

**National Sun Yat-sen University**  
**Department of Electrical Engineering**

**112-2 機器學習系統設計實務與應用**

**HW5 Jetson-Nano Object Detection**

**組別：你說的隊**

**組員：B103012034 黃麗穎**

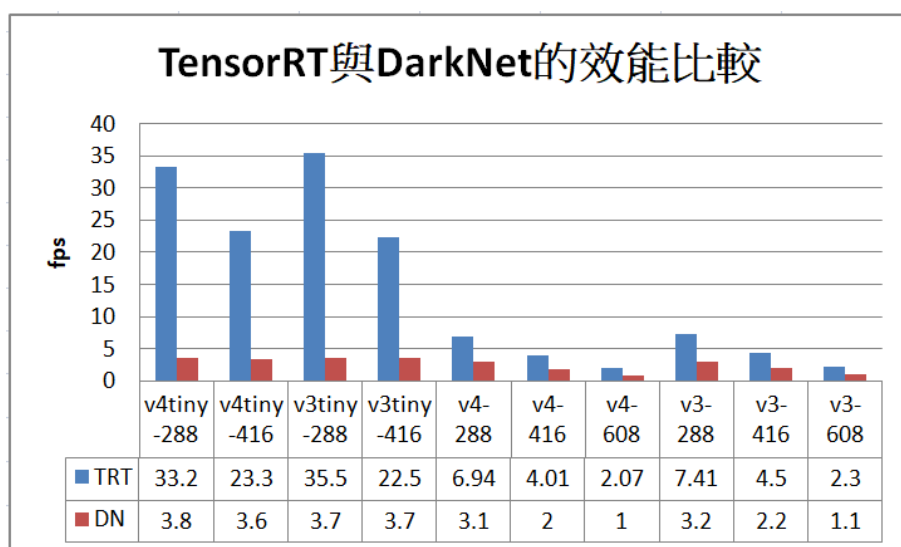
**B103012041 黃詣辰**

**B103015006 胡庭翊**

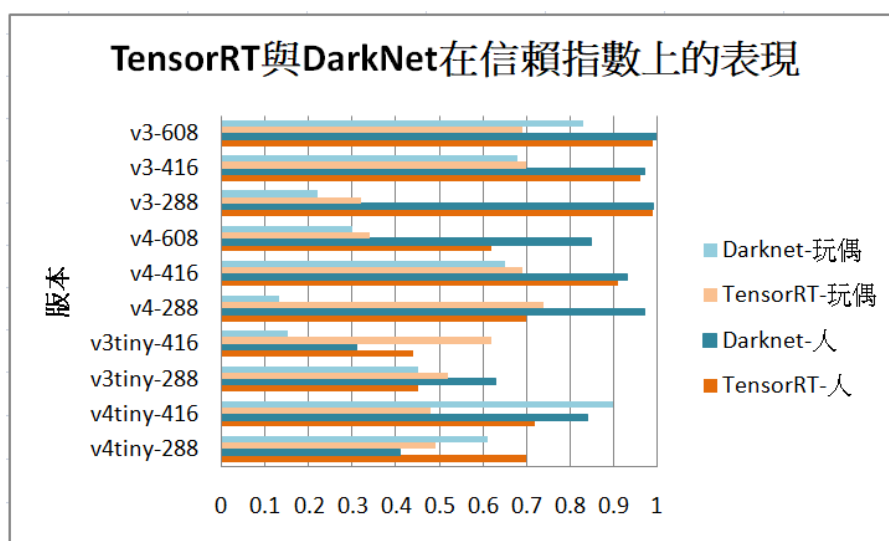
**B103015001 林佳明**

## 一. 第一部分(Darknet 與 TensorRT 的比較)

常見的深度學習框架是 TensorFlow 和 PyTorch，而 YOLO 作者基於 C 和 CUDA 寫了一個相對小眾的深度學習框架—Darknet。Darknet 主要特點在於容易安裝與較輕型，雖然沒有像 TensorFlow 一樣強大的 API，但它的輕巧帶給它一定的靈活性，很好的移植到其他平台上。至於 TensorRT 則是 NVIDIA 推出的一個高性能的深度學習框架，可以讓深度學習模型在 NVIDIA GPU 上實現低延遲，高吞吐量的部署。TensorRT 支持 Caffe，TensorFlow，Mxnet，Pytorch 等主流深度學習框架，其本身是一個 C++庫，並且提供了 C++ API 和 Python API，主要在 NVIDIA GPU 進行高性能的推理加速。



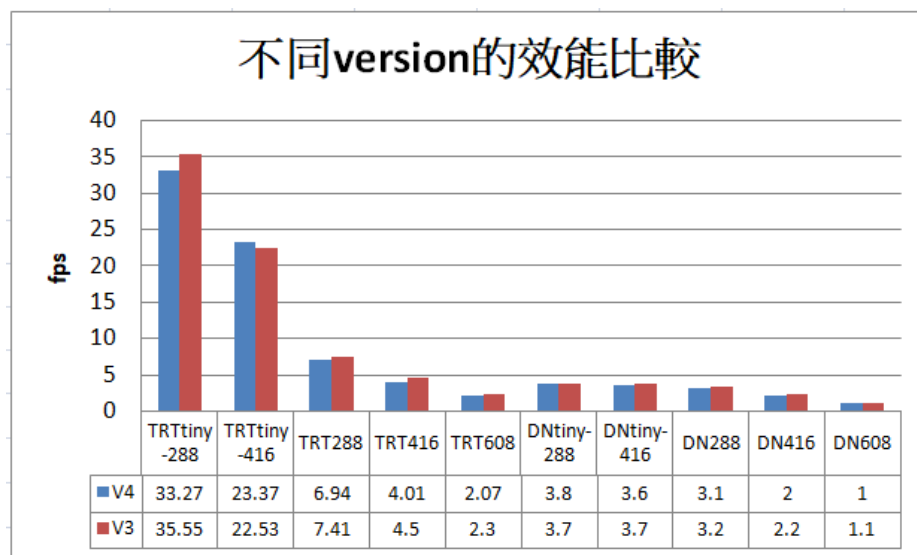
上圖為我們實際在 Jetson-Nano 上進行物件偵測後的實測數據，可以看出 fps 在 TensorRT 的平台下均大於 DarkNet，其原因可能是 TensorRT 相對於 DarkNet 使用了許多優化的技術，使內存使用量減少進而增加每秒影格數。



