AIoT：樹莓派應用

Chapter 6：Tensorflow X 影像辨識

賴秉樑 debugger
學院創辦人

課程網址 https://max543.com/debugger

max 極限翻轉學院
青少年與兒童程式教育

Google MediaPipe 機器學習框架

6-1-1 認識 MediaPipe

6-1-2 MediaPipe 人臉偵測

6-1-3 MediaPipe 臉部網格

6-1-4 MediaPipe 多手勢追蹤

6-1-5 MediaPipe 人體姿態估計

6-1-1 認識 MediaPipe

ex6-1

- MediaPipe 是 Google 公司在 2019 年發表的開放原始碼專案，此專案針對即時串流媒體和電腦視覺 (Computer Vision)，提供開放原始碼且跨平台的機器學習解決方案，這個解決方案就是機器學習管線 (ML Pipeline)。
- 基本上，Google MediaPipe 是一種圖表基礎系統 (Diagram Based System)，可以用來建構多模式影片、聲音和感測器等應用的機器學習管線，我們可以使用圖形方式來組織模組元件，例如：TensorFlow 或 TensorFlow Lite 推論模型和多媒體處理函數等，來建構出一個擁有感知功能的機器學習管線，能夠即時從媒體中辨識出人臉、手勢和姿勢等感知功能，其官方網址如下所示：
  - ✓ https://mediapipe.dev/

在樹莓派安裝 MediaPipe

ex6-2

- 新增一個 Python 3 的虛擬環境 mediapipe，務必先安裝 OpenCV 後 (參見 Ch4)，繼續安裝 Google MediaPipe，因為 MediaPipe 版本的新舊並不相容，請安裝本書使用的 0.8.4.0 版，如下所示：
- 在樹莓派 4 安裝 MediaPipe 的指令：

```
(mediapipe) $ pip install mediapipe-rpi4==0.8.4.0
```

- 在樹莓派 3 安裝 MediaPipe 的指令：

```
(mediapipe) $ pip install mediapipe-rpi3==0.8.4.0
```

6-1-2 MediaPipe 人臉偵測

- MediaPipe 人臉偵測 (Face Detection) 是使用 Blazeface 模型的一種超快速的人臉偵測，Blazeface 模型是 Google 開發的一種快速和輕量級的人臉辨識模型，可以在圖片中辨識出多張人臉和標示臉部 6 個關鍵點 (Key Points)，這是使用 Single Shot Detector 架構和客製化編碼器所建立的人臉辨識模型。
- MediaPipe 人臉偵測所辨識出的臉部可以回傳臉部範圍的方框座標，再加上左眼、右眼、鼻尖、嘴巴、左耳和右耳共 6 個關鍵點座標。

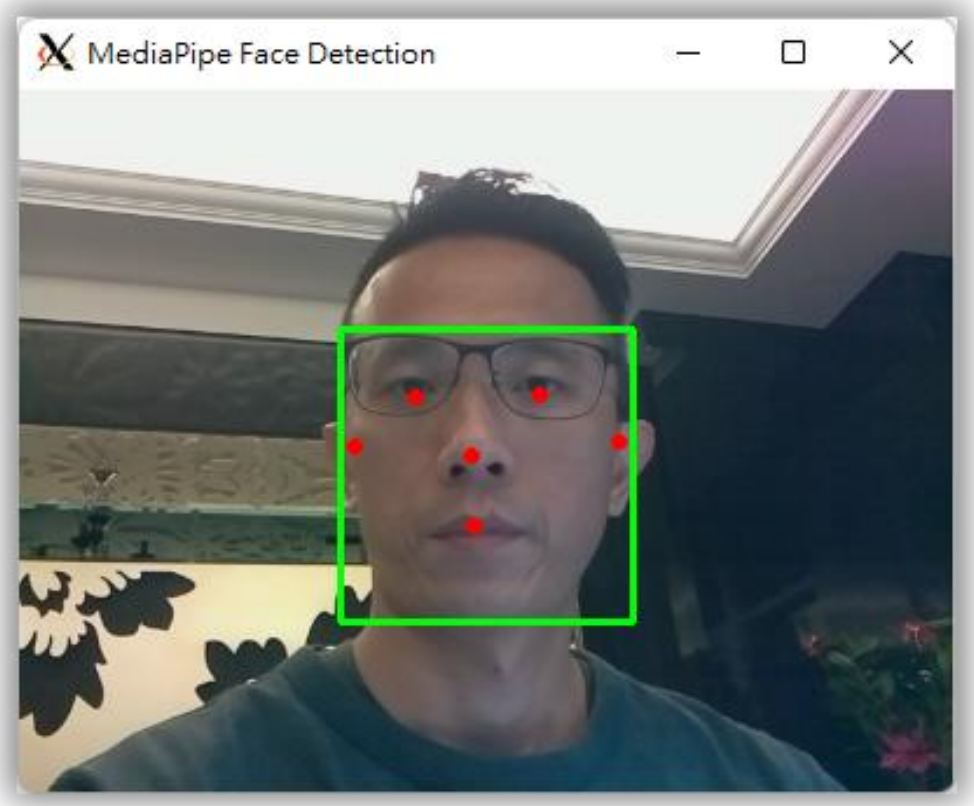
6-1 MediaPipe 人臉偵測與臉部六個關鍵點。(6-1.py)

ex6-3

```
import cv2
import mediapipe as mp

mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

face_detection = mp_face_detection.FaceDetection(min_detection_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()

    if ret:
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
        frame = cv2.rotate(frame, rotateCode = 1)
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame.flags.writeable = False
        results = face_detection.process(frame)
        frame.flags.writeable = True
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        if results.detections:
            for detection in results.detections:
                mp_drawing.draw_detection(frame, detection)

        cv2.imshow("MediaPipe Face Detection", frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-1.py

- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 人臉偵測進行人臉辨識，程式執行的結果可以看到綠色框出的多張人臉和紅色的 6 個關鍵點，如圖所示：



6-1-3 MediaPipe 臉部網格

- MediaPipe 臉部網格 (MediaPipe Face Mesh) 是使用 Blazeface 模型為基礎，可以預測出 468 個關鍵點，和使用網格來繪出 3D 臉部模型。

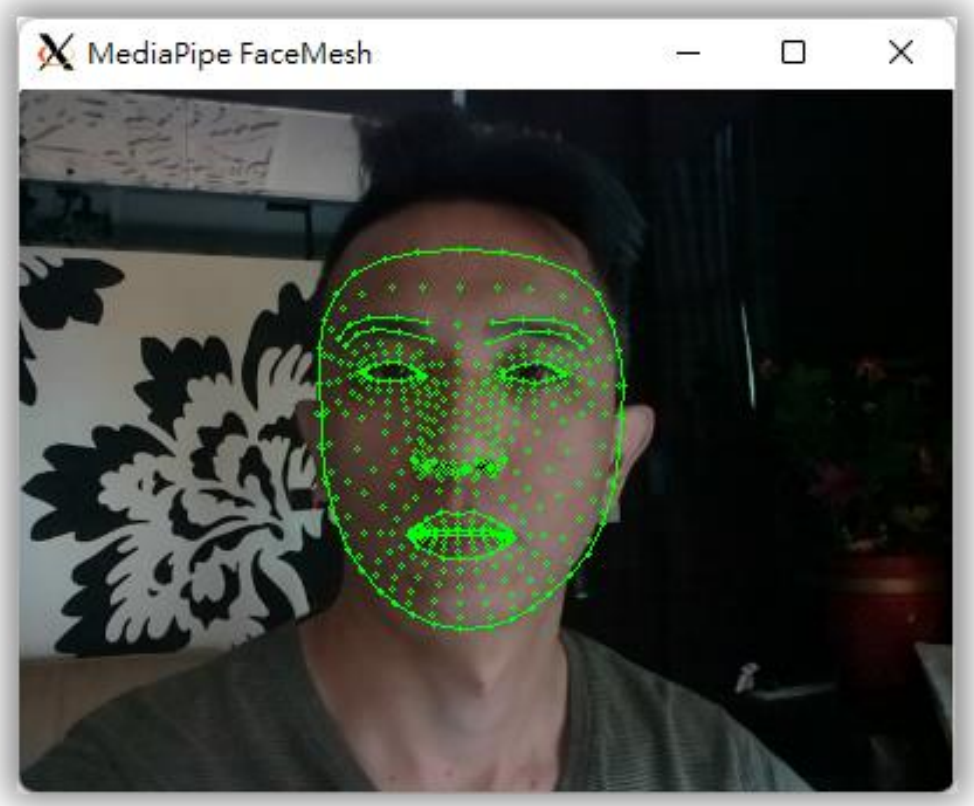
6-2 MediaPipe 臉部網格。(6-2.py)

ex6-4

