nn.conv_transpose

```

Copy conv_transpose( X:@Tensor, W:@Tensor, B:Option<@Tensor>, auto_pad:Option, dilations:Option>, group:Option, kernel_shape:Option>, output_padding:Option>, output_shape:Option>, pads:Option>, strides:Option>, )->Tensor

```

The convolution transpose operator consumes an input tensor and a input weight tensor, and computes the output.

Args

- X
- (@Tensor
-) - Input data tensor, has size (N x C x H x W), where N is the batch size, C is the number of channels, and H and W if 2D, otherwise the size is (N x C x D1 x D2 ... x Dn).
- W
- (@Tensor
-) - The weight tensor, has size (C x M/group x kH x kW), where C is the number of channels, and kH and kW are the height and width of the kernel, and M is the number of feature maps if 2D, for more than 2 dimensions, the weight shape will be (C x M/group x k1 x k2 x ... x kn).
- B
- (Option<@Tensor>
-) - Optional 1D bias to be added to the convolution, has size of M.
- auto_pad
- (Option
-) - Default is NOTSET, auto_pad must be either NOTSET, SAME_UPPER, SAME_LOWER or VALID. NOTSET means explicit padding is used. SAME_UPPER or SAME_LOWER mean pad the input so thatoutput_shape[i] = input_shape[i] * strides[i]
- for each axisi
- .
- dilations
- (Option>
-) - Dilation value along each spatial axis of the filter. If not present, the dilation defaults to 1 along each spatial axis.
- group
- (Option
-) - Default is 1, number of groups input channels and output channels are divided into.
- kernel_shape
- (Option>
-) - The shape of the convolution kernel. If not present, should be inferred from input W.
- output_padding
- (Option>
-) - Additional elements added to the side with higher coordinate indices in the output. Each padding value in "output_padding" must be less than the corresponding stride/dilation dimension. By default, this attribute is a zero vector.
- output_shape
- (Option>
-) - The shape of the output can be explicitly set which will cause pads values to be auto generated. If output_shape is specified pads values are ignored. See doc for details for equations to generate pads.
- pads
- (Option>
-) - Padding for the beginning and ending along each spatial axis, it can take any value greater than or equal to 0. The value represent the number of pixels added to the beginning and end part of the corresponding axis.pads
- format should be as follow [x1_begin, x2_begin...x1_end, x2_end,...], where xi_begin the number of pixels added at the beginning of axisi
- and xi_end, the number of pixels added at the end of axisi
- . This attribute cannot be used simultaneously with auto_pad attribute. If not present, the padding defaults to 0 along start and end of each spatial axis.
- strides
- (Option>
-) - Stride along each spatial axis. If not present, the stride defaults to 1 along each spatial axis.
-

Returns

ATensor that contains the result of the convolution transpose.

Examples

```

Copy useorion::operators::nn::NNTrait; useorion::numbers::FixedTrait; useorion::operators::nn::FP16x16NN; useorion::numbers::FP16x16; useorion::operators::tensor::{Tensor,TensorTrait,FP16x16Tensor};

fnexample_conv_transpose()->Tensor { letmutshape=ArrayTrait:::new(); shape.append(1); shape.append(2); shape.append(3); shape.append(3);

letmutdata=ArrayTrait::new(); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); letW=TensorTrait::new(shape.span(), data.span());

letmutshape=ArrayTrait:::new(); shape.append(1); shape.append(1); shape.append(3); shape.append(3);

letmutdata=ArrayTrait::new(); data.append(FP16x16{ mag:0, sign:false}); data.append(FP16x16{ mag:65536, sign:false}); data.append(FP16x16{ mag:131072, sign:false}); data.append(FP16x16{ mag:196608, sign:false}); data.append(FP16x16{ mag:262144, sign:false}); data.append(FP16x16{ mag:327680, sign:false}); data.append(FP16x16{ mag:393216, sign:false}); data.append(FP16x16{ mag:458752, sign:false}); data.append(FP16x16{ mag:524288, sign:false}); letmutX=TensorTrait::new(shape.span(), data.span());

returnNNTrait::conv_transpose( @X, @W, Option::None, Option::None, Option::None, Option::None, Option::None, Option::None, Option::None, Option::None, Option::None, );

}

        [ [ [ [0.0,1.0,3.0,3.0,2.0], [3.0,8.0,15.0,12.0,7.0], [9.0,21.0,36.0,27.0,15.0],
        [9.0,20.0,33.0,24.0,13.0], [6.0,13.0,21.0,15.0,8.0], ], [ [0.0,1.0,3.0,3.0,2.0],
        [3.0,8.0,15.0,12.0,7.0], [9.0,21.0,36.0,27.0,15.0], [9.0,20.0,33.0,24.0,13.0],
        [6.0,13.0,21.0,15.0,8.0], ], ] ]

```

Previous nn.col2im Next nn.conv

Last updated15 days ago