

摘要

本專題核心在於透過智慧眼鏡 BT-300 即時辨識樹上的香蕉成熟度。主要經過 BT-300 內建攝影機取得畫面，藉由網頁即時通訊（Web Real-Time Communication，以下簡稱 WebRTC）功能與 NVIDIA Jetson xavier 上事先訓練好的 YOLOv8 模型結合。當畫面傳輸給模型辨識後，模型會再傳回辨識結果至 BT-300，即可得出該畫面中香蕉串的成熟度，用以提供香蕉採收時間上的參考。

一、專題研究動機

在香蕉種植過程中，蕉農需精準掌握成熟度以決定適當的採收時間。而如今要知道香蕉的成熟度，多倚賴專業蕉農靠近香蕉串，肉眼觀察第 2 到 3 節的香蕉串，大致能以其形狀（越少棱角越成熟）與飽滿度來決定成熟度(如圖 1)。



圖 1、蕉農講解成熟度大致判斷方式

因肉眼辨識需靠近詳細觀察，且需有經驗蕉農才能準確判斷。為使經驗不足者也能辨識，以及減少辨識時間，故期望建構出一套完整設備，使得戴上智慧眼鏡操作即可辨識出香蕉串的成熟度，以判斷採收時間，並增加蕉農的效率。

二、研究方法與步驟

1. 建立資料集

為建立 YOLOv8 香蕉成熟度辨識模型，本專題與藝隆農產有限公司合作，首先來到藝隆位於屏東縣南州鄉的香蕉農場，拍攝樹上香蕉串影片並向蕉農請教成熟度判斷方式。再藉由 Roboflow 將影片轉成圖片並一張張標註(如圖 2)。

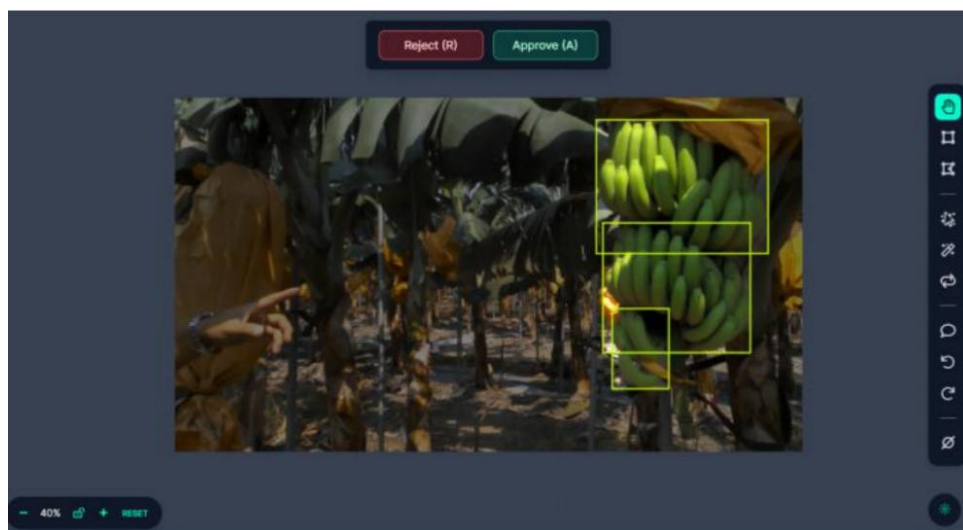


圖 2、照片標註示意圖（標註出照片中香蕉串部分，以一節為單位）

在標註過程中，本專題依據現場香蕉情況以及蕉農的觀察判斷，共分為六個 class：4 分熟、5 分熟、7 分熟、7.5 分熟、8 分熟、8.5 分熟（如圖 3）。在標準方面，根據蕉農的判斷方法以平視畫面為主，並以一節為單位（如圖 4），且刪除掉角度過大（如圖 5）與無法看到整節（如圖 6）等照片。最後再由加入 Data Augmentation（Rotation: Between -15° and $+15^{\circ}$ 、Brightness: Between -15% and $+15\%$ 、Exposure: Between -10% and $+10\%$ 、Noise: Up to 0.5% of pixels）得到完整且泛化的 Dataset 共 1200 張。

