

National Sun Yat-sen University

機器學習系統設計實務與應用

期末專題報告

Title：猴子偵測警報系統

第 11 組：我想畢業

授課教授：謝東佑教授

姓名：B083022053 黃啟桓

姓名：B093040038 徐睿鍾

姓名：B093040056 鄭宗韜

姓名：B123040042 邱柏翔

目錄

1. 專題主題
 - 1.1 主題概念
 - 1.2 與其他方式差異
2. 功能與使用流程
 - 2.1 功能
 - 2.2 使用流程
3. 模型介紹
 - 3.1 架構
 - 3.2 模型優點
4. 訓練方法
 - 4.1 工具介紹
 - 4.1.1 Roboflow
 - 4.1.2 Ultralytics
 - 4.1.3 NXP GuiGuider & MCUXpresso (IDE)
 - 4.1.3.1 NXP GuiGuider 介紹
 - 4.1.3.2 MCUXpresso (IDE) 介紹
 - 4.1.3.3 NXP GuiGuider 使用方法
 - 4.2 資料選擇
 - 4.2.1 資料來源
 - 4.2.2 資料標註
 - 4.3 資料增強與預處理
 - 4.4 模型訓練
5. 效能評估
6. 方法實現成果與 demo
 - 6.1 Line Notify
 - 6.2 猴子分佈地圖
 - 6.3 上鎖功能與影片展示
7. 參考文獻
8. 分工比例&簽名

1. 專題主題

1.1 主題概念：

許多學生在宿舍常有食物被猴子搶奪的情況，為了解決這個問題，我們設計了一個能用機器學習偵測猴子的系統，系統能利用攝影機拍攝畫面，並使用 YOLOv8 nano 模型識別猴子，利用其較低的資源消耗與硬體需求降低架設成本，當猴子進入攝影機範圍時，透過 line notify 通知警告並發送猴子位置地圖以及猴子照片，讓使用者能主動避開猴子出沒地，系統也加入了櫃子自動上鎖功能，讓使用者即使不在附近，系統也能立即阻止猴子可能的偷竊行為。

1.2 與其他方式差異：

因需求較特殊，目前少有猴子相關機器學習應用，而相較目前主要使用漆彈驅趕猴子的方式，本系統除了能大幅減少人力成本，也能擺脫時間與空間的限制，在多處進行設置，並達到 24 小時不間斷的偵測。

2. 功能與使用流程

2.1 功能

- 偵測到猴子時，系統透過 Line Notify 發送即時警報並拍攝照片。
- 偵測到猴子時，系統自動將附近的櫃子或垃圾桶上鎖。
- 根據多個攝影機的偵測結果，提供猴子位置的即時地圖。
- 可使用觸控面板設定功能。

2.2 使用流程

1. 使用觸控面板開啟偵測功能
2. 攝影機拍攝畫面
3. 使用 Pi5 運行機器學習模型辨識猴子
4. 偵測到猴子時，攝影機自動拍攝照片並發送 Line Notify
5. 利用 wifi 將猴子的位置發送至 Server，創建猴子地圖
6. 利用 wifi 連接 Pi4，將櫃子上鎖

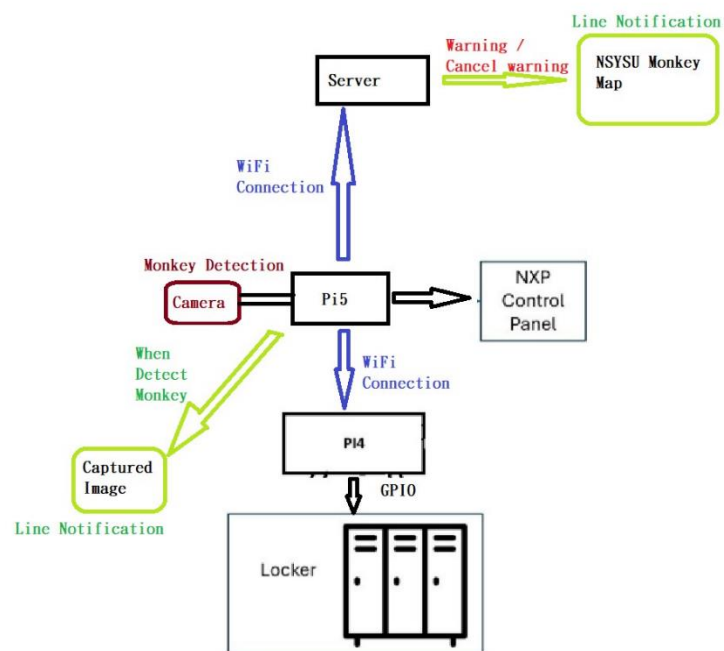


圖 2.2.1 猴子警報系統架構圖

3. 模型介紹

本系統選用的模型是 YOLOv8 nano，下面是關於這個模型的介紹

3.1 架構：

YOLOv8 的架構主要分為以下三個部分：

Backbone：使用卷積和反卷積層來提取特徵並使用了殘差連接和瓶頸結構來減少網路的大小。

Neck：使用多尺度特徵融合，將 backbone 不同階段的特徵圖融合增強特徵表現能力。

Head：負責最後的目標檢測和分類任務，包含了一個檢測頭和分類頭。檢測頭包含了卷積層和反卷積層，用於生成檢測結果。分類頭使用全局平均池化層對每個特徵圖分類。

YOLOv8 提供了五種模型大小：YOLOv8n、YOLOv8s、YOLOv8m、YOLOv8l、YOLOv8x，每種模型大小對應不同的層數，本次選用的是其中的 YOLOv8n，不同模型的具體數據如下

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)
YOLOv8n	640	37.3	80.4	0.99	3.2
YOLOv8s	640	44.9	128.4	1.20	11.2
YOLOv8m	640	50.2	234.7	1.83	25.9
YOLOv8l	640	52.9	375.2	2.39	43.7
YOLOv8x	640	53.9	479.1	3.53	68.2

表 3-1 YOLOv8 模型比較

3.2 模型優點：

- 高效性：YOLOv8 nano 雖然層數少，但依然能夠達到相對較高的準確率，特別是在處理動物檢測等特定類別的任務時表現良好。
- 快速處理：因為模型較小，所以處理速度快，特別適合在實時系統中使用，如移動裝置或嵌入式系統。
- 低資源消耗：適合在資源有限的硬體上運行，不需要高端 GPU 支持。

Performance Comparison of YOLOv8 vs YOLOv5

Model Size	Detection*	Segmentation*	Classification*
Nano	+33.21%	+32.97%	+3.10%
Small	+20.05%	+18.62%	+1.12%
Medium	+10.57%	+10.89%	+0.66%
Large	+7.96%	+6.73%	0.00%
Xtra Large	+6.31%	+5.33%	-0.76%

*Image Size = 640 *Image Size = 224

表 3-2 YOLOv8 與 v5 整體比較

Instance Segmentation Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	27.6	36.7	+32.97%
Small	37.6	44.6	+18.62%
Medium	45	49.9	+10.89%
Large	49	52.3	+6.73%
Xtra Large	50.7	53.4	+5.33%

*Image Size = 640

表 3-4 YOLOv8 與 v5 實體分割比較

Object Detection Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	28	37.3	+33.21%
Small	37.4	44.9	+20.05%
Medium	45.4	50.2	+10.57%
Large	49	52.9	+7.96%
Xtra Large	50.7	53.9	+6.31%

*Image Size = 640

表 3-3 YOLOv8 與 v5 物件檢測比較

Image Classification Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	64.6	66.6	+3.10%
Small	71.5	72.3	+1.12%
Medium	75.9	76.4	+0.66%
Large	78	78	0.00%
Xtra Large	79	78.4	-0.76%

*Image Size = 224

表 3-5 YOLOv8 與 v5 影像分類比較

從上面的比較中可以看出，YOLOv8 nano 除了體積較小，在各方面的判斷能力也都有著較大的提升。

對於我們的猴子偵測系統，YOLOv8 nano 快速的判斷能即時對猴子做出反應，並且較低的資源需求也更容易在宿舍區多處設置，能夠有效降低成本而不犧牲過多的準確率。

4. 訓練方法

4.1 工具介紹

我們使用 Roboflow 與 Ultralytics 進行資料處理與訓練，這些工具有以下特點

4.1.1 Roboflow

- 資料管理: 支持多種資料格式的上傳和轉換，靈活的資料集管理功能。支持的格式包括圖片（.jpg, .png, .bmp, .webp）和影片（.mov, .mp4, .avi），也可以使用 YouTube 影片。
- 資料標註: 內置標註工具，支持多種標註類型，如物體檢測、圖像分類和分割。提供自動與手動標記選項，可以將檔案分配給指定參與者進行標記。
- 資料增強/前處理: 提供豐富的資料增強技術，如旋轉、翻轉、裁剪、顏色變換等，以提高模型的泛化能力。
- 模型訓練: 支持主流深度學習框架，如 TensorFlow、PyTorch 等，用戶可以選擇預訓練模型或自行訓練模型。
- 部署與集成: 提供多種部署方式，包括雲端部署、本地部署和邊緣設備部署，支持 API 集成以便快速實現應用。

