

- Y : **T1**
 - **shape** : [seq_length, num_directions, batch_size, hidden_size]
 - * $seqlength \in (0, 8192]$
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_h : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer
- Y_c : **T1** (optional)
 - **shape** : [num_directions, batch_size, hidden_size]
 - * num directions: 1 or 2
 - * $batchsize \in (0, 8192]$, 大于 1 时仅支持 4 的倍数
 - * $hiddensize \in (0, 8192]$
 - 量化支持
 - * per-layer

属性列表

- direction (extern): string
 - 指定 LSTM 的运算方向
 - forward: 指定 LSTM 的运算方向为前向
 - reverse: 指定 LSTM 的运算方向为反向

- **bidirectional**: 指定 LSTM 的运算方向为双向
- **sequence_size (extern): int**
 - 指定 LSTM 输入的 seqsize, 无限制, 建议 4 对齐
- **hidden_size (extern): int**
 - LSTM 单元中的 hiddensize, 无限制, 建议 8 对齐
- **proj_size (extern): int**
 - projection 时的 proj_size, $proj_size \in [0, hiddensize]$
 - 目前限定 0, 即尚不支持 projection 功能
- **input_forget (extern): int**
 - cifg 变种的选择: 1(T) or 0(F) 目前限定 0, 即尚不支持
- **has_projection (extern): int**
 - projection 变种: 1(T) or 0(F) 目前限定 0, 即尚不支持
- **input_layout (extern): int**
 - 指定与对应输入 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size。
 - **snc**: 指定 layout 对应的输入 shape 为 [seqs, batches, input_size]
 - **(sn)c**: 指定 layout 对应的输入 shape 为 [seqs*batches, input_size,1,1]
- **output_layout (extern): int**
 - 指定与对应输出 shape 含义一致的 layout
 - 要求填写指定的 layout, 同时要求填写该 op 实际对应的 batch_size、sequence_size、hidden_size、directions。
 - **sbnc**: 指定 layout 对应的输出 shape 为 [seqs,directions,batches, hidden_size]
 - **(sn)c**: 指定 layout 对应的输出 shape 为 [seqsbatches, directionsinput_size,1,1]

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

其他支持

- **多核联合**: 暂不支持

3.6.23 Max

计算输入张量的元素级最大值

- **输入列表**
- **A: T1**
 - **shape**: [batch, channel, height, width]
 - **广播支持**
 - * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:

- $OP(A(N,C,H,W),B(N,C,H,W))$, 即两个维度相同的 tensor 进行操作
- $OP(A(N,C,H,W),B(C,1,1))$, 即 C 维度做 broadcasting
- $OP(A(N,C,H,W),B(\text{scalar}))$, 即以单个标量做 broadcasting

- B: **T1**

- **shape** : [batch, channel, height, width]

- 广播支持

- * 支持两个 tensor 的广播操作, 以 ONNX 默认排列 NCHW 做说明:

- $OP(A(N,C,H,W),B(N,C,H,W))$, 即两个维度相同的 tensor 进行操作
 - $OP(A(N,C,H,W),B(C,1,1))$, 即 C 维度做 broadcasting
 - $OP(A(N,C,H,W),B(\text{scalar}))$, 即以单个标量做 broadcasting

输出列表

- C: **T1**

- **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

其他支持

3.6.24 Mul

执行元素级二进制乘法 ($C = A * B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**

- **shape** : [batch, channel, height, width]

- 量化支持

- * per-layer

- 广播支持

- * 支持任意维度广播操作
 - * 广播约束同Expand算子

- B: **T1, T2**

- **shape** : [batch, channel, height, width]

- 量化支持

- * per-layer

- 广播支持

- * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.6.25 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul

其他支持

- 多核联合: 支持

3.6.26 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape** : [1]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持

- * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: int64(Tensor)
- **T3**: float(Scalar), int8(Scalar), float16(Scalar)

属性列表

- mode : **string**(默认值 constant)
 - 支持模式: constant, reflect
 - * constant 无限制
 - * reflect $\text{channel} \in (0, 8192]$, $\text{height} \in (0, 8192]$, $\text{width} \in (0, 8176]$

其他支持

- 多核联合: 尚不支持

3.6.27 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $\text{channel} \in (0, 8192]$
 - * $\text{height} \in (0, 8192]$
 - * $\text{width} \in (0, 8192]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - $\text{kernel_h} \in (0, 7]$
 - $\text{kernel_w} \in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)

- $\text{strides_h} \in (0, 8]$
- $\text{strides_w} \in (0, 8]$
- $\text{count_include_pad} : \text{int64}[]$
 - count_include_pad 是否包含 pad 数值进行计算: 1
- $\text{pad} : \text{int64}(\text{pad_top}, \text{pad_left}, \text{pad_bottom}, \text{pad_right})$
 - $\text{pad_top} \in [0, 7]$
 - $\text{pad_left} \in [0, 7]$
 - $\text{pad_bottom} \in [0, 7]$
 - $\text{pad_right} \in [0, 7]$

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.6.28 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $\text{channel} \in (0, 8192]$
 - * $\text{height} \in (0, 7]$
 - * $\text{width} \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- $\text{kernel_shape} : \text{int64}(\text{kernel_h}, \text{kernel_w})$
 - $\text{kernel_h} \in (0, 7]$
 - $\text{kernel_w} \in (0, 7]$
- $\text{auto_pad} : \text{string}$
 - pad 的方式: 仅支持 NOTSET
- $\text{ceil_mode} : \text{int64}$
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- $\text{strides} : \text{int64}(\text{strides_h}, \text{strides_w})$
 - $\text{strides_h} \in (0, 8]$
 - $\text{strides_w} \in (0, 8]$
- $\text{count_include_pad} : \text{int64}[]$

- count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.6.29 AveragePool

- 不支持 (工具端转换成卷积实现)

3.6.30 GlobalAveragePool

- 不支持 (工具端转换成卷积实现)

3.6.31 Pow

$$Z = X^Y$$

指数运算，采用输入数据（张量）和指数，并产生一个输出数据（张量）

输入列表

- X: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- Y: **T2**
 - **shape** : [1]
 - * 当前仅支持 0、1、2、3、0.5、-0.5

输出列表

- Z: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: int8(Tensor), int32(Tensor), int64(Tensor), float16(Tensor), float(Tensor)

其他支持

- **多核联合**: 尚不支持

3.6.32 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持

3.6.33 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **广播支持**
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)
-

其他支持

- **多核联合**: 暂不支持
-

3.6.34 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.6.35 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.6.36 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

属性列表

- shape : int
 - 指定输出张量的形状

其他支持

- 多核联合: 尚不支持

3.6.37 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- X : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- roi (optional) : **T2**
 - 目前暂不支持
- scales (optional) : **T2**
 - 沿每个维度的缩放比例数组
- sizes (optional) : **T3**
 - 输出张量的目标大小

输出列表

- Y : **T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- antialias : int (默认值为 0)
 - 目前不支持设置
- axes : int[]
 - 目前不支持设置
- coordinate_transformation_mode : strings (默认值为" half_pixel")
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- cubic_coeff_a : float (默认为-0.75)
 - 目前不支持设置
- exclude_outside : int (默认值为 0)
 - 目前不支持设置
- extrapolation_value : float (默认为 0)
 - 目前不支持设置
- keep_aspect_ratio_policy : strings (默认值为" stretch")
 - 目前不支持设置
- mode : strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- nearest_mode : strings (默认值为" round_prefer_floor")
 - 目前不支持设置

3.6.38 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T**: float16(Tensor)

属性列表

- starts: int(Tensor)
 - 切分的起始位置: 无限制
- ends: int(Tensor)
 - 切分的终止位置: 无限制
- axes: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴, 支持同时多轴选择
- steps: int(Tensor)
 - 选取切分对应轴的步长
 - 当 $height * width == 1 \ \&\& \ channel \% subc == 0 \ \&\& \ starts[1] \% subc == 0 \ \&\& \ ends[1] \% subc == 0 \ \&\& \ steps[1] \leq subc$ 时, steps 可以不为 1, 否则只有 steps == [1,1,1,1] 时, 才可通过 NPU 运行 Slice OP

