tf.depth_to_space

```
depth_to_space(
    input,
    block_size,
    name=None,
    data_format='NHWC'
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

DepthToSpace for tensors of type T.

Rearranges data from depth into blocks of spatial data. This is the reverse transformation of SpaceToDepth. More specifically, this op outputs a copy of the input tensor where values from the `depth` dimension are moved in spatial blocks to the `height` and `width` dimensions. The attr `block_size` indicates the input block size and how the data is moved.

- Chunks of data of size `block_size * block_size` from depth are rearranged into non-overlapping blocks of size `block_size x block_size`
- The width the output tensor is `input_depth * block_size`, whereas the height is `input_height * block_size`.
- The Y, X coordinates within each block of the output image are determined by the high order component of the input channel index.
- The depth of the input tensor must be divisible by `block_size * block_size`.

The `data_format` attr specifies the layout of the input and output tensors with the following options: "NHWC": `[ batch, height, width, channels ]` "NCHW": `[ batch, channels, height, width ]` "NCHW_VECT_C": `qint8 [ batch, channels / 4, height, width, channels % 4 ]`

It is useful to consider the operation as transforming a 6-D Tensor. e.g. for data_format = NHWC, Each element in the input tensor can be specified via 6 coordinates, ordered by decreasing memory layout significance as: n,iY,iX,bY,bX,oC (where n=batch index, iX, iY means X or Y coordinates within the input image, bX, bY means coordinates within the output block, oC means output channels). The output would be the input transposed to the following layout: n,iY,bY,iX,bX,oC

This operation is useful for resizing the activations between convolutions (but keeping all data), e.g. instead of pooling. It is also useful for training purely convolutional models.

For example, given an input of shape `[1, 1, 1, 4]`, data_format = "NHWC" and block_size = 2:

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

This operation will output a tensor of shape `[1, 2, 2, 1]`:

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

Here, the input has a batch of 1 and each batch element has shape `[1, 1, 4]`, the corresponding output will have 2x2 elements and will have a depth of 1 channel (1 = `4 / (block_size * block_size)`). The output element shape is `[2, 2, 1]`.

For an input tensor with larger depth, here of shape `[1, 1, 1, 12]`, e.g.

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

This operation, for block size of 2, will return the following tensor of shape `[1, 2, 2, 3]`

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

Similarly, for the following input of shape `[1 2 2 4]`, and a block size of 2:

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

the operator will return the following tensor of shape `[1 4 4 1]`:

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

Args:

- `input`: A `Tensor`.
- `block_size`: An `int` that is `>= 2`. The size of the spatial block, same as in Space2Depth.
- `data_format`: An optional `string` from: `"NHWC", "NCHW", "NCHW_VECT_C"`. Defaults to `"NHWC"`.
- `name`: A name for the operation (optional).

Returns:

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

Last updated November 2, 2017.