4.1.2 Ultralytics

- YOLO 模型: Ultralytics 提供了多個版本的 YOLO 模型（如 YOLOv5、YOLOv8），這些模型在各種物體檢測任務中均表現出色。用戶可以使用官方提供的模型或自訓模型進行再訓練。
- 高效性: YOLO 模型因其結構簡單、運算速度快而廣受歡迎，特別適合需要即時檢測的應用場景。
- 簡易使用: Ultralytics 提供了易於使用的開源代碼庫，配有詳細的文檔和教程，幫助用戶快速上手。
- 可擴展性: YOLO 模型具有很強的可擴展性，用戶可以根據自己的需求進行模型調整和優化。
- 社區支持: Ultralytics 擁有活躍的開源社區，提供豐富的資源和技術支持，幫助用戶解決在使用過程中遇到的各種問題。

4.1.3 NXP GuiGuider & MCUXpresso (IDE)

4.1.3.1 MCUXpresso IDE

- MCUXpresso IDE 是一個專為 NXP 的 Arm Cortex-M 微控制器設計的集成開發環境。其主要功能包括：
- 全面的 SDK 支持：訪問 NXP 的廣泛軟體開發套件 (SDK)，這些套件提供驅動程序、中介軟體和示例應用程序。
- 先進的調試功能：包含功能豐富的調試器，具有多核調試、實時變量更新和跟踪支持等功能。
- 直觀的界面：以用戶友好為設計理念，支持拖放功能、引腳配置工具和項目嚮導。
- 性能優化：提供性能分析工具來優化代碼執行和內存使用。
- 與其他工具的集成：無縫集成其他 NXP 工具和第三方軟體，提升開發工作流程。

4.1.3.2 NXP GuiGuider

- GuiGuider 是一個專為在 NXP 微控制器上創建圖形用戶界面 (GUI) 而設計的工具。它通過以下功能簡化了嵌入式 GUI 的開發：
- 拖放式 GUI 設計：允許開發者以可視化的方式設計界面，減少手動編碼的需要。
- 豐富的組件庫：包括各種預構建的組件，如按鈕、滑塊和圖表，這些組件可以根據應用需求進行自定義。
- 實時預覽：提供即時反饋，顯示 GUI 在實際硬件上的外觀和行為。
- 代碼生成：自動生成高效、易於維護的代碼，可直接在 MCUXpresso IDE 中使用。
- 跨平台支持：可用於各種 NXP 微控制器，為不同項目提供靈活性。

- 工具配合使用

當 MCUXpresso IDE 和 GuiGuider 結合使用時，為開發具有先進用戶界面的嵌入式應用提供了強大的組合。典型的工作流程包括：

GUI 設計：使用 GuiGuider 設計圖形界面。

代碼生成：從 GuiGuider 生成相應的代碼。

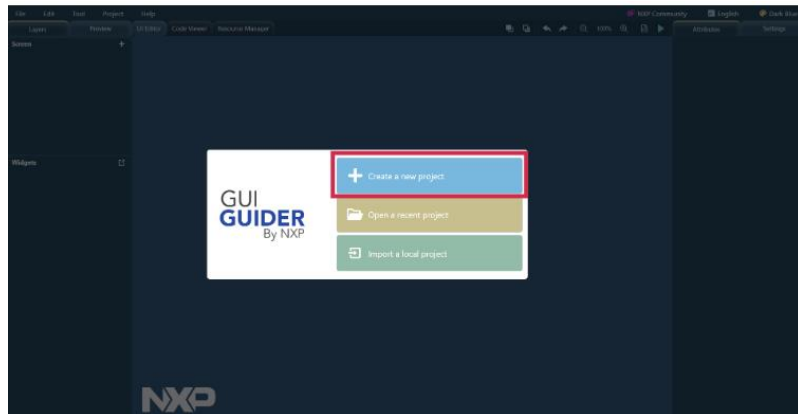
項目集成：將生成的代碼導入 MCUXpresso IDE。

應用開發：在 MCUXpresso IDE 中開發應用邏輯，利用提供的 SDK 和調試工具。

測試與優化：使用 MCUXpresso IDE 中的先進調試和性能分析工具對應用進行測試和優化。

4.1.3.3 GuiGuider

a. 建立一個 Project

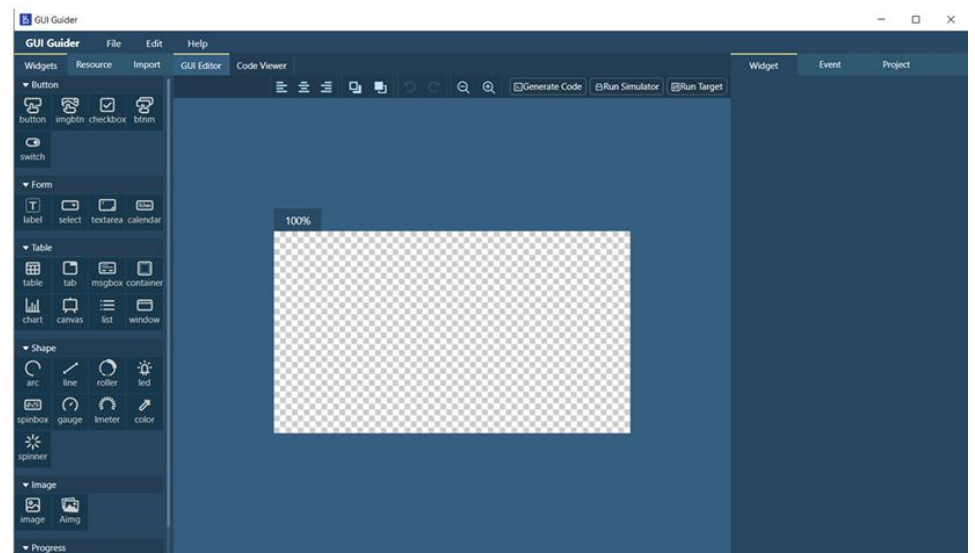


b. 選擇您想要使用的 LVGL 版本。例如，v8。點擊“下一步”或雙擊。向導的“選擇板模板”頁面將顯示。

c. 點擊“模擬器”、“i.MXRT”或“LPC”標籤，然後從模板列表中選擇一個板。例如，選擇 MIMXRT1050-EVKB。點擊“下一步”或雙擊。我們在這次的 Project 中使用的是 MXRT1060 EVKB & Display Panel RK043FN66HS.



