



Denglin Hamming™ V2

dINNE TU Operator

DL-DG/SW-035C-02

2025-2-12

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

商标和许可



和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。

本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼5号1101室，江苏，中国

<http://www.denglin.ai>

Email : support@denglin.ai

更新历史

版本	更新描述
02	文档由《登临 Hamming V2 dINNE Quantization TU Operator》更名为《登临 Hamming V2 dINNE TU Operator》。 优化内容描述。
01	第一次发布。

内容目录

[更新历史](#)

[内容目录](#)

[1 概述](#)

[1.1 TU 算子简介](#)

[1.2 TU 算子支持列表](#)

[2 TU 算子匹配规则](#)

[2.1 基本通用规则](#)

[2.2 算子匹配规则](#)

[2.2.1 OutputQuantConvPattern 规则](#)

[2.2.2 SoftmaxPluginPattern 规则](#)

[2.2.3 NormPluginPattern 规则](#)

[2.2.4 OutputTensorListPattern 规则](#)

[2.2.4.1 具体推导规则](#)

[OutputTensorList 产生式](#)

[OutputTensor 产生式](#)

[InterTensor 产生式](#)

[TDOUnary 产生式](#)

[TDOBypassOut 产生式](#)

[TDOQuantize 产生式](#)

[Unary 产生式](#)

[Binary 产生式](#)

[Trinary 产生式](#)

[Pooling 产生式](#)

[Reduce 产生式](#)

[ConvOutputTensor 产生式](#)

[Conv 产生式](#)

[ConvAdd 产生式](#)

[ConvInputTensor 产生式](#)

[ConvAddedTensor 产生式](#)

[TDIProcess 产生式](#)

[TDIUnaryProcess 产生式](#)

[Add 产生式](#)

[Quantize 产生式](#)

[Dequantize 产生式](#)

[Relu 产生式](#)

[2.2.4.2 终结符表](#)

[3 TU 算子描述](#)

[3.1 符号说明](#)

[Shape](#)

[Consts](#)

[DataType](#)

[Layout](#)

[3.2 算子详述](#)

[pre_cast](#)

[post_cast](#)

[quantize](#)

[dequantize](#)

[mad](#)

[All Binary OPs](#)

[All Unary OPs](#)

reduce
padding
upsampling
normal_pooling
normal_conv
normal_conv_add
transpose_conv
transpose_conv_add
output_quantize_normal_conv
output_quantize_transpose_conv
softmax_plugin
matmul
norm_plugin
logic

3.3 Shape 范围限制

3.3.1 Shape 限制1

3.3.2 Shape 限制2

3.3.3 Shape 限制3

1 概述

1.1 TU 算子简介

TU (Tensor Unit) 为张量运算硬件加速单元，实现对神经网络中常见的计算密集型任务加速。本文主要介绍TU算子以及TU算子模式匹配的相关内容。

1.2 TU 算子支持列表

登临 dINNE 当前支持的TU算子如下表所示：

关于TU算子的详细介绍，请参见章节 **3 TU 算子描述**。

算子类别	算子名称
Unary ops	leaky_p_relu
Unary ops	sigmoid
Unary ops	tanh
Unary ops	gelu
Unary ops	relu
Unary ops	relu6
Unary ops	log2
Unary ops	sqrt
Unary ops	exp2
Unary ops	rsq
Unary ops	rcp
Binary ops	multiply
Binary ops	add
Binary ops	min
Binary ops	max
Binary ops	cmp
Trinary ops	mad
Conv ops	normal_conv
Conv ops	normal_conv_add

算子类别	算子名称
Conv ops	transpose_conv
Conv ops	transpose_conv_add
Conv ops	output_quantize_normal_conv
Conv ops	output_quantize_transpose_conv
Opaque ops	reduce
Opaque ops	padding
Opaque ops	normal_pooling
Opaque ops	upsampling
Opaque ops	logic (and, or, xor)
Plugin ops	softmax_plugin
Plugin ops	norm_plugin
Data type conversion ops	quantize
Data type conversion ops	pre_quantize_cast
Data type conversion ops	dequantize
Data type conversion ops	pre_cast
Data type conversion ops	post_cast

2 TU 算子匹配规则

神经网络通常表达为DAG图（有向无环图）。如果DAG图中的部分算子及其连接关系，以算子输出点起始**反向推导**的方式，满足一定的匹配规则（称为 TU Pattern），那么这个DAG子图将会被dINNE判断为一个TU Program，并由TU张量运算硬件单元进行加速计算。

下文将描述具体的匹配规则。

2.1 基本通用规则

1. DAG子图是单连通的。
2. DAG子图中最多含有一个 conv (matmul) OP。
3. DAG子图中最多含有一个 pooling OP。
4. 所有的 InputTensor 必须具有相同的 Shape（Conv InputTensor 除外）。
5. 有 padding 的 pooling OP 和 reduce OP 不能在同一个 TU Program中。
6. 如果 normal_pooling 的 padding 数量非0，那么在TU Pattern中，只能在 normal_pooling 这一路的结果有输出。

