tf.train.batch

```
batch(
    tensors,
    batch_size,
    num_threads=1,
    capacity=32,
    enqueue_many=False,
    shapes=None,
    dynamic_pad=False,
    allow_smaller_final_batch=False,
    shared_name=None,
    name=None
)
```

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

See the guides: Inputs and Readers > Input pipeline, Reading data > Preloaded data

Creates batches of tensors in `tensors`.

The argument `tensors` can be a list or a dictionary of tensors. The value returned by the function will be of the same type as `tensors`.

This function is implemented using a queue. A `QueueRunner` for the queue is added to the current `Graph`'s `QUEUE_RUNNER` collection.

If `enqueue_many` is `False`, `tensors` is assumed to represent a single example. An input tensor with shape `[x, y, z]` will be output as a tensor with shape `[batch_size, x, y, z]`.

If `enqueue_many` is `True`, `tensors` is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of `tensors` should have the same size in the first dimension. If an input tensor has shape `[*, x, y, z]`, the output will have shape `[batch_size, x, y, z]`. The `capacity` argument controls the how long the prefetching is allowed to grow the queues.

The returned operation is a dequeue operation and will throw `tf.errors.OutOfRangeError` if the input queue is exhausted. If this operation is feeding another input queue, its queue runner will catch this exception, however, if this operation is used in your main thread you are responsible for catching this yourself.

N.B.: If `dynamic_pad` is `False`, you must ensure that either (i) the `shapes` argument is passed, or (ii) all of the tensors in `tensors` must have fully-defined shapes. `ValueError` will be raised if neither of these conditions holds.

If `dynamic_pad` is `True`, it is sufficient that the *rank* of the tensors is known, but individual dimensions may have shape `None`. In this case, for each enqueue the dimensions with value `None` may have a variable length; upon dequeue, the output tensors will be padded on the right to the maximum shape of the tensors in the current minibatch. For numbers, this padding takes value 0. For strings, this padding is the empty string. See `PaddingFIFOQueue` for more info.

If `allow_smaller_final_batch` is `True`, a smaller batch value than `batch_size` is returned when the queue is closed and there are not enough elements to fill the batch, otherwise the pending elements are discarded. In addition, all output tensors' static shapes, as accessed via the `shape` property will have a first `Dimension` value of `None`, and operations that depend on fixed batch_size would fail.

Args:

- `tensors`: The list or dictionary of tensors to enqueue.
- `batch_size`: The new batch size pulled from the queue.
- `num_threads`: The number of threads enqueuing `tensors`. The batching will be nondeterministic if `num_threads > 1`.
- `capacity`: An integer. The maximum number of elements in the queue.
- `enqueue_many`: Whether each tensor in `tensors` is a single example.
- `shapes`: (Optional) The shapes for each example. Defaults to the inferred shapes for `tensors`.
- `dynamic_pad`: Boolean. Allow variable dimensions in input shapes. The given dimensions are padded upon dequeue so that tensors within a batch have the same shapes.
- `allow_smaller_final_batch`: (Optional) Boolean. If `True`, allow the final batch to be smaller if there are insufficient items left in the queue.
- `shared_name`: (Optional). If set, this queue will be shared under the given name across multiple sessions.
- `name`: (Optional) A name for the operations.

Returns:

A list or dictionary of tensors with the same types as `tensors` (except if the input is a list of one element, then it returns a tensor, not a list).

Raises:

- `ValueError`: If the `shapes` are not specified, and cannot be inferred from the elements of `tensors`.