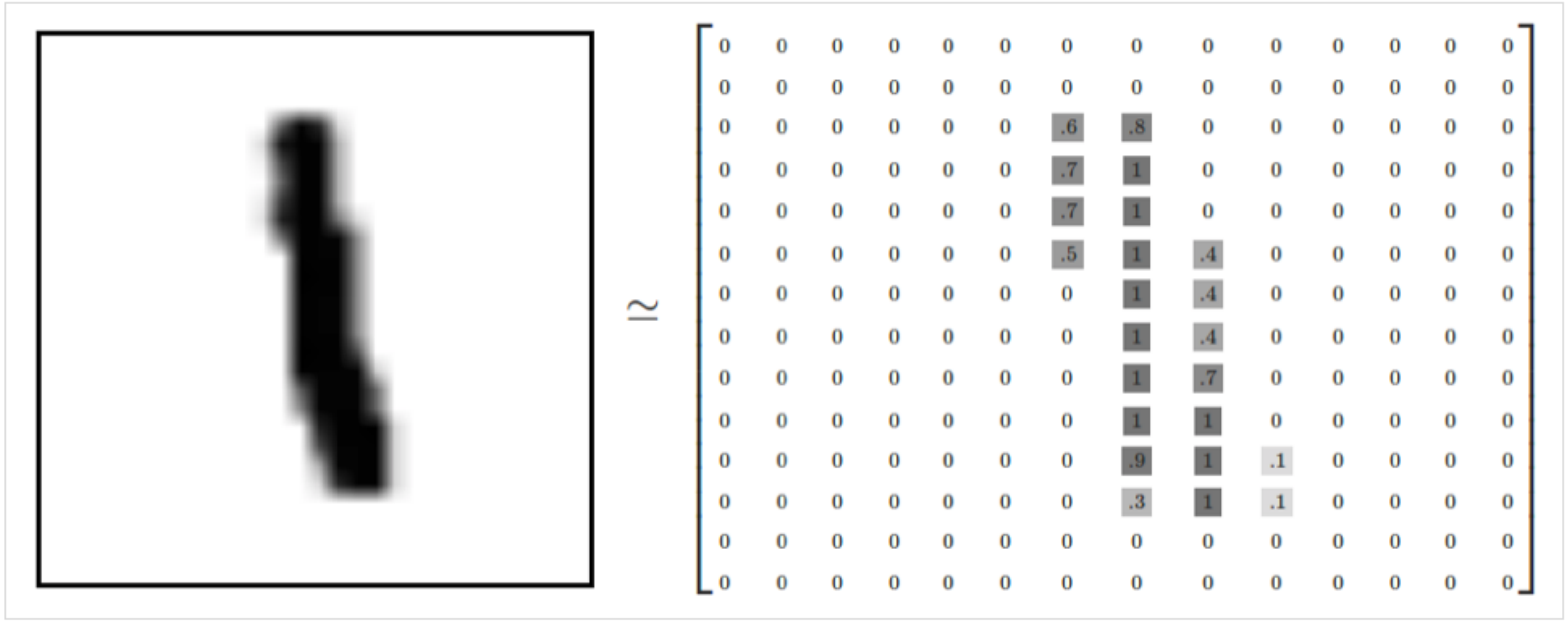
TensorFlow+Keras
卷積神經網路CNN

吳佳諺　老師

TensorFlow
卷積神經網路CNN

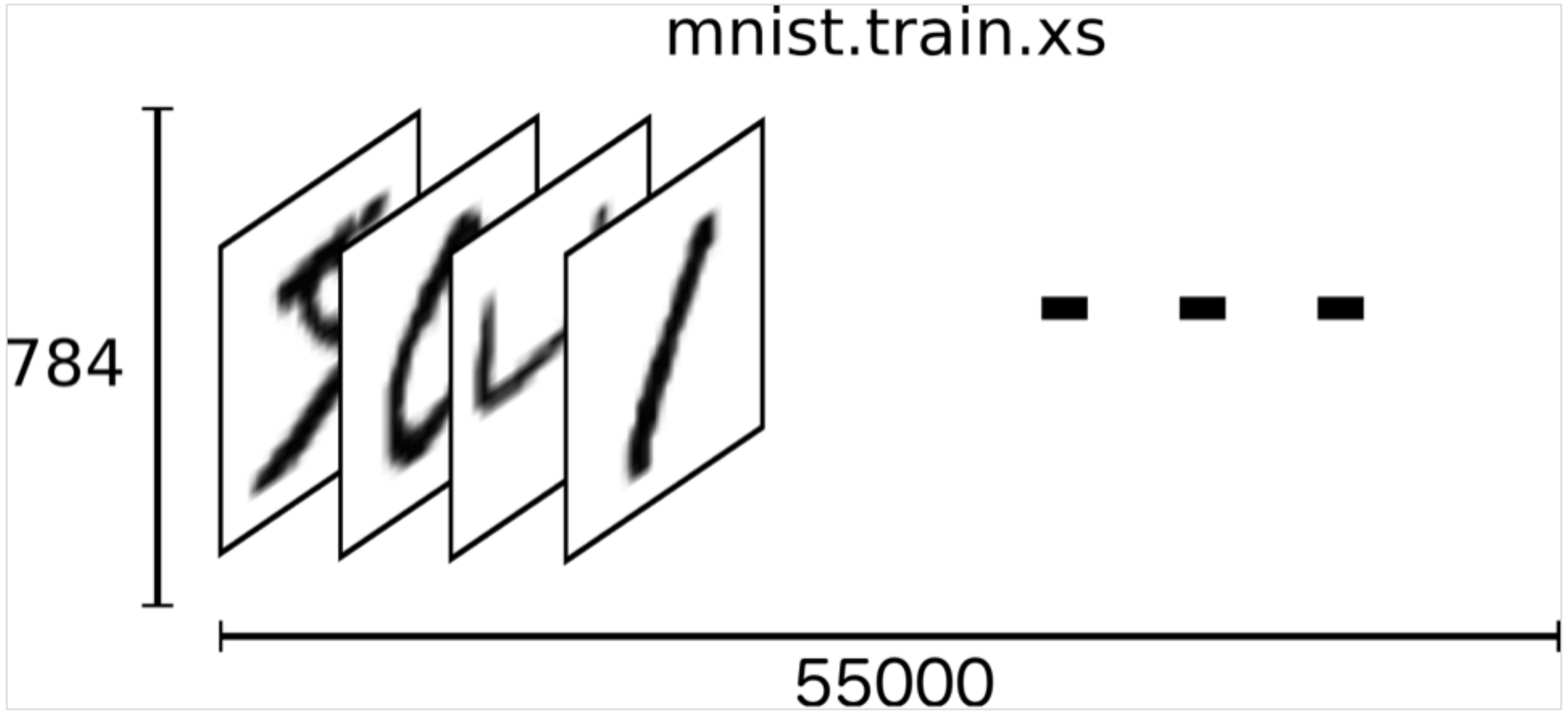
1.卷積神經網路CNN
2.CNN,特徵取樣
3.CNN程式使用keras

