What's the difference of name scope and a variable scope in tensorflow?

What's the differences between these functions?

tf.**variable_op_scope**(values, name, default_name, initializer=None)
Returns a context manager for defining an op that creates variables. This context manager validates that the given values are from the same graph, ensures that that graph is the default graph, and pushes a name scope and a variable scope.

tf.**op_scope**(values, name, default_name=None) Returns a context manager for use when defining a Python op. This context manager validates that the given values are from the same graph, ensures that that graph is the default graph, and pushes a name scope.

tf.**name_scope**(name)
Wrapper for Graph.name_scope() using the default graph. See Graph.name_scope() for more details.

tf.**variable_scope**(name_or_scope, reuse=None, initializer=None)
Returns a context for variable scope. Variable scope allows to create new variables and to share already created ones while providing checks to not create or share by accident. For details, see the Variable Scope How To, here we present only a few basic examples.

tensorflow

asked Mar 10 '16 at 14:19
user2886263
584 1 7 10

6 Answers

Let's begin by a short introduction to variable sharing. It is a mechanism in TensorFlow that allows for sharing variables accessed in different parts of the code without passing references to the variable around. The method `tf.get_variable` can be used with the name of the variable as the argument to either create a new variable with such name or retrieve the one that was created before. This is different from using the `tf.Variable` constructor which will create a new variable every time it is called (and potentially add a suffix to the variable name if a variable with such name already exists). It is for the purpose of the variable sharing mechanism that a separate type of scope (variable scope) was introduced.

As a result, we end up having two different types of scopes:

- *name scope*, created using `tf.name_scope`
- *variable scope*, created using `tf.variable_scope`

Both scopes have the same effect on all operations as well as variables created using `tf.Variable`, i.e., the scope will be added as a prefix to the operation or variable name.

However, name scope is ignored by `tf.get_variable`. We can see that in the following example:

```
with tf.name_scope("my_scope"):
    v1 = tf.get_variable("var1", [1], dtype=tf.float32)
    v2 = tf.Variable(1, name="var2", dtype=tf.float32)
    a = tf.add(v1, v2)

print(v1.name)  # var1:0
print(v2.name)  # my_scope/var2:0
print(a.name)   # my_scope/Add:0
```

The only way to place a variable accessed using `tf.get_variable` in a scope is to use a variable scope, as in the following example:

```
with tf.variable_scope("my_scope"):
    v1 = tf.get_variable("var1", [1], dtype=tf.float32)
    v2 = tf.Variable(1, name="var2", dtype=tf.float32)
    a = tf.add(v1, v2)

print(v1.name)  # my_scope/var1:0
print(v2.name)  # my_scope/var2:0
print(a.name)   # my_scope/Add:0
```

This allows us to easily share variables across different parts of the program, even within different name scopes:

```
with tf.name_scope("foo"):
    with tf.variable_scope("var_scope"):
        v = tf.get_variable("var", [1])
with tf.name_scope("bar"):
    with tf.variable_scope("var_scope", reuse=True):
        v1 = tf.get_variable("var", [1])
assert v1 == v
print(v.name)   # var_scope/var:0
print(v1.name)  # var_scope/var:0
```

UPDATE: op_scope/variable_op_scope is deprecated!

As of version r0.11, `op_scope` and `variable_op_scope` are both deprecated and replaced by `name_scope` and `variable_scope`.

edited May 30 at 5:05
Bishwajit Purkaystha
145 1 9

answered May 30 '16 at 23:09
Andrzej Pronobis
10.1k 5 32 56

5 Thanks for the clear explanation. Naturally, a follow up question would be "*Why* does Tensorflow have both of these confusingly similar mechanisms? Why not replace them with just one `scope` method which effectively does a `variable_scope`?" – John Feb 25 at 18:02

I don't think I understand conceptually why the distinction between `variable_scope` vs `name_scope` is even needed. If one creates a variable (in any way with `tf.Variable` or `tf.get_variable`), it seems more natural to me that we should always be able to get it if we specify the scope or its full name. I don't understand why one ignores the scope name thing while the other doesn't. Do you understand the rational for this weird behaviour? – Charlie Parker Mar 20 at 19:31

5 The reason is that with variable scope, one can define separate scopes for re-usable variables that are not affected by the current name scope used to define operations. – Andrzej Pronobis Mar 21 at 5:23

What I understand is name_scope is mainly useful when you name two variables with same name but in different operations. – DINESHKUMAR MURUGAN Jul 14 at 17:45

Namespaces is a way to organize names for variables and operators in hierarchical manner (e.g. "scopeA/scopeB/scopeC/op1")

- tf.name_scope creates namespace for operators in the default graph.
- tf.variable_scope creates namespace for both variables and operators in the default graph.
- tf.op_scope same as tf.name_scope, but for the graph in which specified variables were created.
- tf.variable_op_scope same as tf.variable_scope, but for the graph in which specified variables were created.

