中華民國第 65 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

052517

防跌守護，步步安心

學校名稱： 國立金門高級中學

作者：	指導老師：
高一 蔡秉翰 高一 陳玟安	楊素菀

關鍵詞： YOLO、影像檢測

摘要

隊員的外婆在台灣獨居，曾因跌倒而未能及時求救，引發家人擔憂，高齡化與獨居老人數量增加，導致安全照護挑戰，需改善意外求助系統。本研究目的是使用 OpenCV 和 YOLO 技術辨識人體動作，檢測跌倒情況及其影響發送郵件通知家人。本研究使用 Pexels 和 Google 收集圖片進行訓練，Roboflow 工具加快標註過程。過程中整合偵測、影像擷取及郵件發送功能，設計時間間隔機制避免重複發送郵件。訓練結果顯示模型在辨識「躺下」動作上準確率高，對其他動作表現尚可，而系統運行時 CPU 負擔輕，確保穩定性與擴展性。本系統對人物重複辨識和跌倒檢測準確性有改進空間，考慮增加資料量及調整標註類別。未來計畫開發專用 APP 以提升訊息傳送功能，增強用戶體驗。

壹、 前言

一、研究動機

幾年前，我們隊員的外婆獨自一人在台灣生活，而她與父母當時則住在金門。有一天外婆不慎在家中跌倒，家人直到 2 至 3 小時後才得知情況。幸好當時並未造成嚴重傷害，但這起事件也讓我們深刻體認到，若未來再發生類似情況，後果可能難以挽回。

全球高齡化與少子化的趨勢日益明顯，獨居老人的數量持續增加，帶來了嚴峻的安全照護挑戰。許多獨居長者在發生意外時，常因無法及時求助而錯失黃金救援時間，導致原本可避免的悲劇發生。為了降低這類事件造成的傷害與遺憾，我們決定開發一款智慧跌倒偵測系統。當系統偵測到跌倒事件發生時，會即時透過電子郵件通知家屬或照護人員，協助迅速應變，從而提升獨居老人的安全與生活品質。

目前市面上多數跌倒偵測產品依賴穿戴式裝置（如智慧手錶），但對年長者而言，這類設備可能因為忘記佩戴、感到不適或操作不便而降低效用。因此，我們選擇跳脫穿戴式裝置的限制，開發一套**非穿戴式**智慧跌倒偵測系統，以提高實用性與穩定性。

事實上，跌倒並非只影響老年人，而是全球性的公共安全議題。根據世界衛生組織（WHO）統計，每年全球約有 68.4 萬人死於跌倒。在不同年齡層中，跌倒都是常見的意外原因。例如：

- 在中國，0 至 19 歲人群的跌倒發生率為 6.5%；
- 在香港，14 歲以下兒童的受傷事件中，有 37.2%與跌倒有關；
- 在 15 至 64 歲的成年人中，跌倒亦佔所有受傷事件的 26.5%。

基於這些資料，我們的系統設計將不僅侷限於長者，也將考慮不同年齡層的風險因素，進一步提出相對應的預防措施，全面降低跌倒所帶來的健康風險。

二、研究目的

本研究之目的為:

(一)利用 OpenCV 和 YOLO 技術辨識人體動作變化，進而檢測是否發生跌倒情況。

(二)測試不同參數之 YOLO 的結果差別

(三)測試不同標註方式對訓練結果的影響

三、文獻回顧

(一)YOLO

YOLOv11 是 Ultralytic 推出的最新物件檢測模型系列，基於前幾代模型的技術優勢，進一步在速度和精度上取得了突破。它提供了一個統一的框架，不僅支援目標檢測，還支援實例分割和圖像分類任務。

預設模型能夠辨識 COCO 資料集中的 80 種物件類別；若需偵測其他類別，則需要準備自訂資料集並進行專屬模型訓練（例如本研究就使用自訂資料集做訓練）。

此外，YOLOv11 分為五種不同尺寸的子模型(而其中最主要的兩種為 Nano 和 Extra Large)：

1. **Nano (YOLO11n)**：體積最小、運行最快，適合資源有限的環境。
2. **Extra Large (YOLO11x)**：雖然速度較慢，但精度最高，適用於要求極高精度的應用。

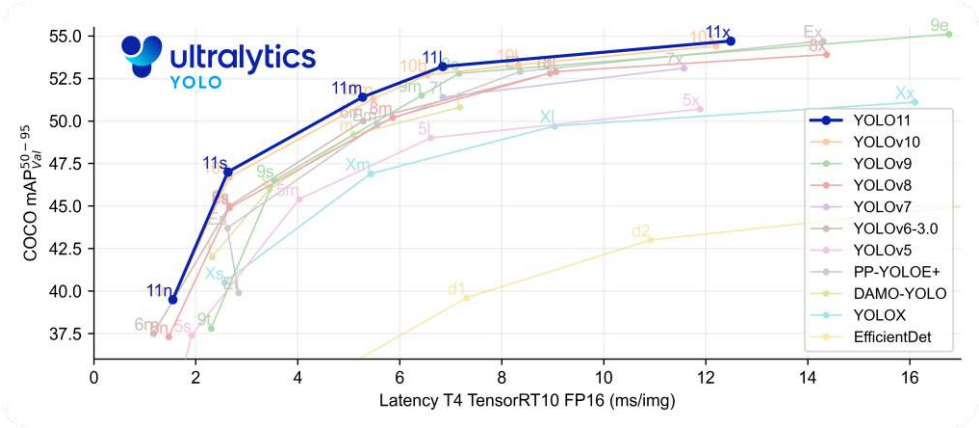


圖 1 YOLO 各版本比較(出處詳圖片來源一)

與前一版本 YOLOv8 相比，YOLOv11 在特徵提取、速度與效率優化和任務範圍上都有所改進，從而實現了更快的運行速度和更高的檢測精度。這使得 YOLOv11 在多種電腦視覺任務，使其在不同任務(物體檢測、姿勢評估、圖像分類等)達到最前端，且處理能力也進一步的進行優化。

(二)Roboflow

Roboflow 是一個專為計算機視覺項目設計的平臺，它不僅提供強大的資料標註工具和數據預處理功能，還能訪問大量開源數據集。更重要的是，在資料導出之前，Roboflow 內建的數據增強功能能自動生成多樣的訓練樣本，從而增強模型泛化能力與精度。

(三)OpenCV

根據最新的 OpenCV 文檔和實際應用經驗，基於 Haar 特徵的級聯分類器仍沿用了

Viola-Jones 論文中的基本思路——在初步從約 160,000 個候選特徵中選取出最具區分能力的特徵，但隨著多次優化，實際用於分類器中的有效特徵數量通常在 5,000 至 7,000 個之間，且級聯階段數(cascade stages)會根據具體任務（例如人臉檢測）調整為約 30 到 40 個階段。這種分層篩選機制使得在一幅圖像中，初期可以快速剔除絕大部分非目標區域，僅對可能包含目標的窗口進行更深入的檢測，從而大幅提升了檢測效率。

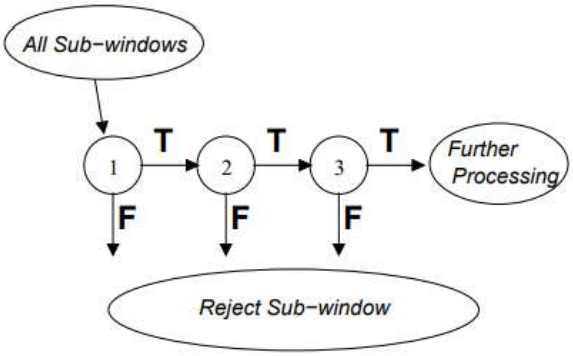


圖 2 偵測級聯的原理示意圖(出處詳圖片來源二)

此外，從 OpenCV 3.3 開始，除了傳統的 Haar 特徵級聯方法外，OpenCV 的 DNN 模組也已引入了多種基於深度學習的檢測方法，如 CNN、Caffe、TensorFlow 等，進一步擴展了其在複雜場景下的應用能力，同時也支援 OpenVINO 等硬體加速方案，滿足不同設備和應用的需求。

(四)Smtplib

Python 內建的 smtplib 模組可用來與任何支持 SMTP 或 ESMTP 協議的郵件伺服器建立連線並發送郵件。而 email 模組中的 MIME 子模組（正確拼寫為 email.mime，而非 email.meme）則提供了構造 MIME 格式郵件的工具，這使得我們能夠輕鬆地發送包含非 ASCII 字元、HTML 格式內容及二進位元附件（如圖片、音訊、檔案等）的郵件。

在本研究中，我們利用 smtplib 連接 Google 的 SMTP 伺服器，並通過 email.mime 模組構造一封同時包含圖片和 HTML 內容的郵件，實現郵件自動化發送。這種方法不僅能夠確保郵件格式豐富多樣，而且能夠靈活處理各種類型的附件，從而滿足更高層次的郵件自動化需求。

(五)PyTorch

PyTorch 由 Meta 開發，是一款以 Python 為主要介面的開源深度學習結構。它採用動態計算圖設計，使得模型定義與調試直觀且靈活，同時支援高效的 GPU 加速，大幅縮短訓練時間。活躍的社群和豐富的學習資源進一步推動了其在電腦視覺、自然語言處理等多個領域的廣泛應用，成為深度學習研究和實踐的重要工具。

(六)卷積神經網路（CNN）

卷積神經網路（CNN）是一種深度學習模型，主要用於處理與分析數據，如圖像或影片。其核心結構包括：

1. **卷積層（Convolutional Layer）**

(1). **核心概念**：負責提取圖像的局部特徵，類似於一個過濾器（Filter）在圖像上滑動，逐步檢測不同的特徵，如邊緣、紋理和形狀。

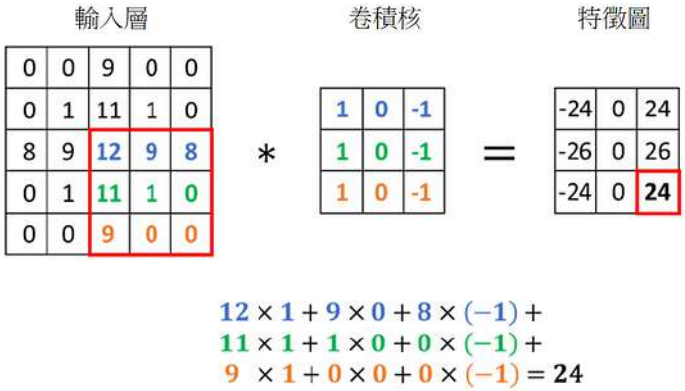


圖 3 捲積示意圖(出處詳圖片來源三)