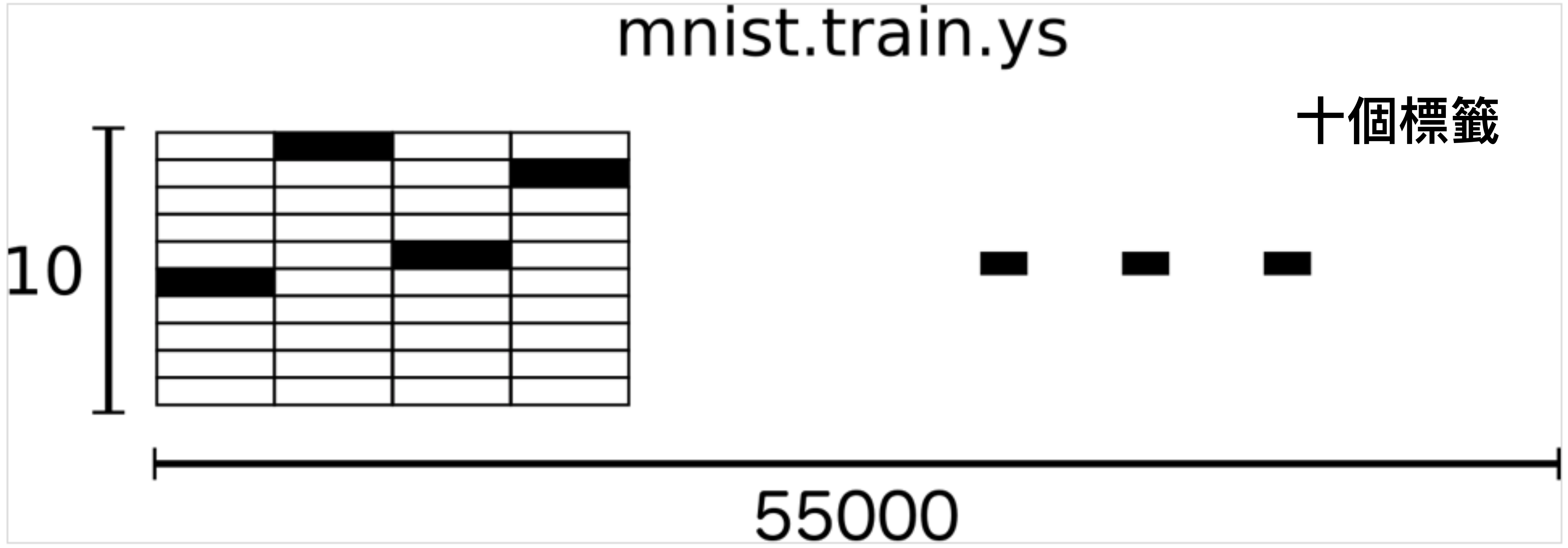
1的圖形由28*28的矩陣形成



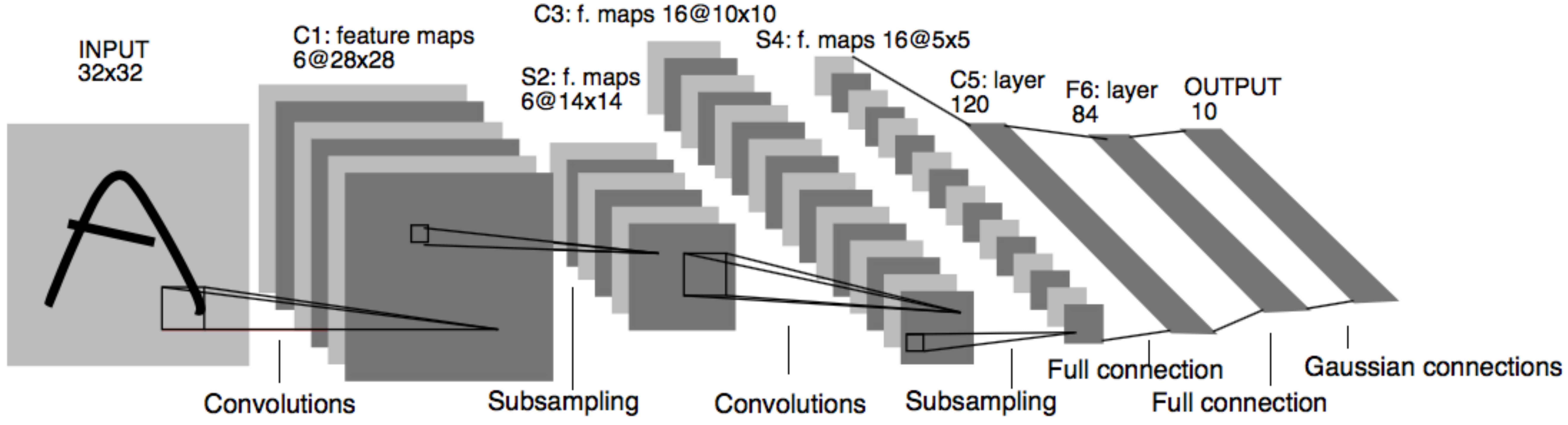
$$\cong \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

