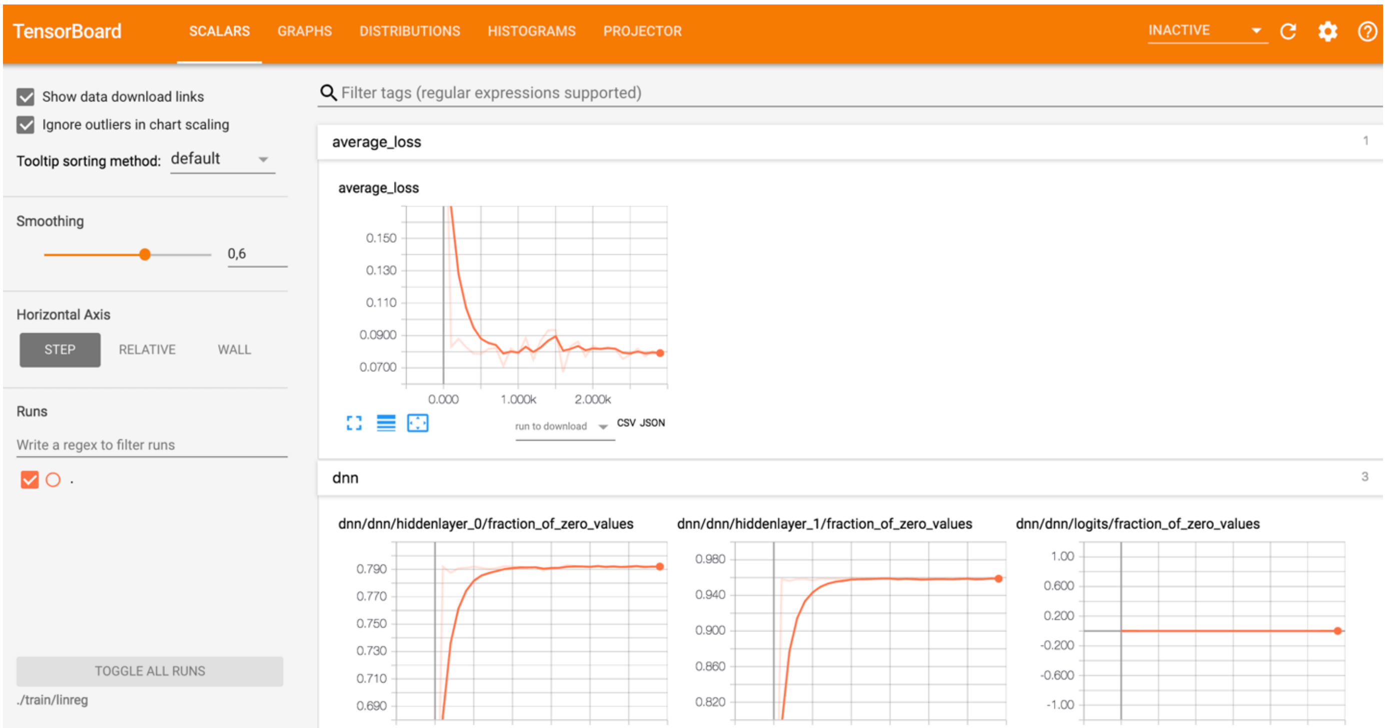
Tensorboard Tutorial: Graph Visualization with Example

What is TensorBoard?

Tensorboard is the interface used to visualize the graph and other tools to understand, debug, and optimize the model.

Example

The image below comes from the graph you will generate in this tutorial. It is the main panel:



☑ Show data download links

☑ Ignore outliers in chart scaling

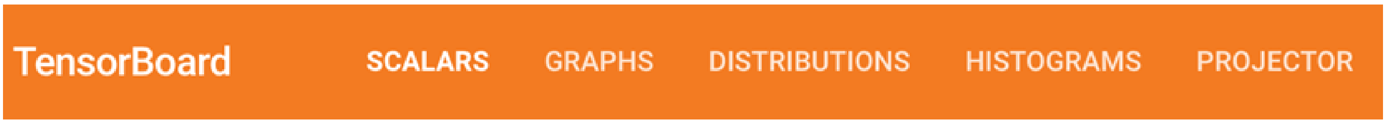
Tooltip sorting method: default

Smoothing 0,6

./train/linreg

(/images/1/080618_0516_Tensorboard1.png)

From the picture below, you can see the panel of Tensorboard. The panel contains different tabs, which are linked to the level of information you add when you run the model.