```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh

drawing_spec = mp_drawing.DrawingSpec(thickness = 1, circle_radius = 1)

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence = 0.5, min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()

    if ret:
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
        frame = cv2.rotate(frame, rotateCode = 1)
        frame = cv2.flip(frame, 1)

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame.flags.writeable = False
        results = face_mesh.process(frame)
        frame.flags.writeable = True
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                mp_drawing.draw_landmarks(image = frame,
                   landmark_list = face_landmarks,
                   connections = mp_face_mesh.FACE_CONNECTIONS,
                   landmark_drawing_spec = drawing_spec,
                   connection_drawing_spec = drawing_spec)

        cv2.imshow("MediaPipe FaceMesh", frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

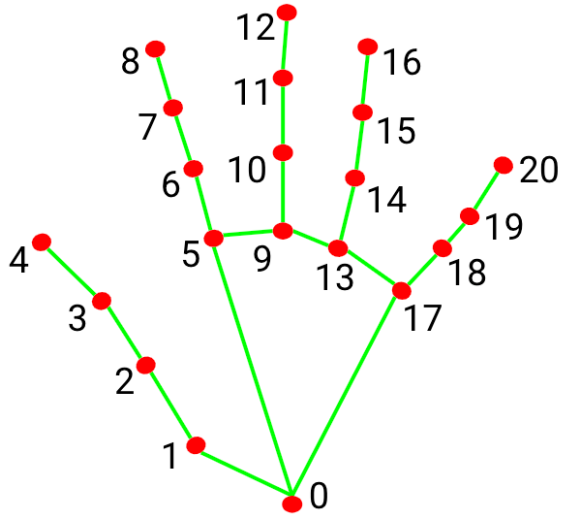
Demo 6-2.py

- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 臉部網格進行人臉辨識和繪出 3D 臉部模型，程式執行的結果可以看到綠色線標示出的 3D 臉部網格，如圖所示：



6-1-4 MediaPipe 多手勢追蹤

- MediaPipe 手勢 (MediaPipe Hands) 是使用手掌偵測模型 (Palm Detection Model) 進行多手勢追蹤，首先偵測出手掌和拳頭，然後使用手部地標模型 (Hand Landmark Model) 偵測出手部的 21 個關鍵點，如下圖所示：



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

https://google.github.io/mediapipe/solutions/hands.html

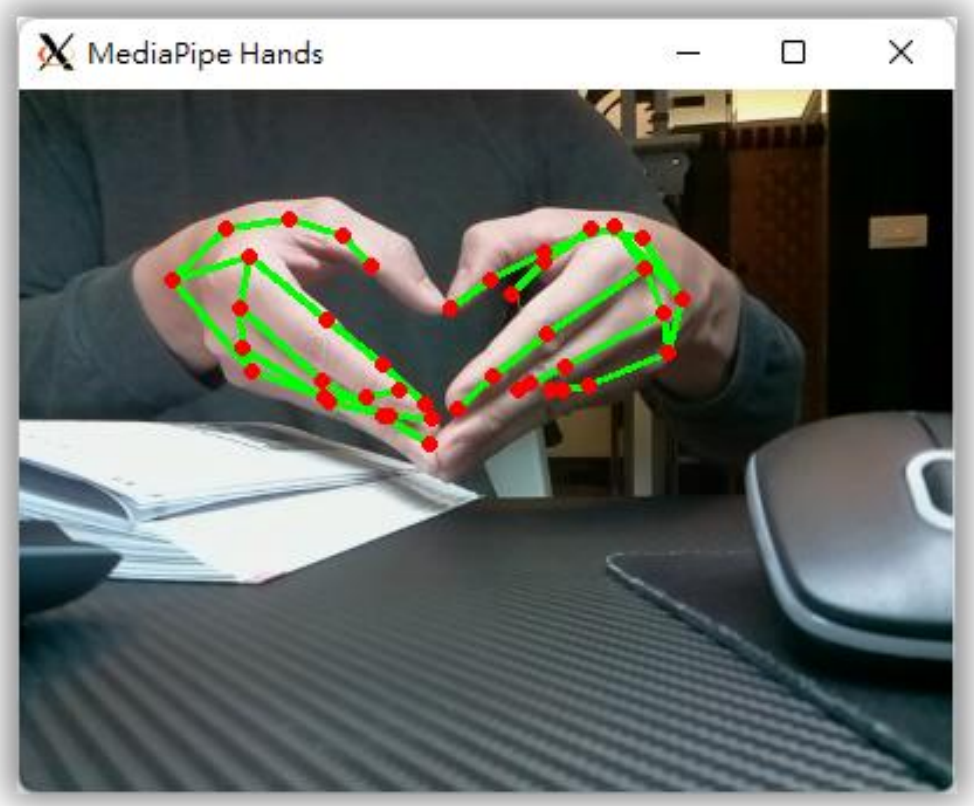
6-3 MediaPipe 多手勢追蹤。(6-3.py)

