tf.contrib.cudnn_rnn.CudnnRNNRelu

Contents

Class CudnnRNNRelu
Properties
direction
input_mode

Class **`CudnnRNNRelu`**

Defined in `tensorflow/contrib/cudnn_rnn/python/ops/cudnn_rnn_ops.py`.

Cudnn implementation of the RNN-relu model. Cudnn RNN has an opaque parameter buffer that can be used for inference and training. But it is possible that the layout of the parameter buffers changes between generations. So it is highly recommended to use CudnnOpaqueParamsSaveable to save and restore weights and biases in a canonical format.

This is a typical use case:

- The user creates a CudnnRNN model.
- The user query that parameter buffer size.
- The user creates a variable of that size that serves as the parameter buffers.
- The user either initialize the parameter buffer, or load the canonical weights into the parameter buffer.
- The user calls the model with the parameter buffer for inference, or training.
- If training, the user creates a Saver object.
- If training, the user creates a CudnnOpaqueParamsSaveable object from the parameter buffer for it to be later saved in the canonical format. When creating a CudnnOpaqueParamsSaveable object, a name could be provided, which is useful in distinguishing the names of multiple CudnnOpaqueParamsSaveable objects (e.g. for an encoder-decoder model).
- Once a while, the user saves the parameter buffer into model checkpoints with Saver.save().
- When restoring, the user creates a CudnnOpaqueParamsSaveable object and uses Saver.restore() to restore the parameter buffer from the canonical format to a user-defined format, as well as to restore other savable objects in the checkpoint file.

Properties

`direction`

`input_mode`

`input_size`

`num_layers`

`num_units`

`rnn_mode`

Methods

`__init__`

```
__init__(
    num_layers,
    num_units,
    input_size,
    input_mode=CUDNN_INPUT_LINEAR_MODE,
    direction=CUDNN_RNN_UNIDIRECTION,
    dtype=tf.float32,
    dropout=0.0,
    seed=0
)
```

Creates a Cudnn RNN model from model without hidden-state C.

Args:

- `num_layers` : the number of layers for the RNN model.
- `num_units` : the number of units within the RNN model.
- `input_size` : the size of the input, it could be different from the num_units.
- `input_mode` : indicate whether there is a linear projection between the input and The actual computation before the first layer. It could be 'skip_input', 'linear_input' or 'auto_select'. 'skip_input' is only allowed when input_size == num_units; 'auto_select' implies 'skip_input' when input_size == num_units; otherwise, it implies 'linear_input'.
- `direction` : the direction model that the model operates. Could be either 'unidirectional' or 'bidirectional'
- `dtype` : dtype of params, tf.float32 or tf.float64.
- `dropout` : whether to enable dropout. With it is 0, dropout is disabled.
- `seed` : the seed used for initializing dropout.

Raises:

- `ValueError` : if direction is not 'unidirectional' or 'bidirectional'.

`__call__`

```
__call__(
    input_data,
    input_h,
    params,
    is_training=True
)
```

Runs the forward step for the Cudnn LSTM model.

Args:

- `input_data` : the input sequence to the RNN model. A Tensor of shape [?, batch_size, input_size].

- `input_h` : the initial hidden state for h. A Tensor of shape [num_layers, batch_size, num_units].
- `params` : the parameter buffer created for this model.
- `is_training` : whether this operation will be used in training or inference.

Returns:

- `output` : the output sequuence.
- `output_h` : the final state for h.

`canonical_to_params`

```
canonical_to_params(
    weights,
    biases
)
```

Converts params from the canonical format to a specific format of cuDNN.

Args:

- `weights` : a Tensor for weight parameters.
- `biases` : a Tensor for bias parameters.

Returns:

A function for the canonical-to-params-to-specific conversion..

`params_size`

```
params_size()
```

Calculates the size of the opaque parameter buffer needed for this model.

Returns:

The calculated parameter buffer size.

`params_to_canonical`

```
params_to_canonical(params)
```

Converts params from a specific format of cuDNN to the canonical format.

Args:

- `params` : a Variable for weight and bias parameters.

Returns:

A function for the specific-to-canonical conversion.