TensorFlow 2.0

Tensorflow 2.x

吳佳諺 老師

1.TensorFlow2.x模型

2.設定TensorFlow版本

3.TensorFlow 2直接運算執行

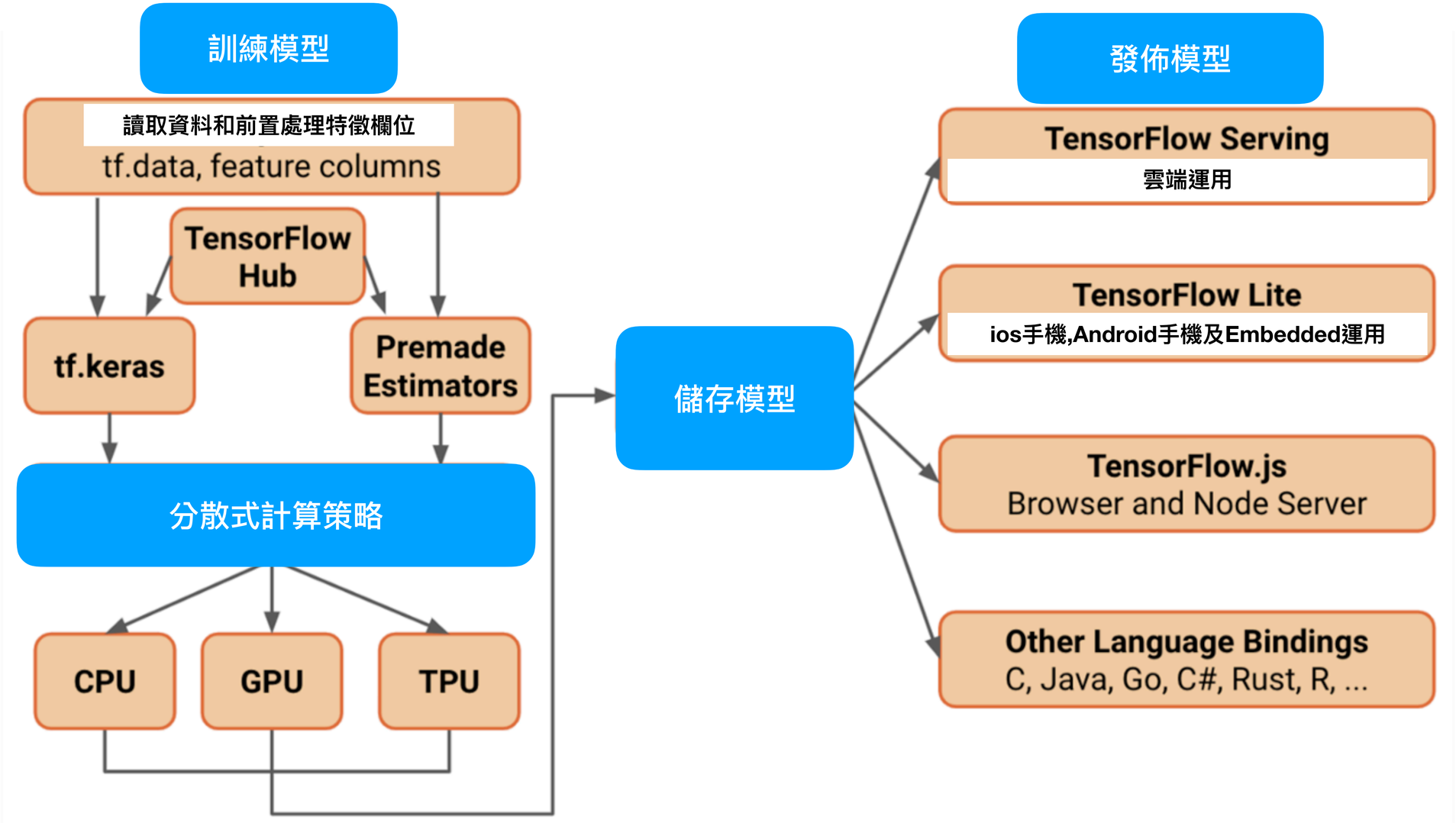
4.Keras 應用程式

5.tf.data

6.TensorFlow2.x實作手寫辨識mnist

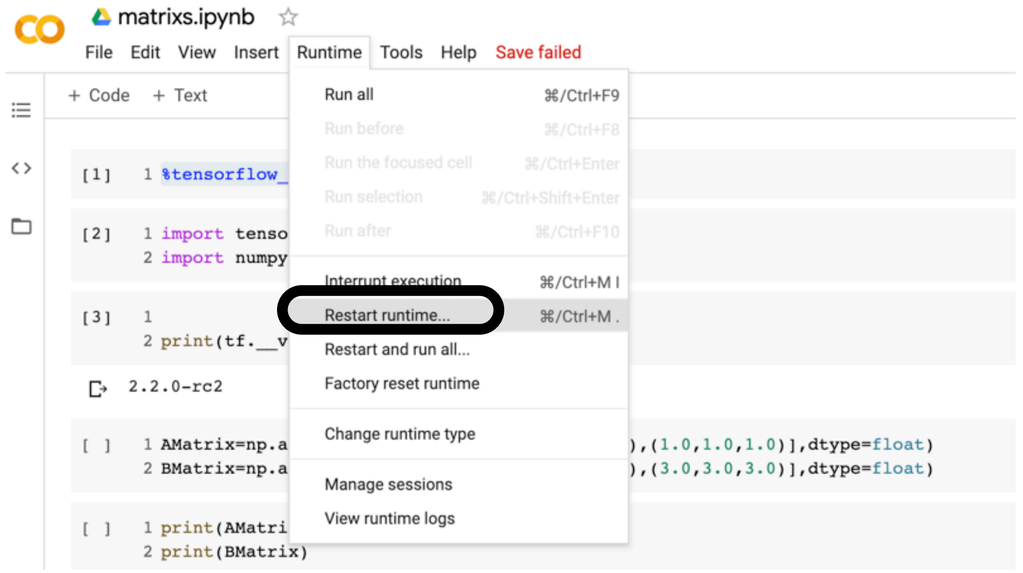
TensorFlow 2.0

1.TensorFlow2.x模型



2.設定TensorFlow版本

- 設定TensorFlow 2.x版
- %tensorflow_version 2.x
- 設定完後重新啟動Runtime
- 設定TensorFlow 1.x版
- %tensorflow_version 1.x



進入Google Colab

3.TensorFlow 2直接運算執行

```
[4]  1 %tensorflow_version 2.x
```

```
[5]  1 import tensorflow as tf
     2 import numpy as np
```

```
[6]  1
     2 print(tf.__version__)
```

```
2.2.0-rc2
```

```
[7]  1 AMatrix=np.array([(1.0,1.0,1.0),(1.0,1.0,1.0),(1.0,1.0,1.0)],dtype=float)
     2 BMatrix=np.array([(3.0,3.0,3.0),(3.0,3.0,3.0),(3.0,3.0,3.0)],dtype=float)
```

```
[8]  1 print(AMatrix)
     2 print(BMatrix)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
```

```
[11]  1 ATMatrix=tf.constant(AMatrix)
      2 BTMatrix=tf.constant(BMatrix)
```

Eager Excution直接運算執行

```
[12]  1 SumMatrix=tf.add(ATMatrix,BTMatrix)
      2 ProductMatrix=tf.matmul(ATMatrix,BTMatrix)
      3 DetMatrix=tf.linalg.det(AMatrix)
```

tf.add()函數直接運算

```
[15]  1 print(SumMatrix)
      2 print(ProductMatrix)
      3 print(DetMatrix)
```

```
tf.Tensor(
[[4. 4. 4.]
 [4. 4. 4.]
 [4. 4. 4.]], shape=(3, 3), dtype=float64)
tf.Tensor(
[[9. 9. 9.]
 [9. 9. 9.]
 [9. 9. 9.]], shape=(3, 3), dtype=float64)
tf.Tensor(0.0, shape=(), dtype=float64)
```

```
1
```

