tensor.resize

tensor.resize

```

Copy fnresize( self:@Tensor, roi:Option>, scales:Option>, sizes:Option>, antialias:Option, axes:Option>, coordinate_transformation_mode:Option, cubic_coeff_a:Option, exclude_outside:Option, extrapolation_value:Option, keep_aspect_ratio_policy:Option, mode:Option, nearest_mode:Option, )->Tensor;

```

Resizes the input tensor. In general, it calculates every value in the output tensor as a weighted average of neighborhood in the input tensor.

Args

- self
- (@Tensor
-) - The input tensor.
- roi
- (Option>
-) (optional) - 1-D tensor given as [start1, ..., startN, end1, ..., endN], where N is the rank of X or the length of axes, if provided. It only takes effect when coordinate_transformation_mode is "tf_crop_and_resize"
- scales
- (Option>
-) (optional) - The scale array along each dimension. It takes value greater than 0. If it's less than 1, it's sampling down, otherwise, it's upsampling. The number of elements of 'scales' should be the same as the rank of input 'X' or the length of 'axes', if provided. One and only one of 'scales' and 'sizes' MUST be specified.
- sizes
- (Option>
-) (optional) - Target size of the output tensor. Its interpretation depends on the 'keep_aspect_ratio_policy' value. The number of elements of 'sizes' should be the same as the rank of input 'X', or the length of 'axes', if provided. One and only one of 'scales' and 'sizes' MUST be specified.
- antialias
- (Option
-) (default is 0) - If set to 1, "linear" and "cubic" interpolation modes will use an antialiasing filter when downscaling. Antialiasing is achieved by stretching the resampling filter by a factor max(1, 1 / scale).
- axes
- (Option>
-) - If provided, it specifies a subset of axes that 'roi', 'scales' and 'sizes' refer to. If not provided, all axes are assumed [0, 1, ..., r-1], where r = rank(data).
- coordinate_transformation_mode
- (Option
-) (default is half_pixel) - This attribute describes how to transform the coordinate in the resized tensor to the coordinate in the original tensor.
- cubic_coeff_a
- (Option
-) (default is -0.75) - The coefficient 'a' used in cubic interpolation.
- exclude_outside
- (Option
-) (default is false) - If set to true, the weight of sampling locations outside the tensor will be set to 0 and the weight will be renormalized so that their sum is 1.0.
- extrapolation_value
- (Option
-) (default is 0.0) - When coordinate_transformation_mode is "tf_crop_and_resize" and x_original is outside the range [0, length_original - 1], this value is used as the corresponding output value.
- keep_aspect_ratio_policy
- (Option
-) (default is stretch) - This attribute describes how to interpret thesizes
- input with regard to keeping the original aspect ratio of the input, and it is not applicable when thescales
- input is used.
- mode
- (Option
-) (default is nearest) - Three interpolation modes: "nearest", "linear" and "cubic".
- nearest_mode
- (Option

-) (default is round_prefer_floor) - Four modes: "round_prefer_floor" (as known as round half down), "round_prefer_ceil" (as known as round half up), "floor", "ceil". Only used by nearest interpolation.
-

Panics

- Panics if both scales and sizes areOption::None
- .
- Panics if roi isOption::None
- for the coordinate_transformation_modetf_crop_and_resize
- .
- Panics if antialias is notOption::None
- for modenearest
- .
-

Returns

A new resizedTensor of the dimension given by output_dimension = floor(input_dimension * (roi_end - roi_start) * scale) is scale is specified, or output_size if size is specified (note that some value of the parameterkeep_aspect_ratio_policy can change sizes and therefore the dimension of the output tensor)