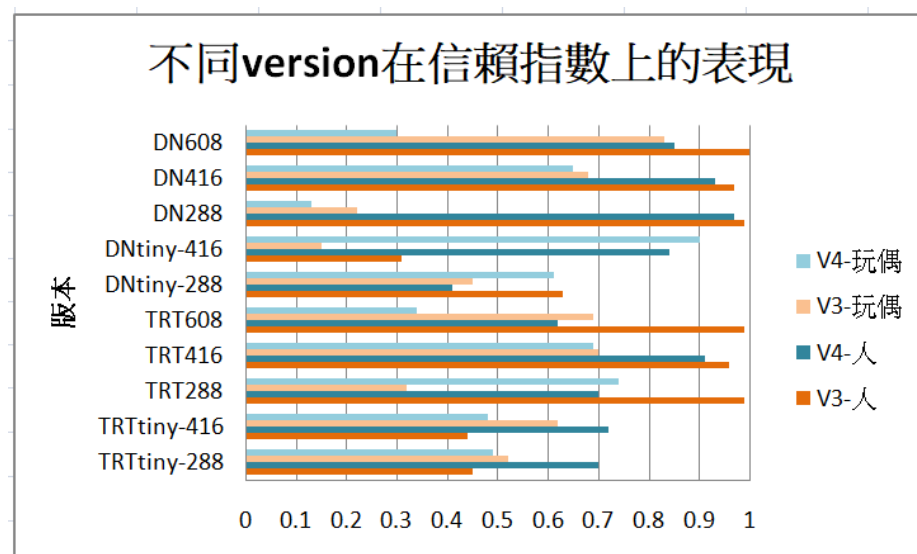
由上面的圖表我們可以觀察到，TensorRT 和 Darknet 偵測人及玩偶在不同模型大小及版本下會略有不同，不過看起來綜合比較兩者的優劣其實差不多。若是將 fps 的大小納入考量，在兩種平台之間做選擇時，應該仍會優先選擇 TensorRT。

本次作業我們將在 Darknet 以及 TensorRT 上面比較不同 version 、 tiny model 、 image size 對於 YOLO 物件偵測情形的差異。在使用 Darknet 執行 YOLO 時不需再另外做轉換，至於使用 TensorRT 則需要先將檔案轉為 ONNX 的通用深度學習模型，然后再對 ONNX 模型做解析。

## 1. Version



在 TensorRT 以及 Darknet 上比較使用 YOLOv3 與 YOLOv4 在原版本、輕型版本，以及各種圖像輸出大小的實際效能可以發現，由上表所示，普遍來說 yolov3 的 fps 比 yolov4 高一點點，也就是說在執行效能上 YOLOv4 略低於 YOLOv3。



至於在準確度的部分，由上圖實作結果可以得知，不管是在 Darknet 或是在 TensorRT 上，YOLOv3 原版模型普遍信賴指數都高於 YOLOv4，而在 tiny 版本

的模型上，反而是 YOLOv4 的平均表現優於 YOLOv3。YOLOv3 到了 tiny 版本信賴指數有著大幅地降低，而 YOLOv4 則相對的較沒甚麼差異。

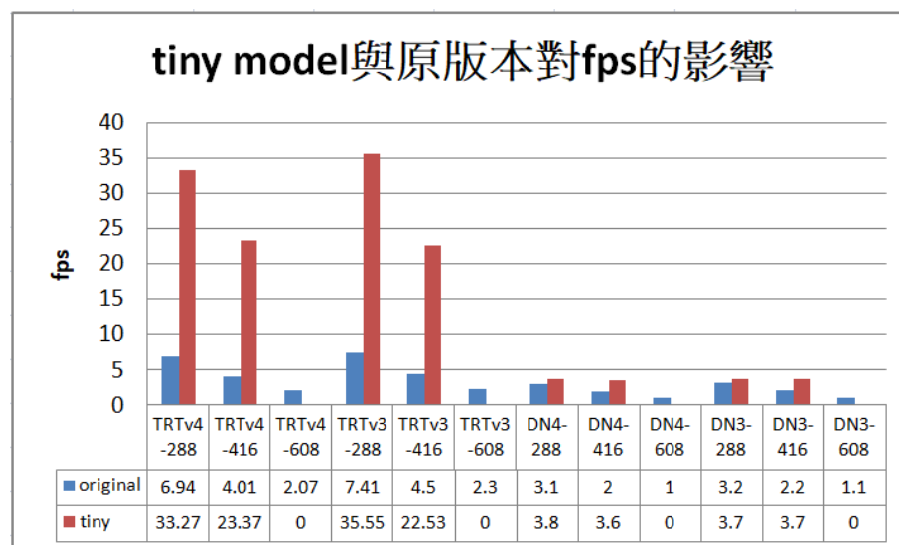
	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully convolution	Fully convolution	Fully convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy	Binary cross entropy and Logits loss function
Neck	FPN	SSP and PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

YOLOv3, YOLOv4, YOLOv5 架構比較(摘自[1])

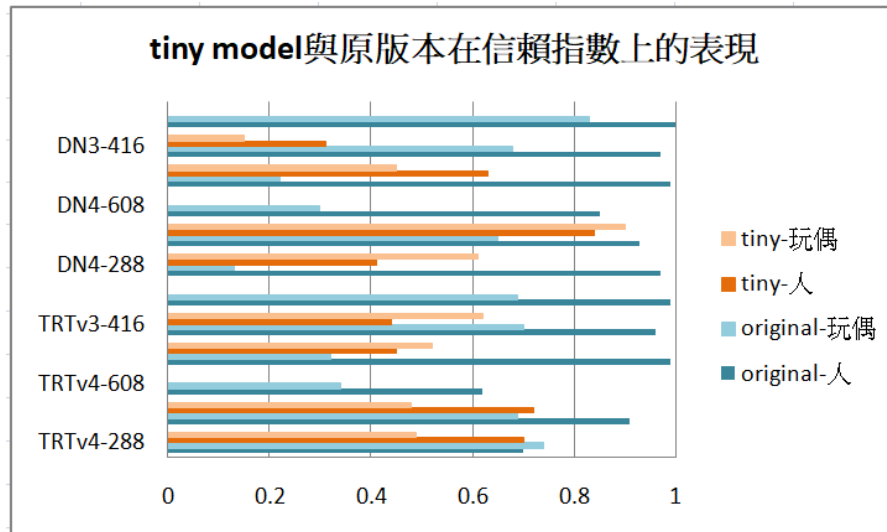
由上表可以看出 YOLOv3 與 YOLOv4 在架構上的差異，綜合前面的實做結果，若不考慮模型大小，我們認為選擇 YOLOv3 較佳，而若有模型大小的需求，需要使用到 tiny 版本的話，使用 YOLOv4-tiny 較佳。

## 2. tiny model

透過模型架構簡化及減少參數使得 yolov4-tiny 比 yolov4 可以更快速的訓練及偵測，這樣的優點有利於將模型使用在行動裝置與嵌入式系統中。然而，tiny 版本的模型只支援大小為 288 以及 416 的圖像，608 的話則沒有辦法使用 tiny model。

