Tensor types

class tf.DType

Represents the type of the elements in a Tensor.

The following DType objects are defined:

* tf.float16: 16-bit half-precision floating-point.
* tf.float32: 32-bit single-precision floating-point.
* tf.float64: 64-bit double-precision floating-point.
* tf.bfloat16: 16-bit truncated floating-point.
* tf.complex64: 64-bit single-precision complex.
* tf.complex128: 128-bit double-precision complex.
* tf.int8: 8-bit signed integer.
* tf.uint8: 8-bit unsigned integer.
* tf.uint16: 16-bit unsigned integer.
* tf.int16: 16-bit signed integer.
* tf.int32: 32-bit signed integer.
* tf.int64: 64-bit signed integer.
* tf.bool: Boolean.
* tf.string: String.
* tf.qint8: Quantized 8-bit signed integer.
* tf.quint8: Quantized 8-bit unsigned integer.
* tf.qint16: Quantized 16-bit signed integer.
* tf.quint16: Quantized 16-bit unsigned integer.
* tf.qint32: Quantized 32-bit signed integer.
* tf.resource: Handle to a mutable resource.

In addition, variants of these types with the \_ref suffix are defined for reference-typed tensors.

The tf.as\_dtype() function converts numpy types and string type names to a DType object.

tf.DType.is\_compatible\_with(other)

Returns True if the other DType will be converted to this DType.

The conversion rules are as follows:

DType(T)       .is\_compatible\_with(DType(T))        == True  
DType(T)       .is\_compatible\_with(DType(T).as\_ref) == True  
DType(T).as\_ref.is\_compatible\_with(DType(T))        == False  
DType(T).as\_ref.is\_compatible\_with(DType(T).as\_ref) == True

**Args:**

* **other**: A DType (or object that may be converted to a DType).

**Returns:**

True if a Tensor of the other DType will be implicitly converted to this DType.

tf.DType.name

Returns the string name for this DType.

tf.DType.base\_dtype

Returns a non-reference DType based on this DType.

tf.DType.real\_dtype

Returns the dtype correspond to this dtype's real part.

tf.DType.is\_floating

Returns whether this is a (non-quantized, real) floating point type.

tf.DType.is\_complex

Returns whether this is a complex floating point type.

tf.DType.is\_integer

Returns whether this is a (non-quantized) integer type.

tf.DType.is\_quantized

Returns whether this is a quantized data type.

tf.DType.is\_unsigned

Returns whether this type is unsigned.

Non-numeric, unordered, and quantized types are not considered unsigned, and this function returns False.

**Returns:**

Whether a DType is unsigned.

tf.DType.as\_numpy\_dtype

Returns a numpy.dtype based on this DType.

tf.DType.as\_datatype\_enum

Returns a types\_pb2.DataType enum value based on this DType.

tf.DType.limits

Return intensity limits, i.e. (min, max) tuple, of the dtype.

**Args:**

clip\_negative : bool, optional If True, clip the negative range (i.e. return 0 for min intensity) even if the image dtype allows negative values. Returns min, max : tuple Lower and upper intensity limits.

Other Methods

tf.DType.\_\_eq\_\_(other) {:#DType.**eq**}

Returns True iff this DType refers to the same type as other.

tf.DType.\_\_hash\_\_() {:#DType.**hash**}

tf.DType.\_\_init\_\_(type\_enum) {:#DType.**init**}

Creates a new DataType.

NOTE(mrry): In normal circumstances, you should not need to construct a DataType object directly. Instead, use the tf.as\_dtype() function.

**Args:**

* **type\_enum**: A types\_pb2.DataType enum value.

**Raises:**

* **TypeError**: If type\_enum is not a value types\_pb2.DataType.

tf.DType.\_\_ne\_\_(other) {:#DType.**ne**}

Returns True iff self != other.

tf.DType.\_\_repr\_\_() {:#DType.**repr**}

tf.DType.\_\_str\_\_() {:#DType.**str**}

tf.DType.is\_numpy\_compatible

tf.DType.max

Returns the maximum representable value in this data type.

