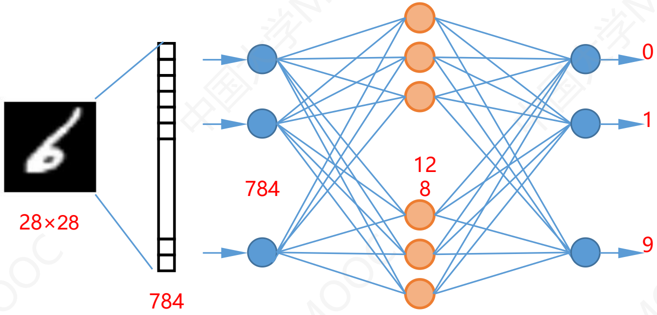
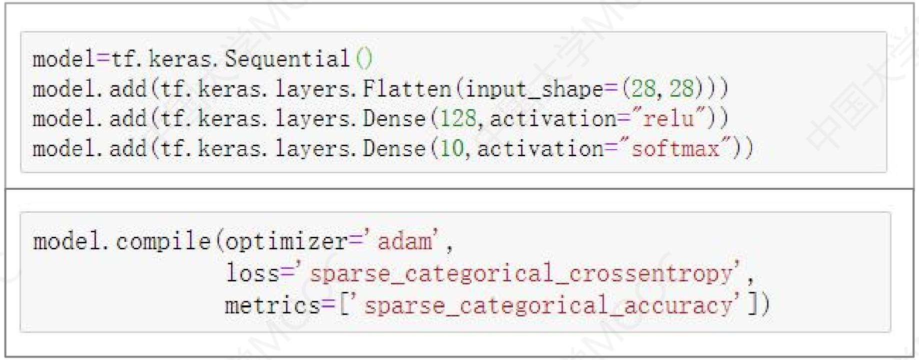
13.6 实例：模型的保存和加载

中国大学MOOC

实例：使用Sequential模型实现手写数字识别

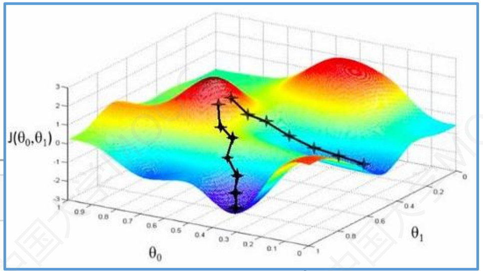


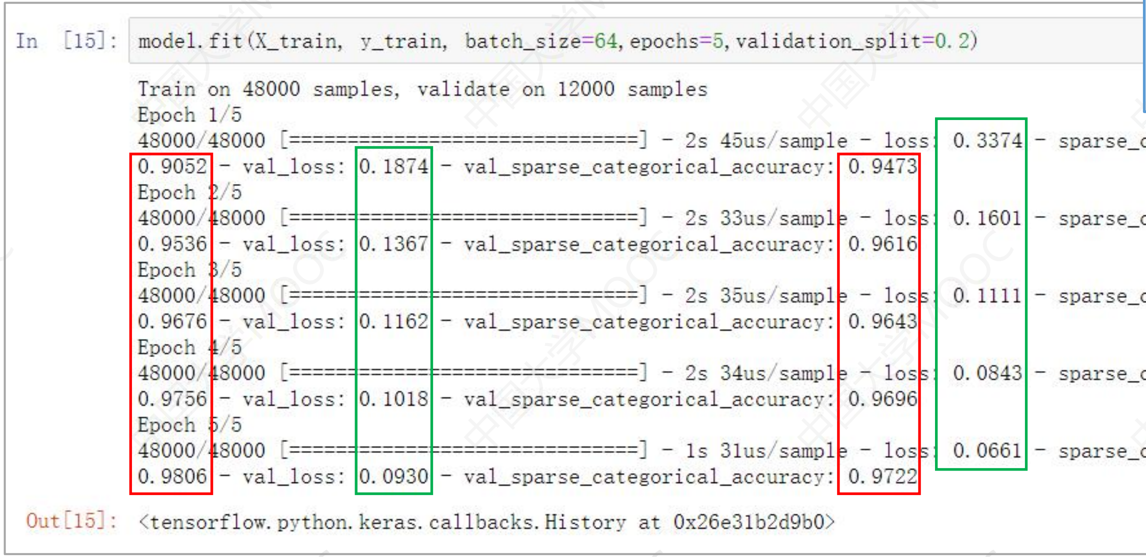


```
model=tf.keras.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(128,activation="relu"))
model.add(tf.keras.layers.Dense(10,activation="softmax"))
```

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

训练模型





```
In  [15]:  model.fit(X_train, y_train, batch_size=64, epochs=5, validation_split=0.2)

           Train on 48000 samples, validate on 12000 samples
           Epoch 1/5
           48000/48000 [==============================] - 2s 45us/sample - loss: 0.3374 - sparse_categorical_accuracy:
           0.9052 - val_loss: 0.1874 - val_sparse_categorical_accuracy: 0.9473
           Epoch 2/5
           48000/48000 [==============================] - 2s 33us/sample - loss: 0.1601 - sparse_categorical_accuracy:
           0.9536 - val_loss: 0.1367 - val_sparse_categorical_accuracy: 0.9616
           Epoch 3/5
           48000/48000 [==============================] - 2s 35us/sample - loss: 0.1111 - sparse_categorical_accuracy:
           0.9676 - val_loss: 0.1162 - val_sparse_categorical_accuracy: 0.9643
           Epoch 4/5
           48000/48000 [==============================] - 2s 34us/sample - loss: 0.0843 - sparse_categorical_accuracy:
           0.9756 - val_loss: 0.1018 - val_sparse_categorical_accuracy: 0.9696
           Epoch 5/5
           48000/48000 [==============================] - 1s 31us/sample - loss: 0.0661 - sparse_categorical_accuracy:
           0.9806 - val_loss: 0.0930 - val_sparse_categorical_accuracy: 0.9722

Out[15]:   <tensorflow.python.keras.callbacks.History at 0x26e31b2d9b0>
```

13.6 模型的保存与加载

保存模型参数

```
model.save_weights( filepath,
                    overwrite=True,
                    save_format=None)
```