2.2 算子匹配规则

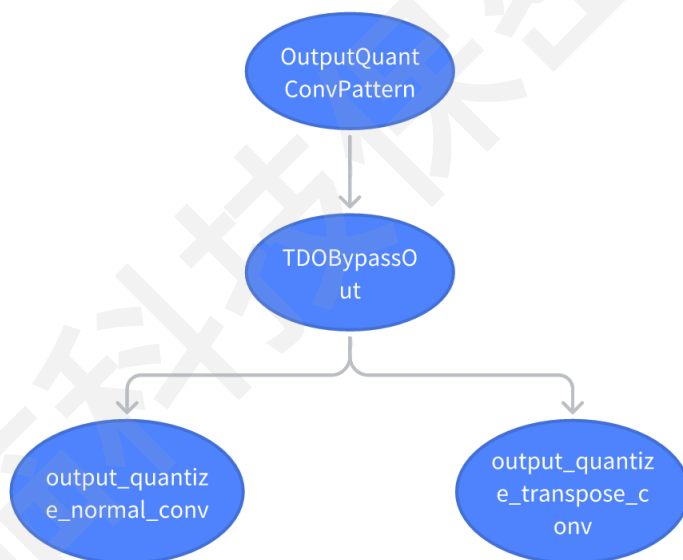
在介绍具体的算子匹配规则之前，先对描述中使用的术语及符号进行说明如下：

- "Fixed pipeline"表示固定顺序的算子操作，需要满足顺序的要求。
- "|"表示满足多个算子操作中的其中一个即可。
- " ϵ "表示在固定顺序中该处的算子操作可以为空操作。

当前，如果满足以下四种匹配规则的其中任意一种，即可被dINNE判断为一个TU Program。

1. OutputQuantConvPattern
2. SoftmaxPluginPattern
3. NormPluginPattern
4. OutputTensorListPattern (普遍规则，涵盖大部分TU支持的算子)

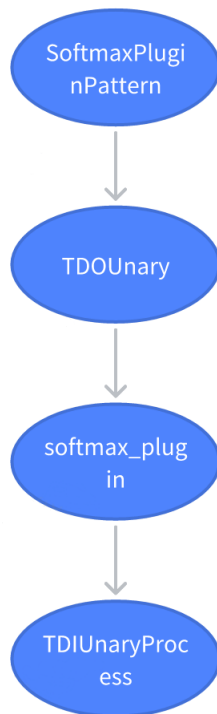
2.2.1 OutputQuantConvPattern 规则



TDOBypassOut (Fixed pipeline)

$$TDOBypassOut \rightarrow (out)(store_layout_convert|\epsilon)$$

2.2.2 SoftmaxPluginPattern 规则



- **TDOUnary (Fixed pipeline)**

$TDOUnary \rightarrow (out)(store_layout_convert|\epsilon)(TDOQuantize|post_cast|\epsilon)(relu|relu6|\epsilon)$

- **TDIUnaryProcess (Fixed pipeline)**

相比于TDIProcess，TDIUnaryProcess不允许对load的surface产生任何形状上的变化。

$TDIUnaryProcess \rightarrow (tdi_dequantize|pre_cast|\epsilon)(load_layout_convert|\epsilon)(in)$

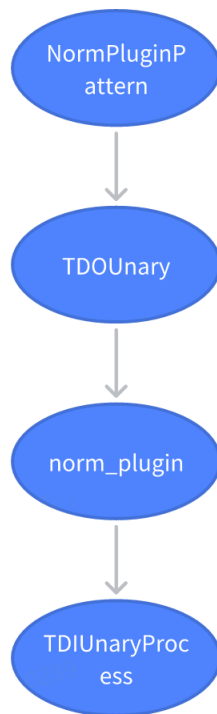
- **TDOQuantize (Fixed pipeline)**

$TDOQuantize \rightarrow out\#tdo_quantize * tdo_pre_quantize_cast$

说明：

tdo_quantize * tdo_pre_quantize_cast中的“*”号表示，这里两个OP在pattern中必须连用。

2.2.3 NormPluginPattern 规则



- **TDOUnary (Fixed pipeline)**

$$TDOUnary \rightarrow (out)(store_layout_convert|\epsilon)(TDOQuantize|post_cast|\epsilon)(relu|relu6|\epsilon)$$

- **TDIUnaryProcess (Fixed pipeline)**

相比于TDIProcess，TDIUnaryProcess不允许对load的surface产生任何形状上的变化。

$$TDIUnaryProcess \rightarrow (tdi_dequantize|pre_cast|\epsilon)(load_layout_convert|\epsilon)(in)$$

