# tf.space_to_batch_nd

```
space_to_batch_nd(
    input,
    block_shape,
    paddings,
    name=None
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

SpaceToBatch for N-D tensors of type T.

This operation divides "spatial" dimensions `[1, ..., M]` of the input into a grid of blocks of shape `block_shape`, and interleaves these blocks with the "batch" dimension (0) such that in the output, the spatial dimensions `[1, ..., M]` correspond to the position within the grid, and the batch dimension combines both the position within a spatial block and the original batch position. Prior to division into blocks, the spatial dimensions of the input are optionally zero padded according to `paddings`. See below for a precise description.

## Args:

- `input`: A `Tensor`. N-D with shape `input_shape = [batch] + spatial_shape + remaining_shape`, where spatial_shape has `M` dimensions.
- `block_shape`: A `Tensor`. Must be one of the following types: `int32`, `int64`. 1-D with shape `[M]`, all values must be >= 1.
- `paddings`: A `Tensor`. Must be one of the following types: `int32`, `int64`. 2-D with shape `[M, 2]`, all values must be >= 0. `paddings[i] = [pad_start, pad_end]` specifies the padding for input dimension `i + 1`, which corresponds to spatial dimension `i`. It is required that `block_shape[i]` divides `input_shape[i + 1] + pad_start + pad_end`.

  This operation is equivalent to the following steps:

  1. Zero-pad the start and end of dimensions `[1, ..., M]` of the input according to `paddings` to produce `padded` of shape `padded_shape`.
  2. Reshape `padded` to `reshaped_padded` of shape:

     [batch] + [padded_shape[1] / block_shape[0], block_shape[0], ..., padded_shape[M] / block_shape[M-1], block_shape[M-1]] + remaining_shape
  3. Permute dimensions of `reshaped_padded` to produce `permuted_reshaped_padded` of shape:

     block_shape + [batch] + [padded_shape[1] / block_shape[0], ..., padded_shape[M] / block_shape[M-1]] + remaining_shape
  4. Reshape `permuted_reshaped_padded` to flatten `block_shape` into the batch dimension, producing an output tensor of shape:

     [batch * prod(block_shape)] + [padded_shape[1] / block_shape[0], ..., padded_shape[M] / block_shape[M-1]] + remaining_shape

  Some examples:

  (1) For the following input of shape `[1, 2, 2, 1]`, `block_shape = [2, 2]`, and `paddings = [[0, 0], [0, 0]]`:

`x = [[[[1], [2]], [[3], [4]]]]`

The output tensor has shape `[4, 1, 1, 1]` and value:

`[[[[1]]], [[[2]]], [[[3]]], [[[4]]]]`

(2) For the following input of shape `[1, 2, 2, 3]`, `block_shape = [2, 2]`, and `paddings = [[0, 0], [0, 0]]`:

`x = [[[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]]]`

The output tensor has shape `[4, 1, 1, 3]` and value:

`[[[1, 2, 3]], [[4, 5, 6]], [[7, 8, 9]], [[10, 11, 12]]]`

(3) For the following input of shape `[1, 4, 4, 1]`, `block_shape = [2, 2]`, and `paddings = [[0, 0], [0, 0]]`:

`x = [[[[1], [2], [3], [4]], [[5], [6], [7], [8]], [[9], [10], [11], [12]], [[13], [14], [15], [16]]]]`

The output tensor has shape `[4, 2, 2, 1]` and value:

`x = [[[[1], [3]], [[9], [11]]], [[[2], [4]], [[10], [12]]], [[[5], [7]], [[13], [15]]], [[[6], [8]], [[14], [16]]]]`

(4) For the following input of shape `[2, 2, 4, 1]`, block_shape = `[2, 2]`, and paddings = `[[0, 0], [2, 0]]`:

`x = [[[[1], [2], [3], [4]], [[5], [6], [7], [8]]], [[[9], [10], [11], [12]], [[13], [14], [15], [16]]]]`

The output tensor has shape `[8, 1, 3, 1]` and value:

`x = [[[[0], [1], [3]]], [[[0], [9], [11]]], [[[0], [2], [4]]], [[[0], [10], [12]]], [[[0], [5], [7]]], [[[0], [13], [15]]], [[[0], [6], [8]]], [[[0], [14], [16]]]]`

Among others, this operation is useful for reducing atrous convolution into regular convolution. * `name`: A name for the operation (optional).

### Returns:

A `Tensor`. Has the same type as `input`.



*Last updated November 2, 2017.*