HDF5格式

.h5 ,.keras

save_format=None

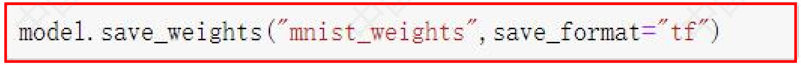
分层数据格式（Hierarchical Data Format)

- group
- dataset

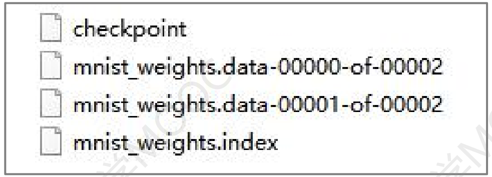
SavedModel

save_format= "tf"

TensorFlow序列化文件格式



```
model.save_weights("mnist_weights",save_format="tf")
```

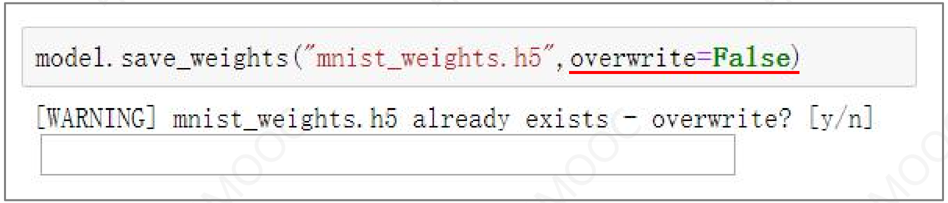


```
checkpoint
mnist_weights.data-00000-of-00002
mnist_weights.data-00001-of-00002
mnist_weights.index
```

13.6 模型的保存与加载

保存模型参数

```
model.save_weights( filepath,
                    overwrite=True,
                    save_format=None)
```



```
model.save_weights("mnist_weights.h5",overwrite=False)
```

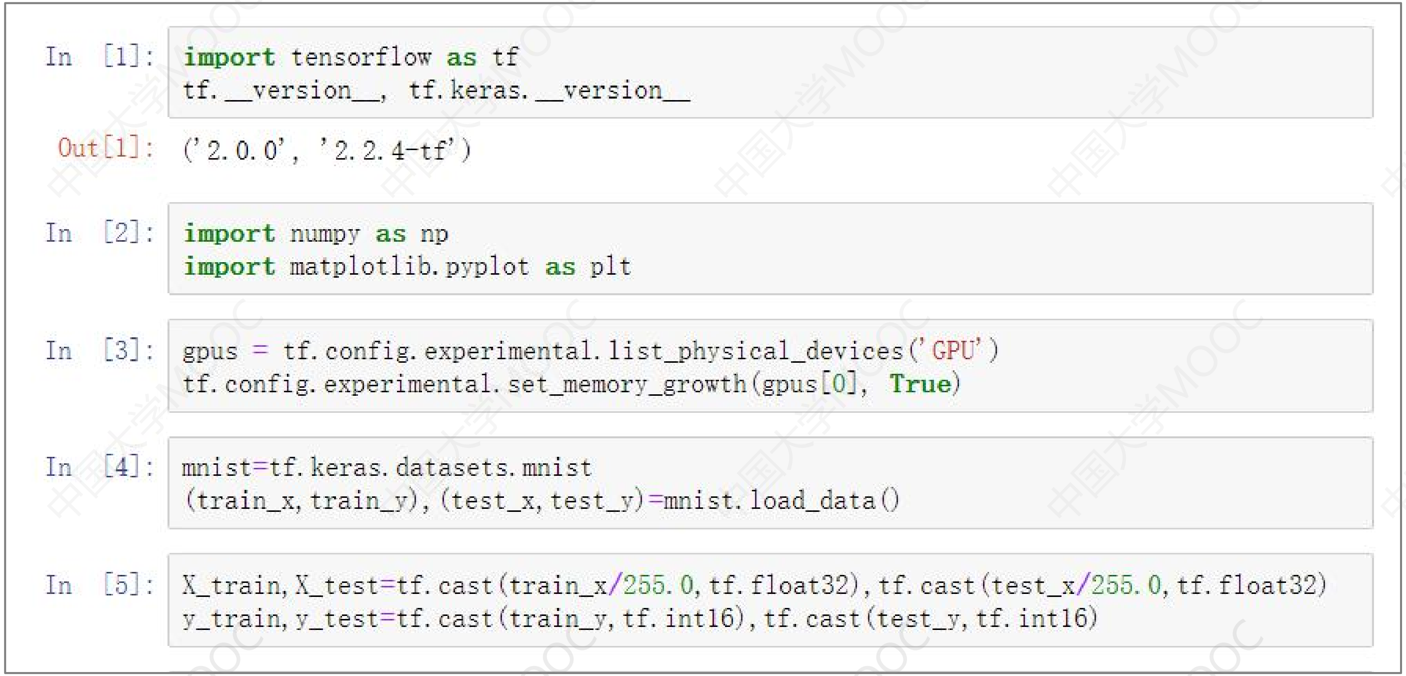
```
[WARNING] mnist_weights.h5 already exists - overwrite? [y/n]
```