TensorFlow 1.x

```
[1]  1 %tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
[2]  1 import tensorflow as tf
     2 import numpy as np
```

```
[3]  1 AMatrix=np.array([(1.0,1.0,1.0),(1.0,1.0,1.0),(1.0,1.0,1.0)],dtype=float)
     2 BMatrix=np.array([(3.0,3.0,3.0),(3.0,3.0,3.0),(3.0,3.0,3.0)],dtype=float)
```

```
[4]  1 print(AMatrix)
     2 print(BMatrix)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
```

```
[5]  1 ATMatrix=tf.constant(AMatrix)
     2 BTMatrix=tf.constant(BMatrix)
```

```
[7]  1 SumMatrix=tf.add(ATMatrix,BTMatrix)
     2 ProductMatrix=tf.matmul(ATMatrix,BTMatrix)
     3 DetMatrix=tf.matrix_determinant(AMatrix)
```

使用tf.Session()會議運算

```
[8]  1 print(SumMatrix)
     2 print(ProductMatrix)
```

```
Tensor("Add_1:0", shape=(3, 3), dtype=float64)
Tensor("MatMul_1:0", shape=(3, 3), dtype=float64)
```

```
[9]  1 with tf.Session() as sess:
     2     AResult=sess.run(SumMatrix)
     3     BResult=sess.run(ProductMatrix)
     4     CResult=sess.run(DetMatrix)
```

```
1 print(AResult)
2 print(BResult)
3 print(CResult)
```

```
[[4. 4. 4.]
 [4. 4. 4.]
 [4. 4. 4.]]
[[9. 9. 9.]
 [9. 9. 9.]
 [9. 9. 9.]]
0.0
```

```
[ ]  1
```

4.TensorFlow內建Keras 應用程式

```
[1]  1 %tensorflow_version 2.x
```

```
[3]  1 import tensorflow as tf
     2 print(tf.__version__)
```

```
2.2.0-rc2
```

```
[5]  1 mnist = tf.keras.datasets.mnist
```

TensorFlow載入keras手寫辨識資料集

```
[6]  1 (x_train, y_train), (x_test, y_test) = mnist.load_data()
     2 x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

```
[7]  1 model = tf.keras.models.Sequential([
     2   tf.keras.layers.Flatten(input_shape=(28, 28)),
     3   tf.keras.layers.Dense(128, activation='relu'),
     4   tf.keras.layers.Dropout(0.2),
     5   tf.keras.layers.Dense(10)
     6 ])
```

使用keras的模型Sequential

```
[8]  1 predictions = model(x_train[:1]).numpy()
     2 predictions
```

```
array([[-0.42112458, -0.04447089, -0.07521288,  0.18263417, -0.44413954,
         0.19368899, -0.28518748,  0.36920494, -0.02900008,  0.03707186]],
      dtype=float32)
```

```
[9]  1 tf.nn.softmax(predictions).numpy()
```

```
array([[0.06694654, 0.09756786, 0.09461407, 0.12244393, 0.06542337,
        0.12380503, 0.07669462, 0.14755839, 0.09908905, 0.10585719]],
      dtype=float32)
```

```
[10]  1 loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
[11]  1 loss_fn(y_train[:1], predictions).numpy()
```

```
2.0890472
```

```
[12]  1 model.compile(optimizer='adam',
      2               loss=loss_fn,
      3               metrics=['accuracy'])
```

訓練及計算模型