(2). **運作方式**：

A. 透過多個**卷積核（Filter）**來提取圖像特徵。

B. 卷積核在圖像上逐步移動，每次移動的距離稱為**步長（Stride）**。

(3). **步長大小的影響**：

A. **較小的步長**：能捕捉更多細節特徵，但會增加計算量。

B. **較大的步長**：計算效率較高，但可能遺漏部分細節資訊。

卷積層的作用是將輸入影像轉換為一組特徵圖（Feature Maps），作為後續層進一步學習與分析的基礎。

2. 池化層（Pooling Layer）

(1). **核心概念** ： 位於捲積層之後，減少特徵圖片之尺寸，降低計算量的同時，維持特徵。

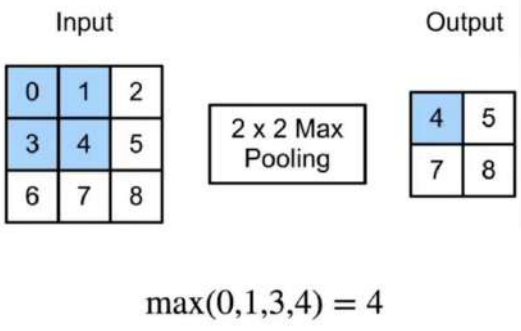


圖 4 池化層示意圖(詳圖片來源四)

(2). **常見類型：**

A. **最大池化 (Max Pooling)**：選取區域內最大值，保持主要特徵。

B. **平均池化 (Average Pooling)**：取平均值，用於平滑特徵。

3. **啟動函數 (Activation Function)**

(1). **核心概念 ：** 每個神經元都配有一個啟動函數，決定該神經元的輸出如何傳遞到下一層神經元，以此獲得特徵組合，增強學習。

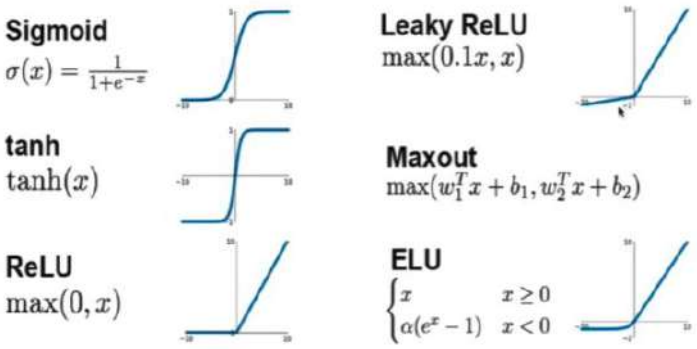


圖 5 啟動函數示意圖(詳圖片來源五)

(2). **啟動函數的作用：**

A. **非線性轉換**：引入非線性，使神經網路能夠學習複雜的模式，而不僅僅是線性映射。

B. **特徵組合**：幫助模型提取和組合數據中的關鍵特徵，提升表現能力。

常見啟動函數種類有 Sigmoid、Tanh、ReLU 等，依據不同情況去選擇不同的線性函數，但 ReLU 及其變體在深度學習中應用較廣泛。

CNN 透過捲積、池化、啟動等多種運算，成為機器學習領域中強大的技術之一。其特性使其特別適用於需要處理大量數據的任務，例如影像辨識與分類。CNN 能夠有效提取數據中的關鍵特徵，並透過深度學習模型進行準確分類，因此被廣泛應用於各種電腦視覺領域，如物件偵測、人臉識別及醫學影像分析等。

貳、 研究設備與器材

一、設備

(一)筆記型電腦

(二)Colab

二、軟體

軟體	python	GPT	DeepSeek	Roboflow	Pexeles
套件	time	os	opencv	ultralytics	smtplib

	email.mime	matplotlib	random	seaborn	Pytorch

參、 研究過程與方法

一、研究流程

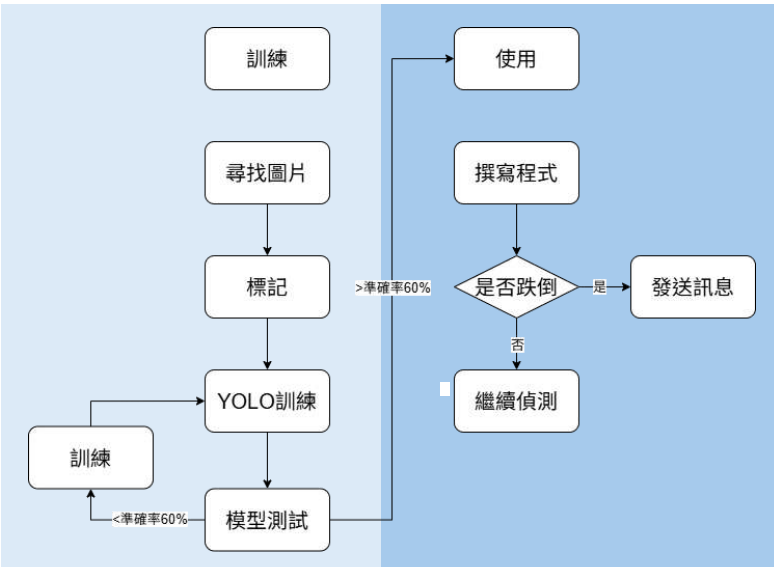


圖 6 研究流程圖(自製)

一、學習神經網路之架設與學習

在剛開始製作此專案時，老師要求我們先透過網路資料學習並理解神經網路的基本概念，並推薦了一位名為「土堆」的 YouTuber 所製作的 **PyTorch 入門教學影片**。透過觀看該影片，我們學習了 PyTorch 的基礎操作、TensorBoard 的使用方式，以及神經網路架構的設計等重要內容。

接著，我們依照影片的教學步驟，成功撰寫了一個能夠使用 CIFAR-10 數據集進行模型訓練的程式，並在實作過程中深入理解了 **卷積神經網路**（CNN）、**梯度下降**（Gradient Descent）等關鍵技術。這些學習經驗為我們後續的研究與開發奠定了穩固的基礎，使我們能夠更有信心地進行進階模型的設計與優化。

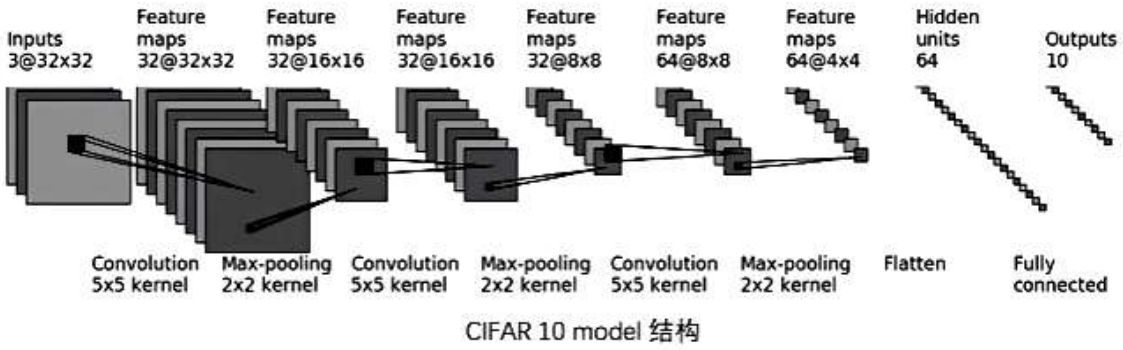


圖 7 訓練結構(詳圖片來源六)

神經網路架設

```
class Tudui(nn.Module):
    def __init__(self):
        super(Tudui, self).__init__()
        self.model1 = nn.Sequential(
            nn.Conv2d(3,32,5,1,2),  # 輸入通道 3，輸出通道 32，卷積核尺寸 5x5，步長 1，填充 2
            nn.MaxPool2d(2),
            nn.Conv2d(32,32,5,1,2),
            nn.MaxPool2d(2),
            nn.Conv2d(32,64,5,1,2),
            nn.MaxPool2d(2),
            nn.Flatten(),  # 展平後變成 64*4*4
            nn.Linear(64*4*4,64),
            nn.Linear(64,10)
        )

    def forward(self, x):
        x = self.model1(x)
        return x
```

模型訓練

```
# 訓練步驟開始
    tudui.train()  # 當網路中有 dropout 層、batchnorm 層時，這些層能起作用
    for data in train_dataloader:
        imgs, targets = data
        if torch.cuda.is_available():
            imgs = imgs.cuda()  # 數據放到 cuda 上
            targets = targets.cuda() # 數據放到 cuda 上
        outputs = tudui(imgs)
        loss = loss_fn(outputs, targets)  # 計算實際輸出與目標輸出的差距

        # 優化器對模型調優
        optimizer.zero_grad()  # 梯度清零
        loss.backward()  # 反向傳播，計算損失函數的梯度
        optimizer.step()   # 根據梯度，對網路的參數進行調優

        total_train_step = total_train_step + 1
```

```
        if total_train_step % 100 == 0:
            print("訓練次數：{}，Loss：
{}".format(total_train_step,loss.item()))  # 方式二：獲得 loss 值
            writer.add_scalar("train_loss",loss.item(),total_train_step)
```

模型測試

```
    # 測試步驟開始（每一輪訓練後都查看在測試資料集上的 loss 情況）
    tudui.eval()  # 當網路中有 dropout 層、batchnorm 層時，這些層不能起作用
    total_test_loss = 0
    total_accuracy = 0
    with torch.no_grad():  # 沒有梯度了
        for data in test_dataloader: # 測試資料集提取資料
            imgs, targets = data # 數據放到 cuda 上
            if torch.cuda.is_available():
                imgs = imgs.cuda() # 數據放到 cuda 上
                targets = targets.cuda()
            outputs = tudui(imgs)
            loss = loss_fn(outputs, targets) # 僅 data 資料在網路模型上的損失
            total_test_loss = total_test_loss + loss.item() # 所有 loss
            accuracy = (outputs.argmax(1) == targets).sum()
            total_accuracy = total_accuracy + accuracy
```

```
-----第 10 轮训练开始-----
训练次数：7100，Loss：1.3032206296920776
训练次数：7200，Loss：0.9773870706558228
训练次数：7300，Loss：1.1180598735809326
训练次数：7400，Loss：0.8660035133361816
训练次数：7500，Loss：1.2430258989334106
训练次数：7600，Loss：1.2568778991699219
训练次数：7700，Loss：0.898413896560669
训练次数：7800，Loss：1.2846760749816895
整体测试集上的Loss：192.72152745723724
整体测试集上的正确率：0.564300000667572
模型已保存
```

圖 8 訓練過程(截圖畫面)

此外，在尋找資料的過程中，我們進一步理解了許多 人工智慧（AI）領域的重要概

念，例如 歸一化層 (Normalization Layer)、動態 Tanh (Dynamic Tanh)、卷積神經網路 (CNN) 等技術，並學習到這些架構如何相互配合，以構建更完整且高效的 AI 模型。

