2-3 數據操作

確認某層的權重形狀是否如預期，可使用下面兩種語法 1) 類似 python 基本slice 方式選取

In [3]:

```
import tensorflow as tf
#Example for select instance
a = tf.ones([1,5,5,3])
a[0][0].shape
#Output shape:TensorShape([5, 3])
a[...,2].shape
#Output shape:TensorShape([1, 5, 5])
```

Out[3]:

```
TensorShape([1, 5, 5])
```

2) 使用tensorflow API 操作索引

In [4]:

```
#use index select
tf.gather_nd(a,[0]).shape
#Output shape: TensorShape([5, 5, 3])

#select axis=1 , row 2 and 3
tf.gather(a,axis=1,indices=[2,3]).shape
#Output shape: TensorShape([1, 2, 5, 3])
```

Out[4]:

```
TensorShape([1, 2, 5, 3])
```

將資料中欄位對調

In [5]:

```
tf.transpose(a, perm=[0,1,3,2]).shape
#Output: TensorShape([1, 5, 3, 5])
```

Out[5]:

```
TensorShape([1, 5, 3, 5])
```

增加資料維度

In [7]:

```
a = tf.random.normal([32,32,3])
tf.expand_dims(a,axis=0).shape
#output:TensorShape([1, 32, 32, 3])
```

Out[7]:

```
TensorShape([1, 32, 32, 3])
```

兩個資料合併

tf.concat

In [10]:

```
a= tf.ones([6,32,3])
b= tf.ones([6,32,3])
tf.concat([a, b], axis=0).shape
#output: TensorShape([12, 32, 3])
```

Out[10]:

```
TensorShape([12, 32, 3])
```

透過axis 來調控合併哪軸axis 為標準

In [11]:

```
a= tf.ones([6,32,3])
b= tf.ones([6,32,3])
tf.concat([a, b], axis=1).shape
#output: TensorShape([6, 64, 3])
```

Out[11]:

```
TensorShape([6, 64, 3])
```

tf.stack

In [16]:

```
a= tf.ones([6,32,3])
b= tf.ones([6,32,3])
tf.stack([a, b], axis=1).shape
#output: TensorShape([2, 6, 32, 3])
```

Out[16]:

```
TensorShape([6, 2, 32, 3])
```

分割資料

unstack

In [26]:

```
a = tf.ones([2,32,3])
aa,ab = tf.unstack(a,axis=0)
aa.shape
#output: TensorShape([32, 3])
data_all = tf.unstack(a, axis= 1)
data_all[0].shape
#output: TensorShape([2, 3])
len(data_all)
#output: 32
```

Out[26]:

```
32
```

split:

直接指定切割份數

In [29]:

```
data_all = tf.split(a, axis=1, num_or_size_splits=2)
data_all[0].shape
#output: TensorShape([2, 16, 3])
len(data_all)
#output: 2
```

Out[29]:

```
2
```

In [ ]:

```
重整維度

Note: 維度不清楚，可使用-1自動產生
```

In [33]:

```
#Create samples
a = tf.random.normal([6,32,32,3])
#Resphape to different type
tf.reshape(a, [6, 32*32, 3]).shape
#output : TensorShape([6, 1024, 3])
tf.reshape(a,[6,-1,3]).shape
#output : TensorShape([6, 1024, 3])
```

Out[33]:

```
TensorShape([6, 1024, 3])
```

Sorting

In [34]:

```
#Example for sort
data = tf.random.normal([10],mean=0,stddev=1)
tf.sort(data,direction='DESCENDING')
#or
tf.gather(data,tf.argsort(data,direction='ASCENDING'))
```

Out[34]:

```
<tf.Tensor: id=402, shape=(10,), dtype=float32, numpy=
array([-1.135849  , -0.8082828 , -0.5859008 , -0.546737  , -0.30197838,
        0.22137827,  0.4376939 ,  0.46138355,  1.065124  ,  1.4488674 ],
      dtype=float32)>
```

In [ ]:

```
拿取 上面k 個 值
```

In [20]:

```
#Example top_k
top_data = tf.math.top_k(data,k=5)
#Get indies or values
top_data.indices
#or
top_data.values
```

Out[20]:

```
<tf.Tensor: shape=(5,), dtype=float32, numpy=
array([0.8253954 , 0.6794364 , 0.40747476, 0.38604498, 0.2376566 ],
      dtype=float32)>
```

padding: 補齊

In [4]:

```
tf.random.normal([3,3],mean=0,stddev=1)
```

Out[4]:

```
<tf.Tensor: id=25, shape=(3, 3), dtype=float32, numpy=
array([[ 0.34054196, -0.04776992, -0.28590366],
       [ 0.38979727,  0.34608367,  0.14123367],
       [ 0.178043  ,  0.37295932, -1.0362846 ]], dtype=float32)>
```

In [5]:

```
import tensorflow as tf
data = tf.random.normal([3,3],mean=0,stddev=1)
tf.pad(data,[[1,2],[3,4]])
```

Out[5]:

```
<tf.Tensor: id=33, shape=(6, 10), dtype=float32, numpy=
array([[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , -1.1288279 , -0.58975124,
        -1.6395129 ,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  1.407286  ,  2.29677   ,
        -0.23337239,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.2617132 , -0.4755918 ,
         0.39954367,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ]],
      dtype=float32)>
```

Clip 裁切

In [6]:

```
#Example clipping
tf.clip_by_value(data,0,1)
#range if number less or above fix at 0 or 1
```

Out[6]:

```
<tf.Tensor: id=37, shape=(3, 3), dtype=float32, numpy=
array([[0.        , 0.        , 0.        ],
       [1.        , 1.        , 0.        ],
       [0.2617132 , 0.        , 0.39954367]], dtype=float32)>
```

2-4 數據運算

基本運算

1. Element wise: 相對元素運算

In [7]:

```
a = tf.fill([2,2],5)
b = tf.fill([2,2],6)
a+b,a*b
```

Out[7]:

```
(<tf.Tensor: id=44, shape=(2, 2), dtype=int32, numpy=
 array([[11, 11],
        [11, 11]])>,
 <tf.Tensor: id=45, shape=(2, 2), dtype=int32, numpy=
 array([[30, 30],
        [30, 30]])>)
```

2. Matrix wise: 矩陣運算

In [25]:

```
a = tf.fill([2,32,4],5)
b = tf.fill([2,4,6],6)
(a@b).shape # or tf.matmul(a,b)
#Output:TensorShape([2, 32, 6])
```

Out[25]:

```
TensorShape([2, 32, 6])
```

3. Dimension wise: 取平均值 或 準確度

In [8]:

```
#Example for dimension wise
data = [[1,2,3],
      [1,2,3]]

tf_data = tf.cast(data,tf.float32)

mean_all = tf.reduce_mean(tf_data, keepdims=False)
mean_0 = tf.reduce_mean(tf_data, axis=0, keepdims=False)
mean_1 = tf.reduce_mean(tf_data, axis=1, keepdims=False)
```

In [9]:

```
mean_all,mean_0,mean_1
```

Out[9]:

```
(<tf.Tensor: id=49, shape=(), dtype=float32, numpy=2.0>,
 <tf.Tensor: id=51, shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dty
pe=float32)>,
 <tf.Tensor: id=53, shape=(2,), dtype=float32, numpy=array([2., 2.], dtype=f
loat32)>)
```

直接取機率最大數值得索引

In [30]:

```
a = tf.constant([1,2,3])
tf.math.argmax(a)
```

Out[30]:

```
<tf.Tensor: shape=(), dtype=int64, numpy=2>
```