TensorFlow: Restoring variables from from multiple checkpoints

I have the following situation:

- I have 2 models written in 2 separate scripts:
- Model A consists of variables `a1`, `a2`, and `a3`, and is written in `A.py`
- Model B consists of variables `b1`, `b2`, and `b3`, and is written in B.py

In each of `A.py` and `B.py`, I have a `tf.train.Saver` that saves the checkpoint of all the local variables, and let's call the checkpoint files `ckptA` and `ckptB` respectively.

I now want to make a model C that uses `a1` and `b1`. I can make it so that the exact same variable name for `a1` is used in both A and C by using the var_scope (and the same for `b1`).

The question is how might I load `a1` and `b1` from `ckptA` and `ckptB` into model C? For example, would the following work?

```
saver.restore(session, ckptA_location)
saver.restore(session, ckptB_location)
```

Would an error be raised if you are try to restore the same session twice? Would it complain that there are no allocated "slots" for the extra variables (`b2` , `b3` , `a2` , `a3`), or would it simply restore the variables it can, and only complain if there are some other variables in C that are uninitialized?

I'm trying to write some code to test this now but I would love to see a canonical approach to this problem, because one encounters this often when trying to re-use some pre-trained weights.

Thanks!

tensorflow

edited Mar 1 '16 at 22:37

mrry
46.8k 3 112 162

asked Mar 1 '16 at 21:29

Evan Pu
660 5 18

1 Answer

You would get a `tf.errors.NotFoundError` if you tried to use a saver (by default representing all six variables) to restore from a checkpoint that does not contain all of the variables that the saver represents. (Note however that you are free to call `Saver.restore()` multiple times in the same session, for any subset of the variables, as long as all of the requested variables are present in the corresponding file.)

The canonical approach is to define **two separate** `tf.train.Saver` **instances** covering each subset of variables that is entirely contained in a single checkpoint. For example:

```
saver_a = tf.train.Saver([a1])
saver_b = tf.train.Saver([b1])
```

```
saver_a.restore(session, ckptA_location)
saver_b.restore(session, ckptB_location)
```

Depending on how your code is built, if you have pointers to `tf.Variable` objects called `a1` and `b1` in the local scope, you can stop reading here.

On the other hand, if variables `a1` and `b1` are defined in separate files, you might need to do something creative to retrieve pointers to those variables. Although it's not ideal, what people typically do is to use a common prefix, for example as follows (assuming the variable names are `"a1:0"` and `"b1:0"` respectively):

```
saver_a = tf.train.Saver([v for v in tf.all_variables() if v.name == "a1:0"])
saver_b = tf.train.Saver([v for v in tf.all_variables() if v.name == "b1:0"])
```

One final note: you don't have to make heroic efforts to ensure that the variables have the same names in A and C. You can pass a name-to-`Variable` dictionary as the first argument to the `tf.train.Saver` constructor, and thereby remap names in the checkpoint file to `Variable` objects in your code. This helps if `A.py` and `B.py` have similarly-named variables, or if in `C.py` you want to organize the model code from those files in a `tf.name_scope()`.

edited Mar 1 '16 at 22:39

answered Mar 1 '16 at 22:02

mrry
46.8k 3 112 162

---

the first code fragments refers to stuff that should be written in C.py correct? and it is written toward the end of the graph definition where a1 a2 a3 and b1 b2 b3 are defined in C.py? My original question was what if only a1 and b1 are defined in C and nothing else? – Evan Pu Mar 1 '16 at 22:29

Apologies - updated the answer to cover this case. If you redefine the variables in C.py then things are much easier! – mrry Mar 1 '16 at 22:33

One more follow up: when we execute this statement "saver_a.restore(session, ckptA_location)" saver_a is instantiated with the single variable [a1] and nothing else, yet ckptA contains values for all a1,a2,a3. You are saying this wouldn't be an issue, as the saver will only search for a1 in the ckpt and restore a1 to model C, while ignoring a2 and a3? And one last question: the a1 in C will be identified as the a1 in the ckptA as long as both a1 are instantiated (in C and A) with the same name right? – Evan Pu Mar 2 '16 at 3:15

1 That's correct (both parts). For the latter part you can specify an explicit name-to-`Variable` map if `a1` has a different name in the checkpoint. By default, the `Saver` constructor uses the `Variable.name` property to look this up. – mrry Mar 2 '16 at 3:23