這些基礎知識的累積，不僅幫助我們強化對 神經網路 的理解，也讓我們能夠在模型訓練時做出更合理的超參數調整，例如選擇合適的 啟動函數 (Activation Function)、優化器 (Optimizer) 和學習率 (Learning Rate)，以提升模型的收斂速度與準確性。此外，這些經驗也為我們後續開發更進階的 AI 應用奠定了穩固的基礎，使我們能夠更靈活地選擇與調整適合的演算法，進一步優化模型效能。

二、 資料收集和標註

(一)資料來源:

我們分別從 Pexels 和 Google 各蒐集了 500 張照片，標準為畫面中必須有人物且能清楚辨識其姿態，並涵蓋不同年齡層與人物類型。作為模型訓練的數據來源。Pexels 提供大量免費且開源的真人照片，確保影像來源真實，並避免使用 AI 生成的內容，提升資料的可靠性。而 Google 則透過不同主題的搜尋方式，幫助我們獲取更多樣化的影像，進一步增加數據集的廣泛性與適用性。這樣的組合讓我們能夠建立一個更具代表性的資料集，確保模型在不同場景與人物條件下都能有良好的辨識能力。

(二)資料標註:

網路上確實有其他標註工具，例如 LabelImg，但該工具需要提前安裝，且安裝步驟較為繁瑣。此外，LabelImg 主要支援矩形標註，無法標註多邊形、點、線等較為複雜的標註類型，這在某些應用場景下可能會有所限制。同時，當處理大量圖片時，LabelImg 的效率較低，且支援的標註格式較為有限，相較之下不如 Roboflow 來得便利。

由於本專案包含近 1000 張圖片，這時候 Roboflow 的優勢便顯現出來。我們只需將影像上傳至 Roboflow 平臺，便能透過其直覺化介面快速進行標註，並能直接匯出標註完成的 data 資料夾，而無需手動撰寫標註程式，也不用擔心儲存空間不足的問題。此外，Roboflow 還提供大量公開數據集，讓我們在練習模型訓練時能直接使用其他研究者整理的標註數據，而在實作時，我們則是自己找圖片是進行標註。

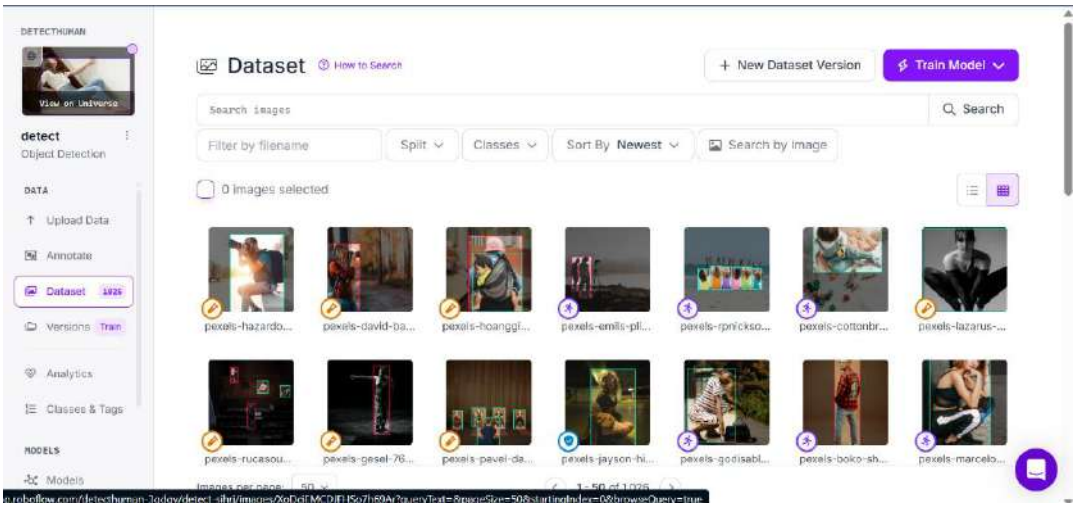


圖 9　Roboflow 標註完成之畫面(截圖畫面)

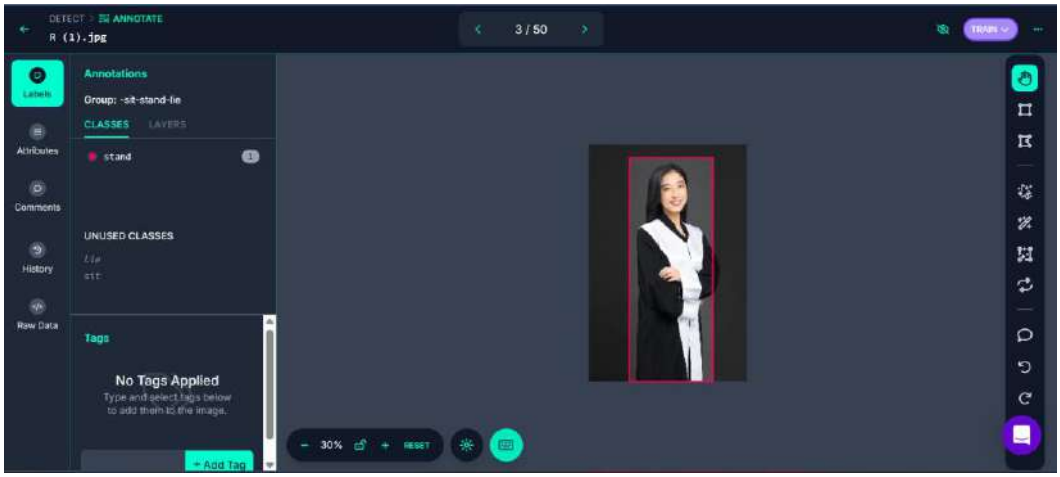


圖 10 roboflow 標註情形(截圖畫面)

三、資料訓練

起初，我們原本計劃使用 PyTorch 來訓練模型，但在嘗試過程中發現 Roboflow 無法直接輸出符合 PyTorch 所需的數據格式，而網路上也缺乏相關的使用教學，使得我們無法順利導入數據進行訓練。因此，我們最終決定改用 YOLO 來訓練本專案的模型，不僅因為 YOLO 在影像辨識領域長期以來表現優異，且其在即時物件偵測上的高效能也更適合我們的需求。

由於筆電硬體資源有限，我們選擇利用 Google Colab 提供的 GPU 來進行模型訓練，藉此提升運算效率並增加訓練次數，以縮短模型收斂所需的時間。此外，我們對照了有無改變參數的訓練方式，如學習率（learning rate）、批次大小（batch size）以及優化器（optimizer），以觀察這些變數對模型結果的影響，最終尋找出最佳的參數組合，以提升模型的辨識準確率與穩定性。

參數	調整前	調整後

Seed(隨機種子)	0	42
Batch(批量大小)	16	21s

表格 1 改變的參數

(一)不修改參數，直接複製官網的程式:

不修改參數的訓練

```
from ultralytics import YOLO

# Load a data YOLO11s model
model = YOLO("yolo11s.pt")

# Train the model on the dataset for 250 epochs
results = model.train(data="../data/data.yaml", epochs=250, imgsz=640)
```

(二)自己設定參數的程式:

修改參數的訓練

```
# Loading a pretrained model
model = YOLO('yolo11s.pt')

# free up GPU memory
torch.cuda.empty_cache()

# Training the model
model.train(data = '../data/data.yaml',
            epochs = 250,
               imgsz = (height, width, channels),
               seed = 42,
              batch = 24,
               workers = 1)
```

訓練的過程如下:

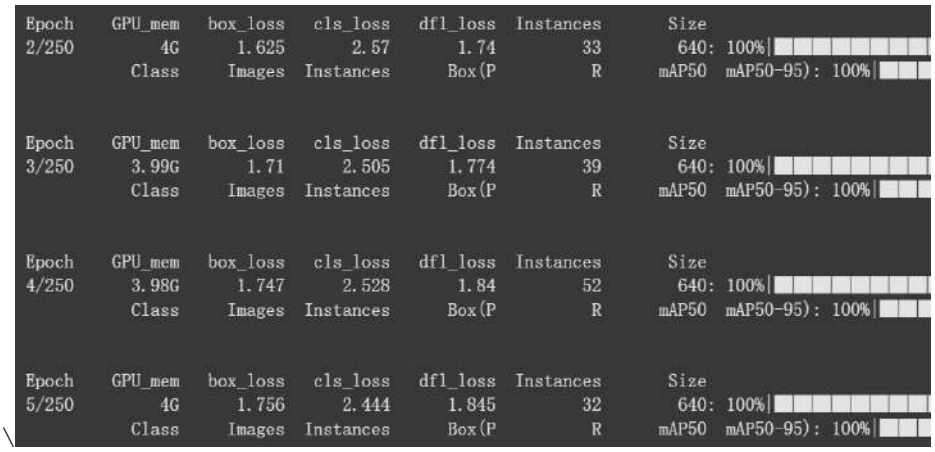


圖 11 訓練過程(截圖畫面)

四、撰寫偵測程式

我們運用了 OpenCV (cv2) 來連接攝像鏡頭，並透過 result 函數匯入訓練好的模型，以進行即時影像辨識。接著，我們使用 result[0].plot() 方法來標示影像中的類別，使其能夠直觀顯示辨識結果。最後，透過 results[0].boxes.data.tolist() 方法獲取偵測到的物體類別，進一步檢查是否屬於 "lie" 類別，以判斷目標是否處於躺臥狀態。這樣的流程使我們能夠即時分析影像內容，並依據辨識結果進行後續處理。

偵測程式

```
for box in results[0].boxes.data.tolist():
    cls = int(box[-1])  # 最後一個數字為類別索引
    # 如果對應的類別名稱是 "跌倒"
    # ，就觸發警報（避免短時間內重複發送郵件）
    if model.names[cls] == "lie":
        detect(result_frame)
```

五、撰寫自動傳送 Mail 的程式

我們用 SMTP 的方法撰寫了一個能夠傳送圖片和文字的程式。實現了透過 Gmail 的 SMTP 伺服器發送電子郵件的基本功能。主要功能包括使用 smtplib 庫來建立與 Gmail 的連接、啟用 TLS 加密傳輸、登入 Gmail 帳戶並發送電子郵件。錯誤處理部分則使用 try...except 捕捉並列印錯誤訊息。

Mail 傳送

```
def img_set(area):
    # 使用 python 內建的 open 方法開啟指定目錄下的檔案
    with open(area, 'rb') as file:
        img = file.read()
    img_file = MIMEApplication(img, Name='detect.jpg')    # 設定附加檔案圖片
    msg.attach(img_file)
def mail_set():
    msg['Subject'] = 'WARNING! 警告'
    msg['From'] = '[redacted]'  # 發送方郵箱
    msg['To'] = '[redacted]'  # 接收方郵箱，可換成其他郵寄位址
    return msg  # 返回郵件物件
def mail_send():
    try:
        mail = mail_set()  # 獲取郵件物件
        smtp = smtplib.SMTP('smtp.gmail.com', 587)  # 連接 Gmail 伺服器
        smtp.ehlo()  # 伺服器握手
        smtp.starttls()  # 啟用 TLS 加密
        smtp.login('[redacted]', '[redacted]')
        smtp.send_message(mail)  # 發送郵件
        smtp.quit()  # 關閉連接
        print("☑ 郵件已成功發送！")
    except Exception as e:
        print(f"❌ 郵件發送失敗: {e}")
```