28*28為784的像素

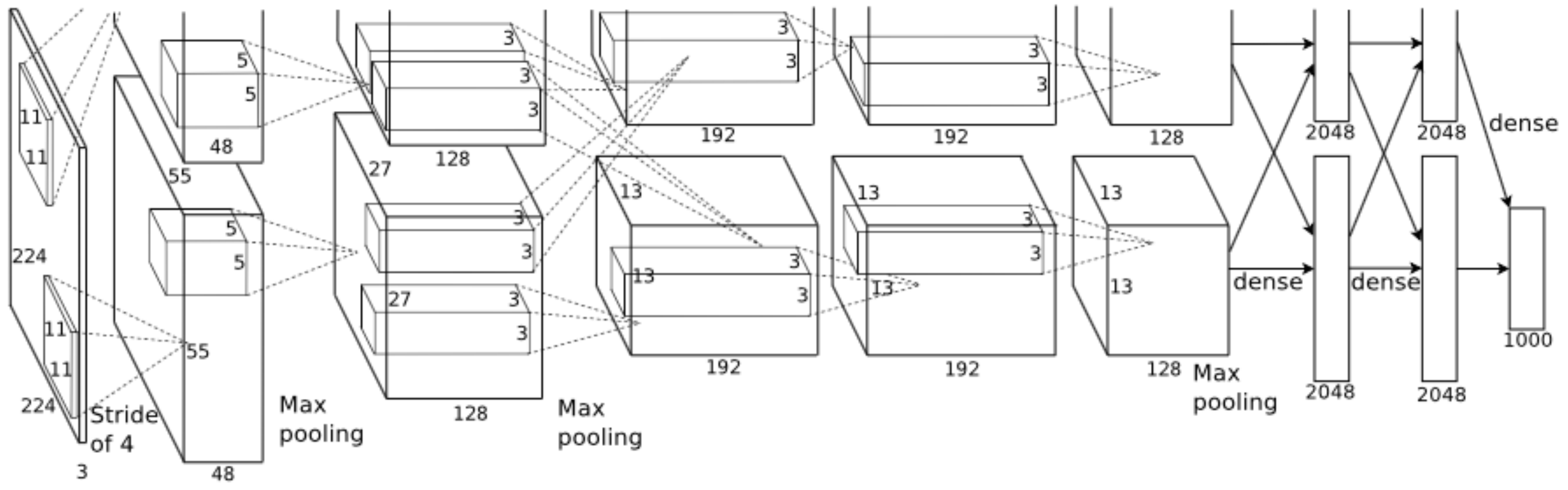




1.卷積神經網路CNN


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

CNN架構圖


11
11
224
224
3
Stride of 4
55
55
5
5
48
48
Max pooling
27
27
128
128
3
Max pooling
13
192
192
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

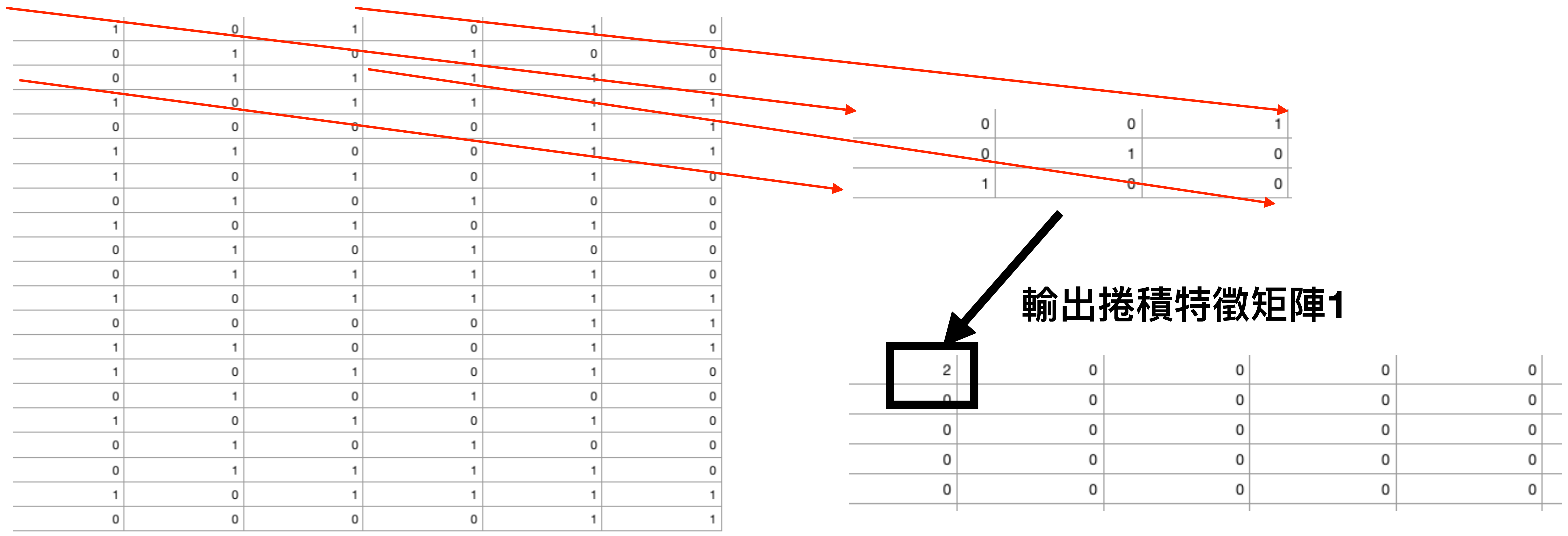
Convolutional Layer #1

- model.add(Conv2D(32, kernel_size=(3, 3),

  activation='relu',

  input_shape=input_shape))
- 運用32個3*3的影像過濾器
- 啟用ReLU啟用函數
- 28*28*1的輸入,影像28*28尺寸,1是黑白

影像過濾擷取特徵矩陣



影像矩陣

1	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	1
0	0	0	0	1	1
1	1	0	0	1	1
1	0	1	0	1	0
0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	1
0	0	0	0	1	1
1	1	0	0	1	1
1	0	1	0	1	0
0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	1
0	0	0	0	1	1

