Assignment#1: Basics in TensorBoard

prepared by Teeradaj Racharak (r.teeradaj@gmail.com)

TensorBoard is a visualization software that comes with any standard TensorFlow installation. In Google's word:

"The computations you'll use TensorFlow for many things can be complex and confusing. To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools call TensorBoard."

In this assignment, we'll practice how to use a special operation called **summary** to:

- visualize model parameters (*e.g.* $\theta_0, \theta_1, \ldots$)
- visualize metrics (*e.g.* loss values)
- visualize images (*e.g.* input images to a network)

Problem 1. (tf.summary.scalar)

tf.summary.scalar is used to write a single scalar-valued tensor. In general, we use it to write a scalar tensor that changes over time or iterations.

Let's run the following example to understand the point.

Assume we want to randomly pick 100 values from a standard Normal distribution, $\mathcal{N}(0,1)$, and plot them one after the other. One way to do so is simply create a variable and initialize it from a normal distribution (with mean = 0 and standard deviation = 1), then run a for loop in the session and initialize it 100 times. The complete code will be as follows and the required steps to write the summary is commented in the code.

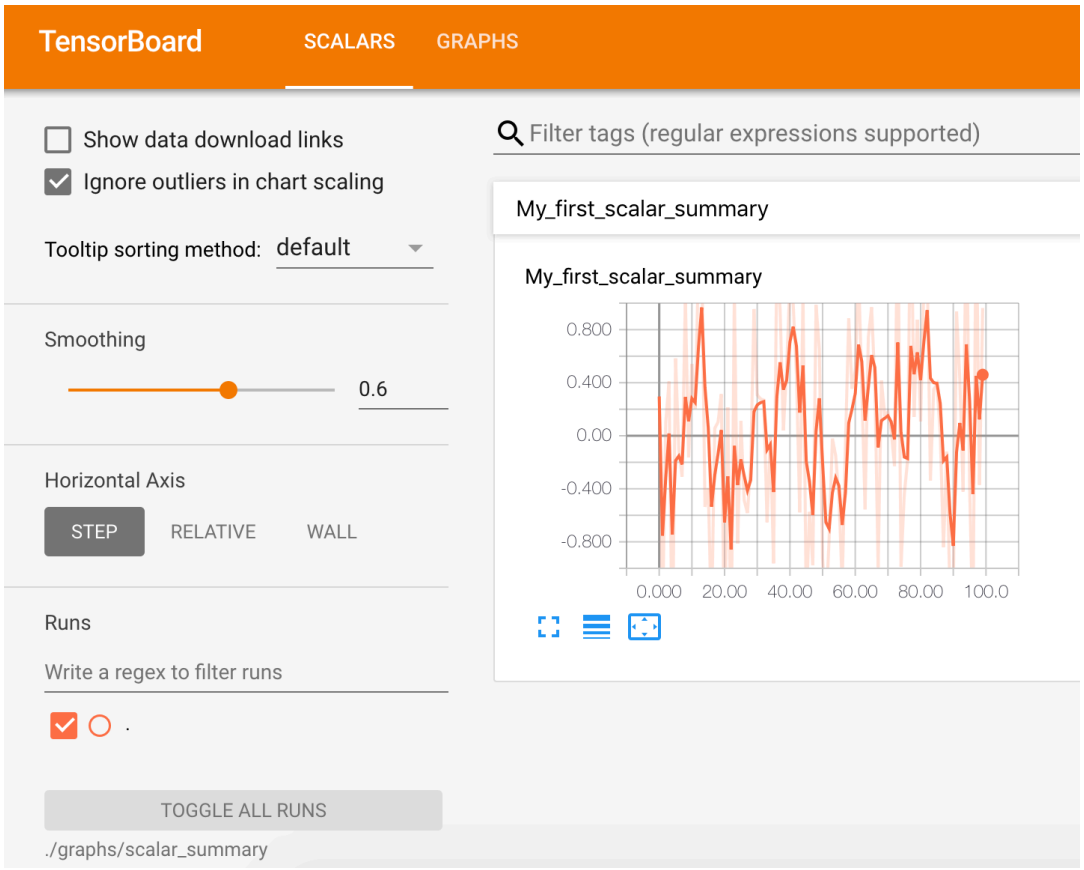
Before running the following code, let's create a Python file and name it as 'first-scalar-tensorboard.py'. After that, copy the code and execute it in console.

```
import tensorflow as tf

# To clear the defined variables and operations of the previous cell
tf.reset_default_graph()

# Create a scalar variable
x_scalar = tf.get_variable('x_scalar', shape=[],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))

# Step1: Create the scalar summary
first_summary = tf.summary.scalar(name='My_first_scalar_summary', tensor=x_scalar)

init = tf.global_variables_initializer()

# Launch the graph in a session
with tf.Session() as session:
    # Step2: Create the writer inside the session
    writer = tf.summary.FileWriter('./graphs/scalar_summary', session.graph)

    for step in range(100):
        # Loop over several initializations of the variable
        session.run(init)

        # Step3: Evaluate the scalar summary
        summary = session.run(first_summary)

        # Step4: Add the summary to the writer (i.e. to the event file)
        writer.add_summary(summary, step)


print('Done with writing the scalar summary')
writer.close()
```

