4 Tensorflow 2 基础操作

4.1 数据类型

数据载体

- ✓ list
- ✓ np.array
- ✓ tf.Tensor

什么是 Tensor ?

✓ scalar	→ 1
✓ vector	→[1], [1,2…]
✓ matrix	→[[1,2,3],[4,5,6]]
✓ tensor	rank > 2

Tensorflow 中的基本数据类型

- ✓ int, float, doubles
- ✓ bool
- ✓ string

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import tensorflow as tf

print(tf.constant(1))
# tf.Tensor(1, shape=(), dtype=int32)
print(tf.constant(1.1))
tf.Tensor(1.1, shape=(), dtype=float32)
# print(tf.constant(2.2, dtype=tf.int32))
# TypeError: Cannot convert 2.2 to EagerTensor of dtype int32
print(tf.constant(2., dtype=tf.double))
# tf.Tensor(2.0, shape=(), dtype=float64)
print(tf.constant([True, False]))
# tf.Tensor([ True False], shape=(2,), dtype=bool)
print(tf.constant("hello world."))
# tf.Tensor(b'hello world.', shape=(), dtype=string)
```

Tensorflow 的属性

```
with tf.device("cpu"):
    a = tf.constant([1])
with tf.device("gpu"):
    b = tf.range(4)
print(a.device) # /job:localhost/replica:0/task:0/device:CPU:0
print(b.device) # /job:localhost/replica:0/task:0/device:GPU:0

aa = a.gpu()
print(aa.device) # /job:localhost/replica:0/task:0/device:GPU:0

bb = b.cpu()
print(bb.device) # /job:localhost/replica:0/task:0/device:CPU:0

print(b.numpy())
# [0 1 2 3]
print(b.ndim)
# 1
print(tf.rank(b))
# tf.Tensor(1, shape=(), dtype=int32)
print(tf.rank(tf.ones([3,4,2])))
# tf.Tensor(3, shape=(), dtype=int32)
print(b.name)
# AttributeError: Tensor.name is meaningless when eager execution is
enabled.
```

检查 Tensor 类型

```
a = tf.constant([1.])
b = tf.constant([True, False])
c = tf.constant("hello world.")
d = np.arange(4)

print(isinstance(a, tf.Tensor))     # True
print(tf.is_tensor(b))              # True
print(tf.is_tensor(d))              # False
print(a.dtype, b.dtype, c.dtype)
# <dtype: 'float32'> <dtype: 'bool'> <dtype: 'string'>

print(a.dtype == tf.float32)        # True
print(c.dtype == tf.string)         # True
```

Tensor 转换

```
a = np.arange(5)         # array([0 1 2 3 4])
print(a.dtype)           # int32

aa = tf.convert_to_tensor(a)
print(aa)            # tf.Tensor([0 1 2 3 4], shape=(5,), dtype=int32)
aa = tf.convert_to_tensor(a, dtype=tf.int64)
print(aa)            # tf.Tensor([0 1 2 3 4], shape=(5,), dtype=int64)

b = tf.cast(aa, dtype=tf.float32)
print(b)             # tf.Tensor([0. 1. 2. 3. 4.], shape=(5,), dtype=f
loat32)
c = tf.cast(aa, dtype=tf.double)
print(c)             # tf.Tensor([0. 1. 2. 3. 4.], shape=(5,), dtype=f
loat64)
d = tf.cast(c, dtype=tf.int32)
print(d)             # tf.Tensor([0 1 2 3 4], shape=(5,), dtype=int32)
```

bool ←→ int

```
b = tf.constant([0,1])
print(tf.cast(b, dtype=tf.bool))
# tf.Tensor([False  True], shape=(2,), dtype=bool)

bb = tf.cast(b, dtype=tf.bool)
print(tf.cast(bb, tf.int32))
# tf.Tensor([0 1], shape=(2,), dtype=int32)
```

tf.Variable

```
a = tf.range(5)
print(a)             # tf.Tensor([0 1 2 3 4], shape=(5,), dtype=int32)

b = tf.Variable(a)  # 具有如下的属性
print(b.dtype)       # <dtype: 'int32'>
print(b.name)        # Variable:0

b = tf.Variable(a, name='input_data')
print(b.name)        # input_data:0
print(b.trainable)   # True

print(isinstance(b, tf.Tensor))     # False, 所以不推荐使用 isinstance
print(isinstance(b, tf.Variable))   # True
print(tf.is_tensor(b))              # True,推荐 is_tensor 检测

print(b.numpy())                    # [0 1 2 3 4]
```

To numpy

```
a = tf.constant([[1,2],[3,4]])
# tf.Tensor(
# [[1 2]
#  [3 4]], shape=(2, 2), dtype=int32)
print(a.numpy(), a.dtype, type(a.numpy()))
# [[1 2]
#  [3 4]] <dtype: 'int32'> <class 'numpy.ndarray'>
b = tf.ones([])
print(b.numpy())            # 1.0
print(int(b.numpy()))       # 1
print(float(b.numpy()))     # 1.0
```

4.2 创建 Tensor

主要有如下几种创建方式：

- ✓ from numpy or list
- ✓ zeros, ones
- ✓ fill
- ✓ random
- ✓ constant
- ✓ Application

From Numpy, list

