tf.dequantize

```
dequantize(
    input,
    min_range,
    max_range,
    mode='MIN_COMBINED',
    name=None
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

Dequantize the 'input' tensor into a float Tensor.

[min_range, max_range] are scalar floats that specify the range for the 'input' data. The 'mode' attribute controls exactly which calculations are used to convert the float values to their quantized equivalents.

In 'MIN_COMBINED' mode, each value of the tensor will undergo the following:

```
if T == qint8, in[i] += (range(T) + 1)/ 2.0
out[i] = min_range + (in[i]* (max_range - min_range) / range(T))
```

here `range(T) = numeric_limits<T>::max() - numeric_limits<T>::min()`

MIN_COMBINED Mode Example

If the input comes from a QuantizedRelu6, the output type is quint8 (range of 0-255) but the possible range of QuantizedRelu6 is 0-6. The min_range and max_range values are therefore 0.0 and 6.0. Dequantize on quint8 will take each value, cast to float, and multiply by 6 / 255. Note that if quantizedtype is qint8, the operation will additionally add each value by 128 prior to casting.

If the mode is 'MIN_FIRST', then this approach is used:

```
number_of_steps = 1 << (# of bits in T)
range_adjust = number_of_steps / (number_of_steps - 1)
range = (range_max - range_min) * range_adjust
range_scale = range / number_of_steps
const double offset_input = static_cast<double>(input) - lowest_quantized;
result = range_min + ((input - numeric_limits<T>::min()) * range_scale)
```

SCALED mode Example

`SCALED` mode matches the quantization approach used in `QuantizeAndDequantize{V2|V3}`.

If the mode is `SCALED`, we do not use the full range of the output type, choosing to elide the lowest possible value for symmetry (e.g., output range is -127 to 127, not -128 to 127 for signed 8 bit quantization), so that 0.0 maps to 0.

We first find the range of values in our tensor. The range we use is always centered on 0, so we find m such that

```
  m = max(abs(input_min), abs(input_max))
```

Our input tensor range is then `[-m, m]`.

Next, we choose our fixed-point quantization buckets, `[min_fixed, max_fixed]`. If T is signed, this is

```
num_bits = sizeof(T) * 8
[min_fixed, max_fixed] =
    [-(1 << (num_bits - 1) - 1), (1 << (num_bits - 1)) - 1]
```

Otherwise, if T is unsigned, the fixed-point range is

```
[min_fixed, max_fixed] = [0, (1 << num_bits) - 1]
```

From this we compute our scaling factor, s:

```
s = (2 * m) / (max_fixed - min_fixed)
```

Now we can dequantize the elements of our tensor:

```
result = input * s
```

Args:

- `input`: A `Tensor`. Must be one of the following types: `qint8`, `quint8`, `qint16`, `quint16`, `qint32`.
- `min_range`: A `Tensor` of type `float32`. The minimum scalar value possibly produced for the input.
- `max_range`: A `Tensor` of type `float32`. The maximum scalar value possibly produced for the input.
- `mode`: An optional `string` from: `"MIN_COMBINED", "MIN_FIRST", "SCALED"`. Defaults to `"MIN_COMBINED"`.
- `name`: A name for the operation (optional).

Returns:

A `Tensor` of type `float32`.

Last updated November 2, 2017.