人工智慧 -- 影像辨識篇

Chapter 2：OpenCV + YOLO

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

實驗 2-1：特定區域處裡

目的：OpenCV 進行指定範圍的區域處理

特定區域處理

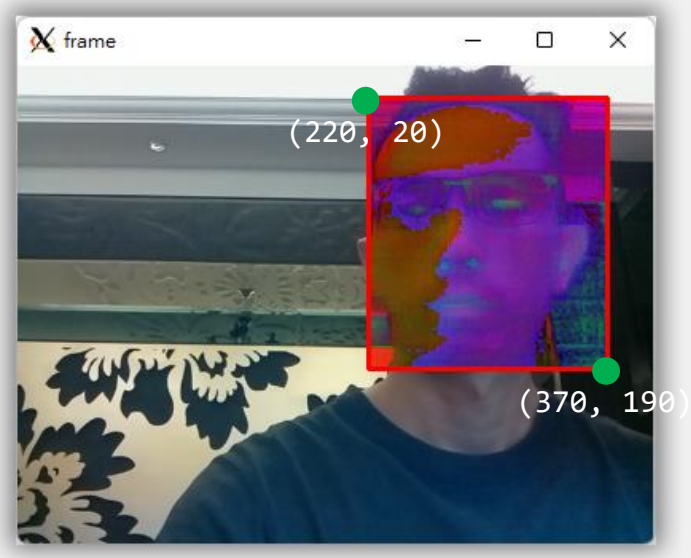
- 只處理有興趣的區域 (Region of Interest, ROI)。例如：針對有人進出『大門』這個區域，進行影像辨識就好，其餘就不管。
- OpenCV 的影像資料為矩陣型態，所以透過矩陣計算就可輕易計算出 ROI。