ex6-5

```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

hands = mp_hands.Hands(min_detection_confidence = 0.5, min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()

    if ret:
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
        frame = cv2.rotate(frame, rotateCode = 1)
        frame = cv2.flip(frame, 1)

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame.flags.writeable = False
        results = hands.process(frame)
        frame.flags.writeable = True
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(frame, hand_landmarks,
                                          mp_hands.HAND_CONNECTIONS)

        cv2.imshow("MediaPipe Hands", frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

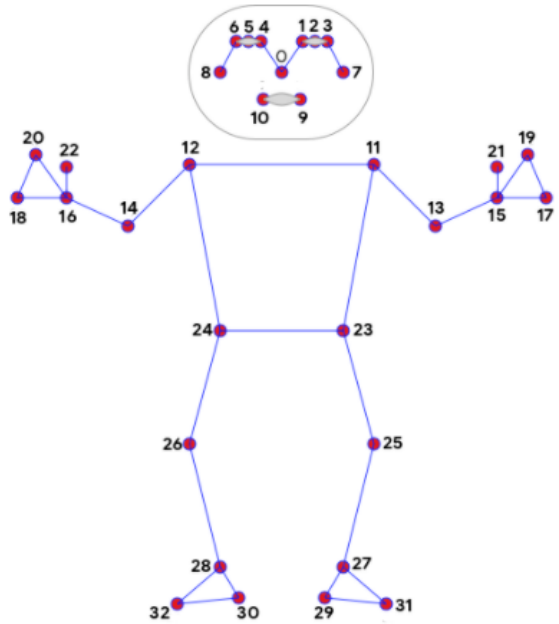
Demo 6-3.py

- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 多手勢追蹤進行手勢辨識，程式執行的結果可以看到綠色線標示出手部地標的 21 個關鍵點 (紅色點)，如圖所示：



6-1-5 MediaPipe 人體姿態估計

- MediaPipe 姿勢 (MediaPipe Pose) 是使用 BlazePose 偵測模型來進行人體姿態估計 (Human Pose Estimation)，首先偵測出人體後，使用人體地標模型 (Pose Landmark Model, BlazePose GHUM 3D) 偵測出人體的 33 個關鍵點，如下圖所示：



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

https://google.github.io/mediapipe/solutions/pose.html

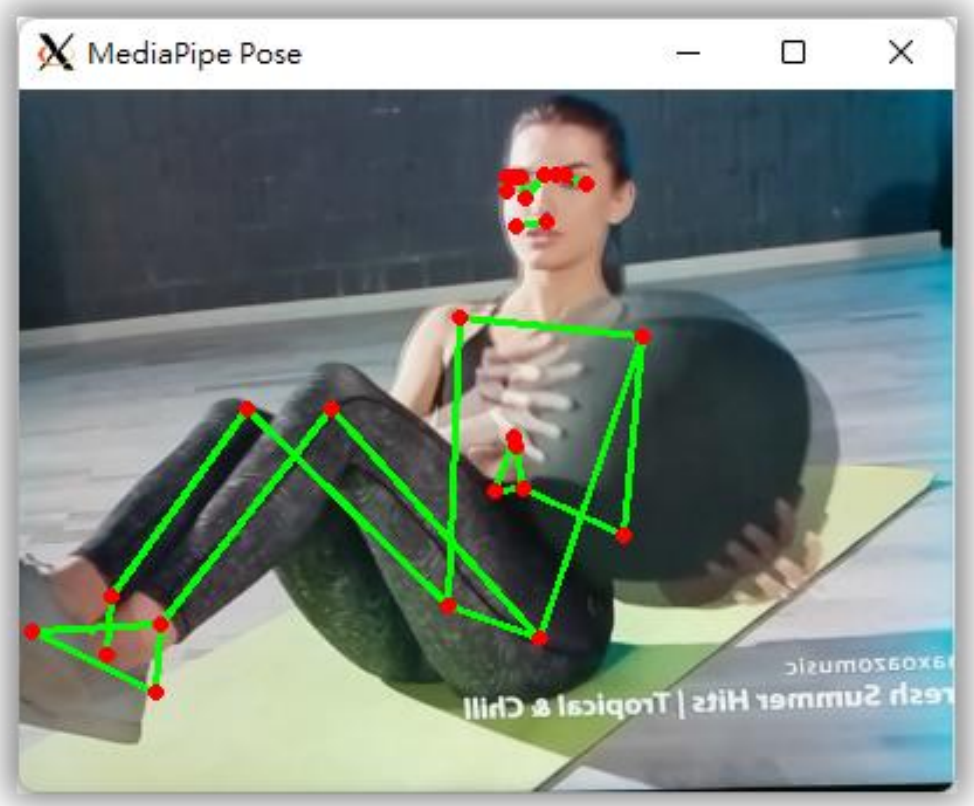
6-4 MediaPipe 人體姿態估計。(6-4.py)

ex6-6

```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

pose = mp_pose.Pose(min_detection_confidence = 0.5, min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()

    if ret:
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
        frame = cv2.rotate(frame, rotateCode = 1)
        frame = cv2.flip(frame, 1)

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame.flags.writeable = False
        results = pose.process(frame)
        frame.flags.writeable = True
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        mp_drawing.draw_landmarks(frame,
            results.pose_landmarks,
            mp_pose.POSE_CONNECTIONS)
        cv2.imshow("MediaPipe Pose", frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-4.py

- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 人體姿態估計進行姿勢的辨識，程式執行的結果可以看到綠色線標示出人體地標的 33 個關鍵點（紅色），如圖所示：



CVZone 電腦視覺套件

6-2-1 認識 CVZone

6-2-2 CVZone 人臉偵測

6-2-3 CVZone 臉部網格

6-2-4 CVZone 多手勢追蹤

6-2-5 CVZone 人體姿態估計

6-2-1 認識 CVZone

ex6-7

- CVZone 是基於 OpenCV 和 MediaPipe 的 Python 套件，我們可以使用更少的程式碼，和更容易的方式來輕鬆進行人臉辨識、3D 臉部網格、多手勢追蹤和人體姿態估計等電腦視覺應用，其官方網址如下所示：
  - ✓ https://github.com/cvzone/cvzone
- 請啟動 mediapipe 虛擬環境後，使用 pip 安裝 1.5.2 版的 CVZone，如下所示：

```
(mediapipe) $ pip install cvzone==1.5.2
```

6-2-2　CVZone 人臉偵測

- CVZone 是使用 FaceDetector 物件進行人臉偵測，然後呼叫 findFaces() 方法來找出可能的人臉。

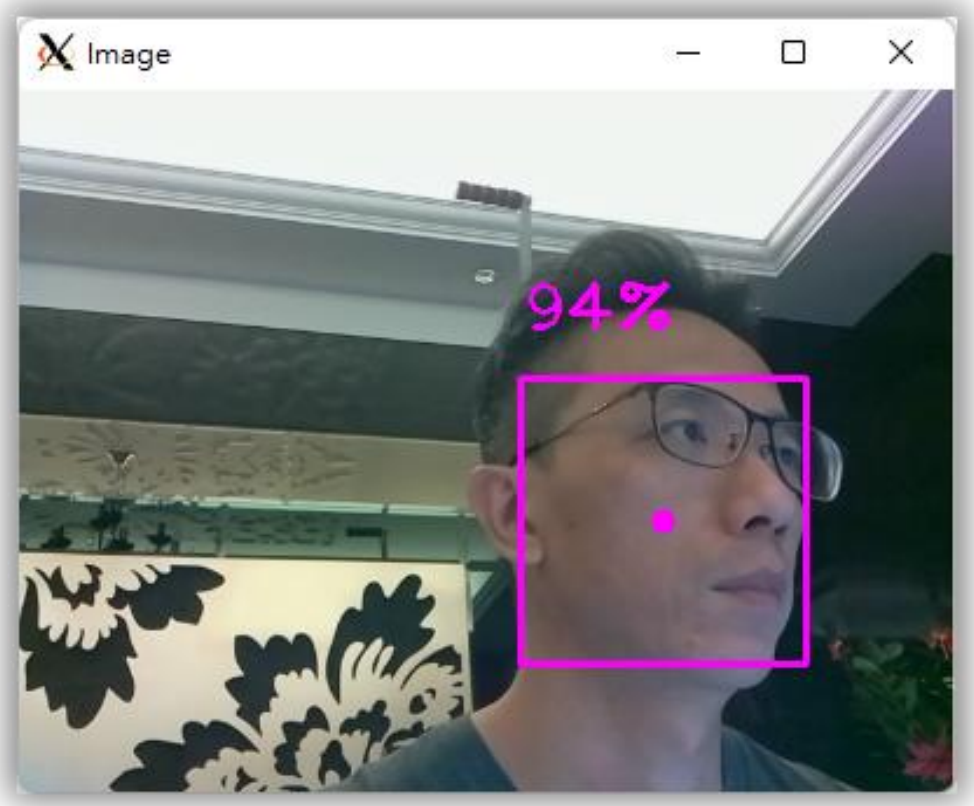
6-5 CVZone 人臉偵測。(6-5.py)