Links to the sources above help to disambiguate this documentation issue.

UPDATE This example shows that all types of scopes define namespaces for both variables and operators with following differences:

1. scopes defined by tf.variable_op_scope or tf.variable_scope are compatible with tf.get_variable (it ignores two other scopes)
2. tf.op_scope and tf.variable_op_scope just select a graph from a list of specified variables to create a scope for. Other than than their behavior equal to tf.name_scope and tf.variable_scope accordingly
3. tf.variable_scope and variable_op_scope add specified or default initializer.

edited Mar 11 '16 at 5:46 answered Mar 10 '16 at 23:37
Alexander Gorban
560 3 12

For the graph in which specified variables were created? Does this means such as above example by fabrizioM, with tf.variable_op_scope([a,b],name,"mysum2") as scope, here parameter a and b are not affected by this function and variables defined in this scope are affected? – user2886263 Mar 11 '16 at 1:54

The answer for both questions is yes: the graph in which specified variables were created and they are not modified. – Alexander Gorban Mar 11 '16 at 4:56

Does this mean that tf.name_scope and tf.variable_scope only be used in default graph, but when you obviously define and constuct a graph using tf.Graph(), the other two functions tf.op_scope and tf.variable_op_scope can not be used in this graph! – user2886263 Mar 12 '16 at 11:21

As for API r0.11, `op_scope` and `variable_op_scope` are both deprecated. `name_scope` and `variable_scope` can be nested:

```
with tf.name_scope('ns'):
    with tf.variable_scope('vs'):
        v1 = tf.get_variable("v1",[1.0])   #v1.name = 'vs/v1:0'
        v2 = tf.Variable([2.0],name = 'v2')  #v2.name= 'ns/vs/v2:0'
        v3 = v1 + v2       #v3.name = 'ns/vs/add:0'
```

edited Jan 17 at 22:17 answered Jan 16 at 18:20
sgu
119 1 7

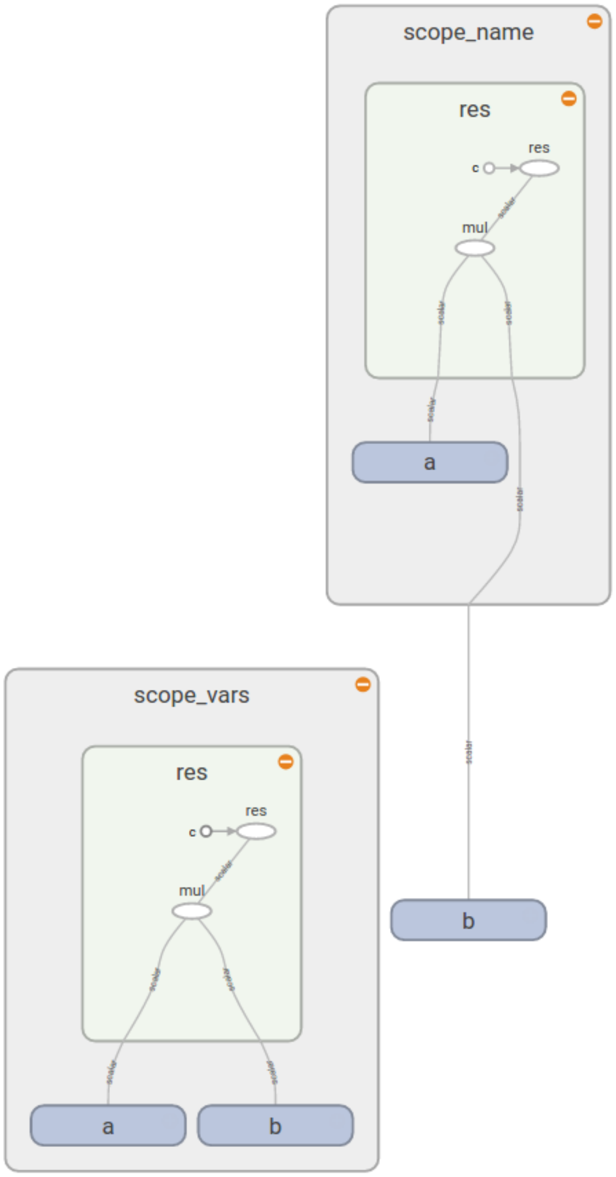
Both variable_op_scope and op_scope are now deprecated and should not be used at all. Regarding the other two, I also had problems understanding the difference between variable_scope and name_scope (they looked almost the same) before I tried to visualize everything by creating a simple example:

```
import tensorflow as tf
def scoping(fn, scope1, scope2, vals):
    with fn(scope1):
        a = tf.Variable(vals[0], name='a')
        b = tf.get_variable('b', initializer=vals[1])
        c = tf.constant(vals[2], name='c')
        with fn(scope2):
            d = tf.add(a * b, c, name='res')

        print '\n  '.join([scope1, a.name, b.name, c.name, d.name]), '\n'
    return d

d1 = scoping(tf.variable_scope, 'scope_vars', 'res', [1, 2, 3])
d2 = scoping(tf.name_scope,     'scope_name', 'res', [1, 2, 3])

with tf.Session() as sess:
    writer = tf.summary.FileWriter('logs', sess.graph)
    sess.run(tf.global_variables_initializer())
    print sess.run([d1, d2])
    writer.close()
```