3.6.39 Softmax

该运算符计算给定输入的归一化指数值:

$$\text{Softmax}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表 *

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * width :
 - axis=1, 无限制;
 - $axis=3/-1, width \in (0, 8192]$

且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.6.40 exSoftmax13

该运算符计算给定输入的归一化指数值：

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表 *

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - * *channel* $\in (0, 8192]$
 - * *width*:
 - axis=1, 无限制;
 - axis=3/-1, *width* $\in (0, 8192]$

且受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

axis: int64

- 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.6.41 exSoftmaxMask

- 尚不支持

3.6.42 Split

将向量沿着 axis 方向分成 num_outputs 个向量，split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor: **T**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T**
 - 一或多个输出向量，由 num_outputs 属性决定

- **shape** : 根据切分情况决定
- 量化支持
 - * per-layer

数据类型约束

- **T**: float16(Tensor)

属性列表

- axis :int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs :int
- split : int(Tensor)

3.6.43 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播
-

输入列表

- A : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.6.44 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats : **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

3.6.45 Transpose

对输入张量进行转置

输入列表

- X : **T1**
 - **shape** : [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

输出列表

- Y : **T1**
 - **shape** : [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - 量化支持
 - * per-layer

属性列表

- perm 转置的轴顺序:
 - 不同 perm 参数限制如下: (其中 p=268435456, q=65536)
 - perm=[0,1,2,3]
-

- * 无限制
- perm=[0,1,3,2]
 - * **T1**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
- perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
- perm=[0,2,1,3]
 - * **T1**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
- perm=[0,3,1,2]
 - * **T1**: $h1 * w1 < p, w1 < 8192, h1 * c1 < p$
- perm=[0,3,2,1]
 - * **T1**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 < p * 8$
- perm=[1,0,2,3]
 - * **T1**: $h1 * w1 < 8 * p, n1 < 8192, h1 * w1 < p$
- perm=[1,0,3,2]
 - * **T1**: $h1 \% 8 = 0, c1 * h1 * w1 < 8 * p, h1 * w1 < 2 * q$
- perm=[1,2,0,3]
 - * **T1**: $n1 * w1 < p * 8, c1 * h1 * w1 < p, w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1**: $h1 * w1 < p, n1 < p * 8, c1 * h1 * w1 < p * 8, n1 * w1 < p * 8$
- perm=[1,3,0,2]
 - * **T1**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1**: $c1 < 8192, c1 * h1 * w1 < p, n1 * w1 < p$
- perm=[2,1,3,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * w1 < p * 8, c1 < 8192, n1 * w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1**: $c1 < 8192, c1 * h1 * w1 < p$
- perm=[2,0,1,3]
 - * **T1**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1**: $h1 * w1 < 8192, n1 * c1 < p$
- perm=[3,0,2,1]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 8 = 0$

- perm=[3,2,0,1]
 - * **T1**: $h1*w1 < p$, $n1*c1 < 8192$, $n1*c1*w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1*h1*w1 < p$, $h1*w1 < 8192$, $2*h1*w1 < q$, $n1*c1 < p*8$, $h1\%8=0$
- perm=[3,1,0,2]
 - * **T1**: $h1*w1 < p$, $c1*w1 < 8192$, $n1*h1 < p$
- perm=[3,1,2,0]
 - * **T1**: $h1*w1 < p$, $n1*c1 < 8192$, $n1*h1 < p$, $n1*c1*w1 < p$

数据类型约束

- **T1**: int16(Tensor), float16(Tensor)

其他支持

- 多核联合: 暂不支持

3.6.46 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.47 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**
-

- **shape** : [batch, channel, height, width]
- 量化支持: per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.48 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- **X: T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.49 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **input: T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- **output: T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.50 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$, $\beta = 0.5$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.51 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.52 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.53 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.54 exGelu

给定输入张量, 函数 $y = x * \Phi(x)$ 按元素应用于输入张量, 得到输出张量, 当 $\Phi(x)$ 函数设置成 \tanh 时, $y = 0.5 * x * (1 + \tanh(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$

输入列表

- **input: T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)

3.6.55 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor : *T2
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

输出列表

- output : **T2**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: float16(Tensor)

3.6.56 exConvStreaming

为了减少语音模型的时延，把原始卷积的输入分成若干个时间帧片，将每个时间帧片作为输入，计算卷积，最后将计算结果缓存起来供后续时间帧使用，从而达到音频流式处理的目的。

输入列表

- input_tensor: **T1**
 - **shape** : [batch, channel, height, width]
 - * width : 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- weight: **T1**
 - **shape** : [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 7]$
 - stride_w $\in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$

- pads_right $\in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: float16(Tensor)

其他支持

- 多核联合: 尚不支持

3.6.57 exConvStreamingPRelu/LeakyRelu

exConvStreaming 与 PRelu/LeakyRelu 融合计算, Output = PRelu(exConvStreaming(Input)) 或者 Output = LeakyRelu(exConvStreaming(Input))

说明及规格限制同[exConvStreaming](#)

3.6.58 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: float16(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

3.6.59 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- **A: T1**
 - **shape**: [b, k, 1, n]
 - * $k \in (0, 8192]$
 - 量化支持
 - * per-layer
- **B: T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- **C (optional): T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- **Y: T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(Tensor)

属性列表

- **c_type**: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

3.6.60 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。
原论文中输入特征被划分为多个不重叠的窗口，每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape，在高和宽方向上都进行划分。
pytorch 实现的计算方法如下：

```
def window_partition(x, window_size):
    """
    Args:
        x: (B, C, H, W)
        window_size (int): window size

    Returns:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
    windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
    return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
        window_size (int): Window size
        H (int): Height of image
        W (int): Width of image

    Returns:
        x: (B, C, H, W)
    """
    B, C, _, _ = windows.shape
    x = windows.view(B, C, H // window_size, window_size, W // window_size, window_size)
    x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)
```

(下页继续)

(续上页)

```
return x
```

输入列表

- X : **T1**
 - **shape** : [batch, channel, height, width]

输出列表

- Y : **T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: float16(Tensor)

属性列表

- mode : **string**
 - **支持模式**: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;
 - * partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
- window_sizes : **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
- num_windows : **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。

其他支持

- **多核联合**: 尚不支持

3.7 RV1103B

3.7.1 Add

执行元素级二进制加法 ($C = A + B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.7.2 AddRelu

Add与Relu融合计算, $C = \text{Relu}(A + B)$

输入列表

- 同Add

输出列表

- 同Add

其他支持

- 多核联合: 支持

3.7.3 BatchNormalization

按照论文 <https://arxiv.org/abs/1502.03167> 中的描述对输入张量进行批量归一化计算

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

属性列表

- epsilon : float (默认值 = 1e-5)
 - 除以标准差时加上防止除 0 的实数
 - $\epsilon \in (0, \infty]$
- momentum : float
 - 训练时的滑动平均参数

3.7.4 Concat

将一系列向量在 axis 指定的方向上组合成一个向量，所有向量除 axis 指定的方向外大小必须相同

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor)

属性列表

- axis int64
 - $axis \in \{0, 1, 2, 3\}$

3.7.5 Convolution

卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当 $dilated_kernel_h > 1$ 时, $width < 16383$ 。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * $kernel_h \in [1, 31]$
 - * $kernel_w \in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - $stride_h \in [1, 7]$
 - $stride_w \in [1, 7]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)
 - $pads_top \in [0, 15]$
 - $pads_left \in [0, 15]$
 - $pads_bottom \in [0, 15]$
 - $pads_right \in [0, 15]$
- group: int
- dilations: int(dilations_h, dilations_w)
 - $dilations_h \in [1, 32]$

- `dilations_w` $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor)

3.7.6 Depthwise Convolution

深度可分离卷积

输入列表

- `input_tensor`: **T1**
 - **shape**: [batch, channel, height, width]
 - * `width`: 当`dilated_kernel_h` > 1 时, `width` < 16383。此外, 对首层输入 `width` 存在限制, 详见模型输入说明。
 - 量化支持
 - * per-layer
- `weight`: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * `kernel_h` $\in [1, 8]$
 - * `kernel_w` $\in [1, 8]$
 - 量化支持
 - * per-layer
 - * per-channel

