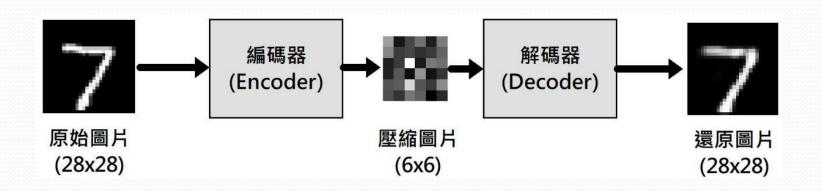
自編碼器(Autoencoder)

吳庭育 tyw@mail.npust.edu.tw

Cifar-10彩色圖片資料集

• 自編碼器(Autoencoder)是一種實現編碼和解碼的神經網路 ,一種資料壓縮演算法,可以將原始資料透過編碼器 (Encoder)的神經網路進行壓縮,和使用解碼器(Decoder) 的神經網路還原成原始資料。



自編碼器的特色

- 只適用特定資料(Data-specific)
 - 自編碼器只適用與訓練資料集相似的資料壓縮。
- 資料損失
 - 自編碼器壓縮和還原會有資料損失,還原資料不會和原始資料完全相同。
- 非監督是學習
 - 自編碼器是自行從資料學習,因為訓練資料集就是和自己比較損失來進行學習。
- 自編碼器常用來減少資料集的維度,但仍然可以保留資料集的主要特徵,就是『主成分分析』(Principal Component Analysis, PCA),就是降維(Dimensionality Reduction)的特徵擷取

Keras 約 Functional API

Keras神經層物件就是一個函式

在Keras建立的神經層物件可以當成函式來呼叫,也就是將各神經 層視為是一個函式:

```
a = Input(shape=(32,))
b = Dense(32, activation="relu")(a)
```

上述程式碼先建立Input輸入層物件,傳回值是張量a(這就是輸入層輸入神經網路的特徵資料),然後建立Dense物件,可以將Dense物件視為函式呼叫,函式的參數是此層神經層的輸入張量a,傳回值是此層神經層的輸出張量b,然後,我們可以建立Model模型:

model = Model(inputs=a, outputs=b)

- Model()的inputs參數是輸入模型的張量: outputs參數是輸出張量。
- 如果是多輸入和多輸出模型,可以使用清單來指定輸入和輸出張量:

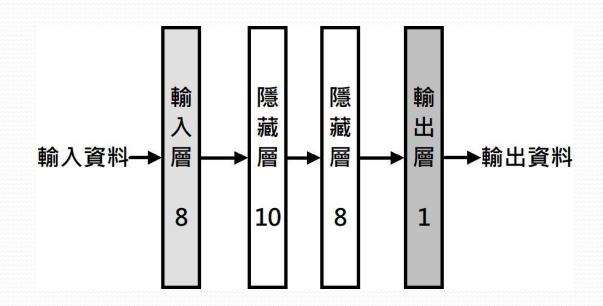
model = Model(inputs=[a1, a2], outputs=[b1, b2, b3])

使用Functional API定義神經網路模型

```
1 import numpy as np
2 import pandas as pd
3 from keras.models import Model
4 from keras. layers import Input, Dense
6 np. random. seed(10) # 指定亂數種子
7 # 載入糖尿病資料集
8 df = pd. read csv("./diabetes.csv")
9 dataset = df. values
10 np. random. shuffle(dataset) # 使用亂數打亂資料
11 # 分割成特徵資料和標籤資料
12 X = dataset[:, 0:8]
13 Y = dataset[:, 8]
14 # 定義模型 - 使用 Function API
15 inputs = Input(shape=(8,))
16 hidden1 = Dense(10, activation="relu")(inputs)
17 hidden2 = Dense(8, activation="relu")(hidden1)
18 outputs = Dense(1, activation="sigmoid") (hidden2)
19 model = Model(inputs=inputs, outputs=outputs)
20 model. summarv() # 顯示模型摘要資訊
21 # 編譯模型
22 model.compile(loss="binary_crossentropy", optimizer="sgd",
                           metrics=["accuracy"])
23
24 # 訓練模型
25 model.fit(X, Y, epochs=15, batch_size=10)
26 # 評估模型
27 loss, accuracy = model.evaluate(X, Y)
28 print("準確度 = {:.2f}".format(accuracy))
```

使用Functional API定義神經網路模型

 將之前糖尿病模型的Sequential模型改用Functional API來建立 Model模型(只有定義模型部分不同,其他部分完全相同),這是 一個四層的深度神經網路。



Functional API定義網路模型

```
建立input層
                                                 input層 的傳回值當
     定義模型
                 使用
                       Function
                                                 作hidden1的輸入
            Input (shape=(8,))
15 inputs
             Dense(10, activation="relu")(inputs)
  hidden1
             Dense (8, activation="relu") (hidden1)
  hidden2
          =
18 outputs = Dense(1, activation="sigmoid") (hidden2)
19 model = Model(inputs=inputs, outputs=outputs)
                      # 顯示模型摘要資訊
20 model. summary()
```