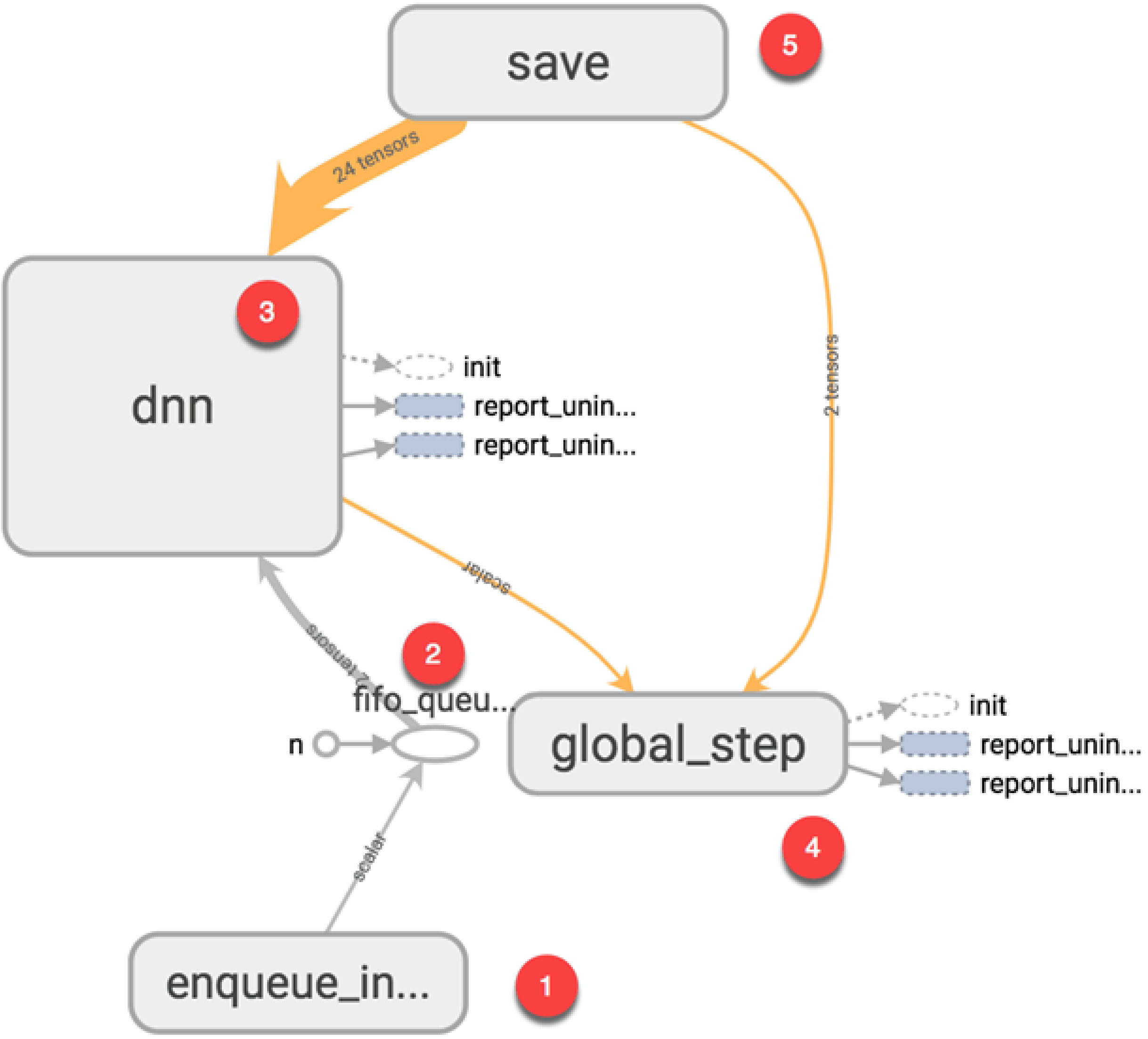
(/images/1/080618_0516_Tensorboard2.png)

- Scalars: Show different useful information during the model training
- Graphs: Show the model
- Histogram: Display weights with a histogram
- Distribution: Display the distribution of the weight
- Projector: Show Principal component analysis and T-SNE algorithm. The technique uses for dimensionality reduction

During this tutorial, you will train a simple deep learning model. You will learn how it works in a future tutorial.

If you look at the graph, you can understand how the model work.

1. Enqueue the data to the model: Push an amount of data equal to the batch size to the model, i.e., Number of data feed after each iteration
2. Feed the data to the Tensors
3. Train the model
4. Display the number of batches during the training. Save the model on the disk.

