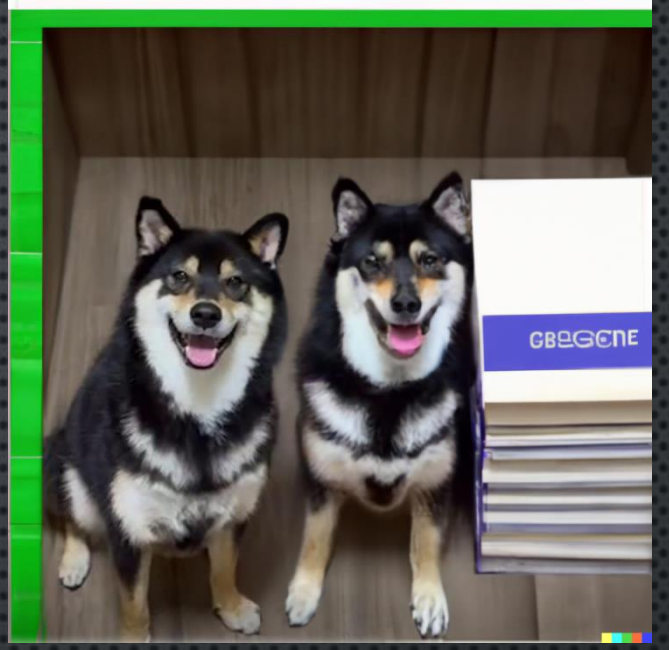
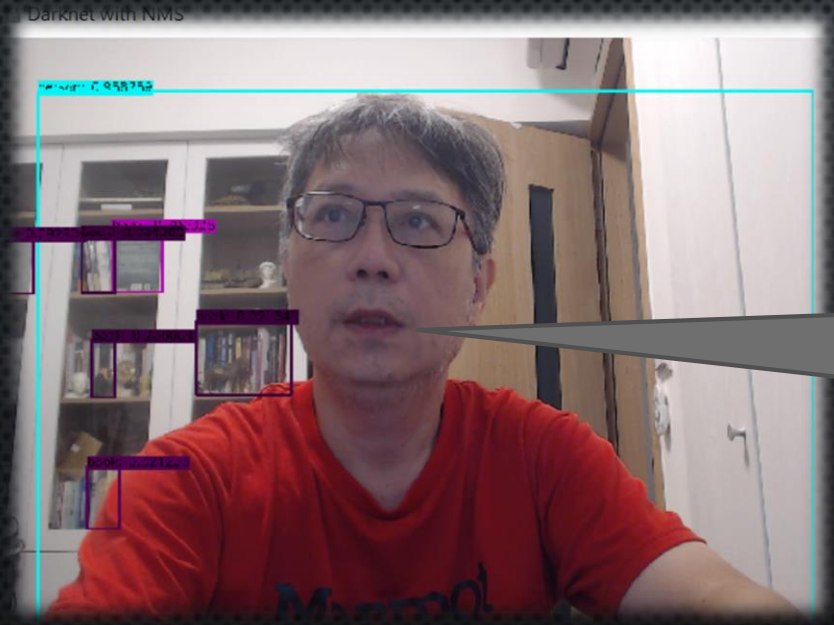
AI人工智慧—
Keras實作神經網路模型





先玩一下Keras實作神經網路模型

bon anwendeng

Bard對Keras的簡介

Keras 是 Google 開發的開源高階神經網路 API。它可以作為 TensorFlow、Microsoft-CNTK 和 Theano 的高階應用程序接口，進行深度學習模型的設計、調試、評估、應用和可視化。

Keras 的特色包括：

- 使用 Python 編寫，可以與 NumPy、SciPy 等其他流行的 Python 庫進行無縫集成。
- 提供簡潔、直觀的 API，可以快速實現深度學習模型。
- 支持多種後端，包括 TensorFlow、Microsoft-CNTK 和 Theano。

Keras 可以用來進行各種深度學習任務，包括：

- 圖像識別
- 自然語言處理
- 機器翻譯
- 推荐系統

keras.io/zh/

Keras 中文文檔

cs

基於Python 的深度學習函式

發現了Keras。

則

始：30 秒上手Keras

引

的Keras 後端

援

取名為Keras?

擇Keras?