- **TDOQuantize (Fixed pipeline)**

$$TDOQuantize \rightarrow out\#tdo_quantize * tdo_pre_quantize_cast$$

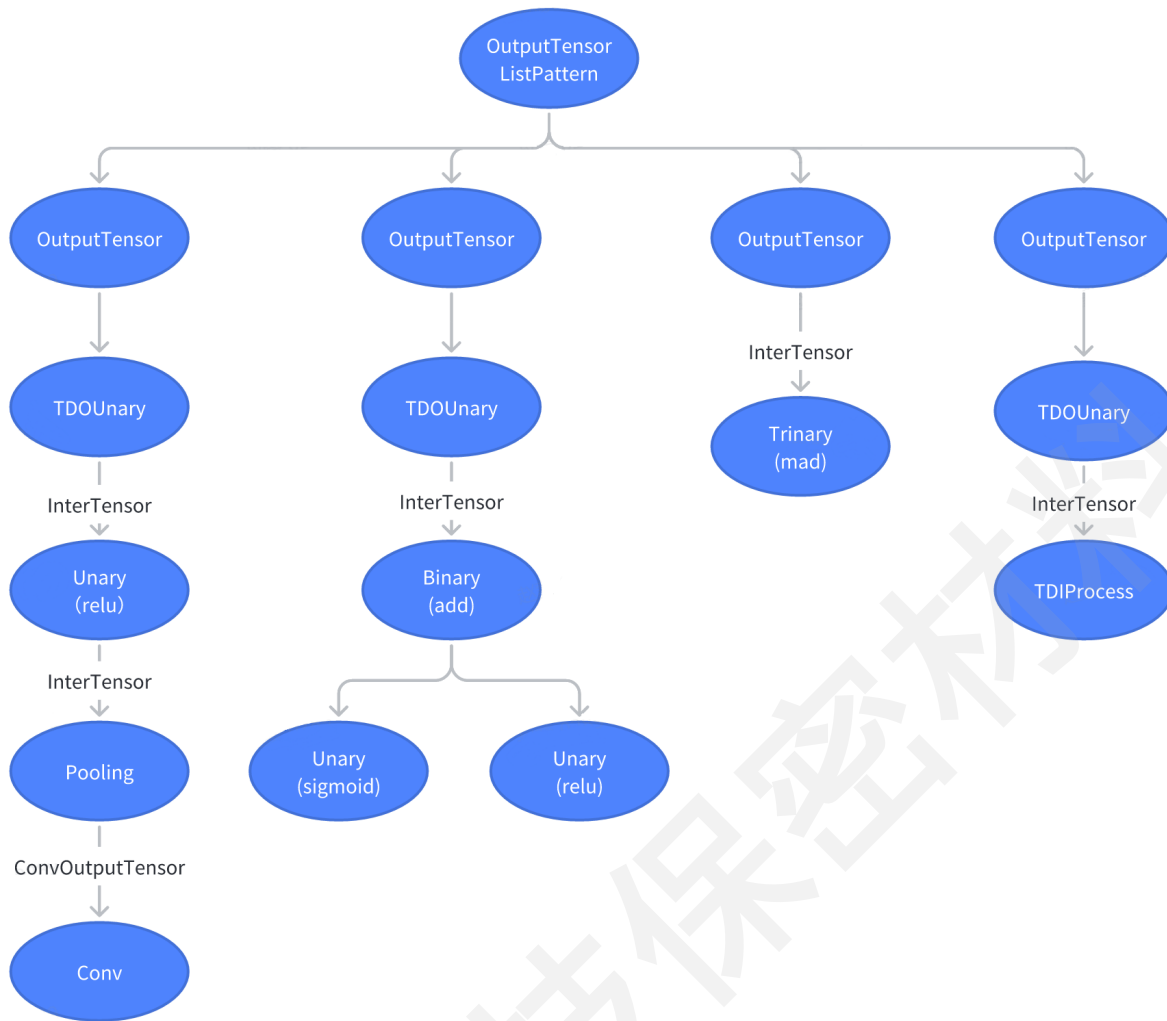
说明：

tdo_quantize * tdo_pre_quantize_cast 中的“*”号表示：这里两个OP在pattern中必须连用。

2.2.4 OutputTensorListPattern 规则

OutputTensorListPattern 规则并非匹配固定的DAG子图，而是在满足“基本通用规则”的前提下，可以在DAG图广度（最多支持四个 OutputTensor）和深度上进行反向推导匹配，直至匹配到表示终结的“终结符”（详细内容参见[2.2.4.2 终结符表](#)）。

下图为表现多样性的一个示例：



- **TDOUnary (Fixed pipeline)**

$TDUnary \rightarrow (out)(store_layout_convert|\epsilon)(TDOQuantize|post_cast|\epsilon)(relu|relu6|\epsilon)$

- **TDIPProcess (Fixed pipeline)**

$TDIPProcess \rightarrow$
 $(padding|\epsilon)(tdi_dequantize|pre_cast|\epsilon)(upsampling|\epsilon)(load_layout_convert|\epsilon)(in)$
 $TDIPProcess \rightarrow$
 $(padding|\epsilon)(upsampling|\epsilon)(tdi_dequantize|pre_cast|\epsilon)(load_layout_convert|\epsilon)(in)$
 $TDIPProcess \rightarrow$
 $(tdi_dequantize|pre_cast|\epsilon)(padding|\epsilon)(upsampling|\epsilon)(load_layout_convert|\epsilon)(in)$

