TensorBoard

TensorBoard is a suite of web applications for inspecting and understanding your TensorFlow runs and graphs. TensorBoard currently supports five visualizations: scalars, images, audio, histograms, and the graph.

This README gives an overview of key concepts in TensorBoard, as well as how to interpret the visualizations TensorBoard provides. For an in-depth example of using TensorBoard, see the tutorial: TensorBoard: Visualizing Learning. For in-depth information on the Graph Visualizer, see this tutorial: TensorBoard: Graph Visualization.

Usage

Before running TensorBoard, make sure you have generated summary data in a log directory by creating a summary writer:

```
# sess.graph contains the graph definition; that enables the Graph Visualizer.

file_writer = tf.summary.FileWriter('/path/to/logs', sess.graph)
```

For more details, see this tutorial. Once you have event files, run TensorBoard and provide the log directory. If you're using a precompiled TensorFlow package (e.g. you installed via pip), run:

```
tensorboard --logdir=path/to/logs
```

Or, if you are building from source:

```
bazel build tensorflow/tensorboard:tensorboard
./bazel-bin/tensorflow/tensorboard/tensorboard --logdir=path/to/logs
```

This should print that TensorBoard has started. Next, connect to http://localhost:6006.

TensorBoard requires a `logdir` to read logs from. For info on configuring TensorBoard, run `tensorboard --help`.

TensorBoard can be used in Google Chrome or Firefox. Other browsers might work, but there may be bugs or performance issues.

Key Concepts

Summary Ops: How TensorBoard gets data from TensorFlow

The first step in using TensorBoard is acquiring data from your TensorFlow run. For this, you need summary ops. Summary ops are ops, like `tf.matmul` or `tf.nn.relu`, which means they take in tensors, produce tensors, and are evaluated from within a TensorFlow graph. However, summary ops have a twist: the Tensors they produce contain serialized protobufs, which are written to disk and sent to TensorBoard. To visualize the summary data in TensorBoard, you should evaluate the summary op, retrieve the result, and then write that result to disk using a summary.FileWriter. A full explanation, with examples, is in the tutorial.

Tags: Giving names to data

When you make a summary op, you will also give it a `tag`. The tag is basically a name for the data recorded by that op, and will be used to organize the data in the frontend. The scalar and histogram dashboards organize data by tag, and group the tags into folders according to a directory/like/hierarchy. If you have a lot of tags, we recommend grouping them with slashes.

Event Files & LogDirs: How TensorBoard loads the data

`summary.FileWriters` take summary data from TensorFlow, and then write them to a specified directory, known as the `logdir`. Specifically, the data is written to an append-only record dump that will have "tfevents" in the filename. TensorBoard reads data from a full directory, and organizes it into the history of a single TensorFlow execution.

Why does it read the whole directory, rather than an individual file? You might have been using supervisor.py to run your model, in which case if TensorFlow crashes, the supervisor will restart it from a checkpoint. When it restarts, it will start writing to a new events file, and TensorBoard will stitch the various event files together to produce a consistent history of what happened.

Runs: Comparing different executions of your model

You may want to visually compare multiple executions of your model; for example, suppose you've changed the hyperparameters and want to see if its converging faster. TensorBoard enables this through different "runs". When TensorBoard is passed a `logdir` at startup, it recursively walks the directory tree rooted at `logdir` looking for subdirectories that contain tfevents data. Every time it encounters such a subdirectory, it loads it as a new `run`, and the frontend will organize the data accordingly.

For example, here is a well-organized TensorBoard log directory, with two runs, "run1" and "run2".

```
/some/path/mnist_experiments/
/some/path/mnist_experiments/run1/
/some/path/mnist_experiments/run1/events.out.tfevents.1456525581.name
/some/path/mnist_experiments/run1/events.out.tfevents.1456525585.name
/some/path/mnist_experiments/run2/
/some/path/mnist_experiments/run2/events.out.tfevents.1456525385.name
/tensorboard --logdir=/some/path/mnist_experiments
```

You may also pass a comma separated list of log directories, and TensorBoard will watch each directory. You can also assign names to individual log directories by putting a colon between the name and the path, as in

```
tensorboard --logdir=name1:/path/to/logs/1,name2:/path/to/logs/2
```

The Visualizations

Events Dashboard

TensorBoard's Events Dashboard visualizes scalar statistics that vary over time; for example, you might want to track the model's loss or learning rate. As described in *Key Concepts*, you can compare multiple runs, and the data is organized by tag. The line charts have the following interactions:

- Clicking on the small blue icon in the lower-left corner of each chart will expand the chart
- Dragging a rectangular region on the chart will zoom in
- Double clicking on the chart will zoom out
- Mousing over the chart will produce crosshairs, with data values recorded in the run-selector on the left.