Example

```

Copy usecore::array::{ArrayTrait,SpanTrait}; useorion::operators::tensor::
{TensorTrait,Tensor,FP16x16Tensor,FP16x16TensorPartialEq}; useorion::operators::tensor::math::resize::{
MODE,NEAREST_MODE,KEEP_ASPECT_RATIO_POLICY,TRANSFORMATION_MODE }; useorion::numbers::
{FP16x16,FP16x16Impl,FixedTrait}; usecore::debug::PrintTrait;

fnexample_resize_downsample_scales_linear()->Tensor{ letmutdata=TensorTrait::< FP16x16

    ::new( shape:array![1,1,2,4].span(), data:array![ FixedTrait::::new(65536,false),//1
    FixedTrait::::new(131072,false),//2 FixedTrait::::new(196608,false),//3 FixedTrait::::new(262144,false),//4
    FixedTrait::::new(327680,false),//5 FixedTrait::::new(393216,false),//6 FixedTrait::::new(458752,false),//7
    FixedTrait::::new(524288,false),//8 ] .span(), ); letmutscales=array![ FixedTrait::::new(65536,false),//1
    FixedTrait::::new(65536,false), FixedTrait::::new(39322,false),//0.6 FixedTrait::::new(39322,false) ] .span();

letscales=Option::Some(scales);

returndata.resize( Option::None, scales, Option::None, Option::None, Option::None, Option::None, Option::None,
Option::None, Option::None, Option::None, Option::Some(MODE::LINEAR), Option::None, );

}

                [[[[2.66666654.3333331]]]]

fnexample_resize_tf_crop_and_resize_extrapolation_value()->Tensor { letmutdata=TensorTrait::< FP16x16

    ::new( shape:array![1,1,4,4].span(), data:array![ FixedTrait::::new(65536,false), FixedTrait::::new(131072,false),
    FixedTrait::::new(196608,false), FixedTrait::::new(262144,false), FixedTrait::::new(327680,false),
    FixedTrait::::new(393216,false), FixedTrait::::new(458752,false), FixedTrait::::new(524288,false),
    FixedTrait::::new(589824,false), FixedTrait::::new(655360,false), FixedTrait::::new(720896,false),
    FixedTrait::::new(786432,false), FixedTrait::::new(851968,false), FixedTrait::::new(917504,false),
    FixedTrait::::new(983040,false), FixedTrait::::new(1048576,false), ] .span(), );

letmutroi=TensorTrait::< FP16x16

    ::new( shape:array![8].span(), data:array![ FixedTrait::::new(0,false), FixedTrait::::new(0,false),
    FixedTrait::::new(26214,false), FixedTrait::::new(39322,false), FixedTrait::::new(65536,false),
    FixedTrait::::new(65536,false), FixedTrait::::new(78643,false), FixedTrait::::new(111411,false), ] .span(), );
    letroi=Option::Some(roi);

letmutsizes=array![1,1,3,3].span(); letsizes=Option::Some(sizes);

letextrapolation_value=Option::Some(FixedTrait::::new(655360,false));

returndata.resize( roi, Option::None, sizes, Option::None, Option::None,
Option::Some(TRANSFORMATION_MODE::TF_CROP_AND_RESIZE), Option::None, Option::None, extrapolation_value,
Option::None, Option::Some(MODE::LINEAR), Option::None, );
```

}

[[[[7.600000410.10.] [12.40000110.10.] [10.10.10.]]]]

fnexample_resize_downsample_sizes_cubic_antialias()->Tensor { letmutdata=TensorTrait::< FP16x16

::new(shape:array![1,1,4,4].span(), data:array![FixedTrait:::new(65536,false), FixedTrait::::new(131072,false), FixedTrait::::new(196608,false), FixedTrait::::new(262144,false), FixedTrait::::new(327680,false), FixedTrait::::new(393216,false), FixedTrait::::new(458752,false), FixedTrait::::new(524288,false), FixedTrait::::new(589824,false), FixedTrait::::new(655360,false), FixedTrait::::new(720896,false), FixedTrait::::new(786432,false), FixedTrait::::new(851968,false), FixedTrait::::new(917504,false), FixedTrait::::new(983040,false), FixedTrait::::new(1048576,false),] .span(),);

letantialias=Option::Some(1);

letmutsizes=array![1,1,3,3].span(); letsizes=Option::Some(sizes);

returndata.resize(Option::None, Option::None, sizes, antialias, Option::None, Option::None, Option::None, Option::None, Option::None, Option::None, Option::Some(MODE::CUBIC), Option::None,); }

[[[[1.77500923.12000734.4650054] [7.15500168.59.844998]
[12.53499413.879992515.224991]]]]

```

Previous tensor.bitwise_or Next tensor.round

Last updated2 months ago
```