```
tf.convert_to_tensor(np.ones([2,3]))    # 从 numpy 创建
# <tf.Tensor: id=1, shape=(2, 3), dtype=float64, numpy=
# array([[1., 1., 1.],
#        [1., 1., 1.]])>
tf.convert_to_tensor(np.zeros([2,3]))   #从 numpy 创建
# <tf.Tensor: id=2, shape=(2, 3), dtype=float64, numpy=
# array([[0., 0., 0.],
#        [0., 0., 0.]])>
tf.convert_to_tensor([1,2])             # 从列表创建
# <tf.Tensor: id=3, shape=(2,), dtype=int32, numpy=array([1, 2])>
```

tf.zeros / tf.ones

```
tf.zeros([])
# <tf.Tensor: id=4, shape=(), dtype=float32, numpy=0.0>
tf.zeros([1])
# <tf.Tensor: id=7, shape=(1,), dtype=float32, numpy=array([0.], dtype=float32)>
tf.zeros([2,2])
# <tf.Tensor: id=10, shape=(2, 2), dtype=float32, numpy=
# array([[0., 0.],
#        [0., 0.]], dtype=float32)>
```

tf.zeros_like / tf.ones_like

```
a = tf.zeros([2,2])
tf.zeros_like(a)    #等同于下面的
# <tf.Tensor: id=18, shape=(2, 2), dtype=float32, numpy=
# array([[0., 0.],
#         [0., 0.]], dtype=float32)>
tf.zeros(a.shape)
# <tf.Tensor: id=21, shape=(2, 2), dtype=float32, numpy=
# array([[0., 0.],
#         [0., 0.]], dtype=float32)>
```

Fill

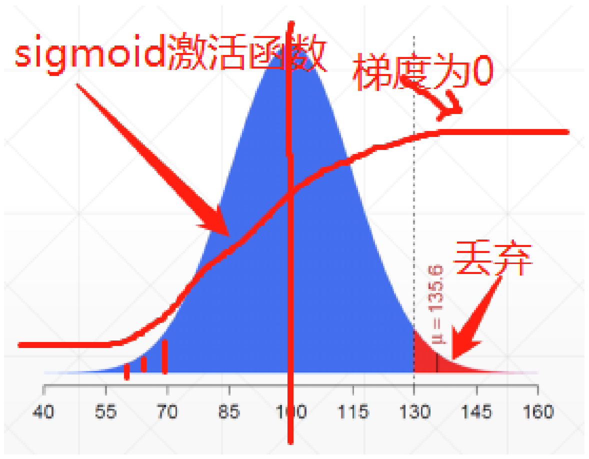
```
tf.fill([2,2],0.)       # 填充形状为(2,2)的值为0.的 tensor
# <tf.Tensor: id=24, shape=(2, 2), dtype=float32, numpy=
# array([[0., 0.],
#         [0., 0.]], dtype=float32)>
tf.fill([2,2],2)
# <tf.Tensor: id=27, shape=(2, 2), dtype=int32, numpy=
# array([[2, 2],
#         [2, 2]])>
```

Normal

```
tf.random.normal([2,2], mean=1, stddev=1)    # 正太分布
# <tf.Tensor: id=33, shape=(2, 2), dtype=float32, numpy=
# array([[ 0.752565 ,  1.9645684],
#         [-1.198905 ,  2.6976666]], dtype=float32)>
tf.random.truncated_normal([2,2],mean=0,stddev=1)    # 截断正太分布
# <tf.Tensor: id=39, shape=(2, 2), dtype=float32, numpy=
# array([[ 0.13609798, -1.1503961 ],
#         [ 1.2531655 ,  1.213268  ]], dtype=float32)>
```

Truncated Distribution

该分布主要为了防止梯度消失(Gradient Vanish)



Uniform

```
tf.random.uniform([2,2], minval=0, maxval=1)  # 均匀分布，从 0~1 均匀采样
# <tf.Tensor: id=46, shape=(2, 2), dtype=float32, numpy=
# array([[0.01785529, 0.36638904],
#         [0.24064624, 0.3041246 ]], dtype=float32)>
```

Random Permutation

```
idx = tf.range(10)

idx = tf.random.shuffle(idx)
# <tf.Tensor: id=51, shape=(10,), dtype=int32, numpy=array([4, 8, 0,
3, 5, 2, 6, 9, 1, 7])>

a = tf.random.normal([10,784])
b = tf.random.uniform([10],maxval=10,dtype=tf.int32)
# <tf.Tensor: id=61, shape=(10,), dtype=int32, numpy=array([0, 2, 9,
6, 4, 6, 4, 8, 7, 7])>

a = tf.gather(a,idx)    # 获取对应索引的数据
tf.gather(b,idx)
# <tf.Tensor: id=65, shape=(10,), dtype=int32, numpy=array([4, 7, 0,
6, 6, 9, 4, 7, 2, 8])>
```

Typical Dim Data

✓	[]	←→ Scalar
✓	[d]	←→ Vector
✓	[h, w]	←→ Matrix
✓	[b, len, vec]	←→ 3 Tensor
✓	[b, h, w, c]	←→ 4 Tensor
✓	[t, b, h, w, c]	←→ 5 Tensor
✓	…	

4.3 索引与切片

4.3.1 索引和切片 I

✓	基本索引	←→ [idx][idx]
✓	start:end	←→ :
✓	start:end:step	←→ : :
✓	…	←→ …

基本索引