參數inputs 是模型的輸入張量(input層的傳回值) 參數outputs是模型的輸出張量(output層的傳回值)

使用MILP建立自編碼器(AE)

```
1 import numpy as np
 2 from keras.datasets import mnist
 3 from keras. models import Model
 4 from keras. layers import Dense, Input
 5
 6 # 指定亂數種子
 7 \text{ seed} = 7
                                                                      39 # 訓練模型
 8 np. random. seed (seed)
                                                                      40 autoencoder.fit(X_train, X_train, validation_data=(X_test, X_test),
 9 # 載入資料集
                                                                                                    epochs=10, batch size=256, shuffle=True, verbose=2)
                                                                      41
10 (X train, ), (X test, ) = mnist.load data()
                                                                      42 # 壓縮圖片
11 # 轉換成 28*28 = 784 的向量
12 X_train = X_train.reshape(X_train.shape[0], 28*28).astype("float32", 43 encoded imgs = encoder.predict(X test)
13 X_test = X_test.reshape(X_test.shape[0], 28*28).astype("float32")
                                                                      44 # 解壓縮圖片
14 # 因為是固定範圍, 所以執行正規化, 從 0-255 至 0-1
                                                                      45 decoded imgs = decoder.predict(encoded imgs)
15 X train = X train / 255
                                                                      46 # 顯示原始, 壓縮和還原圖片
16 X_{test} = X_{test} / 255
                                                                      47 import matplotlib.pyplot as plt
17 # 定義 autoencoder 模型
                                                                      48
18 input img = Input(shape=(784,))
19 x = Dense(128, activation="relu") (input img)
                                                                      49 n = 10 # 顯示幾個數字
20 encoded = Dense(64, activation="relu")(x)
                                                                      50 plt. figure (figsize=(20, 6))
21 x = Dense(128, activation="relu") (encoded)
                                                                      51 for i in range(n):
22 decoded = Dense(784, activation="sigmoid")(x)
                                                                               # 原始圖片
                                                                      52
23 autoencoder = Model (input img, decoded)
                                                                               ax = plt. subplot(3, n, i + 1)
                                                                      53
24 autoencoder. summary()
                          # 顯示模型摘要資訊
                                                                               ax.imshow(X_test[i].reshape(28, 28), cmap="gray")
25 # 定義 encoder 模型
                                                                      54
26 encoder = Model(input img, encoded)
                                                                               ax. axis ("off")
                          # 顯示模型摘要資訊
27 encoder. summary()
                                                                               # 壓縮圖片
                                                                      56
28 # 定義 decoder 模型
                                                                               ax = plt. subplot(3, n, i + 1 + n)
                                                                      57
29 decoder input = Input(shape=(64,))
                                                                               ax. imshow(encoded_imgs[i].reshape(8, 8), cmap="gray")
                                                                      58
30 decoder_layer = autoencoder.layers[-2](decoder_input)
                                                                      59
                                                                               ax. axis ("off")
31 decoder layer = autoencoder.layers[-1] (decoder layer)
                                                                               # 還原圖片
32 decoder = Model (decoder input, decoder layer)
                                                                      60
                          # 顯示模型摘要資訊
33 decoder. summary()
                                                                               ax = plt. subplot(3, n, i + 1 + 2*n)
                                                                      61
34 # 編譯模型
                                                                      62
                                                                               ax. imshow(decoded imgs[i].reshape(28, 28), cmap="gray")
35 autoencoder.compile(loss="binary_crossentropy", optimizer="adam",
                                                                               ax.axis("off")
                                                                      63
                                         metrics=["accuracy"])
36
                                                                      64 plt. show()
```

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

Model: "model_1"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 784)]	0
dense_3 (Dense)	(None, 128)	100480
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 128)	8320
dense_6 (Dense)	(None, 784)	101136

Total params: 218,192 Trainable params: 218,192 Non-trainable params: 0

Model: "model_2"

Layer (type)	Output Shape	Param #	
input_2 (InputLayer)	[(None, 784)]	0	
dense_3 (Dense)	(None, 128)	100480	
dense_4 (Dense)	(None, 64)	8256	

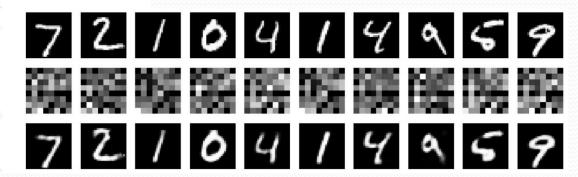
Total params: 108,736 Trainable params: 108,736 Non-trainable params: 0

Model: "model_3"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 64)]	0
dense_5 (Dense)	(None, 128)	8320
dense_6 (Dense)	(None, 784)	101136