ex6-8

```
from cvzone.FaceDetectionModule import FaceDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = FaceDetector()

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    img = cv2.flip(img, 1)

    img, bboxs = detector.findFaces(img)

    if bboxs:
        # bboxInfo - "id","bbox","score","center"
        center = bboxs[0]["center"]
        cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)

    cv2.imshow("Image", img)
    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-5.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人臉辨識，程式執行的結果可以看到框出的人臉和上方顯示百分比指出是人臉可能性，和標示出方框的中心點，如圖所示：



6-2-3　CVZone 臉部網格

- CVZone 是使用 FaceMeshDetector 物件進行人臉偵測，然後呼叫 findFaceMesh() 方法來偵測和繪出臉部網格。

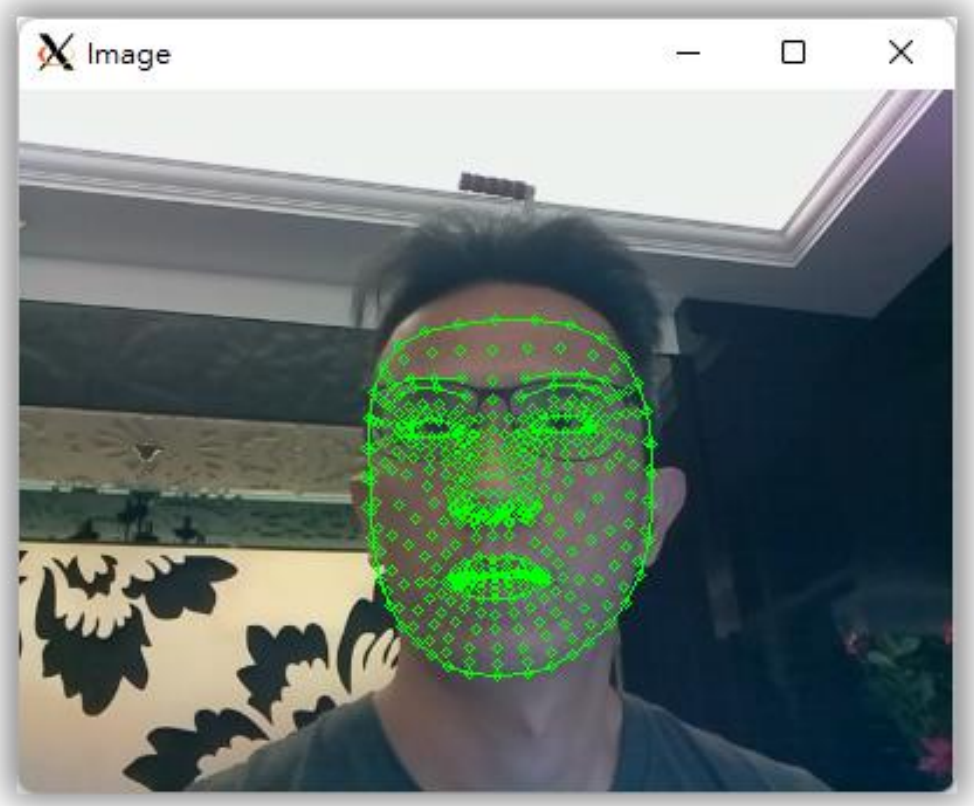
6-6 CVZone 臉部網格。(6-6.py)

ex6-9

```
from cvzone.FaceMeshModule import FaceMeshDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = FaceMeshDetector(maxFaces = 2)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    img = cv2.flip(img, 1)

    img, faces = detector.findFaceMesh(img)

    if faces:
        print(faces[0])

    cv2.imshow("Image", img)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-6.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人臉辨識，和繪出臉部 3D 網格，程式執行的結果：



CVZone 多手勢追蹤

- CVZone 的 HandDetector 物件可以偵測手勢、計算伸出幾個手指，和測量 2 個手指之間的距離。

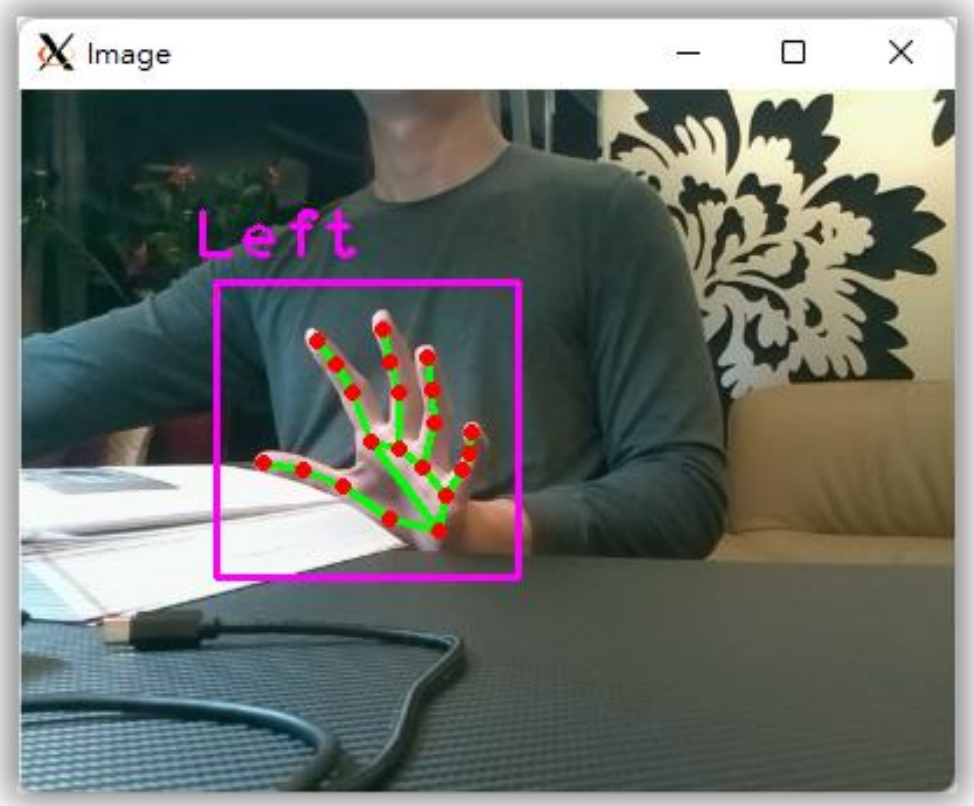
6-7 CVZone 多手勢追蹤。(6-7.py)