Losses

Metrics

ptimizers

Next »

Keras: 基於Python 的深度學習函式庫

你恰好發現了Keras。

Keras 是一個用Python 編寫的高階神經網路API，它能夠以TensorFlow , CNTK , 或Theano作為後端運作。Keras 的開發重點是支援快速的實驗。*能夠以最小的延遲把你的想法轉換成實驗結果，是做好研究的關鍵。*

如果你在以下情況下需要深度學習庫，請使用Keras：

- 允許簡單而快速的原型設計（由於用戶友好，高度模組化，可擴展性）。
- 同時支援卷積神經網路和循環神經網絡，以及兩者的組合。
- 在CPU 和GPU 上無縫運作。

查看文檔，請造訪Keras.io。

MNIST 手寫字元資料集

訓練集為60,000 張28x28 像素灰階影像，測試集為10,000 同規格影像，總共10 類數位標籤。

用法：

```
from keras.datasets import mnist

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

- **返回**：
  - 2 個元組：
    - **x_train, x_test**：uint8 陣列表示的灰階影像，尺寸為(num_samples, 28, 28)。
    - **y_train, y_test**：uint8 陣列表示的數字標籤（範圍在0-9 之間的整數），尺寸為(num_samples,)。
- **參數**：
  - **path**：如果在本機沒有索引檔案(at `'~/.keras/datasets/' + path`)，它將被下載到該目錄。

scholar.google.com/citations?user=U_IVY50AAAAJ

Corinna Cortes

追蹤

Google Research, NY
在 google.com 的電子郵件地址已通過驗證 - 首頁

Machine Learning Datamining

標題	引用次數	年份
Support-vector networks C Cortes, V Vapnik Machine learning 20, 273-297	63331	1995
MNIST handwritten digit database Y LeCun, C Cortes, C Burges	4923	2010
Support vector machine C Cortes, V Vapnik Machine learning 20 (3), 273-297	1446	1995
Comparison of classifier methods: a case study in handwritten digit recognition L Bottou, C Cortes, JS Denker, H Drucker, I Guyon, LD Jackel, Y LeCun, ... Proceedings of the 12th IAPR International Conference on Pattern Recognition …	1031	1994
MNIST handwritten digit database. 2010 Y LeCun, C Cortes, CJ Burges URL http://yann. lecun. com/exdb/mnist 7 (23), 6	909	2010
Comparison of learning algorithms for handwritten digit recognition Y LeCun, L Jackel, L Bottou, A Brunot, C Cortes, J Denker, H Drucker, ...	900	1995

bard

Cortes 與 LeCun 合作創建了 MNIST 數據集。MNIST 數據集包含 70,000 個手寫數字圖像，每個圖像都是一個 28x28 的灰度圖像。這些圖像被分為 60,000 個訓練圖像和 10,000 個測試圖像。MNIST 數據集是深度學習領域中最常用的數據集之一。

- 與 LeCun 合作撰寫了 MNIST 數據集的論文

Cortes 與 LeCun 合作撰寫了論文《The MNIST Database of Handwritten Digit Images》。該論文描述了 MNIST 數據集。MNIST 數據集包含 70,000 個手寫數字圖像，每個圖像都是一個 28x28 的灰度圖像。這些圖像被分為 60,000 個訓練圖像和 10,000 個測試圖像。MNIST 數據集是深度學習領域中最常用的數據集之一。

- 與 LeCun 合作開發了 LeNet 神經網路模型

Cortes 與 LeCun 合作開發了 LeNet 神經網路模型。LeNet 是一個卷積神經網路模型，它在 MNIST 數據集上取得了優異的性能。LeNet 是深度學習領域的里程碑，它證明了卷積神經網路可以用於圖像識別。

from tensorflow.keras.datasets import mnist