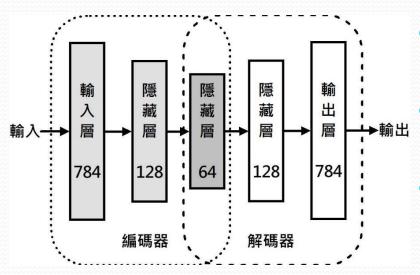
Total params: 109,456 Trainable params: 109,456 Non-trainable params: 0

Epoch 1/10											
235/235 - 4s -	loss:	0.2208	- accuracy:	0.0106	- val_loss:	0.1424	val_accuracy:	0.0135 -	4s/epoch	- 16ms/s	step
Epoch 2/10											
235/235 - 4s -	loss:	0.1256	- accuracy:	0.0113	- val_loss:	0.1120	val_accuracy:	0.0098 -	4s/epoch	- 15ms/s	step
Epoch 3/10											
235/235 - 6s -	loss:	0.1078	- accuracy:	0.0114	<pre>- val_loss:</pre>	0.1019	val_accuracy:	0.0116 -	6s/epoch	- 24ms/s	step
Epoch 4/10											
235/235 - 6s -	loss:	0.0994	- accuracy:	0.0116	<pre>- val_loss:</pre>	0.0951	val_accuracy:	0.0128 -	6s/epoch	- 24ms/s	step
Epoch 5/10											
235/235 - 6s -	loss:	0.0943	- accuracy:	0.0123	<pre>- val_loss:</pre>	0.0911	val_accuracy:	0.0138 -	6s/epoch	- 26ms/s	step
Epoch 6/10											
235/235 - 6s -	loss:	0.0910	- accuracy:	0.0126	<pre>- val_loss:</pre>	0.0886	val_accuracy:	0.0135 -	6s/epoch	- 26ms/s	step
Epoch 7/10											
235/235 - 4s -	loss:	0.0889	- accuracy:	0.0123	<pre>- val_loss:</pre>	0.0869	val_accuracy:	0.0146 -	4s/epoch	- 17ms/s	step
Epoch 8/10											
235/235 - 3s -	loss:	0.0872	- accuracy:	0.0121	<pre>- val_loss:</pre>	0.0854	val_accuracy:	0.0125 -	3s/epoch	- 14ms/s	step
Epoch 9/10											
235/235 - 3s -	loss:	0.0858	- accuracy:	0.0126	<pre>- val_loss:</pre>	0.0842	val_accuracy:	0.0119 -	3s/epoch	- 14ms/s	step
Epoch 10/10											
235/235 - 3s -	loss:	0.0846	- accuracy:	0.0125	<pre>- val_loss:</pre>	0.0832	val_accuracy:	0.0129 -	3s/epoch	- 14ms/s	step



使用MLP建立自編碼器(AE)

Keras並不能使用Sequential模型建立自編碼器,因為需要重組自編碼器的神經層,來建立編碼器和解碼器模型,所以是使用Functional API來建立自編碼器,首先使用MLP建立自編碼器。



- 前半段是編碼器,每一層的神經 元數都比上一層少。
- 後半段是解碼器,每一層的神經 元數都比上一層多。
- 中間的隱藏層為重疊,並且前後 的神經層是對稱

載入資料

```
3 from keras.models import Model
4 from keras.layers import Dense, Input
```

匯入Functional API 的Model和 Input,然後載入資料集。因為是非監督式學習,只需要訓練和測試資料集的特徵資料,不需要標籤資料,故使用『_』變數來代替。

```
9 # 載入資料集
10 (X_train, _), (X_test, _) = mnist.load_data()
```

步驟一:資料預處理

使用MLP打造自編碼器,用來壓縮和解壓縮MNIST手寫辨識資料集的圖片,在載入資料集後,需要執行資料預處理,將特徵資料轉換成28*28=784的向量:

```
11 # 轉換成 28*28 = 784 的向量

12 X_train = X_train.reshape(X_train.shape[0], 28*28).astype("float32")

13 X_test = X_test.reshape(X_test.shape[0], 28*28).astype("float32")
```

因為灰階值是固定範圍0~255,所以執行正規化從0~255轉換成 0~1:

```
      14 # 因為是固定範圍,所以執行正規化,從 0-255 至 0-1

      15 X_train = X_train / 255

      16 X_test = X_test / 255
```

步驟二:定義模型

• 完成資料載入和資料預處理,可以定義自編碼器的神經網路模型,首先定義自編碼器(AE)模型及建立Model物件:

```
17 # 定義 autoencoder 模型
18 input_img = Input(shape=(784,))
19 x = Dense(128, activation="relu")(input_img)
20 encoded = Dense(64, activation="relu")(x)
21 x = Dense(128, activation="relu")(encoded)
22 decoded = Dense(784, activation="sigmoid")(x)
                                                                  對稱
23 autoencoder = Model(input_img, decoded)
24 autoencoder.summary() # 顯示模型摘要資訊
                                     Layer (type)
                                                        Output Shape
                                                                         Param #
                                     input 1 (InputLayer)
                                                        (None, 784)
                                     dense 7 (Dense)
                                                        (None, 128)
                                                                         100480
                                     dense 8 (Dense)
                                                        (None, 64)
                                                                         8256
                                     dense 9 (Dense)
                                                        (None, 128)
                                                                         8320
```

