Conv3D layer

Conv3D class

```
tf.keras.layers.Conv3D(
    filters,
    kernel_size,
    strides=(1, 1, 1),
    padding="valid",
    data_format=None,
    dilation_rate=(1, 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
)
```

3D convolution layer (e.g. spatial convolution over volumes).

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, 128, 1)` for 128x128x128 volumes with a single channel, in `data_format="channels_last"`.

Examples

```
>>> # The inputs are 28x28x28 volumes with a single channel, and the
>>> # batch size is 4
>>> input_shape =(4, 28, 28, 28, 1)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv3D(
... 2, 3, activation='relu', input_shape=input_shape[1:])(x)
>>> print(y.shape)
(4, 26, 26, 26, 2)
```

```
>>> # With extended batch shape [4, 7], e.g. a batch of 4 videos of 3D frames,
>>> # with 7 frames per video.
>>> input_shape = (4, 7, 28, 28, 28, 1)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv3D(
... 2, 3, activation='relu', input_shape=input_shape[2:])(x)
>>> print(y.shape)
(4, 7, 26, 26, 26, 2)
```

Arguments

- **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 3 integers, specifying the depth, height and width of the 3D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
- **strides**: An integer or tuple/list of 3 integers, specifying the strides of the convolution along each spatial dimension. 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_shape + (spatial_dim1, spatial_dim2, spatial_dim3, channels)` while `channels_first` corresponds to inputs with shape `batch_shape + (channels, spatial_dim1, spatial_dim2, spatial_dim3)`. 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 3 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`).
- **use_bias**: Boolean, whether the layer uses a bias vector.
- **kernel_initializer**: Initializer for the `kernel` weights matrix (see `keras.initializers`). Defaults to 'glorot_uniform'.
- **bias_initializer**: Initializer for the bias vector (see `keras.initializers`). Defaults to 'zeros'.
- **kernel_regularizer**: Regularizer function applied to the `kernel` weights matrix (see `keras.regularizers`).
- **bias_regularizer**: Regularizer function applied to the bias vector (see `keras.regularizers`).
- **activity_regularizer**: Regularizer function applied to the output of the layer (its "activation") (see `keras.regularizers`).
- **kernel_constraint**: Constraint function applied to the kernel matrix (see `keras.constraints`).
- **bias_constraint**: Constraint function applied to the bias vector (see `keras.constraints`).

Input shape

5+D tensor with shape: `batch_shape + (channels, conv_dim1, conv_dim2, conv_dim3)` if data_format='channels_first' or 5+D tensor with shape: `batch_shape + (conv_dim1, conv_dim2, conv_dim3, channels)` if data_format='channels_last'.

Output shape

5+D tensor with shape: `batch_shape + (filters, new_conv_dim1, new_conv_dim2, new_conv_dim3)` if data_format='channels_first' or 5+D tensor with shape: `batch_shape + (new_conv_dim1, new_conv_dim2, new_conv_dim3, filters)` if data_format='channels_last'. `new_conv_dim1`, `new_conv_dim2` and `new_conv_dim3` values might have changed due to padding.

Returns

A tensor of rank 5+ representing `activation(conv3d(inputs, kernel) + bias)`.

Raises

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