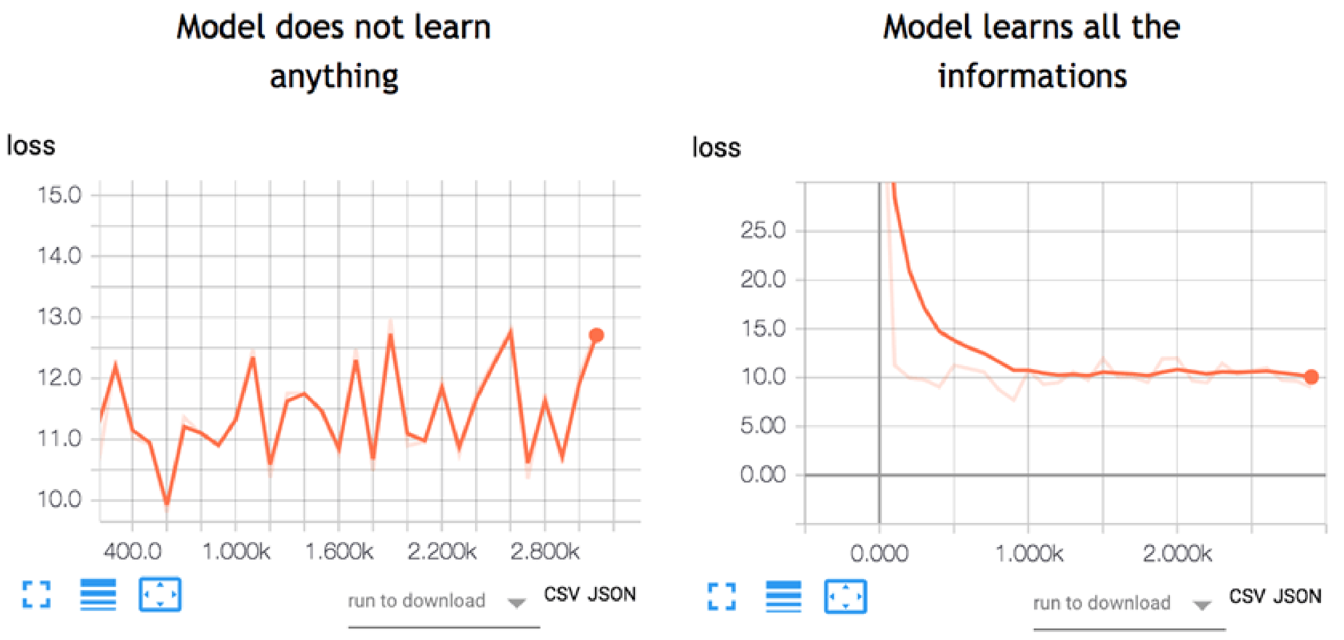


(/images/1/080618_0516_Tensorboard3.png)

The basic idea behind tensorboard is that neural network can be something known as a black box and we need a tool to inspect what's inside this box. You can imagine tensorboard as a flashlight to start dive into the neural network.

It helps to understand the dependencies between operations, how the weights are computed, displays the loss function and much other useful information. When you bring all these pieces of information together, you have a great tool to debug and find how to improve the model.

To give you an idea of how useful the graph can be, look at the picture below:



(/images/1/080618_0516_Tensorboard4.png)

A neural network decides how to connect the different "neurons" and how many layers before the model can predict an outcome. Once you have defined the architecture, you not only need to train the model but also a metrics to compute the accuracy of the prediction. This metric is referred to as a **loss function.** The objective is to minimize the loss function. In different words, it means the model is making fewer errors. All machine learning algorithms will repeat many times the computations until the loss reach a flatter line. To minimize this loss function, you need to define a **learning rate.** It is the speed you want the model to learn. If you set a learning rate too high, the model does not have time to learn anything. This is the case in the left picture. The line is moving up and down, meaning the model predicts with pure guess the outcome. The picture on the right shows that the loss is decreasing over iteration until the curve got flatten, meaning the model found a solution.

TensorBoard is a great tool to visualize such metrics and highlight potential issues. The neural network can take hours to weeks before they find a solution. TensorBoard updates the metrics very often. In this case, you don't need to wait until the end to see if the model trains correctly. You can open TensorBoard check how the training is going and make the appropriate change if necessary.

How to use TensorBoard?

In this tutorial, you will learn how to open TensorBoard from the terminal for MacOS and the Command line for Windows.

The code will be explained in a future tutorial, the focus here is on TensorBoard.

First, you need to import the libraries you will use during the training

```
## Import the library
import tensorflow as tf
import numpy as np
```

You create the data. It is an array of 10000 rows and 5 columns

```
X_train = (np.random.sample((10000,5)))
y_train =  (np.random.sample((10000,1)))
X_train.shape
```

Output

```
(10000, 5)
```

The codes below transform the data and create the model.

Note that the learning rate is equal to 0.1. If you change this rate to a higher value, the model will not find a solution. This is what happened on the left side of the above picture.

During most of the TensorFlow tutorials, you will use TensorFlow estimator. This is TensorFlow API that contains all the mathematical computations.

To create the log files, you need to specify the path. This is done with the argument model_dir.

