tf.contrib.rnn.MultiRNNCell

Class `MultiRNNCell`

Inherits From: `RNNCell` (https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/RNNCell)

Aliases:

- Class `tf.contrib.rnn.MultiRNNCell`
- Class `tf.nn.rnn_cell.MultiRNNCell`

Defined in `tensorflow/python/ops/rnn_cell_impl.py` (https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/python/ops/rnn_cell_impl.py).

See the guide: RNN and Cells (contrib) > Core RNN Cell wrappers (RNNCells that wrap other RNNCells) (https://www.tensorflow.org/api_guides/python/contrib.rnn#Core_RNN_Cell_wrappers_RNNCells_that_wrap_other_RNNCells_)

RNN cell composed sequentially of multiple simple cells.

Properties

`activity_regularizer`

Optional regularizer function for the output of this layer.

`dtype`

`graph`

`input`

Retrieves the input tensor(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer.

Returns:

Input tensor or list of input tensors.

Raises:

- **`AttributeError`**: if the layer is connected to more than one incoming layers.

Raises:

- **`RuntimeError`**: If called in Eager mode.
- **`AttributeError`**: If no inbound nodes are found.

`input_shape`

Retrieves the input shape(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer, or if all inputs have the same shape.

Returns:

Input shape, as an integer shape tuple (or list of shape tuples, one tuple per input tensor).

Raises:

- **`AttributeError`**: if the layer has no defined input_shape.
- **`RuntimeError`**: if called in Eager mode.

`losses`

`name`

`non_trainable_variables`

`non_trainable_weights`

`output`

Retrieves the output tensor(s) of a layer.

Only applicable if the layer has exactly one output, i.e. if it is connected to one incoming layer.

Returns:

Output tensor or list of output tensors.

Raises:

- **`AttributeError`**: if the layer is connected to more than one incoming layers.
- **`RuntimeError`**: if called in Eager mode.

`output_shape`

Retrieves the output shape(s) of a layer.

Only applicable if the layer has one output, or if all outputs have the same shape.

Returns:

Output shape, as an integer shape tuple (or list of shape tuples, one tuple per output tensor).

Raises:

- **`AttributeError`**: if the layer has no defined output shape.
- **`RuntimeError`**: if called in Eager mode.

`output_size`

`scope_name`

`state_size`

`trainable_variables`

`trainable_weights`

`updates`

`variables`

Returns the list of all layer variables/weights.

Returns:

A list of variables.

`weights`

Returns the list of all layer variables/weights.

Returns:

A list of variables.

Methods

`__init__`

```
__init__(
    cells,
    state_is_tuple=True
)
```

Create a RNN cell composed sequentially of a number of RNNCells.

Args:

- **`cells`**: list of RNNCells that will be composed in this order.
- **`state_is_tuple`**: If True, accepted and returned states are n-tuples, where `n = len(cells)`. If False, the states are all concatenated along the column axis. This latter behavior will soon be deprecated.

Raises:

- **`ValueError`**: if cells is empty (not allowed), or at least one of the cells returns a state tuple but the flag **`state_is_tuple`** is **`False`**.

`__call__`

```
__call__(
    inputs,
    state,
    scope=None
)
```

Run this RNN cell on inputs, starting from the given state.

Args:

- **`inputs`**: `2-D` tensor with shape **`[batch_size x input_size]`**.
- **`state`**: if **`self.state_size`** is an integer, this should be a **`2-D Tensor`** with shape **`[batch_size x self.state_size]`**. Otherwise, if **`self.state_size`** is a tuple of integers, this should be a tuple with shapes **`[batch_size x s] for s in self.state_size`**.
- **`scope`**: VariableScope for the created subgraph; defaults to class name.

Returns:

A pair containing:

- Output: A `2-D` tensor with shape **`[batch_size x self.output_size]`**.
- New state: Either a single `2-D` tensor, or a tuple of tensors matching the arity and shapes of **`state`**.

`__deepcopy__`

```
__deepcopy__(memo)
```

`add_loss`

```
add_loss(
    losses,
    inputs=None
)
```

Add loss tensor(s), potentially dependent on layer inputs.

Some losses (for instance, activity regularization losses) may be dependent on the inputs passed when calling a layer. Hence, when reusing a same layer on different inputs **a** and **b**, some entries in **layer.losses** may be dependent on **a** and some on **b**. This method automatically keeps track of dependencies.

The **get_losses_for** method allows to retrieve the losses relevant to a specific set of inputs.

Arguments:

- **losses**: Loss tensor, or list/tuple of tensors.
- **inputs**: Optional input tensor(s) that the loss(es) depend on. Must match the **inputs** argument passed to the **__call__** method at the time the losses are created. If **None** is passed, the losses are assumed to be unconditional, and will apply across all dataflows of the layer (e.g. weight regularization losses).

