

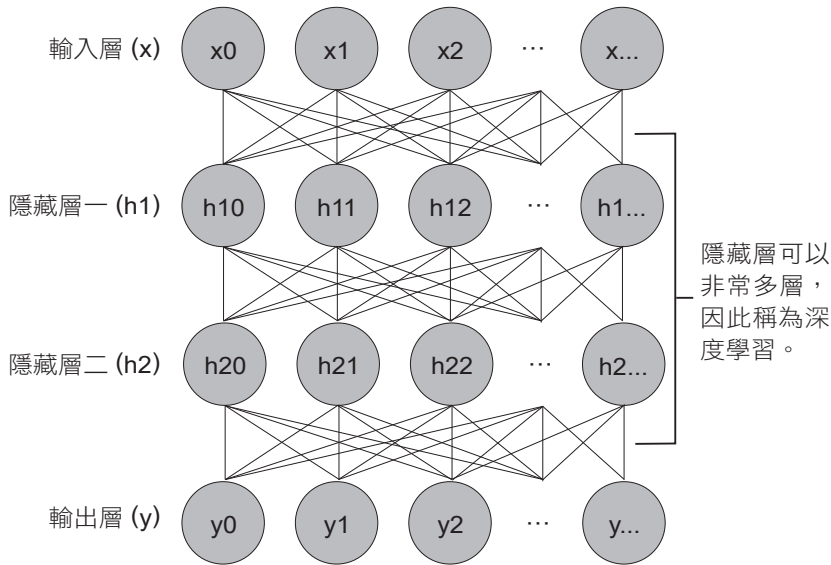


1.3 什麼是深度學習？

簡單的說，深度學習就是透過各種神經網路，如多層感知器 (MLP)、卷積神經網路 (CNN)、循環神經網路 (RNN) 等，將一大堆的數據輸入神經網路中，讓電腦透過大量數據的訓練找出規律並自動學習，最後讓電腦能依據自動學習累積的經驗作出預測。

深度學習利用電腦模擬人類的神經網路，並將神經網路分成多個層，一般會有 1 個輸入層、隱藏層和 1 個輸出層，因為隱藏層可以是 1 層，也可以非常多層，因此稱為深度學習。

每一層中可以包括許多的神經元，當然也可以只有 1 個神經元。



▲ 深度學習基本結構

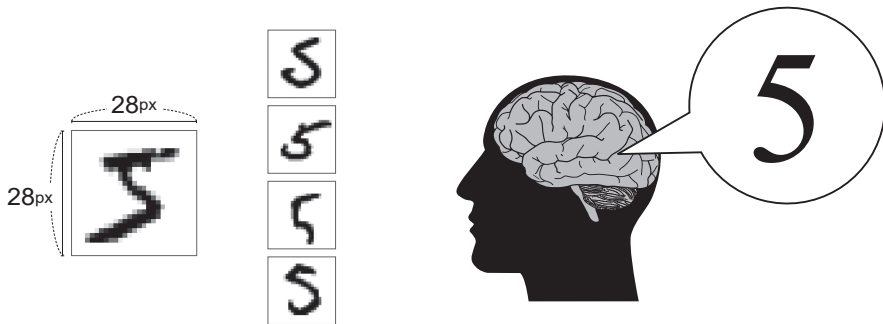
深度學習之所以厲害就在於它堆疊了很多層，但神經網路並不一定是越多層就越好，有時候太多層反而會得到反效果。



2.1 認識多層感知器 (MLP)

2.1.1 認識神經網路

這是一個長寬各 28 像素的手寫數字圖片，但是人類的大腦很神奇，可以很輕鬆就辨識出這個數字來。而且當換上代表相同數字的其他手寫圖片，即使內容前後組成有所差異，也都能被輕易地辨識出來。

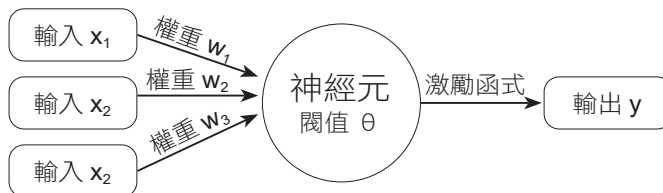


不過當希望寫個程式來重現這樣的能力，這在瞬間就會變成非常困難的任務！神經網路的加入是讓機器學習解決這個問題的很好途徑，但你了解它是怎麼運作的嗎？這裡將用簡單方式說明神經網路，了解它完整的模樣。

神經元的運作

神經網路 (Neural Network) 一如其名，是由人類的大腦神經結構的運作借鏡而來，在機器學習的世界中，神經元就像是大腦的神經細胞，是神經網路最基礎的結構，在它們相互結合下，建構整個龐大的運作網路，實現學習、處理及預測等功能。