2.2.4.1 具体推导规则

OutputTensorList 产生式

$$\begin{aligned}
 &OutputTensorList \rightarrow \\
 &\quad [OutputTensor] \\
 &\quad [OutputTensor, OutputTensor,] \\
 &\quad [OutputTensor, OutputTensor, OutputTensor,] \\
 &\quad [OutputTensor, OutputTensor, OutputTensor, OutputTensor,]
 \end{aligned}$$
说明：

每个 TU Program 最多输出4个Tensor。

OutputTensor 产生式

$$OutputTensor \rightarrow TDOUnary(InterTensor)$$
InterTensor 产生式

$$\begin{aligned}
 &InterTensor \rightarrow \\
 &\quad (InterTensor) \\
 &\quad SEUnary(InterTensor) \\
 &\quad SEBinary(InterTensor, InterTensor) \\
 &\quad SETrinary(InterTensor, InterTensor, InterTensor) \\
 &\quad SEReduce(InterTensor) \\
 &\quad SEPooling(InterTensor) \\
 &\quad ConvOutputTensor \\
 &\quad TDIProcess(input_tensor)
 \end{aligned}$$
TDOUnary 产生式

$$TDOUnary \rightarrow (out)(store_layout_convert|\epsilon)(TDOQuantize|post_cast|\epsilon)(relu|relu6|\epsilon)$$
TDOBypassOut 产生式

$$TDOBypassOut \rightarrow (out)(store_layout_convert|\epsilon)$$
TDOQuantize 产生式

$$TDOQuantize \rightarrow out\#tdo_quantize * tdo_pre_quantize_cast$$
说明：

tdo_quantize * tdo_pre_quantize_cast 中的“*”号表示，这里两个OP在pattern中必须连用。

Unary 产生式

Unary – >
leaky_p_relu
sigmoid
tanh
gelu
se_relu
relu6
log2
sqrt
exp2
rsq
rcp

Binary 产生式

Binary – >
multiply
se_add
min
max
cmp

Trinary 产生式

Trinary – > *mad*

Pooling 产生式

Pooling – > *normal_pooling*

说明：

ROI Pooling另做考虑。

Reduce 产生式

Reduce – > *reduce*

ConvOutputTensor 产生式

ConvOutputTensor – >
Conv(ConvInputTensor, weight_tensor)|ConvAdd(ConvInputTensor, weight_tensor, ConvAddedTensor)

Conv 产生式

$$Conv \rightarrow normal_conv | transpose_conv$$
ConvAdd 产生式

$$ConvAdd \rightarrow normal_conv_add | transpose_conv_add$$
ConvInputTensor 产生式

$$ConvInputTensor \rightarrow (conv_layout_convert | \epsilon)(conv_input_tensor)$$
ConvAddedTensor 产生式

$$ConvAddedTensor \rightarrow (upsampling | \epsilon)(padding | \epsilon)(conv_added_tensor)$$
TDIProcess 产生式

$$TDIProcess \rightarrow (padding | \epsilon)(tdi_dequantize | pre_cast | \epsilon)(upsampling | \epsilon)(load_layout_convert | \epsilon)(in)$$

$$TDIProcess \rightarrow (padding | \epsilon)(upsampling | \epsilon)(tdi_dequantize | pre_cast | \epsilon)(load_layout_convert | \epsilon)(in)$$

$$TDIProcess \rightarrow (tdi_dequantize | pre_cast | \epsilon)(padding | \epsilon)(upsampling | \epsilon)(load_layout_convert | \epsilon)(in)$$
说明：

padding和upsampling前后顺序是固定的（先做upsampling，再做padding），而tdi_dequantize|pre_cast|ε可以出现在它们的前、后或者中间。

TDIUnaryProcess 产生式

相比于TDIProcess，TDIUnaryProcess不允许对load的surface产生任何形状上的变化。

$$TDIUnaryProcess \rightarrow (tdi_dequantize | pre_cast | \epsilon)(load_layout_convert | \epsilon)(in)$$
Add 产生式

$$Add \rightarrow se_add$$

Quantize 产生式

$Quantize \rightarrow TDOQuantize$

Dequantize 产生式

$Dequantize \rightarrow tdi_dequantize$

Relu 产生式

$Relu \rightarrow tdo_relu|se_relu$

2.2.4.2 终结符表

TU 算子的终结符如下表所示：

算子类别	算子名称
Mark	in
Mark	out
Tensor	input_tensor
Tensor	conv_input_tensor
Tensor	conv_added_tensor
Tensor	weight_tensor
Trinary ops	mad
Binary ops	multiply
Binary ops	se_add
Binary ops	min
Binary ops	max
Binary ops	cmp
Unary ops	leaky_p_relu
Unary ops	sigmoid
Unary ops	tanh
Unary ops	gelu
Unary ops	tdo_relu