输出列表

- `output`: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

属性列表

- `strides`: int(strides_h, strides_w)
 - `stride_h` $\in [1, 7]$
 - `stride_w` $\in [1, 7]$
- `pads`: int(pads_top, pads_left, pads_bottom, pads_right)
 - `pads_top` $\in [0, 15]$
 - `pads_left` $\in [0, 15]$
 - `pads_bottom` $\in [0, 15]$
 - `pads_right` $\in [0, 15]$
- `group`: int
- `dilations`: int(dilations_h, dilations_w)

- dilations_h $\in [1, 32]$
- dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor)

3.7.7 ConvolutionRelu

Convolution 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.8 ConvolutionClip

Convolution 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.9 ConvolutionPRelu/LeakyRelu

Convolution 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{Conv}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.10 ConvolutionAdd

Convolution 与 Add 融合计算, $\text{Output} = \text{Add}(\text{Conv}(\text{Input0}), \text{Input1})$

说明及规格限制同[Convolution](#)

3.7.11 ConvolutionSigmoid

Convolution 与 Sigmoid 融合计算, $\text{Output} = \text{Sigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.12 ConvolutionTanh

Convolution 与 Tanh 融合计算, $\text{Output} = \text{Tanh}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.13 ConvolutionSoftplus

Convolution 与 Softplus 融合计算, $\text{Output} = \text{Softplus}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.14 ConvolutionHardSigmoid

Convolution 与 HardSigmoid 融合计算, $\text{Output} = \text{HardSigmoid}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.15 ConvolutionHardSwish

Convolution 与 HardSwish 融合计算, $\text{Output} = \text{HardSwish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.16 ConvolutionElu

Convolution 与 Elu 融合计算, $\text{Output} = \text{Elu}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.17 ConvolutionSwish

Convolution 与 Swish 融合计算, $\text{Output} = \text{Swish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.18 ConvolutionMish

Convolution 与 Mish 融合计算, $\text{Output} = \text{Mish}(\text{Conv}(\text{Input}))$

说明及规格限制同[Convolution](#)

3.7.19 ConvTranspose

转置卷积

输入列表

- input_tensor: **T1**
 - **shape**: [batch, channel, height, width]
 - * width: 当dilated_kernel_h > 1 时, width < 16383。此外, 对首层输入 width 存在限制, 详见模型输入说明。
 - **量化支持**
 - * per-layer
- weight: **T1**
 - **shape**: [output_channel, input_channel, kernel_h, kernel_w]
 - * kernel_h $\in [1, 31]$
 - * kernel_w $\in [1, 31]$
 - **量化支持**
 - * per-layer
 - * per-channel

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

属性列表

- strides: int(strides_h, strides_w)
 - stride_h $\in [1, 8]$
 - stride_w $\in [1, 8]$
- pads: int(pads_top, pads_left, pads_bottom, pads_right)

设置 pad 时注意:

不支持 $\text{kernel_h} * \text{dilations_h} - \text{dilations_h} - \text{pads_top} < 0$

不支持 $\text{kernel_w} * \text{dilations_w} - \text{dilations_w} - \text{pads_left} < 0$

不支持 $\text{stride_h} * (\text{height} - 1) - \text{pads_top} + 1 < \text{output_h}$

不支持 $\text{stride_w} * (\text{width} - 1) - \text{pads_left} + 1 < \text{output_w}$

 - pads_top $\in [0, 15]$
 - pads_left $\in [0, 15]$
 - pads_bottom $\in [0, 15]$
 - pads_right $\in [0, 15]$
- group: int

- group: 仅支持 1, 当且仅当 num_input=num_output 时支持 num_output
- dilations: int(dilations_h, dilations_w)
 - dilations_h $\in [1, 32]$
 - dilations_w $\in [1, 32]$

数据类型约束

- **T1**: int8(Tensor)

3.7.20 ConvTranposeRelu

ConvTranpose 与 Relu 融合计算, $\text{Output} = \text{Relu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.7.21 ConvTranposeClip

ConvTranpose 与 Clip 融合计算, $\text{Output} = \text{Clip}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.7.22 ConvTranposePRelu/LeakyRelu

ConvTranpose 与 PRelu/LeakyRelu 融合计算, $\text{Output} = \text{PRelu}(\text{ConvTranpose}(\text{Input}))$ 或者 $\text{Output} = \text{LeakyRelu}(\text{ConvTranpose}(\text{Input}))$

说明及规格限制同[ConvTranspose](#)

3.7.23 Div

执行元素级二进制除法 ($C = A / B$), 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
 - **广播支持**
 - * [b, c, 1, 1]
 - * [scalar]
- B: **T1, T2**
 - **shape**: [batch, channel, height, width]
 - **量化支持**

- * per-layer
- 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: float16(Tensor)
- **T2**: float(const Tensor)

3.7.24 Expand

根据给定的 shape 和广播规则广播输入张量。广播规则类似于 `numpy.array(input) * numpy.ones(shape)`: 维度右对齐; 两个对应维度必须具有相同的值, 或者其中一个等于 1

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
- shape: **T2**
 - **shape**: [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int64(Tensor)

广播约束

- [b, c, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, 1, 1, w]→[b, c, h, w], $h \in [1, 8192]$
- [b, c, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, 1, h, 1]→[b, c, h, w], $w \in [1, 8192]$
- [b, c, 1, 1]→[b, c, h, w], $h \in [1, 8192], w \in [1, 8192]$

3.7.25 Mul

执行元素级二进制乘法 ($C = A * B$) , 支持多向 Numpy 样式的广播

输入列表

- A: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子
- B: **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.7.26 MulRelu

Mul与Relu融合计算, $C = \text{Relu}(A * B)$

输入列表

- 同Mul

输出列表

- 同Mul
-

其他支持

- 多核联合: 支持
-

3.7.27 Pad

给定一个包含要填充的数据的张量 (data), 包含轴的开始和结束填充数值的张量 (pads), 模式 (mode), 常量数值 (constant_value), 生成一个填充张量 (output)

输入列表

- data : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- pads : **T2**
 - **shape** : [batch_begin, channel_begin, height_begin, width_begin, batch_end, channel_end, height_end, width_end]
- constant_value (optional): **T3**
 - **shape** : [1]
 - **量化支持**
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int64(Tensor)
- **T3**: float(Scalar), int8(Scalar), float16(Scalar)

属性列表

- mode : **string**(默认值 constant)
 - **支持模式**: constant, reflect
 - * constant 无限制
 - * reflect channel $\in (0, 8192]$, height $\in (0, 8192]$, width $\in (0, 8176]$

其他支持

- **多核联合**: 尚不支持

3.7.28 MaxPool

MaxPool 消耗输入张量 X 并根据内核大小、步幅大小和 pad 长度在张量上应用最大池化。最大池化包括根据内核大小计算输入张量子集的所有值的最大值，并将数据下采样到输出张量 Y 中以进行进一步处理。输出空间形状的计算方式有所不同，具体取决于是否使用显式填充（在使用 pads 的情况下）或使用自动填充（在使用 auto_pad 的情况下）。仅使用显式填充

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.7.29 GlobalMaxPool

GlobalMaxPool 使用输入张量 X 并对同一通道中的值应用最大池化。这相当于 MaxPool 的 kernel_shape 大小等于 input 的空间维度。

输入列表

- input : **T1**
 - **shape** : [batch, channel, height, width]
 - * $channel \in (0, 8192]$
 - * $height \in (0, 7]$
 - * $width \in (0, 7]$

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]

属性列表

- kernel_shape: int64(kernel_h, kernel_w)
 - kernel_h $\in (0, 7]$
 - kernel_w $\in (0, 7]$
- auto_pad: string
 - pad 的方式: 仅支持 NOTSET
- ceil_mode: int64
 - 使用 ceil 或 floor 的方式计算输出的 shape : 不支持
- strides : int64(strides_h, strides_w)
 - strides_h $\in (0, 8]$
 - strides_w $\in (0, 8]$
- count_include_pad : int64[]
 - count_include_pad 是否包含 pad 数值进行计算: 1
- pad : int64(pad_top, pad_left, pad_bottom, pad_right)
 - pad_top $\in [0, 7]$
 - pad_left $\in [0, 7]$
 - pad_bottom $\in [0, 7]$
 - pad_right $\in [0, 7]$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)
- **T3**: float(Scalar)