```
a = tf.ones([2,2,2])

a[0][0]
# <tf.Tensor: id=76, shape=(2,), dtype=float32, numpy=array([1., 1.],
 dtype=float32)>
```

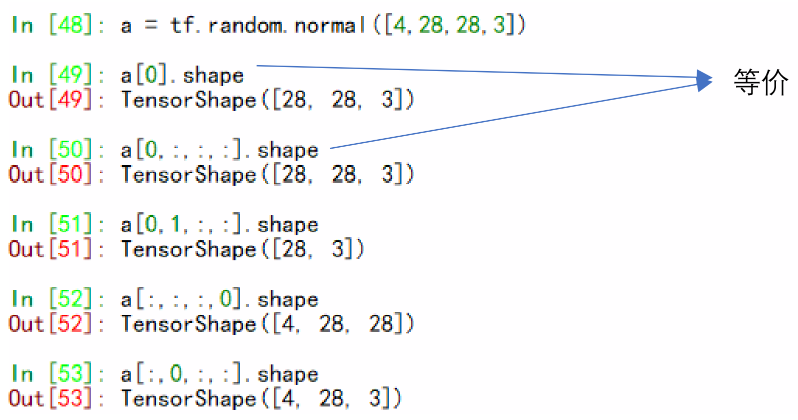
Numpy-style-indexing

```
a = tf.random.normal([2,3,4])
a[1].shape
# TensorShape([3, 4])
a[1,2].shape
# TensorShape([4])
a[1,2,3].shape
# TensorShape([])     # 是一个标量，即一个数值
```

start:end

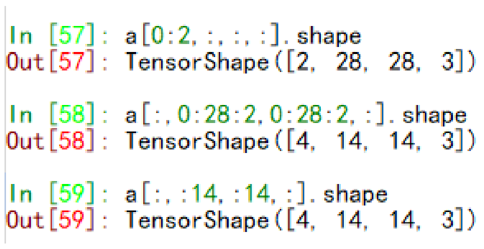
```
a = tf.range(10)
# <tf.Tensor: id=104, shape=(10,), dtype=int32, numpy=array([0, 1, 2,
 3, 4, 5, 6, 7, 8, 9])>
a[-1:]
# <tf.Tensor: id=108, shape=(1,), dtype=int32, numpy=array([9])>
a[-2:]          # 从倒数第二个取
# <tf.Tensor: id=112, shape=(2,), dtype=int32, numpy=array([8, 9])>
a[:2]           # 取前两个元素
# <tf.Tensor: id=116, shape=(2,), dtype=int32, numpy=array([0, 1])>
a[:-1]          # 最后一个元素不取
# <tf.Tensor: id=120, shape=(9,), dtype=int32, numpy=array([0, 1, 2,
3, 4, 5, 6, 7, 8])>
```

Indexing by ：

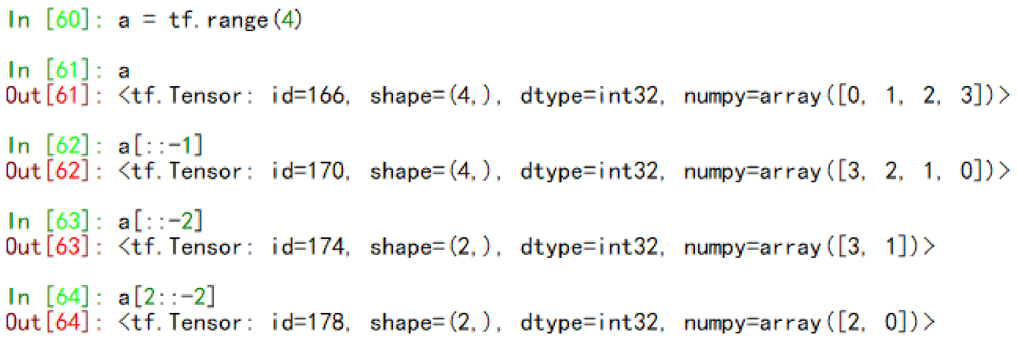


```
In [48]: a = tf.random.normal([4,28,28,3])

In [49]: a[0].shape
Out[49]: TensorShape([28, 28, 3])

In [50]: a[0,:,:,:].shape
Out[50]: TensorShape([28, 28, 3])

In [51]: a[0,1,:,:].shape
Out[51]: TensorShape([28, 3])

In [52]: a[:,:,:,0].shape
Out[52]: TensorShape([4, 28, 28])

