13.5 实例：Sequential模型实现手写数字识别

中国大学MOOC

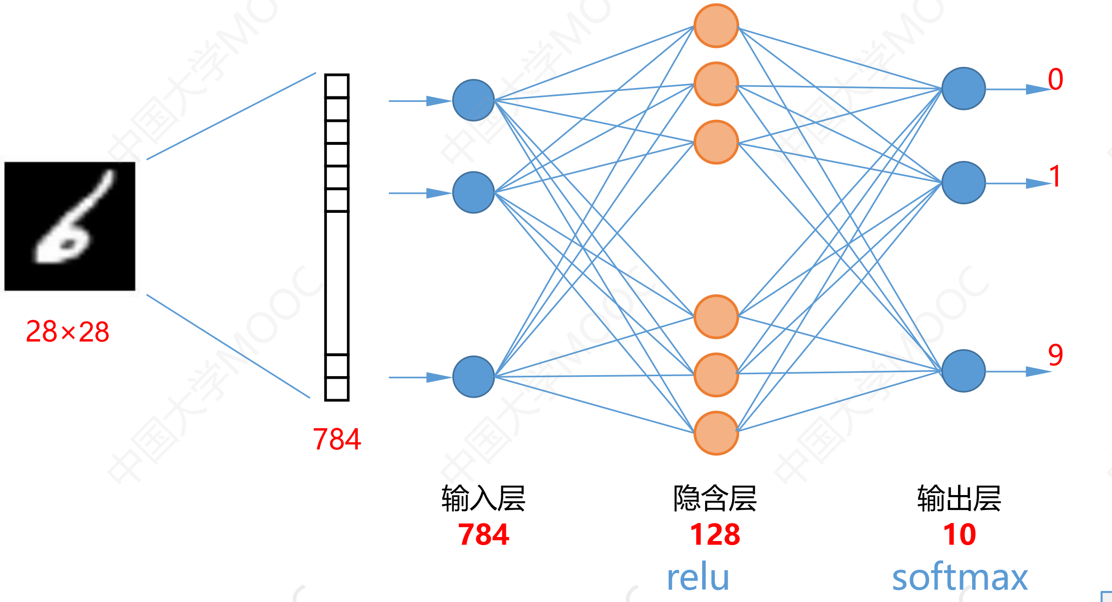
Sequential模型

- 只有**一组输入**和**一组输出**
- 各个**层**按照先后顺序**堆叠**

建立模型	model=tf.keras.Sequential( ) model.add( )
查看摘要	model.summary( )
配置训练方法	model.compile( )
训练模型	model.fit( )
评估模型	model.evaluate( )
使用模型	model.predict( )

□ 手写数字数据集MNIST

设计神经网络结构



784×128+128=10048   128×10+10=1290

100480+1290=101770

标签值：0~9

损失函数：

SparseCategoricalCrossentropy

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
```

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

加载数据

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

In [5]: print(train_x.shape)
        print(train_y.shape)
        print(test_x.shape)
        print(test_y.shape)

(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)

In [6]: type(train_x),type(train_y)
Out[6]: (numpy.ndarray, numpy.ndarray)

In [7]: type(test_x),type(test_y)
Out[7]: (numpy.ndarray, numpy.ndarray)
```

```
In [8]: train_x.min(),train_x.max()
Out[8]: (0, 255)

In [9]: train_y.min(),train_y.max()
Out[9]: (0, 9)
```

数据预处理

```
In [10]: X_train=train_x.reshape((60000,28*28))
         X_test=test_x.reshape((10000,28*28))
```

→ tf.keras.layers.Flatten( )

```
In [11]: print(X_train.shape)
         print(X_test.shape)

(60000, 784)
(10000, 784)
```

```
In [10]: 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 [11]: type(X_train),type(y_train)

Out[11]: (tensorflow.python.framework.ops.EagerTensor,
          tensorflow.python.framework.ops.EagerTensor)
```

建立模型

```
In  [12]: 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  [13]: model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

配置训练方法

```
In  [14]: model.compile(optimizer='adam',
                        loss='sparse_categorical_crossentropy',
                        metrics=['sparse_categorical_accuracy'])
```

标签值：0~9

预测值：概率分布

训练模型

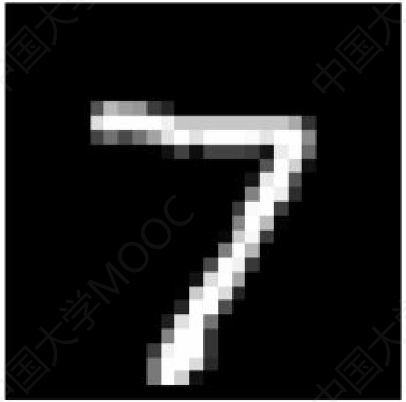
```
In [15]: model.fit(X_train, y_train, batch_size=64,epochs=5,validation_split=0.2)