```
[1]: print('神經網路模型')
```

```
神經網路模型
```

```
[2]: from tensorflow.keras.datasets import mnist
```

```
[3]: ? mnist
```

```
Type:        module
String form: <module 'keras.api._v2.keras.datasets.mnist' from 'd:\\ProgramData\\Anaconda3\\lib\\site-pac
it__.py'>
File:        d:\programdata\anaconda3\lib\site-packages\keras\api\_v2\keras\datasets\mnist\__init__.py
Docstring:   Public API for tf.keras.datasets.mnist namespace.
```

[5]:
```python
import matplotlib.pyplot as plt
import numpy as np
```

[6]:
```python
(X_train, y_train), (X_test, y_test)=mnist.load_data()
```

[7]:
```python
X_train.shape
```

[7]:
```
(60000, 28, 28)
```

[11]:
```python
np.set_printoptions(linewidth=np.inf)
X_train[7]
```

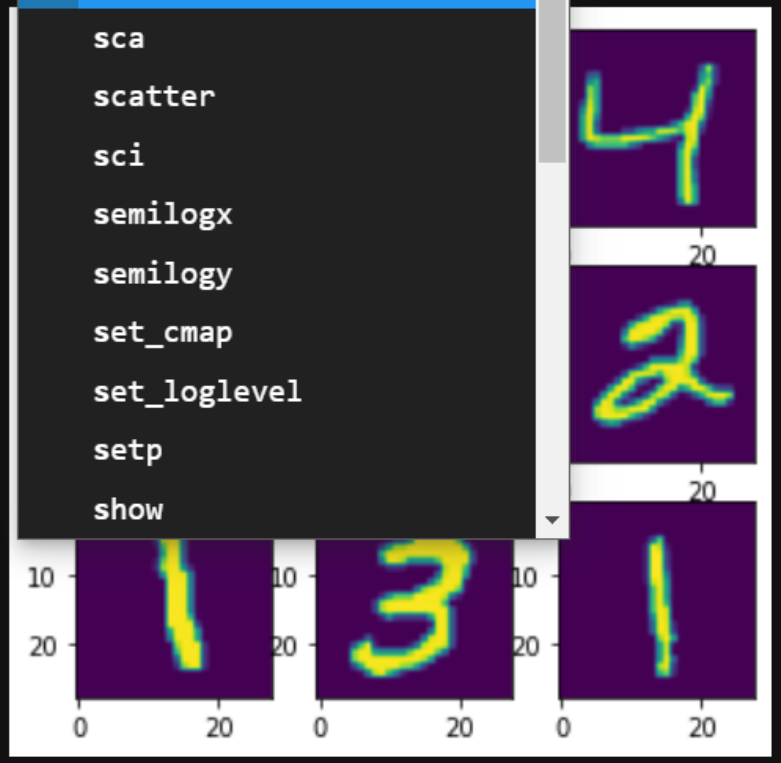
[11]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  38,  43, 105, 255, 253, 253, 253, 253, 253, 174,   6,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  43, 139, 224, 226, 252, 253, 252, 252, 252, 252, 252, 252, 158,  14,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0, 178, 252, 252, 252, 252, 253, 252, 252, 252, 252, 252, 252, 252,  59,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0, 109, 252, 252, 230, 132, 133, 132, 132, 189, 252, 252, 252, 252,  59,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   4,  29,  29,  24,   0,   0,   0,   0,  14, 226, 252, 252, 172,   7,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  85, 243, 252, 252, 144,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  88, 189, 252, 252, 252,  14,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  91, 212, 247, 252, 252, 252, 204,   9,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  32, 125, 193, 193, 193, 253, 252, 252, 252, 238, 102,  28,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  45, 222, 252, 252, 252, 252, 253, 252, 252, 252, 177,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  45, 223, 253, 253, 253, 253, 255, 253, 253, 253, 253,  74,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  31, 123,  52,  44,  44,  44,  44, 143, 252, 252,  74,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15, 252, 252,  74,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  86, 252, 252,  74,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   5,  75,   9,   0,   0,   0,   0,   0,   0,  98, 242, 252, 252,  74,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  61, 183, 252,  29,   0,   0,   0,   0,  18,  92, 239, 252, 252, 243,  65,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0, 208, 252, 252, 147, 134, 134, 134, 134, 203, 253, 252, 252, 188,  83,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0, 208, 252, 252, 252, 252, 252, 252, 252, 252, 253, 230, 153,   8,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  49, 157, 252, 252, 252, 252, 252, 217, 207, 146,  45,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   7, 103, 235, 252, 172, 103,  24,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   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=uint8)
```

