tf.keras.layers.Conv2D

TensorFlow 1 version (/versions/r1.15/api_docs/python/tf/keras/layers/Conv2D)

View source on GitHub (https://github.com/tensorflow/tensorflo L678)

2D convolution layer (e.g. spatial convolution over images).

Inherits From: `Layer` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer), `Module` (https://www.tensorflow.org/api_docs/python/tf/Module)

View aliases

Main aliases

`tf.keras.layers.Convolution2D` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D)

Compat aliases for migration

See Migration guide (https://www.tensorflow.org/guide/migrate) for more details.

`tf.compat.v1.keras.layers.Conv2D` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D), `tf.compat.v1.keras.layers.Convolution2D` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D)

```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid',
    data_format=None, dilation_rate=(1, 1), groups=1, activation=None,
    use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None,
    bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,
    bias_constraint=None, **kwargs
)
```

Used in the notebooks

Used in the guide	Used in the tutorials

Used in the guide	Used in the tutorials
• The Functional API (https://www.tensorflow.org/guide/keras/functional) • The Sequential model (https://www.tensorflow.org/guide/keras/sequential_model) • Migrate your TensorFlow 1 code to TensorFlow 2 (https://www.tensorflow.org/guide/migrate) • Eager execution (https://www.tensorflow.org/guide/eager) • Customize what happens in Model.fit (https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit)	• Custom layers (https://www.tensorflow.org/tutorials/custo • Image classification (https://www.tensorflow.org/tutorials/image • Data augmentation (https://www.tensorflow.org/tutorials/image • Intro to Autoencoders (https://www.tensorflow.org/tutorials/gener • Pix2Pix (https://www.tensorflow.org/tutorials

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If **`use_bias`** is True, a bias vector is created and added to the outputs. Finally, if **`activation`** is not **`None`**, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument **`input_shape`** (tuple of integers or **`None`**, does not include the sample axis), e.g. **`input_shape=(128, 128, 3)`** for 128x128 RGB pictures in **`data_format="channels_last"`**. You can use **`None`** when a dimension has variable size.

Examples:

```
>>> # The inputs are 28x28 RGB images with `channels_last` and the batch
>>> # size is 4.
>>> input_shape = (4, 28, 28, 3)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv2D(
... 2, 3, activation='relu', input_shape=input_shape[1:])(x)
>>> print(y.shape)
(4, 26, 26, 2)
```

```
>>> # With `dilation_rate` as 2.
>>> input_shape = (4, 28, 28, 3)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv2D(
... 2, 3, activation='relu', dilation_rate=2, input_shape=input_shape[1:])(x)
>>> print(y.shape)
(4, 24, 24, 2)
```

```
>>> # With `padding` as "same".
>>> input_shape = (4, 28, 28, 3)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv2D(
... 2, 3, activation='relu', padding="same", input_shape=input_shape[1:])(x)
```

```
>>> print(y.shape)
(4, 28, 28, 2)
```

```
>>> # With extended batch shape [4, 7]:
>>> input_shape = (4, 7, 28, 28, 3)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv2D(
... 2, 3, activation='relu', input_shape=input_shape[2:])(x)
>>> print(y.shape)
(4, 7, 26, 26, 2)
```

Args	
`filters`	Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
`kernel_size`	An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
`strides`	An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any **`dilation_rate`** value != 1.
`padding`	one of **`"valid"`** or **`"same"`** (case-insensitive). **`"valid"`** means no padding. **`"same"`** results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
`data_format`	A string, one of **`channels_last`** (default) or **`channels_first`**. The ordering of the dimensions in the inputs. **`channels_last`** corresponds to inputs with shape **`(batch_size, height, width, channels)`** while **`channels_first`** corresponds to inputs with shape **`(batch_size, channels, height, width)`**. It defaults to the **`image_data_format`** value found in your Keras config file at **`~/.keras/keras.json`**. If you never set it, then it will be **`channels_last`**.
`dilation_rate`	an integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any **`dilation_rate`** value != 1 is incompatible with specifying any stride value != 1.
`groups`	A positive integer specifying the number of groups in which the input is split along the channel axis. Each group is convolved separately with **`filters / groups`** filters. The output is the concatenation of all the **`groups`** results along the channel axis. Input channels and **`filters`** must both be divisible by **`groups`**.
`activation`	Activation function to use. If you don't specify anything, no activation is applied (see **`keras.activations`** (https://www.tensorflow.org/api_docs/python/tf/keras/activations)).
`use_bias`	Boolean, whether the layer uses a bias vector.
`kernel_initializer`	Initializer for the **`kernel`** weights matrix (see **`keras.initializers`** (https://www.tensorflow.org/api_docs/python/tf/keras/initializers)). Defaults to 'glorot_uniform'.

`bias_initializer`	Initializer for the bias vector (see `keras.initializers` (https://www.tensorflow.org/api_docs/python/tf/keras/initializers)). Defaults to 'zeros'.
`kernel_regularizer`	Regularizer function applied to the `kernel` weights matrix (see `keras.regularizers` (https://www.tensorflow.org/api_docs/python/tf/keras/regularizers)).
`bias_regularizer`	Regularizer function applied to the bias vector (see `keras.regularizers` (https://www.tensorflow.org/api_docs/python/tf/keras/regularizers)).
`activity_regularizer`	Regularizer function applied to the output of the layer (its "activation") (see `keras.regularizers` (https://www.tensorflow.org/api_docs/python/tf/keras/regularizers)).
`kernel_constraint`	Constraint function applied to the kernel matrix (see `keras.constraints` (https://www.tensorflow.org/api_docs/python/tf/keras/constraints)).
`bias_constraint`	Constraint function applied to the bias vector (see `keras.constraints` (https://www.tensorflow.org/api_docs/python/tf/keras/constraints)).

Input shape:

4+D tensor with shape: `batch_shape + (channels, rows, cols)` if `data_format='channels_first'` or 4+D tensor with shape: `batch_shape + (rows, cols, channels)` if `data_format='channels_last'`.

Output shape:

4+D tensor with shape: `batch_shape + (filters, new_rows, new_cols)` if `data_format='channels_first'` or 4+D tensor with shape: `batch_shape + (new_rows, new_cols, filters)` if `data_format='channels_last'`. `rows` and `cols` values might have changed due to padding.

Returns

A tensor of rank 4+ representing `activation(conv2d(inputs, kernel) + bias)`.

Raises

`ValueError`	if `padding` is `"causal"`.
`ValueError`	when both `strides > 1` and `dilation_rate > 1`.