*

3*3特徵擷取矩陣1

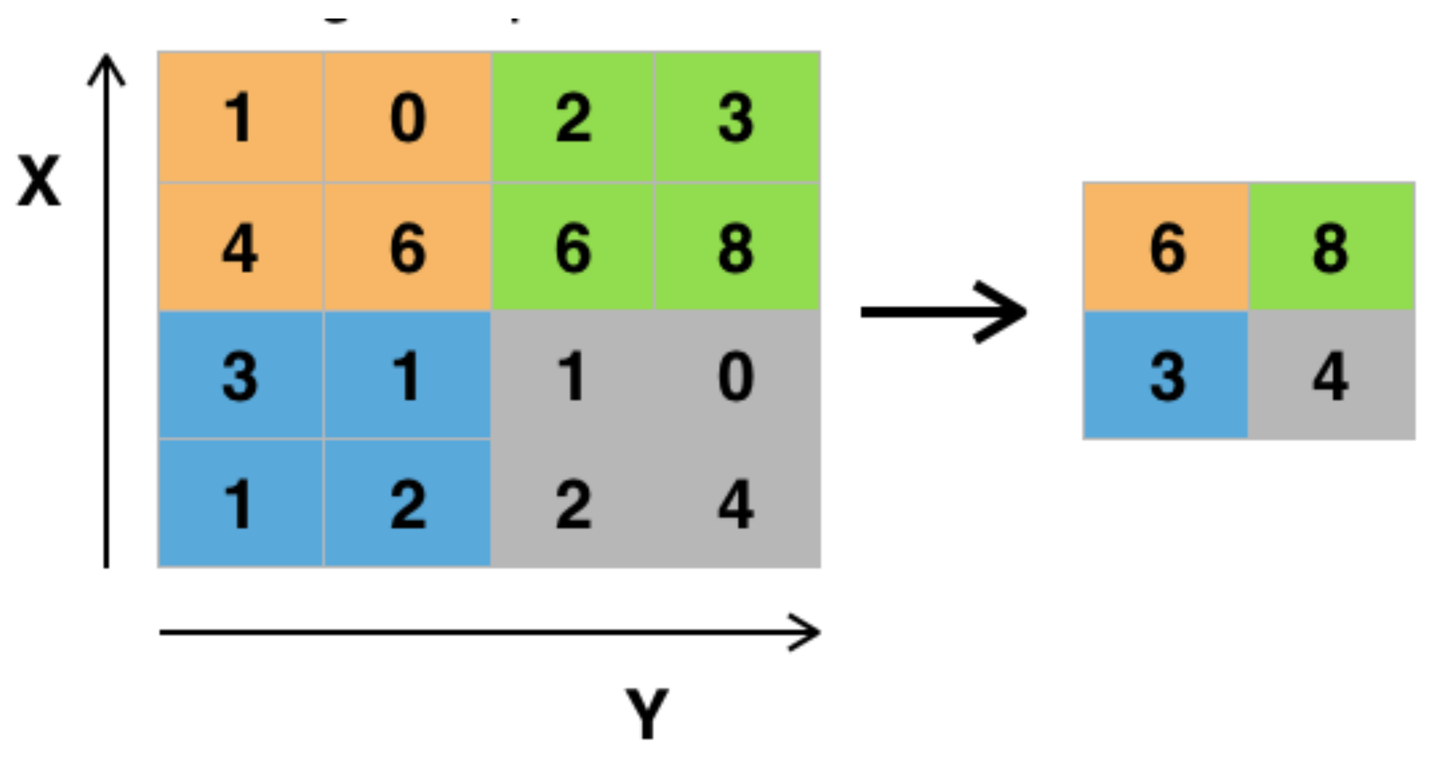
0	0	1
0	1	0
1	0	0

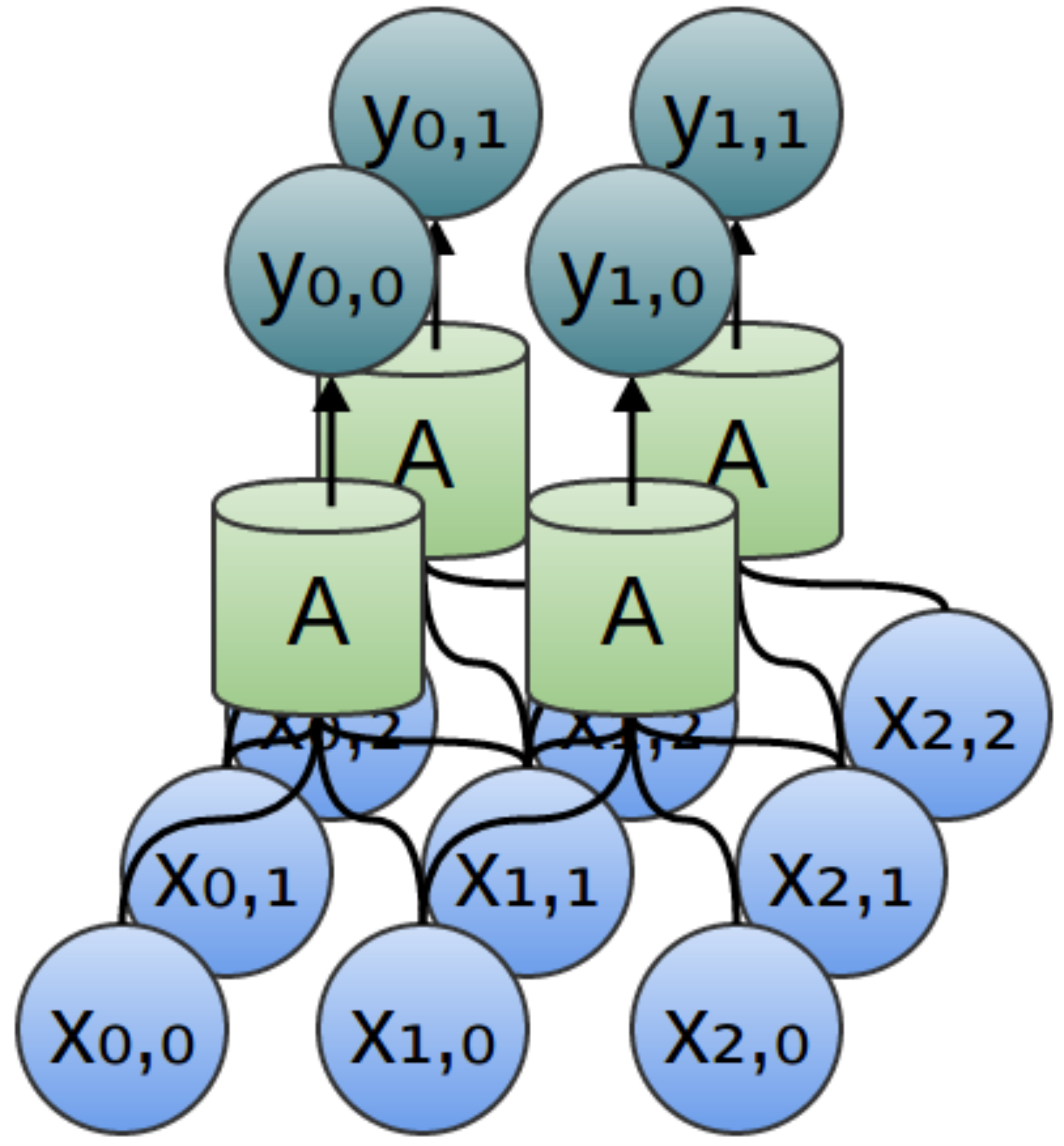
輸出捲積特徵矩陣1

$\boxed{2}$	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Pooling Layer #1第一層最大池

- model.add(MaxPooling2D(pool_size=(2, 2)))
- 最大池用2*2的矩陣