加载模型参数

```
model.load_weights( filepath)
```

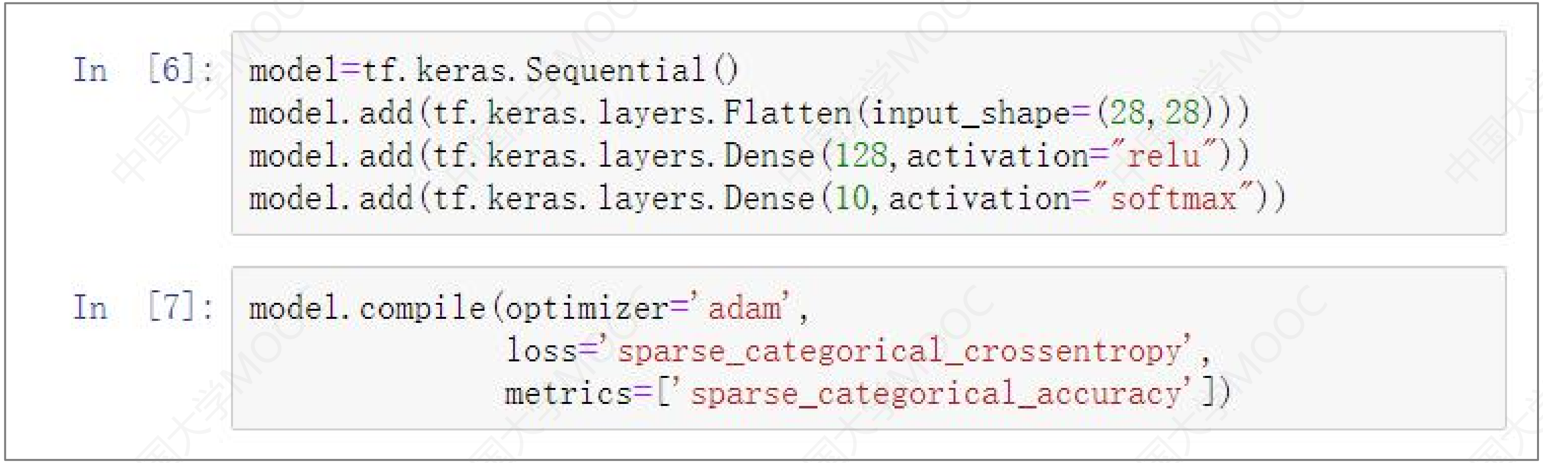
13.6 模型的保存与加载

实例： 手写数字识别



```
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

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

In [4]: mnist=tf.keras.datasets.mnist
        (train_x,train_y),(test_x,test_y)=mnist.load_data()

In [5]: X_train,X_test=tf.cast(train_x/255.0,tf.float32),tf.cast(test_x/255.0,tf.float32)
        y_train,y_test=tf.cast(train_y,tf.int16),tf.cast(test_y,tf.int16)
```

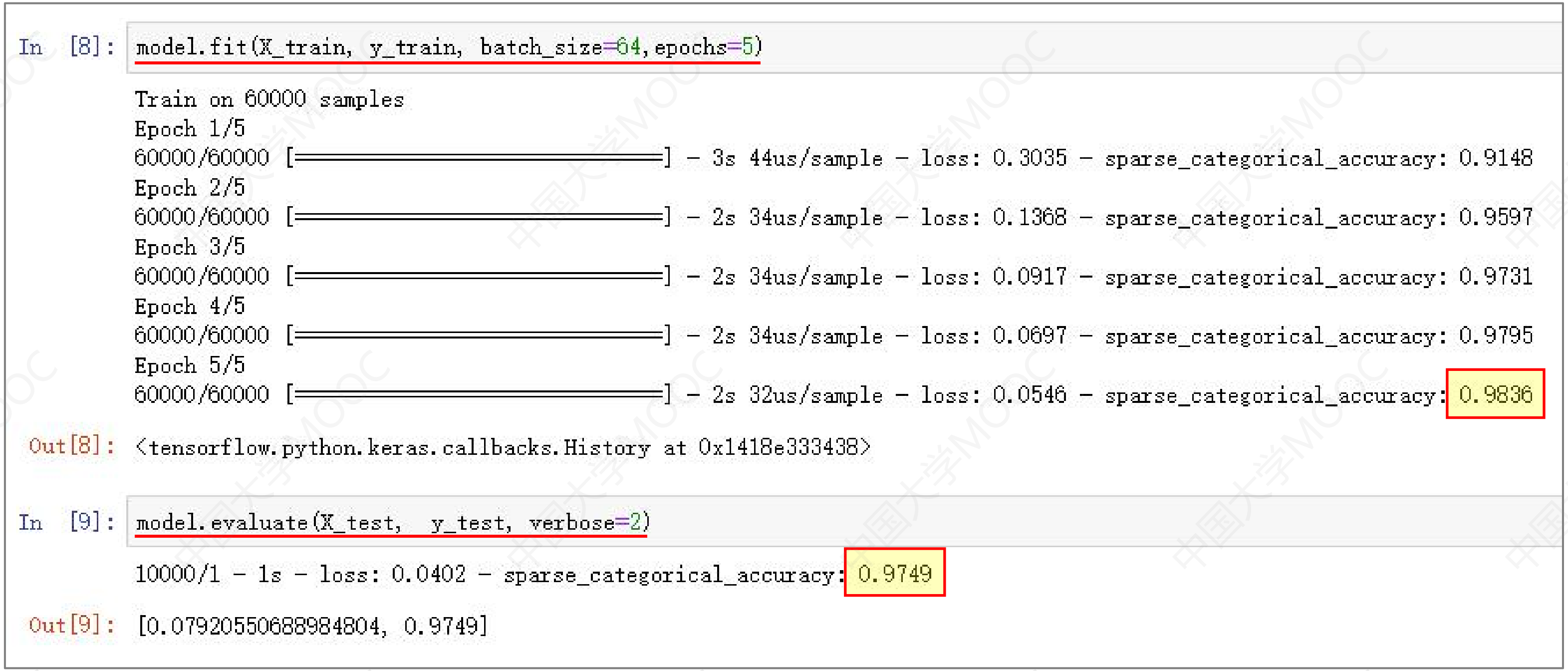
13.6 模型的保存与加载



```
In  [6]: model=tf.keras.Sequential()
         model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
         model.add(tf.keras.layers.Dense(128,activation="relu"))
         model.add(tf.keras.layers.Dense(10,activation="softmax"))
```

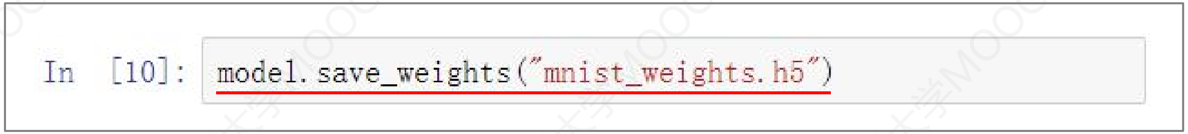
```
In  [7]: model.compile(optimizer='adam',
                       loss='sparse_categorical_crossentropy',
                       metrics=['sparse_categorical_accuracy'])
```