3.7.30 AveragePool

- 不支持 (工具端转换成卷积实现)

3.7.31 GlobalAveragePool

- 不支持 (工具端转换成卷积实现)

3.7.32 Relu

将输入张量的负值设为零，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.7.33 Clip/ReLU6

将输入张量的值裁剪到 [0, 6] 范围内

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无
-

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.7.34 PRelu

将输入张量的负值按可学习参数缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer/per-channel

属性列表

- slope: **PRelu 系数**
 - 仅支持单个标量或 C 维度系数

数据类型约束

- **T1**: int8(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.7.35 LeakyRelu

将输入张量的负值按固定比例缩放，正值保持不变

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 无

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
-

其他支持

- 多核联合: 暂不支持

3.7.36 Reshape

在不改变输入数据和元素数量的情况下，返回一个具有指定形状的张量。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

属性列表

- shape: int
-

- 指定输出张量的形状

其他支持

- **多核联合**: 尚不支持

3.7.37 Resize

调整输入张量的大小。一般来说，它将输出张量中的每个值计算为输入张量中邻域（即采样位置）的加权平均值

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer
- **roi (optional): T2**
 - 目前暂不支持
- **scales (optional): T2**
 - 沿每个维度的缩放比例数组
- **sizes (optional): T3**
 - 输出张量的目标大小

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)
- **T3**: int64(Tensor)

属性列表

- **antialias**: int（默认值为 0）
 - 目前不支持设置
- **axes**: int[]
 - 目前不支持设置
- **coordinate_transformation_mode**: strings（默认值为" half_pixel"）
 - 目前仅支持 half_pixel, pytorch_half_pixel, align_corners 三种
- **cubic_coeff_a**: float（默认为-0.75）
 - 目前不支持设置

- `exclude_outside`: int (默认值为 0)
 - 目前不支持设置
- `extrapolation_value`: float (默认为 0)
 - 目前不支持设置
- `keep_aspect_ratio_policy`: strings (默认值为" stretch")
 - 目前不支持设置
- `mode`: strings (默认值为" nearest")
 - 目前仅支持 nearest 和 linear 两种可配置
- `nearest_mode`: strings (默认值为" round_prefer_floor")
 - 目前不支持设置

3.7.38 Slice

获得当前向量的一个切片，具体规则和numpy切片相同

输入列表

- `input_tensor`: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- `output`: **T**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T**: int8(Tensor)

属性列表

- `starts`: int(Tensor)
 - 切分的起始位置: 无限制
- `ends`: int(Tensor)
 - 切分的终止位置: 无限制
- `axes`: int(Tensor)
 - 选取切分的轴: 支持任意 0~3 轴，支持同时多轴选择
- `steps`: int(Tensor)
 - 选取切分对应轴的步长
 - 当 `height * width == 1 && channel % subc == 0 && starts[1] % subc == 0 && ends[1] % subc == 0 && steps[1] <= subc` 时, steps 可以不为 1，否则只有 `steps == [1,1,1,1]` 时，才可通过 NPU 运行 Slice OP

3.7.39 Softmax

该运算符计算给定输入的归一化指数值：

$$\text{Softmax}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
- 受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.7.40 exSoftmax13

该运算符计算给定输入的归一化指数值：

$$\text{exSoftmax13}(\text{input}, \text{axis}) = \frac{\exp(\text{input} - \max(\text{input}, \text{axis}))}{\sum_{\text{axis}} \exp(\text{input} - \max(\text{input}, \text{axis}))}$$

“axis” 属性表示 Softmax 执行的维度。输出张量具有相同的形状并包含相应输入的 Softmax 值。

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
- 受限于 tranpose 的规格限制

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor)

属性列表

- axis: int64
 - 做 softmax 的轴: 1,3, 即 channel 和 width 方向

3.7.41 exSoftmaxMask

- 尚不支持

3.7.42 Split

将向量沿着 axis 方向分成 num_outputs 个向量，split 用于指定切分后向量在 axis 方向上的大小

输入列表

- input_tensor : **T**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output : **T**
 - 一或多个输出向量，由 num_outputs 属性决定
 - **shape** : 根据切分情况决定
 - 量化支持
 - * per-layer

数据类型约束

- **T**: int8(Tensor)

属性列表

- axis :int
 - $axis \in \{0, 1, 2, 3\}$
- num_outputs :int
- split : int(Tensor)

3.7.43 Sub

- 执行元素级二进制减法 ($C = A - B$) , 支持多向 Numpy 样式的广播
-

输入列表

- A : **T1, T2**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * [b, c, 1, 1]
 - * [scalar]
- B : **T1, T2**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer
- 广播支持
 - * [b, c, 1, 1]
 - * [scalar]

输出列表

- C: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(const Tensor)

3.7.44 Tile

通过平铺给定的张量构造张量, 这与 Numpy 中的函数 tile 相同, 但没有广播

输入列表

- input: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
- repeats: **T2**
 - **shape** : [batch, channel, height, width]

输出列表

- output: **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor), float16(Tensor)
- **T2**: int64(Tensor)

3.7.45 Transpose

对输入张量进行转置

输入列表

- X: **T1**
 - **shape**: [n1, c1, h1, w1]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

输出列表

- Y: **T1, T2**
 - **shape**: [n2, c2, h2, w2]
 - * 详细约束规则见属性列表
 - **量化支持**
 - * per-layer

属性列表

- perm 转置的轴顺序:
不同 perm 参数限制如下: (其中 p=268435456, q=65536)
 - perm=[0,1,2,3]
 - * 无限制
 - perm=[0,1,3,2]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q$
 - perm=[0,2,3,1]
 - * **T1**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 16$
 - * **T2**: $c1 < 8192, h1 * w1 \% 8 = 0, w1 * c1 < p * 8$
 - perm=[0,2,1,3]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 * w1 < p$
 - perm=[0,3,1,2]
 - * **T1, T2**: $h1 * w1 < p, w1 < 8192, h1 * c1 < p$
 - perm=[0,3,2,1]
 - * **T1**: $h1 \% 16 = 0, h1 * w1 < 16 * p, h1 * w1 < 2 * q, c1 < p * 16$
 - * **T2**: $h1 \% 8 = 0, h1 * w1 < 8 * p, h1 * w1 < 2 * q, c1 < p * 8$
 - perm=[1,0,2,3]
 - * **T1**: $h1 * w1 < 16 * p, n1 < 8192, h1 * w1 < p$
 - * **T2**: $h1 * w1 < 8 * p, n1 < 8192, h1 * w1 < p$
 - perm=[1,0,3,2]

