tf.nn.convolution

```
convolution(
    input,
    filter,
    padding,
    strides=None,
    dilation_rate=None,
    name=None,
    data_format=None
)
```

Defined in `tensorflow/python/ops/nn_ops.py`.

See the guide: Neural Network > Convolution

Computes sums of N-D convolutions (actually cross-correlation).

This also supports either output striding via the optional `strides` parameter or atrous convolution (also known as convolution with holes or dilated convolution, based on the French word "trous" meaning holes in English) via the optional `dilation_rate` parameter. Currently, however, output striding is not supported for atrous convolutions.

Specifically, in the case that `data_format` does not start with "NC", given a rank (N+2) `input` Tensor of shape

[num_batches, input_spatial_shape[0], ..., input_spatial_shape[N-1], num_input_channels],

a rank (N+2) `filter` Tensor of shape

[spatial_filter_shape[0], ..., spatial_filter_shape[N-1], num_input_channels, num_output_channels],

an optional `dilation_rate` tensor of shape [N] (defaulting to [1]*N) specifying the filter upsampling/input downsampling rate, and an optional list of N* `strides` *(defaulting [1]*N), this computes for each N-D spatial output position (x[0], ..., x[N-1]):

```
  output[b, x[0], ..., x[N-1], k] =
      sum_{z[0], ..., z[N-1], q}
          filter[z[0], ..., z[N-1], q, k] *
          padded_input[b,
                       x[0]*strides[0] + dilation_rate[0]*z[0],
                       ...,
                       x[N-1]*strides[N-1] + dilation_rate[N-1]*z[N-1],
                       q]
```

where b is the index into the batch, k is the output channel number, q is the input channel number, and z is the N-D spatial offset within the filter. Here, `padded_input` is obtained by zero padding the input using an effective spatial filter shape of `(spatial_filter_shape-1) * dilation_rate + 1` and output striding `strides` as described in the comment here.

In the case that `data_format` does start with `"NC"`, the `input` and output (but not the `filter`) are simply transposed as follows:

convolution(input, data_format, **kwargs) = tf.transpose(convolution(tf.transpose(input, [0] + range(2,N+2) + [1]), kwargs), [0, N+1] + range(1, N+1))

It is required that 1 <= N <= 3.

Args:

- `input`: An N-D `Tensor` of type `T`, of shape `[batch_size] + input_spatial_shape + [in_channels]` if data_format does not start with "NC" (default), or `[batch_size, in_channels] + input_spatial_shape` if data_format starts with "NC".
- `filter`: An N-D `Tensor` with the same type as `input` and shape `spatial_filter_shape + [in_channels, out_channels]`.
- `padding`: A string, either `"VALID"` or `"SAME"`. The padding algorithm.
- `strides`: Optional. Sequence of N ints >= 1. Specifies the output stride. Defaults to [1]*N. If any value of strides is > 1, then all values of dilation_rate must be 1.
- `dilation_rate`: Optional. Sequence of N ints >= 1. Specifies the filter upsampling/input downsampling rate. In the literature, the same parameter is sometimes called `input stride` or `dilation`. The effective filter size used for the convolution will be `spatial_filter_shape + (spatial_filter_shape - 1) * (rate - 1)`, obtained by inserting (dilation_rate[i]-1) zeros between consecutive elements of the original filter in each spatial dimension i. If any value of dilation_rate is > 1, then all values of strides must be 1.
- `name`: Optional name for the returned tensor.
- `data_format`: A string or None. Specifies whether the channel dimension of the `input` and output is the last dimension (default, or if `data_format` does not start with "NC"), or the second dimension (if `data_format` starts with "NC"). For N=1, the valid values are "NWC" (default) and "NCW". For N=2, the valid values are "NHWC" (default) and "NCHW". For N=3, the valid values are "NDHWC" (default) and "NCDHW".

Returns:

A `Tensor` with the same type as `input` of shape

```
`[batch_size] + output_spatial_shape + [out_channels]`
```

if data_format is None or does not start with "NC", or

```
`[batch_size, out_channels] + output_spatial_shape`
```

if data_format starts with "NC", where `output_spatial_shape` depends on the value of `padding`.

If padding == "SAME": output_spatial_shape[i] = ceil(input_spatial_shape[i] / strides[i])

If padding == "VALID": output_spatial_shape[i] = ceil((input_spatial_shape[i] - (spatial_filter_shape[i]-1) * dilation_rate[i]) / strides[i]).

Raises:

- `ValueError`: If input/output depth does not match `filter` shape, if padding is other than `"VALID"` or `"SAME"`, or if data_format is invalid.

Last updated November 2, 2017.

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**