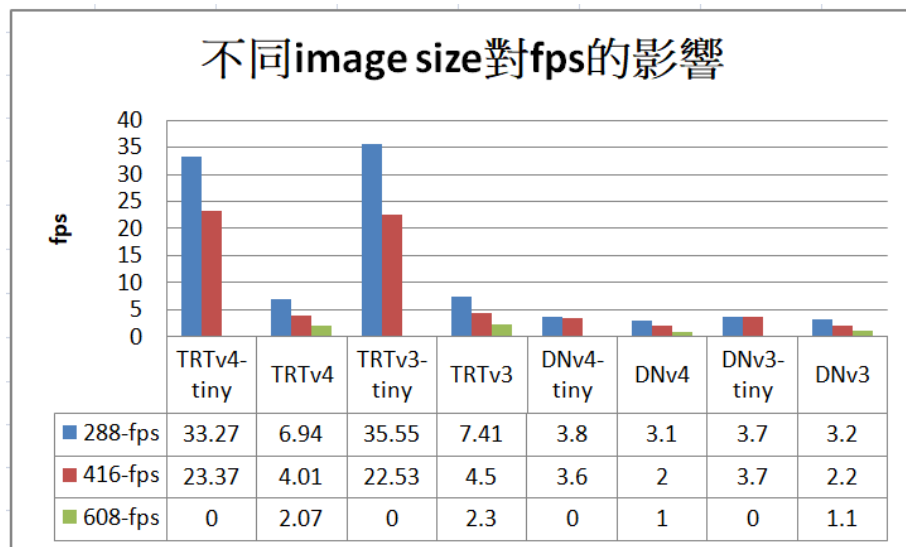


由圖可見，tiny 模型的 fps 均較原版的大。這樣的結果很合理，因為 tiny 模型是原版本的減縮，降低了模型的複雜度和精度使得處理速度可以提高。

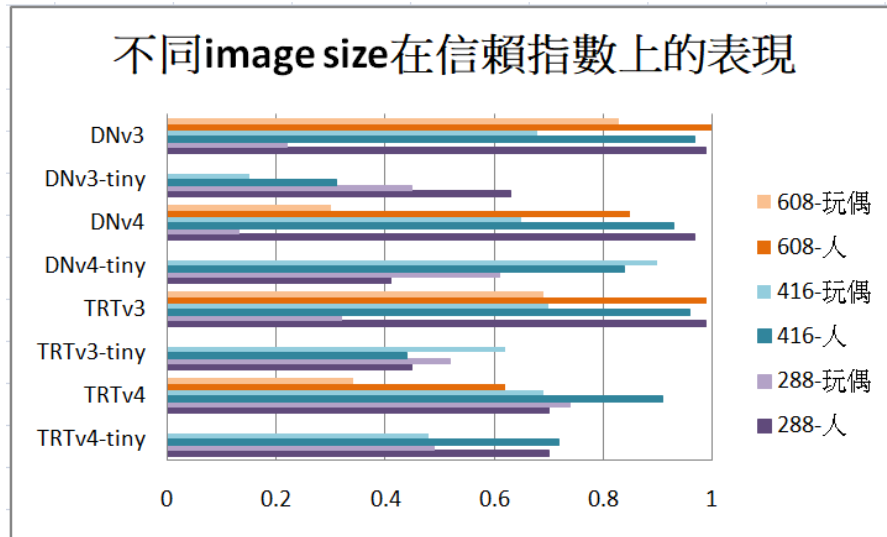


由圖表我們可以看到原版本的信賴指數會比 tiny 模型高。綜合比較 tiny 模型和原版本模型，增加處理速度就會相對需要付出信賴指數下降的代價，因此應該適不同的需要選擇較符合的模型。

### 3. image size



由圖表可以觀察到，fps 的大小會隨著影像維度大小變大而變小。這個結果是合理的，因為越大的影像維度意味著需要處理更多像素，因此幀率會下降。

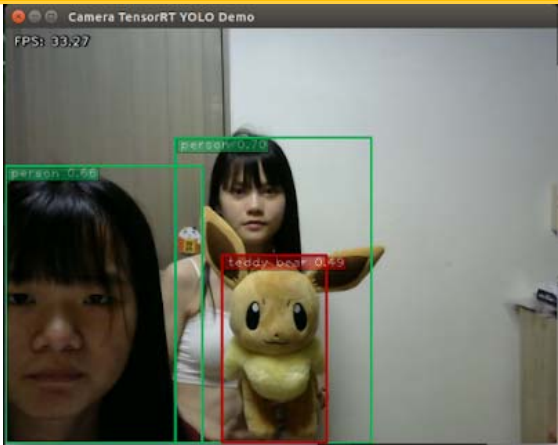
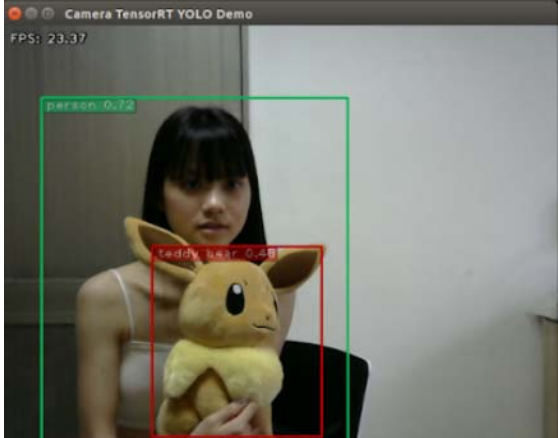
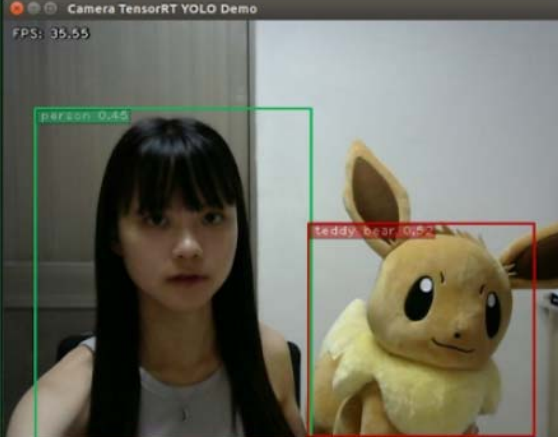



一般來說，較大的影像維度能夠更清晰的呈現細節，因此信賴指數較高，但是此次實作結果的這個現象不明顯。

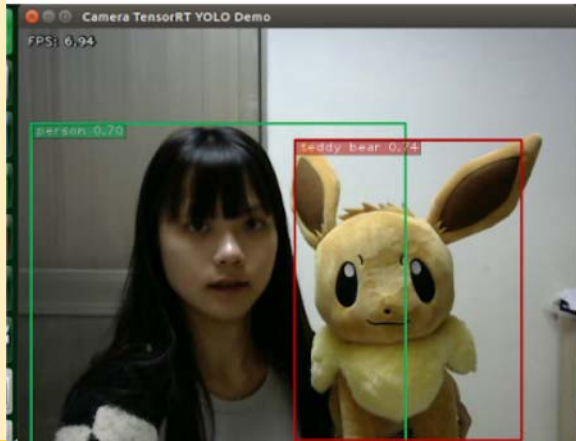
#### 4. 本次實作數據

平台	版本	大小	人	玩偶	FPS
TensorRT	yolov4-tiny	288	0.7	0.49	33.27
TensorRT	yolov4-tiny	416	0.72	0.48	23.37
TensorRT	yolov3-tiny	288	0.45	0.52	35.55
TensorRT	yolov3-tiny	416	0.44	0.62	22.53
TensorRT	yolov4	288	0.70	0.74	6.94
TensorRT	yolov4	416	0.91	0.69	4.01
TensorRT	yolov4	608	0.62	0.34	2.07
TensorRT	yolov3	288	0.99	0.32	7.41
TensorRT	yolov3	416	0.96	0.70	4.50
TensorRT	yolov3	608	0.99	0.69	2.30