神經元是彼此相連的，以下是單獨取出單一神經元的運作模型，每個神經元中都有一個 **閾值**，它的功能是設下一個門檻，如果所接收的訊號值運算後大於這個門檻，神經元就會被觸發，將接收的值經由 **激勵函式** 轉換，輸出到下一個神經元。

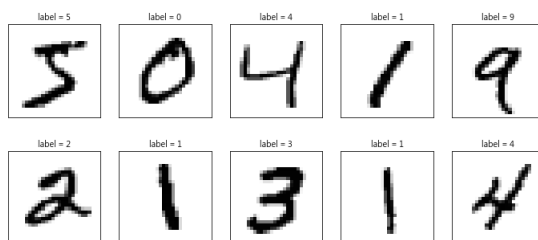


▲ 單一神經元模型：單層感知器

2.2 認識 MNIST 資料集

在電腦螢幕上顯示「Hello World」，是許多程式初學者所學習的第一個範例。在機器學習的領域中，MNIST 資料集也擁有一樣的地位，當你開始接觸相關的資料，無論是學習或是教學，都一定不會陌生。

MNIST 資料集 (Modified National Institute of Standards and Technology database)，是由紐約大學 Yann LeCun 教授蒐集整理許多人 0 到 9 的手寫數字圖片所形成的資料集，其中包含了 60000 筆的訓練資料，10000 筆的測試資料。在 MNIST 資料集中，每一筆資料都是由 images (數字圖片) 和 labels (真實數字) 組成的單色圖片資料，很適合機器學習的初學者，練習建立模型、訓練和預測。



MNIST 資料集的應用範圍很廣，除了進行機器學習的練習，還可以真正使用在生活中，例如用來辨識支票的手寫金額、電話號碼、車牌號碼，甚至改考卷呢！

2.2.1 下載與讀取 MNIST 資料集

本章範例將利用 MNIST 資料集進行機器學習與深度學習，省去收集、整理、格式化資料等繁瑣的工作，將注意力專注在學習上，你也可以藉由這個成熟的資料集磨鍊自己的開發技巧。

下載 MNIST 資料集

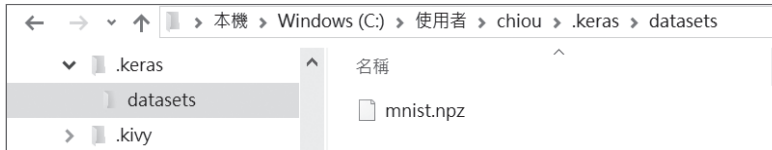
在 Python 中透過 Keras 就可以下載 MNIST 資料集，請先匯入 mnist 模組，再利用 mnist 模組的 load_data 方法，即可載入資料，語法如下：

```
from keras.datasets import mnist
(train_feature, train_label), \
(test_feature, test_label) = mnist.load_data()
```



```
Console I/A x
Downloading data from https://storage.googleapis.com/tensorflow/
tf-keras-datasets/mnist.npz
11493376/11490434 [=====] - 0s 0us/step
```

`mnist.load_data()` 第一次執行會將資料下載到使用者目錄下的 `<.keras\datasets>` 目錄中，檔名為 `<mnist.npz>`。



讀取 MNIST 資料集

第二次以後執行則會先檢查 MNIST 資料集的檔案是否已經存在，如果已經存在就不再重複下載。由於只需要從已經下載檔案中載入資料，因此執行速度會快很多。載入資料後分別放在 `(train_feature, train_label)` 和 `(test_feature, test_label)` 變數中，其中 `(train_feature, train_label)` 是訓練資料，`(test_feature, test_label)` 是測試資料，可以使用 `load_data()` 函式讀入，語法如下：

```
(train_feature, train_label),(test_feature, test_label) = mnist.load_data()
```

2.2.2 查看訓練資料

顯示訓練資料內容

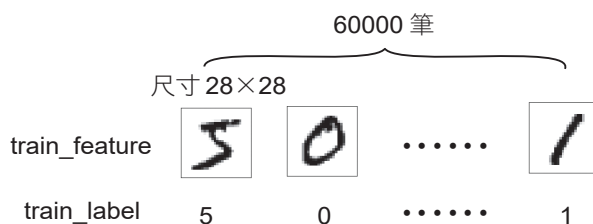
訓練資料是由單色的數字圖片 (`images`) 和數字圖片真實值 (`labels`) 所組成，兩者都是 60000 筆，可以使用 `len()` 函式查看資料的長度：

```
print(len(train_feature),len(train_label)) # 60000 60000
```

每一筆單色的數字圖片是一個 28*28 的圖片檔，真實值則是一個 0~9 的數字。可以使用 `shape` 屬性查看其維度：

```
print(train_feature.shape,train_label.shape)#(60000, 28, 28)(60000,)
```

