tf.train.shuffle_batch_join

```
shuffle_batch_join(
    tensors_list,
    batch_size,
    capacity,
    min_after_dequeue,
    seed=None,
    enqueue_many=False,
    shapes=None,
    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 > Reading from files

Create batches by randomly shuffling tensors.

The `tensors_list` argument is a list of tuples of tensors, or a list of dictionaries of tensors. Each element in the list is treated similarly to the `tensors` argument of `tf.train.shuffle_batch()`.

This version enqueues a different list of tensors in different threads. It adds the following to the current `Graph`:

- A shuffling queue into which tensors from `tensors_list` are enqueued.
- A `dequeue_many` operation to create batches from the queue.
- A `QueueRunner` to `QUEUE_RUNNER` collection, to enqueue the tensors from `tensors_list`.

`len(tensors_list)` threads will be started, with thread `i` enqueuing the tensors from `tensors_list[i]`. `tensors_list[i1][j]` must match `tensors_list[i2][j]` in type and shape, except in the first dimension if `enqueue_many` is true.

If `enqueue_many` is `False`, each `tensors_list[i]` 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_list[i]` is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of `tensors_list[i]` 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.

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_list`: A list of tuples or dictionaries of tensors to enqueue.
- `batch_size`: An integer. The new batch size pulled from the queue.
- `capacity`: An integer. The maximum number of elements in the queue.
- `min_after_dequeue`: Minimum number elements in the queue after a dequeue, used to ensure a level of mixing of elements.
- `seed`: Seed for the random shuffling within the queue.
- `enqueue_many`: Whether each tensor in `tensor_list_list` is a single example.
- `shapes`: (Optional) The shapes for each example. Defaults to the inferred shapes for `tensors_list[i]`.
- `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 number and types as `tensors_list[i]`.

Raises:

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

Last updated November 2, 2017.