- 由圖片濾波器的到最大池矩陣[6,8,3,4]
- 將28*28的矩陣縮成14*14





建立平坦層

- model.add(Flatten())
- 將64個12*12的矩陣,轉換成一維向量9216個神經元

Dense Layer #1神經元連階層

- model.add(Dense(128, activation=‘relu'))
- 神經元連接層,有128個神經元
- 啟動函數為relu

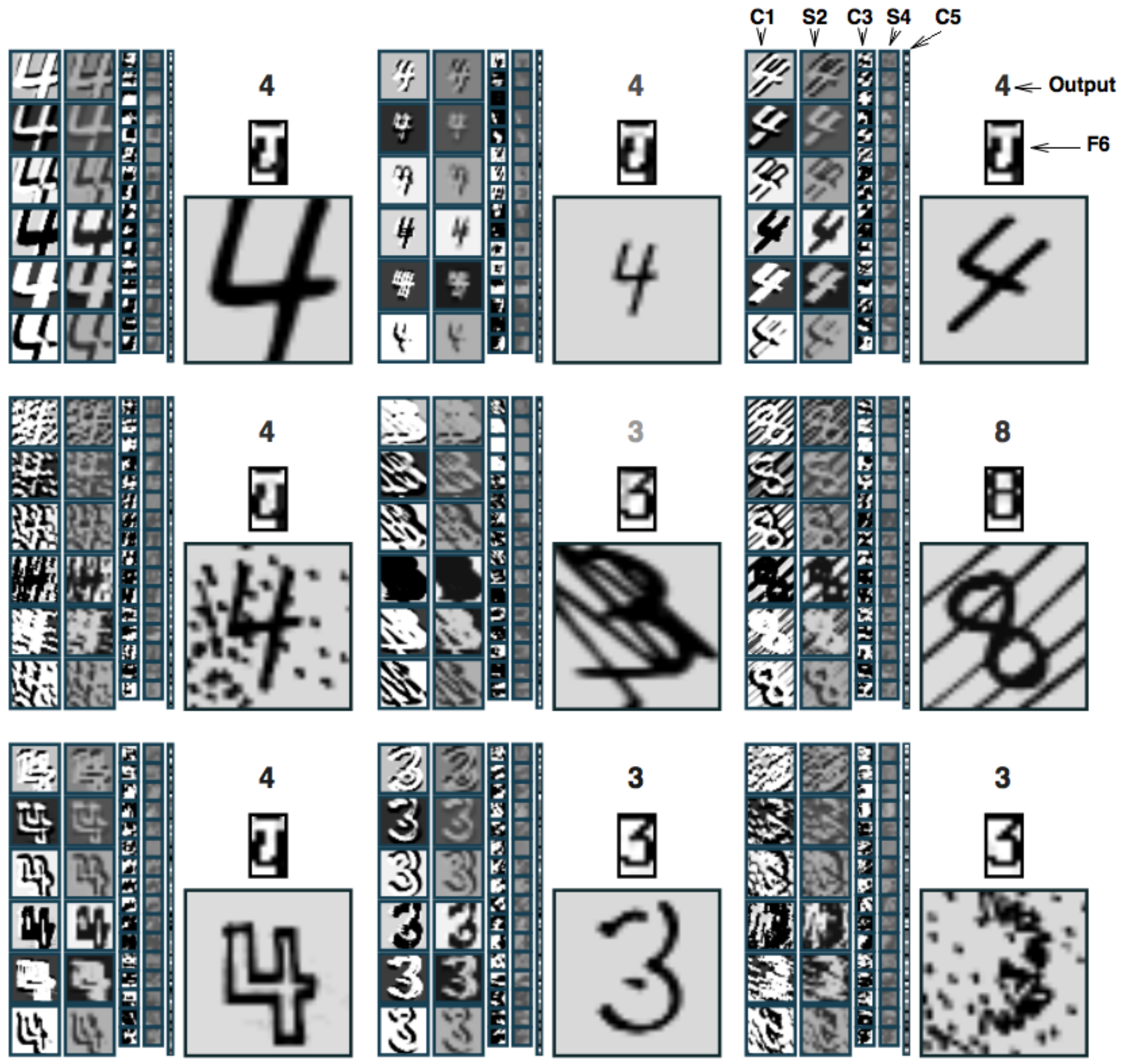
- model.add(Dropout(0.3))
- Droupout(0.3)比率為30%