ex6-10

```
from cvzone.HandTrackingModule import HandDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = HandDetector(detectionCon = 0.5, maxHands = 2)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    # img = cv2.flip(img, 1)

    hands, img = detector.findHands(img)

    if hands:
        # Hand 1
        hand1 = hands[0]
        lmList1 = hand1["lmList"]
        bbox1 = hand1["bbox"]
        centerPoint1 = hand1['center']
        handType1 = hand1["type"]
        fingers1 = detector.fingersUp(hand1)

        if len(hands) == 2:
            # Hand 2
            hand2 = hands[1]
            lmList2 = hand2["lmList"]
            bbox2 = hand2["bbox"]
            centerPoint2 = hand2['center']
            handType2 = hand2["type"]
            fingers2 = detector.fingersUp(hand2)
            length, info, img = detector.findDistance(lmList1[8], lmList2[8], img)
    cv2.imshow("Image", img)
    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-7.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行多手勢追蹤，和標示偵測出的是右手或左手，程式執行的結果：



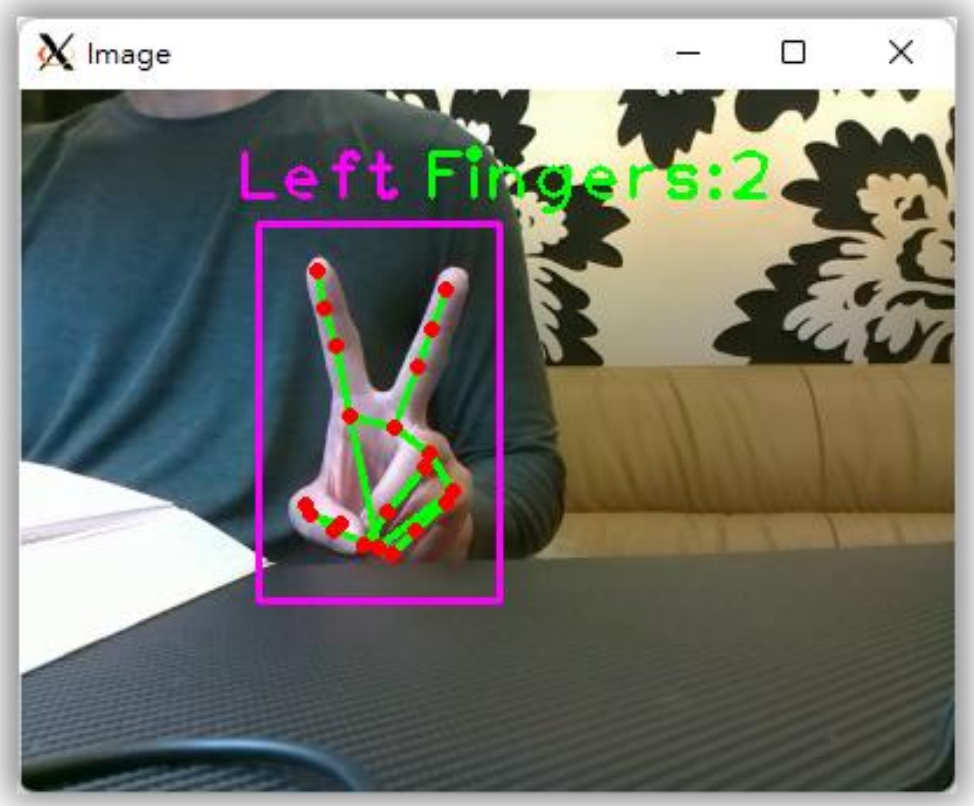
6-7a CVZone 偵測手指數。(6-7a.py)

ex6-11

```
from cvzone.HandTrackingModule import HandDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = HandDetector(detectionCon = 0.5, maxHands = 1)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    # img = cv2.flip(img, 1)

    hands, img = detector.findHands(img)

    if hands:
        hand = hands[0]
        bbox = hand['bbox']
        fingers = detector.fingersUp(hand)
        totalFingers = fingers.count(1)
        cv2.putText(img, f'Fingers:{totalFingers}', (bbox[0] + 50, bbox[1] - 30),
                    cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 2)

    cv2.imshow("Image", img)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

6-7a.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行多手勢追蹤，可以偵測出手勢共伸出幾個手指，以此例 Fingers:2，就是 2 個手指，程式執行的結果：



6-7b 計算指定 2 個手指之間的距離。(6-7b.py)

ex6-12

```
from cvzone.HandTrackingModule import HandDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = HandDetector(detectionCon = 0.5, maxHands = 2)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    # img = cv2.flip(img, 1)

    hands, img = detector.findHands(img)

    if hands:
        # Hand 1
        hand1 = hands[0]
        lmList1 = hand1["lmList"]
        bbox1 = hand1["bbox"]
        centerPoint1 = hand1['center']
        handType1 = hand1["type"]
        fingers1 = detector.fingersUp(hand1)
        length, info, img = detector.findDistance(lmList1[8], lmList1[12], img)
        cv2.putText(img, f'Dist:{int(length)}', (bbox1[0] + 50, bbox1[1] - 30),
                    cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 2)

        if len(hands) == 2:
            # Hand 2
            hand2 = hands[1]
            lmList2 = hand2["lmList"]
            bbox2 = hand2["bbox"]
            centerPoint2 = hand2['center']
            handType2 = hand2["type"]
            fingers2 = detector.fingersUp(hand2)
            length, info, img = detector.findDistance(lmList1[8], lmList2[8], img)
            cv2.putText(img, f'Dist:{int(length)}',(bbox2[0] + 50, bbox2[1] - 30),
                            cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 2)

    cv2.imshow("Image", img)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-7b.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行多手勢追蹤，可以計算指定 2 個手指之間的距離，以此例是食指和中指之間的距離 120，程式執行的結果如下圖所示：

CVZone 人體姿態估計

- CVZone 是使用 PoseDetector 物件進行人體姿態估計，我們可以呼叫 findPose() 和 findPosition() 方法來偵測出人體和找出關鍵點。

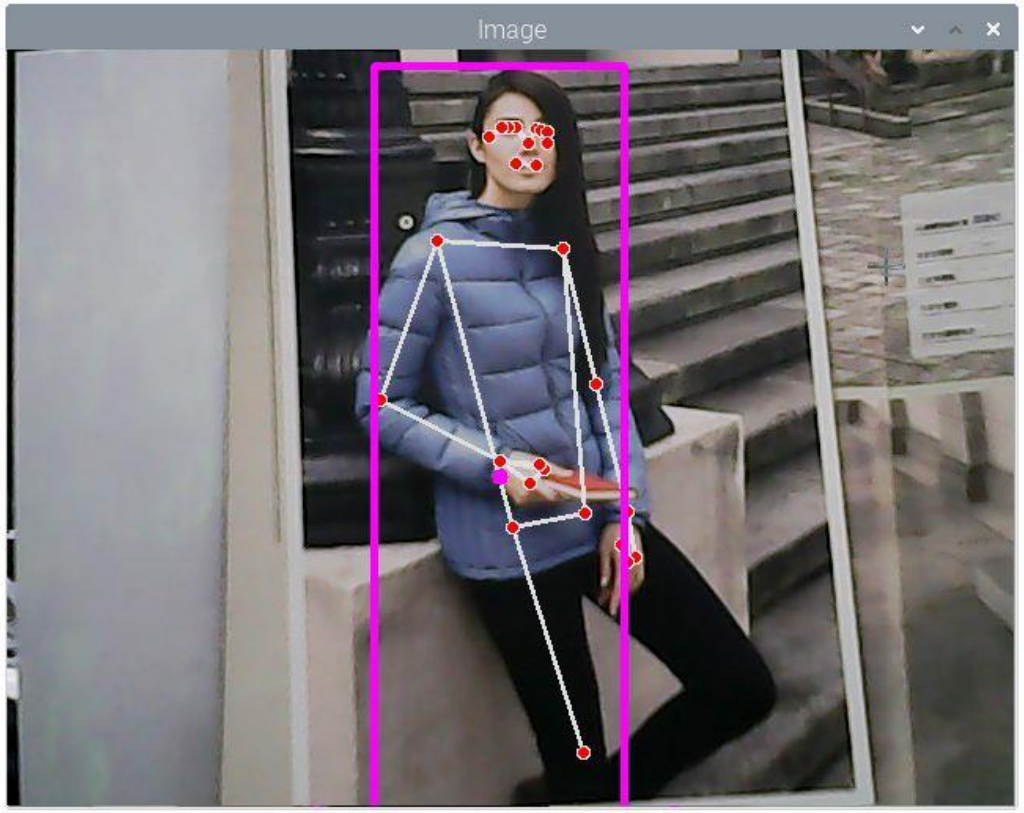
6-8 CVZone 人體姿態估計。(6-8.py)

ex6-13

