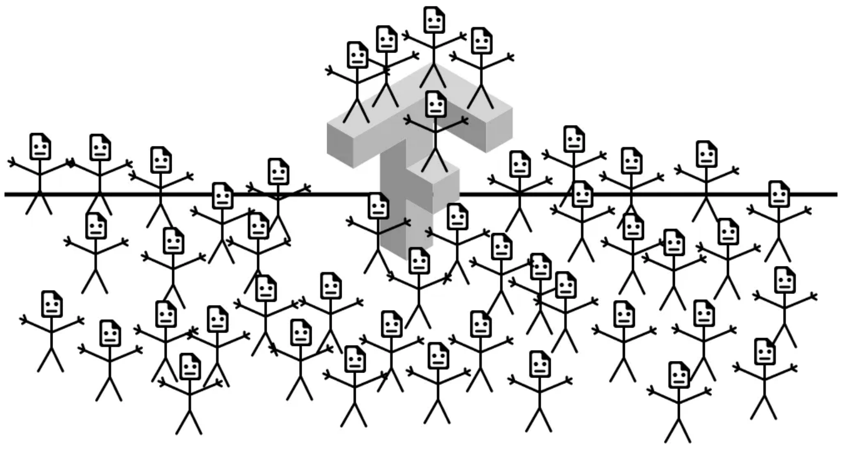
Thousands of CSV files, Keras and TensorFlow

Guide on how to deal with this hell

I have about 15 000 CSV files with 6 000 rows in each, and I need to train a neural network using all these data: about 90 000 000 instances totally. That's how real machine learning looks like!

I hope that I will save you time telling how to train NNs using `generators`, `tf.data.Dataset`, and other pretty interesting stuff.



They go for you. Image author: Denis Shilov (that's me).

Intro

There is no way to concat all these data into one file or in one array, as it will be something huge.

So the only way to handle this huge data array is to do it by batches:

1. We get a list of files
2. We split it into training and testing datasets
3. Then we use something to put data by batches to Keras

Easy part

Chilling, that's really easy. Important notice: all files should be in one directory for this code to work.

Tricky part #1

There are several approaches for doing handling by batches.

For example, one of the approaches is to use `generator`. It is a function that returns an iterator, and we can iterate through its values: one value at a time.

Keras allows us to pass this `generator` to `.fit` by default.

So let's write that iterator!

Pretty self-explaining: you pass in the `files` array and the `batch_size` , and corresponding `input` and `output` are now returned.

Then you can init your generators the following way:

`batch_size` is how many rows will be returned at once. Typically you can init it like the number of rows in a single CSV, but if this number is too enormous, then set something not so enormous (I don't know, 5 000, for example).

And you fit a model.

`callback_list` is a thing which monitors if some parameter of training starts to decrease too slow, and there is no reason to continue training.

`steps_per_epoch` tells when it is necessary to start a new epoch. In case you don't provide it Keras won't know the length of your data and will print "Unknown" in the log.

`use_multiprocessing` indicates if you want to process data in several threads.

`workers` is a number of such threads. This number should be less than the number of cores of your CPU.

`x` is your generator. As it returns both input and output, then we don't set `y` .

`verbose` is how much log is detailed.

As you have several processes preprocessing data for training, they add these data somewhere for Keras to take them and train NN.
And `max_queue_size` specifies the limit of the number of data stored but not yet processed. You should set it, as there is no need to preprocess more data than Keras can consume at once: your dataset is huge and the memory will be overloaded.

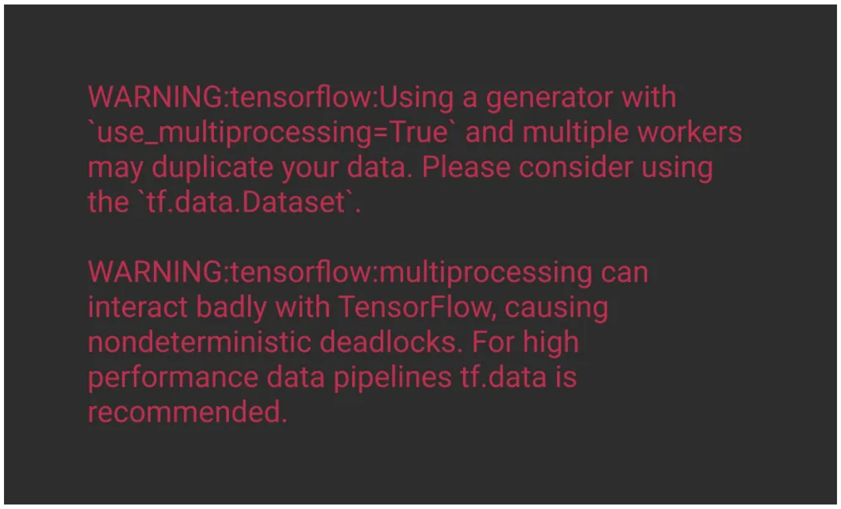
`epochs` is the number of iterations Keras will do through your dataset

`validation_data` is the data you'll which will be used to validate the accuracy.

`validation_steps` has the same meaning as `steps_per_epoch`.

Tricky part #2

So you try to start the training and you see… you see…



```
WARNING:tensorflow:Using a generator with
`use_multiprocessing=True` and multiple workers
may duplicate your data. Please consider using
the `tf.data.Dataset`.

WARNING:tensorflow:multiprocessing can
interact badly with TensorFlow, causing
nondeterministic deadlocks. For high
performance data pipelines tf.data is
recommended.
```

Image author: Denis Shilov (that's me)

Now let's go deeper: we will use `tf.data.Dataset`. It is a special thing to handle the Datasets: actually, that's pretty self-explanatory too.

There is an easy way to do this `Dataset` out of our `generator`.

We pass this lambda with `generator`, as it should be an executable thing and we need to pass some arguments inside the `generator`.

`output_types` is what your `generator` produces both in input and in output. Replace it with yours.

`output_shapes` is the shapes of input and output. Replace it with yours.

Important: **`NOT`** `[[None, 129], [None, 3]]`, **`NOT`** `[(None, 129), (None, 3)]`.
`It is` `([None, 129], [None, 3])`.

We set None, as we don't want to set the first dimension.

Last step

Replace `x` in `.fit` to be equal to `train_dataset` and `validation_data` to be equal to `test_dataset`.

Now start the training and it should work fine.