In the example below, you store the model inside the working directory, i.e., where you store the notebook or python file. Inside this path, TensorFlow will create a folder called train with a child folder name linreg.

```
feature_columns = [
      tf.feature_column.numeric_column('x', shape=X_train.shape[1:])]
DNN_reg = tf.estimator.DNNRegressor(feature_columns=feature_columns,
# Indicate where to store the log file
     model_dir='train/linreg',
     hidden_units=[500, 300],
     optimizer=tf.train.ProximalAdagradOptimizer(
          learning_rate=0.1,
          l1_regularization_strength=0.001
      )
)
```

Output

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': 'train/linreg', '_tf_random_seed': Non
e, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_s
ecs': 600, '_session_config': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_eve
ry_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute': None, '_servi
ce': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec objec
t at 0x1818e63828>, '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster':
0, '_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas':
0, '_num_worker_replicas': 1}
```

The last step consists to train the model. During the training, TensorFlow writes information in the model directory.

```
# Train the estimator
train_input = tf.estimator.inputs.numpy_input_fn(
     x={"x": X_train},
     y=y_train, shuffle=False,num_epochs=None)
DNN_reg.train(train_input,steps=3000)
```

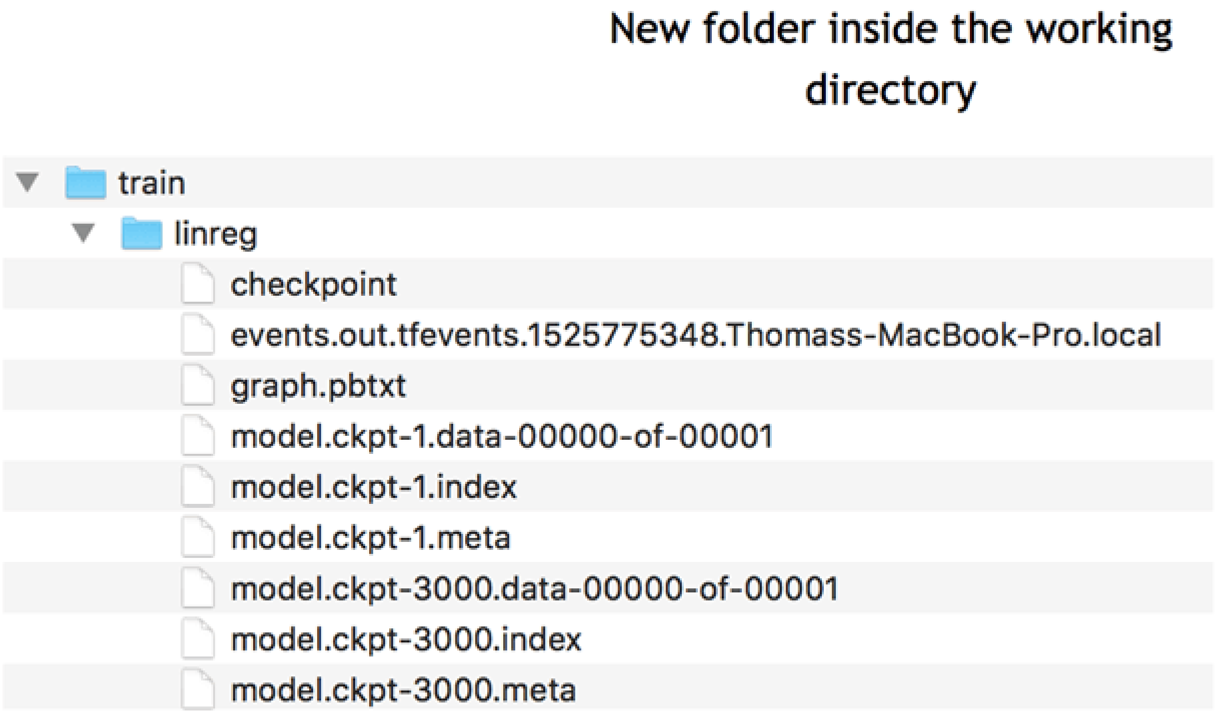
Output

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1 into train/linreg/model.ckpt.
INFO:tensorflow:loss = 40.060104, step = 1
INFO:tensorflow:global_step/sec: 197.061
INFO:tensorflow:loss = 10.62989, step = 101 (0.508 sec)
INFO:tensorflow:global_step/sec: 172.487
INFO:tensorflow:loss = 11.255318, step = 201 (0.584 sec)
INFO:tensorflow:global_step/sec: 193.295
INFO:tensorflow:loss = 10.604872, step = 301 (0.513 sec)
INFO:tensorflow:global_step/sec: 175.378
INFO:tensorflow:loss = 10.090343, step = 401 (0.572 sec)
INFO:tensorflow:global_step/sec: 209.737
INFO:tensorflow:loss = 10.057928, step = 501 (0.476 sec)
INFO:tensorflow:global_step/sec: 171.646
INFO:tensorflow:loss = 10.460144, step = 601 (0.583 sec)
INFO:tensorflow:global_step/sec: 192.269
INFO:tensorflow:loss = 10.529617, step = 701 (0.519 sec)
INFO:tensorflow:global_step/sec: 198.264
INFO:tensorflow:loss = 9.100082, step = 801 (0.504 sec)
INFO:tensorflow:global_step/sec: 226.842
INFO:tensorflow:loss = 10.485607, step = 901 (0.441 sec)
INFO:tensorflow:global_step/sec: 152.929
INFO:tensorflow:loss = 10.052481, step = 1001 (0.655 sec)
INFO:tensorflow:global_step/sec: 166.745
INFO:tensorflow:loss = 11.320213, step = 1101 (0.600 sec)
INFO:tensorflow:global_step/sec: 161.854
INFO:tensorflow:loss = 9.603306, step = 1201 (0.619 sec)
INFO:tensorflow:global_step/sec: 179.074
INFO:tensorflow:loss = 11.110269, step = 1301 (0.556 sec)
INFO:tensorflow:global_step/sec: 202.776
INFO:tensorflow:loss = 11.929443, step = 1401 (0.494 sec)
INFO:tensorflow:global_step/sec: 144.161
INFO:tensorflow:loss = 11.951693, step = 1501 (0.694 sec)
INFO:tensorflow:global_step/sec: 154.144
INFO:tensorflow:loss = 8.620987, step = 1601 (0.649 sec)
INFO:tensorflow:global_step/sec: 151.094
INFO:tensorflow:loss = 10.666125, step = 1701 (0.663 sec)
INFO:tensorflow:global_step/sec: 193.644
INFO:tensorflow:loss = 11.0349865, step = 1801 (0.516 sec)
INFO:tensorflow:global_step/sec: 189.707
INFO:tensorflow:loss = 9.860596, step = 1901 (0.526 sec)
INFO:tensorflow:global_step/sec: 176.423
INFO:tensorflow:loss = 10.695, step = 2001 (0.567 sec)
INFO:tensorflow:global_step/sec: 213.066
INFO:tensorflow:loss = 10.426752, step = 2101 (0.471 sec)
INFO:tensorflow:global_step/sec: 220.975
INFO:tensorflow:loss = 10.594796, step = 2201 (0.452 sec)
INFO:tensorflow:global_step/sec: 219.289
```

```
INFO:tensorflow:loss = 10.4212265, step = 2301 (0.456 sec)
INFO:tensorflow:global_step/sec: 215.123
INFO:tensorflow:loss = 9.668612, step = 2401 (0.465 sec)
INFO:tensorflow:global_step/sec: 175.65
INFO:tensorflow:loss = 10.009649, step = 2501 (0.569 sec)
INFO:tensorflow:global_step/sec: 206.962
INFO:tensorflow:loss = 10.477722, step = 2601 (0.483 sec)
INFO:tensorflow:global_step/sec: 229.627
INFO:tensorflow:loss = 9.877638, step = 2701 (0.435 sec)
INFO:tensorflow:global_step/sec: 195.792
INFO:tensorflow:loss = 10.274586, step = 2801 (0.512 sec)
INFO:tensorflow:global_step/sec: 176.803
INFO:tensorflow:loss = 10.061047, step = 2901 (0.566 sec)
INFO:tensorflow:Saving checkpoints for 3000 into train/linreg/model.ckpt.
INFO:tensorflow:Loss for final step: 10.73032.

<tensorflow.python.estimator.canned.dnn.DNNRegressor at 0x1818e63630>
```