Additionally, you can create new folders to organize tags by writing regular expressions in the box in the top-left of the dashboard.

Distribution Dashboard

The Distribution Dashboard is for visualizing how the statistical distribution of a Tensor has varied over time. It visualizes data recorded via a tf.summary.histogram. Right now, its name is a bit of a misnomer, as it doesn't show histograms; instead, it shows some high-level statistics on a distribution. Each line on the chart represents a percentile in the distribution over the data: for example, the bottom line shows how the minimum value has changed over time, and the line in the middle shows how the median has changed. Reading from top to bottom, the lines have the following meaning: `[maximum, 93%, 84%, 69%, 50%, 31%, 16%, 7%, minimum]`

These percentiles can also be viewed as standard deviation boundaries on a normal distribution: `[maximum, μ+1.5σ, μ+σ, μ+0.5σ, μ, μ-0.5σ, μ-σ, μ-1.5σ, minimum]` so that the colored regions, read from inside to outside, have widths `[σ, 2σ, 3σ]` respectively.

This histogram visualization is a bit weird, and cannot meaningfully represent multimodal distributions. We are currently working on a true-histogram replacement.

Image Dashboard

The Image Dashboard can display pngs that were saved via a tf.summary.image. The dashboard is set up so that each row corresponds to a different tag, and each column corresponds to a run. Since the image dashboard supports arbitrary pngs, you can use this to embed custom visualizations (e.g. matplotlib scatterplots) into TensorBoard. This dashboard always shows you the latest image for each tag.

Audio Dashboard

The Audio Dashboard can embed playable audio widgets for audio saved via a tf.summary.audio. The dashboard is set up so that each row corresponds to a different tag, and each column corresponds to a run. This dashboard always embeds the latest audio for each tag.

Graph Explorer

The Graph Explorer can visualize a TensorBoard graph, enabling inspection of the TensorFlow model. To get best use of the graph visualizer, you should use name scopes to hierarchically group the ops in your graph - otherwise, the graph may be difficult to decipher. For more information, including examples, see the graph visualizer tutorial.

Frequently Asked Questions

My TensorBoard isn't showing any data! What's wrong?

The first thing to do is ensure that TensorBoard is properly loading data from the correct directory. Launch `tensorboard --logdir=DIRECTORY_PATH --debug` and look for output of the form

```
INFO:tensorflow:TensorBoard path_to_run is: {'DIRECTORY_PATH': None}
```

Verify that the DIRECTORY_PATH TensorBoard is looking at is the path you expect. (Note: There's a known issue where TensorBoard does not handle paths starting in ~ properly).

If you're loading from the proper path, make sure that event files are present. TensorBoard will recursively walk its logdir, it's fine if the data is nested under a subdirectory. Try running the command:

```
find DIRECTORY_PATH | grep tfevents
```

If you have at least one result, then TensorBoard should be able to load data.

Finally, let's make sure that the event files actually have data. Run tensorboard in inspector mode to inspect the contents of your event files.

```
tensorboard --inspect --logdir=DIRECTORY_PATH
```

If after running this procedure, it's still not working, please file an issue on GitHub. It will be much easier for us to debug it if you provide an event file that isn't working.

TensorBoard is showing only some of my data, or isn't properly updating!

This issue usually comes about because of how TensorBoard iterates through the `tfevents` files: it progresses through the events file in timestamp order, and only reads one file at a time. Let's suppose we have files with timestamps `a` and `b`, where `a<b`. Once TensorBoard has read all the events in `a`, it will never return to it, because it assumes any new events

are being written in the more recent file. This could cause an issue if, for example, you have two `FileWriters` simultaneously writing to the same directory. If you have multiple summary writers, each one should be writing to a separate directory.