算子类别	算子名称
Unary ops	se_relu
Unary ops	relu6
Unary ops	log2
Unary ops	sqrt
Unary ops	exp2
Unary ops	rsq
Unary ops	rcp
Data type conversion	quantize
Data type conversion	pre_quantize_cast
Data type conversion	dequantize
Data type conversion	pre_cast
Data type conversion	post_cast
Conv	normal_conv
Conv	normal_conv_add
Conv	transpose_conv
Conv	transpose_conv_add
Conv	output_quantize_normal_conv
Conv	output_quantize_transpose_conv
Opaque	reduce
Opaque	padding
Opaque	normal_pooling
Opaque	upsampling
Opaque	logic (and, or, xor)
Plugin	softmax_plugin
Plugin	norm_plugin

3 TU 算子描述

3.1 符号说明

在描述具体的TU算子之前，先对描述中使用的符号进行说明如下。

Shape

- FEATURE_PICONST: integer, 1~65535 or -1 (for dynamic)
- FILTER_PICONST: integer, 1~255
- TensorShape: [FEATURE_PICONST, FEATURE_PICONST, FEATURE_PICONST, FEATURE_PICONST, FEATURE_PICONST]
- WeightShape: [FEATURE_PICONST, FILTER_PICONST, FILTER_PICONST, FILTER_PICONST, FEATURE_PICONST]

Consts

- PADDING_ICONST: integer, 0~511
- POOLING_PADDING_ICONST: integer, 0~254
- POOLING_WINDOW_ICONST: integer, 1~255
- POOLING_STRIDE_ICONST: integer, 1~255
- POWER2_PICONST: integer, 1,2,4,8 ...
- CONV_DILATION_ICONST: integer, 1~255
- CONV_FILTER_ICONST: integer, 1~255
- CONV_STRIDE_ICONST: integer, 1~255

DataType

- UINT4, INT4, UINT8, INT8, UINT32, INT32, FP16, BF16, FP32

Layout

- Layout 列表
 - NDHWC、NCDHW、NWHC、WHNC、WNHC、HWNC、HNWC、NHCW、HNCW、HCNW、CNHW、CHNW
- Layout分类
 - DSLayout
 - NCDHW、NDHWC、WCUniteAlignLayout、WC3CLayout、TiledLayout
 - WSLayout
 - OHWILayout、OILayout
 - TDILayout
 - NCDHW、NDHWC、TiledLayout
 - TDOLayout

- WLastLinearLayoutArbitraryStride
- CLastLinearLayoutArbitraryStride
- TiledLayout
- Layout 分类描述
 - WLastLinearLayoutArbitraryStride : 以W为最低维的任意线性layout , 不要求任何对齐。
 - CLastLinearLayoutArbitraryStride : 以C为最低维的任意线性layout , 不要求任何对齐。
 - TiledLayout : NDHWCTiled, NCDHWCTiled
 - WCUniteAlignLayout : NDHWC layout , 但是要求WC联合对齐32Byte。
 - WC3CLayout : NDHWC layout , 但是要求C=3 , W的stride具备32Byte对齐。
 - OHWILayout : conv 的 weight 的 tiled layout , 对于fw,fh > 1的情况 , 只支持这种layout。
 - OILayout : conv 的 weight 的 linear layout。

3.2 算子详述

本节对TU算子的内容做进一步详细的描述。

pre_cast

.expression

output_tensor = pre_cast (input_tensor, out_dtype)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.out_dtype (enum)

- Enum: INT32 | FP16 | BF16 | FP32

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: INT32 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- support datatype combinations:
 - FP16 -> BF16 | FP32
 - BF16 -> FP16 | FP32
 - FP32 -> FP16 | BF16
 - UINT4 | INT4 -> FP16 | FP32
 - UINT8 | INT8 -> FP16 | FP32
 - UINT4 | INT4 | UINT8 | INT8 -> INT32

post_cast

.expression

output_tensor = post_cast(input_tensor, out_dtype)

.input_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.out_dtype(enum)

- Enum: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- support datatype combinations:
BF16 -> FP16 | FP32
FP32 -> FP16 | BF16
FP32 | FP16-> UINT4 | INT4
FP32 | FP16-> UINT8 | INT8

quantize

.expression

output_tensor = quantize (input_tensor, scale, zero_point, out_dtype)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16
- TensorClassAttr.shape: TensorShape

.scale (scalar)

- Scalar : FP32

.zero_point (scalar)

- Scalar : INT32

.out_dtype (enum)

- Enum: UINT4 | INT4 | UINT8 | INT8

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape

.description

- Support datatype combinations: FP16 -> UINT4 | INT4 | UINT8 | INT8

dequantize

.expression

output_tensor = dequantize (input_tensor, scale, zero_point, out_dtype)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape

.scale (scalar)

- Datatype : FP32

.zero_point (scalar)

- Scalar : INT32

.out_dtype (enum)

- Enum: FP16 | BF16 | FP32

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- Support datatype combinations: UINT4 | INT4 | UINT8 | INT8 -> FP16 | BF16 | FP32

The dequantize's behavior is: src_datatype dequantize to FP32 and then cast to dst_datatype.

mad

.expression

output_tensor = mad (input_tensor_a, input_tensor_b, input_tensor_c)

.input_tensor_a (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.input_tensor_b (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.input_tensor_c (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32

- TensorClassAttr.shape: TensorShape

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- Broadcast axis support in dims of D, H, W.

