Keras

Star 56,009

About Keras

Getting started

Developer guides

Keras API reference

Models API

Layers API

Callbacks API

Optimizers

Metrics

Losses

Data loading

Built-in small datasets

Keras Applications

Mixed precision

Utilities

KerasTuner

KerasCV

KerasNLP

Code examples

Why choose Keras?

Community & governance

Contributing to Keras

KerasTuner

KerasCV

KerasNLP

Search Keras documentation...

» Keras API reference / Callbacks API / TensorBoard

TensorBoard

`TensorBoard` class

[source]

```
tf.keras.callbacks.TensorBoard(
    log_dir="logs",
    histogram_freq=0,
    write_graph=True,
    write_images=False,
    write_steps_per_second=False,
    update_freq="epoch",
    profile_batch=0,
    embeddings_freq=0,
    embeddings_metadata=None,
    **kwargs
)
```

Enable visualizations for TensorBoard.

TensorBoard is a visualization tool provided with TensorFlow.

This callback logs events for TensorBoard, including:

- Metrics summary plots
- Training graph visualization
- Weight histograms
- Sampled profiling

When used in `Model.evaluate`, in addition to epoch summaries, there will be a summary that records evaluation metrics vs `Model.optimizer.iterations` written. The metric names will be prepended with `evaluation`, with `Model.optimizer.iterations` being the step in the visualized TensorBoard.

If you have installed TensorFlow with pip, you should be able to launch TensorBoard from the command line:

```
tensorboard --logdir=path_to_your_logs
```

You can find more information about TensorBoard here.

Arguments

- **log_dir**: the path of the directory where to save the log files to be parsed by TensorBoard. e.g. log_dir = os.path.join(working_dir, 'logs') This directory should not be reused by any other callbacks.
- **histogram_freq**: frequency (in epochs) at which to compute weight histograms for the layers of the model. If set to 0, histograms won't be computed. Validation data (or split) must be specified for histogram visualizations.
- **write_graph**: whether to visualize the graph in TensorBoard. The log file can become quite large when write_graph is set to True.
- **write_images**: whether to write model weights to visualize as image in TensorBoard.
- **write_steps_per_second**: whether to log the training steps per second into Tensorboard. This supports both epoch and batch frequency logging.
- **update_freq**: `'batch'` or `'epoch'` or integer. When using `'batch'`, writes the losses and metrics to TensorBoard after each batch. The same applies for `'epoch'`. If using an integer, let's say `1000`, the callback will write the metrics and losses to TensorBoard every 1000 batches. Note that writing too frequently to TensorBoard can slow down your training.
- **profile_batch**: Profile the batch(es) to sample compute characteristics. profile_batch must be a non-negative integer or a tuple of integers. A pair of positive integers signify a range of batches to profile. By default, profiling is disabled.
- **embeddings_freq**: frequency (in epochs) at which embedding layers will be visualized. If set to 0, embeddings won't be visualized.
- **embeddings_metadata**: Dictionary which maps embedding layer names to the filename of a file in which to save metadata for the embedding layer. In case the same metadata file is to be

used for all embedding layers, a single filename can be passed.

Examples

Basic usage:

```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./logs")
model.fit(x_train, y_train, epochs=2, callbacks=[tensorboard_callback])
# Then run the tensorboard command to view the visualizations.
```

Custom batch-level summaries in a subclassed Model:

```
class MyModel(tf.keras.Model):

  def build(self, _):
    self.dense = tf.keras.layers.Dense(10)

  def call(self, x):
    outputs = self.dense(x)
    tf.summary.histogram('outputs', outputs)
    return outputs

model = MyModel()
model.compile('sgd', 'mse')

# Make sure to set `update_freq=N` to log a batch-level summary every N batches.
# In addition to any [`tf.summary`](https://www.tensorflow.org/api_docs/python/tf/summary)
contained in `Model.call`, metrics added in
# `Model.compile` will be logged every N batches.
tb_callback = tf.keras.callbacks.TensorBoard('./logs', update_freq=1)
model.fit(x_train, y_train, callbacks=[tb_callback])
```

Custom batch-level summaries in a Functional API Model:

```
def my_summary(x):
  tf.summary.histogram('x', x)
  return x

inputs = tf.keras.Input(10)
x = tf.keras.layers.Dense(10)(inputs)
outputs = tf.keras.layers.Lambda(my_summary)(x)
model = tf.keras.Model(inputs, outputs)
model.compile('sgd', 'mse')

# Make sure to set `update_freq=N` to log a batch-level summary every N batches.
# In addition to any [`tf.summary`](https://www.tensorflow.org/api_docs/python/tf/summary)
contained in `Model.call`, metrics added in
# `Model.compile` will be logged every N batches.
tb_callback = tf.keras.callbacks.TensorBoard('./logs', update_freq=1)
model.fit(x_train, y_train, callbacks=[tb_callback])
```

Profiling:

```
# Profile a single batch, e.g. the 5th batch.
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir='./logs', profile_batch=5)
model.fit(x_train, y_train, epochs=2, callbacks=[tensorboard_callback])

# Profile a range of batches, e.g. from 10 to 20.
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir='./logs', profile_batch=(10,20))
model.fit(x_train, y_train, epochs=2, callbacks=[tensorboard_callback])
```

Terms | Privacy