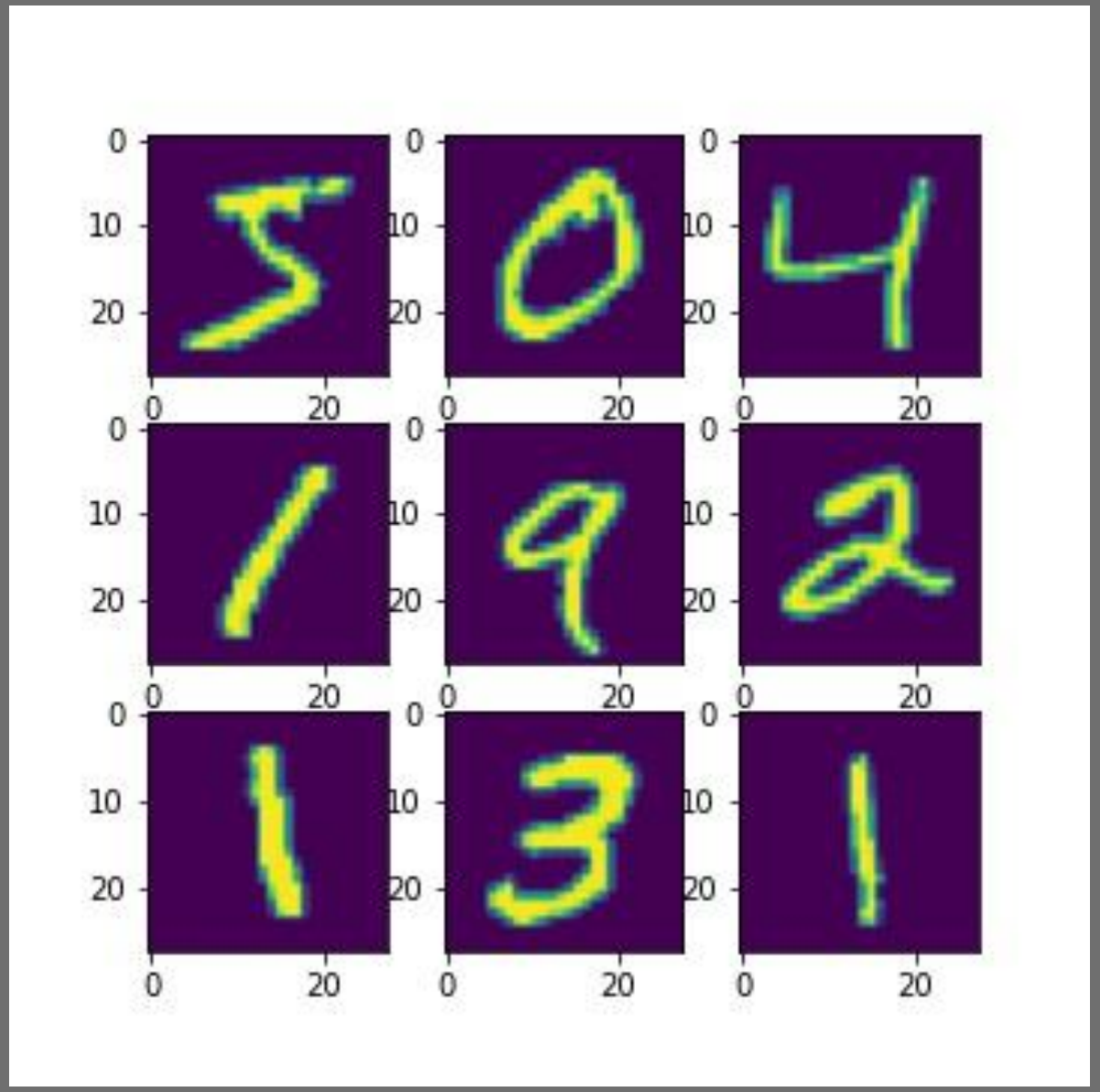
```
[9]: y_train[0:10]
```

```
[9]: array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

```
[10]: plt.figure(figsize=(5, 5))
      for i in range(9):
          plt.subplot(3, 3, i+1)
          plt.imshow(X_train[i])
      plt.s
```

```
f savefig          function
  sca
  scatter
  sci
  semilogx
  semilogy
  set_cmap
  set_loglevel
  setp
  show
```




0
10
20
0
20

快速開始：30 秒上手Keras

Keras 的核心資料結構是**model**，一種組織網路層的方式。最簡單的模型是Sequential 順序模型，它由多個網路層線性堆疊。對於更複雜的結構，你應該使用Keras 函數式API，它允許建立任意的神經網路圖。

`Sequential` 模型如下圖所示：

