tensorboard-pytorch Documentation

Release

tensorboard-pytorch Contributors

Jan 18, 2018

Contents:

CHAPTER 1

tensorboard-pytorch

A module for visualization with tensorboard

class tensorboardX.**SummaryWriter**(*log_dir=None*, *comment=''*)

Writes *Summary* directly to event files. The *SummaryWriter* class provides a high-level api to create an event file in a given directory and add summaries and events to it. The class updates the file contents asynchronously. This allows a training program to call methods to add data to the file directly from the training loop, without slowing down training.

__init__(*log_dir=None*, *comment=''*)

Parameters

- **log_dir** (*string*) – save location, default is: runs/**CURRENT_DATETIME_HOSTNAME**, which changes after each run. Use hierarchical folder structure to compare between runs easily. e.g. 'runs/exp1', 'runs/exp2'
- **comment** (*string*) – comment that appends to the default log_dir

add_audio(*tag*, *snd_tensor*, *global_step=None*, *sample_rate=44100*)

Add audio data to summary.

Parameters

- **tag** (*string*) – Data identifier
- **snd_tensor** (*torch.Tensor*) – Sound data
- **global_step** (*int*) – Global step value to record
- **sample_rate** (*int*) – sample rate in Hz

Shape: snd_tensor: $(1, L)$. The values should between [-1, 1].

add_embedding(*mat*, *metadata=None*, *label_img=None*, *global_step=None*, *tag='default'*)

Add embedding projector data to summary.

Parameters

- **mat** (*torch.Tensor*) – A matrix which each row is the feature vector of the data point

- **metadata** (*list*) – A list of labels, each element will be convert to string
- **label_img** (*torch.Tensor*) – Images correspond to each data point
- **global_step** (*int*) – Global step value to record
- **tag** (*string*) – Name for the embedding

Shape: mat: (N, D), where N is number of data and D is feature dimension

label_img: (N, C, H, W)

Examples:

```
import keyword
import torch
meta = []
while len(meta)<100:
    meta = meta+keyword.kwlist # get some strings
meta = meta[:100]

for i, v in enumerate(meta):
    meta[i] = v+str(i)

label_img = torch.rand(100, 3, 10, 32)
for i in range(100):
    label_img[i]*=i/100.0

writer.add_embedding(torch.randn(100, 5), metadata=meta, label_img=label_img)
writer.add_embedding(torch.randn(100, 5), label_img=label_img)
writer.add_embedding(torch.randn(100, 5), metadata=meta)
```

add_graph(*model*, *input_to_model*, *verbose=False*)

Add graph data to summary.

Parameters

- **model** (*torch.nn.Module*) – model to draw.
- **input_to_model** (*torch.autograd.Variable*) – a variable or a tuple of variables to be fed.

add_histogram(*tag*, *values*, *global_step=None*, *bins='tensorflow'*)

Add histogram to summary.

Parameters

- **tag** (*string*) – Data identifier
- **values** (*numpy.array*) – Values to build histogram
- **global_step** (*int*) – Global step value to record
- **bins** (*string*) – one of {'tensorflow','auto', 'fd', ...}, this determines how the bins are made. You can find other options in: https://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html

add_image(*tag*, *img_tensor*, *global_step=None*)

Add image data to summary.

Note that this requires the `pillow` package.

Parameters

- **tag** (*string*) – Data identifier
- **img_tensor** (*torch.Tensor*) – Image data
- **global_step** (*int*) – Global step value to record

Shape: img_tensor: $(3, H, W)$. Use `torchvision.utils.make_grid()` to prepare it is a good idea.

add_pr_curve(*tag*, *labels*, *predictions*, *global_step=None*, *num_thresholds=127*, *weights=None*)
Adds precision recall curve.

Parameters

- **tag** (*string*) – Data identifier
- **labels** (*torch.Tensor*) – Ground thuth data. Binary label for each element.
- **predictions** (*torch.Tensor*) – The probability that an element be classified as true. Value should in [0, 1]
- **global_step** (*int*) – Global step value to record
- **num_thresholds** (*int*) – Number of thresholds used to draw the curve.

add_scalar(*tag*, *scalar_value*, *global_step=None*)
Add scalar data to summary.

Parameters

- **tag** (*string*) – Data identifier
- **scalar_value** (*float*) – Value to save
- **global_step** (*int*) – Global step value to record

add_scalars(*main_tag*, *tag_scalar_dict*, *global_step=None*)
Adds many scalar data to summary.

Parameters

- **tag** (*string*) – Data identifier
- **main_tag** (*string*) – The parent name for the tags
- **tag_scalar_dict** (*dict*) – Key-value pair storing the tag and corresponding values
- **global_step** (*int*) – Global step value to record

Examples:

```
writer.add_scalars('run_14h',{'xsinx':i*np.sin(i/r),
                              'xcosx':i*np.cos(i/r),
                              'arctanx': numsteps*np.arctan(i/r)}, i)
# This function adds three values to the same scalar plot with the tag
# 'run_14h' in TensorBoard's scalar section.
```

add_text(*tag*, *text_string*, *global_step=None*)
Add text data to summary.

Parameters

- **tag** (*string*) – Data identifier
- **text_string** (*string*) – String to save

- **global_step** (*int*) – Global step value to record

Examples:

```
writer.add_text('lstm', 'This is an lstm', 0)
writer.add_text('rnn', 'This is an rnn', 10)
```

export_scalars_to_json(*path*)

Exports to the given path an ASCII file containing all the scalars written so far by this instance, with the following format: {writer_id : [[timestamp, step, value], ...], ...}

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

Symbols

A

E

S

T