ROI 處理

ex2-1

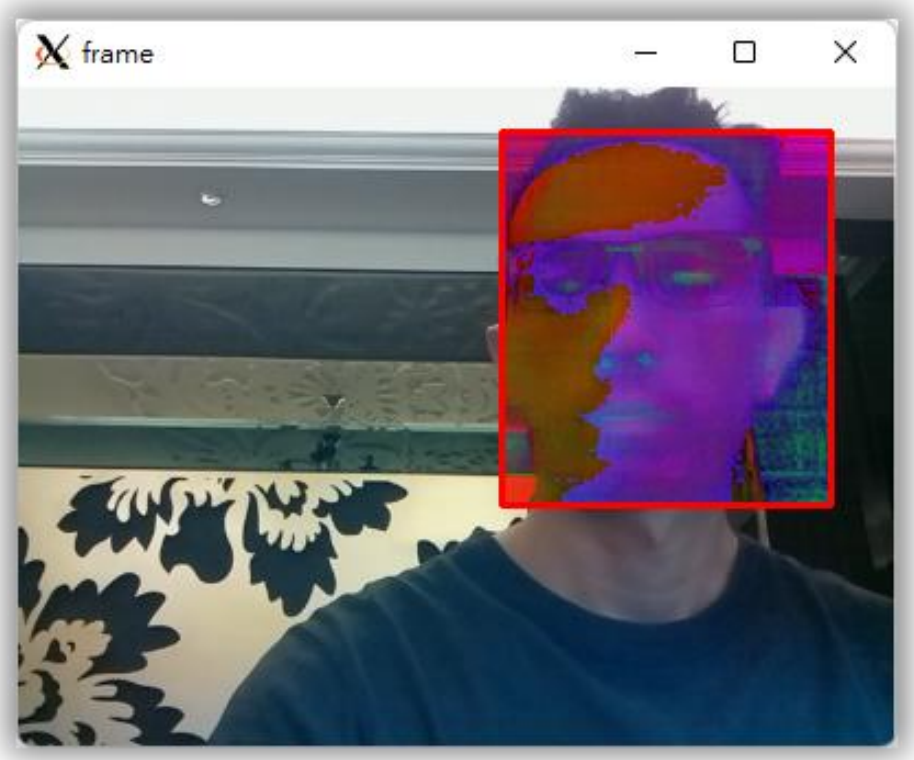
2-1 指定一個區域處理。(2-1.py)

```
import cv2

RECT = ((220, 20), (370, 190))
(left, top), (right, bottom) = RECT

def roiarea(frame):
    return frame[top:bottom, left:right]

def replaceroi(frame, roi):
    frame[top:bottom, left:right] = roi
    return frame

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)
```



```
