CHAPTER

09

電腦視覺

9.1 認識 GPT-4o 的視覺功能

9.1.1 GPT-4o 視覺功能簡介

OpenaAI 公司推出最新的 GPT-4o 版本是目前的旗艦版，和前一版 GPT-4 Turbo 一樣具有視覺功能，不但效率更高而且價格更低。GPT-4o 是 OpenAI 所開發的大型語言模型 (LMM)，該模型同時提供**自然語言處理**和**電腦視覺** (Vision) 解析兩大功能，可分析**圖像** (image) 並以文字回答該圖像的相關問題。GPT-4o 打破語言模型僅專注於文字的限制，實現更高階的電腦視覺圖像解析。GPT-4o 模型可以回答針對圖像中的一般問題，例如圖像中物件的識別、物件的顏色、物件的數量、描述圖像的內容、**光學字元辨識** (OCR)、手寫文字辨識、給予圖像適當的標籤…等，甚至可根據食材的圖像，給予適當的食譜建議。

要使用電腦視覺功能來解析圖像時，可以透過 GPT-4o 的 Chat Completions API 來取得 OpenAI 的視覺服務。雖然目前 gpt-4o 模型仍可能會有一些不足，例如：不能解釋專業醫學影像、相似顏色的識別、圖

9.2.1 電腦視覺功能常用參數

透過 OpenAI API 的 chat.completions 所提供的 create() 方法，可以解析圖像然後用自然語言回答指定的問題。

1. **model**：指定解析圖像的模型，為必要參數。設定值為 'gpt-4o' 指定採用 GPT-4o 版本，雖可指定為舊版，但新版又快又好而且更便宜。
2. **messages**：指定解析圖像時的相關資訊，為必要參數。資料型別為串列，其元素值資料型別為字典。元素值中必要的 鍵/值 說明如下：
   (1) **'role'**：指定角色，值可以為 'user'、'system'、'assistant'，只有設定為 'user' 時才支援載入圖像。
   (2) **'content'**：指定解析圖像時的資訊內容，資料型別為串列，其中的元素值資料型別為字典。其常用 鍵/值 說明如下：
      ① **'type'**：值為 'text' 時是指定型態為文字。
      ② **'text'**：要求模型的文字資訊，例如 'role' 為 'user' 時，就是使用者要求模型處理的相關訊息。

      如果 'type' 鍵的值為 'image_url' 時，其常用 鍵/值 說明如下：
      ① **'type'**：值為 'image_url' 時是指定型態為圖像的網址。支援的圖像格式為 PNG (.png)、JPEG (.jpeg、.jpg)、WEBP (.webp) 和 GIF (.gif，不含動畫)。
      ② **'image_url'**：指定圖檔的相關資訊，資料型別為字典，其常用 鍵/值 說明如下：
         ❶ **'url'**：值為圖像所在的網址或是 base64 編碼的圖像資料。
         ❷ **'detail'**：指定圖檔的解析度，值可以為 'low'、'high' 或 'auto'。預設值為 'auto'，模型會依照圖像大小自動決定採用的解析度。值為 'low' 時，模型將採用低解析度圖像

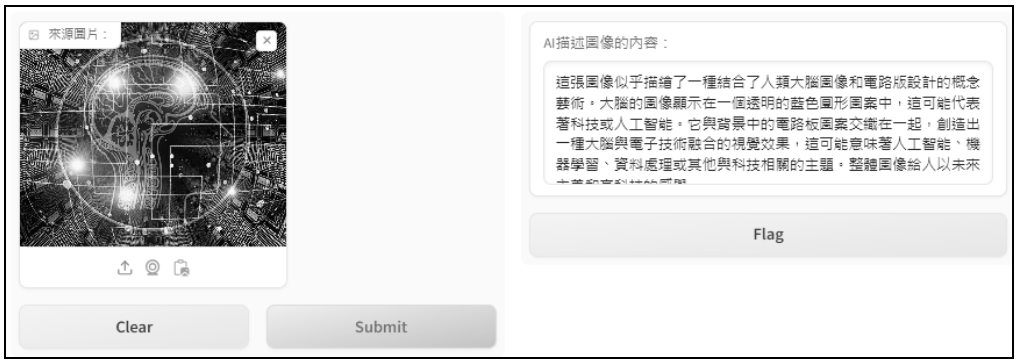
(512×512 像素)，模型處理速度快而且價格也最便宜。值為 'high' 時，模型將採用高解析度圖像。