d. 進入 GUI 編輯器后就可以開始編輯 GUI



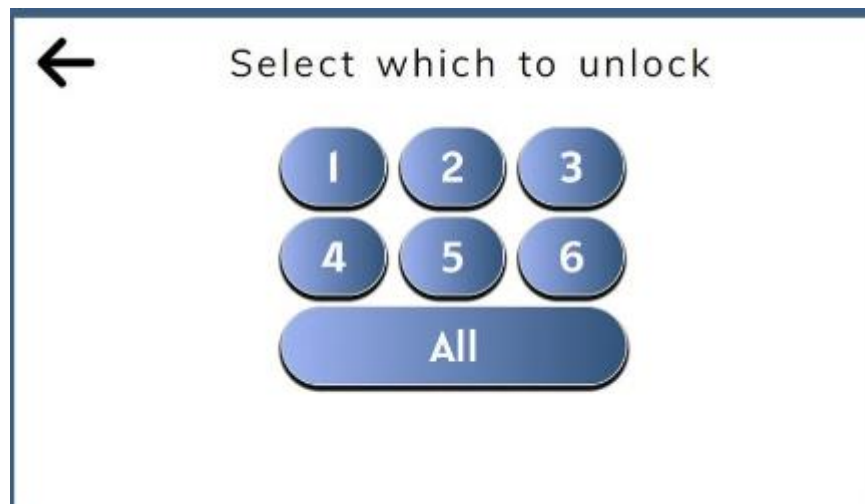
e. 頁面 1 登錄畫面：防止猴子/人形猴亂碰系統導致打開



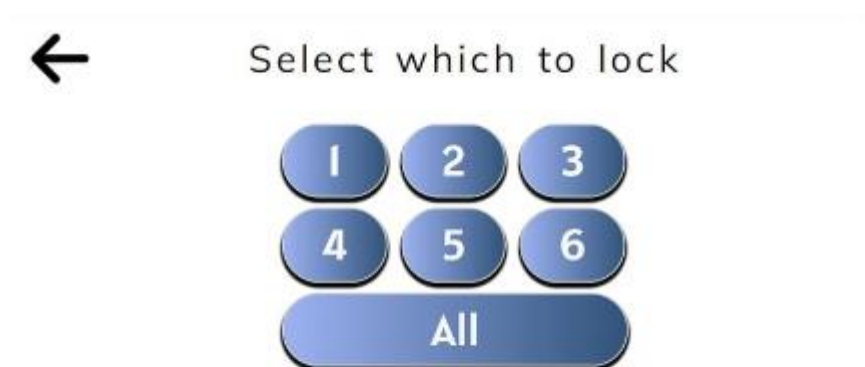
- f. 控制頁面：開啓偵測系統的 switch & 解鎖，鎖上控制



- g. 鎖上的頁面：號碼代表哪個東西要鎖上，或全部



- h. 解鎖的頁面：號碼代表哪個東西要解鎖，或全部



4.2 資料選擇

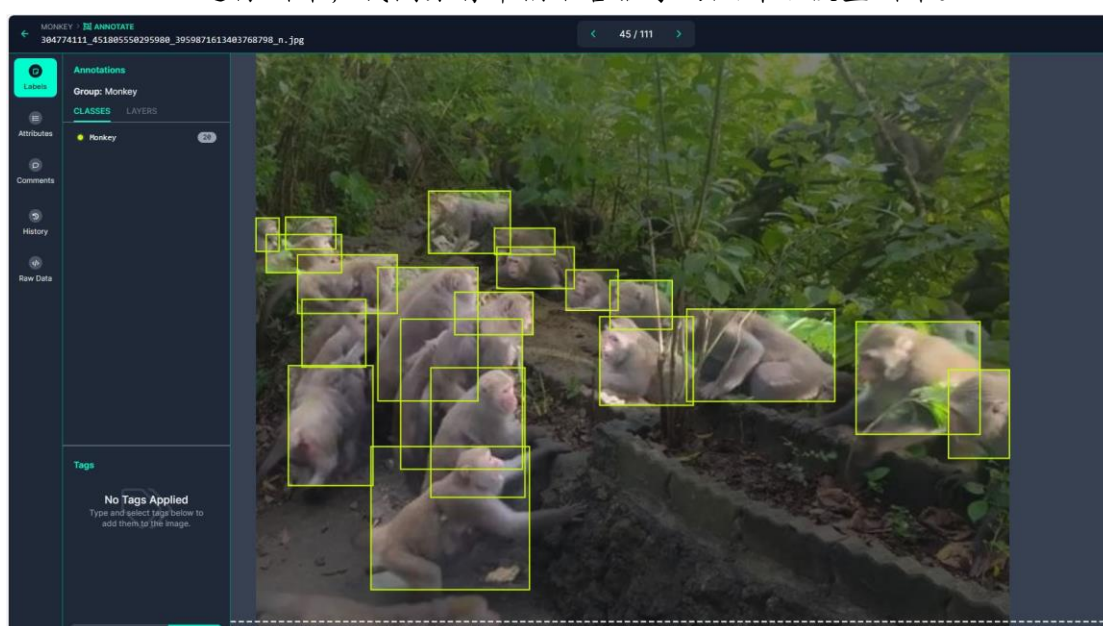
4.2.1 資料來源:

我們使用的資料集為自行標註的照片集，除了自行拍攝，也從 FB 社團、新聞、YouTube 等來源獲得猴子的影像對於影片資料，我們使用 Roboflow 中的影片分割功能將影片分割為多個圖片進行處理，其中包括含有猴子和不含猴子的圖片。