`shape` 分別為 `(60000, 28, 28)` 和 `(60000,)`，表示有 60000 張 28*28 的數字圖片和 60000 個數字圖片真實值 (又稱標籤)，示意如下：



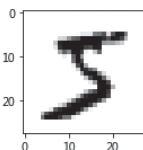
顯示訓練資料的圖片與值

以下利用自訂程序 `show_image` 以白底黑字來顯示 2*2 吋大小的數字圖片，參數 `image` 是指定要顯示的圖片。

```
import matplotlib.pyplot as plt
def show_image(image):
    fig = plt.gcf()
    fig.set_size_inches(2, 2) # 數字圖片大小
    plt.imshow(image, cmap='binary') # 白底黑字顯示
    plt.show()
```

例如：顯示要訓練資料第 1 個數字圖片 `train_feature[0]`，會看到數字 5 的圖形。

```
show_image(train_feature[0])
```



`train_label` 是訓練資料數字圖片的真實值，可以 `print` 直接顯示。例如：顯示訓練資料第 1 個數字圖片真實值的 `train_label[0]`，將會得到數字 5。

```
print(train_label[0])
```

2.2.3 查看多筆訓練資料

如果顯示數字圖片時也同時顯示真實值和預測值，會更方便觀察預測結果是否正確，以下利用自訂程序 `show_images_labels_predictions` 即可達成：

```
def show_images_labels_predictions(images, labels,
                                  predictions, start_id, num=10):
```



```
plt.gcf().set_size_inches(12, 14)
if num>25: num=25
for i in range(num):
    ax=plt.subplot(5,5, i+1)
    # 顯示黑白圖片
    ax.imshow(images[start_id], cmap='binary')

    # 有 AI 預測結果資料，才在標題顯示預測結果
    if( len(predictions) > 0 ) :
        title = 'ai = ' + str(predictions[start_id])
        # 預測正確顯示 (o)，錯誤顯示 (x)
        title += (' (o)' if predictions[start_id]==
                 labels[start_id] else ' (x)')
        title += '\nlabel = ' + str(labels[start_id])
    # 沒有 AI 預測結果資料，只在標題顯示真實數值
    else :
        title = 'label = ' + str(labels[start_id])

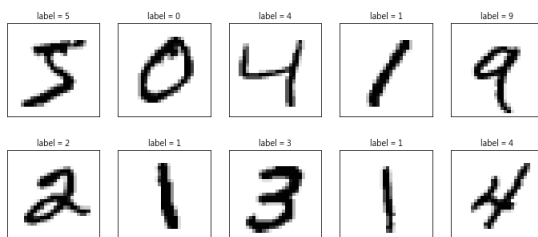
    # X, Y 軸不顯示刻度
    ax.set_title(title,fontsize=12)
    ax.set_xticks([]);ax.set_yticks([])
    start_id+=1

plt.show()
```

- 參數 `images` 是數字圖片，`labels` 是真實值，`predictions` 是預測值。
- `start_id` 是開始顯示的索引編號，`num` 是要顯示的圖片個數，最多可顯示 25 張，預設為 10 張。
- 如果有傳入預測值就會在標題上顯示預測值、預測是否正確、真實值，否則只會顯示真實值。

例如：顯示訓練資料前 10 筆資料。

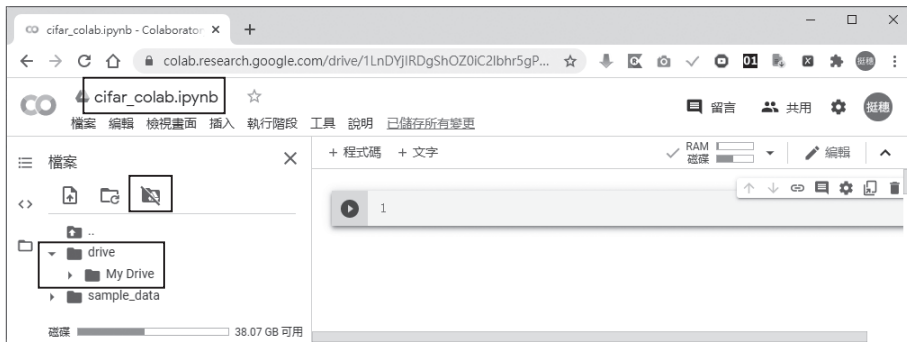
```
show_images_labels_predictions(train_feature,train_label,[],0,10)
```



