tf.nn.static_rnn

Contents

Aliases:

Aliases:

- `tf.contrib.rnn.static_rnn`
- `tf.nn.static_rnn`

```
static_rnn(
    cell,
    inputs,
    initial_state=None,
    dtype=None,
    sequence_length=None,
    scope=None
)
```

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

See the guide: RNN and Cells (contrib) > Recurrent Neural Networks

Creates a recurrent neural network specified by RNNCell `cell`.

The simplest form of RNN network generated is:

```
  state = cell.zero_state(...)
  outputs = []
  for input_ in inputs:
    output, state = cell(input_, state)
    outputs.append(output)
  return (outputs, state)
```

However, a few other options are available:

An initial state can be provided. If the sequence_length vector is provided, dynamic calculation is performed. This method of calculation does not compute the RNN steps past the maximum sequence length of the minibatch (thus saving computational time), and properly propagates the state at an example's sequence length to the final state output.

The dynamic calculation performed is, at time `t` for batch row `b`,

```
  (output, state)(b, t) =
    (t >= sequence_length(b))
      ? (zeros(cell.output_size), states(b, sequence_length(b) - 1))
      : cell(input(b, t), state(b, t - 1))
```

Args:

- `cell`: An instance of RNNCell.
- `inputs`: A length T list of inputs, each a `Tensor` of shape `[batch_size, input_size]`, or a nested tuple of such elements.

- `initial_state`: (optional) An initial state for the RNN. If `cell.state_size` is an integer, this must be a `Tensor` of appropriate type and shape `[batch_size, cell.state_size]`. If `cell.state_size` is a tuple, this should be a tuple of tensors having shapes `[batch_size, s] for s in cell.state_size`.
- `dtype`: (optional) 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, T)`.
- `scope`: VariableScope for the created subgraph; defaults to "rnn".

Returns:

A pair (outputs, state) where:

- outputs is a length T list of outputs (one for each input), or a nested tuple of such elements.
- state is the final state

Raises:

- `TypeError`: If `cell` is not an instance of RNNCell.
- `ValueError`: If `inputs` is `None` or an empty list, or if the input depth (column size) cannot be inferred from inputs via shape inference.

Last updated November 2, 2017.