Dense Layer #2輸出層,10個神經元

- model.add(Dense(num_classes, activation=‘softmax'))
- 可輸出0到9的10個輸出層,啟動函數是softmax

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad for \;\; j=1,\ldots,K$$

2.CNN,特徵取樣

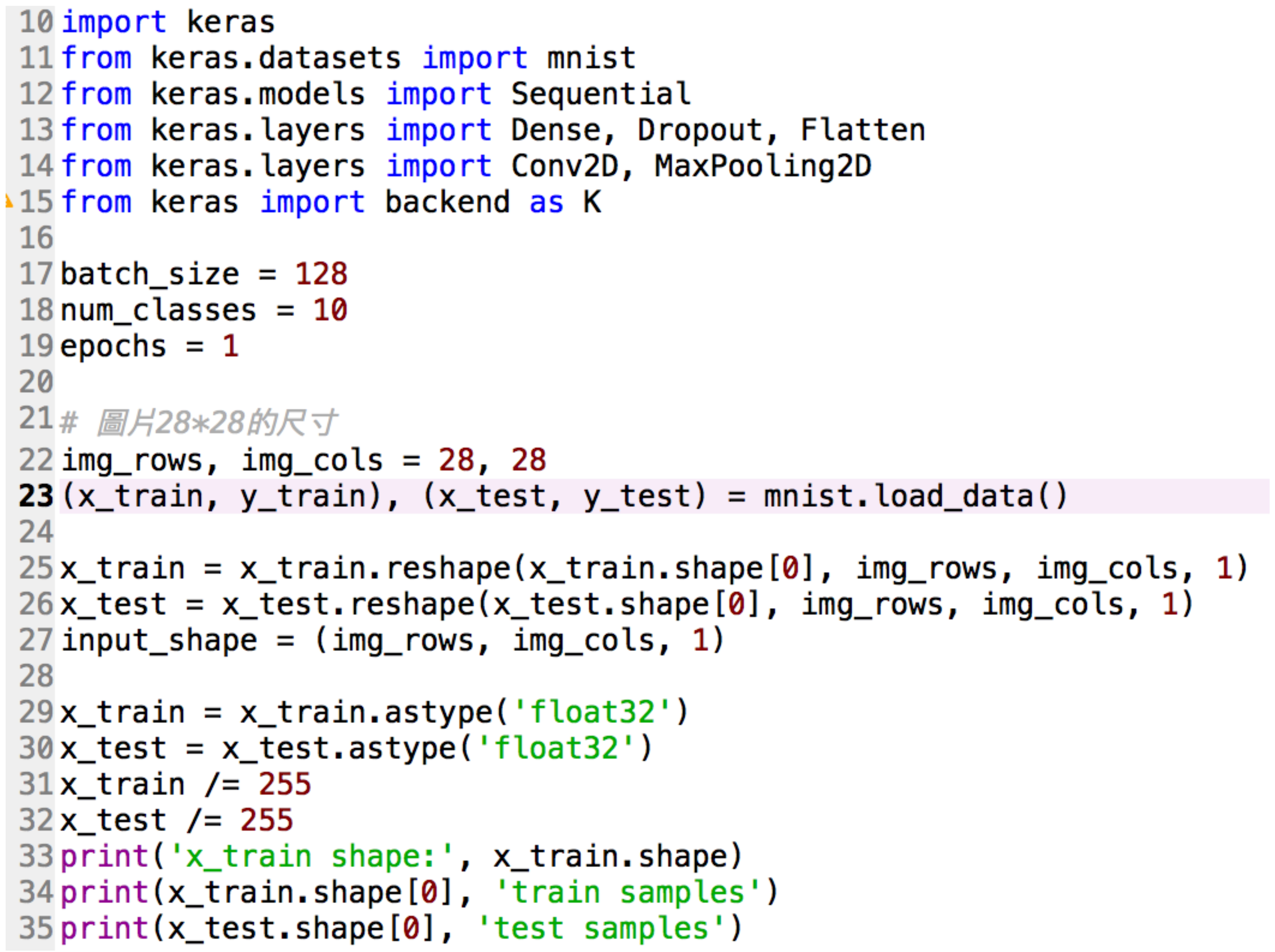


CNN,特徵取樣

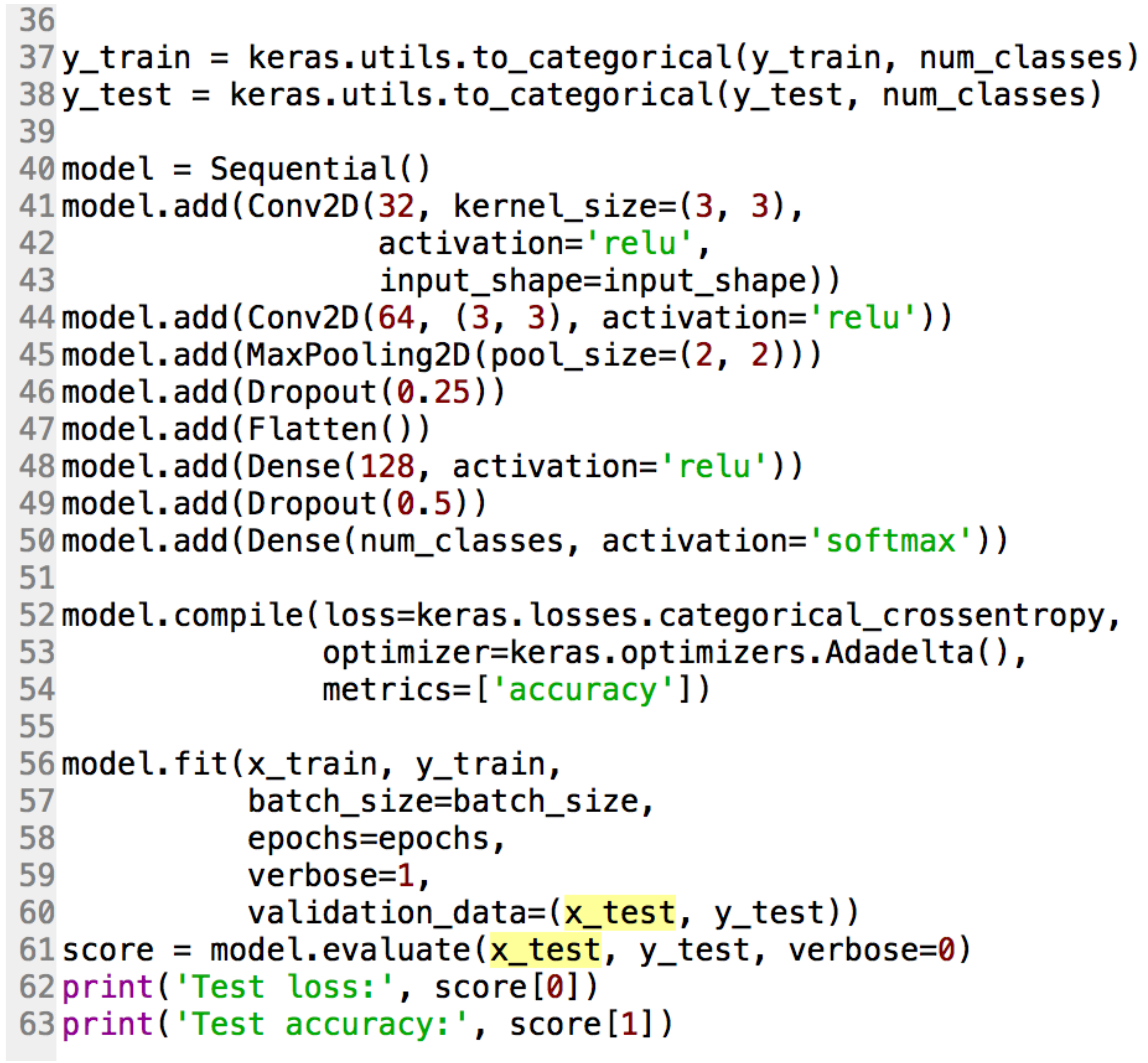
- CNN的基本結構包括兩層，其一為特徵選取層
- 每個神經元的輸入與前一層的局部接受域相連，並提取該局部的特徵。
- 局部特徵被擷取後，它與其它特徵間的位置關係也隨之確定下來.
- 特徵映射層，網絡的每個計算層由多個特徵映射組成，每個特徵映射是一個平面，平面上所有神經元的權值相等。
