Table of Contents

## CONV2D

CLASS torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding\_mode='zeros', device=None, dtype=None) [SOURCE]

 $\mathscr{S}$ 

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size  $(N, C_{
m in}, H, W)$  and output  $(N, C_{
m out}, H_{
m out}, W_{
m out})$  can be precisely described as:

$$\operatorname{out}(N_i, C_{\operatorname{out}_j}) = \operatorname{bias}(C_{\operatorname{out}_j}) + \sum_{k=0}^{C_{\operatorname{in}}-1} \operatorname{weight}(C_{\operatorname{out}_j}, k) \star \operatorname{input}(N_i, k)$$

where  $\star$  is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

This module supports TensorFloat32.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

- stride controls the stride for the cross-correlation, a single number or a tuple.
- padding controls the amount of padding applied to the input. It can be either a string {'valid', 'same'} or an int / a tuple of ints giving the amount of implicit padding applied on both sides.
- dilation controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this link has a nice visualization of what dilation does.
- groups controls the connections between inputs and outputs. in\_channels and out\_channels must both be divisible by groups. For example,
  - At groups=1, all inputs are convolved to all outputs.
  - At groups=2, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels and producing half the output channels, and both subsequently concatenated.
  - $\bullet \quad \text{At groups= in\_channels, each input channel is convolved with its own set of filters (of size } \frac{\text{out\_channels}}{\text{in\_channels}}).$

The parameters kernel\_size, stride, padding, dilation can either be:

- a single int in which case the same value is used for the height and width dimension
- a tuple of two ints in which case, the first *int* is used for the height dimension, and the second *int* for the width dimension

• NOTE

When  $groups == in\_channels$  and  $out\_channels == K*in\_channels$ , where K is a positive integer, this operation is also known as a "depthwise convolution".

In other words, for an input of size  $(N, C_{in}, L_{in})$ , a depthwise convolution with a depthwise multiplier K can be performed with the arguments  $(C_{\rm in} = C_{\rm in}, C_{\rm out} = C_{\rm in} imes K, ..., {
m groups} = C_{
m in})$ .

• NOTE

In some circumstances when given tensors on a CUDA device and using CuDNN, this operator may select a nondeterministic algorithm to increase performance. If this is undesirable, you can try to make the operation deterministic (potentially at a performance cost) by setting torch.backends.cudnn.deterministic = True See Reproducibility for more information.

NOTE

padding='valid' is the same as no padding. padding='same' pads the input so the output has the shape as the input. However, this mode doesn't support any stride values other than 1.

• NOTE

This module supports complex data types i.e. complex32, complex64, complex128.

# Parameters:

- in\_channels (int) Number of channels in the input image
- a aut channels (int) Number of channels produced by the convolution

To analyze traffic and optimize your experience, we serve cookies on this site. By clicking or navigating, you agree to allow our usage of cookies. As the current maintainers of this site, Facebook's Cookies Policy applies. Learn more, including about available controls: Cookies Policy.

- dilation (int or tuple, optional) Spacing between kernel elements. Default: 1
- groups (int, optional) Number of blocked connections from input channels to output channels. Default: 1
- bias (bool, optional) If True, adds a learnable bias to the output. Default: True

### Shape:

- Input:  $(N, C_{in}, H_{in}, W_{in})$  or  $(C_{in}, H_{in}, W_{in})$
- ullet Output:  $(N, C_{out}, H_{out}, W_{out})$  or  $(C_{out}, H_{out}, W_{out})$ , where

$$H_{out} = igg \lfloor rac{H_{in} + 2 imes ext{padding}[0] - ext{dilation}[0] imes ( ext{kernel\_size}[0] - 1) - 1}{ ext{stride}[0]} + 1 igg 
floor$$
 $W_{out} = igg \lfloor rac{W_{in} + 2 imes ext{padding}[1] - ext{dilation}[1] imes ( ext{kernel\_size}[1] - 1) - 1}{ ext{stride}[1]} + 1 igg 
floor$ 

#### Variables:

- weight (*Tensor*) the learnable weights of the module of shape (out\_channels,  $\frac{\text{in\_channels}}{\text{groups}}$ ,  $\text{kernel\_size}[0]$ ,  $\text{kernel\_size}[1]$ ). The values of these weights are sampled from  $\mathcal{U}(-\sqrt{k},\sqrt{k})$  where  $k=\frac{groups}{C_{\text{in}}*\prod_{i=0}^{1}\text{kernel\_size}[i]}$
- **bias** (*Tensor*) the learnable bias of the module of shape (out\_channels). If bias is True, then the values of these weights are sampled from  $\mathcal{U}(-\sqrt{k},\sqrt{k})$  where  $k=rac{groups}{C_{ ext{in}}*\prod_{i=0}^1 \text{kernel\_size}[i]}$

#### **Examples**

```
>>> # With square kernels and equal stride
>>> m = nn.Conv2d(16, 33, 3, stride=2)
>>> # non-square kernels and unequal stride and with padding
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2))
>>> # non-square kernels and unequal stride and with padding and dilation
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2), dilation=(3, 1))
>>> input = torch.randn(20, 16, 50, 100)
>>> output = m(input)
```

✓ Previous

Next >

© Copyright 2023, PyTorch Contributors.

Built with Sphinx using a theme provided by Read the Docs.

Docs

Access comprehensive developer documentation for PyTorch

View Docs

Tutorials

Get in-depth tutorials for beginners and advanced developers

View Tutorials

Resources
Find development resources and get your questions
answered
View Resources

PyTorch Resources Stay up to date PyTorch Podcasts

Get Started **Tutorials** Facebook Spotify Features Docs **Twitter** Apple Discuss Ecosystem YouTube Google Blog Github Issues LinkedIn Amazon

Contributing Brand Guidelines

Terms | Privacy

© Copyright The Linux Foundation. The PyTorch Foundation is a project of The Linux Foundation. For web site terms of use, trademark policy and other policies applicable to The PyTorch Foundation please see www.linuxfoundation.org/policies/. The PyTorch Foundation supports the PyTorch open source project, which has been established as PyTorch Project a Series of LF Projects, LLC. For policies applicable to the PyTorch Project a Series of LF Projects, LLC, please see www.lfprojects.org/policies/.

To analyze traffic and optimize your experience, we serve cookies on this site. By clicking or navigating, you agree to allow our usage of cookies. As the current maintainers of this site, Facebook's Cookies Policy applies. Learn more, including about available controls: Cookies Policy.