圖 12 傳送結果(截圖畫面)

六、編寫螢幕擷取程式

我們編寫的程式會在偵測到標註框名稱為 "lie" 時，首先檢查是否存在名為

detect_img 的資料夾，若該資料夾不存在，則自動創建。當資料夾已存在時，程式會使用 OpenCV（cv2）進行截圖並將影像保存至該資料夾。隨後，程式會呼叫 mes_send 函數，將截取的圖像與警報訊息一同傳送，確保即時回報可疑情況，提高系統的警示功能與即時性。

螢幕擷取程式

```
def detect(frame):
    global img_count, last_mail

    # 取得目前時間
    currnt_time = time.time()

    if currnt_time - last_mail > mail_interval:
        img_name = f"detect_{img_count}.jpg"
        cv2.imwrite(f"{area_save}/{img_name}", frame)
        area = f"{area_save}/{img_name}"
        img_set(area)
        mail_send()  # 發送郵件
        last_mail = currnt_time  # 更新最後發送時間
        img_count += 1
```

七、程式連接

我們將三個程式模組合在一起。系統啟動後會自動開啟攝像頭並持續進行人物偵測。一旦偵測到物件標註資訊顯示為 'lie'（代表跌倒事件），系統便會在符合時間間隔規則下即時擷取當前畫面，並將圖片儲存到預先建立的名為 detect_img 的資料夾中。接著，mail 程式會讀取該資料夾中的圖片，並連同預設的警示文字一起發送給使用者，從而達到及時警報的效果。

並且確認彼此之間能正確達到預期效果

修改後的偵測程式

```
from mes_send import mail_send
import time
import cv2
import os

area_save = "detect_img"
img_count = 0
last_mail = 0  # 記錄最後一次發送郵件的時間（時間戳記）
mail_interval = 15  # 設定最小郵件發送間隔（秒）
```

```
# 確認目錄是否存在，若不存在則創建
if not os.path.exists(area_save):
os.makedirs(area_save)

def detect(frame):
global img_count, last_mail

# 取得目前時間
currnt_time = time.time()

if currnt_time - last_mail > mail_interval:
img_name = f"detect_{img_count}.jpg"
img_path = os.path.join(area_save, img_name)

# 嘗試存儲圖片
if cv2.imwrite(img_path, frame):
print(f"☑ 照片已成功存儲: {img_path}")
mail_send(img_path)  # 發送郵件
last_mail = currnt_time  # 更新最後發送時間
img_count += 1  # 更新圖片計數
else:
print(f"❌ 照片存儲失敗: {img_path}")
```

修改後的 mail 傳送

```
import smtplib
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

# HTML 格式的警告信件
html = '''
<h1 style="color:red">⚠ 警告! 警告! 有人跌倒了</h1>
<h1 style="color:red">⚠ WARNING! SOMEONE IS FALLING</h1>
'''


def img_set(msg, area):
    # 使用 python 內建的 open 方法開啟指定目錄下的檔案
    with open(area, 'rb') as file:
        img = file.read()
```

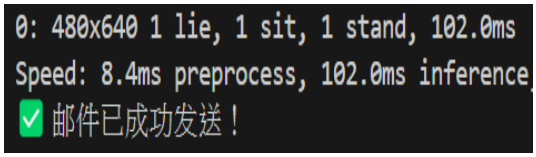
```
    attach_file = MIMEApplication(img, Name='detect.jpg')    # 設定
附加檔案圖片
    msg.attach(attach_file)

def mail_set(area):
    # 每次創建新的 MIMEMultipart 對象
    msg = MIMEMultipart()
    mail_content = MIMEText(html, 'html', 'utf-8')
    msg.attach(mail_content)

    # 設定郵件標頭
    msg['Subject'] = 'WARNING! 警告'
    msg['From'] = '[redacted]'  # 發送方郵箱
    msg['To'] = '[redacted]'  # 接收方郵箱

    # 添加圖片附件
    img_set(msg, area)
    return msg  # 返回郵件對象

def mail_send(area):
    try:
        msg = mail_set(area)  # 獲取郵件對象
        smtp = smtplib.SMTP('smtp.gmail.com', 587)  # 連接 Gmail 伺
服器
        smtp.ehlo()  # 伺服器握手
        smtp.starttls()  # 啟用 TLS 加密

        smtp.login('[redacted]', '[redacted]
[redacted])
        smtp.send_message(msg)  # 發送郵件
        smtp.quit()  # 關閉連接
        print("☑ 郵件已成功發送！")

    except Exception as e:
        print(f"❌ 郵件發送失敗: {e}")
```

八、程式新增

我們將偵測、圖片擷取與 E-mail 發送分為不同模組，但在整合時遇到連接失敗、圖片儲存錯誤及函數調用錯誤等問題。經過與老師討論並參考 AI 建議，我們進行以下優化：

(一)加入時間檢測

為了避免因目標長時間倒地而導致程式連續發送 E-mail，進而造成郵件發送失敗或程式崩潰，我們決定加入發送間隔機制，並將間隔時間設為 30 秒。之所以為 30 秒的原因是一開始設 15 秒的時候，時間還是太短無法有效使用，而 60 秒又太長，於是我們決定改成 30 秒，經過測試好結果還不錯。

修改前	修改後
 (截圖畫面)	 (截圖畫面)

經過調整後，修改前因平率過短導致程式不段崩潰，使郵件無法發送，而增加時間限制後讓程式每次都能成功運行，明顯改善了原本程式容易崩潰的問題，提升了系統的穩定性與可靠性。

(二)調整函數順序

確保圖片儲存成功後才發送 E-mail，避免發生指傳送了文字或傳不出去的錯誤，使使用者收到訊息時才能去判斷。

修改前	修改後
 (截圖畫面)	 (截圖畫面)