Here I create a function that creates some variables and constants and groups them in scopes (depending by the type I provided). In this function I also print the names of all the variables. After that I executes the graph to get values of the resulting values and save event-files to investigate them in tensorboard. If you run this, you will get the following:

```
scope_vars
  scope_vars/a:0
  scope_vars/b:0
  scope_vars/c:0
  scope_vars/res/res:0

scope_name
  scope_name/a:0
  b:0
  scope_name/c:0
  scope_name/res/res:0
```

You see the similar pattern if you open TB (as you see `b` is outside of `scope_name`



rectangular):

This gives you the answer:

Now you see that `tf.variable_scope()` adds a prefix to the names of all variables (no matter how you create them), ops, constants. On the other hand `tf.name_scope()` ignores variables created with `tf.get_variable()` because it assumes that you know which variable and in which scope you wanted to use.

A good documentation on Sharing variables tells you that

> `tf.variable_scope()` : Manages namespaces for names passed to `tf.get_variable()` .

The same documentation provides a more details how does Variable Scope work and when it is useful.

edited Jul 12 at 6:49
Alexander Yau
2,515 1 24 46

answered Apr 24 at 6:48
Salvador Dali
71.9k 56 339 425

You can think them as two groups: `variable_op_scope` and `op_scope` take a set of variables as input and are designed to create operations. The difference is in how they affect the creation of variables with `tf.get_variable` :

```
def mysum(a,b,name=None):
    with tf.op_scope([a,b],name,"mysum") as scope:
        v = tf.get_variable("v", 1)
        v2 = tf.Variable([0], name="v2")
        assert v.name == "v:0", v.name
        assert v2.name == "mysum/v2:0", v2.name
        return tf.add(a,b)

def mysum2(a,b,name=None):
    with tf.variable_op_scope([a,b],name,"mysum2") as scope:
        v = tf.get_variable("v", 1)
        v2 = tf.Variable([0], name="v2")
        assert v.name == "mysum2/v:0", v.name
        assert v2.name == "mysum2/v2:0", v2.name
        return tf.add(a,b)

with tf.Graph().as_default():
    op = mysum(tf.Variable(1), tf.Variable(2))
    op2 = mysum2(tf.Variable(1), tf.Variable(2))
    assert op.name == 'mysum/Add:0', op.name
    assert op2.name == 'mysum2/Add:0', op2.name
```

notice the name of the variable `v` in the two examples.

same for `tf.name_scope` and `tf.variable_scope` :

```
with tf.Graph().as_default():
    with tf.name_scope("name_scope") as scope:
        v = tf.get_variable("v", [1])
        op = tf.add(v, v)
        v2 = tf.Variable([0], name="v2")
        assert v.name == "v:0", v.name
        assert op.name == "name_scope/Add:0", op.name
        assert v2.name == "name_scope/v2:0", v2.name

with tf.Graph().as_default():
    with tf.variable_scope("name_scope") as scope:
        v = tf.get_variable("v", [1])
        op = tf.add(v, v)
        v2 = tf.Variable([0], name="v2")
        assert v.name == "name_scope/v:0", v.name
        assert op.name == "name_scope/Add:0", op.name
        assert v2.name == "name_scope/v2:0", v2.name
```

You can read more about variable scope in the tutorial. A similar question was asked before on Stack Overflow.

edited Jul 12 at 6:51
Alexander Yau
2,515 1 24 46

answered Mar 10 '16 at 21:42
fabrizioM
23.7k 8 52 71

Thanks very very much, I have already saw the tutorial, but maybe too many stuff to find the key. – user2886263 Mar 11 '16 at 1:49

From the last section of this page of the tensorflow documentation: Names of ops in `tf.variable_scope()`

> [...] when we do `with tf.variable_scope("name")`, this implicitly opens a `tf.name_scope("name")`. For example:

```
with tf.variable_scope("foo"):
  x = 1.0 + tf.get_variable("v", [1])
assert x.op.name == "foo/add"
```

> Name scopes can be opened in addition to a variable scope, and then they will only affect the names of the ops, but not of variables.

```
with tf.variable_scope("foo"):
    with tf.name_scope("bar"):
        v = tf.get_variable("v", [1])
        x = 1.0 + v
assert v.name == "foo/v:0"
assert x.op.name == "foo/bar/add"
```

> When opening a variable scope using a captured object instead of a string, we do not alter the current name scope for ops.

edited Jul 12 at 6:50
Alexander Yau
2,515 1 24 46

answered Mar 7 at 21:49
Guillermo González de Garibay
31 1