```
from keras.models import Sequential

model = Sequential()
```

可以簡單地使用 `.add()` 來堆疊模型：

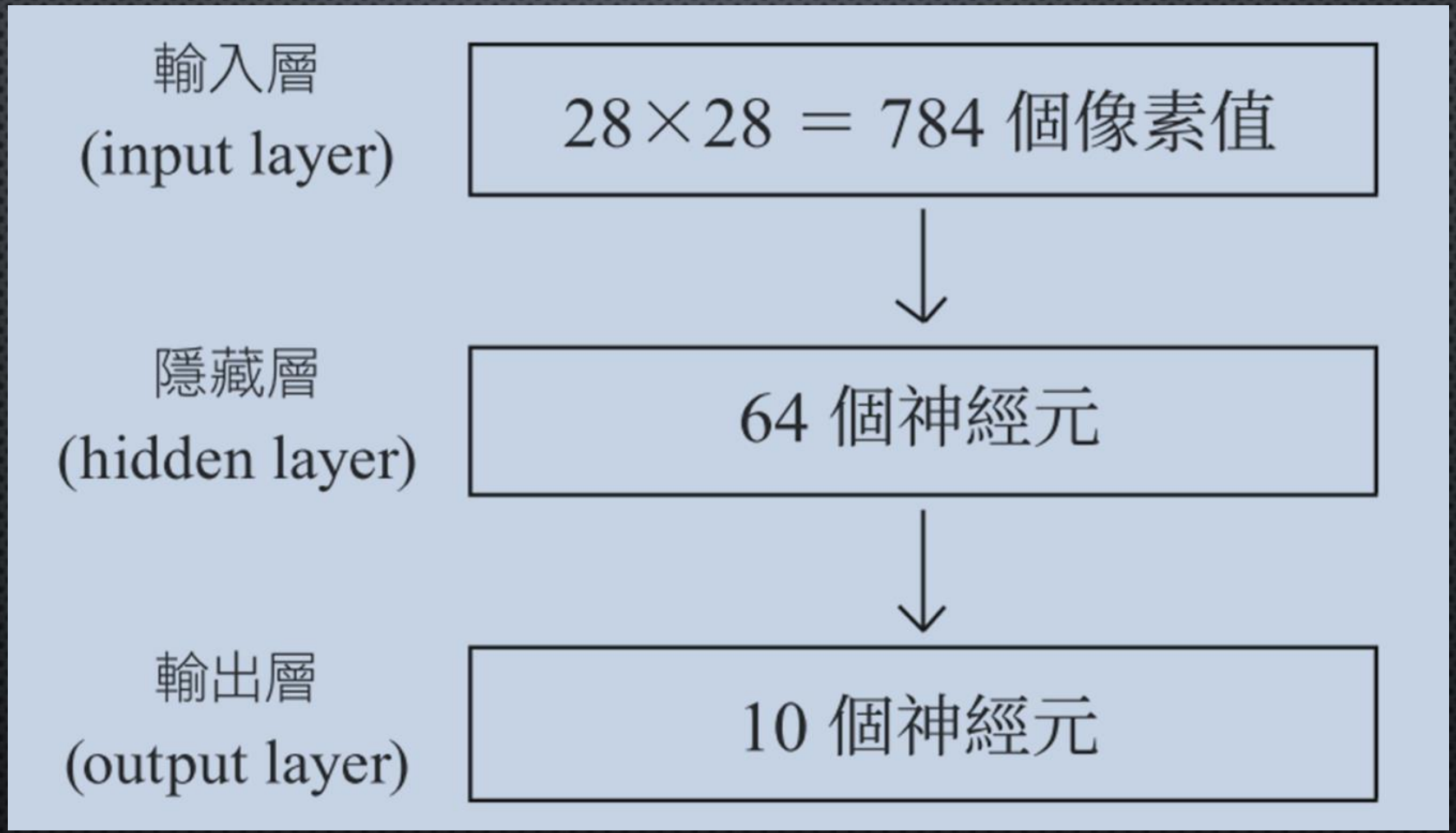
```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

在完成了模型的建構後, 可以使用 `.compile()` 來配置學習過程：

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

如果需要，你也可以進一步配置你的優化器。Keras 的核心原則是讓事情變得相當簡單，同時又允許使用者在需要的時候能夠進行完全的控制（終極的控制是原始碼的易擴展性）。

[4]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers
from tensorflow.keras.utils import *
```


輸入層
(input layer)
28×28 ＝ 784 個像素值
隱藏層
(hidden layer)
64 個神經元
輸出層
(output layer)
10 個神經元

Flatten, Normalization

```
X_train=X_train.reshape(60000, 784).astype('float32')
X_test=X_test.reshape(10000, 784).astype('float32')
```

```
#normalizing
X_train/=255
X_test/=255
X_train[0]
```

```
1 n_class=10
2 y_train=to_categorical(y_train, n_class) #one-hot coding
3 y_test=to_categorical(y_test, n_class)
```

建立三層的淺層神經網路

```
print('欲處理好了後面就是關鍵，剩下5個指令就好了')
```

```
model=Sequential()
model.add(Dense(64, activation='relu', input_shape=(784, )))  #sigmoid, tanh, relu
model.add(Dense(10, activation='softmax'))
```

Loss, 優化器、metric

```
model.compile(
    loss='mean_squared_error',
    optimizer=optimizers.SGD(learning_rate=0.01), #隨機梯度下降法(Stochastic gradient descent, SGD)
    metrics=['accuracy']
)
```

學習與驗證

```
model.fit(
    X_train, y_train, batch_size=128, epochs=200,
    verbose=1,
    validation_data=(X_test, y_test)
)
```

軟體實作

von anwendeng

感謝觀賞

Herzlichen Dank für die Aufmerksamkeit

von anwendeng