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.rotate(frame, rotateCode = 1)
    frame = cv2.flip(frame, 1)

    # 取出子畫面
    roi = roiarea(frame)
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
    # 將處理完的子畫面，貼回到原本畫面中
    frame = replaceroi(frame, roi)

    # 在 ROI 範圍處畫個框
    cv2.rectangle(frame, RECT[0], RECT[1], (0, 0, 255), 2)
    cv2.imshow('frame', frame)

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

Demo 2-1.py



實驗 2-2：物體移動追蹤

目的：指定區域，物體移動追蹤

物體移動追蹤

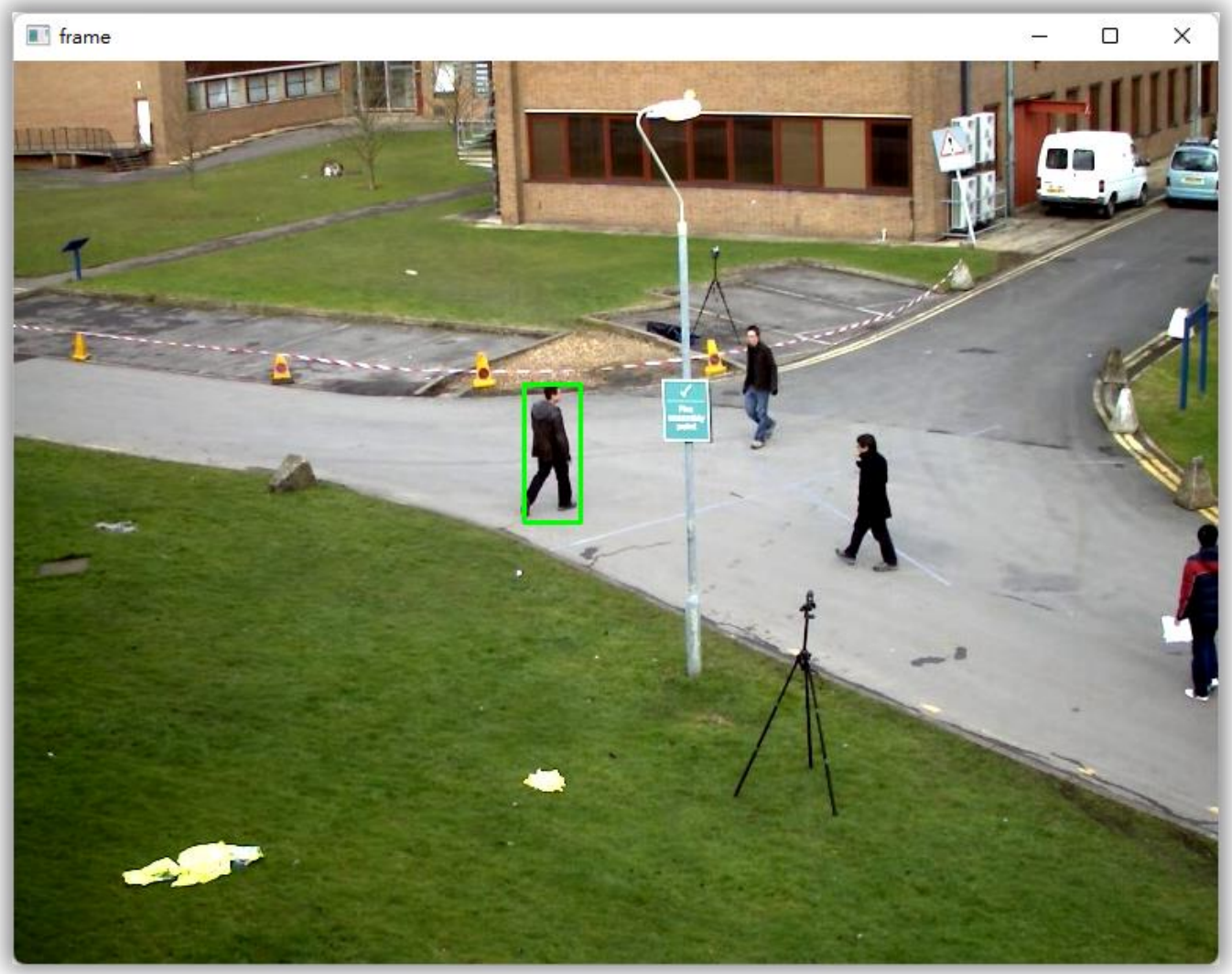
- 移動追蹤演算法：先設定好想要追蹤物體在畫面上的座標，然後當該物體移動到新地方時，OpenCV 會自動計算出移動後的座標。
  - ✓ 比每一個畫面都用物件辨識演算法 (例如：類神經網路)，去找物件所在位置要快多了。
  - ✓ OpenCV 提供了 8 種不同的演算法：Boosting、CSRT、GOTURN、KCF、 MedianFlow、MIL、MOSSE、TLD。
- selectROI 函數：藉由這個函數在畫面上拖曳出一個矩形範圍，然後就可以得出這個矩形在畫面上的座標值了。

物體移動追蹤演算法

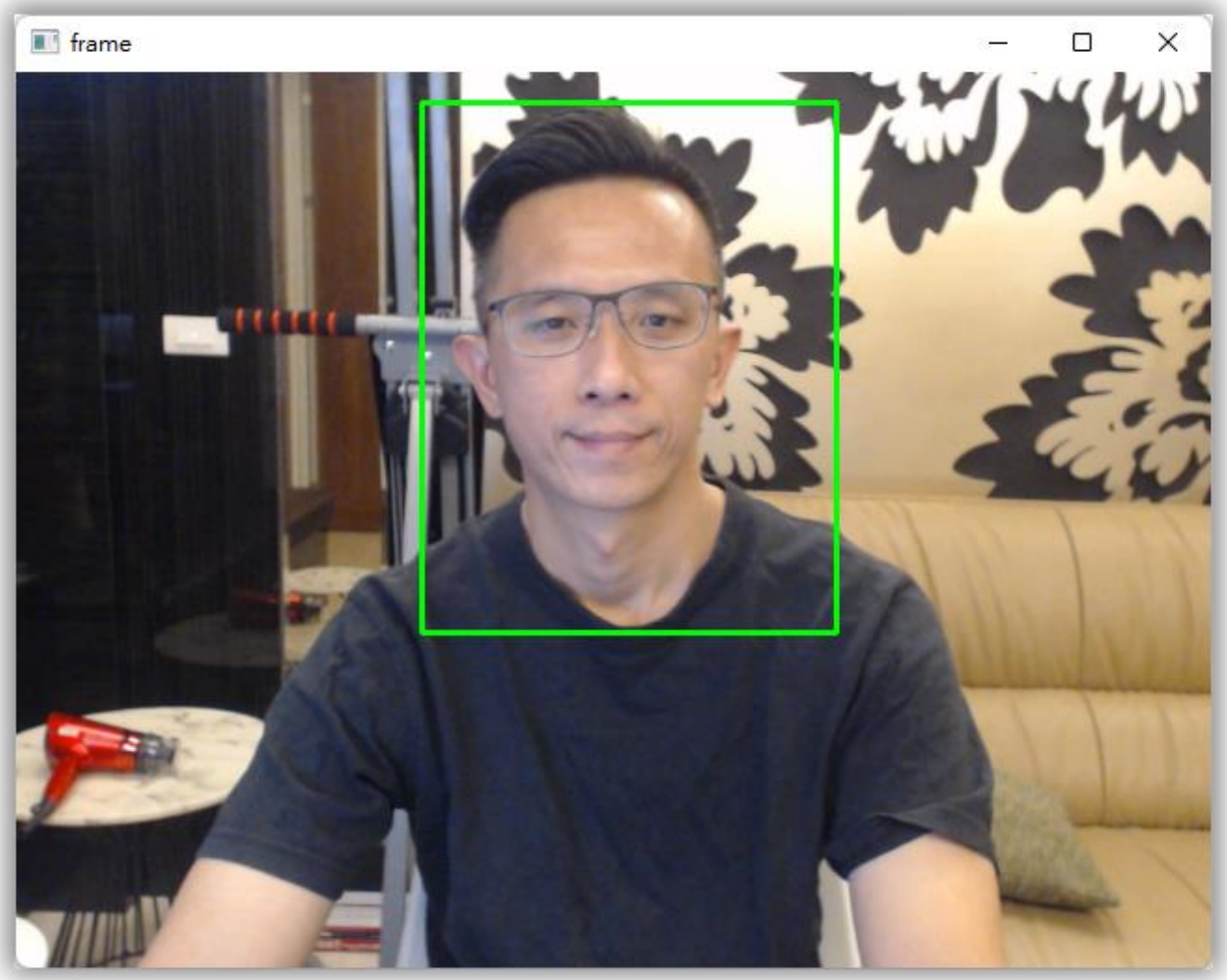
2-2 物體移動追蹤。(2-2.py)