Let's pull up TensorBoard and checkout the result. Like what we did before, you have to open the terminal and type:

```
$ tensorboard - -logdir = ./graphs/scalar_summary
```

In TensorBoard, we find a new tab named **scalar** next to the **graphs** tab. The whole window looks like the following.



In the figure, the plot panel is under the name **My_first_scalar_summary** *i.e.* the name that we defined in our code. The x-axis and y-axis shows the 100 steps and the corresponding values (random values from a standard normal distribution) of the variables.

Problem 2. (tf.summary.histogram)

tf.summary.histogram is used to plot the histogram of the values of a non-scalar tensor. It provides us a view of how the histogram (and the distribution) of the tensor values change over time or iterations. In the case of neural network, it is:

- commonly used to monitor the changes of weight and bias distributions;
- useful in detecting irregular behavior of the network parameters (*e.g.* when our weights explode or shrink abnormally).

Let's run the following example to understand the point. Before running the following code, let's create a Python file and name it as 'first-histogram-tensorboard.py'.

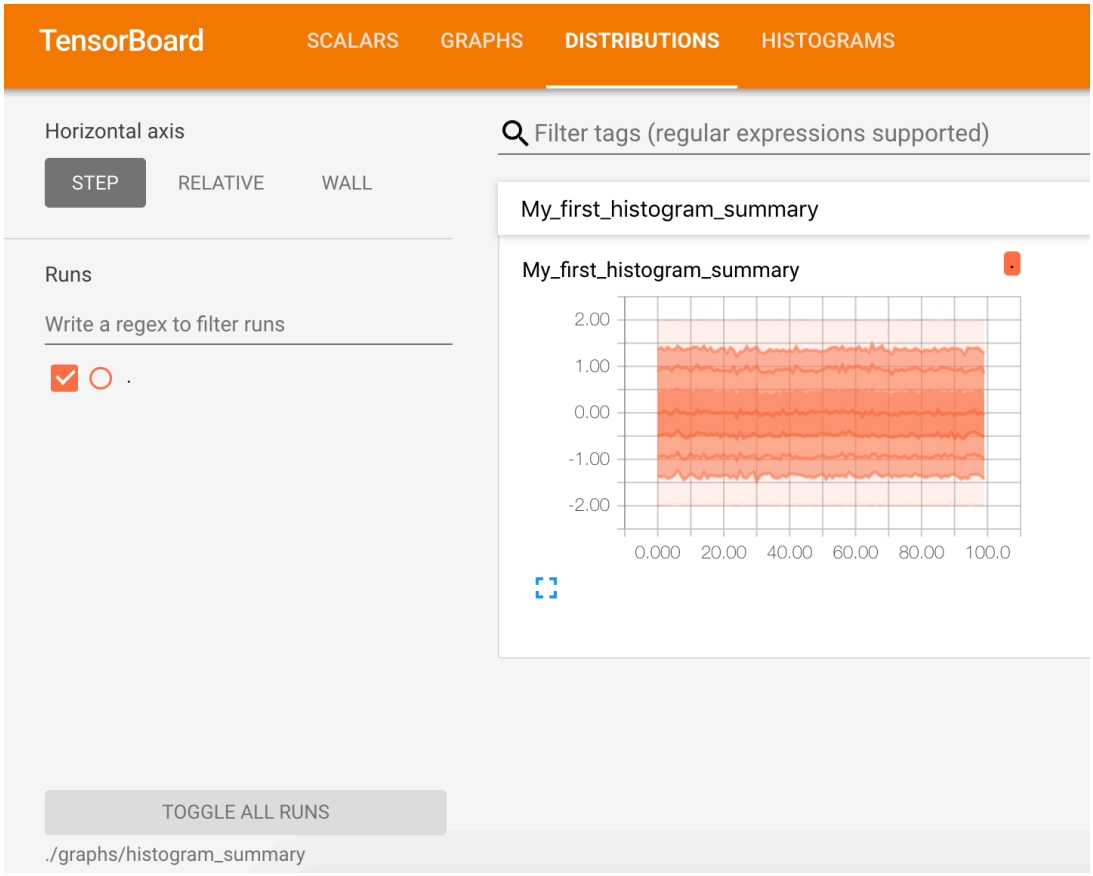
Assume we want to add a matrix of size 30x40, whose entries come from a standard normal distribution. Initialize this matrix 100 times and plot the distribution of its entries over time.

```
import tensorflow as tf

# To clear the defined variables and operations of the previous cell
tf.reset_default_graph()

# Create variables
x_scalar = tf.get_variable('x_scalar', shape=[],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))
x_matrix = tf.get_variable('x_matrix', shape=[30, 40],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))

# Step1.1: Create the scalar summary
scalar_summary = tf.summary.scalar(name='My_first_scalar_summary', tensor=x_scalar)

# Step1.2: Create the histogram summary for the non-scalar (i.e. 2D or matrix) tensor
histogram_summary = tf.summary.histogram('My_first_histogram_summary',
values=x_matrix)

init = tf.global_variables_initializer()

# Launch the graph in a session
with tf.Session() as session:
    # Step2: Create the writer inside the session
