tf.train.QueueRunner

Contents

Class **`QueueRunner`**

Aliases:

- Class `tf.train.QueueRunner`
- Class `tf.train.queue_runner.QueueRunner`

Defined in `tensorflow/python/training/queue_runner_impl.py`.

See the guides: Exporting and Importing a MetaGraph > What's in a MetaGraph, Reading data > Reading from files, Threading and Queues > Manual Thread Management, Training > Coordinator and QueueRunner

Holds a list of enqueue operations for a queue, each to be run in a thread.

Queues are a convenient TensorFlow mechanism to compute tensors asynchronously using multiple threads. For example in the canonical 'Input Reader' setup one set of threads generates filenames in a queue; a second set of threads read records from the files, processes them, and enqueues tensors on a second queue; a third set of threads dequeues these input records to construct batches and runs them through training operations.

There are several delicate issues when running multiple threads that way: closing the queues in sequence as the input is exhausted, correctly catching and reporting exceptions, etc.

The `QueueRunner`, combined with the `Coordinator`, helps handle these issues.

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,
    queue_closed_exception_types=None,
    queue_runner_def=None,
    import_scope=None
)
```

Create a QueueRunner.

On construction the `QueueRunner` adds an op to close the queue. That op will be run if the enqueue ops raise exceptions.

When you later call the `create_threads()` method, the `QueueRunner` will create one thread for each op in `enqueue_ops`. Each thread will run its enqueue op in parallel with the other threads. The enqueue ops do not have to all be the same op, but it is expected that they all enqueue tensors in `queue`.

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.
- `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,)`. Another common case includes `(tf.errors.OutOfRangeError, tf.errors.CancelledError)`, when some of the enqueue ops may dequeue from other Queues.
- `queue_runner_def`: Optional `QueueRunnerDef` protocol buffer. If specified, recreates the QueueRunner from its contents. `queue_runner_def` and the other arguments are mutually exclusive.
- `import_scope`: Optional `string`. Name scope to add. Only used when initializing from protocol buffer.

Raises:

* `ValueError`: If both `queue_runner_def` and `queue` are both specified.
* `ValueError`: If `queue` or `enqueue_ops` are not provided when not restoring from `queue_runner_def`.

`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`

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

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

`to_proto`

```
to_proto(export_scope=None)
```

Converts this `QueueRunner` to a `QueueRunnerDef` protocol buffer.

Args:

- `export_scope`: Optional `string`. Name scope to remove.

Returns:

A `QueueRunnerDef` protocol buffer, or `None` if the `Variable` is not in the specified name scope.