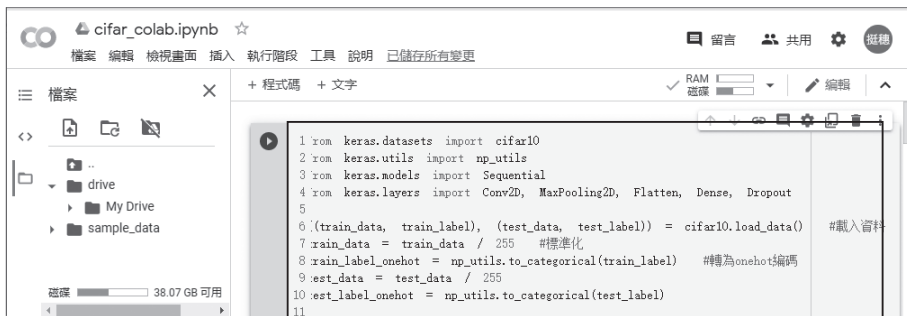
5.2.2 Colab 中訓練模型

在 Colab 中新建筆記本時會分配到一個全新的虛擬機器，本節範例所需的模組如 tensorflow、keras、numpy 等，系統都已預先裝好，不必自行安裝，直接引用就可以了！

在 Colab 建立一個新筆記本來訓練模型辨識 CIFAR-10 資料集的圖片：將新筆記本命名為「cifar_colab.ipynb」，設定使用 GPU，連接 Google Drive 雲端硬碟。



新筆記本預設自行產生一個程式儲存格，在儲存格輸入下面訓練 CIFAR-10 資料集的程式碼：以 CNN 卷積神經網路訓練。



```

1 from keras.datasets import cifar10
2 from keras.utils import np_utils
3 from keras.models import Sequential
4 from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
5
6 ((train_data, train_label), (test_data, test_label)) =
   cifar10.load_data() # 載入資料
7 train_data = train_data / 255 # 標準化
8 train_label_onehot = np_utils.to_categorical(train_label) # 轉為 onehot 編碼

```





```
9 test_data = test_data / 255
10 test_label_onehot = np_utils.to_categorical(test_label)
11
12 dropvalue = 0.3
13 model = Sequential()
14 model.add(Conv2D(input_shape=(32, 32, 3), filters=32, kernel_
    size=(5, 5), padding='same', activation='relu')) # 加入卷積層
15 model.add(MaxPooling2D(pool_size=(2, 2))) # 加入池化層
16 model.add(Dropout(dropvalue)) # 加入 Dropout 層
17 model.add(Conv2D(filters=64, kernel_size=(5, 5),
    padding='same', activation='relu'))
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 model.add(Dropout(dropvalue))
20 model.add(Flatten())
21 model.add(Dense(units=256, kernel_initializer='normal',
    activation='relu'))
22 model.add(Dropout(dropvalue))
23 model.add(Dense(units=10, kernel_initializer='normal',
    activation='softmax')) # 輸出層
24
25 # 訓練
26 model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
27 model.fit(x=train_data, y=train_label_onehot, validation_
    split=0.2, epochs=50, batch_size=500, verbose=2)
28
29 # 用測試資料評估 準確率
30 evalu = model.evaluate(test_data, test_label_onehot)
31 print('測試資料正確率:', evalu[1])
32
33 model.save('/content/drive/My Drive/Colab Notebooks/
    cifarModel.h5') # 儲存模型
```

程式說明

- 1-4 含入模組。
- 6 載入 CIFAR-10 資料集。第二章在本機訓練模型時，只有在第一次執行載入 MNIST 資料集時會下載資料檔案並將其儲存於本機，以後就不必下載，由本機載入資料集即可。而每次開啟 Colab 虛擬機器或虛擬機器重新連線就會取得一個全新的虛擬機器，前面下載的 CIFAR-10 資料集檔案會被清除，所以必須重新下載資料集檔案。
- 7 將訓練資料標準化（介於 0 與 1 之間）。

- 8 將訓練資料轉為 onehot 編碼。
- 9-10 將測試資料標準化及轉為 onehot 編碼。
- 12 設定 Dropout 層捨棄資料的比例。
- 13 以 Sequential 方式建立模型。
- 14 加入卷積層。
- 15 加入池化層。
- 16 加入 Dropout 層。
- 17-19 加入第二個卷積層、池化層及 Dropout 層。
- 20 加入平坦層，將資料轉為一維。
- 21-22 加入隱藏層 Dropout 層。
- 23 加入輸出層。
- 26 以 compile 方法定義 Loss 損失函式、Optimizer 最佳化方法和 metrics 評估準確率方法。
- 27 進行模型訓練。
- 30-31 使用 CIFAR-10 資料集的 10000 筆資料進行預測，並列印測試資料準確率。
- 33 儲存模型檔。注意檔案路徑為 Google 雲端硬碟。