能看出原本無法傳送圖片，但重新調整順序後，就能成功傳送圖片

(三)增加註解

使錯誤發生時能盡快找出並處理，且在調整參數時，不會不知如何去調，提高程式穩定性。

最終，系統能穩定偵測人物跌倒，擷取影像並依規則發送 E-mail，減少錯誤並提升使用體驗。

肆、 研究結果

一、訓練結果

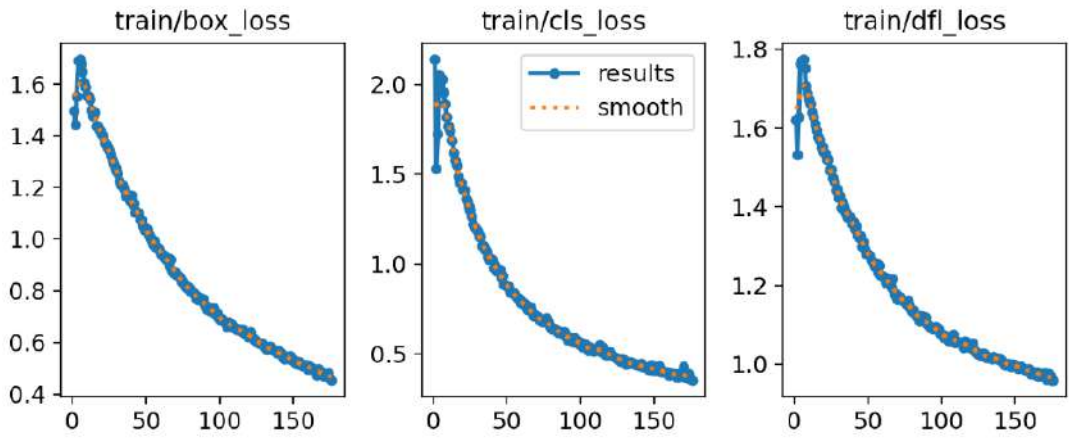


圖 13 方法 1 訓練過程 loss 值（截圖畫面）

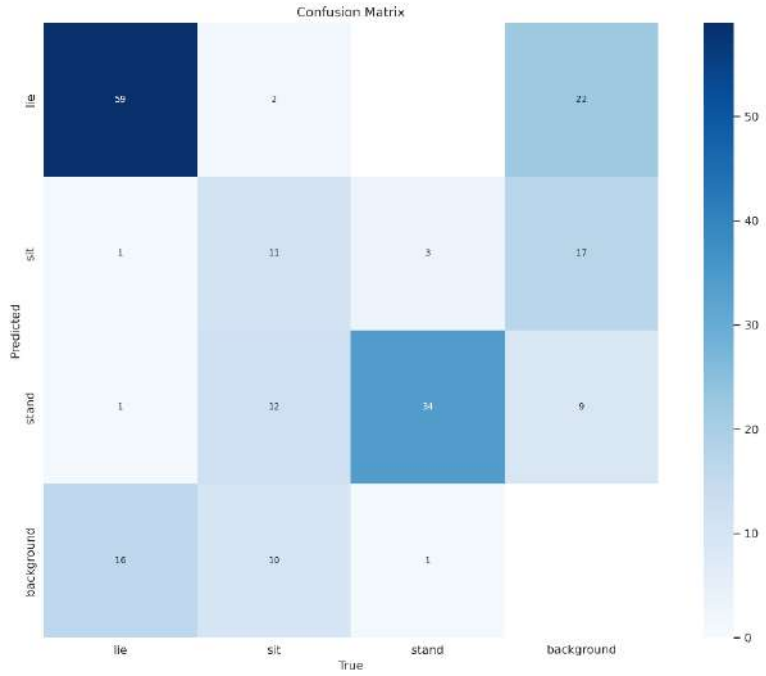


圖 14 方法 1 的混淆矩陣（截圖畫面）

圖 15 為方法二訓練後所得的混淆矩陣，其色彩深淺反映了數值的大小，矩陣中的數字則表示真實標籤被預測為各類別的次數。從中可以觀察到模型對各類別的辨識效果如下：

- lie（躺下） 的真實標籤被正確預測了 59 次，準確率較高。

- sit（坐下） 的真實標籤被正確預測了 11 次，模型對此動作的識別表現尚可。。
- stand（站立） 的真實標籤被正確預測了 34 次，辨識能力中等。

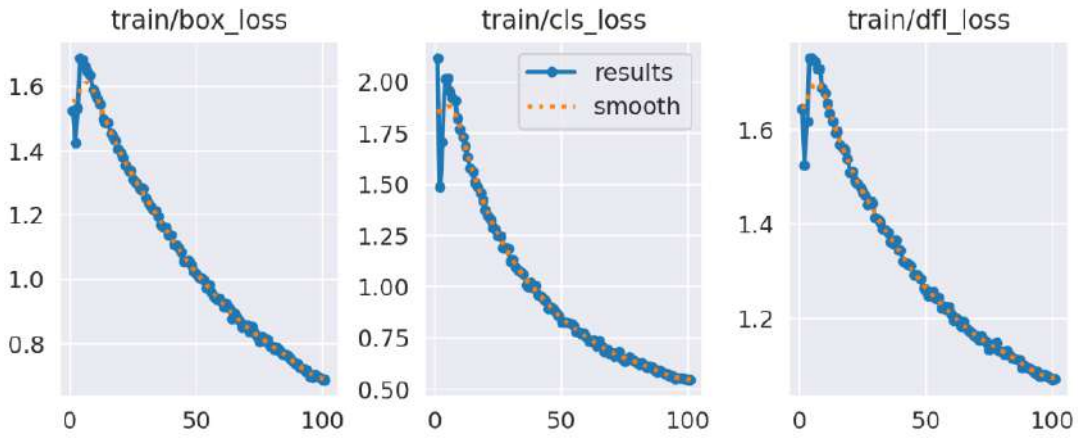


圖 15 方法二訓練過程 loss 值（截圖畫面）

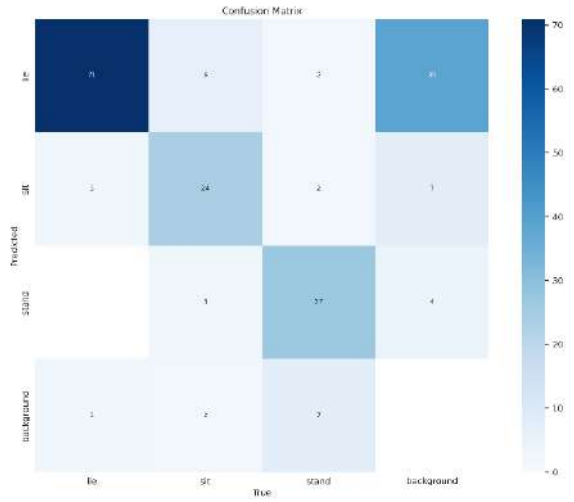


圖 16 方法二的混淆矩陣（截圖畫面）

圖 17 為方法二訓練後所得的混淆矩陣，其色彩深淺反映了數值的大小，矩陣中的數字則表示真實標籤被預測為各類別的次數。從中可以觀察到模型對各類別的辨識效果如下：

- lie（躺下） 的真實標籤被正確預測了 71 次，準確率較高。
- sit（坐下） 的真實標籤被正確預測了 24 次，辨識能力中等。
- stand（站立） 的真實標籤被正確預測了 27 次，模型對此動作的識別表現尚可。

二、偵測精準度

經過多次測試後，我們觀察到當模型訓練次數設定為 250 次時，無論是否調整參數，損失值（loss 值）皆呈現顯著的下降趨勢。然而，進一步分析結果發現，方法一的最終分類損失（Cls_loss）可收斂至約 0.35 左右，而方法二則僅收斂至約 0.39。儘管如此，從實際運行的結果(表 2)來看，方法二的準確率反而更高。由此可知，loss 值的高低並不必然代表模型的偵測準確率，兩者之間可能存在落差，因此反而需要用實際比較來進行判斷。

Box_loss（框回歸損失）	數值越小，模型位置定位越準
Cls_loss（類別分類損失）	數值越小，類別區分越準
Dfl_loss（損失函數）	數值越小，偵測精度越高

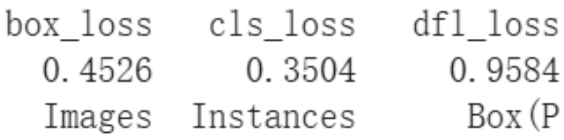


圖 17 方法一的最終結果(截圖畫面)

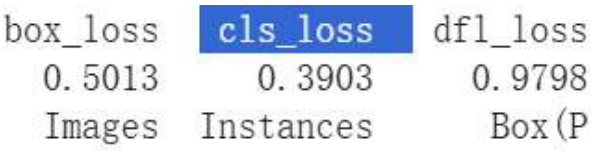


圖 18 方法二的最終結果(截圖畫面)

方法 1	方法 2
	
（截圖畫面）	（截圖畫面）
	
（截圖畫面）	（截圖畫面）

表格 2 兩種方法之比較

本實驗結果突顯了超參數調整對提升模型效能的關鍵作用。即使是像學習率、批次大

小、正則化強度等看似微小的調整，也可能對最終模型的表現產生顯著影響，並進一步凸顯了 loss 值與準確率之間的差異。透過系統性地進行超參數調整，我們能更精確地掌控模型的學習過程，使其更有效地從訓練資料中提取有用特徵，進而提升模型的準確性與泛化能力。

另外我們也把其圖片分類改成只有跌倒和沒跌倒兩種分類，並作比較，結果如下:

標註方式	圖片 1	圖片 2
單一標籤分類（跌倒/未跌倒）	 （截圖畫面）	 （截圖畫面）
多階段姿態標註（站立／行走／跌倒）	 （截圖畫面）	 （截圖畫面）

結果顯示反而是改變前的準確率較高，推測是因為照片數量不一樣。

三、程式流程

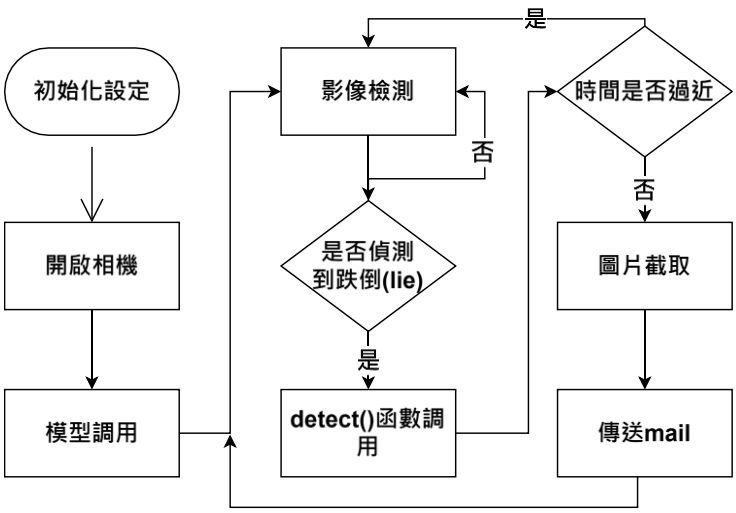


圖 19 程式流程圖(自製)

四、程式結果

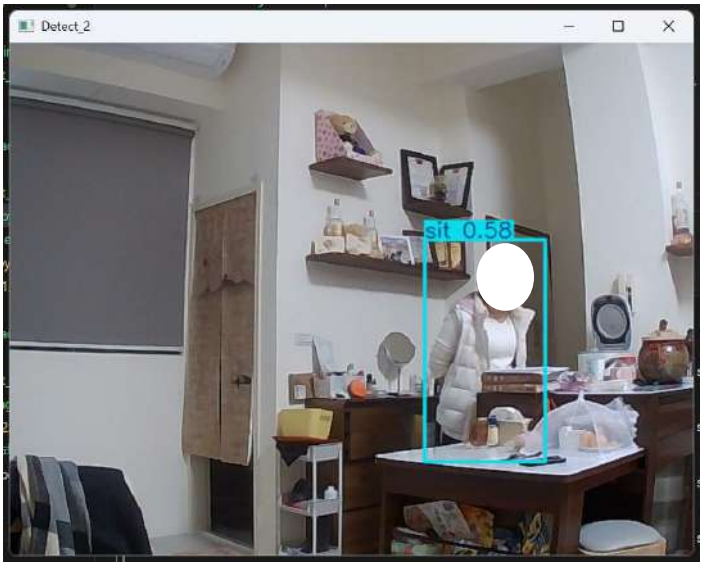


圖 20 偵測視窗(截圖畫面)

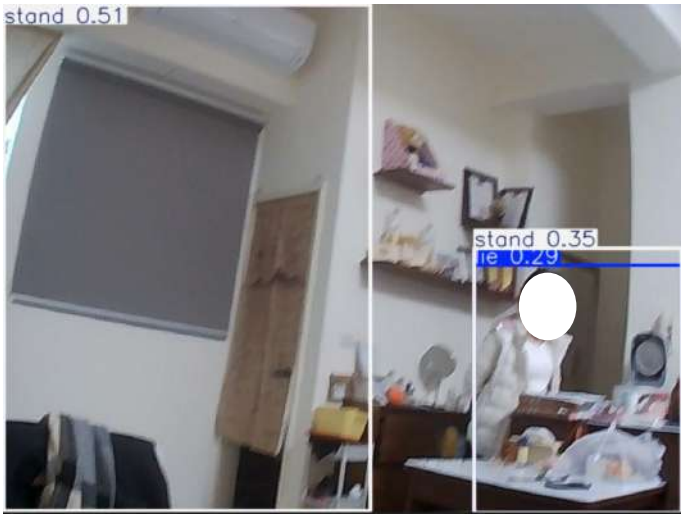
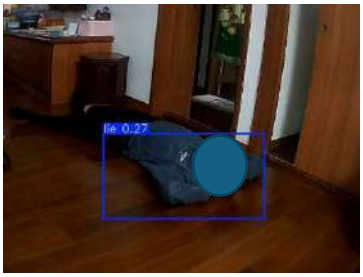


圖 21 程式擷取圖片(截圖畫面)

圖 22 mail 通知結果(截圖畫面)

(一)沒做任何改變

第一次	第二次	第三次
		

表格 3 未做改變之結果（截圖畫面）

(二)不同亮度

沒開燈		
第一次	第二次	第三次
		
開燈		
第四次	第五次	第六次

表格 4 不同亮度之結果(截圖畫面)

(三)不同距離

遠距離		
第一次	第二次	第三次
		
近距離		
第四次	第五次	第六次
		

表格 5 不同距離之結果(截圖畫面)

從上方表格可知，儘管模型整體準確度較高，但在無光照的情況下仍無法正確辨識目標，導致使用者需額外提供光源以輔助偵測。此外，系統仍可能出現重複偵測的情形，因此在郵件通知中附上擷取影像便顯得格外重要。透過影像輔助，使用者可依據實際畫面進行判斷，進一步決定是否需採取行動或進行通報。此舉不僅提升了系統的實用性，也有助於使用者在真實場景中做出更準確的決策。

