Module: tf.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/experimental#top_of_page)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/experimental#functions)

Public API for tf.experimental namespace.

Functions

[function\_executor\_type(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/experimental/function_executor_type): Context manager for setting the executor of eager defined functions.

# tf.compat.v1.layers.experimental.keras\_style\_scope

Use Keras-style variable management.

tf.compat.v1.layers.experimental.keras\_style\_scope()

Defined in [python/layers/base.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/layers/base.py).

All tf.layers and tf RNN cells created in this scope use Keras-style variable management. Creating such layers with a scope= argument is disallowed, and reuse=True is disallowed.

The purpose of this scope is to allow users of existing layers to slowly transition to a Keras layers API without breaking existing functionality.

One example of this is when using TensorFlow's RNN classes with Keras Models or Networks. Because Keras models do not properly set variable scopes, users of RNNs may either accidentally share scopes between two different models, or get errors about variables that already exist.

#### Example:

class RNNModel(tf.keras.Model):  
  def \_\_init\_\_(self, name):  
    super(RNNModel, self).\_\_init\_\_(name=name)  
    self.rnn = tf.compat.v1.nn.rnn\_cell.MultiRNNCell(  
      [tf.compat.v1.nn.rnn\_cell.LSTMCell(64) for \_ in range(2)])  
  
  def call(self, input, state):  
    return self.rnn(input, state)  
  
model\_1 = RNNModel("model\_1")  
model\_2 = RNNModel("model\_2")  
# OK  
output\_1, next\_state\_1 = model\_1(input, state)  
# Raises an error about trying to create an already existing variable.  
output\_2, next\_state\_2 = model\_2(input, state)

The solution is to wrap the model construction and execution in a keras-style scope:

with keras\_style\_scope():  
  model\_1 = RNNModel("model\_1")  
  model\_2 = RNNModel("model\_2")  
  
  # model\_1 and model\_2 are guaranteed to create their own variables.  
  output\_1, next\_state\_1 = model\_1(input, state)  
  output\_2, next\_state\_2 = model\_2(input, state)  
  
  assert len(model\_1.weights) > 0  
  assert len(model\_2.weights) > 0  
  assert(model\_1.weights != model\_2.weights)

#### Yields:

A keras layer style scope.

# tf.compat.v1.layers.experimental.set\_keras\_style

Use Keras-style variable management.

tf.compat.v1.layers.experimental.set\_keras\_style()

Defined in [python/layers/base.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/layers/base.py).

All tf.layers and tf RNN cells created after keras style ha been enabled use Keras-style variable management. Creating such layers with a scope= argument is disallowed, and reuse=True is disallowed.

The purpose of this function is to allow users of existing layers to slowly transition to Keras layers API without breaking existing functionality.

For more details, see the documentation for keras\_style\_scope.

Note, once keras style has been set, it is set globally for the entire program and cannot be unset.

#### Example:

set\_keras\_style()  
  
model\_1 = RNNModel(name="model\_1")  
model\_2 = RNNModel(name="model\_2")  
# model\_1 and model\_2 are guaranteed to create their own variables.  
output\_1, next\_state\_1 = model\_1(input, state)  
output\_2, next\_state\_2 = model\_2(input, state)  
assert len(model\_1.weights) > 0  
assert len(model\_2.weights) > 0  
assert(model\_1.weights != model\_2.weights)

# tf.experimental.function\_executor\_type

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/experimental/function_executor_type#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/experimental/function_executor_type#aliases)

Context manager for setting the executor of eager defined functions.

Aliases:

* tf.compat.v1.experimental.function\_executor\_type
* tf.compat.v2.experimental.function\_executor\_type
* tf.experimental.function\_executor\_type

tf.experimental.function\_executor\_type(executor\_type)

Defined in [python/eager/context.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/eager/context.py).

Eager defined functions are functions decorated by tf.contrib.eager.defun.

#### Args:

* **executor\_type**: a string for the name of the executor to be used to execute functions defined by tf.contrib.eager.defun.

#### Yields:

Context manager for setting the executor of eager defined functions.