- * **T1**: $h1 \% 16 = 0, c1 * h1 * w1 < 16 * p, h1 * w1 < 2 * q$
- * **T2**: $h1 \% 8 = 0, c1 * h1 * w1 < 8 * p, h1 * w1 < 2 * q$
- perm=[1,2,0,3]
 - * **T1**: $n1 * w1 < p * 16, c1 * h1 * w1 < p, w1 \% 16 = 0$
 - * **T2**: $n1 * w1 < p * 8, c1 * h1 * w1 < p, w1 \% 8 = 0$
- perm=[1,2,3,0]
 - * **T1**: $h1 * w1 < p, n1 < p * 16, c1 * h1 * w1 < p * 16, n1 * w1 < p * 16$
 - * **T2**: $h1 * w1 < p, n1 < p * 8, c1 * h1 * w1 < p * 8, n1 * w1 < p * 8$
- perm=[1,3,0,2]
 - * **T1**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 16 = 0$
 - * **T2**: $c1 * w1 < 8192, n1 * h1 < p, w1 \% 8 = 0$
- perm=[1,3,2,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * h1 < p, w1 < 8192, w1 \% 8 = 0$
- perm=[2,1,0,3]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p, n1 * w1 < p$
- perm=[2,1,3,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * w1 < p * 16, c1 < 8192, n1 * w1 \% 4 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * w1 < p * 8, c1 < 8192, n1 * w1 \% 4 = 0$
- perm=[2,0,3,1]
 - * **T1, T2**: $c1 < 8192, c1 * h1 * w1 < p$
- perm=[2,0,1,3]
 - * **T1**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 16 = 0$
 - * **T2**: $n1 * c1 * h1 * w1 < p, h1 * w1 \% 8 = 0$
- perm=[2,3,1,0]
 - * **T1**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 16 = 0$
 - * **T2**: $c1 * h1 * w1 < p, n1 * c1 < p, h1 * w1 < 8192, h1 * w1 \% 8 = 0$
- perm=[2,3,0,1]
 - * **T1, T2**: $h1 * w1 < 8192, n1 * c1 < p$
- perm=[3,0,2,1]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * w1 \% 8 = 0$
- perm=[3,0,1,2]
 - * **T1**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 16 = 0$
 - * **T2**: $h1 * w1 < p, n1 * c1 < 8192, c1 * h1 < p, n1 * w1 < 8192, n1 * h1 * w1 \% 8 = 0$
- perm=[3,2,0,1]
 - * **T1, T2**: $h1 * w1 < p, n1 * c1 < 8192, n1 * c1 * w1 < p$
- perm=[3,2,1,0]
 - * **T1**: $c1 * h1 * w1 < p, h1 * w1 < 8192, 2 * h1 * w1 < q, n1 * c1 < p * 16, h1 \% 16 = 0$

- * **T2**: $c1 \cdot h1 \cdot w1 < p$, $h1 \cdot w1 < 8192$, $2 \cdot h1 \cdot w1 < q$, $n1 \cdot c1 < p \cdot 8$, $h1 \% 8 = 0$
- perm=[3,1,0,2]
 - * **T1, T2**: $h1 \cdot w1 < p$, $c1 \cdot w1 < 8192$, $n1 \cdot h1 < p$
- perm=[3,1,2,0]
 - * **T1, T2**: $h1 \cdot w1 < p$, $n1 \cdot c1 < 8192$, $n1 \cdot h1 < p$, $n1 \cdot c1 \cdot w1 < p$

数据类型约束

- **T1**: int8(Tensor)
- **T2**: int16(Tensor)

其他支持

- 多核联合: 暂不支持

3.7.46 Sigmoid

给定输入张量，函数 $y = 1 / (1 + \exp(-x))$ 按元素应用于输入张量，得到输出张量

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.47 Tanh

计算给定输入张量单元的双曲正切

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.48 Softplus

按元素应用 Softplus 函数, $y = \ln(\exp(x) + 1)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.49 HardSigmoid

给定输入张量, 函数 $y = \max(0, \min(1, \alpha * x + \beta))$ 按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1/6$, $\beta =$

0.5

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.50 HardSwish

按元素应用 HardSwish 函数, $y = x * \max(0, \min(1, \alpha * x + \beta)) = x * \text{HardSigmoid}(\alpha, \beta)(x)$, 这里 $\alpha = 1/6$,

$\beta = 0.5$

输入列表

- X: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Y: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.51 Elu

给定输入张量, 函数 $f(x) = \alpha * (\exp(x) - 1.)$ for $x < 0$, $f(x) = x$ for $x \geq 0$

按元素应用于输入张量, 得到输出张量, 这里 $\alpha = 1$

输入列表

- input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

输出列表

- output: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.52 exSwish

按元素应用 Sigmoid 线性单元函数, Swish 函数也称为 SiLU 函数, $y = x * \text{sigmoid}(x)$

输入列表

- **X: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

输出列表

- **Y: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持: per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.53 exMish

给定输入张量, 函数 $\text{mish}(x) = x * \tanh(\text{softplus}(x)) = x * \tanh(\ln(1 + e^{\{x\}}))$ 按元素应用于输入张量, 得到输出张量

输入列表

- **input: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

输出列表

- **output: T1**
 - **shape**: [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.54 exGelu

给定输入张量, 函数 $y = x * \Phi(x)$ 按元素应用于输入张量, 得到输出张量, 当 $\Phi(x)$ 函数设置成 \tanh 时, $y =$

$$0.5 * x * (1 + \tanh(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3)))$$

输入列表

- **input: T1**

- **shape** : [batch, channel, height, width]
- 量化支持
 - * per-layer

输出列表

- output : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)

3.7.55 Where

根据 mask_tensor 获取 x_tensor 或 y_tensor 的值，当 mask_tensor 位置为 **True** 时，取 x_tensor 对应位置的值，否则取 y_tensor 的值。

输入列表

- mask_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- x_tensor : *T2
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子
- y_tensor : **T1**
 - **shape** : [batch, channel, height, width]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - 广播约束
 - * 广播约束同Expand算子

输出列表

- output : **T2**

- **shape** : [batch, channel, height, width]
- **量化支持**
 - * per-layer

数据类型约束

- **T1**: bool(Tensor)
- **T2**: int8(Tensor)

3.7.56 exGlu

门控线性单元 (Gated Linear Unit) 函数, 其中 a 是输入矩阵的前一半, b 是后一半

$$GLU(a, b) = a \otimes \sigma(b)$$

输入列表

- Input: **T1**
 - **shape**: [batch, channel, height, width]
 - **量化支持**: per-layer

输出列表

- Output: **T1**
 - **shape**: [batch, channel / 2, height, width]
 - **量化支持**: per-layer

数据类型约束

- **T1**: int8(Tensor)

属性列表

- axis: int(默认值 = 1)
 - 分割输入的维度
 - 约束: 仅支持 1

3.7.57 exMatMul

矩阵乘法操作

$$Y = \begin{cases} A \cdot B + C, & \text{if } c_type = \text{"add"} \\ A \cdot B * C, & \text{if } c_type = \text{"mul"} \end{cases}$$

输入列表

- A: **T1**
 - **shape** : [b, k, 1, n]
 - * $k \in (0, 8192]$
 - **量化支持**

- * per-layer
- B: **T1**
 - **shape**: [b, k, 1, m]
 - 量化支持
 - * per-layer
- C (optional): **T1, T2**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer
 - 广播支持
 - * 支持任意维度广播操作
 - * 广播约束同Expand算子

输出列表

- Y: **T1**
 - **shape**: [b, n, 1, m]
 - 量化支持
 - * per-layer

数据类型约束

- **T1**: int8(Tensor)
- **T2**: float(Tensor)

属性列表

- c_type: strings (默认值 “add”)
 - 支持 add 和 mul 可选, 表示 C 是做加法还是乘法

3.7.58 exWindow

exWindow 来源于swin_transformer的 window_partition 和 window_reverse 操作。

原论文中输入特征被划分为多个不重叠的窗口，每个窗口的大小为 window_size × window_size。要求输入和输出特征都为 4 维 shape，在高和宽方向上都进行划分。

pytorch 实现的计算方法如下：

```
def window_partition(x, window_size):
```

```
    """
```

```
    Args:
```

```
    x: (B, C, H, W)
```

```
    window_size (int): window size
```

```
    Returns:
```

(下页继续)

(续上页)

```

    windows: (B, C, num_windows*num_windows, window_size*window_size)
    """
    B, C, H, W = x.shape
    x = x.view(B, C, H // window_size, window_size, W // window_size, window_size)
    windows = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, -1, window_size*window_size)
    return windows

def window_reverse(windows, window_size, H, W):
    """
    Args:
        windows: (B, C, num_windows*num_windows, window_size*window_size)
        window_size (int): Window size
        H (int): Height of image
        W (int): Width of image

    Returns:
        x: (B, C, H, W)
    """
    B, C, __ = windows.shape
    x = windows.view(B, C, H // window_size, W // window_size, window_size, window_size)
    x = x.permute(0, 1, 2, 4, 3, 5).contiguous().view(B, C, H, W)
    return x

```

输入列表

- **X : T1**
 - **shape** : [batch, channel, height, width]

输出列表

- **Y : T1**
 - **shape** : [batch, channel, height, width]

数据类型约束

- **T1**: int8(Tensor)