3. **max_tokens**：指定可使用最多的 tokens 數量。

範例：

顯示一個來自網路的圖像，然後由 OpenAI API 來描述圖像的內容。

執行結果



程式碼 FileName : Vision_1.ipynb

```
1-01 !pip install gradio
1-02 import gradio as gr

2-01 !pip install openai
2-02 import openai

3-01 image_url='https://upload.wikimedia.org/wikipedia/commons/
         thumb/4/42/Artificial-Intelligence.jpg/800px-Artificial-
         Intelligence.jpg'
3-02 def Vision(img):
3-03   openai.api_key = 'OpenAIAPI 金鑰'
3-04   response = openai.chat.completions.create(
3-05     model = 'gpt-4o',
3-06     messages = [
3-07       {
```

```
3-08          'role':'user',
3-09          'content':[
3-10            {'type':'text','text':'圖像中有甚麼?'},
3-11            {'type':'image_url','image_url':{'url':image_url}}
3-12          ]
3-13        }
3-14      ],
3-15      max_tokens = 300
3-16    )
3-17    return response.choices[0].message.content
```

```
4-01 gr.Interface(
4-02    fn = Vision,
4-03    inputs = gr.Image(label='來源圖片：',value=image_url,
                          height=256,width=256),
4-04    outputs = gr.Textbox(label='AI 描述圖像的內容：')
4-05 ).queue().launch()
```

說明

1. 第 3-01 行：宣告 image_url 變數儲存來自維基百科圖像的網址。
2. 第 3-02~3-17 行：定義 Vision() 函式來處理 gradio 介面的元件值，處理過後傳回視覺解析的結果。
3. 第 3-04 行：使用 openai.chat.completions.create() 方法呼叫 Chat 聊天服務，傳回值指定給 response 變數。
4. 第 3-05 行：設 model 參數值為 ' gpt-4o '，指定使用 GPT-4o 模型。
5. 第 3-06~3-14 行：設 messages 參數值指定對 GPT-4o 模型的相關要求。
6. 第 3-08 行：設定角色為使用者 (user)。
7. 第 3-09~3-12 行：設定解析圖像時的資訊內容。
8. 第 3-10 行：設定使用者要求模型處理的相關訊息，提示詞為「圖像中有甚麼?」，請模型描述圖像的內容。
9. 第 3-11 行：以網址方式設定圖像的來源。

10. 第 3-15 行：設定最多的 tokens 為 300。
11. 第 3-17 行：create() 方法的傳回值中，choices 為串列要取得其中 AI 的回應，寫法為：「response.choices[0].message.content」。
12. create() 方法的傳回值 response 完整內容如下：

```
ChatCompletion(id='chatcmpl-9...',
choices=[Choice(finish_reason='stop', index=0, logprobs=None,
message=ChatCompletionMessage(content='這張圖像展示了一個科技主題的圖案，...，給人一種未來科技和人工智慧的感覺。', role='assistant',
function_call=None, tool_calls=None))], created=1715702503,
model='gpt-4o-2024-05-13', object='chat.completion',
system_fingerprint='fp_927397958d',
usage=CompletionUsage(completion_tokens=88, prompt_tokens=779,
total_tokens=867))
```

13. 第 4-01~4-05 行：定義 gradio 的 Interface 物件，指定 Vision() 函式，建立 Image 為輸入元件圖像來源為網址，TextBox 為輸出元件，並發布網站部署。

9.2.2 載入本機圖像

上面範例的圖像來源為網路網址，如果要處理本機的圖像時，就可將圖像用 Base64 編碼格式編碼成為字串後傳遞給模型。Base64 是使用 64 個可列印的字元來表示二進位資料的一種方法，方便在網路傳輸，通常用來處理傳輸、儲存複雜的文字、圖像…等資料。如果需要長時間運行，官網建議採用 URL 傳遞圖像效率比 base64 佳。另外，自行先將圖像縮小到模型預設的尺寸，也可以提高模型的回覆速度。