All Binary OPs

.expression

output_tensor = binary_op (input_tensor_a, input_tensor_b)

.input_tensor_a (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.input_tensor_b (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- Broadcast axis support in dims of D, H, W.

All Unary OPs

.expression

output_tensor = unary_op(input_tensor)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- Gelu only supports FP16 datatype.

reduce

.expression

output_tensor = reduce (input_tensor)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- input_tensor must have five dimentions.
- only support reduction dims of D, H, W and reduced at same time.

padding

.expression

output_tensor = padding(input_tensor, padding_tuple, padding_mode, padding_value,input_layout,output_layout)

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDILayout

.padding_tuple (tuple)

- Tuple: (padding_front, padding_back, padding_top, padding_bottom, padding_left, padding_right), each of them ranges in PADDING_ICONST.

.padding_mode (enum)

- Enum: constant

.padding_value (value)

- Padding value is a value used for 'constant' mode padding, note the datatype for this value should match with input tensor's.

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

upsampling

.expression

```
output_tensor = upsampling(input_tensor, w_uppooling_window_width, h_uppooling_window_width,
d_uppooling_window_width)
```

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32

.w_uppooling_window_width

- POWER2_PICONST, <= 128

.h_uppooling_window_width

- POWER2_PICONST, <= 128

.d_uppooling_window_width

- POWER2_PICONST, <= 128

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32

.description

- input_tensor must have five dimensions.
- only support nearest upsampling in dims of D, H, W.

normal_pooling

.expression

```
output_tensor = upsampling (input_tensor, padding_tuple, pooling_window_tuple, pooling_stride_tuple,
pooling_type)
```

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.padding_tuple (tuple)

- Tuple: (padding_front, padding_back, padding_top, padding_bottom, padding_left, padding_right), each of them ranges in PADDING_ICONST.

.pooling_window_tuple (tuple)

- Tuple: (pooling_window_d, pooling_window_h, pooling_window_w), each of them ranges in POOLING_WINDOW_ICONST.

.pooling_stride_tuple (tuple)

- Tuple: (pooling_stride_d, pooling_stride_h, pooling_stride_w), each of them ranges in POOLING_STRIDE_ICONST.

.pooling_type (enum)

- Enum: MAXPOOLING, MINPOOLING, MAXPOOLING_WITH_IDX, MINPOOLING_WITH_IDX, AVG_EXCLUDE_POOLING, AVG_INCLUDE_POOLING

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- AVG_EXCLUDE_POOLING do not support BF16.

normal_conv

.expression

output_tensor = normal_conv(input_tensor, weight_tensor, input_padding, dilation_tuple, filter_tuple, stride_tuple, input_c, output_c, group_count, input_scales(optional), input_zero_points(optional), weight_scales(optional), weight_zero_points(optional), output_scales(optional), output_zero_points(optional))

.precision type

- **quantized conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: INT32 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.input_scales (scalar)

- Scalar: FP32

.input_zero_points (scalar)

- Scalar: INT32

.weight_scales (scalar or vector)

- Scalar: FP32, Vector: FP32 (for per-channel quantization)

.weight_zero_points (scalar or vector)

- Scalar: INT32, Vector: INT32 (for per-channel quantization)

.output_scales (scalar)

- Scalar: FP32

.output_zero_points (scalar)

- Scalar: INT32

- **int conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: INT32

- **float conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32

.input_padding

- Same with "padding"

.dilation_tuple (tuple)

- Tuple: (dilation_w, dilation_h, dilation_d), each of them ranges in CONV_DILATION_ICONST.

.filter_tuple (tuple)

- Tuple: (filter_h, filter_w, filter_d), each of them ranges in CONV_FILTER_ICONST.

.stride_tuple (tuple)

- Tuple: (stride_w, stride_h, stride_w), each of them ranges in CONV_STRIDE_ICONST.

.input_c (int)

- Int: range from 1~65535.

.output_c (int)

- Int: range from 1~65535.