属性列表

- mode : **string**
 - 支持模式: partition, reverse, partition_num_first
 - * partition: 参考 window_partition 实现;
 - * reverse: 参考 window_reverse 实现;
 - * partition_num_first: 与 partition 的区别在于, partition_num_first 的 H,W 按照 num_windows 进行划分, 而 partition 的 H,W 按照 window_sizes 进行划分。
 - window_sizes : **list of ints**
 - 2 维向量, 窗口的大小。定义了每个窗口的高度和宽度。窗口内的注意力计算和特征提取都是基于这个大小。
 - num_windows : **list of ints**
 - 2 维向量, 窗口的数量。定义了输入特征的高度和宽度被划分为多少个窗口。
-

其他支持

- 多核联合: 尚不支持
- [3566/3568 OP 规格介绍](#)
- [3588 OP 规格介绍](#)
- [3562 OP 规格介绍](#)
- [1103/1106 OP 规格介绍](#)
- [3576 OP 规格介绍](#)
- [2118 OP 规格介绍](#)
- [1103B OP 规格介绍](#)

CHAPTER 4

CPU OPs 规格介绍

4.1 注:

- RK2118 不支持所有 CPU 算子
 - RV1106/RV1103/RV1103B 仅支持部分 CPU 算子如下
 - Concat, DataConvert, Div, exLayerNorm, Exp, Gather, ReduceL2, ReduceMax, Reshape, ScatterND, Softmax
-

4.2 Add

–描述: 加法操作

–规格约束: 无限制

4.3 AveragePool

–描述: 平均池化

–规格约束: 无限制

4.4 ArgMin

–描述: 取最小值的 index

–规格约束: 无限制

4.5 ArgMax

–描述: 取最大值的 index

–规格约束: 无限制

4.6 BatchNormalization

–描述: 批量归一化

–规格约束: 无限制

4.7 Cast

–描述: 数据类型转换

–规格约束

SRC 支持: float32/bool/int8/float16/int32/int64

DST 支持: float32/int8/int32/float16

4.8 Clip

–描述: 数据截断激活层

–规格约束: 无限制

4.9 Concat

–描述: 合并操作

–规格约束: axis 仅支持 {0, 1, 2, 3}

4.10 Convolution

–描述: 卷积操作

–规格约束: 无限制

4.11 ConvTranspose/Deconvolution

- 描述: 转置卷积
- 规格约束: 无限制

4.12 ConvTransposePad

- 描述: 组合 OP
- 规格约束: 无限制

4.13 Cos

- 描述: 余弦函数
- 规格约束: 无限制

4.14 DataConvert

- 描述: 数据类型转换
- 规格约束: 仅支持 bool/int8/float 类型转换

4.15 DepthToSpace

- 描述: 通道方向空间方向转换
- 规格约束: 无限制

4.16 Div

- 描述: 除法操作
- 规格约束: 无限制

4.17 Equal

- 描述: 等于
- 规格约束: 无限制

4.18 Exp

- 描述: 指数函数
- 规格约束: 无限制

4.19 Flatten

- 描述: 拉平操作
- 规格约束: 无限制

4.20 Gather

- 描述: 聚集操作
- 规格约束: 无限制

4.21 Greater

- 描述: 大于
- 规格约束: 无限制

4.22 GreaterOrEqual

- 描述: 大等于
- 规格约束: 无限制

4.23 GRU

- 描述: 门控循环单元
- 规格约束: 无限制

4.24 exGRU

- 描述: 门控循环单元
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.25 exHardSwish

- 描述: 激活函数
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.26 InstanceNormalization

- 描述: 单例归一化
- 规格约束: 无限制

4.27 exLayerNorm

- 描述: 层归一化
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.28 Less

- 描述: 小于
- 规格约束: 无限制

4.29 LessOrEqual

- 描述: 小等于
- 规格约束: 无限制

4.30 LogSoftmax

- 描述: 激活函数
- 规格约束: batchsize 仅支持 1

4.31 LpNormalization

- 描述: Lp 归一化
- 规格约束: 无限制

4.32 exLRN

- 描述: 局部响应归一化
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.33 MatMul

- 描述: 多维矩阵相乘
- 规格约束: 无限制 (支持四维 x 四维、四维 x 三维计算)

