# tf.Operation

Contents

## Class **`Operation`**

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

See the guide: Building Graphs > Core graph data structures

Represents a graph node that performs computation on tensors.

An `Operation` is a node in a TensorFlow `Graph` that takes zero or more `Tensor` objects as input, and produces zero or more `Tensor` objects as output. Objects of type `Operation` are created by calling a Python op constructor (such as `tf.matmul`) or `tf.Graph.create_op`.

For example `c = tf.matmul(a, b)` creates an `Operation` of type "MatMul" that takes tensors `a` and `b` as input, and produces `c` as output.

After the graph has been launched in a session, an `Operation` can be executed by passing it to `tf.Session.run`. `op.run()` is a shortcut for calling `tf.get_default_session().run(op)`.

## Properties

### **`control_inputs`**

The `Operation` objects on which this op has a control dependency.

Before this op is executed, TensorFlow will ensure that the operations in `self.control_inputs` have finished executing. This mechanism can be used to run ops sequentially for performance reasons, or to ensure that the side effects of an op are observed in the correct order.

#### Returns:

A list of `Operation` objects.

### **`device`**

The name of the device to which this op has been assigned, if any.

#### Returns:

The string name of the device to which this op has been assigned, or an empty string if it has not been assigned to a device.

### `graph`

The `Graph` that contains this operation.

### `inputs`

The list of `Tensor` objects representing the data inputs of this op.

### `name`

The full name of this operation.

### `node_def`

Returns a serialized `NodeDef` representation of this operation.

#### Returns:

A `NodeDef` protocol buffer.

### `op_def`

Returns the `OpDef` proto that represents the type of this op.

#### Returns:

An `OpDef` protocol buffer.

### `outputs`

The list of `Tensor` objects representing the outputs of this op.

### `traceback`

Returns the call stack from when this operation was constructed.

### `traceback_with_start_lines`

Same as traceback but includes start line of function definition.

#### Returns:

A list of 5-tuples (filename, lineno, name, code, func_start_lineno).

### `type`

The type of the op (e.g. `"MatMul"`).

## Methods

### `__init__`

```
__init__(
    node_def,
    g,
    inputs=None,
    output_types=None,
    control_inputs=None,
    input_types=None,
    original_op=None,
    op_def=None
)
```

Creates an `Operation`.

NOTE: This constructor validates the name of the `Operation` (passed as `node_def.name`). Valid `Operation` names match the following regular expression:

```
[A-Za-z0-9.][A-Za-z0-9_.\\-/]*
```

#### Args:

- `node_def`: `node_def_pb2.NodeDef`. `NodeDef` for the `Operation`. Used for attributes of `node_def_pb2.NodeDef`, typically `name`, `op`, and `device`. The `input` attribute is irrelevant here as it will be computed when generating the model.
- `g`: `Graph`. The parent graph.
- `inputs`: list of `Tensor` objects. The inputs to this `Operation`.
- `output_types`: list of `DType` objects. List of the types of the `Tensors` computed by this operation. The length of this list indicates the number of output endpoints of the `Operation`.
- `control_inputs`: list of operations or tensors from which to have a control dependency.
- `input_types`: List of `DType` objects representing the types of the tensors accepted by the `Operation`. By default uses `[x.dtype.base_dtype for x in inputs]`. Operations that expect reference-typed inputs must specify these explicitly.
- `original_op`: Optional. Used to associate the new `Operation` with an existing `Operation` (for example, a replica with the op that was replicated).
- `op_def`: Optional. The `op_def_pb2.OpDef` proto that describes the op type that this `Operation` represents.

#### Raises:

- `TypeError`: if control inputs are not Operations or Tensors, or if `node_def` is not a `NodeDef`, or if `g` is not a `Graph`, or if `inputs` are not tensors, or if `inputs` and `input_types` are incompatible.
- `ValueError`: if the `node_def` name is not valid.

### `colocation_groups`

```
colocation_groups()
```

Returns the list of colocation groups of the op.

### `get_attr`

```
get_attr(name)
```

Returns the value of the attr of this op with the given `name`.

#### Args:

- `name`: The name of the attr to fetch.

#### Returns:

The value of the attr, as a Python object.

#### Raises:

- `ValueError`: If this op does not have an attr with the given `name`.

### `run`

```
run(
    feed_dict=None,
    session=None
)
```

Runs this operation in a `Session`.

Calling this method will execute all preceding operations that produce the inputs needed for this operation.

*N.B.* Before invoking `Operation.run()`, its graph must have been launched in a session, and either a default session must be available, or `session` must be specified explicitly.

#### Args:

- `feed_dict`: A dictionary that maps `Tensor` objects to feed values. See `tf.Session.run` for a description of the valid feed values.
- `session`: (Optional.) The `Session` to be used to run to this operation. If none, the default session will be used.

### `values`

```
values()
```

DEPRECATED: Use outputs.