使用 Base64 編碼格式編碼時，要先載入 base64 套件。讀取圖檔後，可以使用 b64encode() 方法進行 Base64 編碼。程式碼如下：

```
01 import base64
02 img_path = '圖像檔名含路徑'
03 with open(img_path, 'rb') as f:     # 以二進制方式讀取檔案
04   image_data = f.read()             # 讀取圖檔
```

```
# 使用 b64encode() 方法進行 Base64 編碼，成為字串 base64_data
base64_data = base64.b64encode(image_data).decode('utf-8')
```

指定 POST 請求的標頭以符合 OpenAI API 的要求，程式碼如下：

```
headers = {
 "Content-Type": "application/json",    # 設定內容類型為 JSON
 "Authorization": f"Bearer {api_key}"  # 在請求標頭中加入 API 金鑰
}
```

用字典格式指定對模型要求的相關訊息，程式碼如下：

```
payload = {
  "model": "gpt-4o",          # 使用模型名稱
  "messages": [               # 訊息串列
    {
      "role": "user",         # 角色設為使用者
      "content": [            # 內容串列
        {"type": "text","text": "提問內容"},
        {"type": "image_url",
         "image_url": {"url": f"data:image/jpeg;
                       base64,{圖像 base64 字串}"}}
      ]
    }
  ],
  "max_tokens": 300           # 最大生成的 token 數量
}
```

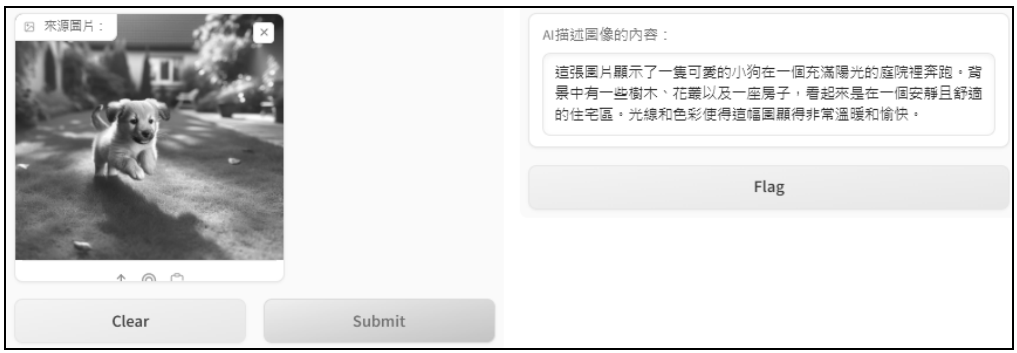
發送 POST 請求到 OpenAI 的 API，其中包含標頭 headers，和對模型要求的字典 payload，模型傳回值指定給 response，程式碼如下：

```
response = requests.post("https://api.openai.com/v1/chat/
           completions",headers=headers,json=payload)
```

範例：

使用者可以將本機中圖檔拖曳到左邊「來源圖片」處，或以檔案總管開啟檔案，然後按下 Submit 鈕，會在右邊顯示 OpenAI API 描述圖像的文字內容。

執行結果



程式碼 FileName : Base64.ipynb

```
1-01 !pip install gradio
1-02 import gradio as gr

2-01 !pip install openai
2-02 import openai

3-01 import base64
3-02 import requests

4-01 def EncodeImage(image_path):
4-02   with open(image_path, "rb") as image_file:
4-03     return base64.b64encode(image_file.read()).decode('utf-8')

5-01 def Vision(image):
5-02   api_key = "OpenAIAPI 金鑰"
5-03   base64_image = EncodeImage(image)
5-04   headers = {
```

```
5-05      "Content-Type": "application/json",
5-06      "Authorization": f"Bearer {api_key}"
5-07    }
5-08    payload = {
5-09      "model":"gpt-4o",
5-10      "messages":[
5-11        {
5-12          "role":"user",
5-13          "content": [
5-14            {"type":"text","text":"圖像中有甚麼?"},
5-15            {"type":"image_url",
                 "image_url":{"url": f"data:image/jpeg;
                              base64,{base64_image}"}}
5-16          ]
5-17        }
5-18      ],
5-19      "max_tokens": 300
5-20    }
5-21    response = requests.post("https://api.openai.com/v1/chat
                  /completions",headers=headers,json=payload)
5-22    dictRes = response.json()            #傳回值轉為字典
5-23    return dictRes['choices'][0]['message']['content']
```

```
6-01 gr.Interface(
6-02    fn = Vision,
6-03    inputs = gr.Image(label='來源圖片：',sources='upload',
                          type='filepath',width=256,height=256),
6-04    outputs = gr.Textbox(label='AI 描述圖像的內容：')
6-05 ).queue().launch()
```