按程式儲存格左方的  圖示執行程式：執行結果顯示大約有 75% 正確率，同時由左方檔案總管可見到已產生 < cifarModel.h5 > 模型檔。



The screenshot shows the Google Colab interface for a notebook named 'cifar_colab.ipynb'. The left sidebar displays a file manager with a folder named 'drive' containing subfolders for 'My Drive', 'AndroidStudio部落格', and 'Appinventor部落格'. Below this, a file named 'cifarModel.h5' is highlighted. The main area shows the execution of a code cell, with the output displaying training progress for epochs 40/50 to 50/50. The final output line shows: '測試資料正確率: 0.744000176429749'.

```

Epoch 40/50
80/80 - 2s - loss: 0.3478 - accuracy: 0.8746 - val_loss: 0.7865 - val_accuracy:
Epoch 41/50
80/80 - 2s - loss: 0.3380 - accuracy: 0.8808 - val_loss: 0.7925 - val_accuracy:
Epoch 42/50
80/80 - 2s - loss: 0.3345 - accuracy: 0.8800 - val_loss: 0.7923 - val_accuracy:
Epoch 43/50
80/80 - 2s - loss: 0.3303 - accuracy: 0.8833 - val_loss: 0.7818 - val_accuracy:
Epoch 44/50
80/80 - 2s - loss: 0.3166 - accuracy: 0.8869 - val_loss: 0.7942 - val_accuracy:
Epoch 45/50
80/80 - 2s - loss: 0.3063 - accuracy: 0.8910 - val_loss: 0.8077 - val_accuracy:
Epoch 49/50
80/80 - 2s - loss: 0.2876 - accuracy: 0.9017 - val_loss: 0.8141 - val_accuracy:
Epoch 50/50
80/80 - 2s - loss: 0.2752 - accuracy: 0.9017 - val_loss: 0.8141 - val_accuracy:
243/243 [====] - 1s 3ms/step - loss: 0.8566 - accuracy:
測試資料正確率: 0.744000176429749

```



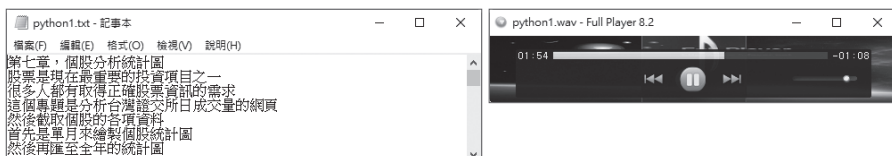
9.1 專題方向

YouTube 是目前全世界最大的影音平台，將個人影片上傳到 YouTube 已成為許多人日常生活的一部分，因此產生了製作 YouTube 影片字幕的需求。製作影片字幕涉及語音辨識 (語音轉文字) 及建立影片時間軸，本專題以語音辨識模組 (SpeechRecognition 模組) 將語音轉換為文字，再利用字幕製作軟體 (Aegisub) 建立影片時間軸將文字加入影片中。

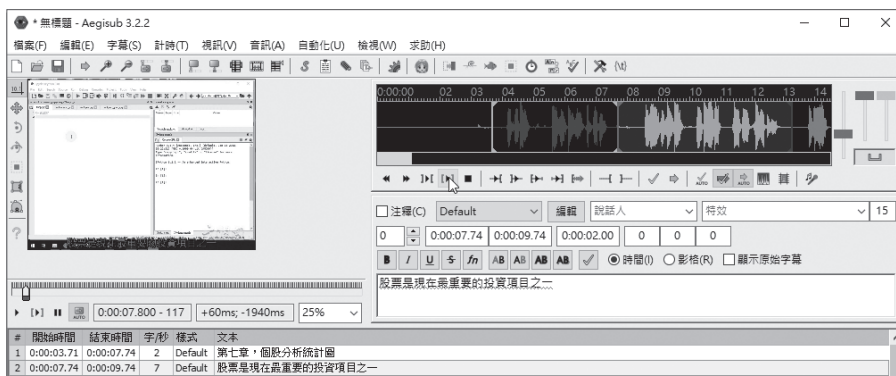
專題檢視

首先將影片上傳到 YouTube，然後由 YouTube 下載影片的語音檔，下載的語音檔為 MP3 格式，語音辨識模組只能辨識 WAV 格式語音檔，所以需將下載的語音檔轉換為 WAV 格式。

接著將語音檔分割為長度 30 秒的小語音檔，讓語音辨識模組轉換為文字，再將所有辨識文字結合成一個文字檔。由於部分辨識結果可能有錯誤，必須播放語音檔進行影片文字的修正。



