tf.batch_to_space_nd

```
batch_to_space_nd(
    input,
    block_shape,
    crops,
    name=None
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

BatchToSpace for N-D tensors of type T.

This operation reshapes the "batch" dimension 0 into `M + 1` dimensions of shape `block_shape + [batch]`, interleaves these blocks back into the grid defined by the spatial dimensions `[1, ..., M]`, to obtain a result with the same rank as the input. The spatial dimensions of this intermediate result are then optionally cropped according to `crops` to produce the output. This is the reverse of SpaceToBatch. 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.
- `crops`: A `Tensor`. Must be one of the following types: `int32`, `int64`. 2-D with shape `[M, 2]`, all values must be >= 0. `crops[i] = [crop_start, crop_end]` specifies the amount to crop from input dimension `i + 1`, which corresponds to spatial dimension `i`. It is required that `crop_start[i] + crop_end[i] <= block_shape[i] * input_shape[i + 1]`.

  This operation is equivalent to the following steps:

  1. Reshape `input` to `reshaped` of shape: [block_shape[0], ..., block_shape[M-1], batch / prod(block_shape), input_shape[1], ..., input_shape[N-1]]
  2. Permute dimensions of `reshaped` to produce `permuted` of shape [batch / prod(block_shape),

     input_shape[1], block_shape[0], ..., input_shape[M], block_shape[M-1],

     input_shape[M+1], ..., input_shape[N-1]]
  3. Reshape `permuted` to produce `reshaped_permuted` of shape [batch / prod(block_shape),

     input_shape[1] * block_shape[0], ..., input_shape[M] * block_shape[M-1],

     input_shape[M+1], ..., input_shape[N-1]]
  4. Crop the start and end of dimensions `[1, ..., M]` of `reshaped_permuted` according to `crops` to produce the output of shape: [batch / prod(block_shape),

     input_shape[1] * block_shape[0] - crops[0,0] - crops[0,1], ..., input_shape[M] * block_shape[M-1] - crops[M-1,0] - crops[M-1,1],

     input_shape[M+1], ..., input_shape[N-1]]

  Some examples:

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

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

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

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

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

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

The output tensor has shape `[1, 2, 2, 3]` and value:

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

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

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

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

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

(4) For the following input of shape `[8, 1, 3, 1]`, `block_shape = [2, 2]`, and `crops = [[0, 0], [2, 0]]`:

`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]]]]`

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

`x = [[[[1], [2], [3], [4]], [[5], [6], [7], [8]]], [[[9], [10], [11], [12]], [[13], [14], [15], [16]]]]` * `name`: A name for the operation (optional).

Returns:

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