4.34 Max

- 描述: 取最大值
- 规格约束: 无限制

4.35 MaxPool

- 描述: 最大池化
- 规格约束: 无限制

4.36 MaxRoiPool

- 描述: 区域最大池化
- 规格约束: 无限制

4.37 MaxUnpool

- 描述: 反向最大池化
- 规格约束: 无限制

4.38 exMish

- 描述: 激活函数
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.39 Min

- 描述: 取最小值
- 规格约束: 无限制

4.40 Mul

- 描述: 乘法
- 规格约束: 无限制

4.41 Pad

- 描述: 填充
- 规格约束: 无限制

4.42 Pow

- 描述: 指数计算
- 规格约束: 无限制

4.43 exProposal

- 描述: 区域提议网络
- 规格约束: batchsize 仅支持 1
- 说明: ONNX 扩展算子

4.44 ReduceMax

- 描述: 沿指定维度计算 Max
- 规格约束: 输出维度不能超过 4 维

4.45 ReduceMean

- 描述: 沿指定维度计算 Mean
- 规格约束: 输出维度不能超过 4 维

4.46 ReduceSum

- 描述: 沿指定维度计算 Sum
- 规格约束: 输出维度不能超过 4 维

4.47 ReduceMin

- 描述: 沿指定维度计算 Min
- 规格约束: 输出维度不能超过 4 维

4.48 Reorg

- 描述: 数据重排
- 规格约束: 无限制

4.49 Reshape

- 描述: 数据形状改变
- 规格约束: 无限制

4.50 Resize

- 描述: 数据宽高方向缩放
- 规格约束: 支持插值方式 bilinear; nearest2d

4.51 ReverseSequence

- 描述: 序列翻转
- 规格约束: 无限制

4.52 exRMSNorm

- 描述: 均方根归一化
- 规格约束: 无限制
- 说明: ONNX 扩展算子

4.53 RoiAlign

-描述: 区域对齐池化

-规格约束: 仅支持 Avg Pool Mode, batchsize 仅支持 1

4.54 ScatterND

-描述: N 维索引取数

-规格约束: 无限制

4.55 Sin

-描述: 正弦函数

-规格约束: 无限制

4.56 Slice

-描述: 切片操作

-规格约束: batchsize 仅支持 1

4.57 Softmax

-描述: 激活函数

-规格约束: batchsize 仅支持 1

-说明: 与 ONNX OPSET 11 规范一致

4.58 exSoftmax13

-描述: 激活函数

-规格约束: batchsize 仅支持 1

-说明: ONNX 扩展算子, 与 ONNX OPSET 13 规范一致

4.59 SpaceToDepth

-描述: 空间方向向通道方向转换

-规格约束: 无限制

4.60 Split

- 描述: 拆分数据
- 规格约束: 无限制

4.61 Sqrt

- 描述: 求平方根
- 规格约束: 无限制

4.62 Squeeze

- 描述: 压缩数据维度
- 规格约束: 无限制

4.63 Sub

- 描述: 减法
- 规格约束: 无限制

4.64 Tanh

- 描述: 双曲正切函数
- 规格约束: 无限制

4.65 Tile

- 描述: 扩充拷贝数据
- 规格约束: batchsize 仅支持 1, 不支持 broadcasting

4.66 Transpose

- 描述: 转置计算
- 规格约束: 无限制

4.67 Upsample

-描述: 上采样

-规格约束: 支持插值方式 bilinear; nearest2d

4.68 Not

-描述: 按元素取非

-规格约束: 无限制

4.69 where

-描述: 通过 mask 取数

-规格约束: 无限制

4.70 Erf

-描述: 误差函数

-规格约束: 无限制

4.71 Floor

-描述: 向下取整函数

-规格约束: 无限制

4.72 Mod

-描述: 取模函数

-规格约束: 无限制

4.73 exMeanVarianceNormalization

-描述: 均方差归一化函数

-规格约束: 无限制

4.74 And

-描述: 与函数

-规格约束: 无限制

4.75 GatherElements

-描述: 元素收集函数

-规格约束: 无限制

4.76 Log

-描述: Log 函数

-规格约束: 无限制

4.77 exDataConvert

-描述: 精度转换

-规格约束: 无限制

4.78 SpaceToDepth

-描述: 维度变换

-规格约束: 无限制

4.79 SoftmaxMask

-描述: where 融合 softmax 的激活函数

-规格约束: 无限制

4.80 TransposeReshape

-描述: 组合 OP

-规格约束: 无限制

4.81 ReshapeTranspose

- 描述: 组合 OP
- 规格约束: 无限制

4.82 ReduceL2

- 描述: 沿某一轴的 L2 范数
- 规格约束: 无限制

4.83 LayerNormalization

- 描述: 层标准化
- 规格约束: 无限制

4.84 PassThrough

- 描述: 直接连接
- 规格约束: 无限制

4.85 Expand

- 描述: 张量数据复制或扩展
- 规格约束: 无限制

4.86 Sigmoid

- 描述: 激活函数
- 规格约束: 无限制

4.87 Tanh

- 描述: 三角函数
- 规格约束: 无限制

4.88 Cos

- 描述: 三角函数
- 规格约束: 无限制

4.89 Sin

- 描述: 三角函数
- 规格约束: 无限制

4.90 exSwish

- 描述: 激活函数
- 规格约束: 无限制

4.91 exGlu

- 描述: 激活函数
- 规格约束: 无限制

4.92 exGelu

- 描述: 激活函数
- 规格约束: 无限制

4.93 exWindow

- 描述: exWindow 来源于 Swin-Transformer 的 window_partition 和 window_reverse 操作
- 规格约束: 无限制

4.94 exNorm

- 描述: 当 mode 设置为 'pixel '时, exNorm 将使用 PixelNorm 进行归一化。mode 缺省时, exNorm 默认使用 LayerNorm 进行归一化。
- 规格约束: 无限制

4.95 Round

- 描述: 数据饱和
- 规格约束: 无限制

4.96 TopK

- 描述: 获取数值排行前 K 个数据
- 规格约束: 无限制

4.97 exSwooshR

- 描述: 激活函数
- 规格约束: 无限制

4.98 exSwooshL

- 描述: 激活函数
- 规格约束: 无限制

4.99 OneHot

- 描述: 独热编码
- 规格约束: 无限制

5.1 模型输入限制说明

1. 该对齐约束仅针对零拷贝 API，普通 API 无此对齐约束
2. 输入宽的对齐要求可从零拷贝 API 中的 `w_stride` 属性查询到，注意:`w_stride` 不支持更改
3. 仅对输入宽 (width) 在不同的通道 (channel) 条件下有对齐要求，其他无约束
4. 若输入不需要 mean 和 scale，需要将 mean 和 scale 配置为 0 和 1
5. 若通道 (channel)> 4，则 mean/scale 将统一使用第一个数值，即 `mean[0]` 和 `scale[0]`
6. 若首层为浮点类型则没有 quant 操作
7. RV1106/RV1103/RV1103B 不支持 CPU 的 mean/scale/quant 操作
8. 详细的用法请参考《Rockchip_RKNPU_User_Guide_RKNN_SDK》

5.1.1 RK3566/RK3568

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽（width）对齐要求单位：元素个数 | | 输入宽（width）大小限制 | | |
|--------------|----------|-------|----------------|-------------------------|-----------------------|-----------------------|---|---|-----|
| | | | | | 当输入通道（channel）为 1,3,4 | 当输入通道（channel）非 1,3,4 | 当输入通道（channel）为 1,3,4 | 当输入通道（channel）非 1,3,4 | |
| RK3566 /3568 | int8 | 4 维度 | uint8 | NPU | 8 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 1. width * dilation_kernel_h < 1024*N 2. 2. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 128) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 4096) ConvTranspose/Deconvolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 128) | 无限制 | |
| | | | int8 | | | | | | |
| | | | float16 | CPU | | | | | |
| | | | 其他类型输入对齐要求可能变动 | | | | | | |
| | float16 | | uint8 | CPU | 4 | 1 | | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 1. width * dilation_kernel_h < 1024*N 2. 2. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 128) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 4096) ConvTranspose/Deconvolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 128) | 无限制 |
| | | | int8 | | | | | | |
| | | | float16 | | | | | | |
| | | | 其他类型输入对齐要求可能变动 | | | | | | |
| | 无限制 | 非 4 维 | 非限制 | CPU | 1 | 1 | 无限制 | | |

5.1.2 RK3588

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽 (width) 对齐要求单位: 元素个数 | | 输入宽 (width) 大小限制 | | | |
|--------|----------|-------|--------------------|-------------------------|--------------------------|-------------------------|--|--|-----|-----|
| | | | | | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 | | |
| RK3588 | int8 | 4 维度 | uint8 | NPU | 16 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式: 1. width * dilation_kernel_h <= 2048 * N 2. width <= 8192 其中 N 必须为 1 到 7 的整数, 超出范围的卷积不受支持, 各卷积类型 N 的计算方式如下: Convolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 128), 3) Depthwise Convolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 2048), 3) ConvTranspose/Deconvolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 128), 3) | 无限制 | | |
| | | | int8 | | | | | | | |
| | | | float16 | CPU | | | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | | | |
| | float16 | | uint8 | CPU | 8 | 1 | | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式: 1. width * dilation_kernel_h <= 1024 * N 2. width <= 8192 其中 N 必须为 1 到 7 的整数, 超出范围的卷积不受支持, 各卷积类型 N 的计算方式如下: Convolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 128), 3) Depthwise Convolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 2048), 3) ConvTranspose/Deconvolution: N = 12 - MAX(CEIL((dilation_kernel_h * dilation_kernel_w) / 128), 3) | 无限制 | |
| | | | int8 | | | | | | | |
| | | | float16 | | | | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | | | |
| | 无限制 | 非 4 维 | 非限制 | CPU | 1 | 1 | 无限制 | | | 无限制 |

5.1.3 RV1103/RV1106

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽 (width) 对齐要求单位: 元素个数 | | 输入宽 (width) 大小限制 | |
|-------------|----------|------|------------|-------------------------|--------------------------|-------------------------|---|-------------------------|
| | | | | | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 | 当输入通道 (channel) 为 1,3,4 (声明见注释 9) | 当输入通道 (channel) 非 1,3,4 |
| RV1103/1106 | int8 | 4 维度 | uint8 | NPU | 16 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式: 1. width * dilation_kernel_h <= 2048 * N 2. width <= 4096 其中 N 必须为 1 到 7 的整数, 超出范围的卷积不受支持, 各卷积类型 N 的计算方式如下: Convolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ Depthwise Convolution: $N = 8 - \text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 4096)$ ConvTranspose/Deconvolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ | 无限制 |

5.1.4 RK3562

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽（width）对齐要求单位：元素个数 | | 输入宽（width）大小限制 | |
|--------|----------|------|--------------------|----------------------------|---------------------------|---------------------------|--|---------------------------|
| | | | | | 当输入通道（channel） 为 1,3,4 | 当输入通道（channel） 非 1,3,4 | 当输入通道（channel）为 1,3,4 | 当输入通道（channel） 非 1,3,4 |
| RK3562 | int8 | 4 维度 | uint8 | NPU | 16 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 1. width * dilation_kernel_h <= 2048 * N 2. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w) / 128), 2) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w) / 4096) ConvTranspose/Deconvolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w) / 128), 2) | 无限制 |
| | | | int8 | | | | | |
| | | | float16 | CPU | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | |
| | float16 | | uint8 | CPU | 8 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 1. width * dilation_kernel_h <= 2048 * N 2. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w) / 128), 2) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w) / 4096) ConvTranspose/Deconvolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w) / 128), 2) | 无限制 |
| | | | int8 | | | | | |
| | | | float16 | | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | |
| 无限制 | 非 4 维 | 非限制 | CPU | 1 | 1 | 无限制 | 无限制 | |