    writer = tf.summary.FileWriter('./graphs/histogram_summary', session.graph)

    for step in range(100):
        # Loop over several initializations of the variable
        session.run(init)

        # Step3: Evaluate the scalar summary
        summary1, summary2 = session.run([scalar_summary, histogram_summary])

        # Step4: Add the summary to the writer (i.e. to the event file)
        writer.add_summary(summary1, step)
        writer.add_summary(summary2, step)


print('Done with writing the histogram summary')
writer.close()
```

Let's pull up TensorBoard and checkout the result. Like what we did before, you have to open the terminal and type:

```
$ tensorboard - -logdir = ./graphs/histogram_summary
```

In TensorBoard, two new tabs are added to top menu *i.e.* **Distribution** and **Histogram**. The results will be as follows:



In the figure, **Distributions** tab contains a plot that shows the distribution of the values of the tensor (y-axis) through steps (x-axis).

What are the light and dark colors?

The answer is that each line on the chart represents a percentile in the distribution over the data. For example, the bottom line (the very light one) 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, $\mu + 1.5\sigma$, $\mu + \sigma$, $\mu + 0.5\sigma$, μ, $\mu - 0.5\sigma$, $\mu - \sigma$, $\mu - 1.5\sigma$, minimum] so that the colored regions, reading from inside to outside, have widths [σ,2σ,3σ], respectively.

What's about **Histograms** tab? It shows temporal 'slices' of data, where each slice is a histogram of the tensor at a given step. It's organized with the oldest timestep in the back and the most recent timestep in the front.

We can see the values on the histograms for any step by:

1. moving your cursor on the plot,
2. seeing the x-y values on the histograms

We can also change the Histogram mode from ‘offset’ to ‘overlay’ to see histograms overlaid with one another.

Problem 3. (Merge all summaries at once)

In practice, we can use any number of summaries to track different parameters in our model. However, this can make the running and the writing summaries inefficiently. To avoid those problems, we have to merge all summaries and rum them at once inside our session.