```
import cv2

cap = cv2.VideoCapture('vtest.avi')   # OpenCV 範例影片 vtest.avi
tracker = cv2.TrackerCSRT_create()
roi = None    # 宣告一個變數 roi 來儲存欲追蹤的座標位置。
while True:
    ret, frame = cap.read()

    if roi is None:
        roi = cv2.selectROI('frame', frame)
        if roi != (0, 0, 0, 0):    # 若有框出一個區域，則將矩形區域傳給追蹤演算法的 init() 函數。
            tracker.init(frame, roi)

    success, rect = tracker.update(frame)  # 物體移動到新的區域，update() 計算新區域的座標，將新
        if success:                        座標交給 rectangle() 畫出一個矩形區域。
            (x, y, w, h) = [int(i) for i in rect]
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow('frame', frame)
        if cv2.waitKey(66) == 27:
            cv2.destroyAllWindows()
            break
```

Demo 2-2.py



Demo 2-2a.py（即時影像）

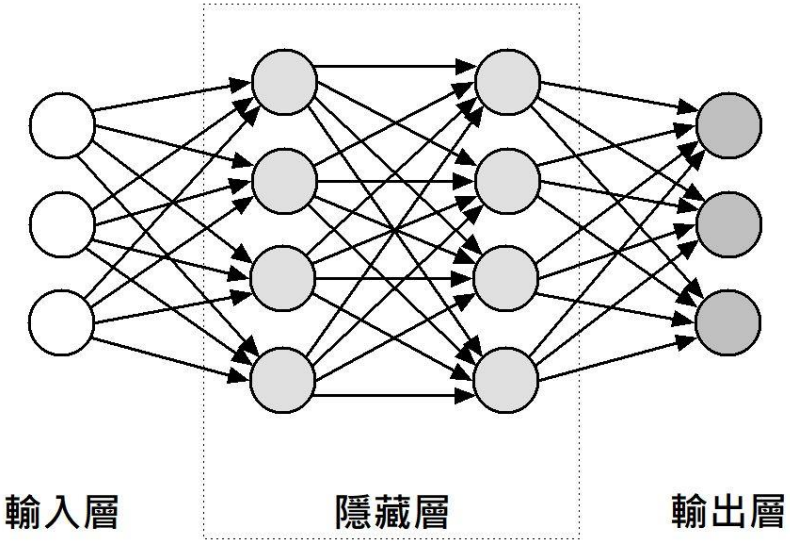


YOLO 物體辨識的深度學習演算法

- YOLO 是一種快速且準確的物體辨識 (Object Recognition) 演算法，也是一種深度學習演算法 (Deep Learning Algorithms)。
- YOLO 演算法是使用深度學習的卷積神經網路 (Convolutional Neural Networks, CNN)，如其英文名稱所述，YOLO 只需單次神經網路的前向傳播 (Forward Propagation)，就可以準確的辨識出物體。其官方網址如下所示：