13.6 模型的保存与加载

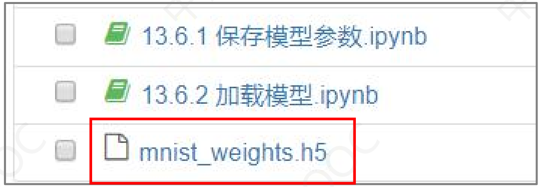


```
In  [8]: model.fit(X_train, y_train, batch_size=64,epochs=5)

         Train on 60000 samples
         Epoch 1/5
         60000/60000 [==============================] - 3s 44us/sample - loss: 0.3035 - sparse_categorical_accuracy: 0.9148
         Epoch 2/5
         60000/60000 [==============================] - 2s 34us/sample - loss: 0.1368 - sparse_categorical_accuracy: 0.9597
         Epoch 3/5
         60000/60000 [==============================] - 2s 34us/sample - loss: 0.0917 - sparse_categorical_accuracy: 0.9731
         Epoch 4/5
         60000/60000 [==============================] - 2s 34us/sample - loss: 0.0697 - sparse_categorical_accuracy: 0.9795
         Epoch 5/5
         60000/60000 [==============================] - 2s 32us/sample - loss: 0.0546 - sparse_categorical_accuracy: 0.9836

Out[8]:  <tensorflow.python.keras.callbacks.History at 0x1418e333438>

In  [9]: model.evaluate(X_test,  y_test, verbose=2)

         10000/1 - 1s - loss: 0.0402 - sparse_categorical_accuracy: 0.9749

Out[9]:  [0.07920550688984804, 0.9749]
```

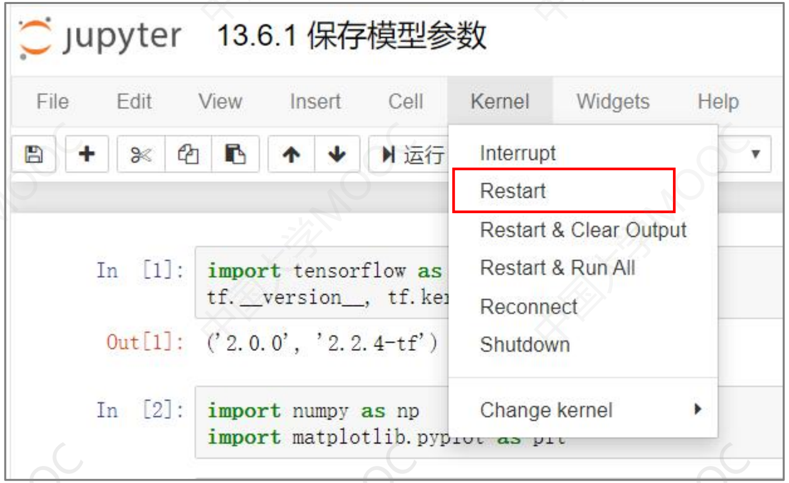
13.6 模型的保存与加载



```
In  [10]: model.save_weights("mnist_weights.h5")
```

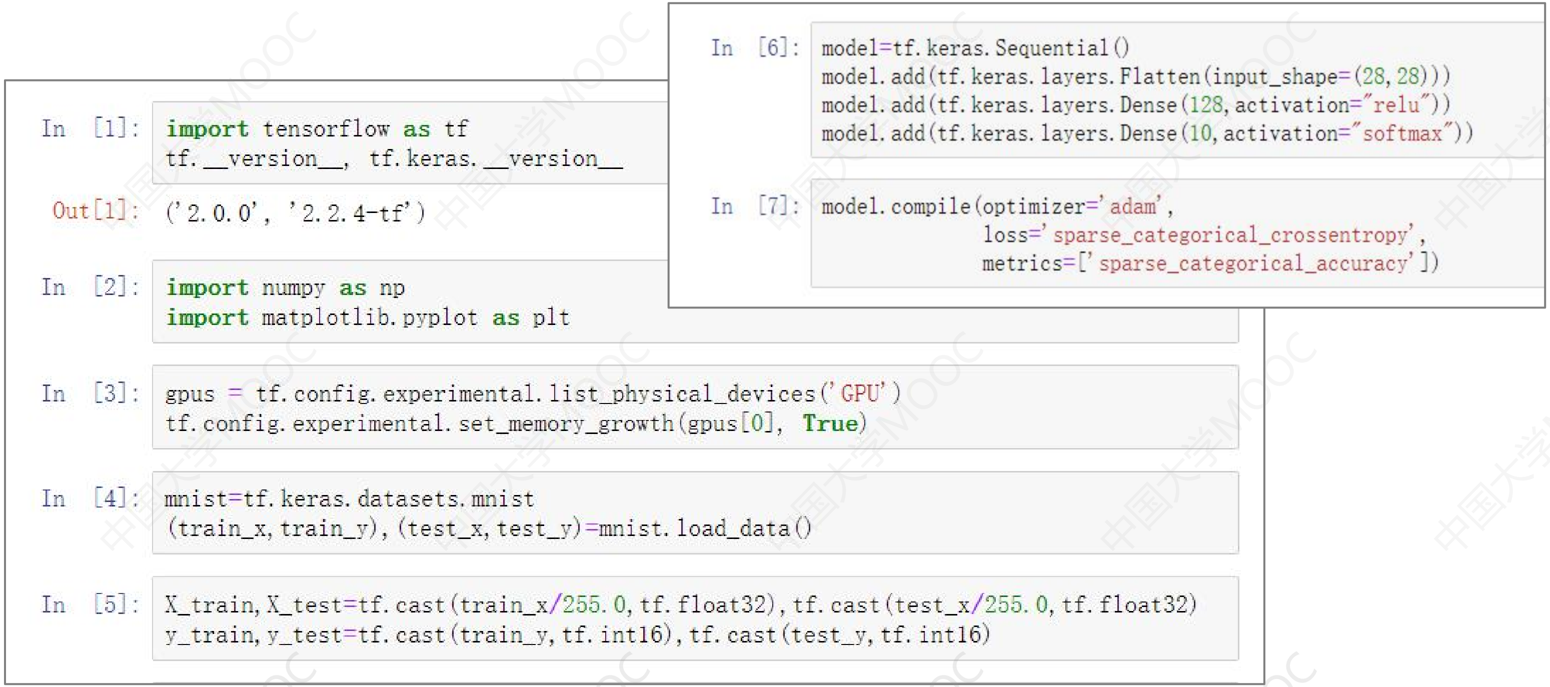


- □ 13.6.1 保存模型参数.ipynb
- □ 13.6.2 加载模型.ipynb
- □ mnist_weights.h5



```
In  [1]: import tensorflow as
         tf.__version__, tf.ke
Out[1]: ('2.0.0', '2.2.4-tf')

In  [2]: import numpy as np
         import matplotlib.pyplot as plt
```