In [53]: a[:,0,:,:].shape
Out[53]: TensorShape([4, 28, 3])
```

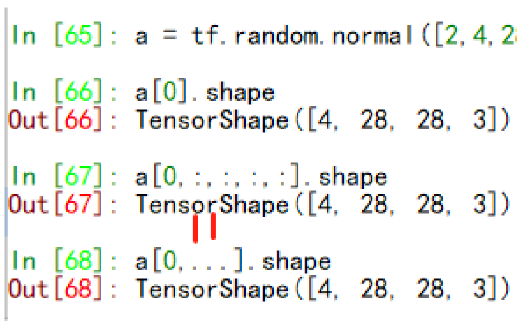
Indexing by :: and …



```
In [57]: a[0:2,:,:,:].shape
Out[57]: TensorShape([2, 28, 28, 3])

In [58]: a[:,0:28:2,0:28:2,:].shape
Out[58]: TensorShape([4, 14, 14, 3])

In [59]: a[:,:14,:14,:].shape
Out[59]: TensorShape([4, 14, 14, 3])
```



```
In [60]: a = tf.range(4)

In [61]: a
Out[61]: <tf.Tensor: id=166, shape=(4,), dtype=int32, numpy=array([0, 1, 2, 3])>

In [62]: a[::-1]
Out[62]: <tf.Tensor: id=170, shape=(4,), dtype=int32, numpy=array([3, 2, 1, 0])>

In [63]: a[::-2]
Out[63]: <tf.Tensor: id=174, shape=(2,), dtype=int32, numpy=array([3, 1])>

In [64]: a[2::-2]
Out[64]: <tf.Tensor: id=178, shape=(2,), dtype=int32, numpy=array([2, 0])>
```



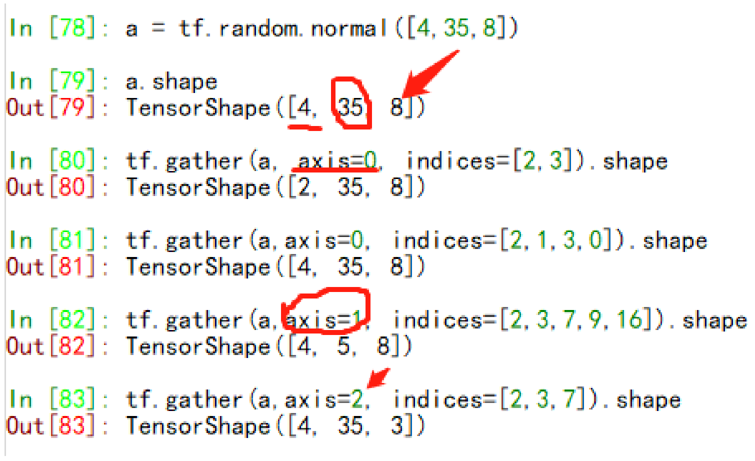
```
In [65]: a = tf.random.normal([2,4,28,28,3])

In [66]: a[0].shape
Out[66]: TensorShape([4, 28, 28, 3])

In [67]: a[0,:,:,:,:].shape
Out[67]: TensorShape([4, 28, 28, 3])

In [68]: a[0,...].shape
Out[68]: TensorShape([4, 28, 28, 3])
```

4.4 索引与切片 II

选择性索引：

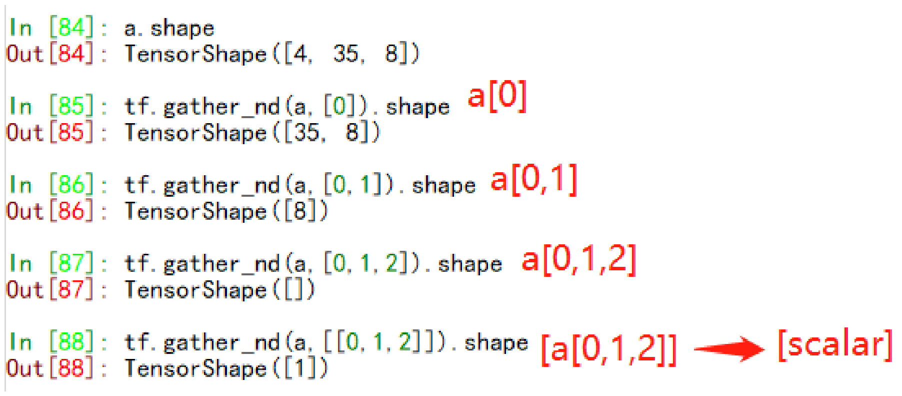
- ✓ tf.gather
- ✓ tf.gather_nd
- ✓ tf.boolean_mask

tf.gather

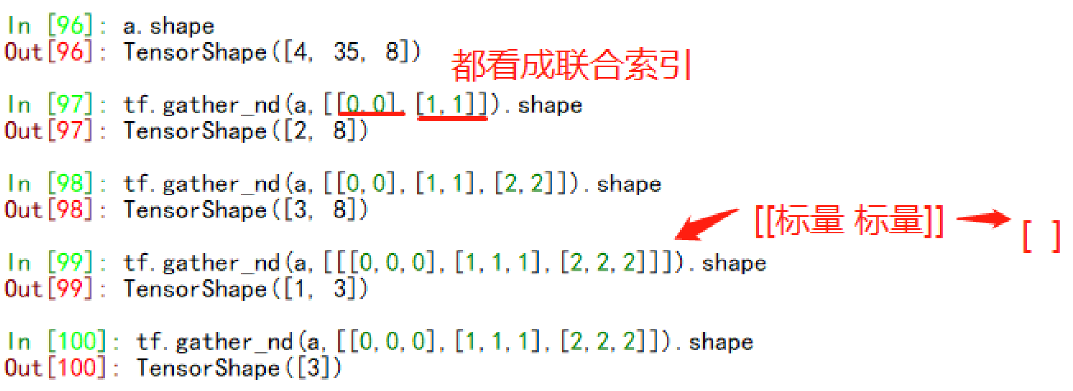


