tf.nn.bidirectional_dynamic_rnn

```
bidirectional_dynamic_rnn(
    cell_fw,
    cell_bw,
    inputs,
    sequence_length=None,
    initial_state_fw=None,
    initial_state_bw=None,
    dtype=None,
    parallel_iterations=None,
    swap_memory=False,
    time_major=False,
    scope=None
)
```

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

See the guide: Neural Network > Recurrent Neural Networks

Creates a dynamic version of bidirectional recurrent neural network.

Takes input and builds independent forward and backward RNNs. The input_size of forward and backward cell must match. The initial state for both directions is zero by default (but can be set optionally) and no intermediate states are ever returned – the network is fully unrolled for the given (passed in) length(s) of the sequence(s) or completely unrolled if length(s) is not given.

Args:

- `cell_fw`: An instance of RNNCell, to be used for forward direction.
- `cell_bw`: An instance of RNNCell, to be used for backward direction.
- `inputs`: The RNN inputs. If time_major == False (default), this must be a tensor of shape: `[batch_size, max_time, ...]`, or a nested tuple of such elements. If time_major == True, this must be a tensor of shape: `[max_time, batch_size, ...]`, or a nested tuple of such elements.
- `sequence_length`: (optional) An int32/int64 vector, size `[batch_size]`, containing the actual lengths for each of the sequences in the batch. If not provided, all batch entries are assumed to be full sequences; and time reversal is applied from time `0` to `max_time` for each sequence.
- `initial_state_fw`: (optional) An initial state for the forward RNN. This must be a tensor of appropriate type and shape `[batch_size, cell_fw.state_size]`. If `cell_fw.state_size` is a tuple, this should be a tuple of tensors having shapes `[batch_size, s] for s in cell_fw.state_size`.
- `initial_state_bw`: (optional) Same as for `initial_state_fw`, but using the corresponding properties of `cell_bw`.
- `dtype`: (optional) The data type for the initial states and expected output. Required if initial_states are not provided or RNN states have a heterogeneous dtype.
- `parallel_iterations`: (Default: 32). The number of iterations to run in parallel. Those operations which do not have any temporal dependency and can be run in parallel, will be. This parameter trades off time for space. Values >> 1 use more memory but take less time, while smaller values use less memory but computations take longer.
- `swap_memory`: Transparently swap the tensors produced in forward inference but needed for back prop from GPU to CPU. This allows training RNNs which would typically not fit on a single GPU, with very minimal (or no) performance

penalty.

- `time_major`: The shape format of the `inputs` and `outputs` Tensors. If true, these `Tensors` must be shaped `[max_time, batch_size, depth]`. If false, these `Tensors` must be shaped `[batch_size, max_time, depth]`. Using `time_major = True` is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.
- `scope`: VariableScope for the created subgraph; defaults to "bidirectional_rnn"

Returns:

A tuple (outputs, output_states) where: `outputs`*: A tuple (output_fw, output_bw) containing the forward and the backward rnn output* `Tensor`*. If time_major == False (default), output_fw will be a* `Tensor` *shaped:* `[batch_size, max_time, cell_fw.output_size]` *and output_bw will be a* `Tensor` *shaped:* `[batch_size, max_time, cell_bw.output_size]`*. If time_major == True, output_fw will be a* `Tensor` *shaped:* `[max_time, batch_size, cell_fw.output_size]` *and output_bw will be a* `Tensor` *shaped:* `[max_time, batch_size, cell_bw.output_size]`*. It returns a tuple instead of a single concatenated* `Tensor`*, unlike in the* `bidirectional_rnn`*. If the concatenated one is preferred, the forward and backward outputs can be concatenated as* `tf.concat(outputs, 2)`. `output_states`: A tuple (output_state_fw, output_state_bw) containing the forward and the backward final states of bidirectional rnn.

Raises:

- `TypeError`: If `cell_fw` or `cell_bw` is not an instance of `RNNCell`.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.