五、CPU 運行效能

CPU 配置：11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

經測試，CPU 使用率僅 12%~14%，顯示程式運行時對處理器的負擔較低，仍有充足的運算資源可供使用，確保系統穩定運行並具備擴展性。

CPU

（截圖畫面）

伍、 討論

一、人物重複辨識

在本研究中，雖然 loss 的值已經足夠低了，但由於某些動作之間具有高度相似性，因此容易出現誤判的情況，例如將「坐下」與「站起」重複辨識，或將「躺下」與「站立」混淆，導致辨識結果不夠準確。

二、檢測不出人物跌倒

在進行模型訓練時，我們發現有幾次明顯可以看到影像中的人物趴在地上，但偵測結果卻顯示為坐姿。這可能是由於圖片數量不足，或者在資料標註過程中，許多坐姿與趴姿具有相似性，導致模型無法正確區分這兩種姿勢。為瞭解決這個問題，我們計劃在增加資料量的同時，注意避免圖片之間的相似性過高，並考慮將標註的類別數量從三類縮減為兩類，即「跌倒」和「站起」，以此來減少誤判的情況。這樣的調整可以讓模型專注於更簡單的區分，從而提高識別準確性，減少類似問題的發生。

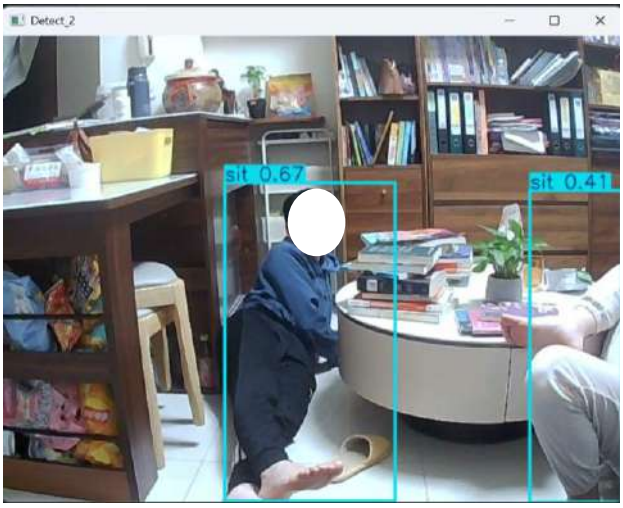


圖 23 檢測失敗之範例(截圖畫面)

三、鏡頭盲區

在網路上的定義中，鏡頭盲區（Camera Blind Spot）指的是「無法有效拍攝或辨識的區域」。

在本專案中，鏡頭的視線可能會受到障礙物（如桌子、衣櫃等）的阻擋，導致部分區域無法被有效監測，如下示意圖所示：

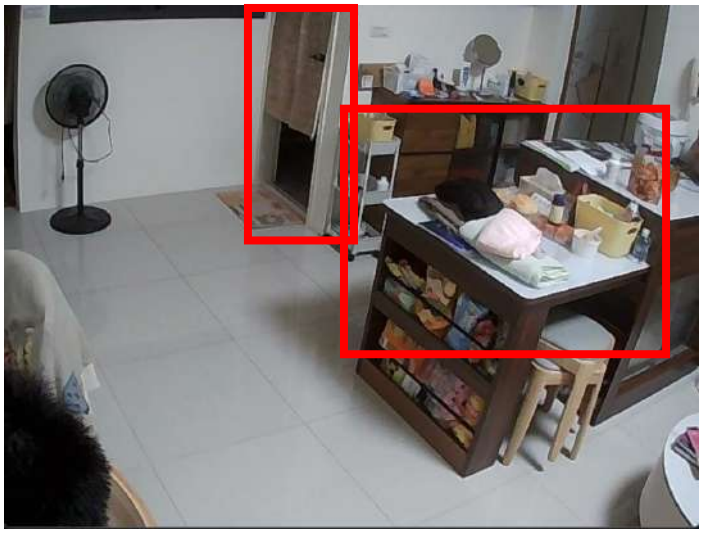


圖 24 鏡頭盲區(截圖畫面)

從圖中可以看出，當紅框內的物件遮擋視線時，鏡頭將無法偵測到其後方的區域。如果人物進入該範圍並不慎跌倒，模型可能無法正確辨識其姿勢或動作，影響跌倒偵測的準確性。

目前，針對鏡頭盲區的主流解決方案是透過多鏡頭配置，以擴大監測範圍並減少盲區的產生，從而提升系統的可靠性與準確性。

四、訊息傳送

起初，我們的構想是透過 LINE notify 來建立一個聊天機器人，藉此向使用者即時傳送訊息，畢竟目前大部分人都在使用 LINE 平臺。然而，在上網搜尋相關資料並進行討論後，我們發現 LINE notify 已經停止服務，無法再用於我們的應用場景。基於此情況，我們隨即考慮自行開發一款 APP，讓使用者可以直接調閱攝影鏡頭所拍攝的影像，進一步提升互動性與便利性。

但在後續的團隊討論中，由於開發時間和資源有限，我們認為自行開發 APP 的計畫在目前的階段難以實現。最終，我們選擇採用電子郵件傳送訊息作為臨時方案，期望藉由郵件通知來提醒使用者。然而，經過初步測試後發現，電子郵件可能會因使用者的設定而無法觸發彈出式提示，導致使用者可能無法及時收到通知，這對於即時性要求較高的應用來說是一個明顯的缺點。

因此，在後續的研究工作中，我們計劃再次投入精力，開發一款專用 APP，以完善整個系統的通知功能，確保訊息能夠及時且直觀地傳達給使用者，從而進一步提升整體應用的實用性與用戶體驗。

陸、 結論

一、在建立資料集的初期階段，老師建議我們每個人先收集約 500 張圖片，這樣便能夠開始

進行初步的模型訓練與測試，並觀察模型在初期階段的表現。老師的建議在當時非常實用，因為這樣可以讓我們在資料量尚未龐大的情況下，先驗證模型的可行性與基本性能。隨著專案的推進，我們在後續的資料收集工作中，持續補充並豐富資料集，最終在完成初步測試後又額外收集負樣本圖像約 200 張左右圖片，從而使得整個訓練資料集的數量達到了 1200 張左右。未來，我們還計劃逐步擴充資料集，以期在模型訓練中獲得更穩定且精確的辨識效果，進一步提升系統的整體效能。

二、模型

本報告只有草草的實作了機器學習模型參數的變動對姿勢辨識性能的影響。研究結果能大概地看出參數調整的影響。此外，經過多方的資料探討研究，發現大型、多樣化和高質量的訓練數據集對於實現準確且穩健的姿勢辨識模型至關重要，能對模型偵測有很好地最大限度地減少誤判錯誤。未來的研究方向會更加深入瞭解不同參數對模型的影響，並找出最合適的參數去訓練模型，使模型達到其所能夠達到的最佳性能。

三、未來展望

(一)姿勢訓練資料集

本研究後續欲找出更多年齡層、不同身高與體重之若干名男性與女性的照片，收集更大量與多元之標記後姿勢訓練資料以提高模型精準度。

未來期待能因姿勢數據集的擴大，使模型精準度也水漲船高，讓誤測的機率減少。

(二)結語

未來，我們希望這個專案能夠普及到各個家庭，不僅僅是針對家中長者在無人照護的情況下，能夠及時發出跌倒警示，防止不幸事件的發生。同時，我們也希望這項技術能夠擴展至保護兒童與成年人，無論是在日常生活中還是特殊情境下，都能提供及時的安全保障，確保每個家庭成員的健康與安全。

柒、 參考文獻資料

一、圖片來源

(一)圖片來源一

引自 Ultralytics YOLO 文件

(二)圖片來源二

引自 Rapid Object Detection using a Boosted Cascade of Simple Features, 作者：Paul Viola, Michael Jones

(三)圖片來源三

引自 Playing Super Mario Bros with Proximal Policy Optimization，網址: https://brandinho.github.io/mario-ppo/

(四)圖片來源四

引自深度學習筆記 019 池化層 - 愛和九九 - 博客園網站

(五)圖片來源五

引自一文詳解啟動函數 - 知乎 網站

(六)圖片來源六

引自 CV/119_完整模型訓練套路.ipynb at main · AccumulateMore/CV · GitHub

二、政府報告或網站

(一)衛生福利部國民健康署 - 老人跌倒常見的危險因數有那些?

(二)二零一八年非故意損傷統計調查報告書

(三)【半數意外傷害發生在家跌落摔傷讓兒童“最受傷”】 - 水文化 - 專題專欄 - 九江市水利局

(四)8_20240705151133067_彰化縣兒童及少年事故傷害執行情形及策進作為.pdf

三、外部網站:

(一)老人跌倒當心後遺症！浴室、客廳、臥室跌倒處理方式、長照如何幫助跌倒老人？ - 萬安社會福利慈善事業基金會

(二)老人跌倒 4 部位易骨折！跌倒後處理流程？如何自救？防跌 10 招、骨折照護 4 原則 - 良醫健康網

(三)1990 年與 2017 年中國 0~19 歲人群跌倒疾病負擔分析

(四)跌傷

(五)跌倒檢測和識別 2：YOLOv5 實現跌倒檢測(含跌倒檢測資料集和訓練代碼)_yolov5 跌倒檢測-CSDN 博客

(六)【PyTorch 教程】P1. PyTorch 環境的配置及安裝 (Configuration and Installation of PyTorch)

(七)[2410.17725] YOLOv11: An Overview of the Key Architectural Enhancements

(八)【人工智慧】DyT 幹掉 Transformer 歸一化層 | 動態 Tanh | 9 行代碼 | 性能不降反升 | 何愷明楊立昆攜手 | 多項任務驗證 | 計算效率提升一倍 | 打破固有觀念

(九)smtplib — SMTP protocol client — Python 3.13.2 documentation

(一〇) Wind Turbines Object Detection using YOLOv8 | Kaggle

(一一) 串接 Gmail 寄送電子郵件 - Python 教學 | STEAM 教育學習網

(一二) Roboflow

(一三) YOLO11 - NEW - Ultralytics YOLO Docs

(一四) 【youcans 的 OpenCV 常式 300 篇】總目錄-CSDN 博客

(一五) 2503.19501

(一六) AI-Generated Fall Data: Assessing LLMs and Diffusion Model for Wearable Fall Detection

（一七） ElderFallGuard: Real-Time IoT and Computer Vision-Based Fall Detection System for Elderly Safety

