tf.QueueBase

Class **QueueBase**

Defined in `tensorflow/python/ops/data_flow_ops.py`.

See the guide: Inputs and Readers > Queues

Base class for queue implementations.

A queue is a TensorFlow data structure that stores tensors across multiple steps, and exposes operations that enqueue and dequeue tensors.

Each queue element is a tuple of one or more tensors, where each tuple component has a static dtype, and may have a static shape. The queue implementations support versions of enqueue and dequeue that handle single elements, versions that support enqueuing and dequeuing a batch of elements at once.

See `tf.FIFOQueue` and `tf.RandomShuffleQueue` for concrete implementations of this class, and instructions on how to create them.

Properties

`dtypes`

The list of dtypes for each component of a queue element.

`name`

The name of the underlying queue.

`names`

The list of names for each component of a queue element.

`queue_ref`

The underlying queue reference.

`shapes`

The list of shapes for each component of a queue element.

Methods

`__init__`

```
__init__(
    dtypes,
    shapes,
    names,
    queue_ref
)
```

Constructs a queue object from a queue reference.

The two optional lists, `shapes` and `names`, must be of the same length as `dtypes` if provided. The values at a given index `i` indicate the shape and name to use for the corresponding queue component in `dtypes`.

Args:

- `dtypes`: A list of types. The length of dtypes must equal the number of tensors in each element.
- `shapes`: Constraints on the shapes of tensors in an element: A list of shape tuples or None. This list is the same length as dtypes. If the shape of any tensors in the element are constrained, all must be; shapes can be None if the shapes should not be constrained.
- `names`: Optional list of names. If provided, the `enqueue()` and `dequeue()` methods will use dictionaries with these names as keys. Must be None or a list or tuple of the same length as `dtypes`.
- `queue_ref`: The queue reference, i.e. the output of the queue op.

Raises:

- `ValueError`: If one of the arguments is invalid.

`close`

```
close(
    cancel_pending_enqueues=False,
    name=None
)
```

Closes this queue.

This operation signals that no more elements will be enqueued in the given queue. Subsequent `enqueue` and `enqueue_many` operations will fail. Subsequent `dequeue` and `dequeue_many` operations will continue to succeed if sufficient elements remain in the queue. Subsequently dequeue and dequeue_many operations that would otherwise block waiting for more elements (if close hadn't been called) will now fail immediately.

If `cancel_pending_enqueues` is `True`, all pending requests will also be canceled.

Args:

- `cancel_pending_enqueues`: (Optional.) A boolean, defaulting to `False` (described above).
- `name`: A name for the operation (optional).

Returns:

The operation that closes the queue.

dequeue

```
dequeue(name=None)
```

Dequeues one element from this queue.

If the queue is empty when this operation executes, it will block until there is an element to dequeue.

At runtime, this operation may raise an error if the queue is `tf.QueueBase.close` before or during its execution. If the queue is closed, the queue is empty, and there are no pending enqueue operations that can fulfill this request, `tf.errors.OutOfRangeError` will be raised. If the session is `tf.Session.close`, `tf.errors.CancelledError` will be raised.

Args:

- `name`: A name for the operation (optional).

Returns:

The tuple of tensors that was dequeued.

dequeue_many

```
dequeue_many(
    n,
    name=None
)
```

Dequeues and concatenates `n` elements from this queue.

This operation concatenates queue-element component tensors along the 0th dimension to make a single component tensor. All of the components in the dequeued tuple will have size `n` in the 0th dimension.

If the queue is closed and there are less than `n` elements left, then an `OutOfRange` exception is raised.

At runtime, this operation may raise an error if the queue is `tf.QueueBase.close` before or during its execution. If the queue is closed, the queue contains fewer than `n` elements, and there are no pending enqueue operations that can fulfill this request, `tf.errors.OutOfRangeError` will be raised. If the session is `tf.Session.close`, `tf.errors.CancelledError` will be raised.

Args:

- `n`: A scalar `Tensor` containing the number of elements to dequeue.
- `name`: A name for the operation (optional).

Returns:

The tuple of concatenated tensors that was dequeued.

`dequeue_up_to`

```
dequeue_up_to(
    n,
    name=None
)
```

Dequeues and concatenates `n` elements from this queue.

Note This operation is not supported by all queues. If a queue does not support DequeueUpTo, then a `tf.errors.UnimplementedError` is raised.

This operation concatenates queue-element component tensors along the 0th dimension to make a single component tensor. If the queue has not been closed, all of the components in the dequeued tuple will have size `n` in the 0th dimension.

If the queue is closed and there are more than `0` but fewer than `n` elements remaining, then instead of raising a `tf.errors.OutOfRangeError` like `tf.QueueBase.dequeue_many`, less than `n` elements are returned immediately. If the queue is closed and there are `0` elements left in the queue, then a `tf.errors.OutOfRangeError` is raised just like in `dequeue_many`. Otherwise the behavior is identical to `dequeue_many`.

Args:

- `n`: A scalar `Tensor` containing the number of elements to dequeue.
- `name`: A name for the operation (optional).

Returns:

The tuple of concatenated tensors that was dequeued.

`enqueue`

```
enqueue(
    vals,
    name=None
)
```

Enqueues one element to this queue.

If the queue is full when this operation executes, it will block until the element has been enqueued.

At runtime, this operation may raise an error if the queue is `tf.QueueBase.close` before or during its execution. If the queue is closed before this operation runs, `tf.errors.CancelledError` will be raised. If this operation is blocked, and either (i) the queue is closed by a close operation with `cancel_pending_enqueues=True`, or (ii) the session is `tf.Session.close`, `tf.errors.CancelledError` will be raised.

Args:

- `vals`: A tensor, a list or tuple of tensors, or a dictionary containing the values to enqueue.
- `name`: A name for the operation (optional).

Returns:

The operation that enqueues a new tuple of tensors to the queue.

`enqueue_many`

```
enqueue_many(
    vals,
    name=None
)
```

Enqueues zero or more elements to this queue.

This operation slices each component tensor along the 0th dimension to make multiple queue elements. All of the tensors in `vals` must have the same size in the 0th dimension.

If the queue is full when this operation executes, it will block until all of the elements have been enqueued.

At runtime, this operation may raise an error if the queue is `tf.QueueBase.close` before or during its execution. If the queue is closed before this operation runs, `tf.errors.CancelledError` will be raised. If this operation is blocked, and either (i) the queue is closed by a close operation with `cancel_pending_enqueues=True`, or (ii) the session is `tf.Session.close`, `tf.errors.CancelledError` will be raised.

Args:

- `vals`: A tensor, a list or tuple of tensors, or a dictionary from which the queue elements are taken.
- `name`: A name for the operation (optional).

Returns:

The operation that enqueues a batch of tuples of tensors to the queue.

`from_list`

```
@staticmethod
from_list(
    index,
    queues
)
```

Create a queue using the queue reference from `queues[index]`.

Args:

- `index`: An integer scalar tensor that determines the input that gets selected.
- `queues`: A list of `QueueBase` objects.

Returns:

A `QueueBase` object.

Raises:

- `TypeError`: When `queues` is not a list of `QueueBase` objects, or when the data types of `queues` are not all the same.

`is_closed`

```
is_closed(name=None)
```

Returns true if queue is closed.

This operation returns true if the queue is closed and false if the queue is open.

Args:

- `name`: A name for the operation (optional).

Returns:

True if the queue is closed and false if the queue is open.

`size`

```
size(name=None)
```

Compute the number of elements in this queue.

Args:

- `name`: A name for the operation (optional).

Returns:

A scalar tensor containing the number of elements in this queue.