```
[13]  1 model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2999 - accuracy: 0.9133
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1458 - accuracy: 0.9562
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1115 - accuracy: 0.9663
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0889 - accuracy: 0.9725
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0763 - accuracy: 0.9763
<tensorflow.python.keras.callbacks.History at 0x7fbf9a1fd3c8>
```

```
[14]  1 model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.0786 - accuracy: 0.9765
[0.07861529290676117, 0.9764999747276306]
```

```
[15]  1 probability_model = tf.keras.Sequential([
      2   model,
      3   tf.keras.layers.Softmax()
      4 ])
```

```
1 probability_model(x_test[:5])
```

```
<tf.Tensor: shape=(5, 10), dtype=float32, numpy=
array([[3.1236567e-08, 3.5835706e-09, 4.0262344e-06, 4.0718497e-04,
        2.3541445e-11, 4.1640180e-07, 8.7307028e-12, 9.9958509e-01,
        3.9199185e-07, 2.8446157e-06],
       [1.5319448e-07, 2.2768494e-05, 9.9997258e-01, 2.4768397e-06,
        5.4067729e-14, 2.1956572e-07, 8.3330560e-09, 5.8585285e-14,
        1.7529120e-06, 2.2059876e-11],
       [1.8163345e-07, 9.9900705e-01, 2.1312492e-04, 2.1608008e-05,
        5.1999097e-05, 8.1704347e-06, 3.6259303e-06, 4.9884320e-04,
        1.9350655e-04, 1.9238464e-06],
       [9.9980229e-01, 6.6407737e-09, 6.8252994e-05, 6.0306741e-07,
        4.8460115e-06, 2.0520731e-06, 1.1235129e-04, 5.6959866e-06,
        2.1841893e-06, 1.7217860e-06],
       [1.0532881e-05, 1.1474831e-09, 1.6449114e-06, 1.0182176e-07,
        9.9608457e-01, 8.2699162e-06, 5.7382753e-05, 3.6127260e-04,
        3.8931466e-06, 3.4722777e-03]], dtype=float32)>
```

5.tf.data資料輸入管線

- tf.data資料輸入管線可以讓我們輸入資料
- tf.data.Dataset 資料集

tf.data.Dataset 資料集

```
%tensorflow_version 2.x
```

```
import tensorflow as tf
```

```
import pathlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

np.set_printoptions(precision=4)
```

```
dataset = tf.data.Dataset.from_tensor_slices([8, 6, 4, 3, 2, 1])
dataset
```

```
<TensorSliceDataset shapes: (), types: tf.int32>
```

```
for elem in dataset:
  print(elem.numpy())
```

```
8
6
4
3
2
1
```

tf.data.Dataset資料集儲存資料
4列十行

```
[ ]  1 dataset1 = tf.data.Dataset.from_tensor_slices(
     2     tf.random.uniform([4, 10], minval=1, maxval=10, dtype=tf.int32))
     3
     4 dataset1
```

```
<TensorSliceDataset shapes: (10,), types: tf.int32>
```

```
[ ]  1 for z in dataset1:
     2   print(z.numpy())
```

```
[6 2 2 9 1 4 9 1 8 4]
[4 5 2 6 7 2 4 8 2 8]
[1 9 5 4 5 7 4 2 2 6]
[8 6 9 9 3 4 5 7 4 3]
```

tf.data.Dataset資料集輸入手寫mnist辨識資料

```
[ ]  1 train, test = tf.keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

```
[ ]  1 images, labels = train
     2 images = images/255
     3
     4 dataset = tf.data.Dataset.from_tensor_slices((images, labels))
     5 dataset
```

```
<TensorSliceDataset shapes: ((28, 28), ()), types: (tf.float64, tf.uint8)>
```

6.TensorFlow2.x實作手寫辨識mnist

[3]
```
%tensorflow_version 2.x
```

[4]
```
import tensorflow as tf
```

[5]
```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

[17]
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.25),
  tf.keras.layers.Dense(10)
])
```

