tf.nn.ctc_loss

```
ctc_loss(
    labels,
    inputs,
    sequence_length,
    preprocess_collapse_repeated=False,
    ctc_merge_repeated=True,
    ignore_longer_outputs_than_inputs=False,
    time_major=True
)
```

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

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

Computes the CTC (Connectionist Temporal Classification) Loss.

This op implements the CTC loss as presented in the article:

A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber. Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks. ICML 2006, Pittsburgh, USA, pp. 369-376.

Input requirements:

```
sequence_length(b) <= time for all b

max(labels.indices(labels.indices[:, 1] == b, 2))
  <= sequence_length(b) for all b.
```

Notes:

This class performs the softmax operation for you, so inputs should be e.g. linear projections of outputs by an LSTM.

The `inputs` Tensor's innermost dimension size, `num_classes`, represents `num_labels + 1` classes, where num_labels is the number of true labels, and the largest value `(num_classes - 1)` is reserved for the blank label.

For example, for a vocabulary containing 3 labels `[a, b, c]`, `num_classes = 4` and the labels indexing is `{a: 0, b: 1, c: 2, blank: 3}`.

Regarding the arguments `preprocess_collapse_repeated` and `ctc_merge_repeated`:

If `preprocess_collapse_repeated` is True, then a preprocessing step runs before loss calculation, wherein repeated labels passed to the loss are merged into single labels. This is useful if the training labels come from, e.g., forced alignments and therefore have unnecessary repetitions.

If `ctc_merge_repeated` is set False, then deep within the CTC calculation, repeated non-blank labels will not be merged and are interpreted as individual labels. This is a simplified (non-standard) version of CTC.

Here is a table of the (roughly) expected first order behavior:

- `preprocess_collapse_repeated=False`, `ctc_merge_repeated=True`

Classical CTC behavior: Outputs true repeated classes with blanks in between, and can also output repeated classes with no blanks in between that need to be collapsed by the decoder.

- `preprocess_collapse_repeated=True` , `ctc_merge_repeated=False`

Never learns to output repeated classes, as they are collapsed in the input labels before training.

- `preprocess_collapse_repeated=False` , `ctc_merge_repeated=False`

Outputs repeated classes with blanks in between, but generally does not require the decoder to collapse/merge repeated classes.

- `preprocess_collapse_repeated=True` , `ctc_merge_repeated=True`

Untested. Very likely will not learn to output repeated classes.

The `ignore_longer_outputs_than_inputs` option allows to specify the behavior of the CTCLoss when dealing with sequences that have longer outputs than inputs. If true, the CTCLoss will simply return zero gradient for those items, otherwise an InvalidArgument error is returned, stopping training.

Args:

- `labels` : An `int32` `SparseTensor` . `labels.indices[i, :] == [b, t]` means `labels.values[i]` stores the id for (batch b, time t). `labels.values[i]` must take on values in `[0, num_labels)` . See `core/ops/ctc_ops.cc` for more details.
- `inputs` : 3-D `float` `Tensor` . If time_major == False, this will be a `Tensor` shaped: `[batch_size x max_time x num_classes]` . If time_major == True (default), this will be a `Tensor` shaped: `[max_time x batch_size x num_classes]` . The logits.
- `sequence_length` : 1-D `int32` vector, size `[batch_size]` . The sequence lengths.
- `preprocess_collapse_repeated` : Boolean. Default: False. If True, repeated labels are collapsed prior to the CTC calculation.
- `ctc_merge_repeated` : Boolean. Default: True.
- `ignore_longer_outputs_than_inputs` : Boolean. Default: False. If True, sequences with longer outputs than inputs will be ignored.
- `time_major` : The shape format of the `inputs` Tensors. If True, these `Tensors` must be shaped `[max_time, batch_size, num_classes]` . If False, these `Tensors` must be shaped `[batch_size, max_time, num_classes]` . Using `time_major = True` (default) is a bit more efficient because it avoids transposes at the beginning of the ctc_loss calculation. However, most TensorFlow data is batch-major, so by this function also accepts inputs in batch-major form.

Returns:

A 1-D `float` `Tensor` , size `[batch]` , containing the negative log probabilities.

Raises:

- `TypeError` : if labels is not a `SparseTensor` .

Last updated November 2, 2017.