ericjang / **tdb**

Watch 74 | Star 1,188 | Fork 128

Code | Issues 8 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

Interactive, node-by-node debugging and visualization for TensorFlow

19 commits | **1** branch | **4** releases | **1** contributor | Apache-2.0

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

ericjang committed on **GitHub** Update README.md — Latest commit `5e78b5d` on 27 Jan

notebooks	test fix	a year ago
tdb	test fix	a year ago
tdb_ext	preparing setup.py	a year ago
.gitignore	setuptools fix	a year ago
AUTHORS	initial commit	a year ago
LICENSE	initial commit	a year ago
README.md	Update README.md	3 months ago
setup.cfg	initial commit	a year ago
setup.py	added README	a year ago

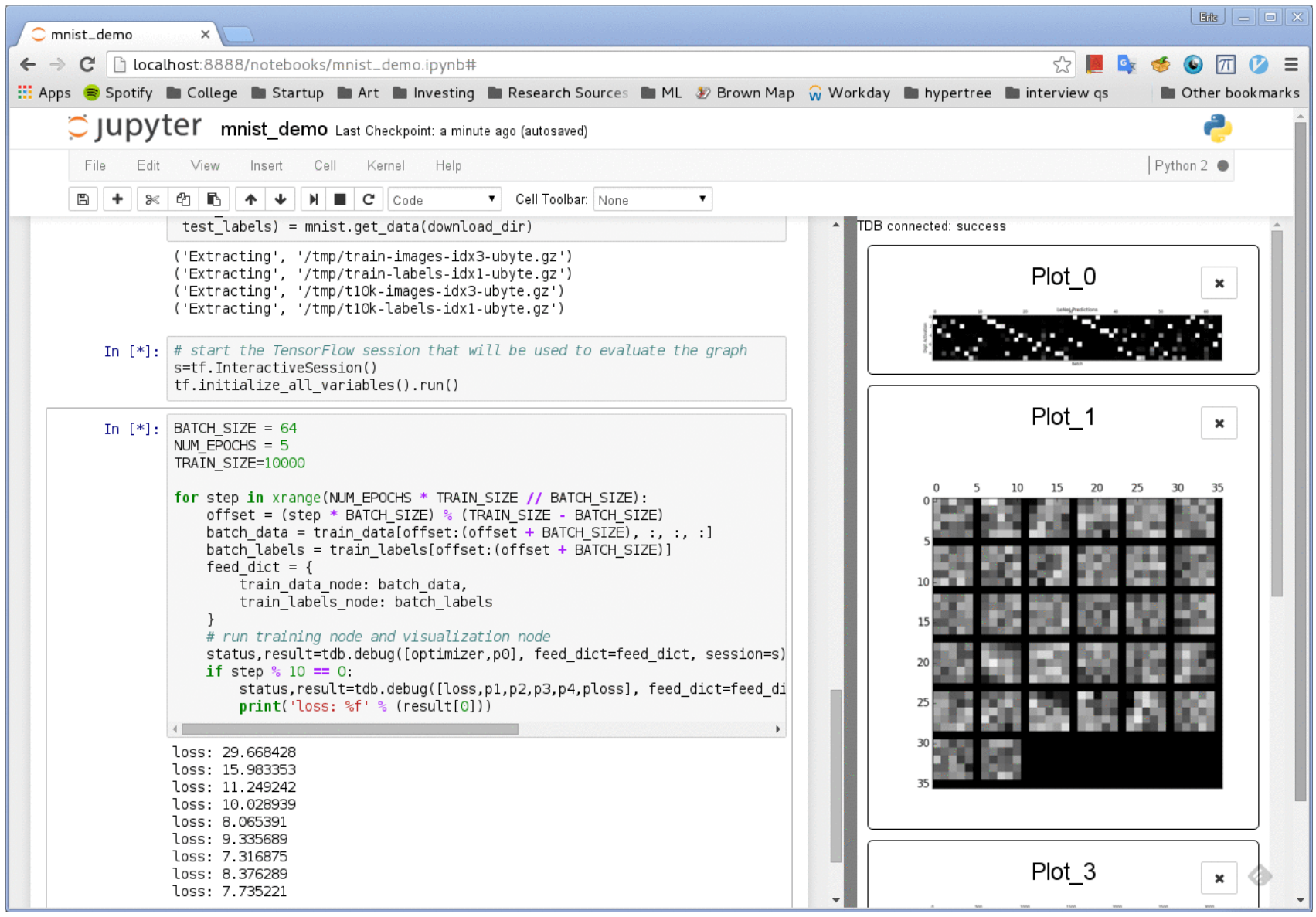
README.md

TDB

*Note: This project is no longer actively being maintained. Please check out the official tfdbg debugger