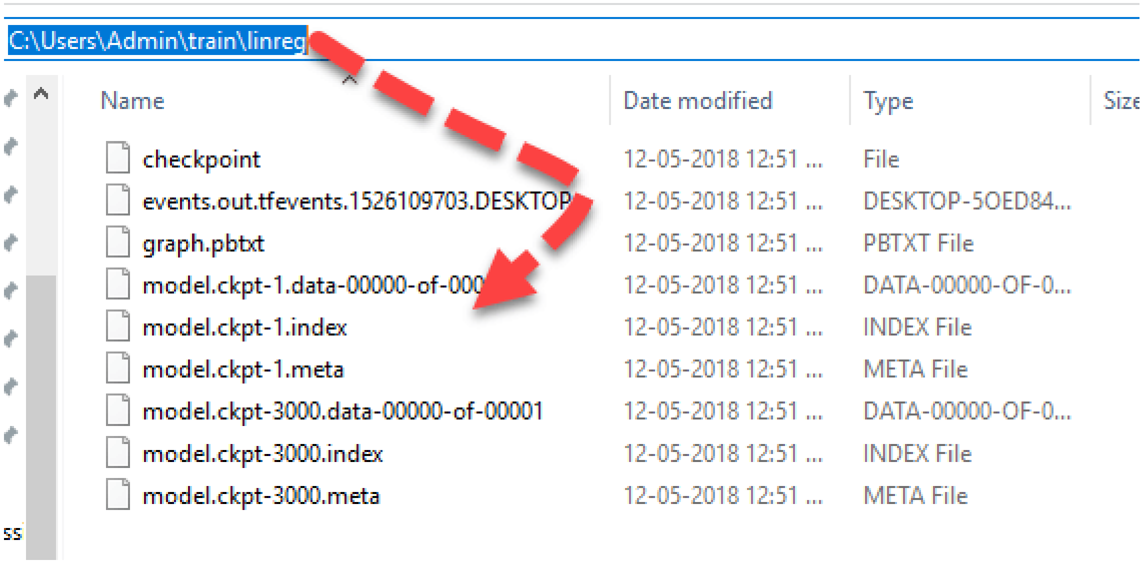
For MacOS user



```
▼ train
    ▼ linreg
        checkpoint
        events.out.tfevents.1525775348.Thomass-MacBook-Pro.local
        graph.pbtxt
        model.ckpt-1.data-00000-of-00001
        model.ckpt-1.index
        model.ckpt-1.meta
        model.ckpt-3000.data-00000-of-00001
        model.ckpt-3000.index
        model.ckpt-3000.meta
```

