tf.while_loop

```
while_loop(
    cond,
    body,
    loop_vars,
    shape_invariants=None,
    parallel_iterations=10,
    back_prop=True,
    swap_memory=False,
    name=None
)
```

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

See the guide: Control Flow > Control Flow Operations

Repeat `body` while the condition `cond` is true.

`cond` is a callable returning a boolean scalar tensor. `body` is a callable returning a (possibly nested) tuple, namedtuple or list of tensors of the same arity (length and structure) and types as `loop_vars`. `loop_vars` is a (possibly nested) tuple, namedtuple or list of tensors that is passed to both `cond` and `body`. `cond` and `body` both take as many arguments as there are `loop_vars`.

In addition to regular Tensors or IndexedSlices, the body may accept and return TensorArray objects. The flows of the TensorArray objects will be appropriately forwarded between loops and during gradient calculations.

Note that `while_loop` calls `cond` and `body` *exactly once* (inside the call to `while_loop`, and not at all during `Session.run()`). `while_loop` stitches together the graph fragments created during the `cond` and `body` calls with some additional graph nodes to create the graph flow that repeats `body` until `cond` returns false.

For correctness, `tf.while_loop()` strictly enforces shape invariants for the loop variables. A shape invariant is a (possibly partial) shape that is unchanged across the iterations of the loop. An error will be raised if the shape of a loop variable after an iteration is determined to be more general than or incompatible with its shape invariant. For example, a shape of [11, None] is more general than a shape of [11, 17], and [11, 21] is not compatible with [11, 17]. By default (if the argument `shape_invariants` is not specified), it is assumed that the initial shape of each tensor in `loop_vars` is the same in every iteration. The `shape_invariants` argument allows the caller to specify a less specific shape invariant for each loop variable, which is needed if the shape varies between iterations. The `tf.Tensor.set_shape` function may also be used in the `body` function to indicate that the output loop variable has a particular shape. The shape invariant for SparseTensor and IndexedSlices are treated specially as follows:

a) If a loop variable is a SparseTensor, the shape invariant must be TensorShape([r]) where r is the rank of the dense tensor represented by the sparse tensor. It means the shapes of the three tensors of the SparseTensor are ([None], [None, r], [r]). NOTE: The shape invariant here is the shape of the SparseTensor.dense_shape property. It must be the shape of a vector.

b) If a loop variable is an IndexedSlices, the shape invariant must be a shape invariant of the values tensor of the IndexedSlices. It means the shapes of the three tensors of the IndexedSlices are (shape, [shape[0]], [shape.ndims]).

`while_loop` implements non-strict semantics, enabling multiple iterations to run in parallel. The maximum number of parallel iterations can be controlled by `parallel_iterations`, which gives users some control over memory consumption and execution order. For correct programs, `while_loop` should return the same result for any parallel_iterations > 0.

For training, TensorFlow stores the tensors that are produced in the forward inference and are needed in back propagation.

These tensors are a main source of memory consumption and often cause OOM errors when training on GPUs. When the flag swap_memory is true, we swap out these tensors from GPU to CPU. This for example allows us to train RNN models with very long sequences and large batches.

Args:

- `cond`: A callable that represents the termination condition of the loop.
- `body`: A callable that represents the loop body.
- `loop_vars`: A (possibly nested) tuple, namedtuple or list of numpy array, `Tensor`, and `TensorArray` objects.
- `shape_invariants`: The shape invariants for the loop variables.
- `parallel_iterations`: The number of iterations allowed to run in parallel. It must be a positive integer.
- `back_prop`: Whether backprop is enabled for this while loop.
- `swap_memory`: Whether GPU-CPU memory swap is enabled for this loop.
- `name`: Optional name prefix for the returned tensors.

Returns:

The output tensors for the loop variables after the loop. When the length of `loop_vars` is 1 this is a Tensor, TensorArray or IndexedSlice and when the length of `loop_vars` is greater than 1 it returns a list.

Raises:

- `TypeError`: if `cond` or `body` is not callable.
- `ValueError`: if `loop_vars` is empty.

Example:

```
i = tf.constant(0)
c = lambda i: tf.less(i, 10)
b = lambda i: tf.add(i, 1)
r = tf.while_loop(c, b, [i])
```

Example with nesting and a namedtuple:

```
import collections
Pair = collections.namedtuple('Pair', 'j, k')
ijk_0 = (tf.constant(0), Pair(tf.constant(1), tf.constant(2)))
c = lambda i, p: i < 10
b = lambda i, p: (i + 1, Pair((p.j + p.k), (p.j - p.k)))
ijk_final = tf.while_loop(c, b, ijk_0)
```

Example using shape_invariants:

```
i0 = tf.constant(0)
m0 = tf.ones([2, 2])
c = lambda i, m: i < 10
b = lambda i, m: [i+1, tf.concat([m, m], axis=0)]
tf.while_loop(
    c, b, loop_vars=[i0, m0],
    shape_invariants=[i0.get_shape(), tf.TensorShape([None, 2])])
```

Last updated November 2, 2017.