說明

1. 第 3-01~3-02 行：引入 base64 套件用於圖像的編碼和解碼，以及 requests 套件用來發送 HTTP 請求。
2. 第 4-01~4-03 行：定義 EncodeImage() 函式，會以二進制模式打開圖像檔案 image_path，將其編碼為 base64 字串然後回傳。

3. 第 5-01~5-23 行：定義 Vision() 函式來處理 gradio 介面的元件值，處理過後傳回視覺解析的結果。
4. 第 5-14 行：提示詞為「圖像中有甚麼?」，請模型描述圖像的內容。
5. 第 5-22 行：將 API 回應的回傳值 response，使用 json() 方法轉為字典格式 dictRes。response 的完整內容如下：

```
{'id': 'chatcmpl-9...', 'object': 'chat.completion', 'created':
1715699273, 'model': 'gpt-4o-2024-05-13', 'choices': [{'index':
0, 'message': {'role': 'assistant', 'content': '圖像中有一隻小狗
在草地上奔跑，...，環境豔麗而充滿生氣。'}, 'logprobs': None,
'finish_reason': 'stop'}], 'usage': {'prompt_tokens': 779,
'completion_tokens': 60, 'total_tokens': 839},
'system_fingerprint': 'fp_927397958d'}
```

6. 第 5-23 行：傳回字典格式 dictRes 中 AI 回應的 'content' 內容，程式寫法如下：

```
  return dictRes['choices'][0]['message']['content']
```

7. 第 6-01~6-05 行：定義 gradio 的 Interface 物件，指定 Vision() 函式，建立 Image 為輸入元件，TextBox 為輸出元件，並發布網站部署。
8. 第 6-03 行：輸入元件 Image 的 **sources 參數設為 'upload'**，表圖檔來源為上傳 (拖曳或以檔案總管開啟)；**type 參數設為 'filepath'**，表元件值為圖像的檔名 (含路徑)，以便傳給 Vision() 函式讀取圖檔。

9.2.3 解析多張圖像

Chat Completions API 也可以同時接收並處理多個圖像，GPT-4o 模型會處理每張圖像並使用所有圖像的資訊來回答問題。輸入圖像的格式可以採用圖像的 URL 或 Base64 編碼的圖像資料。

在 Chat Completions API 的 create() 方法中同時輸入多個圖像，以用圖像的 URL 為例，程式碼寫法如下：

```
01 response = openai.chat.completions.create(
02   model = "gpt-4o",
```

```
    messages = [
      {
        'role':'user',
        'content':[
          {'type':'text','text':'提示詞'},
          {'type':'image_url','image_url':{'url': '第 1 張圖網址'}},
          {'type':'image_url','image_url':{'url': '第 2 張圖網址'}},
            …
          {'type':'image_url','image_url':{'url': '第 n 張圖網址'}},
        ]
      }
    ],
    max_tokens = 300
)
```

