nn.depth_to_space

```

Copy fndepth_to_space(tensor:@Tensor, blocksize:usize)->Tensor;

```

DepthToSpace rearranges (permutes) data from depth into blocks of spatial data. This is the reverse transformation of SpaceToDepth. More specifically, this op outputs a copy of the input tensor where values from the depth dimension are moved in spatial blocks to the height and width dimensions. By default, mode = DCR. In the DCR mode, elements along the depth dimension from the input tensor are rearranged in the following order: depth, column, and then row.

Args

- tensor
- (@Tensor
-) - The input tensor of [N,C,H,W], where N is the batch axis, C is the channel or depth, H is the height and W is the width.
- blocksize
- (usize
-) - The size of the blocks to move along [blocksize, blocksize].
- mode
- (felt252) - DCR (default) for depth-column-row order re-arrangement. Use CRD for column-row-depth order.
-

Returns

ATensor of [N, C/(blocksize * blocksize), H * blocksize, W * blocksize].

Examples

```

Copy usecore::array::{ArrayTrait,SpanTrait}; useorion::operators::tensor::{TensorTrait,Tensor}; useorion::operators::tensor::{I8Tensor,I8TensorAdd}; useorion::numbers::NumberTrait; useorion::operators::nn::NNTrait; useorion::operators::nn::I8NN; useorion::numbers::FixedTrait;

fndepth_to_space_example()->Tensor { letmutshape=ArrayTrait:::new(); shape.append(1); shape.append(4); shape.append(2); shape.append(2);

letmutdata=ArrayTrait::new(); data.append(-2); data.append(0); data.append(-1); data.append(0); data.append(0); data.append(-3); data.append(2); data.append(1); data.append(-2); data.append(-2); data.append(0); data.append(-2); data.append(-1); data.append(-1); data.append(2); data.append(2); lettensor=TensorTrait::new(shape.span(), data.span()); returnNNTrait::depth_to_space(@tensor,2,'DCR'); }

                    [[[[-2,0,0,-3], [-2,-1,-2,-1], [-1,2,0,1], [0,2,-2,2]]]]

```

Previous nn.conv Next nn.space_to_depth