開啟 Aegisub 字幕製作軟體，根據影片時間軸逐句加入影片文字，就可完成影片字幕製作，最後將字幕上傳到 YouTube 即可。



9.2 語音辨識

語音辨識（**speech recognition**）技術，其功能是使用電腦自動將人類的語音內容轉換為相應的文字。

語音辨識技術的應用包括語音撥號、語音導航、室內裝置控制、簡單的聽寫資料等。目前語音辨識的正確率已達到相當不錯的水準，可為多數人接受。

9.2.1 SpeechRecognition 模組

要在程式中進行語音辨識，必須先安裝 **SpeechRecognition** 模組：開啟 **Anaconda Prompt** 命令視窗，輸入下列語法安裝 **SpeechRecognition** 模組。

```
pip install SpeechRecognition==3.8.1
```

程式中首先要含入 **SpeechRecognition** 模組：

```
import speech_recognition as sr
```

接著建立 **SpeechRecognition** 物件，語法為：

```
SpeechRecognition 物件變數 = sr.Recognizer()
```

例如建立變數名稱為「r」的 **SpeechRecognition** 物件：

```
r = sr.Recognizer()
```

SpeechRecognition 物件的 **record** 方法可以讀取語音檔，語法為：

```
with sr.WavFile( 語音檔路徑 ) as 檔案變數：  
    語音變數 = SpeechRecognition 物件變數 .record( 檔案變數 )
```

例如語音檔路徑為 <record1.wav>，語音變數為 **audio**：

```
with sr.WavFile("record1.wav") as source:  
    audio = r.record(source)
```

SpeechRecognition 模組支援的語音檔格式有 **WAV**、**AIFF** 及 **FLAC**。



最後一個步驟就是使用 `SpeechRecognition` 物件的 `recognize_google` 方法進行語音辨識了，語法為：

```
SpeechRecognition 物件變數 .recognize_google( 語音變數 [, language= 語系 ])
```

`SpeechRecognition` 模組預設的辨識語言為英語，若要辨識中文則要加上參數「`language="zh-TW"`」。例如語音變數為 `audio`，以中文辨識的語法為：

```
r.recognize_google(audio, language="zh-TW")
```

下面範例辨識中文語音檔 `<record1.wav>`，請將此語音檔置於 `<voiceReg1.py>` 檔相同的資料夾中。

程式碼：`voiceReg1.py`

```
1 import speech_recognition as sr
2
3 r = sr.Recognizer() # 建立語音辨識物件
4 with sr.WavFile("record1.wav") as source: # 讀取語音檔
5     audio = r.record(source)
6
7 print('開始翻譯...')
8 try:
9     text = r.recognize_google(audio, language="zh-TW") # 辨識結果
10    print(text)
11 except sr.UnknownValueError:
12    print("Google Speech Recognition 無法辨識此語音！")
13 except sr.RequestError as e:
14    print("無法由 Google Speech Recognition 取得結果；{0}".format(e))
15 print('翻譯結束！')
```

程式說明

- 1 含入 `SpeechRecognition` 模組。
- 3 建立 `SpeechRecognition` 物件。
- 4-5 讀取語音檔。
- 9-10 產生辨識文字並顯示。
- 11-14 若產生錯誤則顯示錯誤訊息。

執行結果：



```
IPython console
Console 1/A
In [16]: runfile('D:/AIDL/附書光碟/ch07/voiceReg1.py', wdir='D:/AIDL/附書光碟/ch07')
開始翻譯...
程式設計的學習成時間運算思維教學的重要途徑透過傳寫程式能將運算思維中抽象的運作方式
翻譯結束!
IPython console History log
```

9.2.2 由 YouTube 取得語音檔

現代人的生活已和 YouTube 密不可分，許多人會上傳影片到 YouTube，因此產生了大量製作 YouTube 影片字幕的需求。雖然 YouTube 平台有製作字幕的功能，但其操作介面非常陽春且費時，本章提供半自動製作字幕的方法。

第一步是將影片上傳到 YouTube 平台：

開啟 YouTube 首頁「<https://www.youtube.com/?gl=TW&hl=zh-TW>」，按右上角 **登入** 者圖示，點選 **您的頻道** 登入管理頁面。於管理頁面按 **+** 後點選 **上傳影片**。



按 **↑** 圖示開啟選擇檔案對話方塊，選取本章範例資料夾中 <python1.mp4> 影片檔後按 **開啟** 鈕，等檔案上傳及處理完畢後，填寫 **詳細資訊** 頁面的影片標題及說明，按右下角 **下一步** 鈕繼續。



12.1 專題方向

本專題由於無法取得數以萬計的車牌再進行機器學習的訓練，因此不是對整個車牌進行辨識，而是以前一章建立的車牌號碼偵測模型取得到圖片中車牌號碼的位置後，再將車牌號碼擷取下來，進而分割車牌號碼取得車牌中每一個文字，使用這些車牌號碼文字進行機器學習訓練，建立車牌號碼辨識模型，最後再使用這個模型進行車牌辨識。