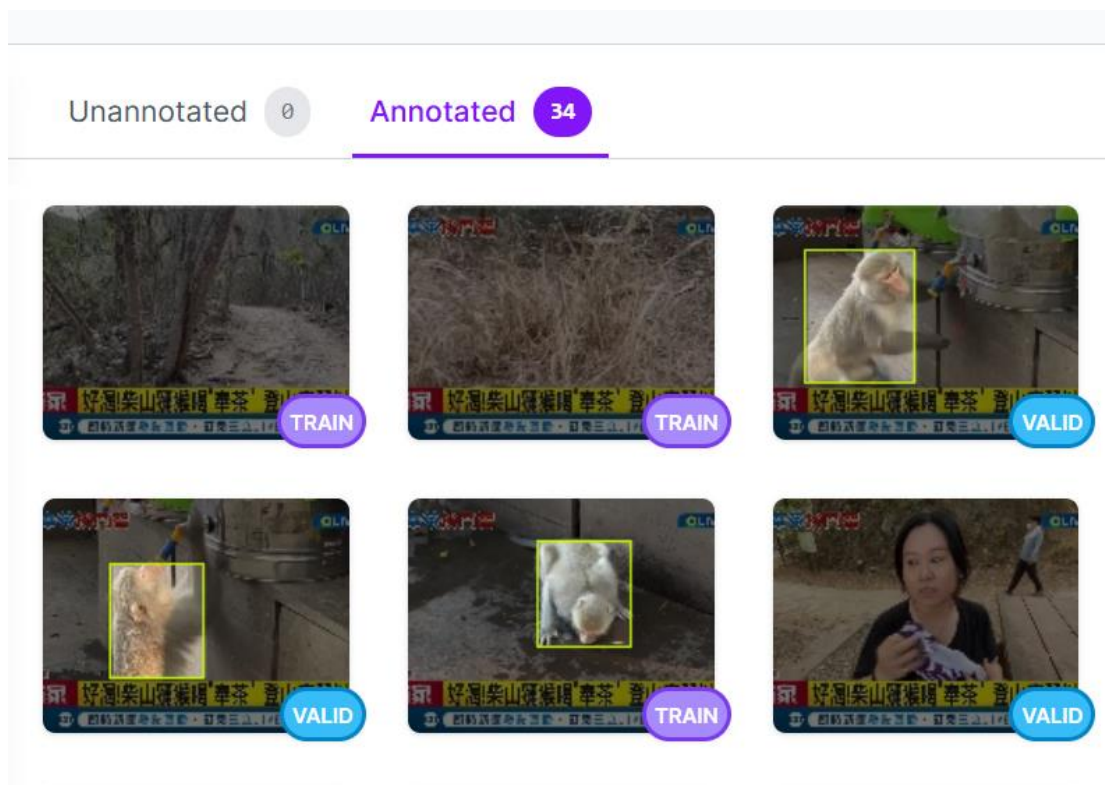
4.2.2 自定義資料分類訓練

利用自行標註的照片完成訓練任務

製作判斷猴子的影像辨識模型，標記照片中猴子的位置，用 yolo 進行訓練；我們亦有準備不含猴子的照片給模型訓練。



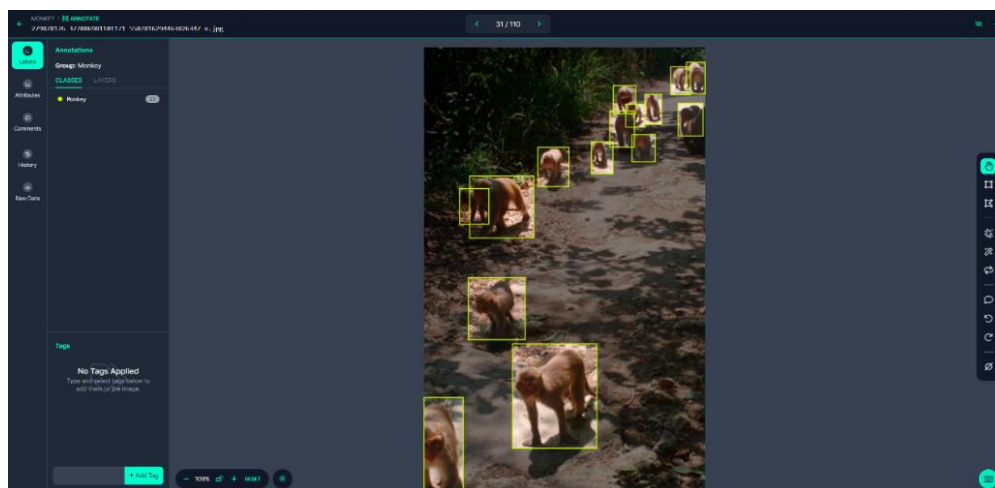
▲圖片為人工標記圖片中有猴子的地方，以一隻猴子為單位框選。

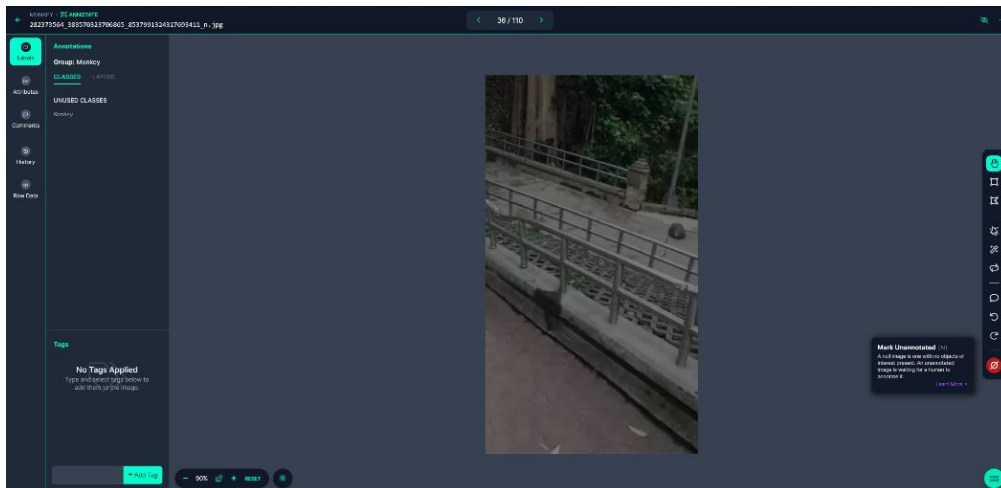


▲圖片為包含猴子及不包含猴子的照片資料

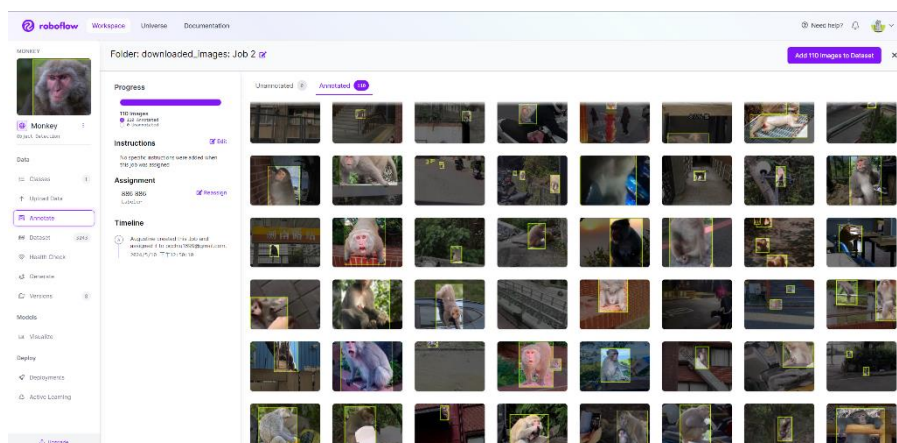
4.2.3 資料標註:

我們使用使用 Roboflow 的內置標註工具手動標記照片中的猴子位置，對圖片中的每一個猴子進行手動框選。





若圖片中沒有猴子，則標註為 mark unannotated。



▲標記完成後的資料集

4.3 資料增強與預處理

我們將資料以 70%訓練集、20%驗證集、10%測試集的比例分割，再使用 Roboflow 進行資料增強和處理，再將標記好的資料集 Export 成 YOLOv8 格式，然後下載資料集以進行模型訓練。關於資料預處理與增強的具體設定如下

資料預處理:

- 靜態裁剪 (Static Crop): 25-75% 水平區域和垂直區域
- 調整大小 (Resize): 調整到 640x640
- 自動調整對比度 (Auto-Adjust Contrast): 使用對比度拉伸
- 過濾無效資料 (Filter Null): 確保至少 30% 的圖像包含標註

資料增強:

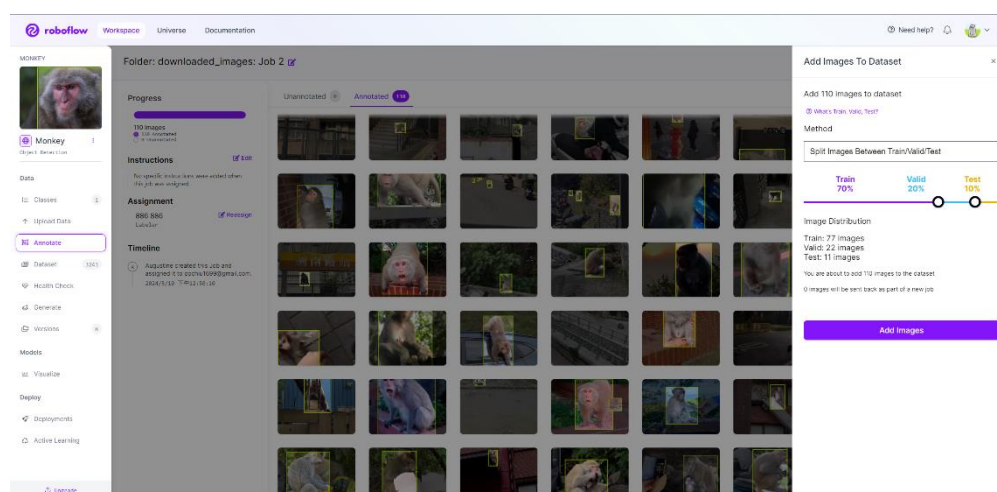
- 水平和垂直翻轉 (Flip)
- 旋轉 90° (Clockwise, Counter-Clockwise, Upside Down)
- 裁剪 (Crop): 0% Minimum Zoom, 20% Maximum Zoom
- 旋轉 (Rotation): -15° 到 +15°
- 剪切 (Shear): ±15° 水平和垂直
- 灰度 (Grayscale): 應用於 15% 的圖像
- 飽和度 (Saturation): -20% 到 +20%
- 亮度 (Brightness): -15% 到 +15%
- 曝光 (Exposure): -15% 到 +15%
- 模糊 (Blur): 最多 1.5 像素
- 噪聲 (Noise): 最多 1.1% 的像素

```
Preprocessing  Auto-Orient: Applied
                Static Crop: 25-75% Horizontal Region, 25-75% Vertical Region
                Resize: Stretch to 640x640
                Auto-Adjust Contrast: Using Contrast Stretching
                Filter Null: Require at least 30% of images to contain annotations.
```

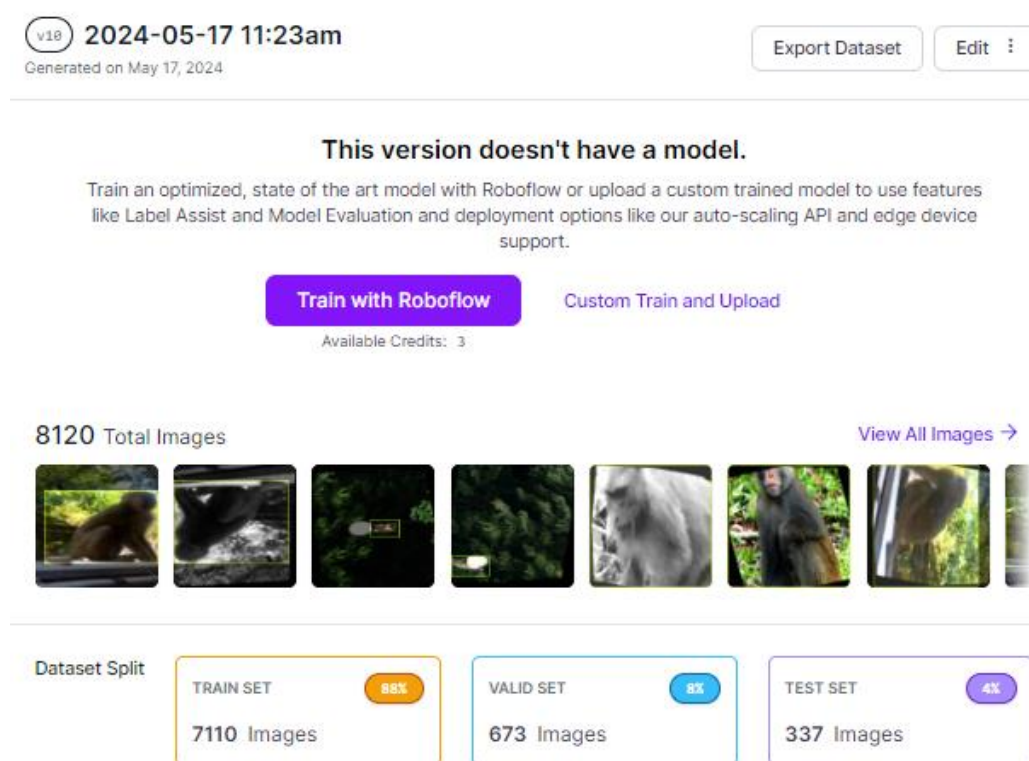
```
Augmentations  Outputs per training example: 3
                Flip: Horizontal, Vertical
                90° Rotate: Clockwise, Counter-Clockwise, Upside Down
                Crop: 0% Minimum Zoom, 20% Maximum Zoom
                Rotation: Between -15° and +15°
                Shear: ±15° Horizontal, ±15° Vertical
                Grayscale: Apply to 15% of images
                Saturation: Between -20% and +20%
                Brightness: Between -15% and +15%
                Exposure: Between -15% and +15%
                Blur: Up to 1.5px
                Noise: Up to 1.1% of pixels
```