**Raises:**

* **TypeError**: if this is a non-numeric, unordered, or quantized type.

tf.DType.min

Returns the minimum representable value in this data type.

**Raises:**

* **TypeError**: if this is a non-numeric, unordered, or quantized type.

tf.DType.size

tf.as\_dtype(type\_value)

Converts the given type\_value to a DType.

**Args:**

* **type\_value**: A value that can be converted to a tf.DType object. This may currently be a tf.DTypeobject, a [DataType enum](https://www.tensorflow.org/code/tensorflow/core/framework/types.proto), a string type name, or a numpy.dtype.

**Returns:**

A DType corresponding to type\_value.

**Raises:**

* **TypeError**: If type\_value cannot be converted to a DType.

Tipos de tensador

class tf.DType

Representa el tipo de elementos en a Tensor.

Los siguientes DTypeobjetos están definidos:

* tf.float16: Punto flotante de precisión media de 16 bits.
* tf.float32: Punto flotante de precisión simple de 32 bits.
* tf.float64: Punto flotante de doble precisión de 64 bits.
* tf.bfloat16: Punto flotante truncado de 16 bits.
* tf.complex64: Complejo de precisión simple de 64 bits.
* tf.complex128: Complejo de doble precisión de 128 bits.
* tf.int8: Entero con signo de 8 bits.
* tf.uint8: Entero sin signo de 8 bits.
* tf.uint16: Entero sin signo de 16 bits.
* tf.int16: Entero con signo de 16 bits.
* tf.int32: Entero con signo de 32 bits.
* tf.int64: Entero con signo de 64 bits.
* tf.bool: Booleano
* tf.string: Cuerda.
* tf.qint8: Entero cuantificado de 8 bits con signo.
* tf.quint8: Número entero sin signo de 8 bits cuantificado.
* tf.qint16: Entero cuantificado de 16 bits con signo.
* tf.quint16: Número entero sin signo de 16 bits cuantificado.
* tf.qint32: Número entero con signo de 32 bits cuantificado.
* tf.resource: Manejar a un recurso mutable.

Además, las variantes de estos tipos con el \_refsufijo se definen para los tensores de referencia.

La tf.as\_dtype()función convierte los tipos numpy y los nombres de tipo de cadena en un DTypeobjeto.

tf.DType.is\_compatible\_with(other)

Devuelve True si el otherDType se convertirá a este DType.

Las reglas de conversión son las siguientes:

DType(T)       .is\_compatible\_with(DType(T))        == True  
DType(T)       .is\_compatible\_with(DType(T).as\_ref) == True  
DType(T).as\_ref.is\_compatible\_with(DType(T))        == False  
DType(T).as\_ref.is\_compatible\_with(DType(T).as\_ref) == True

**Args:**

* **other**: A DType(u objeto que se puede convertir a a DType).

**Devoluciones:**

Verdadero si un Tensor de la other DTypevoluntad se convertirá implícitamente a esto DType.

tf.DType.name

Devuelve el nombre de la cadena para esto DType.

tf.DType.base\_dtype

Devuelve una no referencia DTypebasada en esto DType.

tf.DType.real\_dtype

Devuelve el dtype correspondiente a la parte real de este dtype.

tf.DType.is\_floating

Devuelve si este es un tipo de coma flotante (no cuantificado, real).

tf.DType.is\_complex

Devuelve si este es un tipo de punto flotante complejo.

tf.DType.is\_integer

Devuelve si este es un tipo entero (no cuantificado).

tf.DType.is\_quantized

Devuelve si este es un tipo de datos cuantificado.

tf.DType.is\_unsigned

Devuelve si este tipo no está firmado.

Los tipos no numéricos, desordenados y cuantificados no se consideran sin signo, y esta función retorna False.