四、書籍資料

OpenCV 影像創意邁向 AI 視覺王者歸來初版 - 洪錦魁著 深智數位有限公司出版

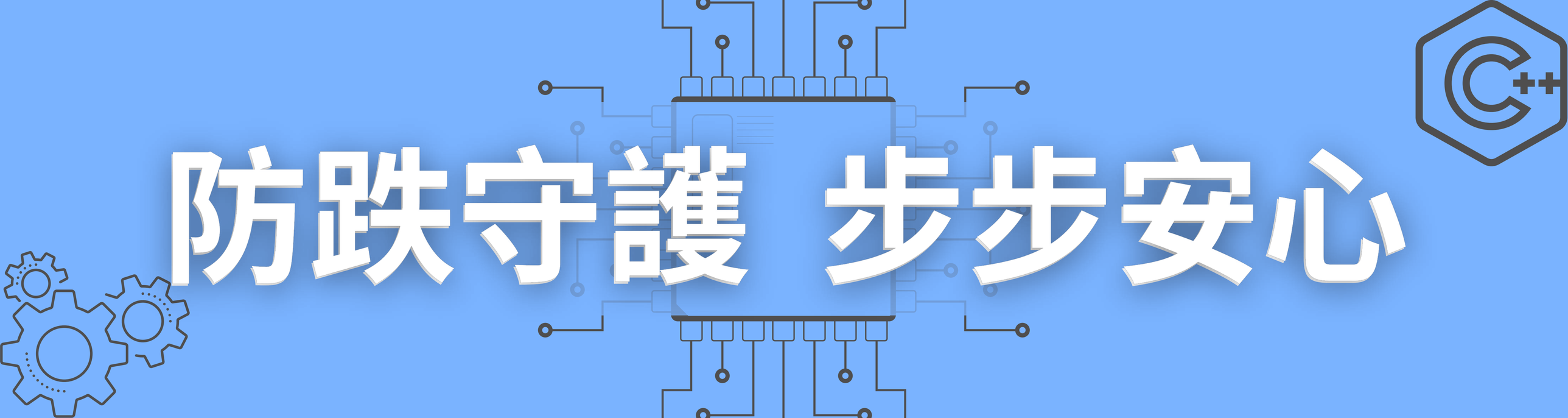
數位元影像處理 Python 程式實作第二版 - 張元翔著 全華出版

【評語】052517

本作品使用 OpenCV 和 YOLO 技術辨識人體動作，檢測居家老人跌倒情況及其影響，以便發送郵件通知家人。老人跌倒偵測與居家照護是很常見的作品方向,建議可更深入分析過去此類研究的不足之處，以便與此類研究有所區隔，並做出具體的貢獻。

本作品主要貢獻在於實作該系統，系統使用了 Pexels 和 Google 收集圖片作為模型訓練資料，並利用 Roboflow 工具加快標註過程。作品最後實驗僅以少數特定情境舉例說明，建議可增加更多實驗以驗證系統穩定性與可靠性。

作品海報


防跌守護 步步安心

壹、摘要

隊員的外婆在台灣獨居，曾因跌倒而未能及時求救，引發家人擔憂，高齡化與獨居老人數量增加，導致安全照護挑戰，需改善意外求助系統。本研究目的是使用OpenCV和YOLO技術辨識人體動作，檢測跌倒情況及其影響，並使用Pexels和Google收集圖片進行訓練，Roboflow工具加快標註過程。過程中整合偵測、影像擷取及郵件發送功能，設計時間間隔機制避免重複發送郵件。訓練結果顯示模型在辨識“躺下”動作上準確率高，對其他動作表現尚可，而系統運行時CPU負擔輕，確保穩定性與擴展性。本系統對人物重複辨識和跌倒檢測準確性有改進空間，考慮增加資料量及調整標註類別。未來計畫開發專用APP以提升訊息傳送功能，增強用戶體驗。

貳、研究目的

(一)利用 OpenCV 和 YOLO 技術辨識人體動作變化，進而檢測是否發生跌倒情況。
(二)測試不同參數之YOLO的結果差別
(三)測試不同標註方式對訓練結果的影響

參、研究設備

一、設備

筆記型電腦， Colab

二、軟體

軟體	python	GPT	DeepSeek	Roboflow	Pexels
套件	time	os	opencv	ultralytics	smtplib
	email.mime	matplotlib	random	seaborn	Pytorch

肆、研究過程&方法

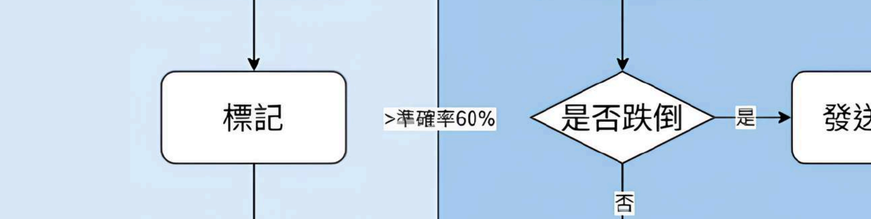
一.架構流程圖

圖片來源：自製

二.影像處理

標註

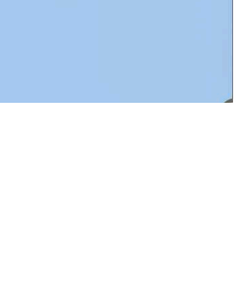
圖片來源：pixabay免費圖片
(圖片來源一)



處理

圖片來源：pexels開放圖庫
(圖片來源二)

匯出



三.模型訓練

我們用了修改參數以及無修改參數的兩種方法來訓練模型。

方法一 ▼

不修改參數的訓練

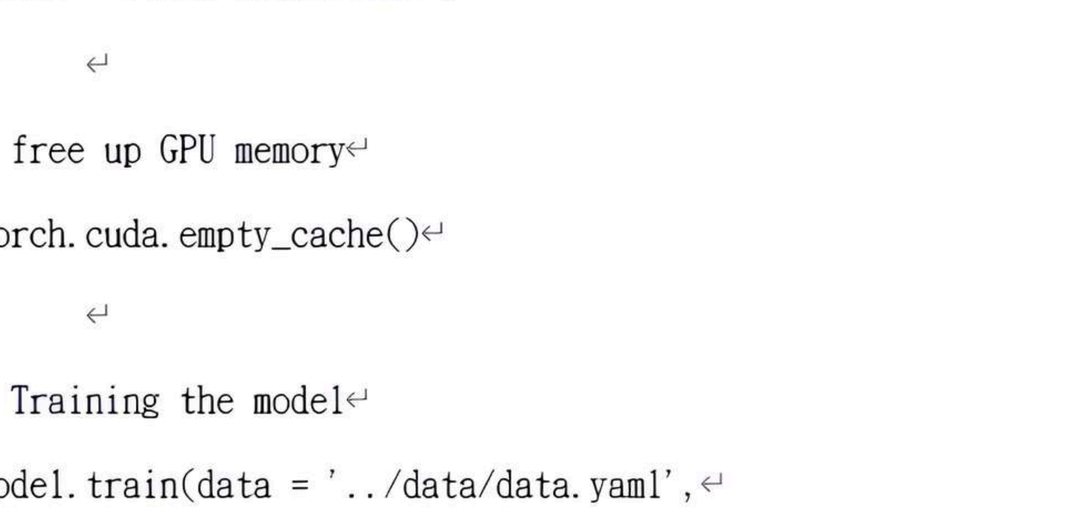
```
from ultralytics import YOLO

# Load a data YOLO11s model
model = YOLO("yolo11s.pt")

# Train the model on the dataset for 250 epochs
results = model.train(data="../data/data.yaml", epochs=250, imgsz=640)
```

方法二 ▼

修改參數的訓練

```
# Loading a pretrained model
model = YOLO('yolo11s.pt')

# free up GPU memory
torch.cuda.empty_cache()

# Training the model
model.train(data = '../data/data.yaml',
            epochs = 250,
            imgsz = (height, width, channels),
            seed = 42,
            batch = 24,
            workers = 1)
```

結果 ▼



圖片來源：截圖畫面

由於對檢測程式來說，最重要的是能否準確檢測人無是否跌倒，因此我們才選擇方法二

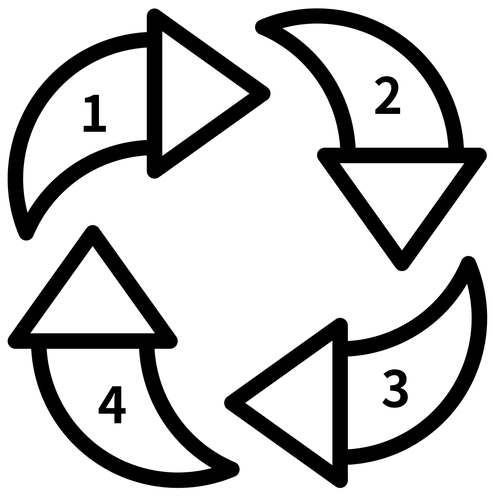
4.程式流程圖

影像獲取

鏡頭獲取環境畫面，持續監控。

Email通知

即時通報家屬或照護者。

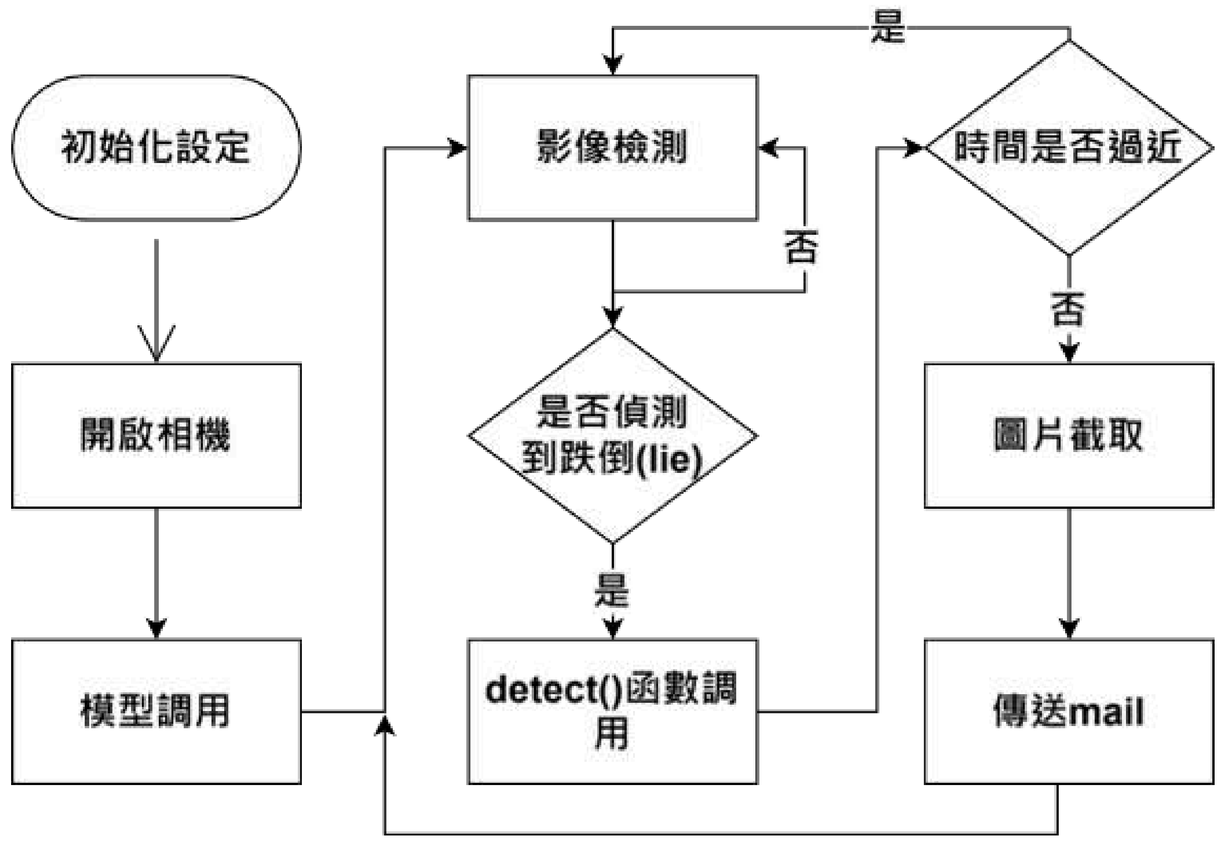


跌倒檢測

運用YOLO即時判別跌倒行為。

畫面擷取

自動擷取相關畫面，用於紀錄與辨識。



圖片來源：自製

影像獲取

使用了OpenCV來連接攝像鏡頭，並用訓練好的模型偵測每偵畫面中人物的姿態，用results[0].boxes.data.tolist() 的方法來獲取人物姿態的資訊，讓程式能知道現在畫面中的人是否跌倒，使程式決定是否需調用「畫面擷取」的函數。