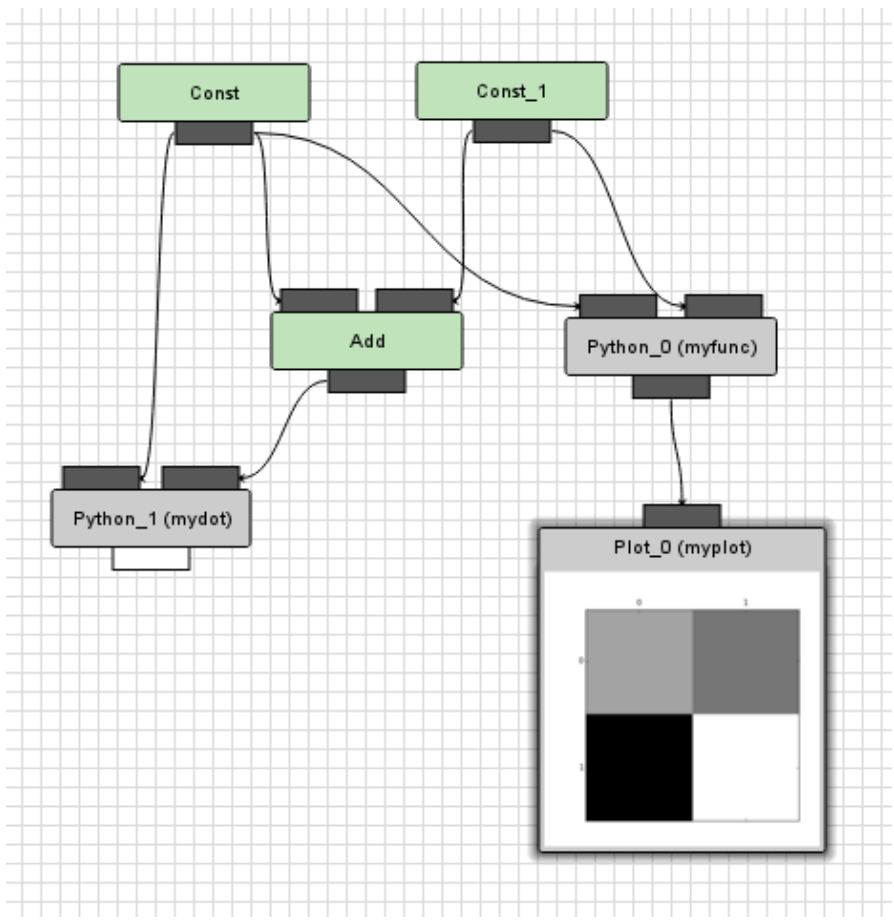
TensorDebugger (TDB) is a visual debugger for deep learning. It extends TensorFlow (Google's Deep Learning framework) with breakpoints + real-time visualization of the data flowing through the computational graph.

Video Demo



Specifically, TDB is the combination of a Python library and a Jupyter notebook extension, built around Google's TensorFlow framework. Together, these extend TensorFlow with the following features:

- **Breakpoints**: Set breakpoints on Ops and Tensors in the graph. Graph execution is paused on breakpoints and resumed by the user (via `tdb.c()`) Debugging features can be used with or without the visualization frontend.
- **Arbitrary Summary Plots**: Real-time visualization of high-level information (e.g. histograms, gradient magnitudes, weight saturation) while the network is being trained. Supports arbitrary, user-defined plot functions.
- **Flexible**: Mix user-defined Python and plotting functions with TensorFlow Nodes. These take in `tf.Tensors` and output placeholder nodes to be plugged into TensorFlow nodes. The below diagram illustrates how TDB nodes can be mixed with the TensorFlow graph.



Motivations

Modern machine learning models are parametrically complex and require considerable intuition to fine-tune properly.