(/images/1/080618_0516_Tensorboard5.png)

For Windows user



C:\Users\Admin\train\linreg

Name	Date modified	Type	Size
checkpoint	12-05-2018 12:51 ...	File	
events.out.tfevents.1526109703.DESKTOP	12-05-2018 12:51 ...	DESKTOP-5OED84...	
graph.pbtxt	12-05-2018 12:51 ...	PBTXT File	
model.ckpt-1.data-00000-of-000	12-05-2018 12:51 ...	DATA-00000-OF-0...	
model.ckpt-1.index	12-05-2018 12:51 ...	INDEX File	
model.ckpt-1.meta	12-05-2018 12:51 ...	META File	
model.ckpt-3000.data-00000-of-00001	12-05-2018 12:51 ...	DATA-00000-OF-0...	
model.ckpt-3000.index	12-05-2018 12:51 ...	INDEX File	
model.ckpt-3000.meta	12-05-2018 12:51 ...	META File	

(/images/1/080618_0516_Tensorboard6.png)

You can see this information in the TensorBoard.

Now that you have the log events written, you can open Tensorboard. Tensorboad runs on port 6006 (Jupyter runs on port 8888). You can use the Terminal for MacOs user or Anaconda prompt for Windows user.

For MacOS user

```
# Different for you
cd /Users/Guru99/tuto_TF
source activate hello-tf!
```

The notebook is stored in the path /Users/Guru99/tuto_TF

For Windows users

```
cd C:\Users\Admin\Anaconda3
activate hello-tf
```

The notebook is stored in the path C:\Users\Admin\Anaconda3

To launch Tensorboard, you can use this code

For MacOS user

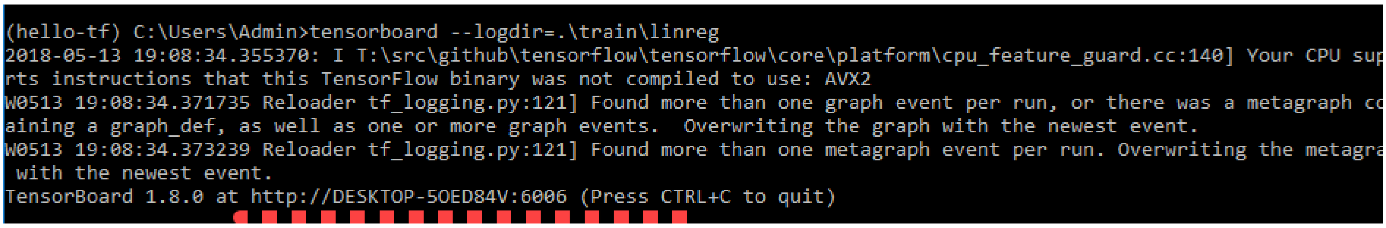
```
tensorboard --logdir=./train/linreg
```

For Windows users

```
tensorboard --logdir=.\train\linreg
```

Tensorboard is located in this URL: http://localhost:6006

It could also be located at the following location.

