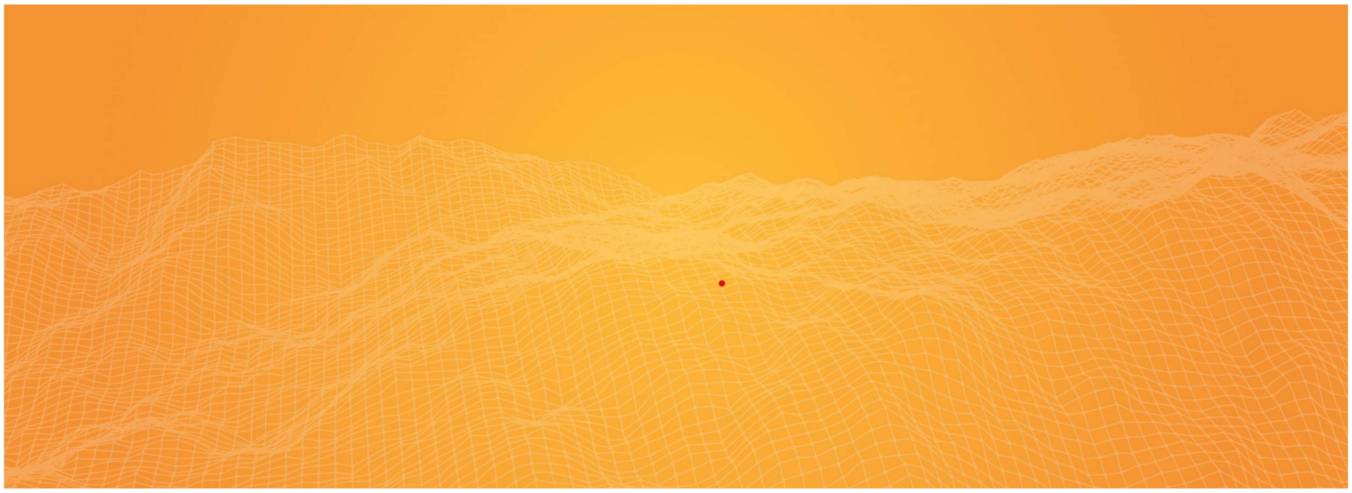
Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

g work and waiting for my own creativity to kick in!
Nov 26, 2016 · 4 min read



Official TensorFlow background

TensorFlow: How to freeze a model and serve it with a python API

We are going to explore two parts of using a ML model in production:

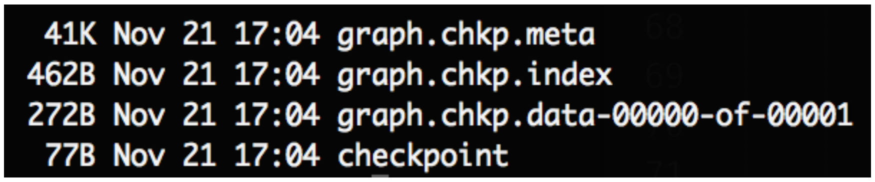
- How to export a model and have a simple self-sufficient file for it
- How to build a simple python server (using flask) to serve it with TF

Note: if you want to see the kind of graph I save/load/freeze, you can here

How to freeze (export) a saved model

If you wonder how to save a model with TensorFlow, please have a look at my previous article before going on.

let's start from a folder containing a model, it probably looks something like this:



```
 41K Nov 21 17:04 graph.chkp.meta
462B Nov 21 17:04 graph.chkp.index
272B Nov 21 17:04 graph.chkp.data-00000-of-00001
 77B Nov 21 17:04 checkpoint
```

Screenshot of the result folder before freezing our model

The important files here are the "**.chkp**" ones**.** If you remember well, for each pair at different timesteps, one is holding the weights (**".data"**) and the other one (**".meta"**) is holding the graph and all its metadata (so you can retrain it etc…)

But when we want to serve a model in production, we don't need any special metadata to clutter our files, we just want our model and its weights nicely packaged in one file. This facilitate storage, versioning and updates of your different models.

