tf.contrib.training.FeedingQueueRunner

Contents

Class **FeedingQueueRunner**

Inherits From: `QueueRunner`

Defined in `tensorflow/python/estimator/inputs/queues/feeding_queue_runner.py`.

A queue runner that allows the feeding of values such as numpy arrays.

Properties

`cancel_op`

`close_op`

`enqueue_ops`

`exceptions_raised`

Exceptions raised but not handled by the `QueueRunner` threads.

Exceptions raised in queue runner threads are handled in one of two ways depending on whether or not a `Coordinator` was passed to `create_threads()`:

- With a `Coordinator`, exceptions are reported to the coordinator and forgotten by the `QueueRunner`.
- Without a `Coordinator`, exceptions are captured by the `QueueRunner` and made available in this `exceptions_raised` property.

Returns:

A list of Python `Exception` objects. The list is empty if no exception was captured. (No exceptions are captured when using a Coordinator.)

`name`

The string name of the underlying Queue.

`queue`

`queue_closed_exception_types`

Methods

`__init__`

```
__init__(
    queue=None,
    enqueue_ops=None,
    close_op=None,
    cancel_op=None,
    feed_fns=None,
    queue_closed_exception_types=None
)
```

Initialize the queue runner.

For further documentation, see `queue_runner.py`. Note that `FeedingQueueRunner` does not support construction from protobuffer nor serialization to protobuffer.

Args:

- `queue`: A `Queue`.
- `enqueue_ops`: List of enqueue ops to run in threads later.
- `close_op`: Op to close the queue. Pending enqueue ops are preserved.
- `cancel_op`: Op to close the queue and cancel pending enqueue ops.
- `feed_fns`: a list of functions that return a dictionary mapping fed `Tensor`s to values. Must be the same length as `enqueue_ops`.
- `queue_closed_exception_types`: Optional tuple of Exception types that indicate that the queue has been closed when raised during an enqueue operation. Defaults to `(tf.errors.OutOfRangeError, tf.errors.CancelledError)`.

Raises:

- `ValueError`: `feed_fns` is not `None` and has different length than `enqueue_ops`.

`create_threads`

```
create_threads(
    sess,
    coord=None,
    daemon=False,
    start=False
)
```

Create threads to run the enqueue ops for the given session.

This method requires a session in which the graph was launched. It creates a list of threads, optionally starting them. There is one thread for each op passed in `enqueue_ops`.

The `coord` argument is an optional coordinator, that the threads will use to terminate together and report exceptions. If a coordinator is given, this method starts an additional thread to close the queue when the coordinator requests a stop.

If previously created threads for the given session are still running, no new threads will be created.

Args:

- `sess`: A `Session`.
- `coord`: Optional `Coordinator` object for reporting errors and checking stop conditions.
- `daemon`: Boolean. If `True` make the threads daemon threads.
- `start`: Boolean. If `True` starts the threads. If `False` the caller must call the `start()` method of the returned threads.

Returns:

A list of threads.

`from_proto`

```
from_proto(
    queue_runner_def,
    import_scope=None
)
```

Returns a `QueueRunner` object created from `queue_runner_def`.

`to_proto`

```
to_proto()
```

Last updated November 2, 2017.