In particular, Deep Learning methods are especially powerful, but hard to interpret in regards to their capabilities and learned representations.

Can we enable better understanding of how neural nets learn, without having to change model code or sacrifice performance? Can I finish my thesis on time?

TDB addresses these challenges by providing run-time visualization tools for neural nets. Real-time visual debugging allows training bugs to be detected sooner, thereby reducing the iteration time needed to build the right model.

Setup

To install the Python library,

```
pip install tfdebugger
```

To install the Jupyter Notebook extension, run the following in a Python terminal (you will need to have IPython or Jupyter installed)

```
import notebook.nbextensions
import urllib
import zipfile
SOURCE_URL = 'https://github.com/ericjang/tdb/releases/download/tdb_ext_v0.1/tdb_ext.zip'
urllib.urlretrieve(SOURCE_URL, 'tdb_ext.zip')
with zipfile.ZipFile('tdb_ext.zip', "r") as z:
    z.extractall("")
notebook.nbextensions.install_nbextension('tdb_ext',user=True)
```

Tutorial

To get started, check out the MNIST Visualization Demo. More examples and visualizations to come soon.

User Guide

Debugging

Start

```
status,result=tdb.debug(evals,feed_dict=None,breakpoints=None,break_immediately=False,session=None)
```

`debug()` behaves just like Tensorflow's Session.run(). If a breakpoint is hit, `status` is set to 'PAUSED' and `result` is set to `None`. Otherwise, `status` is set to 'FINISHED' and `result` is set to a list of evaluated values.

Continue

```
status,result=tdb.c()
```

Continues execution of a paused session, until the next breakpoint or end. Behaves like `debug`.

Step

```
status,result=tdb.s()
```

Evaluate the next node, then pause immediately to await user input. Unless we have reached the end of the execution queue, `status` will remain 'PAUSED'. `result` is set to the value of the node we just evaluated.

Where

```
q=tdb.get_exe_queue()
```

Return value: list of remaining nodes to be evaluated, in order.

print

```
val=tdb.get_value(node)
```

Returns value of an evaluated node (a string name or a tf.Tensor)

Custom Nodes

TDB supports 2 types of custom Ops:

Python

Here is an example of mixing tdb.PythonOps with TensorFlow.

Define the following function:

```
def myadd(ctx,a,b):
        return a+b


a=tf.constant(2)
b=tf.constant(3)
c=tdb.python_op(myadd,inputs=[a,b],outputs=[tf.placeholder(tf.int32)]) # a+b
d=tf.neg(c)
status,result=tdb.debug([d], feed_dict=None, breakpoints=None, break_immediately=False)
```

When `myadd` gets evaluated, `ctx` is the instance of the PythonOp that it belongs to. You can use ctx to store state information (i.e. accumulate loss history).

Plotting

PlotOps are a special instance of PythonOp that send graphical output to the frontend.

This only works with Matplotlib at the moment, but other plotting backends (Seaborn, Bokeh, Plotly) are coming soon.

```
def watch_loss(ctx,loss):
  if not hasattr(ctx, 'loss_history'):
    ctx.loss_history=[]
  ctx.loss_history.append(loss)
  plt.plot(ctx.loss_history)
  plt.ylabel('loss')


ploss=tdb.plot_op(viz.watch_loss,inputs=[loss])
```

Refer to the MNIST Visualization Demo for more examples. You can also find more examples in the tests/ directory.

FAQ

Is TDB affiliated with TensorFlow?

No, but it is built on top of it.

What is TDB good for?

TDB is especially useful at the model prototyping stage and verifying correctness in an intuitive manner. It is also useful for high-level visualization of hidden layers during training.

How is TDB different from TensorBoard?

TensorBoard is a suite of visualization tools included with Tensorflow. Both TDB and TensorBoard attach auxiliary nodes to the TensorFlow graph in order to inspect data.

TensorBoard cannot be used concurrently with running a TensorFlow graph; log files must be written first. TDB interfaces directly with the execution of a TensorFlow graph, and allows for stepping through execution one node at a time.

Out of the box, TensorBoard currently only supports logging for a few predefined data formats.

TDB is to TensorBoard as GDB is to printf. Both are useful in different contexts.

License

Apache 2.0