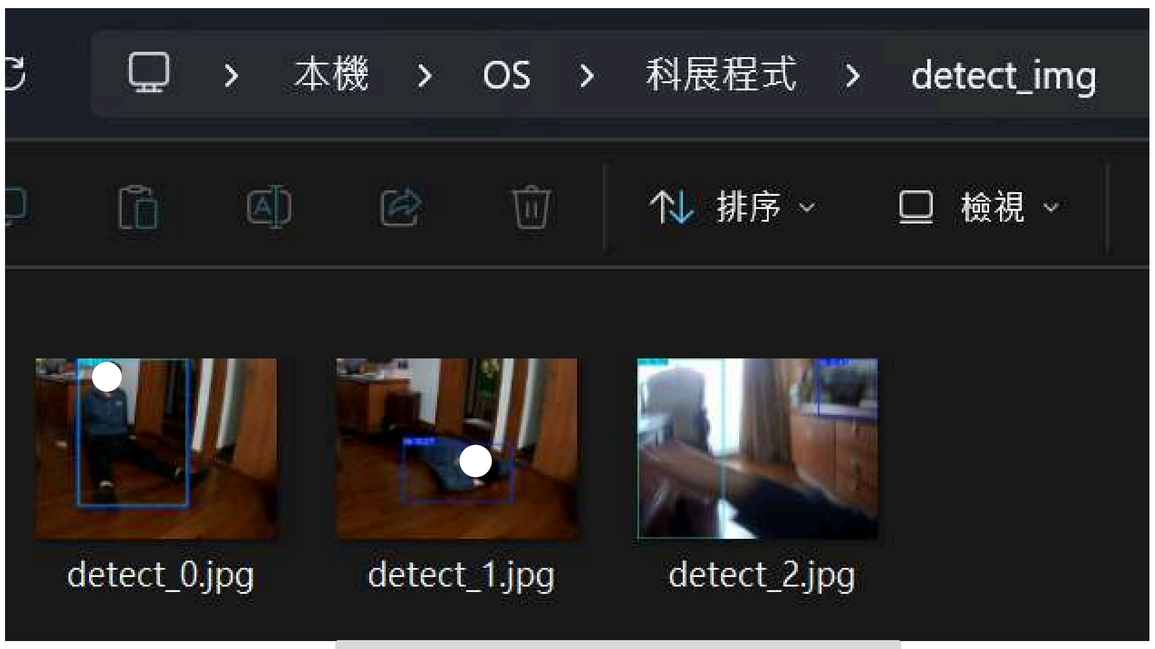
偵測程式

```
for box in results[0].boxes.data.tolist():
    cls = int(box[-1])  # 最後一個數字為類別索引
    # 如果對應的類別名稱是 "跌倒"，就觸發警報
    if model.names[cls] == "lie":
        detect(result_frame)   ← 函數調用
```

程式截圖

當「畫面擷取」的函數被調用時，程式會用OpenCV的imwrite的語法把那一偵的畫面解取下來，並儲存在名為「detect_img」的資料夾中，且每次重啟程式時，也會把資料夾清空避免空間的佔用，接著調用mes_send 函數傳送Email。

影像儲存實例▼



圖片來源：研究者自行實驗並拍攝之截圖畫面

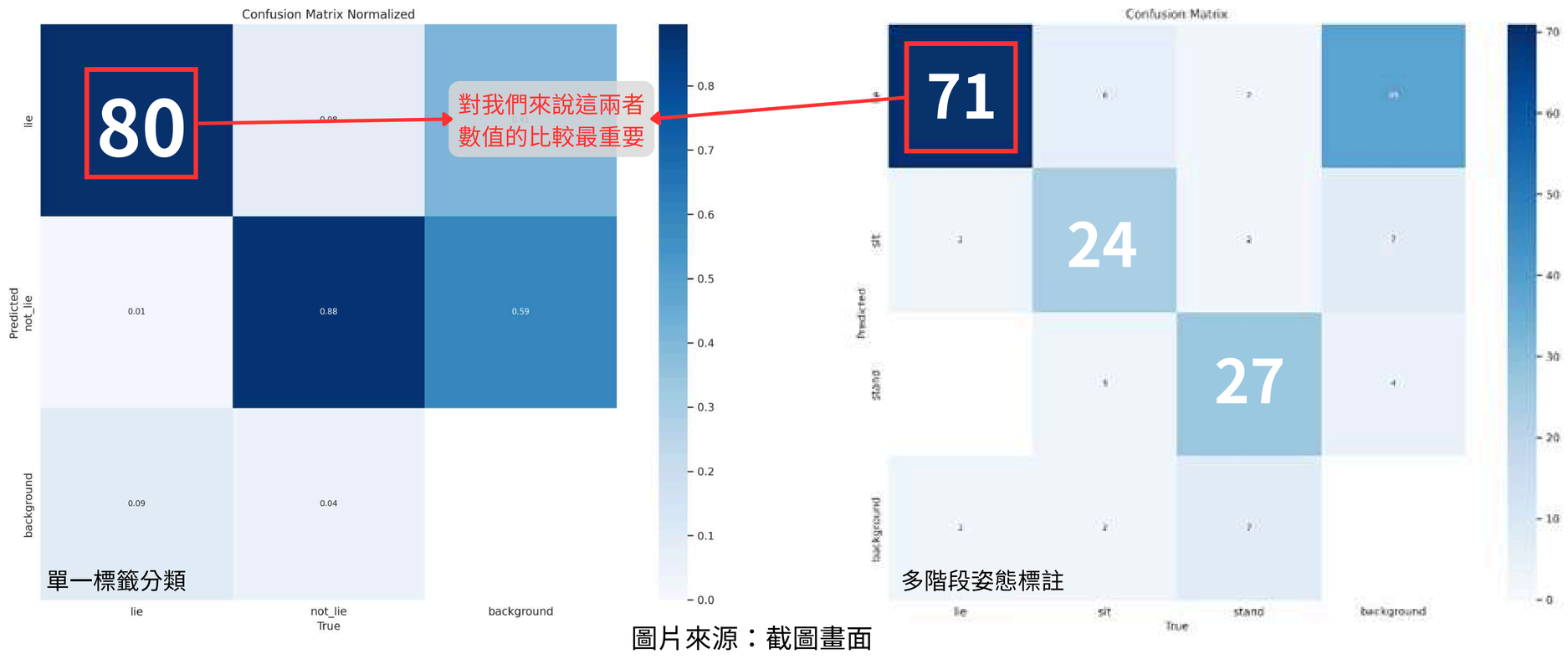
程式會先用os.path.exists來確認資料夾是否存在，如果資料夾不存在就用使用os.makedirs來創建的，而此設計讓程式在之後可以成功的重送圖片，不會發生只傳送文字的問題，且避免了重送錯誤照片的可能性

Email通知

程式接著會傳送Email到使用者的信箱並附上圖片及文字來提醒使用者的同時讓使用者判斷是否通知119。

▲傳送結果

圖片來源：截圖畫面



圖片來源：截圖畫面

接著我們用了兩種標註模型的方式，一種是多階段姿態標註（站立／行走／跌倒）的方式，另一種是單一標籤分類（跌倒/未跌倒），而從上面兩個混淆矩陣來看，單一標籤小果比較好。

伍、結果與討論

1. 從下圖可以看出程式對硬件所需的效能要求不高，能在一些小型設備上運行如樹梅派設備上運行，使設備所佔之空間縮小。

圖片來源：截圖畫面

圖片來源：截圖畫面

2. Email的設計能讓我們傳送文字訊息時，配上圖片提供使用者一個判斷的依據，且Email可以自行設定是否標註，讓我們能夠不用額外花時間學習開發APP。
3. 在標註圖片時，網上提供了兩種方法，一是用labelimg，另外一個是用Roboflow，但因labelimg在使用時出現了錯誤，也找不到原因，因此後來改用較為方便的Roboflow。
4. 在測試中發現單台攝影機會產生死角，推測至少需要 2~3 台互相配合才能覆蓋完整區域。由於模型精度有限，可能發生誤報或漏報，因此設計成只要任一台偵測到跌倒就發送警告，以提高可靠度。
5. 從結果可看出，調整參數後的準確率明顯優於使用預設參數時的結果，顯示在模型訓練中進行超參數調整能有效提升效能。

陸、結論

1. 一開始使用100張照片驗證了模型可行性與基本性能，最後收集了1200張照片豐富資料集，未來計畫持續擴充資料集增加整體效能
2. 此次只有實作機器學習模型參數的變動對姿勢辨識性能的影響，並發現大型、多樣化和高質量數據訓練集的重要

柒、展望

期待未來此程式能幫助很多獨居老人的安危，避免跌倒之後沒人知道的危險性，讓許多憾事不再產生。未來嘗試增加警保解除的功能。

捌、參考文獻

一、圖片來源
(一). 圖片來源一
引自Pixabay，網址：https://pixabay.com/zh
(二). 圖片來源二
引自Pexels，網址：https://www.pexels.com/zh-tw/
二、其他文獻
(一)Vinayak Mali& Saurabh Jaiswal – Pose-Based Fall Detection System: Efficient Monitoring on Standard CPUs
(二)AI-Generated Fall Data: Assessing LLMs and Diffusion Model for Wearable Fall Detection
(三)ElderFallGuard: Real-Time IoT and Computer Vision-Based Fall Detection System for Elderly Safety