tf.scatter_nd_add

```
scatter_nd_add(
    ref,
    indices,
    updates,
    use_locking=False,
    name=None
)
```

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

See the guide: Variables > Sparse Variable Updates

Applies sparse addition between `updates` and individual values or slices

within a given variable according to `indices`.

`ref` is a `Tensor` with rank `P` and `indices` is a `Tensor` of rank `Q`.

`indices` must be integer tensor, containing indices into `ref`. It must be shape `[d_0, ..., d_{Q-2}, K]` where `0 < K <= P`.

The innermost dimension of `indices` (with length `K`) corresponds to indices into elements (if `K = P`) or slices (if `K < P`) along the `K`th dimension of `ref`.

`updates` is `Tensor` of rank `Q-1+P-K` with shape:

```
[d_0, ..., d_{Q-2}, ref.shape[K], ..., ref.shape[P-1]].
```

For example, say we want to add 4 scattered elements to a rank-1 tensor to 8 elements. In Python, that addition would look like this:

```
ref = tf.Variable([1, 2, 3, 4, 5, 6, 7, 8])
indices = tf.constant([[4], [3], [1], [7]])
updates = tf.constant([9, 10, 11, 12])
add = tf.scatter_nd_add(ref, indices, updates)
with tf.Session() as sess:
  print sess.run(add)
```

The resulting update to ref would look like this:

```
[1, 13, 3, 14, 14, 6, 7, 20]
```

See `tf.scatter_nd` for more details about how to make updates to slices.

Args:

- `ref`: A mutable `Tensor`. Must be one of the following types: `float32`, `float64`, `int64`, `int32`, `uint8`, `uint16`, `int16`, `int8`, `complex64`, `complex128`, `qint8`, `quint8`, `qint32`, `half`. A mutable Tensor. Should be from a Variable node.
- `indices`: A `Tensor`. Must be one of the following types: `int32`, `int64`. A Tensor. Must be one of the following

types: int32, int64. A tensor of indices into ref.

- `updates`: A `Tensor`. Must have the same type as `ref`. A Tensor. Must have the same type as ref. A tensor of updated values to add to ref.
- `use_locking`: An optional `bool`. Defaults to `False`. An optional bool. Defaults to True. If True, the assignment will be protected by a lock; otherwise the behavior is undefined, but may exhibit less contention.
- `name`: A name for the operation (optional).

Returns:

A mutable `Tensor`. Has the same type as `ref`. Same as ref. Returned as a convenience for operations that want to use the updated values after the update is done.