```
from cvzone.PoseModule import PoseDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = PoseDetector()

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = cv2.rotate(img, rotateCode = 1)
    img = cv2.flip(img, 1)

    img = detector.findPose(img)

    lmList, bboxInfo = detector.findPosition(img, bboxWithHands = False)

    if bboxInfo:
        center = bboxInfo["center"]
        cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)

    cv2.imshow("Image", img)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Demo 6-8.py

- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人體姿態估計，程式執行的結果如下圖所示：



6-3 TensorFlow Lite 物體辨識

6-3-1 認識 TensorFlow 和 TensorFlow Lite

6-3-2 使用預訓練模型進行物體辨識

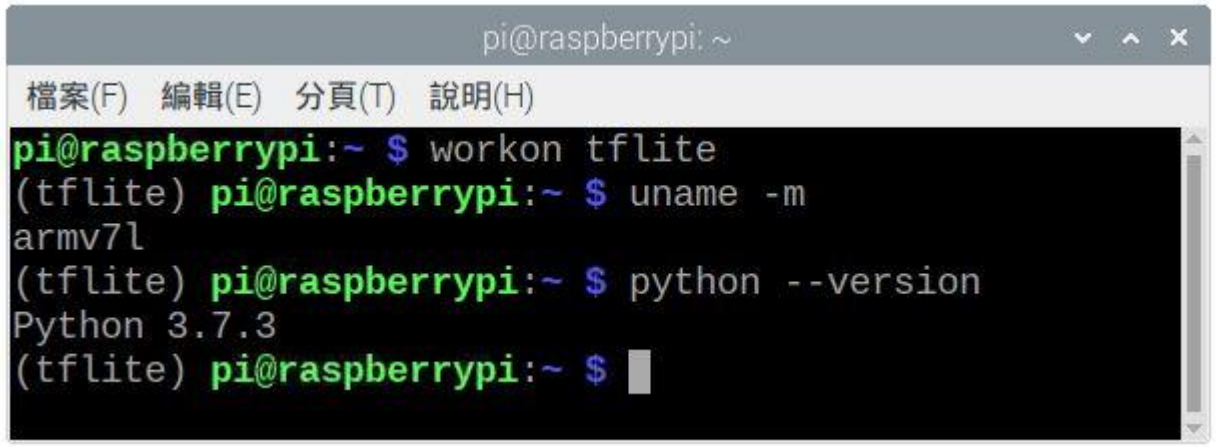
認識 TensorFlow 和 TensorFlow Lite

- TensorFlow 是一套開放原始碼和高效能的數值計算函式庫，一個機器學習/深度學習的框架，事實上，TensorFlow 就是一個完整機器學習的學習平台，提供大量工具和社群資源，可以幫助開發者加速機器學習的研究與開發，和輕鬆部署機器學習的大型應用程式。
- TensorFlow Lite 就是在行動裝置和 IoT 裝置上部署機器學習模型，這是一種開放原始碼深度學習架構，可在直接在裝置端載入模型來執行推論。

在樹莓派安裝 TensorFlow Lite

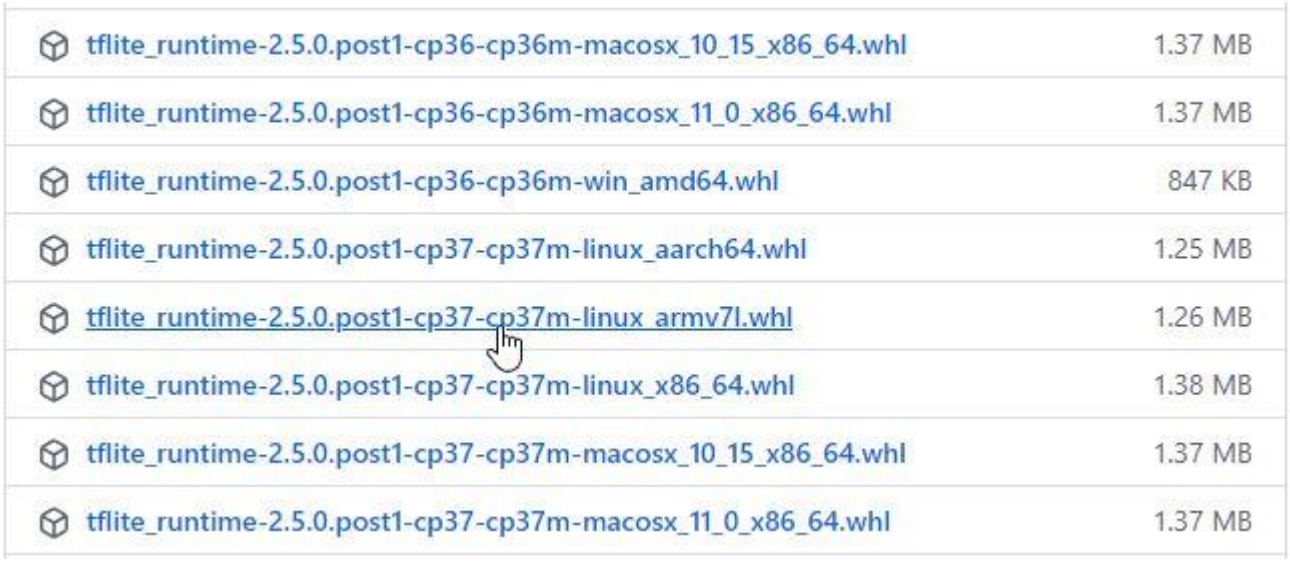
- 在 tflite 的 Python 虛擬環境安裝 OpenCV 後，就可以在 Python 虛擬環境 tflite 安裝 TensorFlow Lite，首先需要查詢 CPU 類型和 Python 版本，其指令如下所示：

```
(tflite) $ uname -m
(tflite) $ python --version
```



