tf.scatter_nd

```
scatter_nd(
    indices,
    updates,
    shape,
    name=None
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

Scatter `updates` into a new (initially zero) tensor according to `indices`.

Creates a new tensor by applying sparse `updates` to individual values or slices within a zero tensor of the given `shape` according to indices. This operator is the inverse of the `tf.gather_nd` operator which extracts values or slices from a given tensor.

WARNING: The order in which updates are applied is nondeterministic, so the output will be nondeterministic if `indices` contains duplicates.

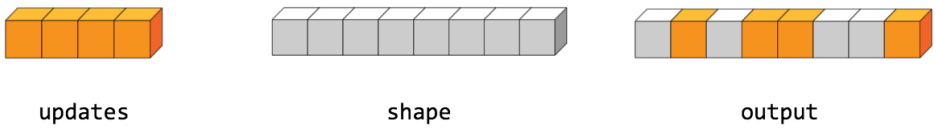
`indices` is an integer tensor containing indices into a new tensor of shape `shape`. The last dimension of `indices` can be at most the rank of `shape`:

```
indices.shape[-1] <= shape.rank
```

The last dimension of `indices` corresponds to indices into elements (if `indices.shape[-1] = shape.rank`) or slices (if `indices.shape[-1] < shape.rank`) along dimension `indices.shape[-1]` of `shape`. `updates` is a tensor with shape

```
indices.shape[:-1] + shape[indices.shape[-1]:]
```

The simplest form of scatter is to insert individual elements in a tensor by index. For example, say we want to insert 4 scattered elements in a rank-1 tensor with 8 elements.



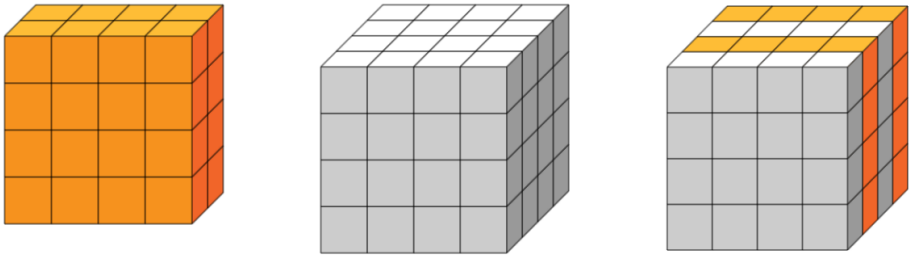
In Python, this scatter operation would look like this:

```
    indices = tf.constant([[4], [3], [1], [7]])
    updates = tf.constant([9, 10, 11, 12])
    shape = tf.constant([8])
    scatter = tf.scatter_nd(indices, updates, shape)
    with tf.Session() as sess:
      print(sess.run(scatter))
```

The resulting tensor would look like this:

```
[0, 11, 0, 10, 9, 0, 0, 12]
```

We can also, insert entire slices of a higher rank tensor all at once. For example, if we wanted to insert two slices in the first dimension of a rank-3 tensor with two matrices of new values.



updates shape output

In Python, this scatter operation would look like this:

```
    indices = tf.constant([[0], [2]])
    updates = tf.constant([[[5, 5, 5, 5], [6, 6, 6, 6],
                            [7, 7, 7, 7], [8, 8, 8, 8]],
                           [[5, 5, 5, 5], [6, 6, 6, 6],
                            [7, 7, 7, 7], [8, 8, 8, 8]]])
    shape = tf.constant([4, 4, 4])
    scatter = tf.scatter_nd(indices, updates, shape)
    with tf.Session() as sess:
      print(sess.run(scatter))
```

The resulting tensor would look like this:

```
[[[5, 5, 5, 5], [6, 6, 6, 6], [7, 7, 7, 7], [8, 8, 8, 8]],
 [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]],
 [[5, 5, 5, 5], [6, 6, 6, 6], [7, 7, 7, 7], [8, 8, 8, 8]],
 [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]]
```

Args:

- `indices`: A `Tensor`. Must be one of the following types: `int32`, `int64`. Index tensor.
- `updates`: A `Tensor`. Updates to scatter into output.
- `shape`: A `Tensor`. Must have the same type as `indices`. 1-D. The shape of the resulting tensor.
- `name`: A name for the operation (optional).

Returns:

A `Tensor`. Has the same type as `updates`. A new tensor with the given shape and updates applied according to the indices.

Last updated November 2, 2017.

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**