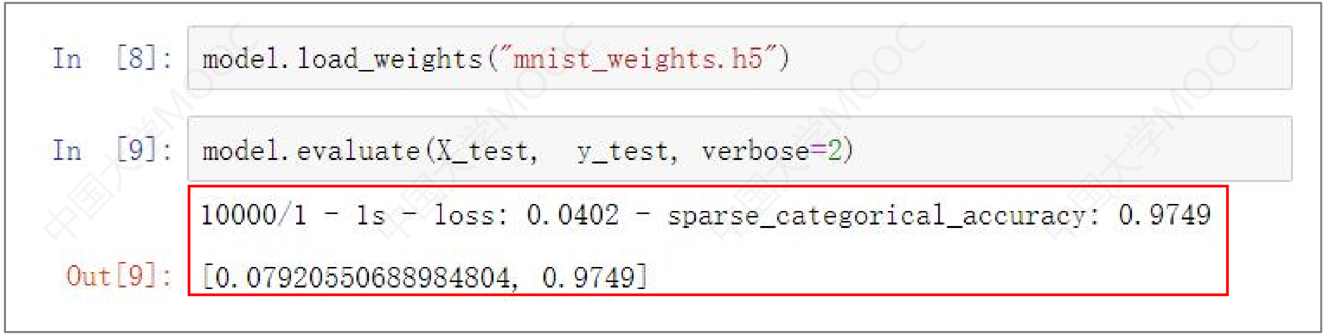
13.6 模型的保存与加载



```
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

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

In [4]: mnist=tf.keras.datasets.mnist
        (train_x,train_y),(test_x,test_y)=mnist.load_data()

In [5]: X_train,X_test=tf.cast(train_x/255.0,tf.float32),tf.cast(test_x/255.0,tf.float32)
        y_train,y_test=tf.cast(train_y,tf.int16),tf.cast(test_y,tf.int16)
```

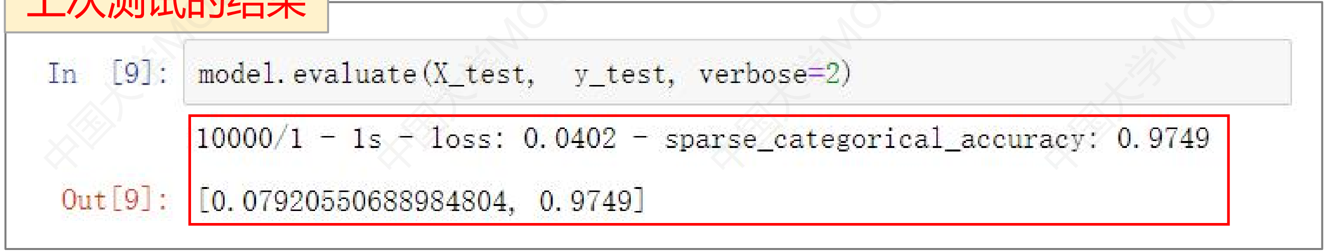
```
In [6]: model=tf.keras.Sequential()
        model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
        model.add(tf.keras.layers.Dense(128,activation="relu"))
        model.add(tf.keras.layers.Dense(10,activation="softmax"))

In [7]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['sparse_categorical_accuracy'])
```

13.6 模型的保存与加载



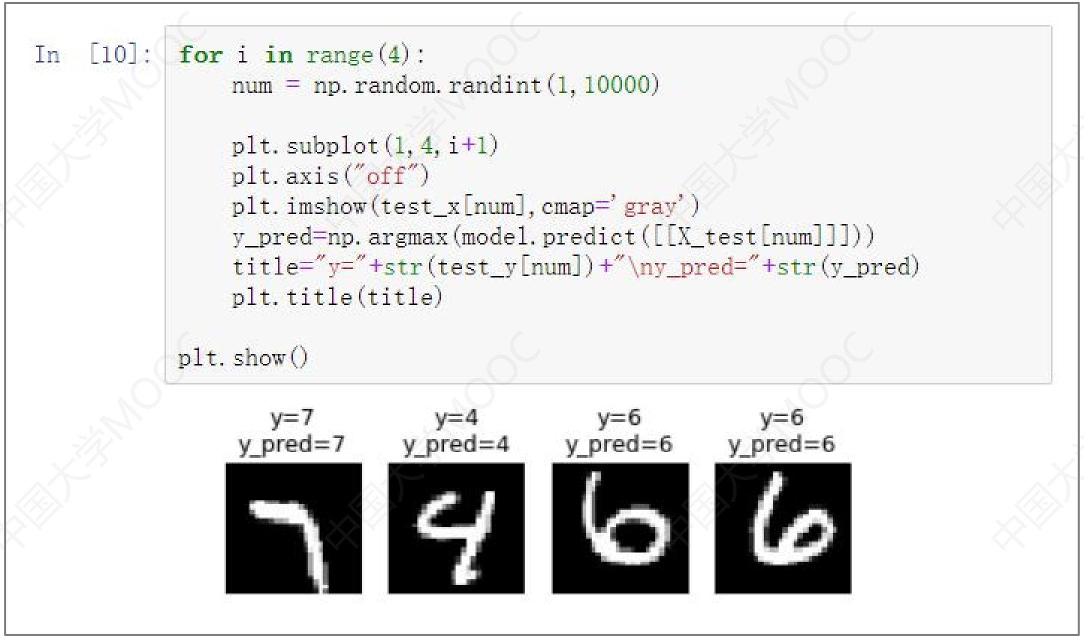
```
In  [8]: model.load_weights("mnist_weights.h5")

In  [9]: model.evaluate(X_test, y_test, verbose=2)

         10000/1 - 1s - loss: 0.0402 - sparse_categorical_accuracy: 0.9749

Out[9]:  [0.07920550688984804, 0.9749]
```