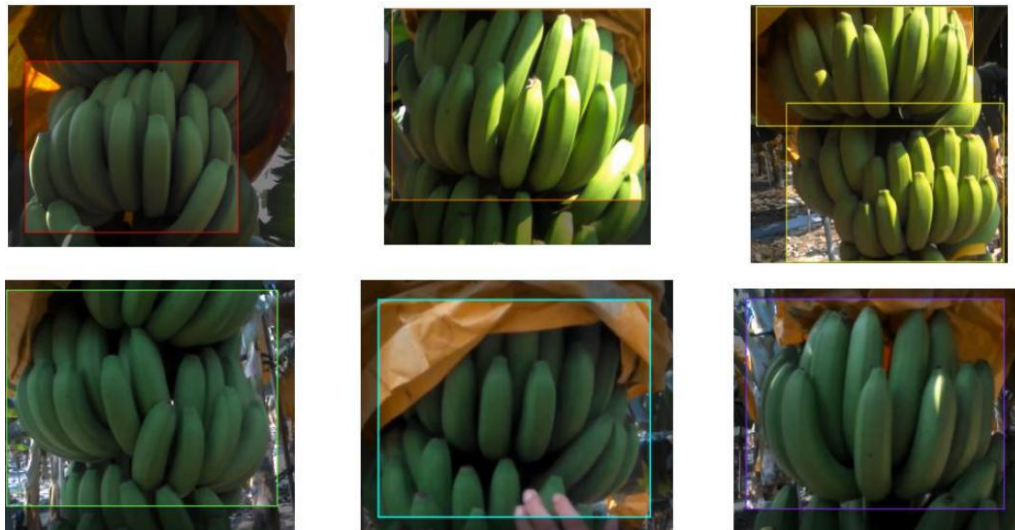


圖 3、根據成熟度，使用不同顏色方框標註（左上到右上依序為 4 分熟、5 分熟、7 分熟，左下到右下依序為 7.5 分熟、8 分熟、8.5 分熟）



圖 4、適合作為 Dataset 的平視香蕉圖片範例



圖 5、角度過大（非平視）故不適合作為 Dataset 的香蕉圖片範例



圖 6、無法看到整節故不適合作為 Dataset 的香蕉圖片範例

2.建立 YOLOv8 辨識模型

得到 Dataset 後，接著便藉由 YOLOv8 訓練出香蕉成熟度辨識模型。我們將 1200 張照片依照 $\text{train} : \text{valid} : \text{test} = 0.7 : 0.2 : 0.1$ 比例的方式平均切開，確保六個不同成熟度的 class 都是依此比例（即每個 class 中的 **train** 照片為 140 張、**valid** 照片為 40 張、**test** 照片為 20 張），且逐一檢查 **test** 中的照片以避免過於相近的情況。最後將其給予 YOLOv8 模型訓練並經過多次調整測試後，得到較好的香蕉成熟度辨識模型結果（如圖 7、圖 8）。並產生測試圖片辨識結果(如圖 9)。


```

Ultralytics YOLOv8.0.228 Python-3.11.7 torch-2.1.2 CUDA:0 (NVIDIA GeForce RTX 4070, 119
Model summary (fused): 168 layers, 3006818 parameters, 0 gradients

```

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95):
all	240	309	0.985	0.988	0.992	0.963
banana_4	240	48	0.997	0.979	0.995	0.959
banana_5	240	51	0.959	0.98	0.981	0.957
banana_7	240	44	0.973	0.977	0.993	0.955
banana_7.5	240	59	0.983	0.99	0.994	0.968
banana_8	240	67	1	1	0.995	0.962
banana_8.5	240	40	0.997	1	0.995	0.976

```

Speed: 0.1ms preprocess, 0.7ms inference, 0.0ms loss, 0.5ms postprocess per image
Results saved to runs/detect/train12