[18]
```
predictions = model(x_train[:1]).numpy()
predictions
```

```
array([[-0.30361855,  0.5115421 , -0.2581011 ,  0.07883194,  0.7422744 ,
        -0.27932325,  0.07319404, -0.23777807,  0.29573333, -0.20737615]],
      dtype=float32)
```

```
[19]  1 tf.nn.softmax(predictions).numpy()
```

```
array([[0.06626833, 0.14973585, 0.0693544 , 0.09714091, 0.18859561,
        0.06789806, 0.09659478, 0.07077831, 0.12067054, 0.07296315]],
      dtype=float32)
```

```
[20]  1 loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
[21]  1 loss_fn(y_train[:1], predictions).numpy()
```

```
2.6897478
```

```
[22]  1 model.compile(optimizer='adam',
      2               loss=loss_fn,
      3               metrics=['accuracy'])
```

```
      1 model.fit(x_train, y_train, epochs=8)
```

```
Epoch 1/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3052 - accuracy: 0.9109
Epoch 2/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1505 - accuracy: 0.9554
Epoch 3/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1142 - accuracy: 0.9654
Epoch 4/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0978 - accuracy: 0.9698
Epoch 5/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0840 - accuracy: 0.9735
Epoch 6/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0763 - accuracy: 0.9761
Epoch 7/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0678 - accuracy: 0.9783
Epoch 8/8
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0631 - accuracy: 0.9800
<tensorflow.python.keras.callbacks.History at 0x7f1aaf168748>
```

```
[13]  1 model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.0736 - accuracy: 0.9779
[0.07360788434743881, 0.9779000282287598]
```

[14]
```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

[16]
```
probability_model(x_test[:8])
```

```
<tf.Tensor: shape=(8, 10), dtype=float32, numpy=
array([[2.7586397e-08, 1.8783714e-08, 9.7347147e-06, 1.3144073e-04,
        6.4785455e-11, 2.6629552e-08, 1.9347763e-12, 9.9985683e-01,
        1.3079305e-07, 1.7853616e-06],
       [3.7765016e-08, 6.5040957e-05, 9.9993324e-01, 1.4967889e-06,
        3.5220236e-14, 3.4037381e-08, 1.9735202e-08, 3.0112035e-13,
        2.2606280e-07, 2.2349128e-13],
       [1.4739068e-06, 9.9758935e-01, 3.5869997e-04, 8.0492238e-05,
        1.3462723e-04, 1.2796521e-05, 8.3531340e-06, 1.3124155e-03,
        4.9989112e-04, 1.8536754e-06],
       [9.9976212e-01, 1.6876067e-09, 2.3286277e-04, 3.5118799e-08,
        1.7211069e-07, 1.3440810e-07, 8.3221259e-08, 3.9692782e-06,
        3.5670735e-09, 5.5049799e-07],
       [6.9346484e-06, 1.5472739e-09, 3.8421236e-05, 8.1156131e-09,
        9.9865282e-01, 5.4478193e-08, 1.6600070e-06, 1.5049252e-04,
        8.5499605e-06, 1.1409743e-03],
       [4.7679762e-08, 9.9839729e-01, 2.1065857e-06, 9.9681838e-06,
        2.3298140e-05, 7.2745955e-08, 3.8766036e-08, 1.5480750e-03,
        1.8464969e-05, 5.8619071e-07],
       [1.0077341e-09, 8.7764667e-09, 1.1059855e-06, 1.1311549e-07,
        9.9963701e-01, 1.4383628e-06, 9.9256914e-09, 1.1392822e-05,
        3.1240963e-04, 3.6471116e-05],
       [2.1889717e-07, 9.0409992e-07, 9.0389563e-05, 1.0508212e-03,
        4.6337708e-03, 4.5318975e-05, 1.4231214e-09, 1.3280280e-03,
        1.7770013e-05, 9.9283278e-01]], dtype=float32)>
```

- Thanks