範例：

顯示兩張來自網路的圖像，然後由 OpenAI API 來分別描述圖像的內容以及兩者的差異。

執行結果

程式碼 FileName : Vision_2.ipynb

```
1-01 !pip install gradio
1-02 import gradio as gr

2-01 !pip install openai
2-02 import openai

3-01 image_url1 = 'https://upload.wikimedia.org/wikipedia/commons/
                   thumb/4/47/American_Eskimo_Dog.jpg/
                   200px-American_Eskimo_Dog.jpg'
3-02 image_url2 = 'https://upload.wikimedia.org/wikipedia/commons/
                  thumb/a/a1/Samoyed_600.jpg/200px-Samoyed_600.jpg'
3-03 def Vision(img1,img2):
3-04   openai.api_key = 'OpenAIAPI 金鑰'
3-05   response = openai.chat.completions.create(
3-06     model = "gpt-4o",
3-07     messages = [
3-08       {
3-09         'role':'user',
3-10         'content':[
3-11           {'type':'text','text':'簡介圖像中的動物以及兩者的差異'},
3-12           {'type':'image_url','image_url':{'url':image_url1}},
3-13           {'type':'image_url','image_url':{'url':image_url2}}]
3-14       }
3-15     ],
3-16     max_tokens = 300
3-17   )
3-18   return response.choices[0].message.content

4-01 gr.Interface(
4-02   fn = Vision,
4-03   inputs = [
4-04       gr.Image(label='圖片 1：',value=image_url1,
                    height=256,width=256),
4-05       gr.Image(label='圖片 2：',value=image_url2,
                    height=256,width=256)
4-06   ],
```

範例：

設計一個食物熱量估算程式。使用者使用網路攝影機 (webcam) 拍攝食物到輸入區，按下 Submit 鈕就會說明食物內容以及熱量。

執行結果

程式碼 FileName : Cal.ipynb

```
1-01 !pip install gradio
1-02 import gradio as gr
```

```
2-01 !pip install openai
2-02 import openai
```

```
3-01 import base64
3-02 import requests
3-03 import io

4-01 def SaveImage(pil):
4-02   byte_arr = io.BytesIO()
4-03   pil.save(byte_arr,format='PNG')
4-04   return base64.b64encode(byte_arr.getvalue()).decode('utf-8')

5-01 def Vision(image):
5-02   api_key = "OpenAIAPI 金鑰"
5-03   base64_image = SaveImage(image)
5-04   headers = {
5-05     "Content-Type": "application/json",
5-06     "Authorization": f"Bearer {api_key}"
5-07   }
5-08   payload = {
5-09     "model":"gpt-4o",
5-10     "messages":[
5-11       {
5-12         "role":"system",
5-13         "content":"你是一位專業的營養師"
5-14       },
5-15       {
5-16         "role":"user",
5-17         "content": [
5-18           {"type":"text","text":"以少於 20 字說明圖像中食物，並估計
                      熱量為多少大卡?輸出格式為 1.說明：食物說明,2.熱量：
                      0.0 kcal,圖像非食物時輸出:無法估算"},
5-19           {"type":"image_url",
                "image_url":{"url": f"data:image/jpeg;
                             base64,{base64_image}"}}
5-20         ]
5-21       }
5-22     ],
```

```
5-23      "max_tokens": 300
5-24    }
5-25    response = requests.post("https://api.openai.com/v1/chat
                      /completions",headers=headers,json=payload)
5-26    dictRes = response.json()
5-27    return dictRes['choices'][0]['message']['content']
```

```
6-01 gr.Interface(
6-02    fn = Vision,
6-03    inputs = gr.Image(label='食物：',sources='webcam',type='pil',
                      width=256,height=256),
6-04    outputs = gr.Textbox(label='熱量：'),
6-05    title='食物熱量估算'
6-06 ).queue().launch()
```

說明

1. 第 3-03 行：引入 io 套件。
2. 第 4-01~4-04 行：定義 SaveImage() 函式可以將 PIL 圖像，編碼為 base64 字串回傳。
3. 第 4-02~4-03 行：將 PIL 格式圖像 pil 轉換為二進位資料串流 byte_arr。
4. 第 4-04 行：將二進位資料串流 byte_arr，編碼為 base64 字串後回傳。
5. 第 5-01~5-27 行：定義 Vision() 函式來處理 gradio 介面的元件值，處理過後傳回視覺解析的結果。
6. 第 5-03 行：呼叫 SaveImage() 函式將 Image 元件所傳入的 PIL 圖像 image，轉換為 base64 字串。
7. 第 5-12~5-13 行：指定角色為 "system"，在 content 中要求模型扮演一位專業的營養師。
8. 第 5-18 行：提示詞為「以少於 20 字說明圖像中食物，並估計熱量為多少大卡？輸出格式為 1.說明：食物說明,2.熱量：0.0 kcal，圖像非食物時輸出：無法估算」，請模型依照指定的格式輸出食物說明和熱量。提示詞中使用幾個指定輸出格式的技巧：