```

圖 7、YOLOv8 模型訓練最終結果（數值呈現），Precision 約為 0.98

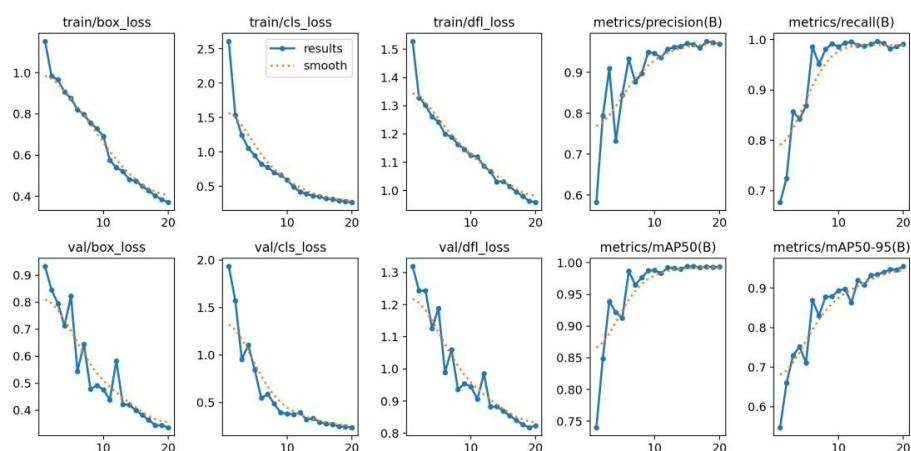


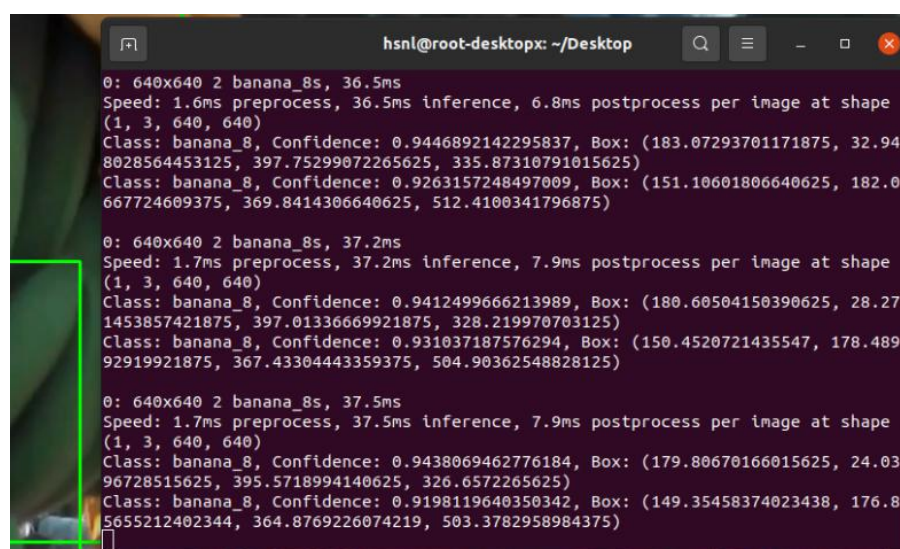
圖 8、YOLOv8 模型訓練初步結果（折線圖呈現）



圖 9、圖片辨識結果（辨識框包含結果與信心數值，每個成熟度圖片各一）

3.將 YOLOv8 辨識模型部署至 NVIDIA Jetson Xavier

雖已做出 YOLOv8 辨識模型，但對於每幀（圖片）的 inference time 約需要 800ms，想要即時辨識顯然太久。為解決此問題，決定透過 NVIDIA Jetson Xavier 上的 GPU 達成加速效果。首先在 Jetson Xavier 上部署好環境，接著將經由前述 YOLOv8 模型訓練產生的 best.pt 檔案放置進來，大致上即可達成。完成後，使用當初作為 test 資料集的照片以及 2 到 3 部影片進行測試，可發現透過 GPU 加速，可以使 inference time 大幅下降至 40ms 左右（如圖 10），且能夠準確辨識香蕉成熟度（如圖 11）。



```
hsnl@root-desktopx: ~/Desktop
0: 640x640 2 banana_8s, 36.5ms
Speed: 1.6ms preprocess, 36.5ms inference, 6.8ms postprocess per image at shape (1, 3, 640, 640)
Class: banana_8, Confidence: 0.9446892142295837, Box: (183.07293701171875, 32.948028564453125, 397.75299072265625, 335.87310791015625)
Class: banana_8, Confidence: 0.9263157248497009, Box: (151.10601806640625, 182.0667724609375, 369.8414306640625, 512.4100341796875)

0: 640x640 2 banana_8s, 37.2ms
Speed: 1.7ms preprocess, 37.2ms inference, 7.9ms postprocess per image at shape (1, 3, 640, 640)
Class: banana_8, Confidence: 0.9412499666213989, Box: (180.60504150390625, 28.271453857421875, 397.01336669921875, 328.219970703125)
Class: banana_8, Confidence: 0.931037187576294, Box: (150.4520721435547, 178.48992919921875, 367.43304443359375, 504.90362548828125)

0: 640x640 2 banana_8s, 37.5ms
Speed: 1.7ms preprocess, 37.5ms inference, 7.9ms postprocess per image at shape (1, 3, 640, 640)
Class: banana_8, Confidence: 0.9438069462776184, Box: (179.80670166015625, 24.0396728515625, 395.5718994140625, 326.6572265625)
Class: banana_8, Confidence: 0.9198119640350342, Box: (149.35458374023438, 176.85655212402344, 364.8769226074219, 503.3782958984375)
```

圖 10、影片中每個 Frame 的辨識相關數據（以中間一幀為例，inference time 為 37.2ms，偵測到 2 節成熟度 class 為 8 的香蕉串，Confidence 皆約 0.94）