.group_count (int)

- Int: range from 1~65535.

normal_conv_add

.expression

```
output_tensor = normal_conv(input_tensor, weight_tensor, add_tensor, input_padding, add_padding, dilation_tuple,
filter_tuple, stride_tuple, input_c, output_c, group_count, input_scales(optional), input_zero_points(optional),
weight_scales(optional), weight_zero_points(optional), add_scales(optional), add_zero_points(optional) ,
output_scales(optional), output_zero_points(optional))
```

.precision type

- **quantized conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLLayout

.add_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDILayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: INT32 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.input_scales (scalar)

- Scalar: FP32

.input_zero_points (scalar)

- Scalar: INT32

.weight_scales (scalar or vector)

- Scalar: FP32, Vector: FP32 (for per-channel quantization)

.weight_zero_points (scalar or vector)

- Scalar: INT32, Vector: INT32 (for per-channel quantization)

.add_scales (scalar)

- Scalar: FP32

.add_zero_points (scalar)

- Scalar: INT32

.output_scales (scalar)

- Scalar: FP32

.output_zero_points (scalar)

- Scalar: INT32

- **int conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLayout

.add_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDILayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: INT32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

• float conv

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLayout

.add_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDILayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.input_padding

- Same with "padding"

.add_padding

- Same with "padding"

.dilation_tuple (tuple)

- Tuple: (dilation_w, dilation_h, dilation_d), each of them ranges in CONV_DILATION_ICONST.

.filter_tuple (tuple)

- Tuple: (filter_h, filter_w, filter_d), each of them ranges in CONV_FILTER_ICONST.

.stride_tuple (tuple)

- Tuple: (stride_w, stride_h, stride_w), each of them ranges in CONV_STRIDE_ICONST.

.input_c (int)

- Int: range from 1~65535.

.output_c (int)

- Int: range from 1~65535.

.group_count (int)

- Int: range from 1~65535.

.description

- add_tensor has the same shape as output_tensor.

transpose_conv

.description

- same as normal_conv, except: stride_tuple must be power of 2.

transpose_conv_add

.description

- same as normal_conv_add, except: stride_tuple must be power of 2.

output_quantize_normal_conv

.expression

output_tensor = output_quantize_normal_conv(input_tensor, weight_tensor, input_padding, dilation_tuple, filter_tuple, stride_tuple, input_c, output_c, group_count, activation, input_scales(optional), input_zero_points(optional), weight_scales(optional), weight_zero_points(optional), output_scales(optional), output_zero_points(optional))

.precision type

- **quantized conv**

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.input_scales (scalar)

- Scalar: FP32

.input_zero_points (scalar)

- Scalar: INT32

.weight_scales (scalar or vector)

- Scalar: FP32, Vector: FP32 (for per-channel quantization)

.weight_zero_points (scalar or vector)

- Scalar: INT32, Vector: INT32(for per-channel quantization)

.output_scales (scalar)

- Scalar: FP32

.output_zero_points (scalar)

- Scalar: INT32

• float conv

.input_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.weight_tensor (class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: WeightShape
- TensorClassAttr.layout: WSLayout

.output_tensor (class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.input_padding

- Same with "padding"

.dilation_tuple (tuple)

- Tuple: (dilation_w, dilation_h, dilation_d), each of them ranges in CONV_DILATION_ICONST.

.filter_tuple (tuple)

- Tuple: (filter_h, filter_w, filter_d), each of them ranges in CONV_FILTER_ICONST.

.stride_tuple (tuple)

- Tuple: (stride_w, stride_h, stride_w), each of them ranges in CONV_STRIDE_ICONST.

.input_c (int)

- Int: range from 1~65535.

.output_c (int)

- Int: range from 1~65535.

.group_count (int)

- Int: range from 1~65535.

.activation (enum)

- Enum: Relu, Relu6, None

.description

- This op has activation.
- It can't fused with other ops in pattern.

output_quantize_transpose_conv

.description

- same as output_quantize_normal_conv , except: stride_tuple must be power of 2.

softmax_plugin

.expression

output_tensor = softmax_plugin(input_tensor)

.input_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | FP32
- TensorClassAttr.shape: TensorShape

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- input_tensor's dims: $D * H * W \leq 1024$

matmul

.expression

output_tensor = matmul(input_tensor1, input_tensor2, group_count)

.tensor shape and layout

- When group_count==1

- input_tensor1 of shape (N,D,H,W,Ic)
- input_tensor2 of shape (Oc, Ic)
- Matmul performs (N,D,H,W,Ic) x (Oc, Ic) -> (N,D,H,W,Oc), programmer may implements input/output-Transpose by choosing input/output layout.
- When group_count>1
 - input_tensor1 of shape (group_count,N,D,H,W,Ic)
 - input_tensor2 of shape (group_count, Oc, Ic)
 - Matmul performs (group_count,N,D,H,W,Ic) x (group_count, Oc, Ic) -> (group_count,N,D,H,W,Oc), programmer may implements input/output-Transpose by choosing input/output layout.
 - group_count axis is "bind to" N axis, and may not choose memory stride arbitrarily.

.precision type

- quantized matmul

.input_tensor1(class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.input_tensor2(class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: 2D Tensor(when not batchmatmul) or 3D tensor(when batchmatmul)
- TensorClassAttr.layout: DSLLayout

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: INT32 | FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

Per channel quantization is supported with input_tensor2.