```
In [78]: a = tf.random.normal([4,35,8])

In [79]: a.shape
Out[79]: TensorShape([4, 35, 8])

In [80]: tf.gather(a, axis=0, indices=[2,3]).shape
Out[80]: TensorShape([2, 35, 8])

In [81]: tf.gather(a,axis=0, indices=[2,1,3,0]).shape
Out[81]: TensorShape([4, 35, 8])

In [82]: tf.gather(a,axis=1, indices=[2,3,7,9,16]).shape
Out[82]: TensorShape([4, 5, 8])

In [83]: tf.gather(a,axis=2, indices=[2,3,7]).shape
Out[83]: TensorShape([4, 35, 3])
```

tf.gather_nd



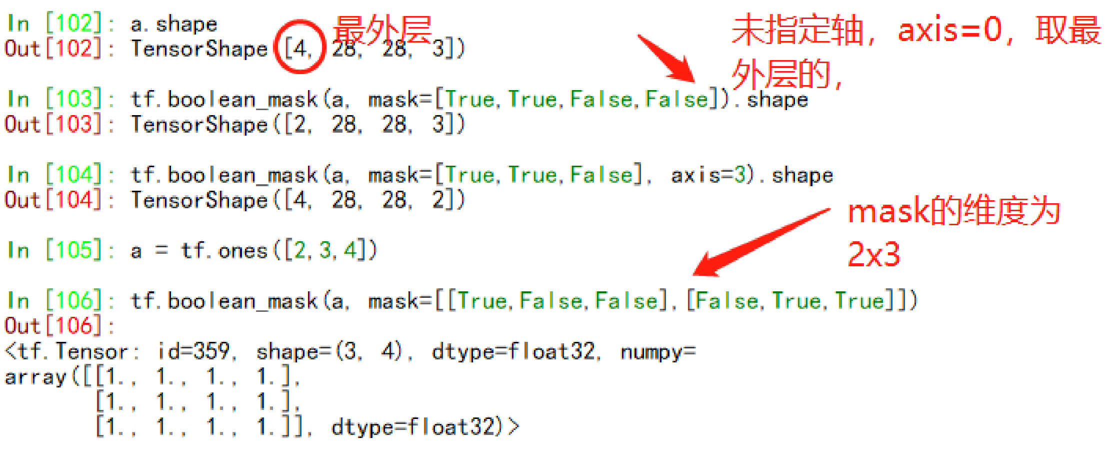
```
In [84]: a.shape
Out[84]: TensorShape([4, 35, 8])

In [85]: tf.gather_nd(a,[0]).shape
Out[85]: TensorShape([35, 8])

In [86]: tf.gather_nd(a,[0,1]).shape
Out[86]: TensorShape([8])

In [87]: tf.gather_nd(a,[0,1,2]).shape
Out[87]: TensorShape([])

In [88]: tf.gather_nd(a,[[0,1,2]]).shape
Out[88]: TensorShape([1])
```



```
In [96]: a.shape
Out[96]: TensorShape([4, 35, 8])

In [97]: tf.gather_nd(a,[[0,0],[1,1]]).shape
Out[97]: TensorShape([2, 8])

In [98]: tf.gather_nd(a,[[0,0],[1,1],[2,2]]).shape
Out[98]: TensorShape([3, 8])

In [99]: tf.gather_nd(a,[[[0,0,0],[1,1,1],[2,2,2]]]).shape
Out[99]: TensorShape([1, 3])

In [100]: tf.gather_nd(a,[[0,0,0],[1,1,1],[2,2,2]]).shape
Out[100]: TensorShape([3])
```

tf.boolean_mask

mask 的维度和取的维度一致的取。

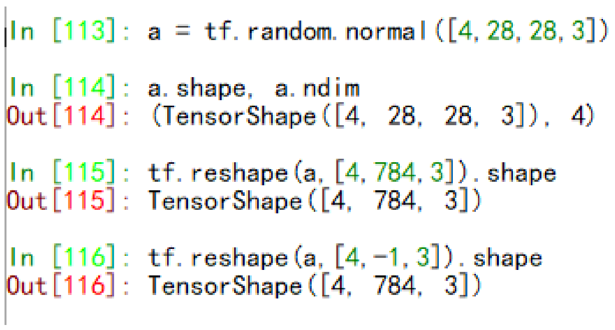


```
In [102]: a.shape
Out[102]: TensorShape([4, 28, 28, 3])

In [103]: tf.boolean_mask(a, mask=[True,True,False,False]).shape
Out[103]: TensorShape([2, 28, 28, 3])

