tf.contrib.training.bucket

```
bucket(
    tensors,
    which_bucket,
    batch_size,
    num_buckets,
    num_threads=1,
    capacity=32,
    bucket_capacities=None,
    shapes=None,
    dynamic_pad=False,
    allow_smaller_final_batch=False,
    keep_input=True,
    shared_name=None,
    name=None
)
```

Defined in `tensorflow/contrib/training/python/training/bucket_ops.py`.

See the guide: Training (contrib) > Bucketing

Lazy bucketing of input tensors according to `which_bucket`.

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`.

The tensors entering this function are put into the bucket given by `which_bucket`. Each bucket has its own queue. When a bucket contains `batch_size` elements, this minibatch is pushed onto a top queue. The tensors returned from this function are a the result of dequeueing the next minibatch from this top queue.

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

As the returned tensors are the result of a dequeue operation, evaluating them will throw a `tf.errors.OutOfRangeError` when the input queue is exhausted. If these tensors are feeding another input queue, its queue runner will catch this exception, however, if they are 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 queues are 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 `get_shape()` method will have a 0th `Dimension` value of `None`, and operations that depend on fixed batch_size would fail.

Args:

- `tensors`: The list or dictionary of tensors, representing a single element, to bucket. Nested lists are not supported.
- `which_bucket`: An `int32` scalar Tensor taking a value in `[0, num_buckets)`.
- `batch_size`: The new batch size pulled from the queue (all queues will have the same size). If a list is passed in then each bucket will have a different batch_size. (python int, int32 scalar or iterable of integers of length num_buckets).
- `num_buckets`: A python integer, the number of buckets.
- `num_threads`: An integer. The number of threads enqueuing `tensors`.
- `capacity`: An integer. The maximum number of minibatches in the top queue, and also (by default) the maximum number of elements within each bucket.
- `bucket_capacities`: (Optional) None or a list of integers, the capacities of each bucket. If None, capacity is used (default). If specified, it must be a list of integers of length num_buckets: the i-th element is used as capacity for the i-th bucket queue.
- `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 batches to be smaller if there are insufficient items left in the queues.
- `keep_input`: A `bool` scalar Tensor. If provided, this tensor controls whether the input is added to the queue or not. If it evaluates `True`, then `tensors` are added to the bucket; otherwise they are dropped. This tensor essentially acts as a filtering mechanism.
- `shared_name`: (Optional). If set, the queues will be shared under the given name across multiple sessions.
- `name`: (Optional) A name for the operations.

Returns:

A tuple `(bucket, outputs)` where `bucket` is a `int32` scalar tensor and `outputs` is a list or dictionary of batched outputs corresponding to elements of `tensors`. Every step will receive a new bucket of outputs.

Raises:

- `ValueError`: If the `shapes` are not specified, and cannot be inferred from the elements of `tensors` or if batch_size is a sequence but its length != num_buckets. Also if bucket_capacities is not None but its length != num_buckets.

English