- 卷積網絡的啟動函數，使得特徵映射具有位移不變性。

- CNN主要用來識別位移、縮放及其他形式扭曲不變性的二維圖形。
- CNN的特徵檢測層通過訓練數據進行學習，所以在使用CNN時，避免了顯示的特徵抽取，而隱式地從訓練數據中進行學習
- 同一特徵映射面上的神經元權值相同，所以網絡學習可以平行處理
- 卷積網絡相對於神經元彼此相連網絡的一大優勢。

3.CNN程式使用keras

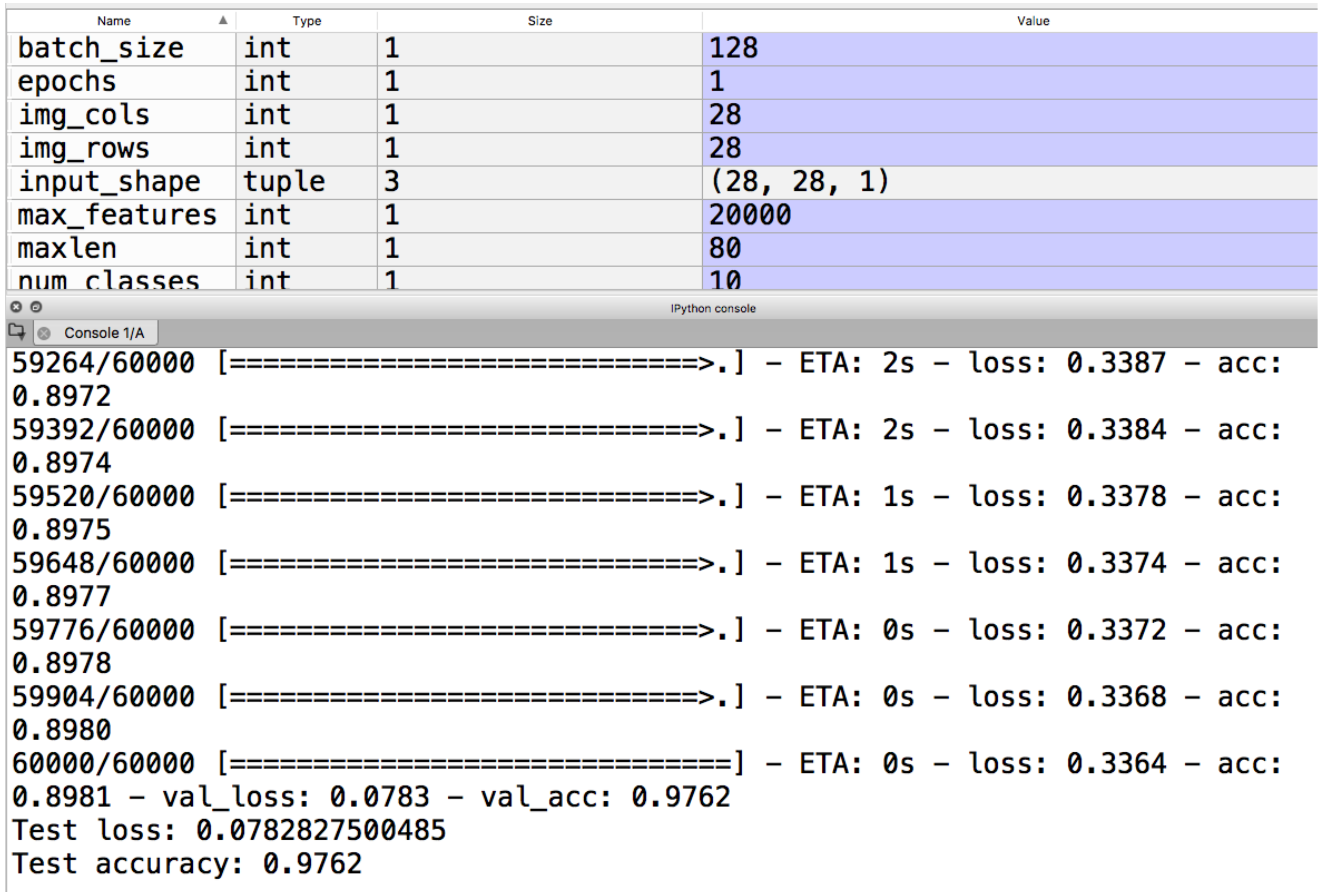


```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 1

# 圖片28*28的尺寸
img_rows, img_cols = 28, 28
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```