In [104]: tf.boolean_mask(a, mask=[True,True,False], axis=3).shape
Out[104]: TensorShape([4, 28, 28, 2])

In [105]: a = tf.ones([2,3,4])

In [106]: tf.boolean_mask(a, mask=[[True,False,False],[False,True,True]])
Out[106]:
<tf.Tensor: id=359, shape=(3, 4), dtype=float32, numpy=
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]], dtype=float32)>
```

4.4 维度变换

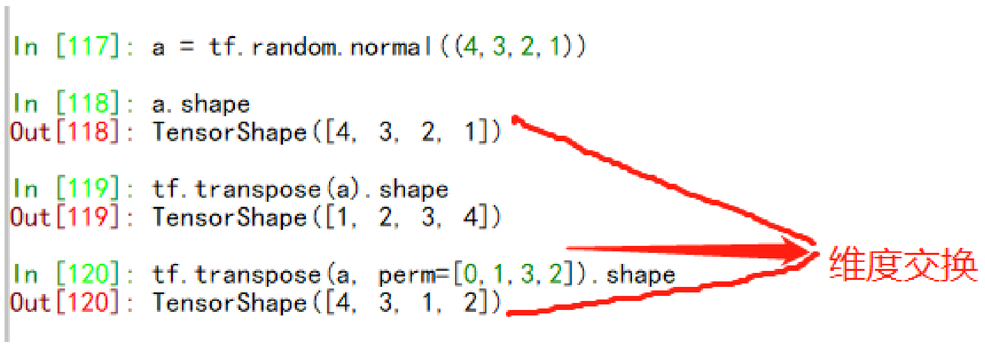
- ✓ shape, ndim
- ✓ reshape
- ✓ expand_dims / squeeze
- ✓ transpose
- ✓ broadcast_to

tf.reshape



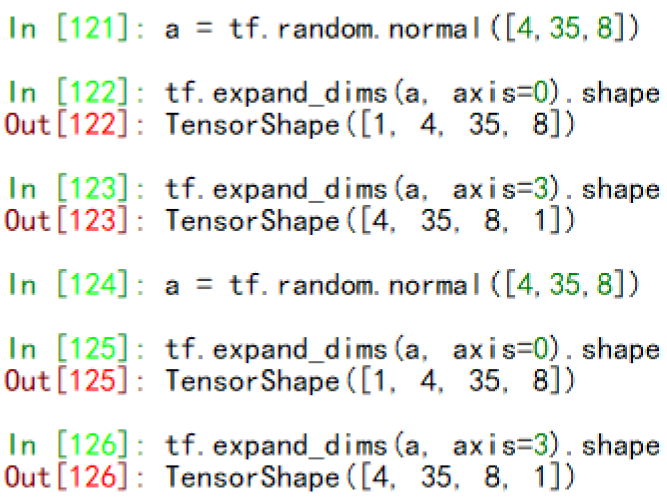
```
In [113]: a = tf.random.normal([4,28,28,3])

In [114]: a.shape, a.ndim
Out[114]: (TensorShape([4, 28, 28, 3]), 4)

In [115]: tf.reshape(a,[4,784,3]).shape
Out[115]: TensorShape([4, 784, 3])

In [116]: tf.reshape(a,[4,-1,3]).shape
Out[116]: TensorShape([4, 784, 3])
```

tf.transpose



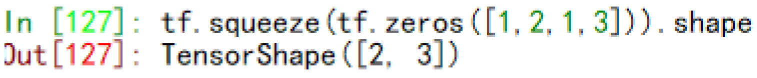
```
In [117]: a = tf.random.normal((4,3,2,1))

In [118]: a.shape
Out[118]: TensorShape([4, 3, 2, 1])

In [119]: tf.transpose(a).shape
Out[119]: TensorShape([1, 2, 3, 4])

In [120]: tf.transpose(a, perm=[0,1,3,2]).shape
Out[120]: TensorShape([4, 3, 1, 2])
```

Squeeze and Expand_dims

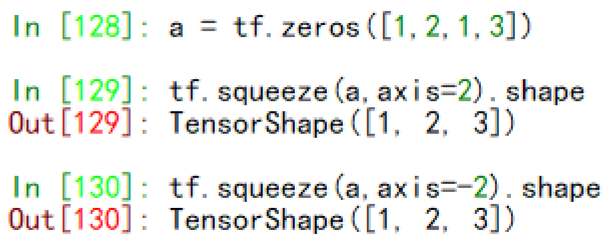


```
In [121]: a = tf.random.normal([4,35,8])

In [122]: tf.expand_dims(a, axis=0).shape
Out[122]: TensorShape([1, 4, 35, 8])

In [123]: tf.expand_dims(a, axis=3).shape
Out[123]: TensorShape([4, 35, 8, 1])

In [124]: a = tf.random.normal([4,35,8])

In [125]: tf.expand_dims(a, axis=0).shape
Out[125]: TensorShape([1, 4, 35, 8])

In [126]: tf.expand_dims(a, axis=3).shape
Out[126]: TensorShape([4, 35, 8, 1])
```

对于 squeeze，只有 shape=1 的维度能够被 squeeze.