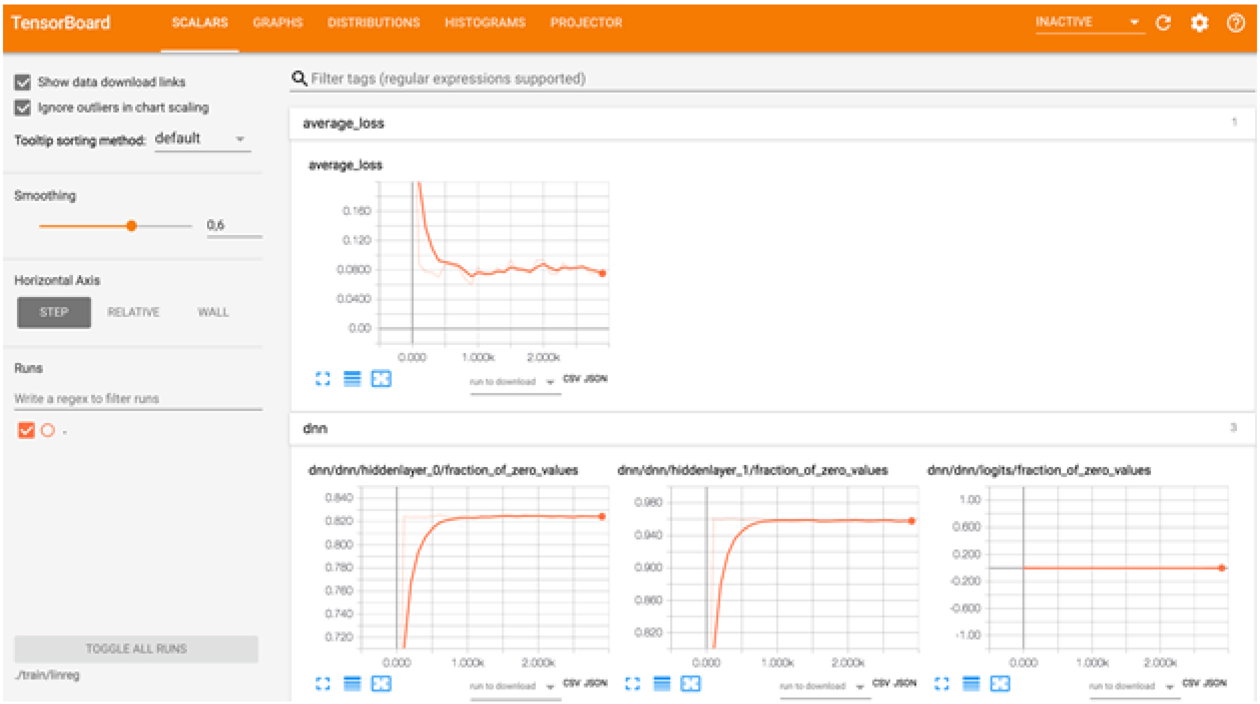


```
(hello-tf) C:\Users\Admin>tensorboard --logdir=.\train\linreg
2018-05-13 19:08:34.355370: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU sup
rts instructions that this TensorFlow binary was not compiled to use: AVX2
W0513 19:08:34.371735 Reloader tf_logging.py:121] Found more than one graph event per run, or there was a metagraph co
aining a graph_def, as well as one or more graph events.  Overwriting the graph with the newest event.
W0513 19:08:34.373239 Reloader tf_logging.py:121] Found more than one metagraph event per run. Overwriting the metagra
 with the newest event.
TensorBoard 1.8.0 at http://DESKTOP-5OED84V:6006 (Press CTRL+C to quit)
```

(/images/1/080618_0516_Tensorboard7.png)

Copy and paste the URL into your favorite browser. You should see this:

Note that, we will learn how to read the graph in the tutorial dedicated to the deep learning.



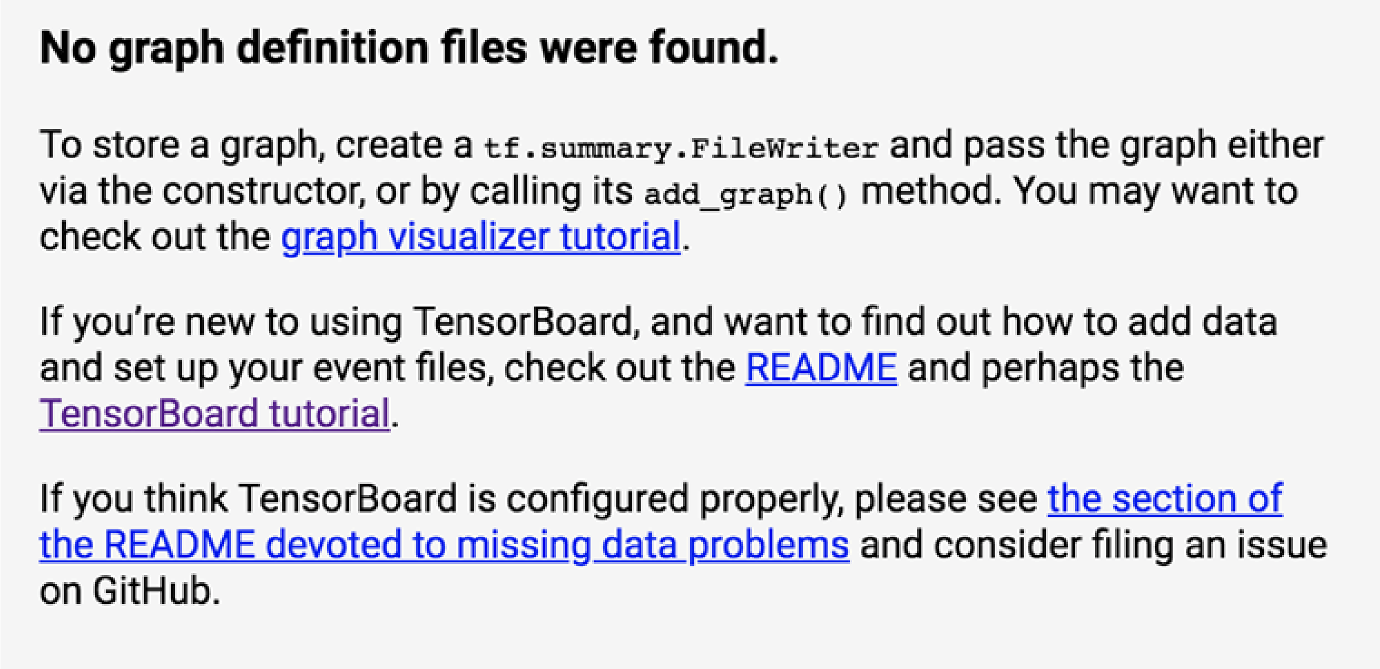
☑ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0,6