  - ✓ https://pjreddie.com/darknet/yolo/

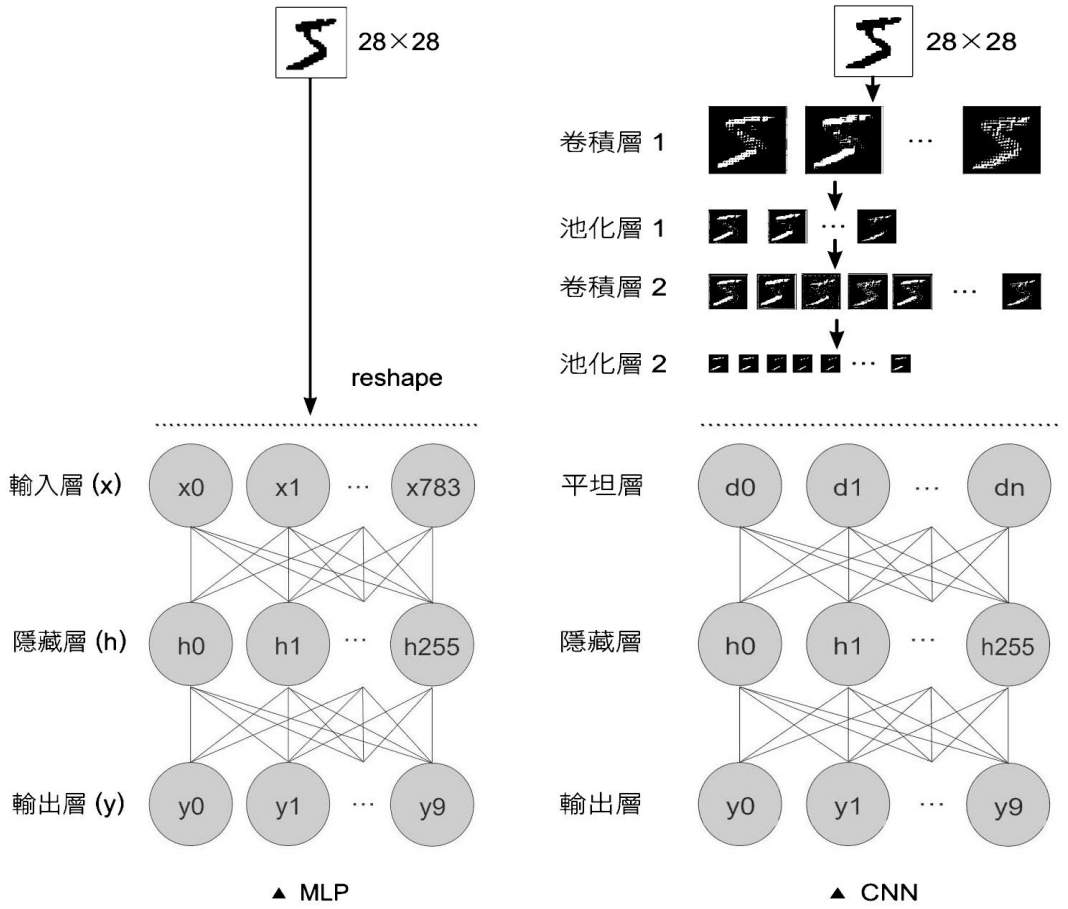
什麼是深度學習

- 在多層神經網路 (Multilayer Perceptron, MLP) 的每一個圓形頂點是一個神經元，整個神經網路包含『輸入層』(Input Layer)、中間的『隱藏層』(Hidden Layers) 和最後的『輸出層』(Output Layer) 。
- 深度學習使用的神經網路稱為『深度神經網路』(Deep Neural Networks，DNNs)，其中間的隱藏層有很多層，意味著整個神經網路十分的深 (Deep)，可能高達 150 層隱藏層。



什麼是深度學習

- 深度學習也是一種機器學習。如下右圖所示（CNN)，左圖為傳統的機器學習(MLP)：



下載 YOLO 相關檔案

ex2-3

- 在本書是使用 OpenCV 執行 YOLO 演算法。在 Python 程式執行 YOLO 演算法需要下載三個檔案，其說明如下所示：
  - ✓ 權重檔 (Weight File)：預訓練模型的權重檔 yolov3.weights，檔案大小 237MB。
  - ✓ 設定檔 (Cfg File)：YOLO 演算法本身的設定檔 yolov3.cfg。
  - ✓ 分類名稱檔 (Name File)：演算法可辨識物體名稱清單的檔案 coco.names。

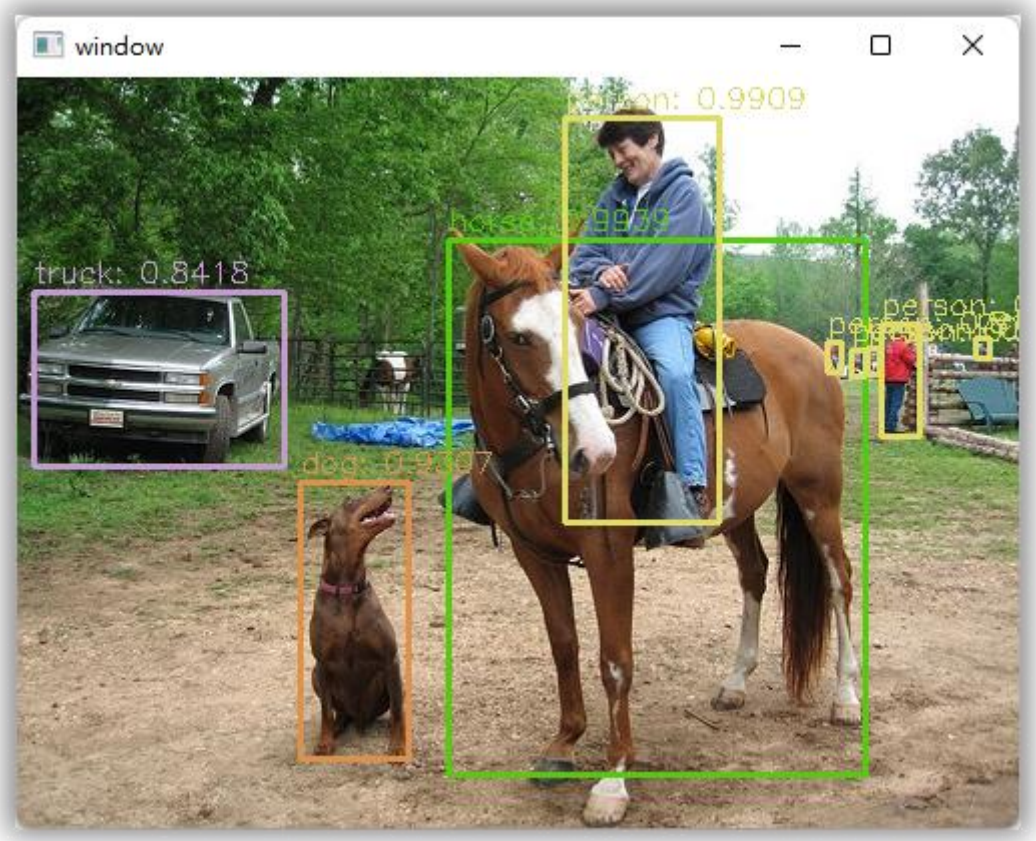
建立 OpenCV + YOLO 物體辨識

ex2-3

- OpenCV + YOLO 物體辨識的官方教學文件，其 URL 網址如下所示：
  - ✓ https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html
- 2-3.py 是修改自 OpenCV + YOLO 物體辨識的官方教學文件。下載的 3 個 YOLO 相關檔案是儲存在『ch02\yolo』目錄。

Demo 2-3.py

- Python 程式在使用 OpenCV 讀取圖檔後，使用 YOLO 進行物體辨識，程式執行的結果可以看到使用不同色彩的方框所標示出的辨識物體，並且在上方顯示分類名稱和信心指數值，如下圖所示：



Demo 2-3a.py（即時影像）