tf.nn.ctc_beam_search_decoder

```
ctc_beam_search_decoder(
    inputs,
    sequence_length,
    beam_width=100,
    top_paths=1,
    merge_repeated=True
)
```

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

See the guide: Neural Network > Connectionist Temporal Classification (CTC)

Performs beam search decoding on the logits given in input.

Note The `ctc_greedy_decoder` is a special case of the `ctc_beam_search_decoder` with `top_paths=1` and `beam_width=1` (but that decoder is faster for this special case).

If `merge_repeated` is `True`, merge repeated classes in the output beams. This means that if consecutive entries in a beam are the same, only the first of these is emitted. That is, when the top path is `A B B B B`, the return value is:

- `A B` if `merge_repeated = True`.
- `A B B B B` if `merge_repeated = False`.

Args:

- `inputs`: 3-D `float` `Tensor`, size `[max_time x batch_size x num_classes]`. The logits.
- `sequence_length`: 1-D `int32` vector containing sequence lengths, having size `[batch_size]`.
- `beam_width`: An int scalar >= 0 (beam search beam width).
- `top_paths`: An int scalar >= 0, <= beam_width (controls output size).
- `merge_repeated`: Boolean. Default: True.

Returns:

A tuple `(decoded, log_probabilities)` where `decoded`: *A list of length top_paths, where* `decoded[j]` *is a* `SparseTensor` *containing the decoded outputs:*
`decoded[j].indices`: *Indices matrix* `(total_decoded_outputs[j] x 2)` *The rows store: [batch, time].*
`decoded[j].values`: *Values vector, size* `(total_decoded_outputs[j])`. *The vector stores the decoded classes for beam j.*
`decoded[j].shape`: *Shape vector, size* `(2)`. *The shape values are:* `[batch_size, max_decoded_length[j]]`.
`log_probability`: A `float` matrix `(batch_size x top_paths)` containing sequence log-probabilities.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**