在樹莓派安裝 TensorFlow Lite

- 查詢結果的 CPU 是 armv7l，Python 是 3.7.3 版，然後請啟動瀏覽器開啟 TensorFlow Lite 網站下載指定 CPU 和 Python 版本的 Wheel 檔來安裝 TensorFlow Lite，其網址如下所示：
  - ✓ https://github.com/google-coral/pycoral/releases/



在樹莓派安裝 TensorFlow Lite

- 請捲動視窗找到 Python 是 3.7 版，即 cp37，最後的 CPU 是 armv7l 的 Wheel 檔 tflite_runtime-2.5.0.post1-cp37-cp37m-linux_armv7l.whl，請使用滑鼠【右】鍵複製 Wheel 檔的 URL 網址，接著，就可以在 Python 虛擬環境 tflite 安裝 TensorFlow Lite，其指令如下所示：

```
(tflite) $ pip install <Wheel檔的URL網址>
```

取得 TensorFlow Lite 預訓練模型：MobileNet

ex6-17

- TensorFlow Lite 提供多種可以馬上使用的預訓練模型，我們可以直接下載模型來進行物體辨識，例如：在 TensorFlow Hub 下載 MobileNet V1 版預訓練模型 (請下載內含標籤檔的 Metadata 版本)，其 URL 網址如下所示：
  - ✓ https://tfhub.dev/tensorflow/lite-model/mobilenet_v1_1.0_224_quantized/1/metadata/1

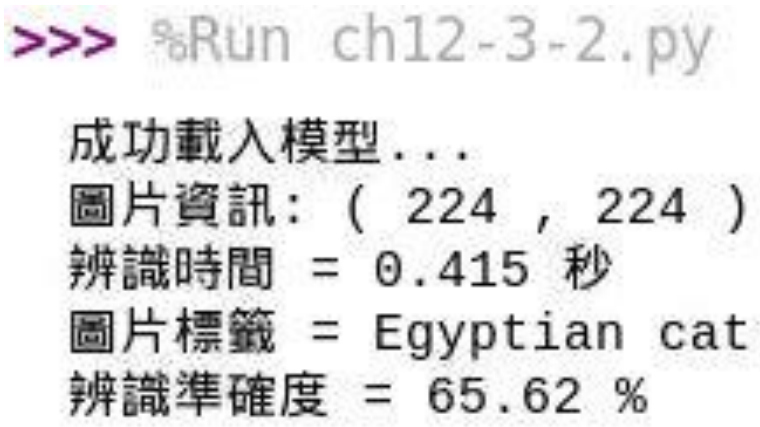
6-8 MobileNet V1 版預訓練模型。(6-9.py)

ex6-13

```
from tflite_runtime.interpreter import Interpreter
import cv2
import numpy as np
import time

# data_folder = "/home/pi/ch06/mobilenet/"
data_folder = "mobilenet/"
model_path = data_folder + "mobilenet_v1_1.0_224_quantized_1_metadata_1.tflite"
label_path = data_folder + "labels.txt"

interpreter = Interpreter(model_path)
print("成功載入模型...")
interpreter.allocate_tensors()
_, height, width, _ = interpreter.get_input_details()[0]["shape"]
print("圖片資訊: (", width, ",", height, ")")

time1 = time.time()

image = cv2.imread("images/test.jpg")
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_resized = cv2.resize(image_rgb, (width, height))
input_data = np.expand_dims(image_resized, axis=0)
interpreter.set_tensor(interpreter.get_input_details()[0]["index"],input_data)
interpreter.invoke()
output_details = interpreter.get_output_details()[0]
output = np.squeeze(interpreter.get_tensor(output_details["index"]))
scale, zero_point = output_details["quantization"]
output = scale * (output - zero_point)
ordered = np.argpartition(-output, 1)
label_id, prob = [(i, output[i]) for i in ordered[:1]][0]

time2 = time.time()
classification_time = np.round(time2-time1, 3)
print("辨識時間 =", classification_time, "秒")

with open(label_path, "r") as f:
    labels = [line.strip() for i, line in enumerate(f.readlines())]
classification_label = labels[label_id]
print("圖片標籤 =", classification_label)
print("辨識準確度 =", np.round(prob*100, 2), "%")
```

Demo 6-9.py

- MobileNet V1 預訓練模型所辨識的圖片尺寸是 (224, 224)，可以辨識 1000 種不同的物體。
- Python 程式在使用 OpenCV 讀取圖檔後，使用 TensorFlow Lite 載入 MobileNet V1 預訓練模型來辨識圖片內容，程式執行的結果可以看到辨識結果是一隻貓，如下圖所示：