Luckily in TF, we can easily build our own function to do it. Let's explore the different steps we have to perform:

> Retrieve our saved graph: we need to load the previously saved meta graph in the default graph and retrieve its graph_def (the ProtoBuf definition of our graph)
>
> Restore the weights: we start a Session and restore the weights of our graph inside that Session
>
> Remove all metadata useless for inference: Here, TF helps us with a nice helper function which grab just what is needed in your graph to perform inference and returns what we will call our new **"frozen graph_def"**
>
> Save it to the disk, Finally we will serialize our frozen graph_def ProtoBuf and dump it to the disk

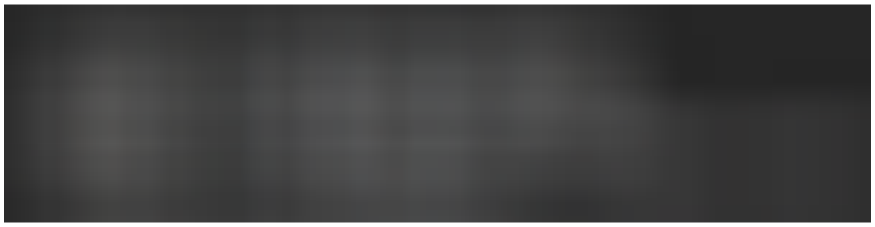
Note that the two first steps are the same as when we load any graph in TF, the only tricky part is actually the graph **"freezing"** and TF has a built-in function to do it!

I provide a slightly different version which is simpler and that I found handy. The original freeze_graph function provided by TF is installed in your `bin` dir and can be called directly if you used PIP to install TF. If not you can call it directly from its folder (see the commented import in the gist).

So let’s see:

```python
import os, argparse

import tensorflow as tf

# The original freeze_graph function
# from tensorflow.python.tools.freeze_graph import fre

dir = os.path.dirname(os.path.realpath(__file__))

def freeze_graph(model_dir, output_node_names):
    """Extract the sub graph defined by the output nod
    all its variables into constant

    Args:
        model_dir: the root folder containing the chec
        output_node_names: a string, containing all th
                            comma separated
    """
    if not tf.gfile.Exists(model_dir):
        raise AssertionError(
            "Export directory doesn't exists. Please s
            "directory: %s" % model_dir)

    if not output_node_names:
        print("You need to supply the name of a node t
        return -1

    # We retrieve our checkpoint fullpath
    checkpoint = tf.train.get_checkpoint_state(model_d
    input_checkpoint = checkpoint.model_checkpoint_pat

    # We precise the file fullname of our freezed grap
    absolute_model_dir = "/".join(input_checkpoint.spl
    output_graph = absolute_model_dir + "/frozen_model

    # We clear devices to allow TensorFlow to control
    clear_devices = True

    # We start a session using a temporary fresh Graph
    with tf.Session(graph=tf.Graph()) as sess:
        # We import the meta graph in the current defa
        saver = tf.train.import_meta_graph(input_check
```

Now we can see a new file in our folder: **“frozen_model.pb”.**



Screenshot of the result folder after freezing our model

As expected, its size is bigger than the weights file size and lower than the sum of the two checkpoints files sizes.

> Note: In this very simple case, the weights size is very small, but it is usually multiple Mbs.

How to use the frozen model

Naturally, after knowing how to freeze a model, one might wonder how to use it.

The little trick to have in mind is to understand that what we dumped to the disk was a graph_def ProtoBuf. So to import it back in a python script we need to:

- Import a graph_def ProtoBuf first
- Load this graph_def into a actual Graph

We can build a convenient function to do so:

Now that we built our function to load our frozen model, let's create a simple script to finally make use of it:

> Note: when loading the frozen model, all operations got prefixed by "prefix". This is due to the parameter "name" in the "import_graph_def" function, by default it prefix everything by "import".

> This can be useful to avoid name collisions if you want to import your graph_def in an existing Graph.

How to build a (very) simple API

For this part, I will let the code speaks for itself, after all this is a TF series about TF and not so much about how to build a server in python. Yet it felt kind of unfinished without it, so here you go, the final workflow:

Note: We are using flask in this example

. . .

TensorFlow best practice series

This article is part of a more complete series of articles about TensorFlow. I've not yet defined all the different subjects of this series, so if you want to see any area of TensorFlow explored, add a comment! So far I wanted to explore those subjects (this list is subject to change and is in no particular order):

- A primer
- How to handle shapes in TensorFlow
- TensorFlow saving/restoring and mixing multiple models
- How to freeze a model and serve it with python (this one!)
- TensorFlow: A proposal of good practices for files, folders and models architecture
- TensorFlow howto: a universal approximator inside a neural net
- How to optimise your input pipeline with queues and multi-threading
- Mutating variables and control flow
- How to handle input data with TensorFlow.
- How to control the gradients to create custom back-prop or fine-tune my models.
- How to monitor my training and inspect my models to gain insight about them.

> ***Note:*** *TF is evolving fast right now, those articles are currently written for the* ***1.0.0*** *version.*

. . .

References

- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/tools/freeze_graph.py