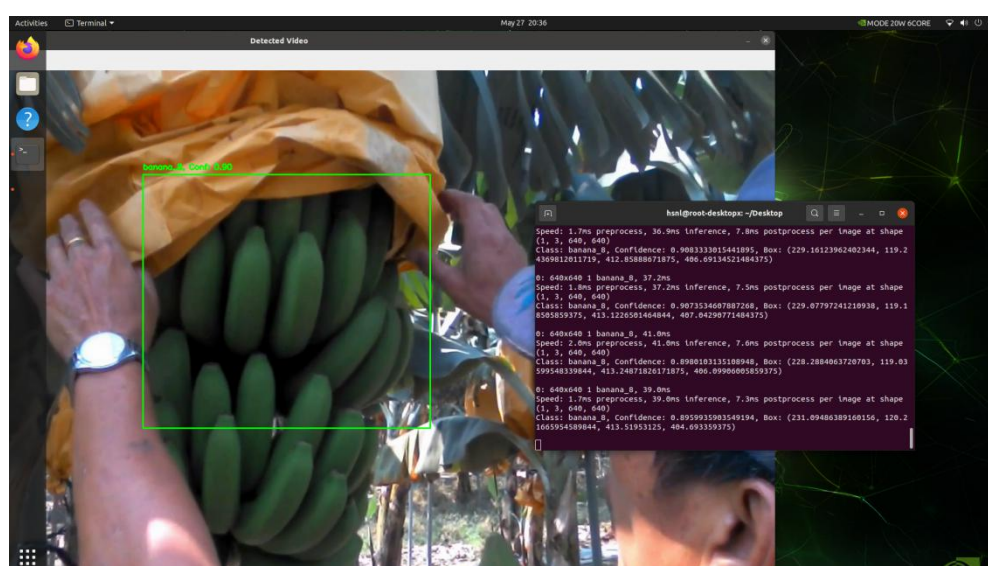


圖 11、透過 Jetson Xavier 進行辨識之畫面示意圖

4. 透過 WebRTC 連結智慧眼鏡 BT-300 與 YOLOv8 辨識模型

完成 YOLOv8 辨識模型的部分後，最後一步需要讓智慧眼鏡 BT-300 與模型結合。雖然將模型直接部署至智慧眼鏡上或許是個選項，然而智慧眼鏡的環境較難達到如 Jetson Xavier 這樣快速的辨識，因此選擇透過 WebRTC 方式讓智慧眼鏡的畫面與辨識模型藉由同一個網路連結。為方便測試與逐步修改，我們先使用筆電代替智慧眼鏡的部分測試。

首先透過一個網頁來實作 WebRTC 的功能（如圖 12），將其分為 uploader 端與 downloader 端。第一步讓欲連線的兩台裝置連接相同網路，這邊我們讓 uploader 端為筆電，downloader 端為 Jetson Xavier。uploader 端在填寫相關 id 後可透過設備內建攝影機取得其畫面；在網頁出現 uploader 已連線後，接下來 downloader 端也輸入其 id，即可成功連線，此時在 downloader 端的裝置就會看到網頁上出現 uploader 端攝影機的畫面（如圖 13）。

在實作好 WebRTC 功能後，再讓 YOLOv8 辨識模型與 WebRTC 相關實作功能結合，並部署在 Jetson Xavier 上，即可達到如圖 13 所呈現之辨識效果。

確認可正常透過 WebRTC 功能連線後，即可將筆電的角色換回智慧眼鏡 BT-300，這樣就能在 Jetson Xavier 上看到由智慧眼鏡的攝影機所拍攝之畫面。然而最終的目標，是希望能夠透過眼鏡看到的香蕉串即時得到其成熟度，因此智慧眼鏡 BT-300 將同時作為 uploader 端及 donwloader 端，也就是畫面會回傳到智慧眼鏡上，達成戴上智慧眼鏡後就能即時看到其辨識結果的目標（如圖 14）。