5.1.5 RK3576

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽（width）对齐要求单位：元素个数 | | 输入宽（width）大小限制 | | |
|--------|----------|-------|--------------------|----------------------------|---------------------------|---------------------------|--|--|-----|
| | | | | | 当输入通道（channel） 为 1,3,4 | 当输入通道（channel） 非 1,3,4 | 当输入通道（channel） 为 1,3,4 | 当输入通道（channel） 非 1,3,4 | |
| RK3576 | int8 | 4 维度 | uint8 | NPU | 16 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 3. width * dilation_kernel_h <= 2048 * N 4. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w / 128), 2) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 4096) ConvTranspose/Deconvolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w / 128), 2) | 无限制 | |
| | | | int8 | | | | | | |
| | | | float16 | | | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | | |
| | float16 | | uint8 | CPU | 8 | 1 | | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式： 2. width * dilation_kernel_h <= 2048 * N 2. width <= 4096 其中 N 必须为 1 到 7 的整数，超出范围的卷积不受支持，各卷积类型 N 的计算方式如下： Convolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w / 128), 2) Depthwise Convolution: N = 8 - CEIL(dilation_kernel_h * dilation_kernel_w / 4096) ConvTranspose/Deconvolution: N = 8 - MAX(CEIL(dilation_kernel_h * dilation_kernel_w / 128), 2) | 无限制 |
| | | | int8 | | | | | | |
| | | | float16 | | | | | | |
| | | | 其他类型 输入对齐要求可能变动 | | | | | | |
| | 无限制 | 非 4 维 | 非限制 | CPU | 1 | 1 | 无限制 | | 无限制 |

5.1.6 RK2118

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽 (width) 对齐要求单位: 元素个数 | | 输入宽 (width) 大小限制 | |
|--------|----------|-------|------------|----------------------------|--------------------------|-------------------------|---|-------------------------|
| | | | | | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 |
| RK2118 | float16 | 4 维度 | float16 | CPU | 4 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式: 3. width * dilation_kernel_h <= 2048 * N 4. width <= 4096 其中 N 必须为 1 到 7 的整数, 超出范围的卷积不受支持, 各卷积类型 N 的计算方式如下: Convolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ Depthwise Convolution: $N = 8 - \text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 4096)$ ConvTranspose/Deconvolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ | 无限制 |
| | | 非 4 维 | | | 1 | 1 | | |

5.1.7 RV1103B

| 芯片平台 | 模型首层精度类型 | 输入维度 | 首层设置输入数据类型 | mean/scale/quant 后端实现设备 | 输入宽 (width) 对齐要求单位: 元素个数 | | 输入宽 (width) 大小限制 | |
|---------|----------|------|------------|----------------------------|--------------------------|-------------------------|---|-------------------------|
| | | | | | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 | 当输入通道 (channel) 为 1,3,4 | 当输入通道 (channel) 非 1,3,4 |
| RV1103B | int8 | 4 维度 | uint8 | NPU | 8 | 1 | 各卷积类型的 width/kernel_h/kernel_w 需要满足以下两式: 1. width * dilation_kernel_h <= 2048 * N 2. width <= 4096 其中 N 必须为 1 到 7 的整数, 超出范围的卷积不受支持, 各卷积类型 N 的计算方式如下: Convolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ Depthwise Convolution: $N = 8 - \text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 4096)$ ConvTranspose/Deconvolution: $N = 8 - \text{MAX}(\text{CEIL}(\text{dilation_kernel_h} * \text{dilation_kernel_w} / 128), 2)$ | 无限制 |
| | | | int8 | | | | | |

5.2 模型输出限制说明

1. 如果输出 tensor 类型是 NHWC 的，输出转换是 NPU 实现的输出，则有对齐要求，CPU 实现的没有对齐要求。
2. 输出精度类型 int8/float16 表示模型最后一层原始输出的数据类型。
3. NCHW 输出，如果是 NPU 实现采用零拷贝接口则输出内存开辟的 size 以 query 出来的 size 为准。
4. NC1HWC2 输出，输出内存开辟的 size 以 query 出来的 size 为准。

5.2.1 RK3566/RK3568

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出 Layout | Channel 对齐要求 | H*W 对齐要求 |
|----------------|----------|--------|-------------|---------------------------------|------------|
| RK3566 /RK3568 | int8 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 8 对齐 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子，16 对齐，最后一层非卷积类算子 8 对齐 | H*W 要 4 对齐 |
| | | | UNDE-FINE | 无 | 无 |
| | float16 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 4 对齐 | |
| | | | NC1HWC2 | 最后一层卷积类算子，8 对齐，最后一层非卷积类算子 4 对齐 | H*W 要 4 对齐 |
| | | | UNDE-FINE | 无 | 无 |
| | 无限制 | 非 4 维度 | UNDE-FINE | 无 | 无 |

5.2.2 RK3588

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出 Layout | Channel 对齐要求 | H*W 对齐要求 |
|--------|----------|--------|-------------|----------------------------------|------------|
| RK3588 | int8 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 16 对齐 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子，32 对齐，最后一层非卷积类算子 16 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | float16 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 8 对齐 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子，16 对齐，最后一层非卷积类算子 8 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | 无限制 | 非 4 维度 | UNDEFINE | 无 | 无 |

5.2.3 RV1103/RV1106

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出Layout | Channel 对齐要求 | H*W 对齐要求 |
|----------------|----------|------|------------|------------------------------------|------------|
| RV1103 /RV1106 | int8 | 4 维度 | NC1HWC2 | 最后一层卷积类算子, 32 对齐, 最后一层非卷积类算子 16 对齐 | H*W 要 4 对齐 |
| | | | NHWC | 无 | 无 |

5.2.4 RK3562

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出Layout | Channel 对齐要求 | H*W 对齐要求 |
|--------|----------|--------|------------|------------------------------------|------------|
| RK3562 | int8 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 无 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子, 32 对齐, 最后一层非卷积类算子 16 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | float16 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 无 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子, 16 对齐, 最后一层非卷积类算子 8 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | 无限制 | 非 4 维度 | UNDEFINE | 无 | 无 |
| | | | UNDEFINE | 无 | 无 |

5.2.5 RK3576

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出Layout | Channel 对齐要求 | H*W 对齐要求 |
|--------|----------|--------|------------|------------------------------------|------------|
| RK3576 | int8 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 无 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子, 32 对齐, 最后一层非卷积类算子 16 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | float16 | 4 维度 | NCHW | 无 | 无 |
| | | | NHWC | 无 | 无 |
| | | | NC1HWC2 | 最后一层卷积类算子, 16 对齐, 最后一层非卷积类算子 8 对齐 | H*W 要 4 对齐 |
| | | | UNDEFINE | 无 | 无 |
| | 无限制 | 非 4 维度 | UNDEFINE | 无 | 无 |
| | | | UNDEFINE | 无 | 无 |

5.2.6 RK2118

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出Layout | Channel 对齐要求 | H*W 对齐要求 |
|--------|----------|--------|------------|----------------------------------|------------|
| RK2118 | float16 | 4 维度 | NC1HWC2 | 最后一层卷积类算子, 8 对齐, 最后一层非卷积类算子 4 对齐 | H*W 要 4 对齐 |
| | | | NHWC | 无 | 无 |
| | | 非 4 维度 | UNDEFINE | 无 | 无 |

5.2.7 RV1103B

| 芯片平台 | 模型输出精度类型 | 输出维度 | 设置输出Layout | Channel 对齐要求 | H*W 对齐要求 |
|---------|----------|------|------------|----------------------------------|------------|
| RV1103B | Bint8 | 4 维度 | NC1HWC2 | 最后一层卷积类算子, 8 对齐, 最后一层非卷积类算子 8 对齐 | H*W 要 4 对齐 |
| | | | NHWC | 无 | 无 |

CHAPTER 6

特殊说明

内容

- 特殊说明
 - dilated_kernel_h
 - dilated_kernel_w

6.1 dilated_kernel_h

$\text{dilated_kernel_h} = \text{kernel_h} * \text{dilations_h} - \text{dilations_h} + 1$

6.2 dilated_kernel_w

$\text{dilated_kernel_w} = \text{kernel_w} * \text{dilations_w} - \text{dilations_w} + 1$