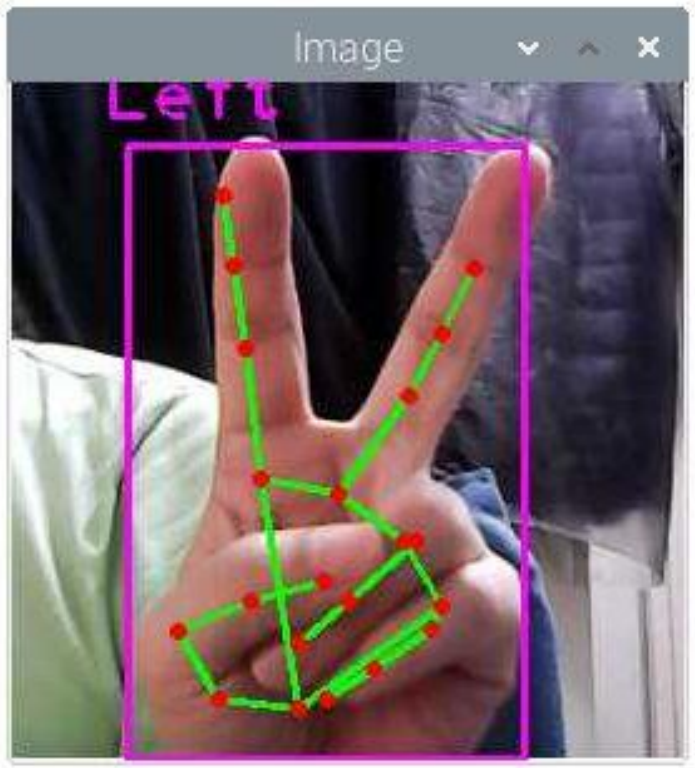


AI 實驗範例：辨識剪刀、石頭和布的手勢

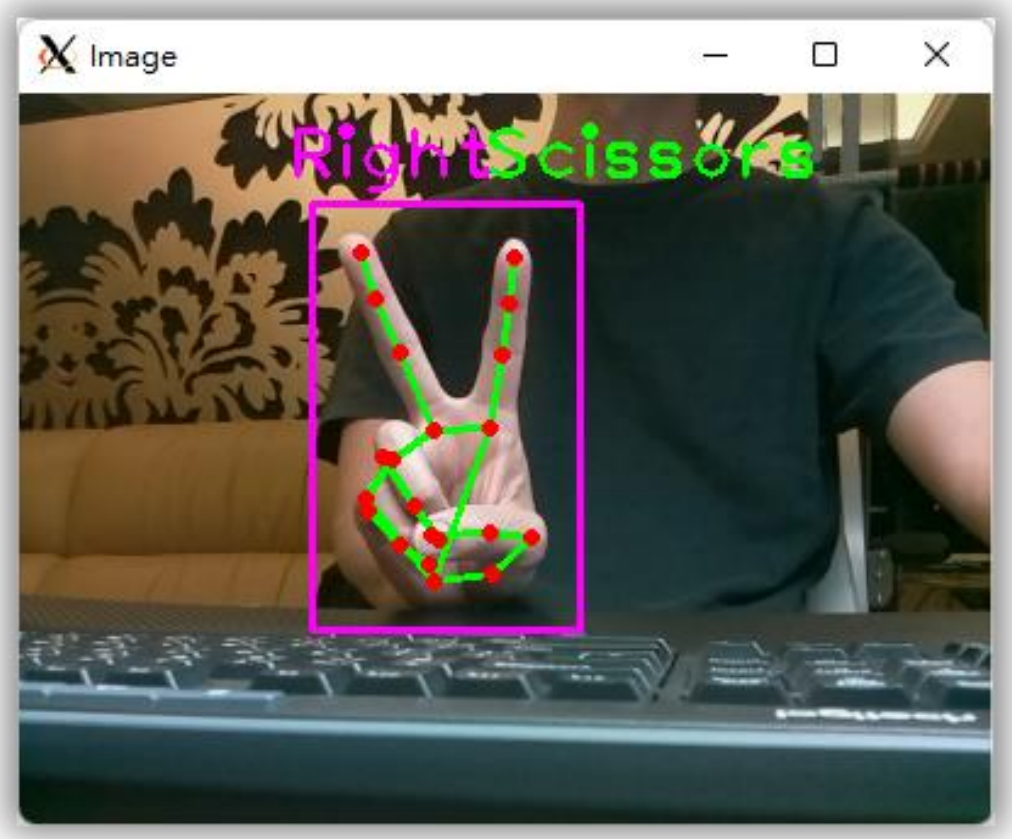
- 在本節的實驗範例準備使用 CVZone 依據手掌伸出的手指數來辨識出手勢是剪刀、石頭或布。
- 在 Python 程式可以使用 HandDetector 物件的 fingersUp() 方法來偵測手掌伸出哪些手指，如下所示：
  - ✓ `fingers = detector.fingersUp(hand)`
- 上述方法的回傳值是 5 根手指（從姆指開始至小指）的清單，例如：[0, 1, 1, 0, 0]，值 1 是伸出；0 沒有，以此例是伸出食指和中指，所以，我們可以使用此清單內容來辨識出手勢是剪刀、石頭或布。

Demo 6-10.py

- Python 程式在使用 OpenCV 讀取圖片後，使用 CVZone 進行手勢追蹤，可以偵測出手掌伸出幾個手指來辨識手勢是剪刀、石頭或布 (需使用 Python 虛擬環境 mediapipe)，程式執行的結果如右圖所示：



Demo 6-10a.py（即時影像）



即時物體辨識-取得預訓練模型：SSD MobileNet

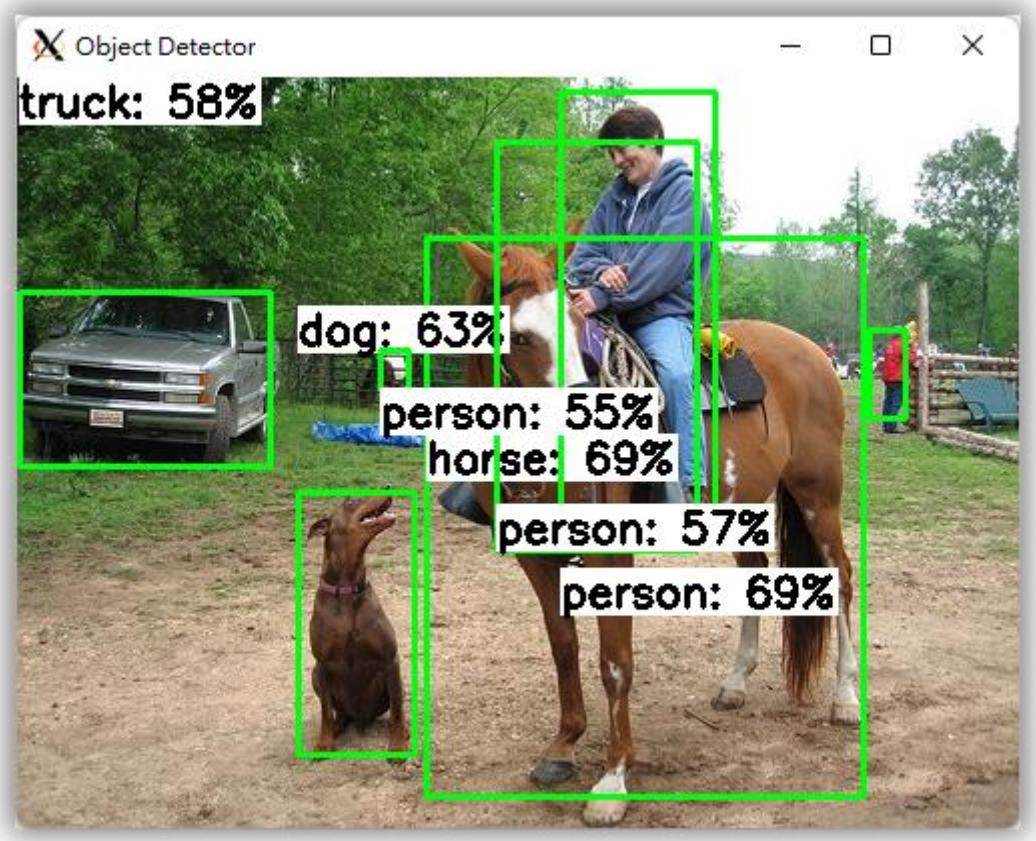
- 如同第 5 章的 YOLO 物體辨識，我們一樣可以使用 TensorFlow Lite 建立即時物體辨識，使用的是 SSD MobileNet 預訓練模型，可以辨識 90 種物體。

即時物體辨識-取得預訓練模型：SSD MobileNet

- 在 TensorFlow Hub 下載 SSD MobileNet V1 預訓練模型 (請下載內含標籤檔的 Metadata 版本)，其 URL 網址如下所示：
  - ✓ https://tfhub.dev/tensorflow/lite-model/ssd_mobilenet_v1/1/metadata/2

Demo 6-11.py

- Python 程式在使用 OpenCV 讀取影像後，使用 SSD MobileNet V1 預訓練模型來進行多物體的即時辨識，程式執行的結果可以看到使用方框標出的多個辨識物體 (需使用 Python 虛擬環境 tflite)，如下圖所示：



Demo 6-11a.py（即時影像）