4.4 資料分配

將資料集分配到訓練集、驗證集、測試集，我們將比例設定為 train 70%、valid 20%、test 10%。



1. 將標記好的資料集 Export 成 YoloV8 的格式點擊 Export Dataset



Dataset:(Train:7110):(Valid:673):(Test:337)

4.5 模型訓練:

使用 Ultralytics 對模型進行訓練，訓練過程如下:

- 模型概述：
225 層，3011043 個參數，3011027 個梯度，8.2 GFLOPs（每秒運算 10 億次浮點運算）。
- 轉移預訓練權重：
從預訓練權重中轉移了 319/355 項。
凍結層 "model.22.dfl.conv.weight"。
- AMP（自動混合精度）：
運行自動混合精度檢查並通過。
- AutoBatch（自動批處理）：
計算 `imgsz=640` 的最佳批處理大小。
使用 CUDA:0（NVIDIA GeForce RTX 4090），分配了 23.59GB 顯存。
- 數據加載：
掃描訓練集的標籤文件，發現 7087 個圖像，1148 個背景圖像。
掃描驗證集的標籤文件，發現 673 個圖像，122 個背景圖像
- 數據緩存：
創建並緩存訓練數據集和驗證數據集的緩存文件。
- 優化器設置：
使用 'SGD'（隨機梯度下降）優化器，自動確定最佳的 'lr'（學習率）和 'momentum'（動量）。
- 訓練準備：
使用 8 個數據加載器工作。
開始訓練 550 個 epochs。
- Batch Size (批次大小)：
自動計算得出的最佳批次大小為 122。

- 優化器參數 (Optimizer Parameters) :
使用的是 SGD (Stochastic Gradient Descent) 優化器，其參數設置如下：
 - Learning Rate (初始學習率, lr0): 0.01
 - Momentum (動量): 0.9
 - Weight Decay (權重衰減): 0.00053125
 - Bias Decay (偏置項衰減): 0.0

```
Model summary: 225 layers, 3011043 parameters, 3011027 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
AMP: checks passed
AutoBatch: Computing optimal batch size for imgsz=640
AutoBatch: CUDA:0 (NVIDIA GeForce RTX 4090) 23.99G total, 0.33G reserved, 0.07G allocated, 23.59G free
  Params  GFLOPs  GPU_mem (GB)  forward (ms)  backward (ms)  input  output
  3011043    8.194    0.428    17.58    110.8    (1, 3, 640, 640)    list
  3011043   16.39    0.451    16.19    17.53    (2, 3, 640, 640)    list
  3011043   32.78    0.694     6.09    22.61    (4, 3, 640, 640)    list
  3011043   65.55    1.193    4.555    27.82    (8, 3, 640, 640)    list
  3011043  131.1    2.072    12.12    27.36   (16, 3, 640, 640)    list
AutoBatch: Using batch-size 122 for CUDA:0 14.45G/23.99G (60%)
train: Scanning C:\Users\user\Desktop\Monkey\datasets\Monkey_v101.yolov8\train\labels... 7087 Images, 1148 backgrounds, 0 corrupt: 100%|██████████| 7087/7087 [00:04:00:00, 1616.79it/s]
train: WARNING C:\Users\user\Desktop\Monkey\datasets\Monkey_v101.yolov8\train\images\large-11-.jpeg.cf.4c700534b2e27126f5d8b4d4c3d6f5d3.jpg: 1 duplicate labels removed

train: New cache created: C:\Users\user\Desktop\Monkey\datasets\Monkey_v101.yolov8\train\labels.cache
train: Caching images (8.16G Disk): 100%|██████████| 7087/7087 [00:05:00:00, 1278.16it/s]
val: Scanning C:\Users\user\Desktop\Monkey\datasets\Monkey_v101.yolov8\valid\labels... 673 Images, 122 backgrounds, 0 corrupt: 100%|██████████| 673/673 [00:00:00:00, 1167.44it/s]
val: New cache created: C:\Users\user\Desktop\Monkey\datasets\Monkey_v101.yolov8\valid\labels.cache

val: Caching images (0.86G Disk): 100%|██████████| 673/673 [00:00:00:00, 1401.04it/s]
Plotting labels to runs\detect\train\labels.jpg...
optimizer: 'optimizer-auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: SGD(lr=0.01, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.00053125), 63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs\detect\train
Starting training for 550 epochs...
```

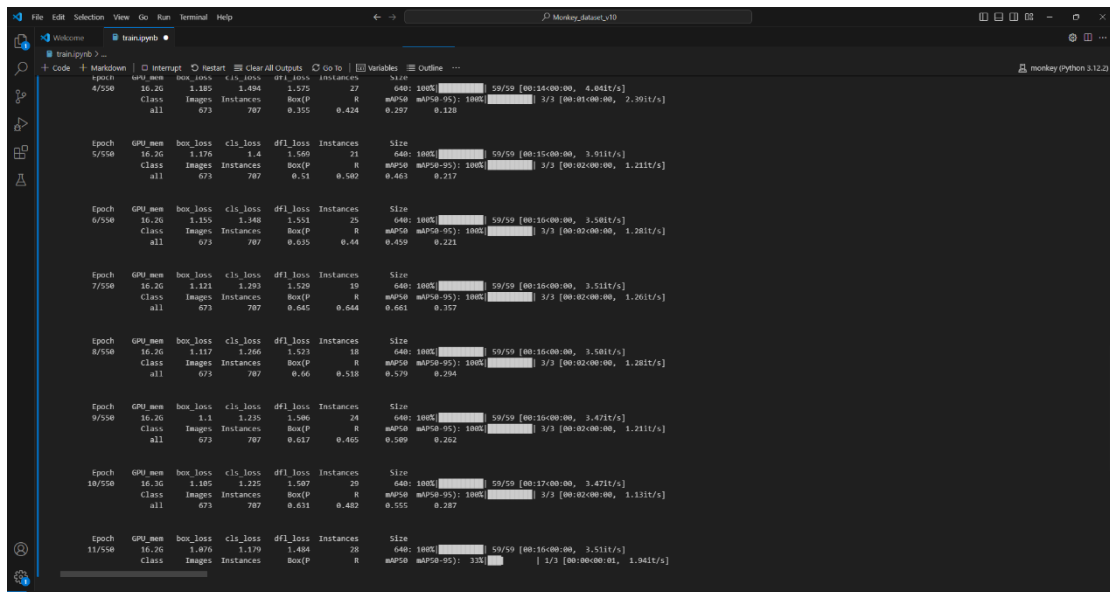
訓練環境 Packages 版本：

```
(monkey) C:\Users\user>conda list ultralytics
# packages in environment at C:\Users\user\anaconda3\envs\monkey:
#
# Name                Version           Build    Channel
ultralytics           8.1.47            pypi_0   pypi

(monkey) C:\Users\user>conda list cuda
# packages in environment at C:\Users\user\anaconda3\envs\monkey:
#
# Name                Version           Build    Channel
cuda-cccl             12.4.127          0        nvidia
cuda-cudart           12.1.105          0        nvidia
cuda-cudart-dev       12.1.105          0        nvidia
cuda-cupti            12.1.105          0        nvidia
cuda-libraries        12.1.0            0        nvidia
cuda-libraries-dev    12.1.0            0        nvidia
cuda-nvrtc            12.1.105          0        nvidia
cuda-nvrtc-dev       12.1.105          0        nvidia
cuda-nvtx            12.1.105          0        nvidia
cuda-opengl           12.4.127          0        nvidia
cuda-opengl-dev      12.4.127          0        nvidia
cuda-profiler-api     12.4.127          0        nvidia
cuda-runtime          12.1.0            0        nvidia
pytorch-cuda          12.1              hde6ce7c_5  pytorch

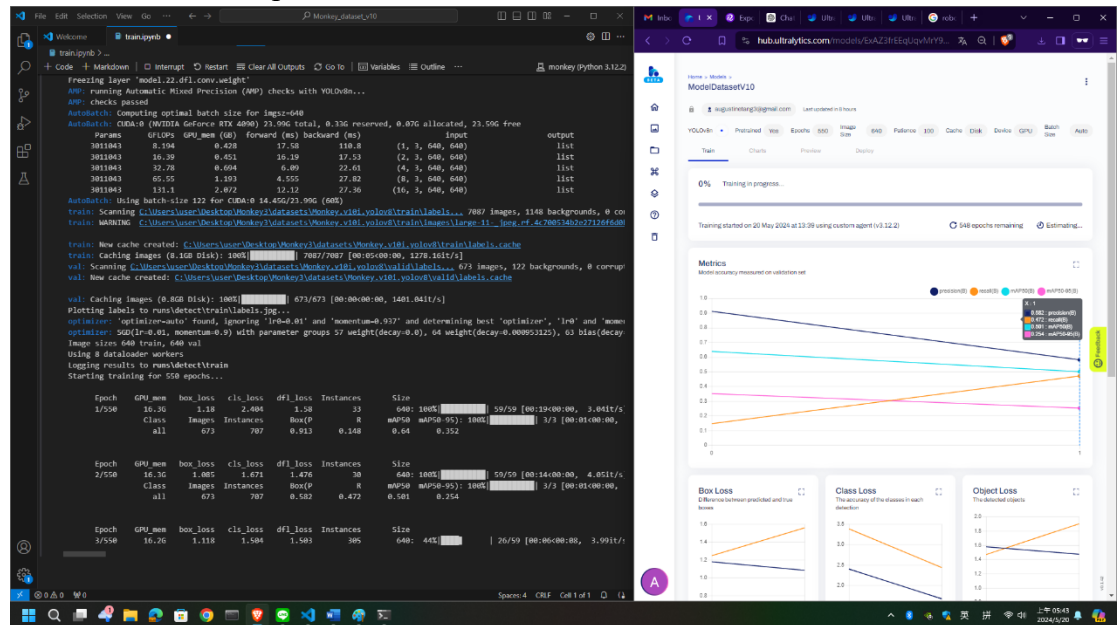
(monkey) C:\Users\user>conda list torch
# packages in environment at C:\Users\user\anaconda3\envs\monkey:
#
# Name                Version           Build    Channel
pytorch              2.2.2            py3.12_cuda12.1_cudnn8_0  pytorch
pytorch-cuda         12.1             hde6ce7c_5  pytorch
pytorch-mutex        1.0              cuda       pytorch
torchaudio           2.2.2            pypi_0    pypi
torchvision          0.17.2           pypi_0    pypi

(monkey) C:\Users\user>
```

