

tensor.dequantize_linear

```
...  
  
Copy fndequantize_linear(self:@Tensor", x_scale:@Tensor, x_zero_point:@Tensor)->Tensor::;"  
...
```

Dequantizes a Tensor using linear dequantization.

The linear dequantization operator. It consumes a quantized tensor, a scale, and a zero point to compute the full precision tensor. The dequantization formula is $y = (x - x_zero_point) * x_scale$. `x_scale` and `x_zero_point` must have same shape, and can be either a scalar for per-tensor / per layer quantization, or a 1-D tensor for per-axis quantization.

Args

- `self`
- (`@Tensor`
-) - The input tensor.
- `x_scale`
- (`@Tensor`
-) - Scale for inputx
- .
- `x_zero_point`
- (`@Tensor`
-) - Zero point for inputx
- .
- .

Returns

A newTensor with the same shape as the input tensor, containing the dequantized values.

Type Constraints

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

Examples

```
...  
  
Copy usecore::array::{ArrayTrait,SpanTrait};  
useorion::operators::tensor::{TensorTrait,Tensor,l8Tensor,l32Tensor};  
  
fndequantize_linear_example()->Tensor { // We instantiate a 1D Tensor here. letx=TensorTrait::new( shape:array!  
[4].span(), data:array![0,3,125,127].span(), );  
  
// We instantiate the x_scale here. letx_scale=TensorTrait::new( shape:array![1].span(), data:array![2].span(), );  
  
// We instantiate the x_zero_point here. letx_zero_point=TensorTrait::new( shape:array![1].span(), data:array![0].span(), );  
  
returnx.dequantize_linear(@x_scale,@x_zero_point); }  
  
[0,6,250,254]  
...
```

[Previous tensor.quantize_linear](#) [Next tensor.qlinear_add](#)

Last updated1 month ago