專題檢視

首先將車牌圖片的檔案名稱更改為車牌號碼，如此在擷取車牌號碼文字後，可使用檔案名稱文字做為車牌號碼文字的標記。接著撰寫程式 (`cropPlate.py`) 將車牌號碼圖形擷取下來，車牌號碼圖形檔仍使用車牌號碼做為檔案名稱。



使用 `opencv` 的輪廓偵測功能可取得車牌號碼中文字輪廓的位置，再分別將文字擷取出來。車牌號碼文字為大寫英文字母及數字，新式車牌英文字母沒有「O」及「I」，因此有 24 個字母及 10 個數字共計 34 個文字，也就是機器學習時分為 34 類。分別以這 34 個文字建立資料夾，將擷取的车牌號碼文字圖形存入對應的資料夾中。

目前的資料量太少，無法進行機器學習訓練，所以要撰寫程式 (`makedata.py`) 增加資料數量：複製原始圖片，然後在圖片上隨機加入一些雜點，就能產生不同圖片。本專題將各分類圖片擴增到 500 筆左右，全體資料數量有一萬七千多筆。

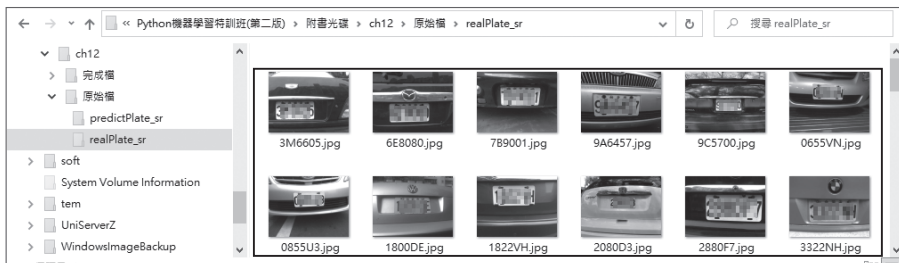
資料準備齊全後就進行機器學習訓練建立模型，然後就可用模型來辨識未知車牌的號碼了！

12.2 車牌號碼機器學習訓練資料

前一章在圖片中偵測車牌號碼位置後，可將車牌號碼擷取下來，進而分別取得車牌號碼每一個文字（英文字母或數字），可以使用這些車牌號碼文字建立機器學習訓練資料，以便進行機器學習訓練建立車牌號碼辨識模型。

12.2.1 原始圖片轉換尺寸

請複製本章範例 <原始檔> 資料夾到硬碟中進行後續操作：<realPlate_sr> 資料夾含有 66 張數位相機拍攝的相片，用於建立機器學習訓練資料；<predictPlate_sr> 資料夾含有 5 張數位相機拍攝的相片，是讓機器學習模型實際測試的預測車牌。為了方便後續判斷車牌號碼辨識是否正確，所有圖片的檔案名稱已更改為車牌號碼，如此只要看檔案名稱即可得知該圖片的車牌號碼。



<haar_carplate.xml> 是前一章建立的車牌號碼 Haar 特徵分類器模型，可偵測圖片中車牌位置。

首先將所有數位相機拍攝的相片尺寸轉換為 300x225 像素圖形，以便讓 <haar_carplate.xml> 模型偵測。轉換尺寸的程式為：

程式碼：resize.py

```
1 def emptydir(dirname): # 清空資料夾
2     if os.path.isdir(dirname): # 資料夾存在就刪除
3         shutil.rmtree(dirname)
4         sleep(2) # 需延遲，否則會出錯
5     os.mkdir(dirname) # 建立資料夾
6
7 def dirResize(src, dst):
8     myfiles = glob.glob(src + '/*.JPG') # 讀取資料夾全部 jpg 檔案
9     emptydir(dst)
10    print(src + ' 資料夾：')
```



```
11 print(' 開始轉換圖形尺寸! ')
12 for f in myfiles:
13     fname = f.split("\\")[ -1]
14     img = Image.open(f)
15     img_new = img.resize((300, 225), PIL.Image.ANTIALIAS) # 尺寸 300x225
16     img_new.save(dst + '/' + fname)
17     print(' 轉換圖形尺寸完成! \n')
18
19 import PIL
20 from PIL import Image
21 import glob
22 import shutil, os
23 from time import sleep
24
25 dirResize('realPlate_sr', 'realPlate')
26 dirResize('predictPlate_sr', 'predictPlate')
```