```python

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

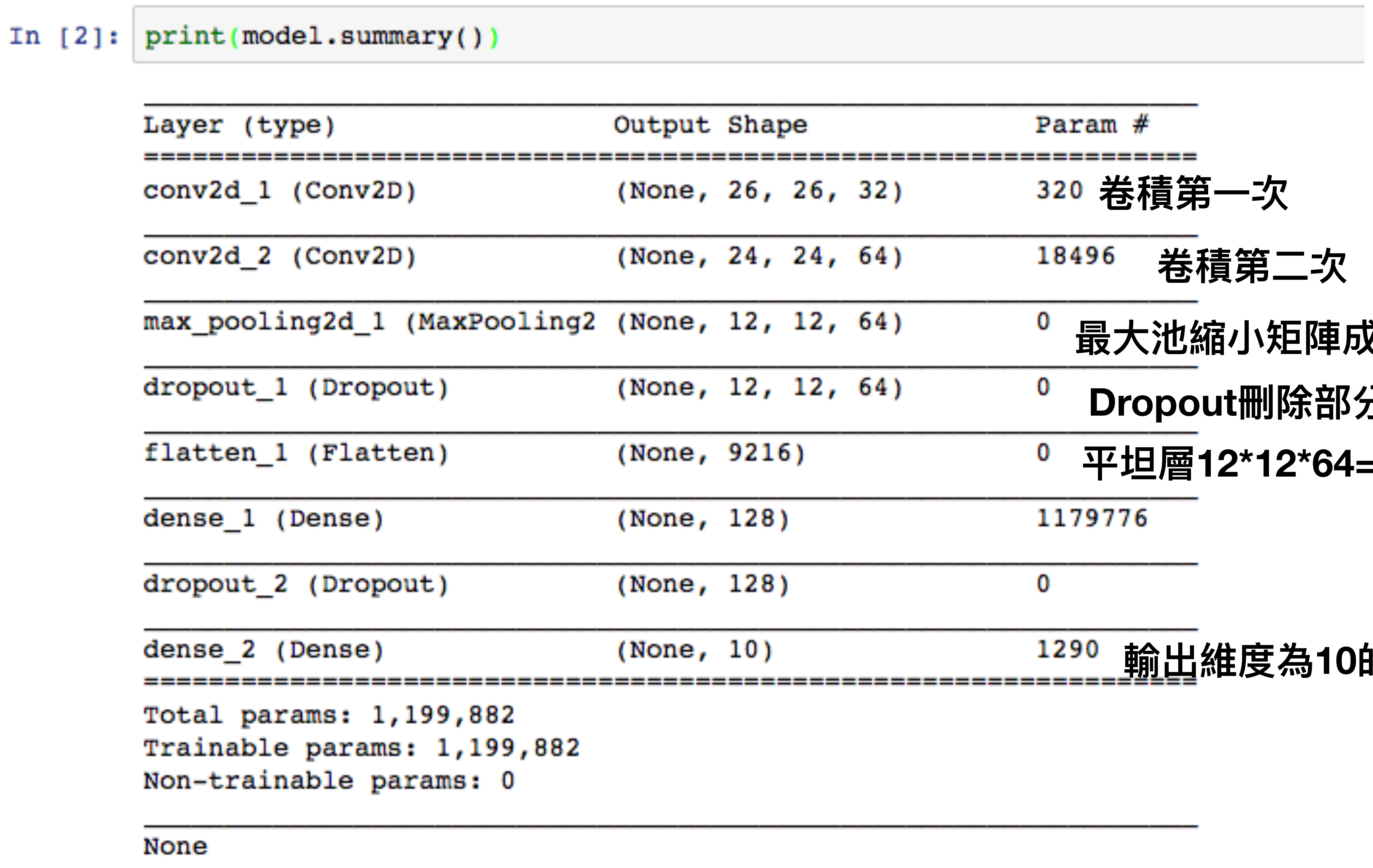
可以看到執行CNN結果



Name	Type	Size	Value
batch_size	int	1	128
epochs	int	1	1
img_cols	int	1	28
img_rows	int	1	28
input_shape	tuple	3	(28, 28, 1)
max_features	int	1	20000
maxlen	int	1	80
num_classes	int	1	10

```
59264/60000 [============================>.] - ETA: 2s - loss: 0.3387 - acc:
0.8972
59392/60000 [============================>.] - ETA: 2s - loss: 0.3384 - acc:
0.8974
59520/60000 [============================>.] - ETA: 1s - loss: 0.3378 - acc:
0.8975
59648/60000 [============================>.] - ETA: 1s - loss: 0.3374 - acc:
0.8977
59776/60000 [============================>.] - ETA: 0s - loss: 0.3372 - acc:
0.8978
59904/60000 [============================>.] - ETA: 0s - loss: 0.3368 - acc:
0.8980
60000/60000 [==============================] - ETA: 0s - loss: 0.3364 - acc:
0.8981 - val_loss: 0.0783 - val_acc: 0.9762
Test loss: 0.0782827500485
Test accuracy: 0.9762
```

卷積類神經網路的架構



```
In [2]: print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
None
```

卷積第一次

卷積第二次

最大池縮小矩陣成為12*12

Dropout刪除部分神經元

平坦層12*12*64=9216

輸出維度為10的矩陣

- Thanks.