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>
```

评估模型

```
In  [16]: model.evaluate(X_test,  y_test,  verbose=2)

          10000/1 - 0s - loss: 0.0446 - sparse_categorical_accuracy: 0.9730

Out[16]: [0.0874457276863046, 0.973]
```

使用模型

```
In  [17]: plt.axis("off")
          plt.imshow(test_x[0],cmap="gray")
          plt.show()
```



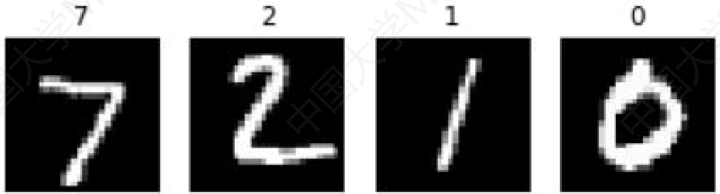
```
In  [18]: y_test[0]

Out[18]: <tf.Tensor: id=15906, shape=(), dtype=int16, numpy=7>
```

使用模型

```
In  [19]: model.predict([[X_test[0]]])

Out[19]: array([[01.5193305e-07, 11.4296833e-07, 25.6485705e-06, 36.1900186e-04,
                 43.7725050e-09, 56.8479079e-07, 61.1266648e-10, 79.9935585e-01,
                 89.8319993e-07, 91.7526505e-05]], dtype=float32)

In  [20]: np.argmax(model.predict([[X_test[0]]]))

Out[20]: 7
```

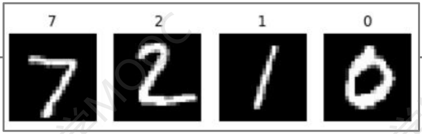
■ 使用模型——测试集中**前4个**数据

In [21]:
```
for i in range(4):

    plt.subplot(1,4,i+1)
    plt.axis("off")
    plt.imshow(test_x[i],cmap='gray')
    plt.title(test_y[i])

plt.show()
```



使用模型——测试集中**前4个**数据

```
In  [22]: model.predict(X_test[0:4])

Out[22]: array([[1.51932895e-07, 1.42968190e-07, 5.64855463e-06, 6.19000697e-04,
                 3.77249032e-09, 6.84789484e-07, 1.12666480e-10, 9.99355853e-01,
                 9.83197992e-07, 1.75264722e-05],
                [2.39225937e-08, 1.46651606e-03, 9.98372793e-01, 1.20054741e-04,
                 2.44312660e-11, 1.68191150e-06, 2.73193677e-06, 3.58970839e-11,
                 3.62173996e-05, 5.76994064e-10],
                [8.24031304e-06, 9.95415926e-01, 4.08277119e-04, 2.54331389e-04,
                 5.98004612e-04, 5.25890682e-05, 4.77820053e-04, 1.27647584e-03,
                 1.49090553e-03, 1.73707685e-05],
                [9.99874830e-01, 1.23330821e-08, 4.77576832e-05, 1.29433559e-07,
                 4.56634234e-06, 1.37567758e-06, 3.26714071e-05, 2.61713703e-05,
                 1.33380125e-08, 1.23481877e-05]], dtype=float32)

In  [23]: np.argmax(model.predict(X_test[0:4]),axis=1)

Out[23]: array([7, 2, 1, 0], dtype=int64)
```

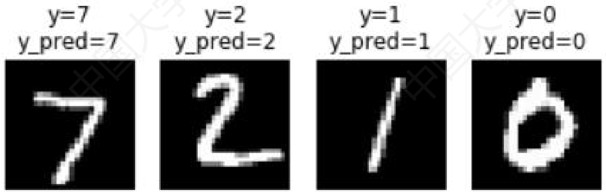


13.5 实例：Sequential实现手写数字识别

■ 使用模型——测试集中**前4个**数据

```
In [24]: y_pred=np.argmax(model.predict(X_test[0:4]),axis=1)
```

```
In [25]: for  i in range(4):

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

         plt.show()
```



- 使用模型——测试集中**随机**取4个数据

```
In [26]: 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()
```