tf.summary.FileWriter

Contents

Class FileWriter
Methods
__init__
__enter__

Class **FileWriter**

Defined in `tensorflow/python/summary/writer/writer.py`.

See the guide: Summary Operations > Generation of Summaries

Writes `Summary` protocol buffers to event files.

The `FileWriter` class provides a mechanism 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.

Methods

`__init__`

```
__init__(
    logdir,
    graph=None,
    max_queue=10,
    flush_secs=120,
    graph_def=None,
    filename_suffix=None
)
```

Creates a `FileWriter` and an event file.

On construction the summary writer creates a new event file in `logdir`. This event file will contain `Event` protocol buffers constructed when you call one of the following functions: `add_summary()`, `add_session_log()`, `add_event()`, or `add_graph()`.

If you pass a `Graph` to the constructor it is added to the event file. (This is equivalent to calling `add_graph()` later).

TensorBoard will pick the graph from the file and display it graphically so you can interactively explore the graph you built. You will usually pass the graph from the session in which you launched it:

```
...create a graph...
# Launch the graph in a session.
sess = tf.Session()
# Create a summary writer, add the 'graph' to the event file.
writer = tf.summary.FileWriter(<some-directory>, sess.graph)
```

The other arguments to the constructor control the asynchronous writes to the event file:

- `flush_secs`: How often, in seconds, to flush the added summaries and events to disk.
- `max_queue`: Maximum number of summaries or events pending to be written to disk before one of the 'add' calls block.

Args:

- `logdir`: A string. Directory where event file will be written.
- `graph`: A `Graph` object, such as `sess.graph`.
- `max_queue`: Integer. Size of the queue for pending events and summaries.
- `flush_secs`: Number. How often, in seconds, to flush the pending events and summaries to disk.
- `graph_def`: DEPRECATED: Use the `graph` argument instead.
- `filename_suffix`: A string. Every event file's name is suffixed with `suffix`.

__enter__

```
__enter__()
```

Make usable with "with" statement.

__exit__

```
__exit__(
    unused_type,
    unused_value,
    unused_traceback
)
```

Make usable with "with" statement.

add_event

```
add_event(event)
```

Adds an event to the event file.

Args:

- `event`: An `Event` protocol buffer.

add_graph

```
add_graph(
    graph,
    global_step=None,
    graph_def=None
)
```

Adds a `Graph` to the event file.

The graph described by the protocol buffer will be displayed by TensorBoard. Most users pass a graph in the constructor instead.

Args:

- `graph`: A `Graph` object, such as `sess.graph`.
- `global_step`: Number. Optional global step counter to record with the graph.
- `graph_def`: DEPRECATED. Use the `graph` parameter instead.

Raises:

- `ValueError`: If both graph and graph_def are passed to the method.

add_meta_graph

```
add_meta_graph(
    meta_graph_def,
    global_step=None
)
```

Adds a `MetaGraphDef` to the event file.

The `MetaGraphDef` allows running the given graph via `saver.import_meta_graph()`.

Args:

- `meta_graph_def`: A `MetaGraphDef` object, often as returned by `saver.export_meta_graph()`.
- `global_step`: Number. Optional global step counter to record with the graph.

Raises:

- `TypeError`: If both `meta_graph_def` is not an instance of `MetaGraphDef`.

add_run_metadata

```
add_run_metadata(
    run_metadata,
    tag,
    global_step=None
)
```

Adds a metadata information for a single session.run() call.

Args:

- `run_metadata` : A `RunMetadata` protobuf object.
- `tag` : The tag name for this metadata.
- `global_step` : Number. Optional global step counter to record with the StepStats.

Raises:

- `ValueError` : If the provided tag was already used for this type of event.

`add_session_log`

```
add_session_log(
    session_log,
    global_step=None
)
```

Adds a `SessionLog` protocol buffer to the event file.

This method wraps the provided session in an `Event` protocol buffer and adds it to the event file.

Args:

- `session_log` : A `SessionLog` protocol buffer.
- `global_step` : Number. Optional global step value to record with the summary.

`add_summary`

```
add_summary(
    summary,
    global_step=None
)
```

Adds a `Summary` protocol buffer to the event file.

This method wraps the provided summary in an `Event` protocol buffer and adds it to the event file.

You can pass the result of evaluating any summary op, using `tf.Session.run` or `tf.Tensor.eval` , to this function. Alternatively, you can pass a `tf.Summary` protocol buffer that you populate with your own data. The latter is commonly done to report evaluation results in event files.

Args:

- `summary` : A `Summary` protocol buffer, optionally serialized as a string.
- `global_step` : Number. Optional global step value to record with the summary.

`close`

```
close()
```

Flushes the event file to disk and close the file.

Call this method when you do not need the summary writer anymore.

`flush`

```
flush()
```

Flushes the event file to disk.

Call this method to make sure that all pending events have been written to disk.

`get_logdir`

```
get_logdir()
```

Returns the directory where event file will be written.

`reopen`

```
reopen()
```

Reopens the EventFileWriter.

Can be called after `close()` to add more events in the same directory. The events will go into a new events file.

Does nothing if the EventFileWriter was not closed.

Last updated November 2, 2017.