tf.nn.fractional_max_pool

```
fractional_max_pool(
    value,
    pooling_ratio,
    pseudo_random=False,
    overlapping=False,
    deterministic=False,
    seed=0,
    seed2=0,
    name=None
)
```

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

See the guide: Neural Network > Pooling

Performs fractional max pooling on the input.

Fractional max pooling is slightly different than regular max pooling. In regular max pooling, you downsize an input set by taking the maximum value of smaller N x N subsections of the set (often 2x2), and try to reduce the set by a factor of N, where N is an integer. Fractional max pooling, as you might expect from the word "fractional", means that the overall reduction ratio N does not have to be an integer.

The sizes of the pooling regions are generated randomly but are fairly uniform. For example, let's look at the height dimension, and the constraints on the list of rows that will be pool boundaries.

First we define the following:

1. input_row_length : the number of rows from the input set
2. output_row_length : which will be smaller than the input
3. alpha = input_row_length / output_row_length : our reduction ratio
4. K = floor(alpha)
5. row_pooling_sequence : this is the result list of pool boundary rows

Then, row_pooling_sequence should satisfy:

1. a[0] = 0 : the first value of the sequence is 0
2. a[end] = input_row_length : the last value of the sequence is the size
3. K <= (a[i+1] - a[i]) <= K+1 : all intervals are K or K+1 size
4. length(row_pooling_sequence) = output_row_length+1

For more details on fractional max pooling, see this paper: Benjamin Graham, Fractional Max-Pooling

Args:

- `value`: A `Tensor`. Must be one of the following types: `float32`, `float64`, `int32`, `int64`. 4-D with shape `[batch, height, width, channels]`.
- `pooling_ratio`: A list of `floats` that has length `>= 4`. Pooling ratio for each dimension of `value`, currently only supports row and col dimension and should be >= 1.0. For example, a valid pooling ratio looks like [1.0, 1.44, 1.73,

1.0]. The first and last elements must be 1.0 because we don't allow pooling on batch and channels dimensions. 1.44 and 1.73 are pooling ratio on height and width dimensions respectively.

- `pseudo_random`: An optional `bool`. Defaults to `False`. When set to True, generates the pooling sequence in a pseudorandom fashion, otherwise, in a random fashion. Check paper Benjamin Graham, Fractional Max-Pooling for difference between pseudorandom and random.
- `overlapping`: An optional `bool`. Defaults to `False`. When set to True, it means when pooling, the values at the boundary of adjacent pooling cells are used by both cells. For example:

  `index 0 1 2 3 4`

  `value 20 5 16 3 7`

  If the pooling sequence is [0, 2, 4], then 16, at index 2 will be used twice. The result would be [20, 16] for fractional max pooling. `deterministic`*: An optional* `bool`*. Defaults to* `False`*. When set to True, a fixed pooling region will be used when iterating over a FractionalMaxPool node in the computation graph. Mainly used in unit test to make FractionalMaxPool deterministic.* `seed`: An optional `int`. Defaults to `0`. If either seed or seed2 are set to be non-zero, the random number generator is seeded by the given seed. Otherwise, it is seeded by a random seed. `seed2`*: An optional* `int`*. Defaults to* `0`*. An second seed to avoid seed collision.* `name`: A name for the operation (optional).

Returns:

A tuple of `Tensor` objects (output, row_pooling_sequence, col_pooling_sequence).

- `output`: A `Tensor`. Has the same type as `value`. output tensor after fractional max pooling.
- `row_pooling_sequence`: A `Tensor` of type `int64`. row pooling sequence, needed to calculate gradient.
- `col_pooling_sequence`: A `Tensor` of type `int64`. column pooling sequence, needed to calculate gradient.