tf.contrib.rnn.LSTMBlockFusedCell

Class **LSTMBlockFusedCell**

Inherits From: `LSTMBlockWrapper`

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

See the guide: RNN and Cells (contrib) > Core RNN Cell wrappers (RNNCells that wrap other RNNCells)

FusedRNNCell implementation of LSTM.

This is an extremely efficient LSTM implementation, that uses a single TF op for the entire LSTM. It should be both faster and more memory-efficient than LSTMBlockCell defined above.

The implementation is based on: http://arxiv.org/abs/1409.2329.

We add forget_bias (default: 1) to the biases of the forget gate in order to reduce the scale of forgetting in the beginning of the training.

The variable naming is consistent with `rnn_cell_impl.LSTMCell`.

Properties

num_units

Number of units in this cell (output dimension).

Methods

__init__

```
__init__(
    num_units,
    forget_bias=1.0,
    cell_clip=None,
    use_peephole=False
)
```

Initialize the LSTM cell.

Args:

- `num_units`: int, The number of units in the LSTM cell.
- `forget_bias`: float, The bias added to forget gates (see above).
- `cell_clip`: clip the cell to this value. Default is no cell clipping.
- `use_peephole`: Whether to use peephole connections or not.

`__call__`

```
__call__(
    inputs,
    initial_state=None,
    dtype=None,
    sequence_length=None,
    scope=None
)
```

Run this LSTM on inputs, starting from the given state.

Args:

- `inputs`: `3-D` tensor with shape `[time_len, batch_size, input_size]` or a list of `time_len` tensors of shape `[batch_size, input_size]`.
- `initial_state`: a tuple `(initial_cell_state, initial_output)` with tensors of shape `[batch_size, self._num_units]`. If this is not provided, the cell is expected to create a zero initial state of type `dtype`.
- `dtype`: The data type for the initial state and expected output. Required if `initial_state` is not provided or RNN state has a heterogeneous dtype.
- `sequence_length`: Specifies the length of each sequence in inputs. An `int32` or `int64` vector (tensor) size `[batch_size]`, values in `[0, time_len).` Defaults to `time_len` for each element.
- `scope`: `VariableScope` for the created subgraph; defaults to class name.

Returns:

A pair containing:

- Output: A `3-D` tensor of shape `[time_len, batch_size, output_size]` or a list of time_len tensors of shape `[batch_size, output_size]`, to match the type of the `inputs`.
- Final state: a tuple `(cell_state, output)` matching `initial_state`.

Raises:

- `ValueError`: in case of shape mismatches

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**