```
In [127]: tf.squeeze(tf.zeros([1,2,1,3])).shape
Out[127]: TensorShape([2, 3])
```



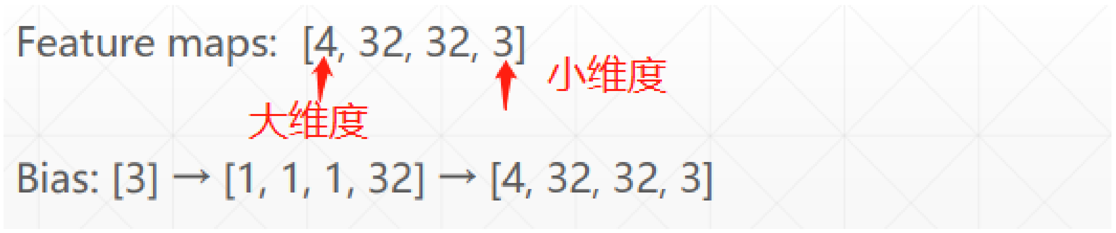
```
In [128]: a = tf.zeros([1,2,1,3])

In [129]: tf.squeeze(a,axis=2).shape
Out[129]: TensorShape([1, 2, 3])

In [130]: tf.squeeze(a,axis=-2).shape
Out[130]: TensorShape([1, 2, 3])
```

broadcast_to

1、expand
2、without copying data
3、tf.broadcast_to

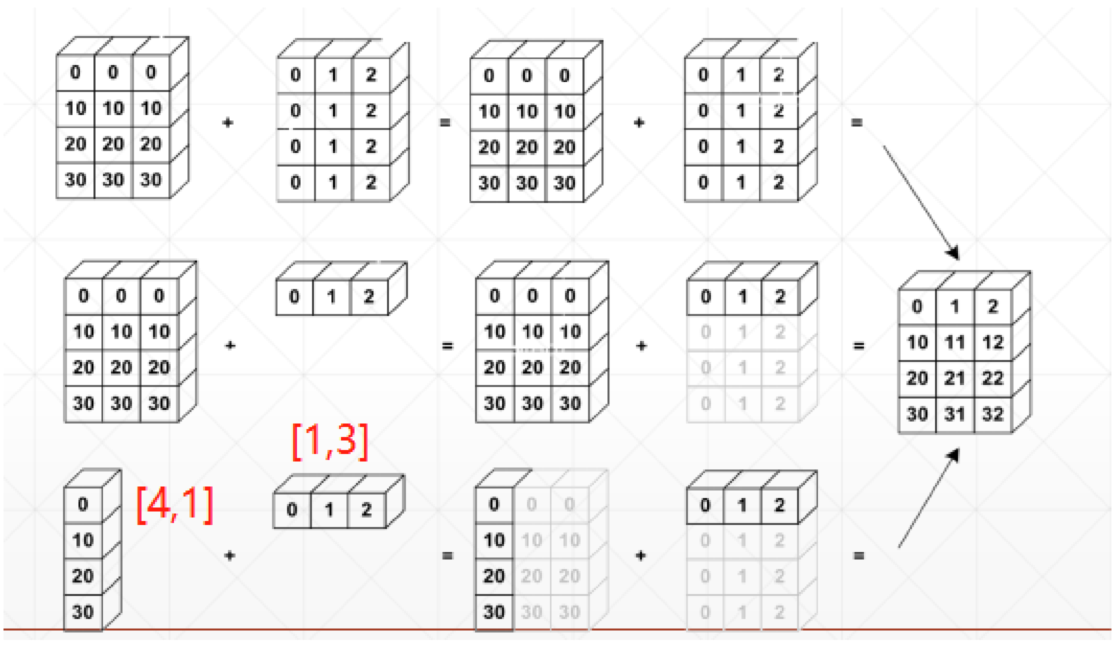
关键点

- insert 1 dim ahead if needed
- expand dims with size 1 to the same size



Feature maps: [4, 32, 32, 3]

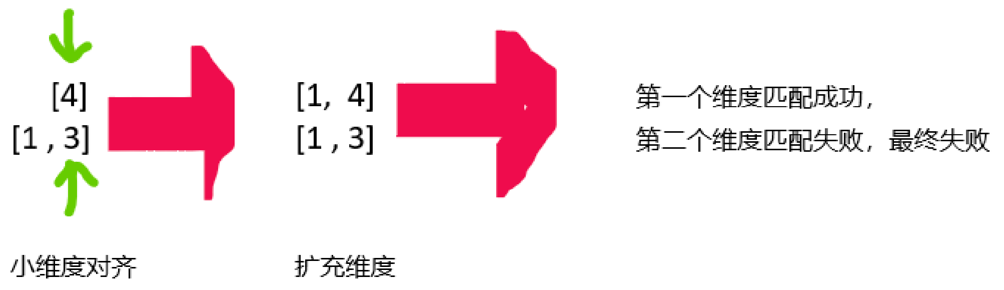
Bias: [3] → [1, 1, 1, 32] → [4, 32, 32, 3]



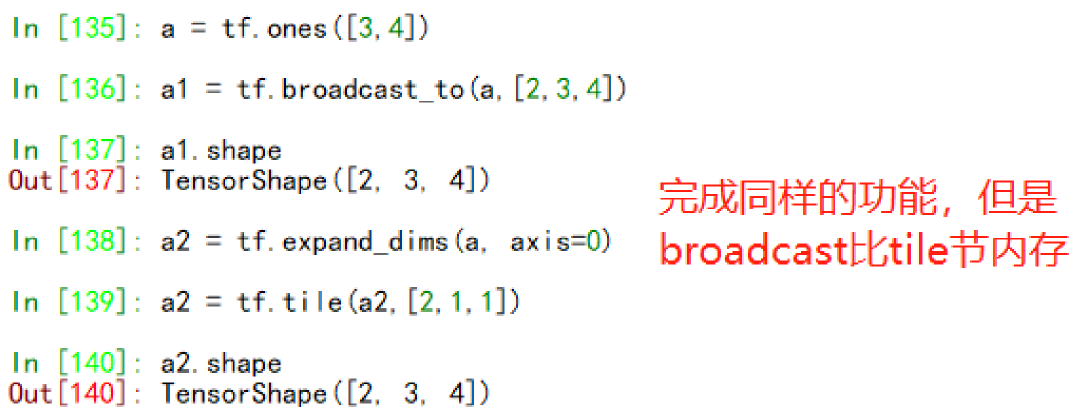
要点：**Match from last dim(即从小维度开始)**

- **if current dim=1, expand to same(维度为 1，可以扩充)**
- **if either hos no dim, insert one dim and expand to same(没有维度，插入维度)**
- **otherwise, NOT broadcastable**

比如维度分别为[4]和[1,3]矩阵，首先从小维度开始对齐匹配，然后扩充维度。



Broadcast VS Tile