Again, let’s copy the following code and run it !

Before running the following code, let’s create a Python file and name it as ‘first-merged-all-tensorboard.py’.

```
import tensorflow as tf

# To clear the defined variables and operations of the previous cell
tf.reset_default_graph()

# Create variables
x_scalar = tf.get_variable('x_scalar', shape=[],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))
x_matrix = tf.get_variable('x_matrix', shape=[30, 40],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))

# Step1.1: Create the scalar summary
scalar_summary = tf.summary.scalar(name='My_first_scalar_summary', tensor=x_scalar)

# Step1.2: Create the histogram summary for the non-scalar (i.e. 2D or matrix) tensor
histogram_summary = tf.summary.histogram('My_first_histogram_summary',
values=x_matrix)

# Step2: Merge all summaries
merged = tf.summary.merge_all()
init = tf.global_variables_initializer()

# Launch the graph in a session
with tf.Session() as session:
    # Step3: Create the writer inside the session
    writer = tf.summary.FileWriter('./graphs/merge_all', session.graph)

    for step in range(100):
        # Loop over several initializations of the variable
        session.run(init)

        # Step4: Evaluate the scalar summary
        summary = session.run(merged)

        # Step5: Add the summary to the writer (i.e. to the event file)
        writer.add_summary(summary, step)


writer.close()
```

Problem 4. (tf.summary.image)

This type of summary is used for writing and visualizing tensors as images. In the case of neural network, this is used for:

- tracking the images fed into a network (*e.g.* say in each batch),
- tracking the images generated in the output (*e.g.* reconstructing images in an auto-encoder).

In general, it can be used to plot any tensor.

In this problem, we will practice to visualize a weight matrix of size 30x40 as an image of 30x40 pixels. An image can be created using:

```
tf.summary.image(name, tensor, max_outputs=3)
```

where **name** is the name of this image summary operation, **tensor** must be 4-D tensor of shape **[batch_size, height, width, channels]** (where **batch_size** is the number of images in the batch, the **hight** and **width** determine the size of the images; finally, the **channels** are 1 for gray-scaled images, 3 for RGB images, and 4 for RGBA images).

Again, let's copy the following code and run it !

Before running the following code, let's create a Python file and name it as 'first-image-tensorboard.py'. With this code, we define a variable of size 30x10 as 3 gray-scaled images of size 10x10 and another variable of size 50x30 as 5 color images of size 10x10.

```
import tensorflow as tf

# To clear the defined variables and operations of the previous cell
tf.reset_default_graph()

# Create the variables
w_grayscale = tf.get_variable('W_Grayscale', shape=[30, 10],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))
w_color = tf.get_variable('W_Color', shape=[50, 30],
initializer=tf.truncated_normal_initializer(mean=0, stddev=1))

# Step0: Reshape it to 4D-tensors
w_grayscale_reshaped = tf.reshape(w_grayscale, [3, 10, 10, 1])
w_color_reshaped = tf.reshape(w_color, [5, 10, 10, 3])

# Step1: Create the summaries
grayscale_summary = tf.summary.image('Grayscale', w_grayscale_reshaped)
color_summary = tf.summary.image('Color', w_color_reshaped, max_outputs=5)

# Step2: Merge all summaries
merged = tf.summary.merge_all()

# Step3: Create the operation for initializing all variables
init = tf.global_variables_initializer()

# Step4: Launch the graph in a session
with tf.Session() as session:
    # Step5: Create the writer inside a session
    writer = tf.summary.FileWriter('./graphs/image_summary', session.graph)

    # Step6: Initialize all variables
    session.run(init)

    # Step7: Evaluate the merged operation to get the summaries
    summary = session.run(merged)

    # Step6: Add summary to the writer (i.e. to the event file) to write on the disc
    writer.add_summary(summary)

writer.close()
```

This handout is revised from https://itnext.io/how-to-use-tensorboard-5d82f8654496. For further reading, see: https://jhui.github.io/2017/03/12/TensorBoard-visualize-your-learning/