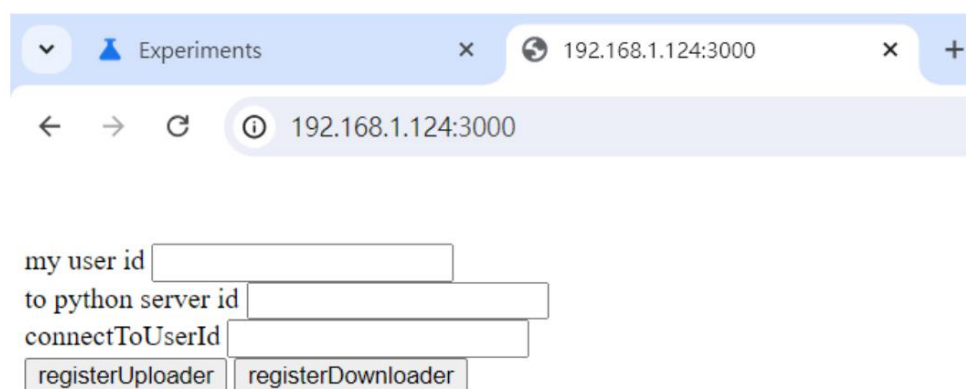


圖 12、WebRTC 功能實作網頁示意圖

my user id
to python server id
connectToUserId

fps:12,w:640,h:480

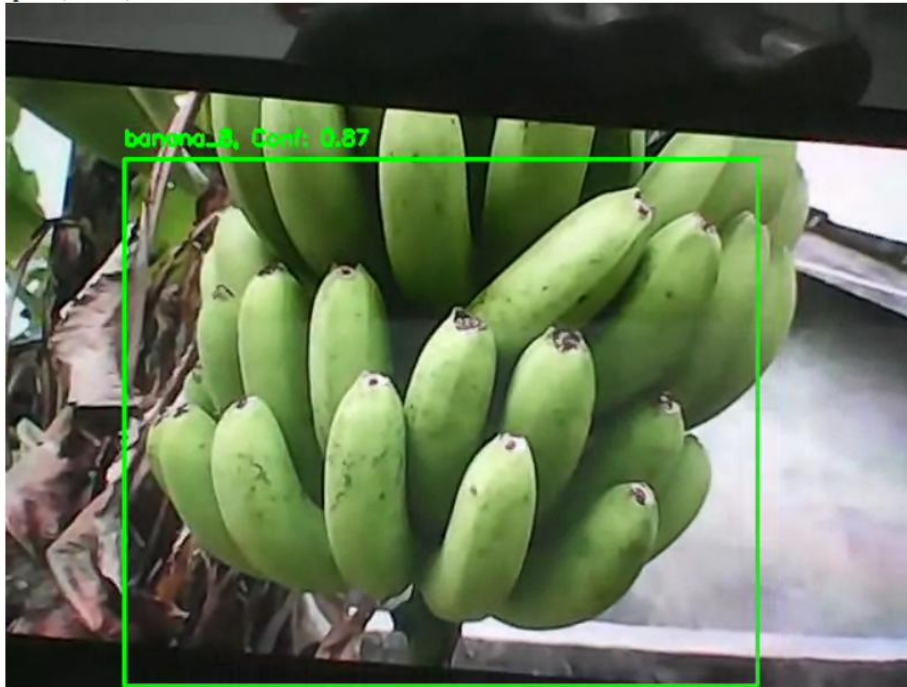


圖 13、downloader 端看到 uploader 端攝影機之畫面



圖 14、由智慧眼鏡 BT-300 上看到的即時辨識畫面示意圖

三、效能評估與成果

在整體設備完成後，我們使用部分未加入資料集之樹上香蕉串影片段落，來進行準確度測試。經過測試，在畫面穩定且不距離過遠的情況下，辨識結果與當初紀錄的成熟度皆符合。此外，也在網路上搜尋部分香蕉串成熟採收影片做為測試，經判斷成熟度多為 8 或者 8.5（即最成熟的兩個 class）。最後也希望能再度回到合作的藝隆香蕉農場，與蕉農實地進行測試，但因日前的數個颱風造成農場香蕉部分毀損，故仍在安排中。

四、結論

透過 WebRTC 的功能並讓智慧眼鏡 BT-300 同時作為 uploader 與 downloader，智慧眼鏡的畫面可以被傳輸到 Jetson Xavier 上，經由 YOLOv8 香蕉成熟度辨識模型以及 GPU 的加速效果，可以立刻得到辨識結果，傳回到智慧眼鏡。藉由該畫面中香蕉串的成熟度辨識結果，蕉農即可知道此香蕉仍需多久時間才能收割，提供其採收上的參考。

五、未來展望

在大致完成整個專題後，我們大致整理出三項可以做為未來發展或增進的部分如下：

- 1.在成熟度分類方面，可以再進一步細分或擴充分類總數。因為香蕉農場的實際情況限制，故本專題只將成熟度分成六類，若能更完整的拍攝，或許能夠再細分出 4.5、5.5、6、6.5 等成熟度。
- 2.環境因素也可納入考量。因為當時為晴天拍攝，然而蕉農在採收時也可能是陰天、雨天等，天氣和光線的考量也可能會讓結果更精準。
- 3.或許可加入自動化流程，將機器人與此設備結合，由其來執行這項辨識工作，減少人工辨識的比例，將人工分配去做其他重要的部分。