**Devoluciones:**

Si a DTypeno está firmado.

tf.DType.as\_numpy\_dtype

Devuelve un numpy.dtypebasado en esto DType.

tf.DType.as\_datatype\_enum

Devuelve un types\_pb2.DataTypevalor enum basado en esto DType.

tf.DType.limits

Límites de intensidad de retorno, es decir, tupla (mínima, máxima) del tipo.

**Args:**

clip\_negative: bool, opcional Si es True, recorta el rango negativo (es decir, devuelve 0 para la intensidad mínima) incluso si la imagen de tipo d permite valores negativos. Devuelve min, max: tuple Límites de intensidad inferior y superior.

Otros metodos

tf.DType.\_\_eq\_\_(other){: #DType. **eq** }

Devuelve verdadero si este DType se refiere al mismo tipo que other.

tf.DType.\_\_hash\_\_(){: #DType. **hash** }

tf.DType.\_\_init\_\_(type\_enum){: #DType. **init** }

Crea una nueva DataType.

NOTA (mrry): en circunstancias normales, no debería necesitar construir un DataTypeobjeto directamente. En cambio, usa la tf.as\_dtype()función.

**Args:**

* **type\_enum**: Un types\_pb2.DataTypevalor enum.

**Subidas:**

* **TypeError**Si type\_enumno es un valor types\_pb2.DataType.

tf.DType.\_\_ne\_\_(other){: #DType. **ne** }

Devuelve True iff self! = Other.

tf.DType.\_\_repr\_\_(){: #DType. **repr** }

tf.DType.\_\_str\_\_(){: #DType. **str** }

tf.DType.is\_numpy\_compatible

tf.DType.max

Devuelve el valor máximo representable en este tipo de datos.

**Subidas:**

* **TypeError**: si se trata de un tipo no numérico, desordenado o cuantificado.

tf.DType.min

Devuelve el valor mínimo representable en este tipo de datos.

**Subidas:**

* **TypeError**: si se trata de un tipo no numérico, desordenado o cuantificado.

tf.DType.size

tf.as\_dtype(type\_value)

Convierte lo dado type\_valuea a DType.

**Args:**

* **type\_value**: Un valor que se puede convertir en un tf.DType objeto. Esto puede ser actualmente un tf.DTypeobjeto, una [DataTypeenumeración](https://www.tensorflow.org/code/tensorflow/core/framework/types.proto) , un nombre de tipo de cadena o a numpy.dtype.

**Devoluciones:**

A DTypecorrespondiente a type\_value.

**Subidas:**

* **TypeError**: Si type\_valueno se puede convertir a a DType.

tf.placeholder

placeholder(  
    dtype,  
    shape=None,  
    name=None  
)

Defined in [tensorflow/python/ops/array\_ops.py](https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/python/ops/array_ops.py).

See the guides: [Inputs and Readers > Placeholders](https://www.tensorflow.org/api_guides/python/io_ops#Placeholders), [Reading data > Feeding](https://www.tensorflow.org/api_guides/python/reading_data#Feeding)

Inserts a placeholder for a tensor that will be always fed.

**Important**: This tensor will produce an error if evaluated. Its value must be fed using the feed\_dict optional argument to Session.run(), Tensor.eval(), or Operation.run().

For example:

x = tf.placeholder(tf.float32, shape=(1024, 1024))  
y = tf.matmul(x, x)  
  
with tf.Session() as sess:  
  print(sess.run(y))  # ERROR: will fail because x was not fed.  
  
  rand\_array = np.random.rand(1024, 1024)  
  print(sess.run(y, feed\_dict={x: rand\_array}))  # Will succeed.

Args:

* **dtype**: The type of elements in the tensor to be fed.
* **shape**: The shape of the tensor to be fed (optional). If the shape is not specified, you can feed a tensor of any shape.
* **name**: A name for the operation (optional).

Returns:

A Tensor that may be used as a handle for feeding a value, but not evaluated directly.