Total params: 218,192 Trainable params: 218,192 Non-trainable params: 0 (None, 784)

dense 10 (Dense)

15

101136

步驟二:定義模型

然後建立編碼器模型,這就是自編碼器(AE)模型的前半段:

```
25 # 定義 encoder 模型
26 encoder = Model(input_img, encoded)
27 encoder.summary() # 顯示模型摘要資訊
```

• 上述程式碼建立Model物件,第1個參數是輸入張量input_img, 第2個參數是輸出張量encoded,其模型摘要資訊如下所示:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 784)	0
dense_7 (Dense)	(None, 128)	100480
dense_8 (Dense)	(None, 64)	8256

Total params: 108,736

Trainable params: 108,736 Non-trainable params: 0

步驟二:定義模型

最後是解碼器模型,除了使用自編碼器(AE)模型的後半段外, 還需要新增Input輸入層(形狀是編碼器模型的輸出層):

```
28 # 定義 decoder 模型
29 decoder_input = Input(shape=(64,))
30 decoder_layer = autoencoder.layers[-2](decoder_input)
31 decoder_layer = autoencoder.layers[-1](decoder_layer)
32 decoder = Model(decoder_input, decoder_layer)
33 decoder.summary() # 顯示模型摘要資訊
```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 64)	0
dense_9 (Dense)	(None, 128)	8320
dense_10 (Dense)	(None, 784)	101136

Total params: 109,456 Trainable params: 109,456 Non-trainable params: 0

步驟三:編譯模型

• 定義好模型後,需要編譯模型來轉換成低階TensorFlow計算圖:

```
34 # 編譯模型
35 autoencoder.compile(loss="binary_crossentropy", optimizer="adam",
36 metrics=["accuracy"])
```

 上述compile()函式的損失函數是binary_crossentropy,優化器是 adam,評估標準是accuracy準確度。

步驟四:訓練模型

在成功編譯模型後,就可以開始訓練模型,fit()中的第一個參數是X_train 訓練資料集,第二個參數也是X_train (標籤資料<自己>),並且使用validation_data參數指定檢驗資料集是測試資料集。Shuffle參數值為True是打亂資料:

```
39 # 訓練模型
40 autoencoder.fit(X_train, X_train, validation_data=(X_test, X_test),
41 epochs=10, batch_size=256, shuffle=True, verbose=2)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
 - 6s - loss: 0.2224 - acc: 0.7899 - val loss: 0.1414 - val acc: 0.8067
Epoch 2/10
- 6s - loss: 0.1253 - acc: 0.8099 - val loss: 0.1113 - val acc: 0.8116
Epoch 3/10
- 6s - loss: 0.1069 - acc: 0.8127 - val loss: 0.1008 - val acc: 0.8125
Epoch 4/10
 - 6s - loss: 0.0995 - acc: 0.8135 - val loss: 0.0957 - val acc: 0.8131
Epoch 5/10
 - 6s - loss: 0.0946 - acc: 0.8140 - val loss: 0.0913 - val acc: 0.8132
Epoch 6/10
 - 6s - loss: 0.0912 - acc: 0.8143 - val loss: 0.0887 - val acc: 0.8134
Epoch 7/10
 - 6s - loss: 0.0888 - acc: 0.8145 - val loss: 0.0868 - val acc: 0.8136
Epoch 8/10
 - 6s - loss: 0.0870 - acc: 0.8146 - val loss: 0.0853 - val acc: 0.8138
Epoch 9/10
 - 6s - loss: 0.0855 - acc: 0.8147 - val loss: 0.0839 - val acc: 0.8138
Epoch 10/10
 - 6s - loss: 0.0842 - acc: 0.8148 - val loss: 0.0828 - val acc: 0.8138 19
```

步驟五:使用自編碼器來編碼和解碼手寫數字圖片

- · 當使用訓練資料集成功訓練模型後,我們可以使用encoder編碼器模型來編碼輸入資料的手寫圖片,也就是壓縮圖片
- 程式碼使用encoder模型的predict()函式壓縮X_test測試資料集的 圖片,可以傳回編碼壓縮後的圖片資料,然後使用decoder解碼 器模型來解壓縮圖片,即解碼圖片:

```
42 # 壓縮圖片

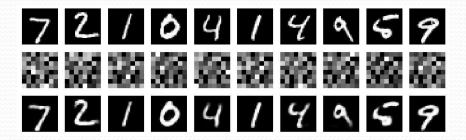
43 encoded_imgs = encoder.predict(X_test)

44 # 解壓縮圖片

45 decoded_imgs = decoder.predict(encoded_imgs)
```