Horizontal Axis: STEP

(/images/1/080618_0516_Tensorboard8.png)

If you see something like this:



No graph definition files were found.

To store a graph, create a `tf.summary.FileWriter` and pass the graph either via the constructor, or by calling its `add_graph()` method. You may want to check out the graph visualizer tutorial.

If you're new to using TensorBoard, and want to find out how to add data and set up your event files, check out the README and perhaps the TensorBoard tutorial.

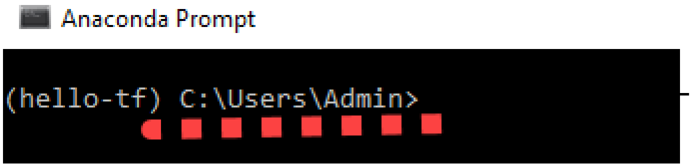
If you think TensorBoard is configured properly, please see the section of the README devoted to missing data problems and consider filing an issue on GitHub.

(/images/1/080618_0516_Tensorboard9.png)

It means Tensorboard cannot find the log file. Make sure you point the cd to the right path or double check if the log event has been creating. If not, re-run the code.

If you want to close TensorBoard Press CTRL+C

Hat Tip: Check your anaconda prompt for the current working directory,



```
(hello-tf) C:\Users\Admin>
```

(/images/1/080618_0516_Tensorboard10.png)

The log file should be created at C:\Users\Admin

Summary:

TensorBoard is a great tool to visualize your model. Besides, many metrics are displayed during the training, such as the loss, accuracy or weights.

To activate Tensorboard, you need to set the path of your file:

```
cd /Users/Guru99/tuto_TF
```

Activate Tensorflow's environment

```
activate hello-tf
```

Launch Tensorboard

```
tensorboard --logdir=.+ PATH
```

Prev (/tensor-tensorflow.html) **Report a Bug** Next (/python-pandas-tutorial.html)

YOU MIGHT LIKE:

DATA WAREHOUSING

(/datastage-tutorial.html)

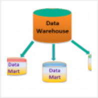
(/datastage-tutorial.html)

DataStage Tutorial: Beginner's Training

(/datastage-tutorial.html)

DATA WAREHOUSING

(/data-warehouse-vs-data-mart.html)



(/data-warehouse-vs-data-mart.html)

Data Warehouse vs Data Mart: Know the Difference

(/data-warehouse-vs-data-mart.html)

DATA WAREHOUSING

(/etl-extract-load-process.html)

(/etl-extract-load-process.html)

ETL (Extract, Transform, and Load) Process

(/etl-extract-load-process.html)

DATA WAREHOUSING

(/data-lake-vs-data-warehouse.html)

(/data-lake-vs-data-warehouse.html)

Data Lake vs Data Warehouse: What's the Difference?

(/data-lake-vs-data-warehouse.html)

DATA WAREHOUSING

(/data-warehousing.html)

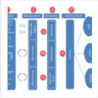
(/data-warehousing.html)

What Is Data Warehousing? Types, Definition & Example

(/data-warehousing.html)

TABLEAU

(/tableau-architecture.html)



(/tableau-architecture.html)

Tableau Architecture & Components

(/tableau-architecture.html)

TensorFlow Tutorial

(https://www.facebook.com/guru99com/) (https://twitter.com/guru99com) (https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqFQ) (https://forms.aweber.com/form/46/724807646.htm)

About

About Us (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact Us (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)
Certificates (/certificate-it-professional.html)

Interesting

Books to Read! (/books.html)
Blog (/blog/)
Quiz (/tests.html)
eBook (/ebook-pdf.html)

Execute online

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)