上次测试的结果



```
In  [9]: model.evaluate(X_test, y_test, verbose=2)

         10000/1 - 1s - loss: 0.0402 - sparse_categorical_accuracy: 0.9749

Out[9]:  [0.07920550688984804, 0.9749]
```

13.6 模型的保存与加载



In [10]:

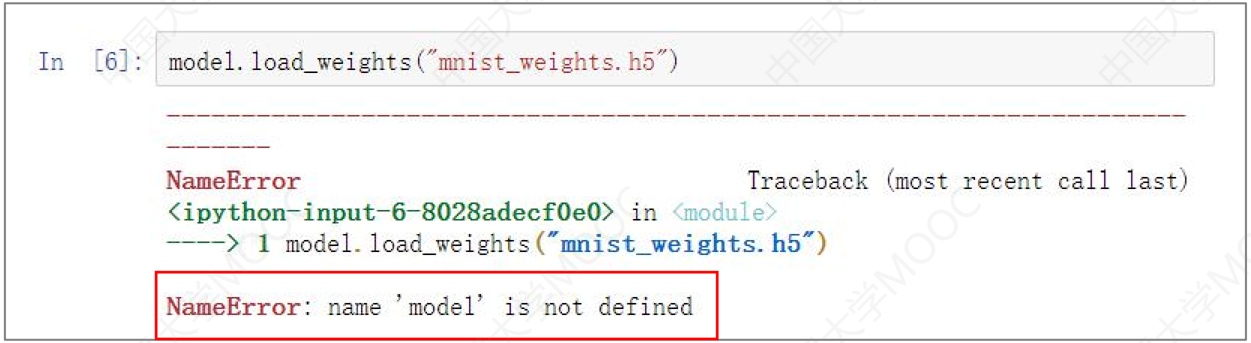
```
for i in range(4):
    num = np.random.randint(1,10000)

    plt.subplot(1,4,i+1)
    plt.axis("off")
    plt.imshow(test_x[num],cmap='gray')
    y_pred=np.argmax(model.predict([[X_test[num]]]))
    title="y="+str(test_y[num])+"\ny_pred="+str(y_pred)
    plt.title(title)

plt.show()
```

```
y=7        y=4        y=6        y=6
y_pred=7   y_pred=4   y_pred=6   y_pred=6
```

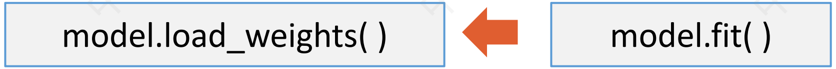
13.6 模型的保存与加载

save_weights()方法仅保存了神经网络的**模型参数**，

使用load_weights方法之前，需要首先定义一个完全相同的神经网络模型



```
In  [6]: model.load_weights("mnist_weights.h5")

---------------------------------------------------------------------------
--------
NameError                                 Traceback (most recent call last)
<ipython-input-6-8028adecf0e0> in <module>
----> 1 model.load_weights("mnist_weights.h5")

NameError: name 'model' is not defined
```



13.6 模型的保存与加载

保存整个模型

```
model.save ( filepath,
             overwrite=True,
             include_optimizer=True,
             save_format=None
           )
```

- □ 神经网络的结构
- □ 模型参数
- □ 配置信息（优化器，损失函数等）
- □ 优化器状态

加载模型

```
tf.keras.models.load_model()
```

□ HDF5格式

```
**.h5 ,**.keras
save_format=None
```

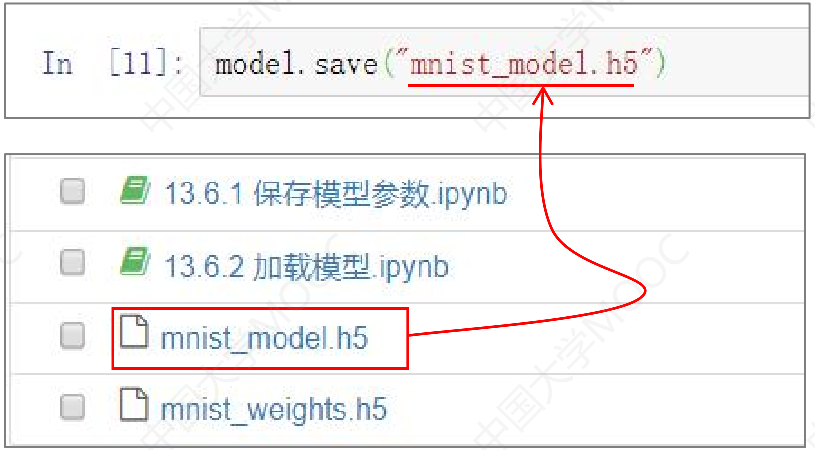
□ SavedModel

```
save_format= "tf"
```

```
----mnist_model_tf
    |-----assets
    |-----variables
          |------variables.data-00000-of-00002
          |------variables.data-00001-of-00002
          |------variables.index
    |-----saved_model.pb
```