- int matmul

.input_tensor1(class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: DSLLayout

.input_tensor2(class:Tensor)

- TensorClassAttr.datatype: UINT4 | INT4 | UINT8 | INT8
- TensorClassAttr.shape: 2D Tensor(when not batchmatmul) or 3D tensor(when batchmatmul)
- TensorClassAttr.layout: DSLLayout

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: INT32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

- float matmul

.input_tensor1(class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape

- TensorClassAttr.layout: DSLLayout

.input_tensor2(class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: 2D Tensor(when not batchmatmul) or 3D tensor(when batchmatmul)
- TensorClassAttr.layout: DSLLayout

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | BF16 | FP32
- TensorClassAttr.shape: TensorShape
- TensorClassAttr.layout: TDOLayout

.description

- output_tensor has the same shape as input tensors.

norm_plugin

.expression

output_tensor = norm_plugin(input_tensor)

.input_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | FP32
- TensorClassAttr.shape: TensorShape

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: FP16 | FP32
- TensorClassAttr.shape: TensorShape

.description

- The reduced axes happen on D, H, W.
- input_tensor's dims: $D * H * W \leq 768$

logic

.expression

output_tensor = logic(input_tensor1, input_tensor2, logic_op, popcount_enable)

output_tensor = logic(input_tensor1, const_value, logic_op, popcount_enable)

.input_tensor1(class:Tensor)

- TensorClassAttr.datatype: UINT8 | UINT16 | UINT32
- TensorClassAttr.shape: TensorShape

.input_tensor2(class:Tensor)

- TensorClassAttr.datatype: UINT8 | UINT16 | UINT32
- TensorClassAttr.shape: TensorShape

.const_value(class:Tensor)

- TensorClassAttr.datatype: UINT8 | UINT16 | UINT32 (same with input tensor1)

.output_tensor(class:Tensor)

- TensorClassAttr.datatype: UINT8 | UINT16 | UINT32
- TensorClassAttr.shape: TensorShape

.logic_op(enum)

- Enum: AND, OR, XOR

.popcount_enable(bool)

If enable, output = popcount(logic_op(input1, input2)).

3.3 Shape 范围限制

对于每个OP，需要检查其Input Shape是否在允许范围内。不同的OP需要满足不同类型的Shape限制。

3.3.1 Shape 限制1

普适限制，没有特殊Shape限制规则的OP会需要遵循此限制。

pre_cast, post_cast, Quantize, tdo_quantize, Dequantize, tdi_dequantize, mad, binary op, unary op, reduce, pad, normal pooling, logic, layout_convert 需要满足此限制。

```
# constraint!!
input_feature_map_h <= 64005
input_feature_map_d <= 64005
input_feature_map_w <= 16775685
input_feature_map_n <= 65535
input_feature_map_c <= 65472

# constraint来源， 比如对于H维度
# input_feature_map_h <=
# 65535 (input rect限制) - 254(max pooling padding)*2 - 511(max pad padding)*2
```

3.3.2 Shape 限制2

normal_conv, normal_conv_add, output_quantize_normal_conv 需要满足此限制。

```
# constraint!!
input_feature_map_h <= 31746 - 508*conv_stride_h
input_feature_map_d <= 31746 - 508*conv_stride_d
input_feature_map_w <= 8387586 - 508*conv_stride_w
input_feature_map_n <= 65535
input_feature_map_c <= 65472
```

```
# d/h/w constraint来源, 比如对于H维度
# input_feature_map_h <=
# 32768 (input rect限制) - 254(max pooling padding)*2*conv_stride_h - 511(max conv
padding)*2

# c constraint来源
# ic_a的计算需要将ic向上用alignmax进行align, alignmax最大为64(UINT4)
# 而ic, ic_a的位宽为16bit, 即最大值为65535。因此为了防止ic_a的位宽溢出, 需要
# ic.max =  $2^{16} - 64 = 65472$ 
```

3.3.3 Shape 限制3

`transpose_conv`, `transpose_conv_add`, `output_quantize_transpose_conv` 需要满足此限制。

```
# constraint!!
output_feature_map_h + Fh + (Fh - 1)(Dh - 1) <= 31238
output_feature_map_d + Fd + (Fd - 1)(Dd - 1) <= 31238
output_feature_map_w + Fw + (Fw - 1)(Dw - 1) <= 8387078
input_feature_map_n <= 65535
input_feature_map_c <= 65472

# constraint来源
# input_feature_map_h * conv_stride_h <=
# 32768 - output_padding.h - 254(max pooling padding)*2
```

`upsampling` 需要满足此限制。

```
# constraint!!
input_feature_map_h * h_upsample_factor <= 64005
input_feature_map_d * d_upsample_factor <= 64005
input_feature_map_w * w_upsample_factor <= 16775685
input_feature_map_n <= 65535
input_feature_map_c <= 65472

# constraint来源, 比如对于H维度
# input_feature_map_h <=
# 65535 (input rect限制) - 254(max pooling padding)*2 - 511(max pad padding)*2
```