步驟五:使用自編碼器來編碼和解碼手寫數字圖片

```
46 # 顯示原始, 壓縮和還原圖片
47 import matplotlib.pyplot as plt
48
49 n = 10 # 顯示幾個數字
50 plt. figure (figsize=(20, 6))
51 for i in range(n):
        # 原始圖片
52
        ax = plt.subplot(3, n, i + 1)
53
        ax.imshow(X test[i].reshape(28, 28), cmap="gray")
54
    ax.axis("off")
55
       # 壓縮圖片
56
     ax = plt.subplot(3, n, i + 1 + n)
57
        ax.imshow(encoded imgs[i].reshape(8, 8), cmap="gray")
58
   ax.axis("off")
59
  # 還原圖片
60
        ax = plt.subplot(3, n, i + 1 + 2*n)
61
        ax.imshow(decoded imgs[i].reshape(28, 28), cmap="gray")
62
        ax.axis("off")
63
64 plt. show()
```



使用CNN建立自編碼器(CAE)

```
1 import numpy as np
 2 from keras. datasets import mnist
                                                                               37 # 定義 decoder 模型
 3 from keras.models import Model
                                                                               38 decoder_input = Input(shape=(4, 4, 8))
 4 from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
                                                                               39 decoder layer = autoencoder.layers[-7] (decoder input)
 5
                                                                               40 decoder_layer = autoencoder.layers[-6](decoder_layer)
                                                                               41 decoder_layer = autoencoder.layers[-5] (decoder_layer)
 6 # 指定亂數種子
                                                                               42 decoder layer = autoencoder.layers[-4] (decoder layer)
 7 \text{ seed} = 7
                                                                               43 decoder_layer = autoencoder.layers[-3] (decoder_layer)
 8 np. random. seed (seed)
                                                                               44 decoder layer = autoencoder.layers[-2](decoder layer)
 9# 載入資料集
                                                                               45 decoder_layer = autoencoder.layers[-1](decoder_layer)
10 (X_train, _), (X_test, _) = mnist.load_data()
                                                                               46 decoder = Model(decoder_input, decoder_layer)
11 # 轉換成 4D 張量
                                                                               47 decoder.summary()
                                                                                                      # 顯示模型摘要資訊
12 X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")48 # 編譯模型
13 X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype("float32")
                                                                               49 autoencoder.compile(loss="binary_crossentropy", optimizer="adam",
14 # 因為是固定範圍, 所以執行正規化, 從 0-255 至 0-1
                                                                                                                     metrics=["accuracy"])
                                                                               51 # 訓練模型
15 X train = X train / 255
                                                                               52 autoencoder.fit(X train, X train, validation data=(X test, X test),
16 X test = X test / 255
                                                                               53
                                                                                                              epochs=10, batch_size=128, shuffle=True, verbose=2)
17 # 定義 autoencoder 模型
                                                                               54 # 壓縮圖片
18 input img = Input(shape=(28, 28, 1))
                                                                               55 encoded_imgs = encoder.predict(X_test)
19 x = Conv2D(16, (3,3), activation="relu", padding="same")(input img)
                                                                               56 # 解厭縮圖片
20 x = \text{MaxPooling2D}((2,2), \text{padding="same"})(x)
                                                                               57 decoded_imgs = decoder.predict(encoded_imgs)
21 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
                                                                               58 # 顯示原始, 壓縮和還原圖片
22 x = MaxPooling2D((2,2), padding="same")(x)
                                                                               59 import matplotlib.pyplot as plt
23 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
                                                                               60
24 encoded = MaxPooling2D((2,2), padding="same")(x)
                                                                               61 n = 10 # 顯示幾個數字
                                                                               62 plt. figure (figsize=(20, 8))
25 x = Conv2D(8, (3,3), activation="relu", padding="same") (encoded)
                                                                               63 for i in range(n):
26 \times = UpSampling2D((2,2))(x)
                                                                                         # 原始圖片
27 \times = \text{Conv2D}(8, (3,3), \text{activation} = \text{"relu"}, \text{padding} = \text{"same"})(x)
                                                                                         ax = plt.subplot(3, n, i + 1)
                                                                               65
28 x = UpSampling2D((2,2))(x)
                                                                                         ax.imshow(X_test[i].reshape(28, 28), cmap="gray")
                                                                               66
29 x = Conv2D(16, (3,3), activation="relu")(x)
                                                                               67
                                                                                         ax.axis("off")
30 x = UpSampling2D((2,2))(x)
                                                                                         # 壓縮圖片
                                                                               68
31 decoded = Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)
                                                                               69
                                                                                         ax = plt.subplot(3, n, i + 1 + n)
                                                                                         ax.imshow(encoded imgs[i].reshape(4, 4*8).T, cmap="gray")
32 autoencoder = Model(input img, decoded)
                                                                               70
                                                                               71
                                                                                         ax. axis ("off")
33 autoencoder.summary()
                               # 顯示模型摘要資訊
                                                                                         # 還原圖片
                                                                               72
34 # 定義 encoder 模型
                                                                                         ax = plt.subplot(3, n, i + 1 + 2*n)
                                                                               73
35 encoder = Model(input img, encoded)
                                                                               74
                                                                                         ax.imshow(decoded imgs[i].reshape(28, 28), cmap="gray")
36 encoder.summarv()
                         # 顯示模型摘要資訊
                                                                               75
                                                                                         ax. axis ("off")
                                                                               76 plt. show()
```