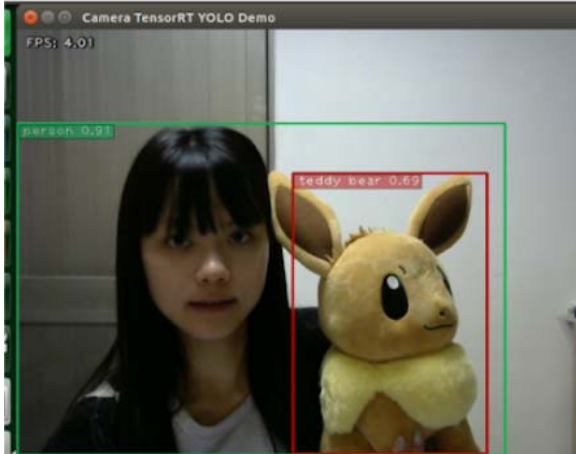
平台	版本	大小	人	玩偶	FPS
Darknet	yolov4-tiny	288	0.41	0.61	3.8
Darknet	yolov4-tiny	416	0.84	0.90	3.6
Darknet	yolov3-tiny	288	0.63	0.45	3.7
Darknet	yolov3-tiny	416	0.31	0.15	3.7
Darknet	yolov4	288	0.97	0.13	3.1
Darknet	yolov4	416	0.96	0.65	2.0
Darknet	yolov4	608	0.85	0.30	1.0
Darknet	yolov3	288	0.99	0.22	3.2
Darknet	yolov3	416	0.97	0.68	2.2
Darknet	yolov3	608	1	0.83	1.1

平台	版本	大小	實作畫面
TensorRT	yolov4-tiny	288	
TensorRT	yolov4-tiny	416	
TensorRT	yolov3-tiny	288	
TensorRT	yolov3-tiny	416	

TensorRT      yolov4      288



TensorRT      yolov4      416



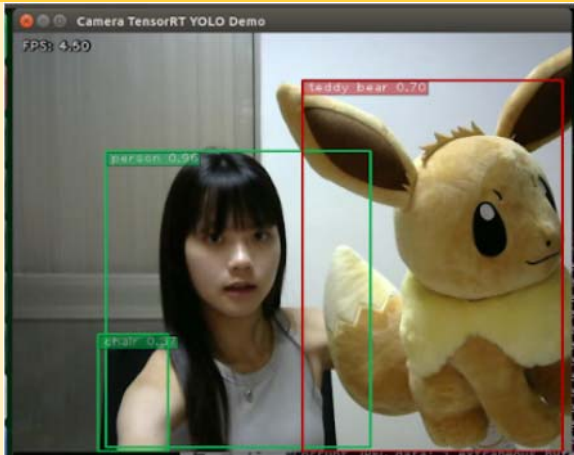

TensorRT      yolov4      608


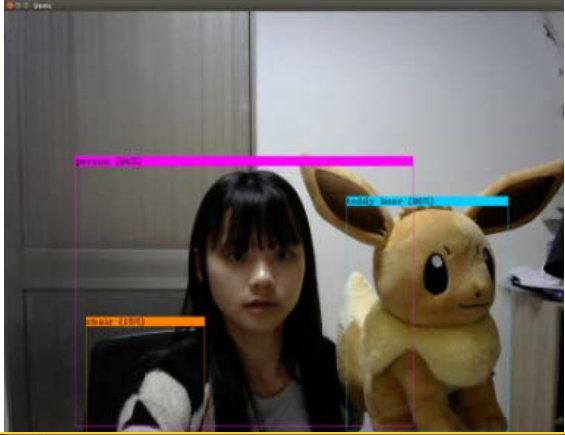

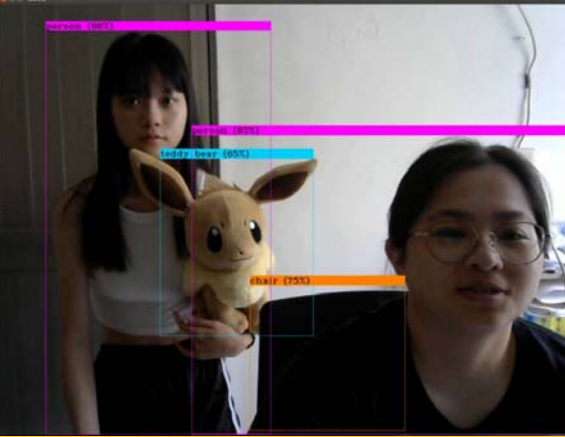



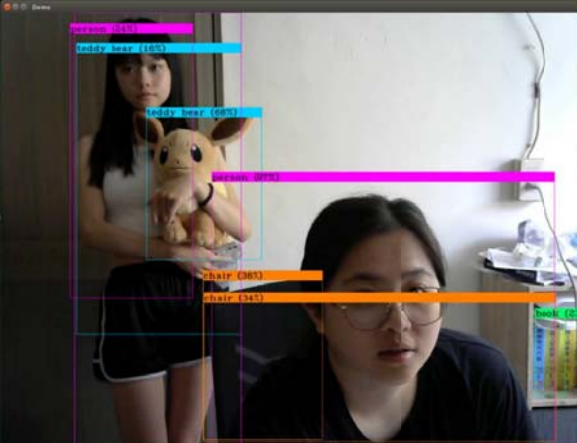

TensorRT      yolov3      288





TensorRT	yolov3	416	
TensorRT	yolov3	608	
Darknet	yolov4-tiny	288	
Darknet	yolov4-tiny	416	

Darknet	yolov3-tiny	288	
Darknet	yolov3-tiny	416	
Darknet	yolov4	288	
Darknet	yolov4	416	