Does TensorBoard support multiple or distributed summary writers?

No. TensorBoard expects that only one events file will be written to at a time, and multiple summary writers means multiple events files. If you are running a distributed TensorFlow instance, we encourage you to designate a single worker as the "chief" that is responsible for all summary processing. See supervisor.py for an example.

I'm seeing data overlapped on itself! What gives?

If you are seeing data that seems to travel backwards through time and overlap with itself, there are a few possible explanations.

- You may have multiple execution of TensorFlow that all wrote to the same log directory. Please have each TensorFlow run write to its own logdir.
- You may have a have a bug in your code where the global_step variable (passed to `FileWriter.add_summary`) is being maintained incorrectly.
- It may be that your TensorFlow job crashed, and was restarted from an earlier checkpoint. See *How to handle TensorFlow restarts*, below.

As a workaround, try changing the x-axis display in TensorBoard from `steps` to `wall_time` . This will frequently clear up the issue.

How should I handle TensorFlow restarts?

TensorFlow is designed with a mechanism for graceful recovery if a job crashes or is killed: TensorFlow can periodically write model checkpoint files, which enable you to restart TensorFlow without losing all your training progress.

However, this can complicate things for TensorBoard; imagine that TensorFlow wrote a checkpoint at step `a` , and then continued running until step `b` , and then crashed and restarted at timestamp `a` . All of the events written between `a` and `b` were "orphaned" by the restart event and should be removed.

To facilitate this, we have a `SessionLog` message in `tensorflow/core/util/event.proto` which can record `SessionStatus.START` as an event; like all events, it may have a `step` associated with it. If TensorBoard detects a `SessionStatus.START` event with step `a` , it will assume that every event with a step greater than `a` was orphaned, and it will discard those events. This behavior may be disabled with the flag `--purge_orphaned_data=false` (in versions after 0.7).

How can I export data from TensorBoard?

If you'd like to export data to visualize elsewhere (e.g. iPython Notebook), that's possible too. You can directly depend on the underlying classes that TensorBoard uses for loading data: `python/summary/event_accumulator.py` (for loading data from a single run) or `python/summary/event_multiplexer.py` (for loading data from multiple runs, and keeping it organized). These classes load groups of event files, discard data that was "orphaned" by TensorFlow crashes, and organize the data by tag.

As another option, there is a script (`tensorboard/scripts/serialize_tensorboard.py`) which will load a logdir just like TensorBoard does, but write all of the data out to disk as json instead of starting a server. This script is setup to make "fake TensorBoard backends" for testing, so it is a bit rough around the edges.

Can I overlap multiple plots?

Right now, you can overlap plots only if they are from different runs, and both have the same tag name.

Can I create scatterplots (or other custom plots)?

This isn't yet possible. As a workaround, you could create your custom plot in your own code (e.g. matplotlib) and then write it into an `SummaryProto` (`core/framework/summary.proto`) and add it to your `FileWriter` . Then, your custom plot will appear in the TensorBoard image tab.

Is my data being downsampled? Am I really seeing all the data?

TensorBoard uses reservoir sampling to downsample your data so that it can be loaded into RAM. You can modify the number of elements it will keep per tag in tensorboard/backend/server.py.

I get a network security popup every time I run TensorBoard on a mac!

This is because by default, TensorBoard serves on host `0.0.0.0` which is publicly accessible. You can stop the popups by specifying `--host=localhost` at startup.

How can I develop TensorBoard?

See tensorflow/tensorboard/DEVELOPMENT.md.

I have a different issue that wasn't addressed here!

First, try searching our GitHub issues and Stack Overflow. It may be that someone else has already had the same issue or question.

If you have a bug, please file a GitHub issue. If the bug is related to your specific data (e.g. the events aren't loading properly), please do both of the following things to make it easier for us to debug and fix:

- Run tensorboard in --inspect mode and copy paste the debug output.
- Upload some events files that will reproduce the issue.

If you have a feature request, please file a GitHub issue.

General usage questions should go to Stack Overflow.