使用CNN建立自編碼器(CAE)

- 使用CNN建立自編碼器,簡稱CAE,CNN自編碼器的結構:
 - 前半段編碼器:3組Conv2D和MaxPooling2D神經層。
 - 後半段解碼器:3組Conv2D和UpSampling2D神經層。
- 上述MaxPooling2D最大池化層會壓縮圖片,UpSampling2D是對 應MaxPooling2D來還原圖片。

```
17 # 定義 autoencoder 模型
18 input img = Input(shape=(28, 28, 1))
19 x = Conv2D(16, (3,3), activation="relu", padding="same")(input_img)
20 x = MaxPooling2D((2,2), padding="same")(x)
21 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
22 x = \text{MaxPooling2D}((2, 2), \text{padding="same"})(x)
23 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
24 encoded = MaxPooling2D((2,2), padding="same")(x)
25 x = Conv2D(8, (3,3), activation="relu", padding="same") (encoded)
26 \times = UpSampling2D((2,2))(x)
27 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
28 \times = UpSampling2D((2,2))(x)
29 x = Conv2D(16, (3,3), activation="relu")(x)
30 x = UpSampling2D((2,2))(x)
31 decoded = Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)
32 autoencoder = Model(input_img, decoded)
33 autoencoder.summary()
```

使用CNN建立自編碼器(CAE)

- 上面前後神經層的形狀幾乎是對稱,因為池化運算縮小2倍,上 升取樣運算放大2倍 Layer (type) Output Shape Par
- 池化運算: 28/2 → 14/2→ 7/2 → 4
- 上升取樣運算: 4*2→8*2→ 16*2→32
- 最後輸出32 不是原來的28
- 在conv2d_6的 Conv2D層沒有使用 padding="same"參數,目的是將尺寸 調整成14,14*2=28, 最後輸出28(14*2)

Layer (type)	Output	Shape	Param #
input_1 (InputLayer)	(None,	28, 28, 1)	0
conv2d_1 (Conv2D)	(None,	28, 28, 16)	160
max_pooling2d_1 (MaxPooling2	(None,	14, 14, 16)	0
conv2d_2 (Conv2D)	(None,	14, 14, 8)	1160
max_pooling2d_2 (MaxPooling2	(None,	7, 7, 8)	0
conv2d_3 (Conv2D)	(None,	7, 7, 8)	584
max_pooling2d_3 (MaxPooling2	(None,	4, 4, 8)	0
conv2d_4 (Conv2D)	(None,	4, 4, 8)	584
up_sampling2d_1 (UpSampling2	(None,	8, 8, 8)	0
conv2d_5 (Conv2D)	(None,	8, 8, 8)	584
up_sampling2d_2 (UpSampling2	(None,	16, 16, 8)	0
conv2d_6 (Conv2D)	(None,	14, 14, 16)	1168
up_sampling2d_3 (UpSampling2	(None,	28, 28, 16)	0
conv2d_7 (Conv2D)	(None,	28, 28, 1)	145
Total params: 4,385 Trainable params: 4,385 Non-trainable params: 0			