Darknet	yolov4	608	
Darknet	yolov3	288	
Darknet	yolov3	416	
Darknet	yolov3	608	

## 二. 第二部分 ( EXP1、EXP2、EXP3 )

### 1. EXP1 ( flashing )

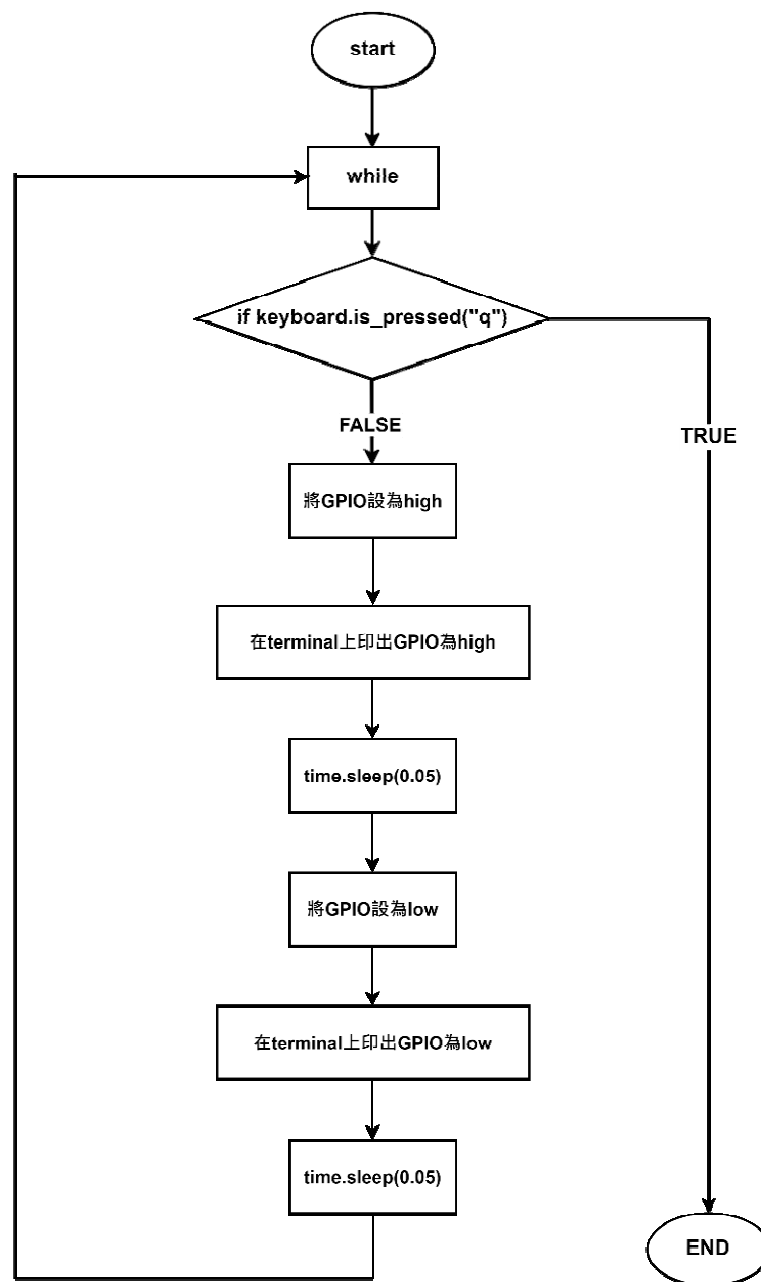
- 實作內容

實作讓 led 循環閃爍，週期為 0.1 秒。

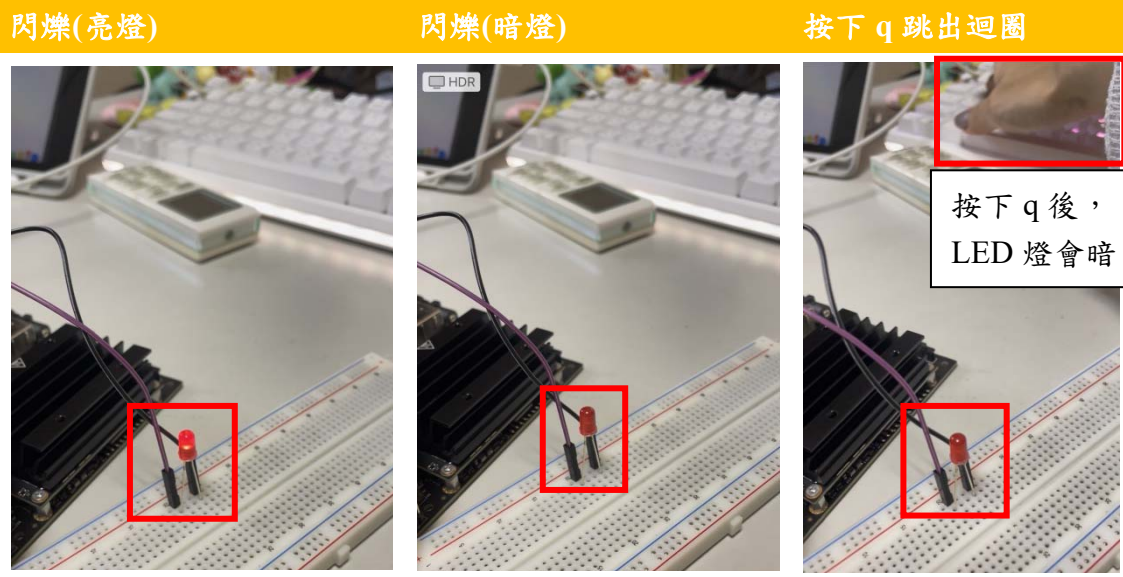
- 實作方式

為了讓 LED 以 0.1 秒之週期反覆閃爍，我們用了 while 迴圈去撰寫，進入迴圈後，用 GPIO\_high 跟 GPIO\_low 去控制 output pin，使其亮燈或暗燈。同時，為了防止 LED 一直閃爍而跳不出迴圈，導致只能強迫終止程式，我們也多加了一個能夠跳出程式的按鍵 ( q )，當 q 鍵按下時，便會 break。

- 簡易流程圖



- 實作結果



- 分析討論

第一個實作是單純希望我們讓 LED 燈閃爍，所以只需要讓 GPIO\_high 及 GPIO\_low 各自停留 0.05 秒即可，比較需要注意的是要先確認 code 裡寫的輸出腳位是哪一個，就要將 LED 接到該腳位，接錯的話會導致 LED 燈不會亮。

## 2. EXP2

- 實作內容

透過修改 trt\_yolo\_cond.py 及 condition.py 建立 4 種模式，並可以透過按鍵作模式切換。

Mode0: 正常模式，與原版相同

Mode1: 在終端顯示偵測資訊

Mode2: 偵測到人時，在終端顯示並點亮 LED 燈

Mode3: 在終端顯示偵測人數，人的數量超過定值後點亮 LED 燈

- 實作方式

在 condition.py 裡把 Mode2 用到的 detect\_human 函式與 Mode3 的 detect\_human\_num 完成，並在 trt\_yolo\_cond.py 中的 main 函式執行程式，藉由按鍵'm'與'M'作為切換模式的指令，按一下按鍵'm'與'M'就可換成下一個模式。

