

tensor.quantize_linear

```
...  
  
Copy fnquantize_linear(self:@Tensor, y_scale:@Tensor, y_zero_point:@Tensor)->Tensor::;"  
...
```

Quantizes a Tensor using linear quantization.

The linear quantization operator. It consumes a high precision tensor, a scale, and a zero point to compute the low precision / quantized tensor. The scale factor and 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. The quantization formula is $y = \text{saturnate}((x / y_scale) + y_zero_point)$. For saturation, it saturates to $[-128, 127]$. For (x / y_scale) , it's rounding to the nearest even.

Args

- self
- (@Tensor
-) - The input tensor.
- y_scale
- (@Tensor
-) - Scale for doing quantization to get y
- .
- y_zero_point
- (@Tensor
-) - Zero point for doing quantization to get y
- .
- .

Returns

A new Tensor with the same shape as the input tensor, containing the quantized values.

Type Constraints

u32 tensor, not supported.

Examples

```
...  
  
Copy usecore::array::{ArrayTrait,SpanTrait};  
use orion::operators::tensor::{TensorTrait,Tensor,I8Tensor,I32Tensor};  
  
fn quantize_linear_example()->Tensor { // We instantiate a 1D Tensor here. let x=TensorTrait::new( shape:array![6].span(),  
data:array![0,2,3,1,-254,-1000].span(), );  
  
// We instantiate the y_scale here. let y_scale=TensorTrait::new( shape:array![1].span(), data:array![2].span(), );  
  
// We instantiate the y_zero_point here. let y_zero_point=TensorTrait::new( shape:array![1].span(), data:array![1].span(), );  
  
return x.quantize_linear(@y_scale,@y_zero_point); }  
  
[1,2,2,127,-126,-128]  
...
```

[Previous tensor.gather](#) [Next tensor.dequantize_linear](#)

Last updated 1 month ago