13.6 模型的保存与加载

实例：保存**手写数字识别**模型



```
In  [11]:  model.save("mnist_model.h5")
```

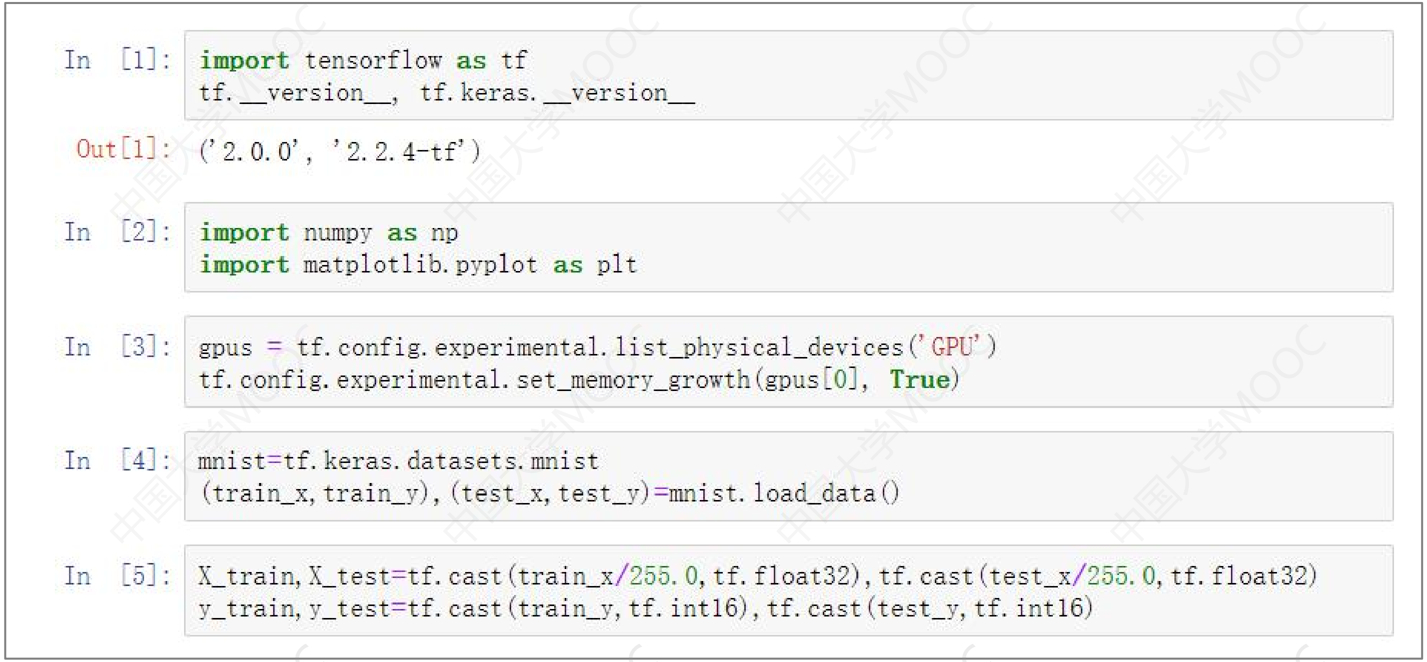
□ 13.6.1 保存模型参数.ipynb

□ 13.6.2 加载模型.ipynb

□ mnist_model.h5

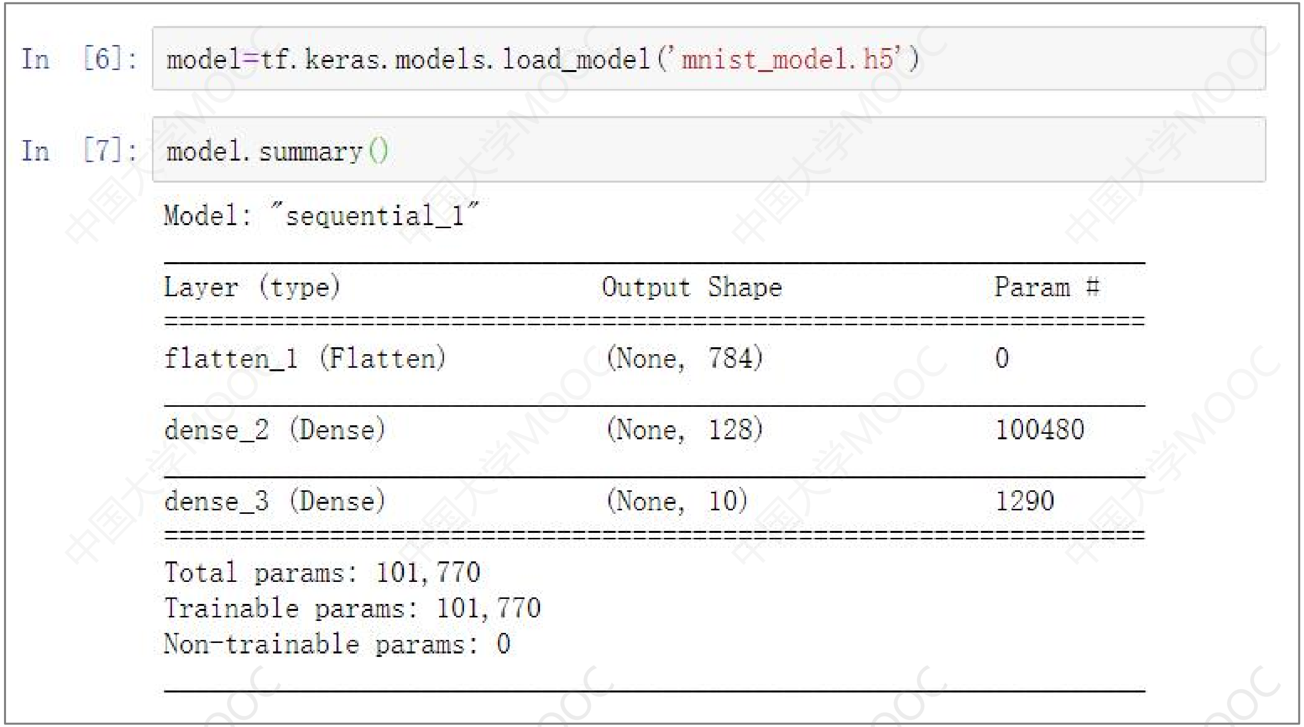
□ mnist_weights.h5

13.6 模型的保存与加载



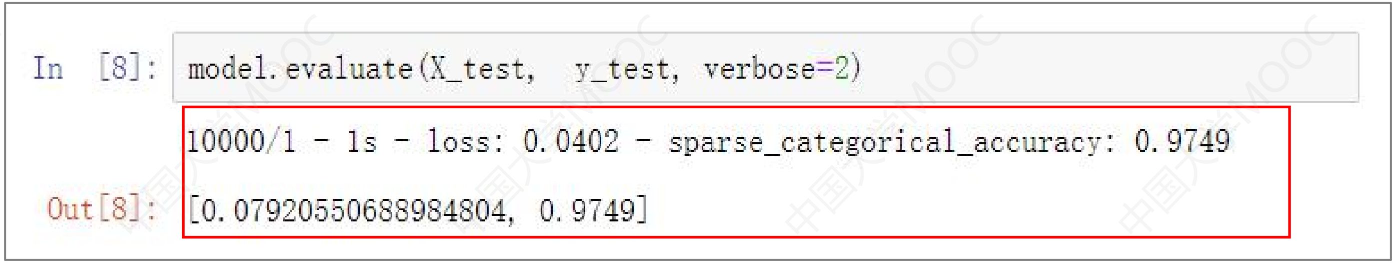
```
In [1]: import tensorflow as tf
        tf.__version__, tf.keras.__version__

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

In [2]: import numpy as np
        import matplotlib.pyplot as plt

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

In [4]: mnist=tf.keras.datasets.mnist
        (train_x,train_y),(test_x,test_y)=mnist.load_data()

In [5]: X_train,X_test=tf.cast(train_x/255.0,tf.float32),tf.cast(test_x/255.0,tf.float32)
        y_train,y_test=tf.cast(train_y,tf.int16),tf.cast(test_y,tf.int16)
```