```
In [135]: a = tf.ones([3,4])

In [136]: a1 = tf.broadcast_to(a, [2,3,4])

In [137]: a1.shape
Out[137]: TensorShape([2, 3, 4])

In [138]: a2 = tf.expand_dims(a, axis=0)

In [139]: a2 = tf.tile(a2, [2,1,1])

In [140]: a2.shape
Out[140]: TensorShape([2, 3, 4])
```

完成同样的功能，但是broadcast比tile节内存

4.5 前向传播(张量)

```
# 实现前向传播
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets
# x : [60k,28,28], y : [60k]
(x, y), _ = datasets.mnist.load_data()
# x : [0~255] => [0~1]
x = tf.convert_to_tensor(x, dtype=tf.float32) / 255.
y = tf.convert_to_tensor(y, dtype=tf.int32)
train_db = tf.data.Dataset.from_tensor_slices((x, y)).batch(128)
train_iter = iter(train_db)     # 创建一个迭代器，可以使用 next 进行迭代
sample = next(train_iter)
print("batch:", sample[0].shape, sample[1].shape)
# [b,784] => [b,256] => [b,128] => [b,10]
# w = [dim_in, dim_out] ; b = [dim_out]
w1 = tf.Variable(tf.random.truncated_normal([784,256],stddev=0.1))
b1 = tf.Variable(tf.zeros([256]))
```

```
w2 = tf.Variable(tf.random.truncated_normal([256,128],stddev=0.1))
b2 = tf.Variable(tf.zeros([128]))
w3 = tf.Variable(tf.random.truncated_normal([128,10],stddev=0.1))
b3 = tf.Variable(tf.zeros([10]))

lr = 1e-3

for epoch in range(10):     # iterate 数据集 for 10 次
    for step,(x,y) in enumerate(train_db):      # for every batch
        x = tf.reshape(x, [-1, 28*28])
        with tf.GradientTape() as tape:
            h1 = x@w1 + b1
            h1 = tf.nn.relu(h1)
            h2 = h1@w2 + b2
            h1 = tf.nn.relu(h2)
            out = h2@w3 + b3

            # 计算误差
            # out: [b,10] ;  y: [10]
            y_onehot = tf.one_hot(y, depth=10)

            loss = tf.square(y_onehot - out)        # loss: [b,10]
            # mean ==> scalar
            loss = tf.reduce_mean(loss)

        # compute gradients
        grads = tape.gradient(loss,[w1,b1,w2,b2,w3,b3])
        # w = w - lr * w_grad
        # w1 = w1 - lr * grads[0]
        # #报错一: 'float' and 'NoneType',(grads[0]是 NoneType 类型),
        # 因为 tensorflow 只会跟踪 tf.Variable 类型, 而 w1 未加
tf.Variable()之前是 tensor 类型
        # 报错二, 是更新了之后又变为了 tensor 类型, 因此需要原地更新函数
assign_sub()
        # #
        w1.assign_sub(lr * grads[0])
        b1.assign_sub(lr * grads[1])
        w2.assign_sub(lr * grads[2])
        b2.assign_sub(lr * grads[3])
        w3.assign_sub(lr * grads[4])
        b3.assign_sub(lr * grads[5])

        if step % 100 == 0:
            print(epoch, step, 'loss:', float(loss))
```