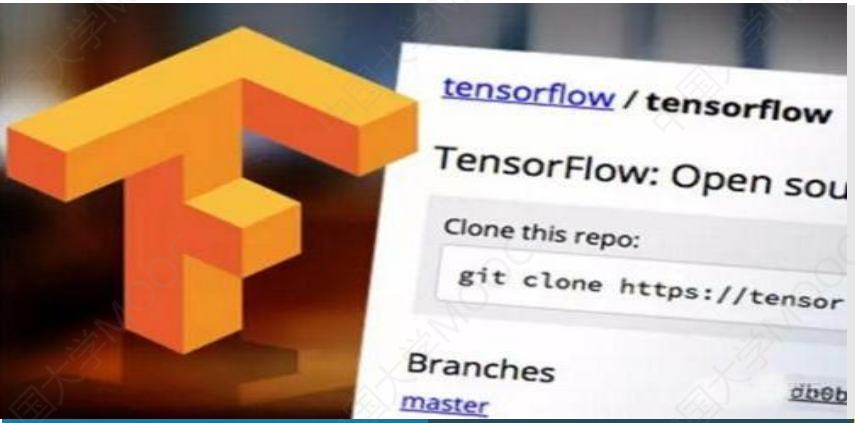

14.6 实例：卷积神经网络识别cifar10图片

导入库

In [1]:
```
import tensorflow as tf
tf.__version__, tf.keras.__version__
```

Out[1]:
```
('2.0.0', '2.2.4-tf')
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers,Sequential
```

In [3]:
```
gpus = tf.config.experimental.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

加载数据集

In [4]:
```
cifar10=tf.keras.datasets.cifar10
(x_train,y_train),(x_test,y_test)=cifar10.load_data()
```

In [5]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(50000, 32, 32, 3)
(50000, 1)
(10000, 32, 32, 3)
(10000, 1)
```

数据预处理

```
In [6]: x_train, x_test = tf.cast(x_train, dtype=tf.float32)/255.0, tf.cast(x_test, dtype=tf.float32)/255.0
        y_train, y_test = tf.cast(y_train, dtype=tf.int32), tf.cast(y_test, dtype=tf.int32)

In [7]: x_train.shape[1:]

Out[7]: TensorShape([32, 32, 3])
```

建立模型

In [8]:
```
model=Sequential([

    # unit 1
    layers.Conv2D(16, kernel_size=(3, 3), padding="same", activation=tf.nn.relu, input_shape=x_train.shape[1:]),
    layers.Conv2D(16, kernel_size=(3, 3), padding="same", activation=tf.nn.relu),
    layers.MaxPool2D(pool_size=(2, 2)),

    # unit 2
    layers.Conv2D(32, kernel_size=(3, 3), padding="same", activation=tf.nn.relu),
    layers.Conv2D(32, kernel_size=(3, 3), padding="same", activation=tf.nn.relu),
    layers.MaxPool2D(pool_size=(2, 2)),

    # unit 3
    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(10, activation="softmax")
])
```

查看摘要

```
In [9]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 16)        2320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 32)        4640
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 280,218
Trainable params: 280,218
Non-trainable params: 0
```

配置训练方法

```
In [10]: model.compile(optimizer='adam',
                       loss='sparse_categorical_crossentropy',
                       metrics=['sparse_categorical_accuracy']
                      )
```

训练模型

```
model.fit(x_train,y_train, batch_size=64, epochs=5, validation_split=0.2)
```

```
Train on 40000 samples, validate on 10000 samples
Epoch 1/5
40000/40000 [==============================] - 6s 160us/sample - loss: 1.5263 - sparse_categorical_accuracy: 0.4437 - val_loss: 1.2471 - val
_sparse_categorical_accuracy: 0.5527
Epoch 2/5
40000/40000 [==============================] - 4s 91us/sample - loss: 1.1293 - sparse_categorical_accuracy: 0.5975 - val_loss: 1.0627 - val_
sparse_categorical_accuracy: 0.6335
Epoch 3/5
40000/40000 [==============================] - 4s 88us/sample - loss: 0.9790 - sparse_categorical_accuracy: 0.6551 - val_loss: 0.9554 - val_
sparse_categorical_accuracy: 0.6653
Epoch 4/5
40000/40000 [==============================] - 3s 84us/sample - loss: 0.8495 - sparse_categorical_accuracy: 0.7032 - val_loss: 0.8870 - val_
sparse_categorical_accuracy: 0.6911
Epoch 5/5
40000/40000 [==============================] - 4s 92us/sample - loss: 0.7581 - sparse_categorical_accuracy: 0.7359 - val_loss: 0.8587 - val_
sparse_categorical_accuracy: 0.7059

<tensorflow.python.keras.callbacks.History at 0x17e9078fc18>
```

评估模型

```
In [12]: model.evaluate(x_test,y_test, verbose=2)
         10000/1 - 1s - loss: 0.8898 - sparse_categorical_accuracy: 0.7031
Out[12]: [0.8679245807647705, 0.7031]
```