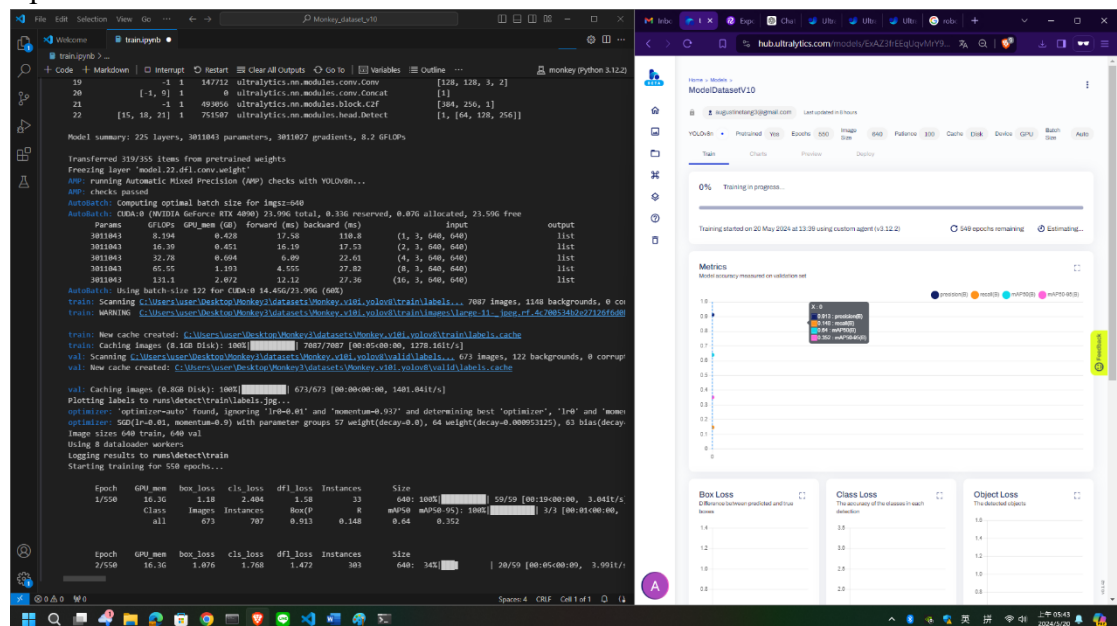


5. 效能評估

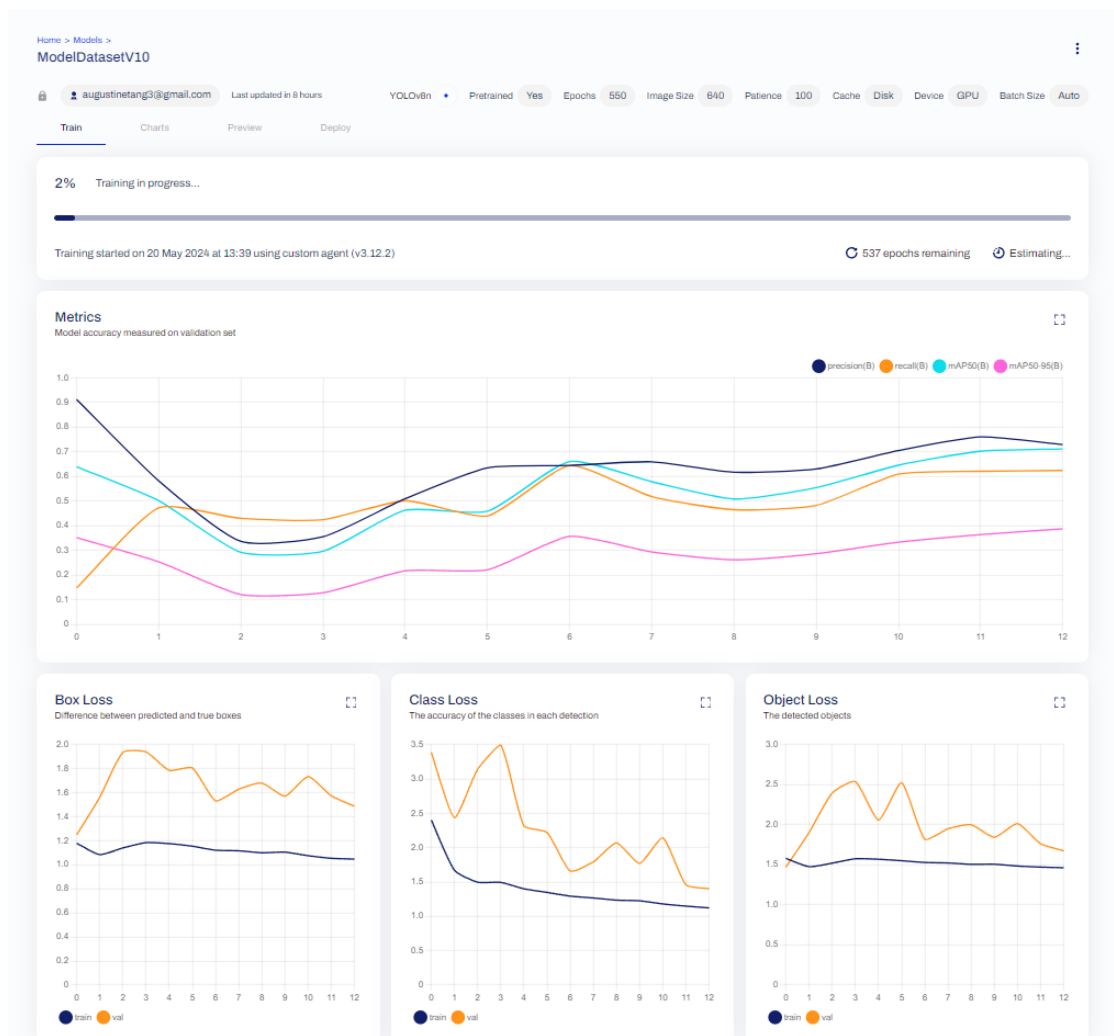
開始成功訓練幾次 epoch 后看到 Hub 可以實時監控訓練情況：



Epoch 1:



Epoch 12 的訓練情況：



Epoch 177 左右的訓練情況：

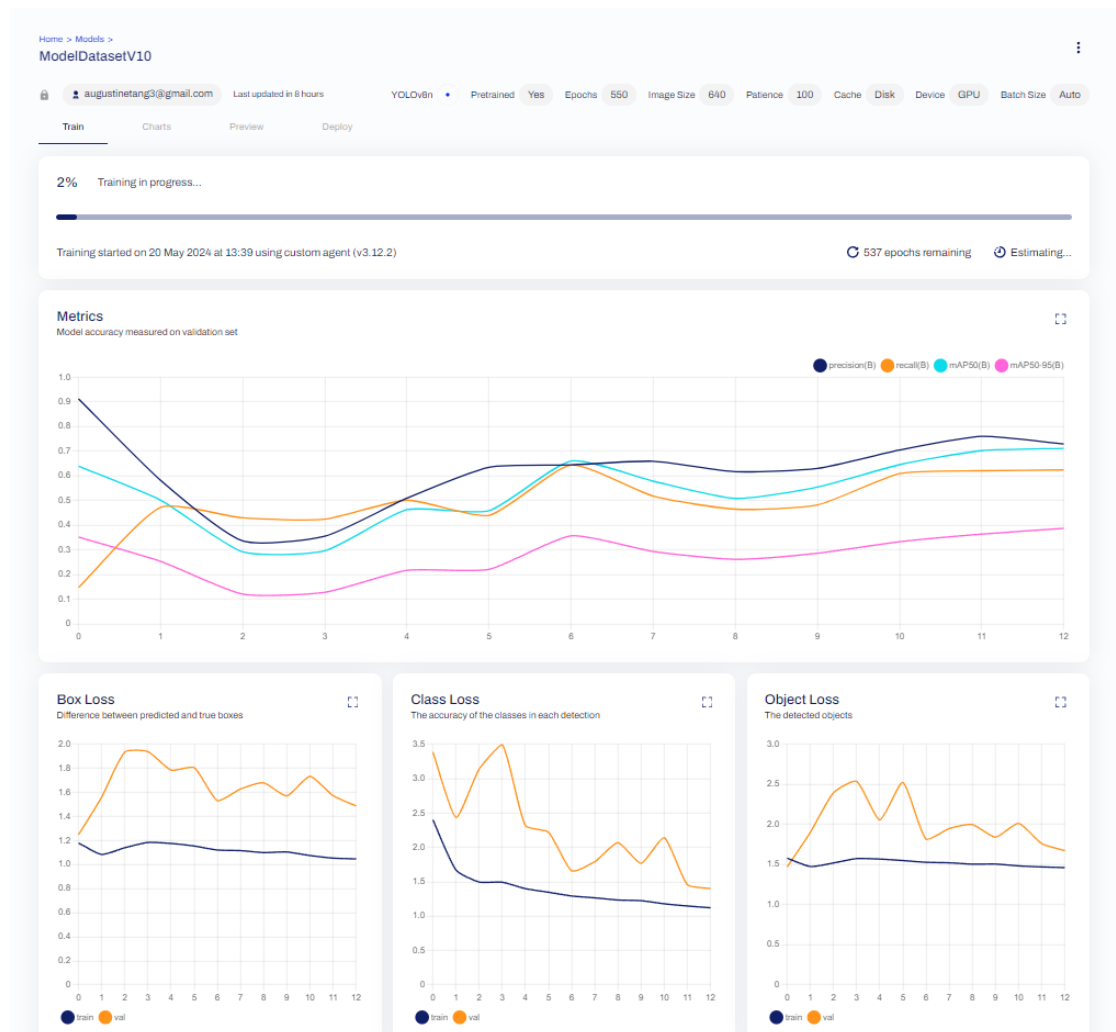


Epoch 300 左右的訓練情況：



以下是 Ultralytics Hub 實時監控訓練的畫面：

如果今天在桌電訓練使用者可在手機上監控訓練怎麼樣目前怎麼樣。



成果截圖
訓練耗時 101 分鐘。

```
from ultralytics import YOLO, checks, hub
checks()

hub.login('0e1bf0ca077b412e5ac43d5d361742a759854cdc')

model = YOLO('https://hub.ultralytics.com/models/ExA23frEEqUqMhY9NwC')
results = model.train()

✓ 10m5h

Ultralytics YOLOv8.1.47 Python-3.12.2 torch-2.2.2 CUDA:0 (NVIDIA GeForce RTX 4090, 24564MiB)
Setup complete (32 CPUs, 63.8 GB RAM, 218.2/553.0 GB disk)
Ultralytics HUB: New authentication successful
Ultralytics HUB: View model at https://hub.ultralytics.com/models/ExA23frEEqUqMhY9NwC
Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt to 'yolov8n.pt'...
100% |#####| 6.23M/6.23M (00:02:00:00, 2.34MB/s)
New https://pypi.org/project/ultralytics/8.2.18 available Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.1.47 Python-3.12.2 torch-2.2.2 CUDA:0 (NVIDIA GeForce RTX 4090, 24564MiB)
Engine trainer: task=detect, mode=train, model=yolov8n.pt, data=https://storage.googleapis.com/ultralytics-hub-ckpt/models/ExA23frEEqUqMhY9NwC/monkey_v10l_yolov8n.pt, epochs=550, time=None,
Downloading https://storage.googleapis.com/ultralytics-hub-ckpt/models/ExA23frEEqUqMhY9NwC/monkey_v10l_yolov8n.pt to 'monkey_v10l_yolov8n.pt'...
100% |#####| 46.7M/46.7M (00:37:00:00, 13.1MB/s)
Unpacking monkey_v10l_yolov8n.pt to C:\Users\user\Desktop\monkey\datasets\monkey_v10l_yolov8n... 100% |#####| 16200/16200 (00:03:00:00, 4177.68file/s)
Overriding model.yaml nc=80 with nc=1

Class      Images  Instances  Box(P   R   mAP50  mAP50-95) 100% |#####| 3/3 (00:01:00:00, 2.491t/s) all 673 787 0.931 0.866 0.929
Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
354/550    16.2G   0.564    0.4369    1.876     18      640: 100% |#####| 59/59 (00:14:00:00, 4.151t/s)
Class      Images  Instances  Box(P   R   mAP50  mAP50-95) 100% |#####| 3/3 (00:01:00:00, 2.531t/s) all 673 787 0.924 0.867 0.929
Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
355/550    16.2G   0.557    0.4359    1.875     24      640: 100% |#####| 59/59 (00:14:00:00, 4.141t/s)
Class      Images  Instances  Box(P   R   mAP50  mAP50-95) 100% |#####| 3/3 (00:01:00:00, 2.461t/s) all 673 787 0.926 0.867 0.929
Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
356/550    16.4G   0.5542   0.4382    1.872     30      640: 100% |#####| 59/59 (00:14:00:00, 4.141t/s)
Class      Images  Instances  Box(P   R   mAP50  mAP50-95) 100% |#####| 3/3 (00:01:00:00, 2.491t/s) all 673 787 0.926 0.867 0.929
[34m] EarlyStopping: [0m] training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 256, best model saved as best.pt.
To update earlystopping(patience=100) pass a new patience value, i.e. 'patience=300' or use 'patience=0' to disable EarlyStopping.

356 epochs completed in 1.649 hours.

Optimizer stripped from runs\detect\train\weights\last.pt, 0.39M
Optimizer stripped from runs\detect\train\weights\best.pt, 0.39M

Validating runs\detect\train\weights\best.pt...
Ultralytics YOLOv8.1.47 Python-3.12.2 torch-2.2.2 CUDA:0 (NVIDIA GeForce RTX 4090, 24564MiB)
Model summary (fused): 168 layers, 3805843 parameters, 0 gradients, 8.1 GFLOPs
Class      Images  Instances  Box(P   R   mAP50  mAP50-95) 100% |#####| 3/3 (00:01:00:00, 1.881t/s)
all        673      787      0.926    0.86  0.933  0.781
Speed: 0.1ms preprocess, 0.4ms inference, 0.0ms loss, 0.5ms postprocess per image
Results saved to [0m] runs\detect\train\weights\best.pt
[34m] [0m] Ultralytics HUB: [0m] Uploading final model...
100% |#####| 5.99M/5.99M (00:01:00:00, 4.49M/s) [34m] [0m] Ultralytics HUB: [0m] Done
[34m] [0m] Ultralytics HUB: [0m] View model at https://hub.ultralytics.com/models/ExA23frEEqUqMhY9NwC
```

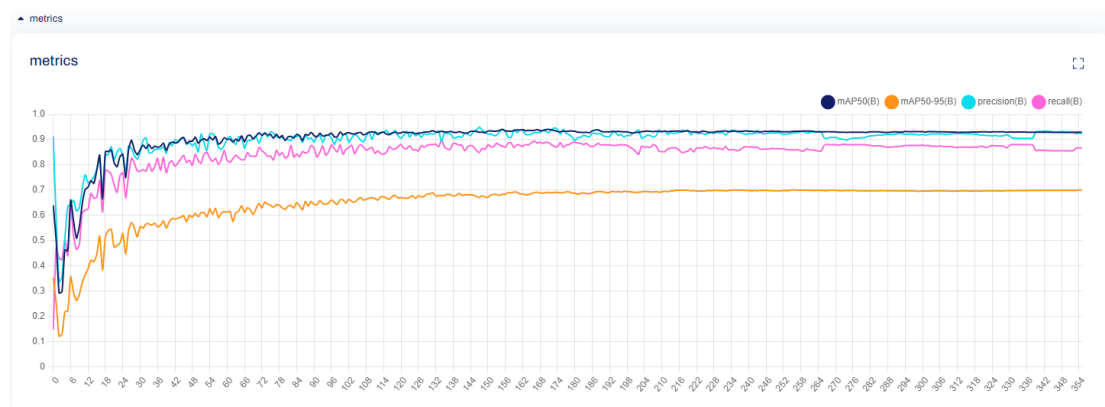
這張截圖顯示了使用 Ultralytics 進行 YOLOv8 模型訓練的 Log。訓練計劃進行 550 個 epoch，但由於早停（EarlyStopping），在性能沒有改善的情況下提前停止了訓練（這裡設置的耐心參數為 100 個 epoch）。最佳模型在第 256 個 epoch 保存，因為根據驗證指標，這是性能最好的結果。