Raises:

- **RuntimeError**: If called in Eager mode.

add_update

```
add_update(
    updates,
    inputs=None
)
```

Add update op(s), potentially dependent on layer inputs.

Weight updates (for instance, the updates of the moving mean and variance in a BatchNormalization layer) may be dependent on the inputs passed when calling a layer. Hence, when reusing a same layer on different inputs **a** and **b**, some entries in **layer.updates** may be dependent on **a** and some on **b**. This method automatically keeps track of dependencies.

The **`get_updates_for`** method allows to retrieve the updates relevant to a specific set of inputs.

This call is ignored in Eager mode.

Arguments:

- **`updates`**: Update op, or list/tuple of update ops.
- **`inputs`**: Optional input tensor(s) that the update(s) depend on. Must match the **`inputs`** argument passed to the **`__call__`** method at the time the updates are created. If **`None`** is passed, the updates are assumed to be unconditional, and will apply across all dataflows of the layer.

`add_variable`

```
add_variable(
    name,
    shape,
    dtype=None,
    initializer=None,
    regularizer=None,
    trainable=True,
    constraint=None
)
```

Adds a new variable to the layer, or gets an existing one; returns it.

Arguments:

- **`name`**: variable name.
- **`shape`**: variable shape.
- **`dtype`**: The type of the variable. Defaults to **`self.dtype`** or **`float32`**.
- **`initializer`**: initializer instance (callable).
- **`regularizer`**: regularizer instance (callable).
- **`trainable`**: whether the variable should be part of the layer's "trainable_variables" (e.g. variables, biases) or "non_trainable_variables" (e.g. BatchNorm mean, stddev).
- **`constraint`**: constraint instance (callable).

Returns:

The created variable.

Raises:

- **`RuntimeError`**: If called in Eager mode with regularizers.

apply

```
apply(
    inputs,
    *args,
    **kwargs
)
```

Apply the layer on a input.

This simply wraps **`self.__call__`**.

Arguments:

- **`inputs`**: Input tensor(s).
- **`*args`**: additional positional arguments to be passed to **`self.call`**.
- **`**kwargs`**: additional keyword arguments to be passed to **`self.call`**.

Returns:

Output tensor(s).

build

```
build(_)
```

call

```
call(
    inputs,
```

```
    state
)
```

Run this multi-layer cell on inputs, starting from state.

count_params

```
count_params()
```

Count the total number of scalars composing the weights.

Returns:

An integer count.

Raises:

- **ValueError**: if the layer isn't yet built (in which case its weights aren't yet defined).

get_input_at

```
get_input_at(node_index)
```

Retrieves the input tensor(s) of a layer at a given node.

Arguments:

- **node_index**: Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

Returns:

A tensor (or list of tensors if the layer has multiple inputs).

Raises:

- **RuntimeError**: If called in Eager mode.

get_input_shape_at

```
get_input_shape_at(node_index)
```

Retrieves the input shape(s) of a layer at a given node.

Arguments:

- **node_index**: Integer, index of the node from which to retrieve the attribute. E.g. **node_index=0** will correspond to the first time the layer was called.

Returns:

A shape tuple (or list of shape tuples if the layer has multiple inputs).

Raises:

- **RuntimeError**: If called in Eager mode.

get_losses_for

```
get_losses_for(inputs)
```

Retrieves losses relevant to a specific set of inputs.

Arguments:

- **inputs**: Input tensor or list/tuple of input tensors. Must match the **inputs** argument passed to the **__call__** method at the time the losses were created. If you pass **inputs=None**, unconditional losses are returned, such as weight regularization losses.

Returns:

List of loss tensors of the layer that depend on **inputs**.

Raises:

- **RuntimeError**: If called in Eager mode.

get_output_at

```
get_output_at(node_index)
```

Retrieves the output tensor(s) of a layer at a given node.

Arguments:

- **`node_index`**: Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

Returns:

A tensor (or list of tensors if the layer has multiple outputs).

Raises:

- **`RuntimeError`**: If called in Eager mode.

get_output_shape_at

```
get_output_shape_at(node_index)
```

Retrieves the output shape(s) of a layer at a given node.

Arguments:

- **`node_index`**: Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

Returns:

A shape tuple (or list of shape tuples if the layer has multiple outputs).

Raises:

- **`RuntimeError`**: If called in Eager mode.

get_updates_for

```
get_updates_for(inputs)
```

Retrieves updates relevant to a specific set of inputs.

Arguments:

- **`inputs`**: Input tensor or list/tuple of input tensors. Must match the **`inputs`** argument passed to the **`__call__`** method at the time the updates were created. If you pass **`inputs=None`**, unconditional updates are returned.

Returns:

List of update ops of the layer that depend on **`inputs`**.

Raises:

- **`RuntimeError`**: If called in Eager mode.

zero_state

```
zero_state(
    batch_size,
    dtype
)
```

上次更新日期：十一月 2, 2017