13.6 模型的保存与加载



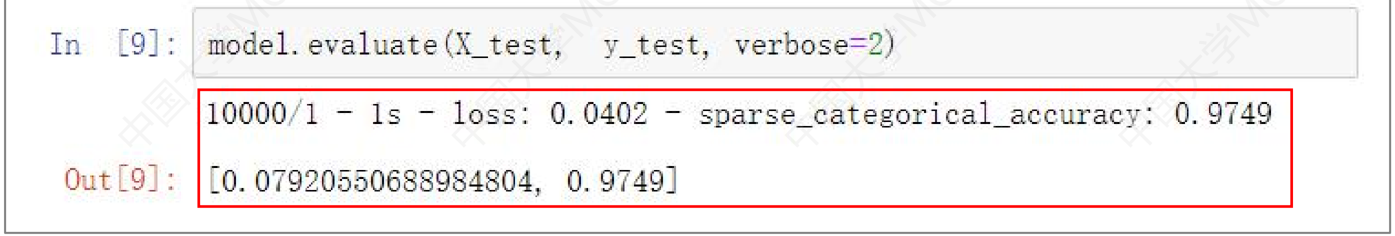
```
In  [6]: model=tf.keras.models.load_model('mnist_model.h5')

In  [7]: model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               100480
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

13.6 模型的保存与加载



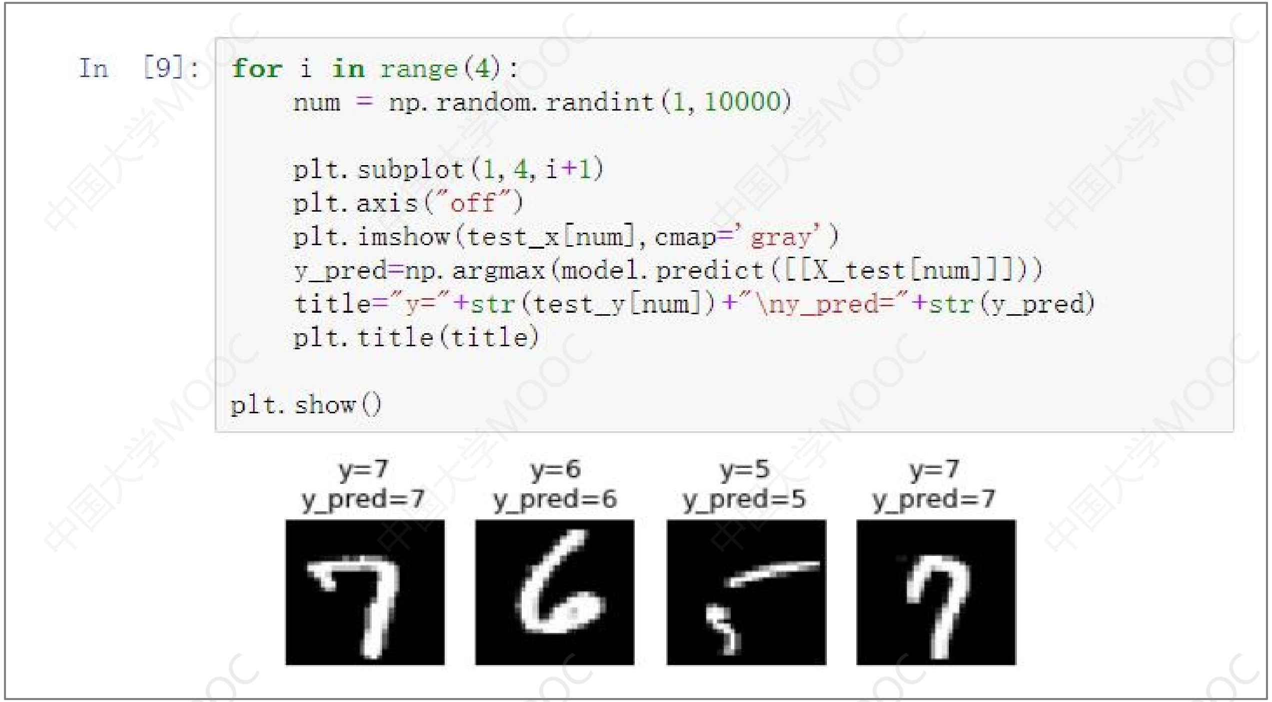
```
In  [8]: model.evaluate(X_test,  y_test, verbose=2)

10000/1 - 1s - loss: 0.0402 - sparse_categorical_accuracy: 0.9749

Out[8]: [0.07920550688984804, 0.9749]
```

上次测试的结果



```
In  [9]: model.evaluate(X_test,  y_test, verbose=2)

10000/1 - 1s - loss: 0.0402 - sparse_categorical_accuracy: 0.9749

Out[9]: [0.07920550688984804, 0.9749]
```

13.6 模型的保存与加载



In [9]:

```
for i in range(4):
    num = np.random.randint(1,10000)

    plt.subplot(1,4,i+1)
    plt.axis("off")
    plt.imshow(test_x[num],cmap='gray')
    y_pred=np.argmax(model.predict([[X_test[num]]]))
    title="y="+str(test_y[num])+"\ny_pred="+str(y_pred)
    plt.title(title)

plt.show()
```

```
   y=7           y=6           y=5           y=7
y_pred=7      y_pred=6      y_pred=5      y_pred=7
```