程式說明

- 1-5 `emptydir` 函式的功能是建立空的資料夾。若資料夾已存在，就先刪除再建立新資料夾。
- 7-17 轉換圖片尺寸函式。
- 8 讀取來源資料夾中所有 `jpg` 圖片檔。
- 9 建立目的資料夾。
- 12-16 逐一將圖片檔案轉換尺寸。
- 25 轉換訓練圖片，轉換後存於 `<realPlate>` 資料夾。
- 26 轉換預測用圖片，轉換後存於 `<predictPlate>` 資料夾。

執行後產生的 `<realPlate>` 及 `<predictPlate>` 資料夾中圖片檔案名稱和原來的 `<realPlate_sr>` 及 `<predictPlate_sr>` 資料夾完全相同，只是所有圖片的尺寸皆轉換為 300x225 像素。

12.2.2 擷取車牌號碼圖形

利用前一章建立的車牌號碼 Haar 特徵分類器模型 (`haar_carplate.xml`)，就可框選出車牌號碼，進而將車牌號碼圖形擷取下來。擷取車牌號碼圖形的程式為：

程式碼：`cropPlate.py`

```
1 def emptydir(dirname): # 清空資料夾
    略.....
```



```
7 import cv2
8 from PIL import Image
9 import glob
10 import shutil, os
11 from time import sleep
12
13 print(' 開始擷取車牌！ ')
14 print(' 無法擷取車牌的圖片：')
15 dstdir = 'cropPlate'
16 myfiles = glob.glob("realPlate\*.JPG")
17 emptydir(dstdir)
18 for imgname in myfiles:
19     filename = (imgname.split('\\'))[-1] # 取得檔案名稱
20     img = cv2.imread(imgname) # 讀入圖形
21     detector = cv2.CascadeClassifier('haar_carplate.xml')
22     signs = detector.detectMultiScale(img, scaleFactor=1.1,
23         minNeighbors=4, minSize=(20, 20)) # 框出車牌
24     # 割取車牌
25     if len(signs) > 0 :
26         for (x, y, w, h) in signs:
27             image1 = Image.open(imgname)
28             image2 = image1.crop((x, y, x+w, y+h)) # 擷取車牌圖形
29             image3 = image2.resize((140, 40),
30                 Image.ANTIALIAS) # 轉換尺寸為 140X40
31             image3.save(dstdir + '/tem.jpg')
32             image4 = cv2.imread(dstdir + '/tem.jpg') # 以 opencv 讀車牌檔
33             img_gray = cv2.cvtColor(image4, cv2.COLOR_RGB2GRAY) # 灰階
34             _, img_thre = cv2.threshold(img_gray, 100, 255,
35                 cv2.THRESH_BINARY) # 黑白
36             cv2.imwrite(dstdir + '/' + filename, img_thre)
37         else:
38             print(filename)
39     os.remove(dstdir + '/tem.jpg') # 移除暫存檔
40 print(' 擷取車牌結束！')
```

程式說明

- 1-5 `emptydir` 函式的功能是建立空的資料夾。若資料夾已存在，就先刪除再建立新資料夾。
- 14 及 35 列印無法擷取車牌的圖片檔案名稱。
- 16 讀取 <realPlate> 資料夾所有 jpg 圖片檔。



- 17 建立空的 <cropPlate> 資料夾儲存擷取的車牌圖片檔案。
- 18-35 逐一處理圖片檔案。
- 19 取得圖片檔案名稱，如「3M6605.jpg」。
- 20 讀取圖片檔案。
- 21 載入車牌號碼 Haar 特徵分類器模型檔。
- 22 偵測車牌號碼。
- 24-33 擷取車牌號碼圖形。
- 25 逐一處理偵測到的車牌（本章圖形皆只有一個車牌）。
- 26 以 Image 模組讀取圖形。
- 27 以 Image 模組的 crop 方法擷取車牌號碼車牌號碼圖形。
- 28 將車牌號碼圖形尺寸轉換為 140x40 像素。
- 29-33 以黑白圖形格式存檔。因擷取的車牌號碼圖形後續要以偵測輪廓方式分割車牌中文字，而偵測輪廓的圖形格式需為黑白圖形，故此處以黑白圖形存檔。
- 29-30 先存檔再以 opencv 模組讀取檔案。
- 31 轉為灰階圖形。
- 32 轉為黑白圖形。
- 33 在 <cropPlate> 資料夾以原來檔名存檔。
- 37 刪除暫存檔案。

執行結果會在 <cropPlate> 資料夾中產生以車牌號碼為檔名的黑白車牌號碼圖。



因為擷取車牌號碼圖形是為了後續分割車牌文字以建立機器學習訓練資料，如果擷取不到車牌號碼的圖片就沒有作用。第 14 及 35 列程式會列出擷取不到車牌號碼圖片的檔案名稱，使用者可將這些檔案由 <realPlate> 資料夾刪除。（若使用本書範例，將不會出現無法擷取車牌號碼圖形的問題。）