· 然後建立編碼器模型,這就是自編碼器(AE)模型的前半段:

```
34 # 定義 encoder 模型
35 encoder = Model(input_img, encoded)
36 encoder.summary() # 顯示模型摘要資訊
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 28, 28, 1)	0
conv2d_1 (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 16)	0
conv2d_2 (Conv2D)	(None, 14, 14, 8)	1160
max_pooling2d_2 (MaxPooling2	(None, 7, 7, 8)	0
conv2d_3 (Conv2D)	(None, 7, 7, 8)	584
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 8)	0

Total params: 1,904 Trainable params: 1,904 Non-trainable params: 0

 首先新增解碼器模型的Input輸入層,然後使用Model物件的 layers屬性取出autoencoder模型最後7層神經層,就可以建立解碼 器的Model模型,其模型摘要資訊如下所示:

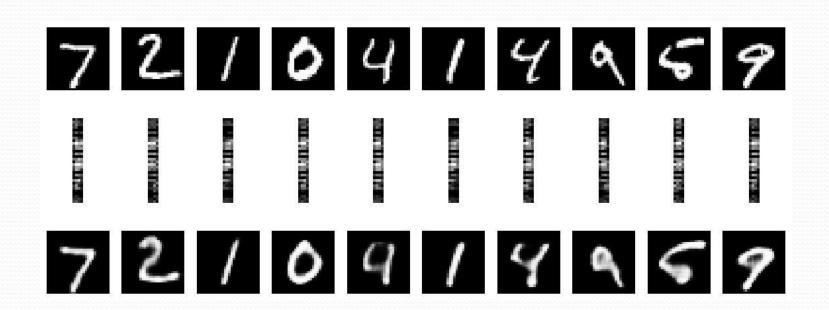
```
37 # 定義 decoder 模型
38 decoder_input = Input(shape=(4,4,8))
39 decoder_layer = autoencoder.layers[-7](decoder_input)
40 decoder_layer = autoencoder.layers[-6](decoder_layer)
41 decoder_layer = autoencoder.layers[-5](decoder_layer)
42 decoder_layer = autoencoder.layers[-4](decoder_layer)
43 decoder_layer = autoencoder.layers[-3](decoder_layer)
44 decoder_layer = autoencoder.layers[-2](decoder_layer)
45 decoder_layer = autoencoder.layers[-1](decoder_layer)
46 decoder = Model(decoder_input, decoder_layer)
47 decoder.summary() # 顯示模型摘要資訊
```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 4, 4, 8)	0
conv2d_4 (Conv2D)	(None, 4, 4, 8)	584
up_sampling2d_1 (UpSampling2	(None, 8, 8, 8)	0
conv2d_5 (Conv2D)	(None, 8, 8, 8)	584
up_sampling2d_2 (UpSampling2	(None, 16, 16, 8)	0
conv2d_6 (Conv2D)	(None, 14, 14, 16)	1168
up_sampling2d_3 (UpSampling2	(None, 28, 28, 16)	0
conv2d_7 (Conv2D)	(None, 28, 28, 1)	145

27

Trainable params: 2,481 Non-trainable params: 0

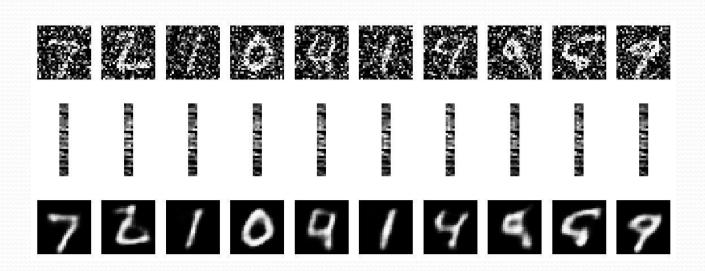
當使用訓練資料集成功訓練模型後,我們可以使用自編碼器來編碼和解碼手寫數字圖片和使用Matplotlib繪出前10張原始圖片、壓縮圖片和最後的還原圖片:



使用CNN自編碼器去除圖片雜訊