Mode0: 不用做更動，程式正常執行即可。

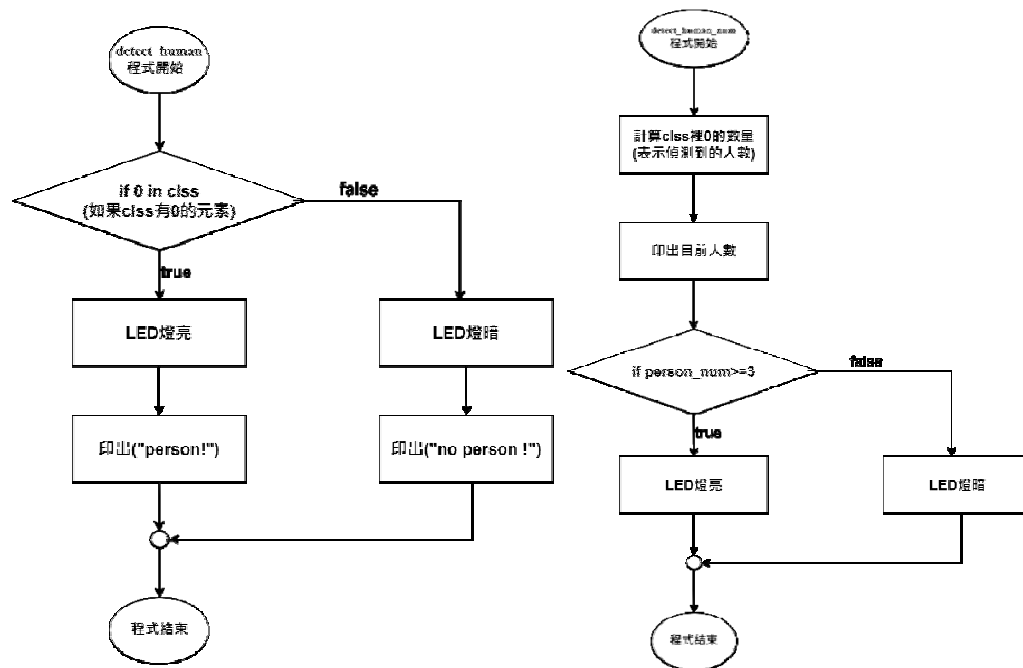
Mode1: 利用 show\_info 函式將 clss, boxes 等資訊 print 出來，顯示在終端機上。

Mode2: 我們先查看 clss 的 py 檔，發現 clss==0 時，代表偵測到人，所以我們用 if 去做條件判斷，若 clss 裡面有 0 的元素時，LED 燈亮(GPIO\_high)，反之 clss 裡面沒有 0 時，燈暗。

Mode3: 因為 clss 為一個 array，所以我們用 numpy 函式庫中的 np.sum(arr==0) 的方式去計算陣列裡有幾個 0，這樣一來我們便可以得到畫面中人的數量。我們將 num 設定為 3，意即，如果畫面中大於等於三個人，LED 燈就會亮且印出 (person!)。

