tensor.qlinear_mul

```
Copy fnqlinear_mul(self:@Tensor, a_scale:@Tensor, a_zero_point:@Tensor, b:@Tensor, b_scale:@Tensor, b_zero_point:@Tensor, y_scale:@Tensor, y_zero_point:@Tensor)->Tensor::;
```

Performs the element-wise multiplication of quantized Tensors

It consumes two quantized input tensors, their scales and zero points, scale and zero point of output, and computes the quantized output. The quantization formula is $y = saturate((x / y\_scale) + y\_zero\_point)$. It perfoms the element-wise multiplication of the two vectors once dequantized, then return the quantization of the result of the multiplication. The broadcasting is supported Scale and zero point must have same shape and the same type. They must be either scalar (per tensor) or N-D tensor (per row for 'a' and per column for 'b'). Scalar refers to per tensor quantization whereas N-D refers to per row or per column quantization.

Args

- self
- (@Tensor
-) - The first tensor to be multiplied (a).
- a_scale
- (@Tensor
-) - Scale for inputa
- .
- a_zero_point
- (@Tensor
-) - Zero point for inputa
- .
- b
- (@Tensor
-) - The second tensor to be multiplied
- b_scale
- (@Tensor
-) - Scale for inputb
- .
- b_zero_point
- (@Tensor
-) - Zero point for inputb
- .
- y_scale
- (@Tensor
-) - Scale for outut.
- y_zero_point
- (@Tensor
-) - Zero point for output.
-

Returns

A newTensor , containing the quantized result of the element-wise multiplication of the dequantized inputs.

Type Constraints

u32 tensor, not supported. fp8x23wide tensor, not supported. fp16x16wide tensor, not supported.

Example

use core::array::{ArrayTrait, SpanTrait};

use orion::operators::tensor::{TensorTrait, Tensor, I8Tensor, FP16x16Tensor}; use orion::numbers::{FP16x16, FP16x16Impl, FixedTrait};

```

Copy

# [test]

# [available_gas(200000000000)]

fnqlinear_mul_example()->Tensor{ leta=TensorTrait::< i8

> ::new( shape:array![2,3].span(), data:array![21,21,21,41,41,41] .span(), ); letb=TensorTrait::< i8 ::new( shape:array![1,3].span(), data:array![4,8,12].span(), );

leta_scale=TensorTrait::< FP16x16

> ::new( shape:array![1].span(), data:array![FixedTrait::::new(131072,false)].span(), ); leta_zero_point=TensorTrait::< FP16x16 ::new(shape:array![1].span(), data:array! [FixedTrait::::new(65536,false)].span(),); letb_scale=TensorTrait::< FP16x16 ::new(shape:array![1].span(), data:array![FixedTrait::::new(16384,false)].span(),); letb_zero_point=TensorTrait::< FP16x16 ::new(shape:array! [1].span(), data:array![FixedTrait::::new(0,false)].span(),);

lety_scale=TensorTrait::< FP16x16

> ::new( shape:array![1].span(), data:array![FixedTrait::::new(393216,false)].span(), ); lety_zero_point=TensorTrait:: < FP16x16 ::new( shape:array![1].span(), data:array![FixedTrait::::new(655360,false)].span(), );

return=a .qlinear_mul( @a_scale,@a_zero_point,@b,@b_scale,@b_zero_point,@y_scale,@y_zero_point );

}

> [[16,23,30], [23,36,50]]

```

Last updated1 month ago