```
1 import numpy as np
 2 from keras.datasets import mnist
 3 from keras. models import Model
                                                                             44 # 定義 encoder 模型
 4 from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
                                                                             45 encoder = Model(input_img, encoded)
                                                                             46 encoder. summary()
                                                                                                       # 顯示模型縮要资訊
6 # 指定亂數種子
                                                                             47 # 定義 decoder 模型
7 \text{ seed} = 7
                                                                             48 decoder input = Input(shape=(4, 4, 8))
                                                                             49 decoder layer = autoencoder.layers[-7](decoder input)
8 np. random. seed (seed)
                                                                             50 decoder layer = autoencoder.layers[-6](decoder layer)
9 # 載入資料集
                                                                             51 decoder layer = autoencoder.layers[-5](decoder layer)
10 (X_train, _), (X_test, _) = mnist.load_data()
                                                                             52 decoder_layer = autoencoder.layers[-4](decoder_layer)
11 # 轉換成 4D 張量
                                                                             53 decoder_layer = autoencoder.layers[-3](decoder_layer)
12 X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")
                                                                             54 decoder_layer = autoencoder.layers[-2](decoder_layer)
13 X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype("float32")
                                                                             55 decoder_layer = autoencoder.layers[-1](decoder_layer)
14 # 因為是固定範圍, 所以執行正規化, 從 0-255 至 0-1
                                                                             56 decoder = Model(decoder input, decoder layer)
15 K_train = K_train / 255
                                                                             57 decoder. summary()
                                                                                                     # 顯示模型摘要资訊
16 X test = X test / 255
                                                                             58 # 編譯模型
17 # 替圖片製造雜訊
                                                                             59 autoencoder.compile(loss="binary_crossentropy", optimizer="adam",
18 \text{ nf} = 0.5
                                                                                                                   metrics=["accuracy"])
                                                                             60
19 size train = X train. shape
                                                                             61 # 訓練模型
                                                                             62 autoencoder.fit(X train noisy, X train,
20 X_train_noisy = X_train+nf*np.random.normal(loc=0.0,
                                                                                                            validation data=(X test noisy, X test),
                                                   scale=1.0, size=size train)
22 X_train_noisy = np.clip(X_train_noisy, 0., 1.)
                                                                            64
                                                                                                            epochs=10, batch size=128, shuffle=True, verbose=2)
                                                                             65 # 壓縮圖片
23 size_test = X_test.shape
                                                                             66 encoded_imgs = encoder.predict(X_test_noisy)
24 X_test_noisy = X_test+nf*np.random.normal(loc=0.0,
                                                                             67# 解壓縮圖片
                                                   scale=1.0, size=size_test)
                                                                             68 decoded_imgs = decoder.predict(encoded_imgs)
26 X_test_noisy = np.clip(X_test_noisy, 0., 1.)
                                                                             69 # 顯示雜訊圖片,壓縮圖片和還原圖片
27 # 定義 autoencoder 模型
                                                                             70 import matplotlib.pyplot as plt
28 input img = Input(shape=(28, 28, 1))
                                                                             71
29 x = Conv2D(16, (3,3), activation="relu", padding="same")(input_img)
                                                                             72 n = 10 # 顯示幾個數字
30 x = \text{MaxPooling2D}((2,2), \text{padding="same"})(x)
                                                                             73 plt.figure(figsize=(20, 8))
31 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
                                                                             74 for i in range(n):
32 \times = MaxPooling2D((2,2), padding="same")(x)
                                                                             75
                                                                                       # 雜訊圖片
                                                                             76
                                                                                       ax = plt.subplot(3, n, i + 1)
33 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
                                                                             77
                                                                                       ax.imshow(X_test_noisy[i].reshape(28, 28), cmap="gray")
34 encoded = MaxPooling2D((2,2), padding="same")(x)
                                                                             78
                                                                                       ax.axis("off")
35 x = Conv2D(8, (3,3), activation="relu", padding="same")(encoded)
                                                                             79
                                                                                       # 壓縮圖片
36 \times = UpSampling2D((2,2))(x)
                                                                             80
                                                                                       ax = plt.subplot(3, n, i + 1 + n)
37 x = Conv2D(8, (3,3), activation="relu", padding="same")(x)
                                                                             81
                                                                                       ax.imshow(encoded_imgs[i].reshape(4, 4*8).T, cmap="gray")
38 x = UpSampling2D((2,2))(x)
                                                                             82
                                                                                       ax.axis("off")
39 \times = \text{Conv2D}(16, (3,3), \text{activation="relu"})(x)
                                                                             83
                                                                                       # 還原圖片
40 x = UpSampling2D((2,2))(x)
                                                                             84
                                                                                       ax = plt.subplot(3, n, i + 1 + 2*n)
41 decoded = Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)
                                                                             85
                                                                                       ax.imshow(decoded_imgs[i].reshape(28, 28), cmap="gray")
42 autoencoder = Model(input_img, decoded)
                                                                                       ax.axis("off")
43 autoencoder.summary()
                           # 顯示模型摘要資訊
                                                                             87 plt. show()
```

• CNN自編碼器只需使用有雜訊的圖片來進行訓練,就可以用來去除圖片上的雜訊,只是改用有雜訊圖片來進行訓練,以便使用CNN自編碼器來去除圖片上的雜訊。



使用CNN自編碼器去除圖片雜訊

```
17 # 替圖片製造雜訊
18 nf = 0.5
19 size_train = X_train.shape
20 X_train_noisy = X_train+nf*np.random.normal(loc=0.0,
21 scale=1.0, size=size_train)
22 X_train_noisy = np.clip(X_train_noisy, 0., 1.)
23 size_test = X_test.shape
24 X_test_noisy = X_test+nf*np.random.normal(loc=0.0,
25 scale=1.0, size=size_test)
26 X_test_noisy = np.clip(X_test_noisy, 0., 1.)
```

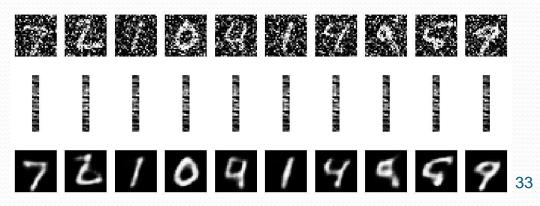
使用CNN自編碼器去除圖片雜訊

• Python程式改用X_train_noisy有雜訊圖片來訓練模型:

```
61#訓練模型
62 autoencoder.fit(X_train_noisy, X_train,
63 validation_data=(X_test_noisy, X_test),
64 epochs=10, batch_size=128, shuffle=True, verbose=2)
```

上述fit()函式的第1個參數是X_train_noisy,在完成訓練後,我們可以壓縮圖片和解壓縮圖片:

```
65 # 壓縮圖片
66 encoded_imgs = encoder.predict(X_test_noisy)
67 # 解壓縮圖片
68 decoded_imgs = decoder.predict(encoded_imgs)
```



End!