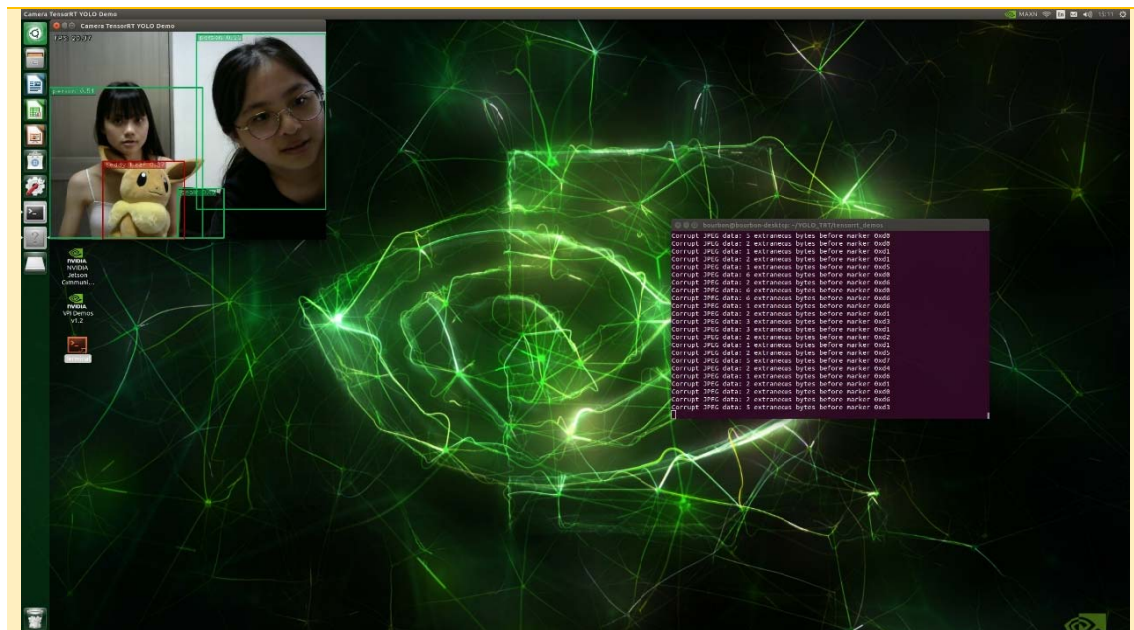


## ● 簡易流程圖

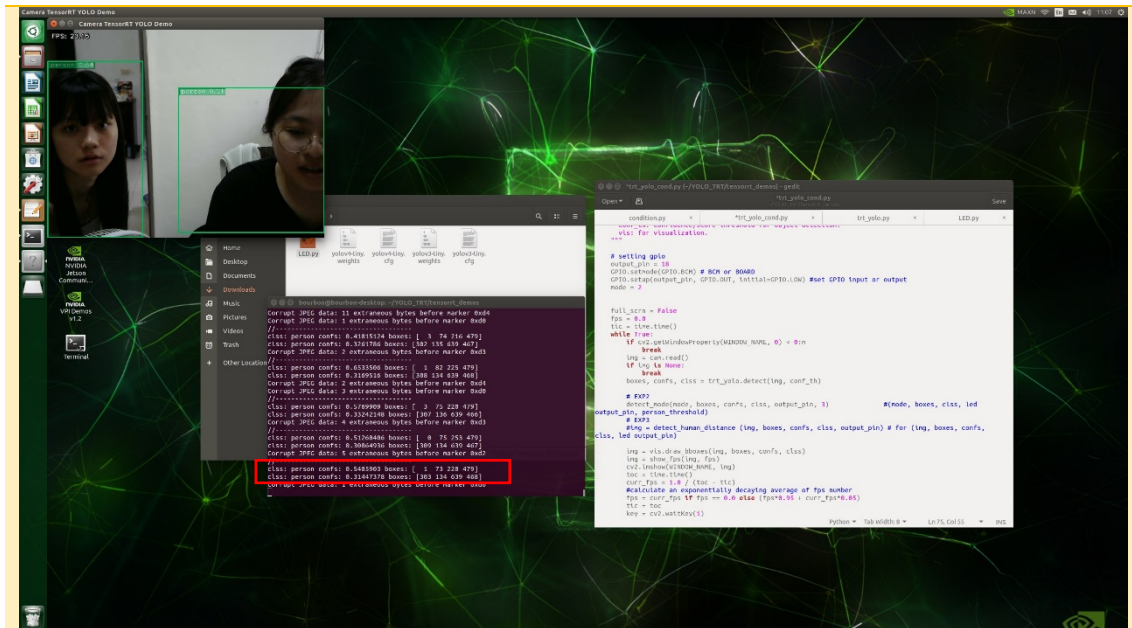


## ● 實作結果

### Mode0 :Pass，無其他新功能



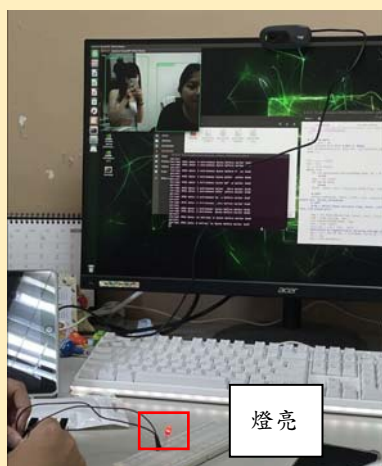
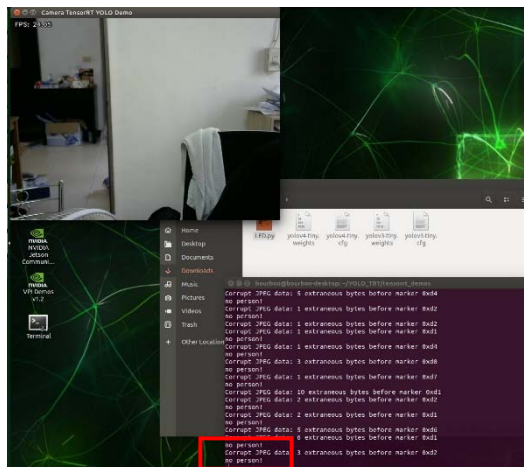
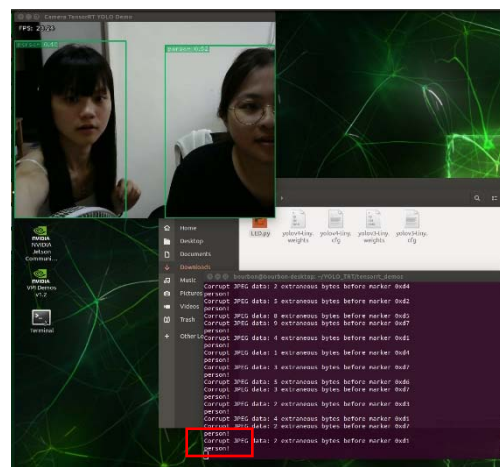
## Mode1:在終端顯示偵測資訊



## Mode2:偵測到人時，在終端顯示並點亮 LED 燈

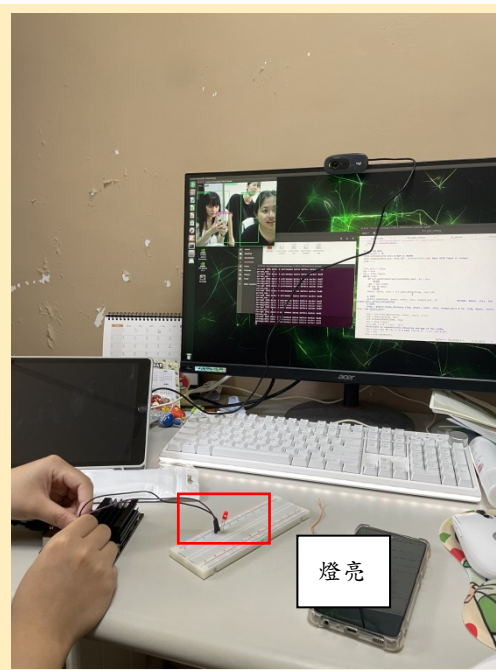
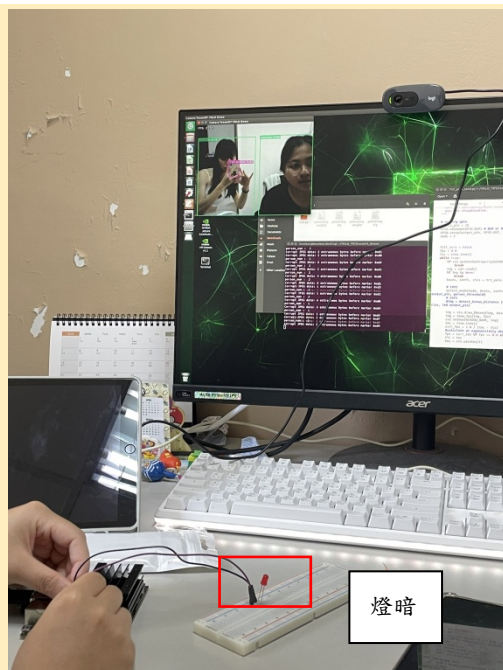
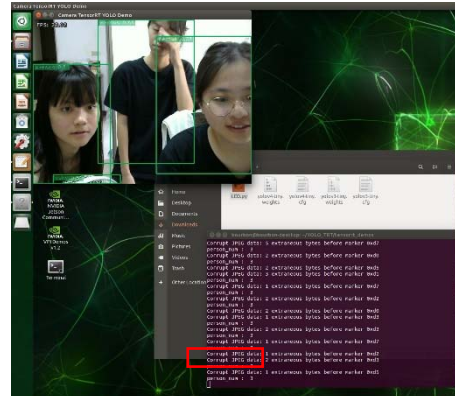
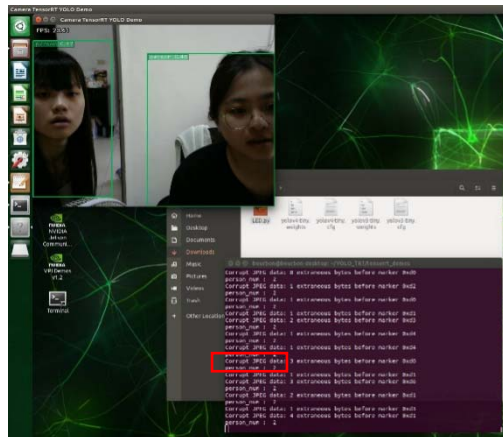
有人:燈亮，顯示 person!

無人:燈暗，顯示 no person!



**Mode3:在終端顯示偵測人數，人的數量超過定值(設定為 3 人)後點亮 LED 燈**

人數 2 人:燈暗，顯示 person\_num:2      人數 3 人:燈亮，顯示 person\_num:3



## ● 分析討論

第二個實作，原本我們以為 clss 為一個整數值，所以只用 `clss==0` 去判斷，但後來把 clss 印出來，發現 clss 是用 array 去儲存物件標籤，所以主要知道 clss 的變數型態是 array 後，模式 2 和 3 就很好完成。



