O PyTorch

1.11.0 ▼

Q Search Docs

Notes [+]

Language Bindings [+]

Python API [-]

torch

torch.nn

torch.nn.functional torch.Tensor

Tensor Attributes Tensor Views

torch.autograd

torch.cuda

torch.cuda.amp torch.backends

torch.distributed

torch.distributed.algorithms.join torch.distributed.elastic

torch.distributed.fsdp torch.distributed.optim

torch.distributions torch.fft

torch.futures torch.fx

torch.jit torch.linalg torch.monitor torch.special torch.overrides

torch.hub

torch.package torch.profiler

tarch no init

Docs > torch.nn > Conv2d

Get Started

**Tutorials** 

Docs 🗸

GitHub

>\_

Resources ~

Blog

Mobile

### 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]

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.

Ecosystem

- 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 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.
  - 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  $(C_{
m in}=C_{
m in}, C_{
m out}=C_{
m in} imes {
m K},...,{
m groups}=C_{
m in})$  .

### • 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.

## Parameters

- in\_channels (int) Number of channels in the input image
- out\_channels (int) Number of channels produced by the convolution
- **kernel\_size** (*int or tuple*) Size of the convolving kernel
- **stride** (*int or tuple*, *optional*) Stride of the convolution. Default: 1
- padding (int, tuple or str, optional) Padding added to all four sides of the input. Default: 0
- padding\_mode (string, optional) 'zeros', 'reflect', 'replicate' or 'circular'. Default: 'zeros'
- 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

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

# Variables

- **~Conv2d.weight** (*Tensor*) the learnable weights of the module of shape (out\_channels,  $\frac{\text{in\_channels}}{\text{groups}}$ ,  $kernel\_size[0], kernel\_size[1]).$  The values of these weights are sampled from  $\mathcal{U}(-\sqrt{k},\sqrt{k})$  where
- **~Conv2d.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  $k=rac{groups}{C_{ ext{in}}*\prod_{i=0}^{1} ext{kernel\_size}[i]}$

# 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 2019, Torch Contributors. Built with Sphinx using a theme provided by Read the Docs.

**Tutorials** 

Resources

Access comprehensive developer documentation for PyTorch

View Docs >

Docs

Get in-depth tutorials for beginners and advanced developers

Find development resources and get your questions answered

View Tutorials >

View Resources >

Get Started Features

Ecosystem

Contributing

Blog

PyTorch

Tutorials Docs Discuss Github Issues

**Brand Guidelines** 

Resources

Email Address



Stay Connected