tf.contrib.training.stratified_sample

```
stratified_sample(
    tensors,
    labels,
    target_probs,
    batch_size,
    init_probs=None,
    enqueue_many=False,
    queue_capacity=16,
    threads_per_queue=1,
    name=None
)
```

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

See the guide: Training (contrib) > Online data resampling

Stochastically creates batches based on per-class probabilities.

This method discards examples. Internally, it creates one queue to amortize the cost of disk reads, and one queue to hold the properly-proportioned batch.

Args:

- `tensors`: List of tensors for data. All tensors are either one item or a batch, according to enqueue_many.
- `labels`: Tensor for label of data. Label is a single integer or a batch, depending on `enqueue_many`. It is not a one-hot vector.
- `target_probs`: Target class proportions in batch. An object whose type has a registered Tensor conversion function.
- `batch_size`: Size of batch to be returned.
- `init_probs`: Class proportions in the data. An object whose type has a registered Tensor conversion function, or `None` for estimating the initial distribution.
- `enqueue_many`: Bool. If true, interpret input tensors as having a batch dimension.
- `queue_capacity`: Capacity of the large queue that holds input examples.
- `threads_per_queue`: Number of threads for the large queue that holds input examples and for the final queue with the proper class proportions.
- `name`: Optional prefix for ops created by this function.

Raises:

- `ValueError`: If `tensors` isn't iterable.
- `ValueError`: `enqueue_many` is True and labels doesn't have a batch dimension, or if `enqueue_many` is False and labels isn't a scalar.
- `ValueError`: `enqueue_many` is True, and batch dimension on data and labels don't match.
- `ValueError`: if probs don't sum to one.
- `ValueError`: if a zero initial probability class has a nonzero target probability.

- `TFAssertion`: if labels aren't integers in [0, num classes).

Returns:

(data_batch, label_batch), where data_batch is a list of tensors of the same length as `tensors`

Example: # Get tensor for a single data and label example. data, label = data_provider.Get(['data', 'label'])

Get stratified batch according to per-class probabilities. target_probs = [...distribution you want...] [data_batch], labels = tf.contrib.training.stratified_sample([data], label, target_probs)

Run batch through network. ...