### 3. EXP3

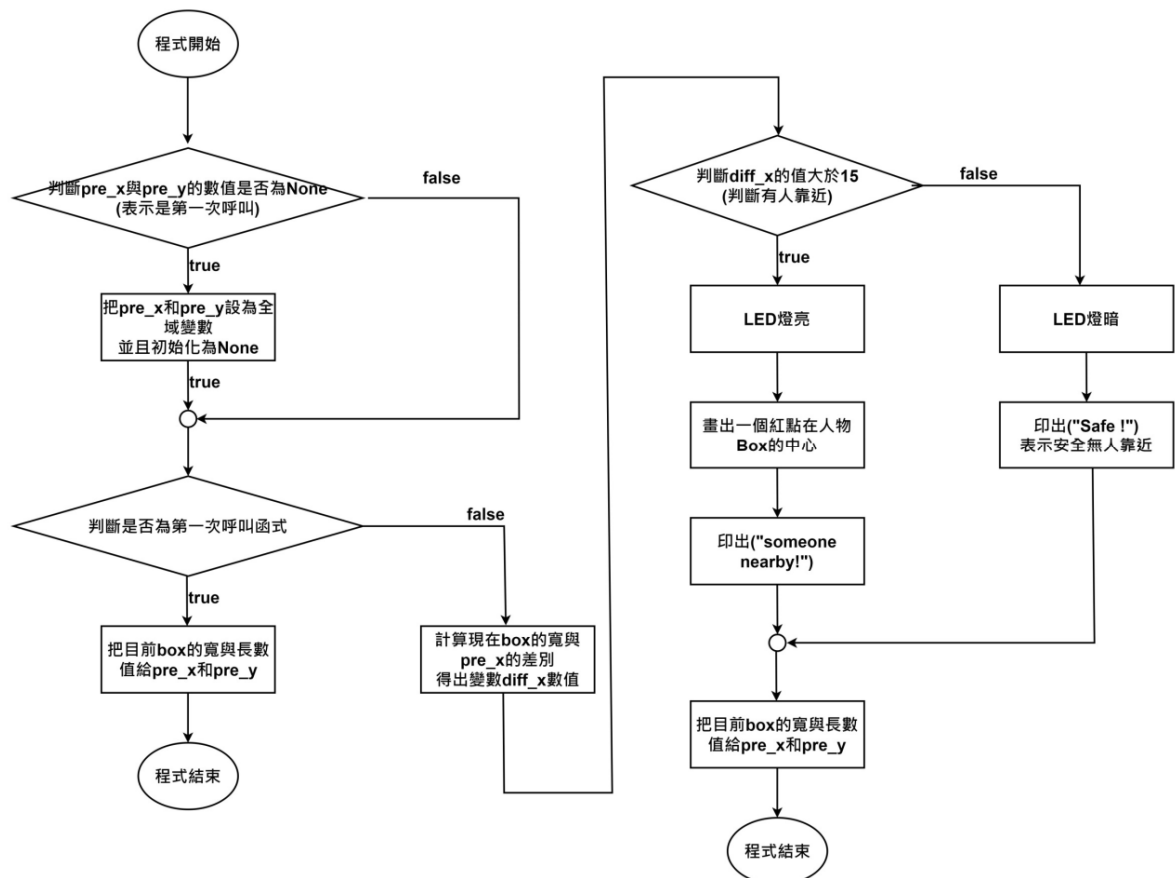
- 實作內容

設計當人靠近時亮燈，並畫出人物中心點位置。

- 實作方式

依題目意思，需要判斷人物是靠近還是遠離，所以需要知道人物位置的變化量。我們利用 boxes 的座標資訊，計算出 boxes 的寬與長，把前一次偵測到的人物 box 寬與長先儲存到全域變數 pre\_x 和 pre\_y 中，在下次偵測到的人物 box 時，把目前的 box 的寬(x)與上一次的寬(pre\_x)做相減，如果是大於 15 的話就表示靠近，此時燈亮，且會在影像中以人物 box 四點的中心畫出紅點，且印出(someone nearby!)；反之如果為靜止或是遠離，此時燈暗，會印出(Safe!)。

- 簡易流程圖

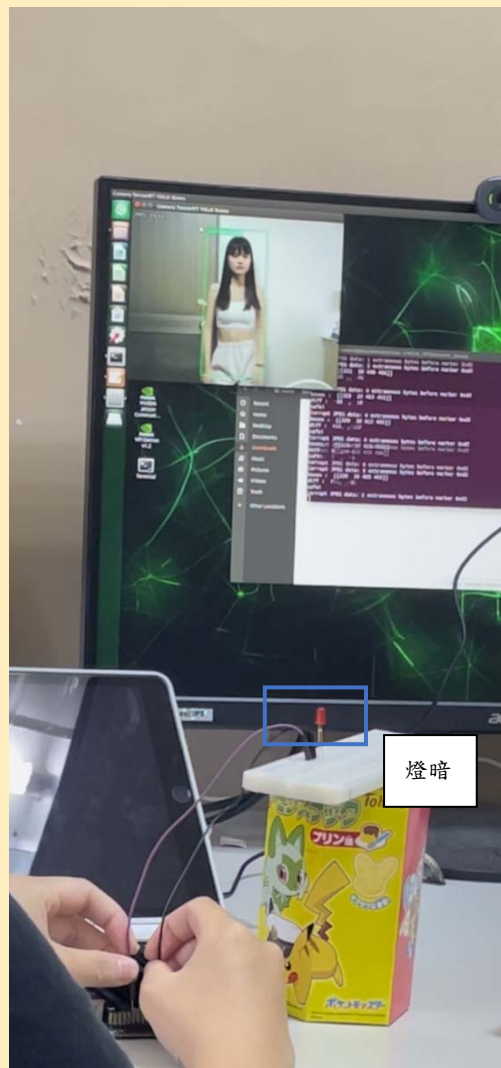


## ● 實作結果

### 人物靠近



### 人物遠離



## ● 分析討論

第三個實作，因為需要判斷人物動態變化，靠近時會亮燈，所以在寫條件時，需要去儲存每次的人物位置，再依位置變化量判斷說有沒有靠近。一開始我們的條件為人物 box 長與寬變化量都增加時就表示靠近，但實作發現，因為受限影像的大小，如果人物和攝像頭太近，box 的長度就等於影像畫面的長度，不會隨人物靠近或遠離而變化，所以後來把條件改成只判斷寬度的變化量，且變化量需大於 15，如果條件設為大於 0，會因為人物些微晃動就可能誤判為靠近。

### 三. 工作分配表

組員	黃麗穎	黃詣辰	胡庭翹	林佳明
工作比例	25%	25%	25%	25%
簽名	黃麗穎	黃詣辰	胡庭翹	林佳明

#### 四. 參考資料

- [1][https://www.researchgate.net/publication/357684232\\_Comparing\\_YOLOv3\\_YOLOv4\\_and\\_YOLOv5\\_for\\_Autonomous\\_Landing\\_Spot\\_Detection\\_in\\_Faulty UAVs](https://www.researchgate.net/publication/357684232_Comparing_YOLOv3_YOLOv4_and_YOLOv5_for_Autonomous_Landing_Spot_Detection_in_Faulty_UAVs)
- [2]<https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec>
- [3]<http://giantpandacv.com/project/%E9%83%A8%E7%BD%B2%E4%BC%98%E5%8C%96/AI%20%E9%83%A8%E7%BD%B2%E5%8F%8A%E5%85%B6%E5%AE%83%E4%BC%98%E5%8C%96%E7%AE%97%E6%B3%95/TensorRT/%E4%B8%80%E5%8C%84%E5%8B%BB%E7%BB%8D%E5%8C%8E%E5%89%E8%A3%85%E5%8F%8A%E5%A6%82%E4%BD%95%E4%BD%BF%E7%94%A8%E5%8C%9F/>
- [4]<https://github.com/hgpvision/darknet>