tf.contrib.bayesflow.custom_grad.custom_gradient

```
custom_gradient(
    fx,
    gx,
    x,
    axis=(),
    fx_gx_manually_stopped=False,
    name=None
)
```

Defined in `tensorflow/contrib/bayesflow/python/ops/custom_grad_impl.py`.

Enables specifying a custom gradient.

This function works by clever application of `stop_gradient`. I.e., observe that:

```
h(x) = x * stop_gradient(g(x)) + stop_gradient(f(x) - x * g(x))
```

is such that `h(x) = stop(f(x))` and `grad[h(x), x] = stop_gradient(g(x)).`

In addition to scalar-domain/scalar-range functions, this function also supports tensor-domain/scalar-range functions. However, in the latter case it is necessary to reduce `x` to a scalar. This can be done by indicating the `axis` over which `f` operates or by appropriately `reduce_sum`-ing `x`, prior to calling this function.

Partial Custom Gradient:

Suppose `h(x) = htilde(x, y)`. Note that `dh/dx = stop(g(x))` but `dh/dy = None`. This is because a `Tensor` cannot have only a portion of its gradient stopped. To circumvent this issue, one must manually `stop_gradient` the relevant portions of `f`, `g`. For example see the unit-test, `test_works_correctly_fx_gx_manually_stopped`.

Args:

- `fx`: `Tensor`. Output of function evaluated at `x`.
- `gx`: `Tensor`. Gradient of function evaluated at `x`.
- `x`: `Tensor`. Point of evaluation for `f, g`.
- `axis`: 1D `int` `Tensor` representing dimensions of `x` which are the domain of `f`. If `()` (the default), `f` is assumed scalar-domain/scalar-range. If `None` `f` is assumed to render one scalar given all of `x`. Otherwise `f` is assumed to output one scalar for each of `axis` dimensions of `x`.
- `fx_gx_manually_stopped`: Python `bool` indicating that `fx`, `gx` manually have `stop_gradient` applied.
- `name`: Python `str` name prefixed to Ops created by this function.

Returns:

- `fx`: Floating-type `Tensor` equal to `f(x)` but which has gradient `stop_gradient(g(x))`.

Last updated November 2, 2017.