TensorFlow API r1.4

# tf.constant_initializer

## Class `constant_initializer`

Inherits From: `Initializer`

### Aliases:

- Class `tf.constant_initializer`
- Class `tf.initializers.constant`
- Class `tf.keras.initializers.Constant`

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

See the guide: Variables > Sharing Variables

Initializer that generates tensors with constant values.

The resulting tensor is populated with values of type `dtype`, as specified by arguments `value` following the desired `shape` of the new tensor (see examples below).

The argument `value` can be a constant value, or a list of values of type `dtype`. If `value` is a list, then the length of the list must be less than or equal to the number of elements implied by the desired shape of the tensor. In the case where the total number of elements in `value` is less than the number of elements required by the tensor shape, the last element in `value` will be used to fill the remaining entries. If the total number of elements in `value` is greater than the number of elements required by the tensor shape, the initializer will raise a `ValueError`.

### Args:

- `value`: A Python scalar, list of values, or a N-dimensional numpy array. All elements of the initialized variable will be set to the corresponding value in the `value` argument.
- `dtype`: The data type.
- `verify_shape`: Boolean that enables verification of the shape of `value`. If `True`, the initializer will throw an error if the shape of `value` is not compatible with the shape of the initialized tensor.

Examples: The following example can be rewritten using a numpy.ndarray instead of the `value` list, even reshaped, as shown in the two commented lines below the `value` list initialization.

```
  >>> import numpy as np
  >>> import tensorflow as tf

  >>> value = [0, 1, 2, 3, 4, 5, 6, 7]
  >>> # value = np.array(value)
  >>> # value = value.reshape([2, 4])
  >>> init = tf.constant_initializer(value)

  >>> print('fitting shape:')
  >>> with tf.Session():
  >>>   x = tf.get_variable('x', shape=[2, 4], initializer=init)
  >>>   x.initializer.run()
  >>>   print(x.eval())

  fitting shape:
  [[ 0.  1.  2.  3.]
   [ 4.  5.  6.  7.]]

  >>> print('larger shape:')
  >>> with tf.Session():
  >>>   x = tf.get_variable('x', shape=[3, 4], initializer=init)
  >>>   x.initializer.run()
  >>>   print(x.eval())

  larger shape:
  [[ 0.  1.  2.  3.]
   [ 4.  5.  6.  7.]
   [ 7.  7.  7.  7.]]

  >>> print('smaller shape:')
  >>> with tf.Session():
  >>>   x = tf.get_variable('x', shape=[2, 3], initializer=init)

* <b>`ValueError`</b>: Too many elements provided. Needed at most 6, but received 8

  >>> print('shape verification:')
  >>> init_verify = tf.constant_initializer(value, verify_shape=True)
  >>> with tf.Session():
  >>>   x = tf.get_variable('x', shape=[3, 4], initializer=init_verify)

* <b>`TypeError`</b>: Expected Tensor's shape: (3, 4), got (8,).
```

## Methods

### `__init__`

```
__init__(
    value=0,
    dtype=tf.float32,
    verify_shape=False
)
```

### `__call__`

```
__call__(
    shape,
    dtype=None,
    partition_info=None,
    verify_shape=None
)
```

## `from_config`

```
from_config(
    cls,
    config
)
```

Instantiates an initializer from a configuration dictionary.

Example:

```
initializer = RandomUniform(-1, 1)
config = initializer.get_config()
initializer = RandomUniform.from_config(config)
```

### Args:

- `config` : A Python dictionary. It will typically be the output of `get_config` .

### Returns:

An Initializer instance.

## `get_config`

```
get_config()
```

*Last updated November 2, 2017.*