Log 包括每個 epoch 的詳細性能指標，如 box loss、分類損失、DFL（Distribution Focal Loss）損失和在不同 IoU（交併比）閾值下的 mAP（平均精度）。最佳模型的最終驗證結果顯示，在 IoU=0.5 時的 mAP 為 0.926，在 IoU=0.5:0.95 時的 mAP 為 0.867。

Model Summary 顯示，它有 168 層，3,005,843 個參數，8.1 GFLOPs（每秒千兆浮點運算次數）。最終模型 best.pt 已保存並驗證，並同步到 Ultralytics HUB。

Ultralytics hub:

Metrics:



以上這張圖顯示了模型在訓練期間的各種性能指標，包括：

mAP50（深藍色）：在 IoU 為 0.5 時的平均精度

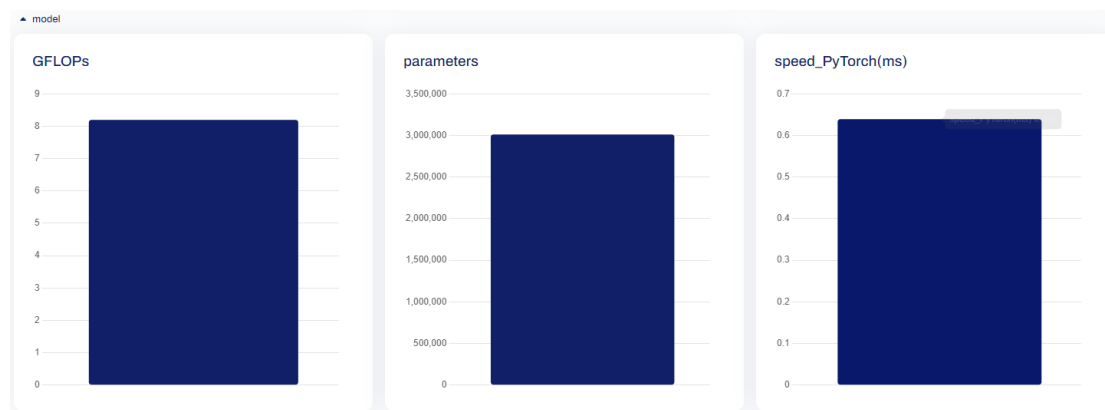
mAP50-95（橙色）：在 IoU 從 0.5 到 0.95 之間變化時的平均精度

precision（青色）：模型在預測正確的物體中，有多少是真正存在的物體

recall（粉紅色）：在所有實際的物體中，模型能夠正確識別出多少

從圖中可以看出，這些指標在初始階段迅速提升，隨後逐漸趨於穩定，表示模型的性能隨著訓練的進行而改善並最終穩定。

Model Information:



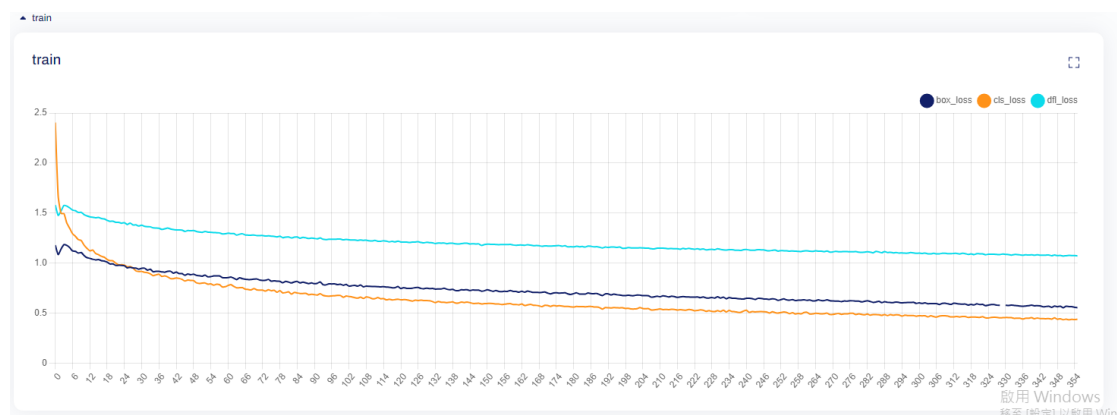
以上這張圖顯示了模型的一些基本信息：

GFLOPs：模型每秒執行的千兆浮點運算次數，顯示為約 8 GFLOPs。

parameters：模型的參數總數，約為 3,000,000。

speed_PyTorch(ms)：模型在 PyTorch 中的運行速度，顯示為約 0.6 毫秒。

Train Loss:



以上這張圖顯示了模型在訓練期間的損失變化，包括：

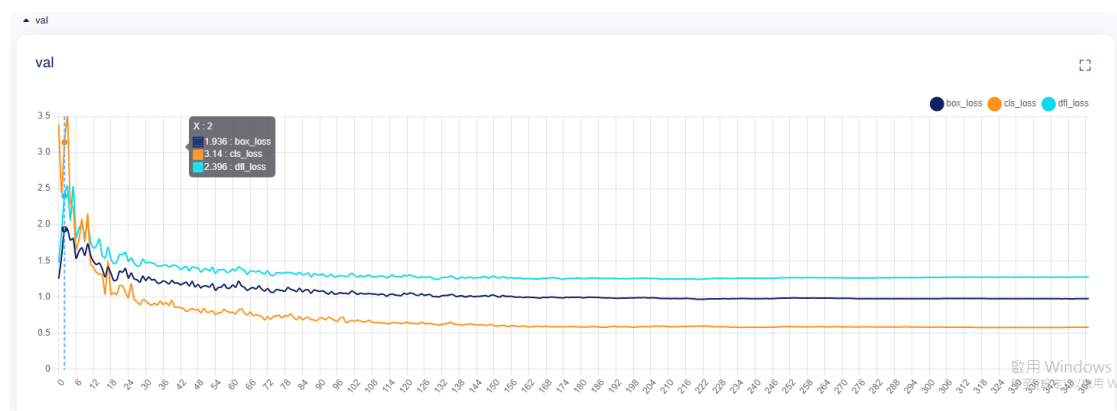
box_loss（深藍色）：框損失

cls_loss（橙色）：分類損失

dfl_loss（青色）：DFL（Distribution Focal Loss）損失

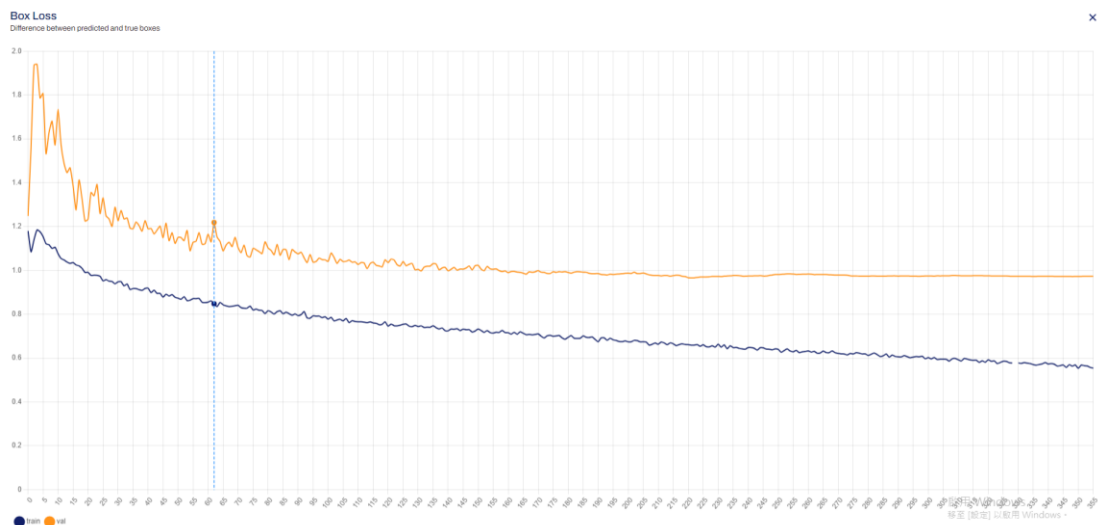
圖中顯示這些損失在初始階段迅速下降，隨後逐漸趨於穩定，表示模型的誤差隨著訓練的進行而減少並最終穩定。

Validation Loss:



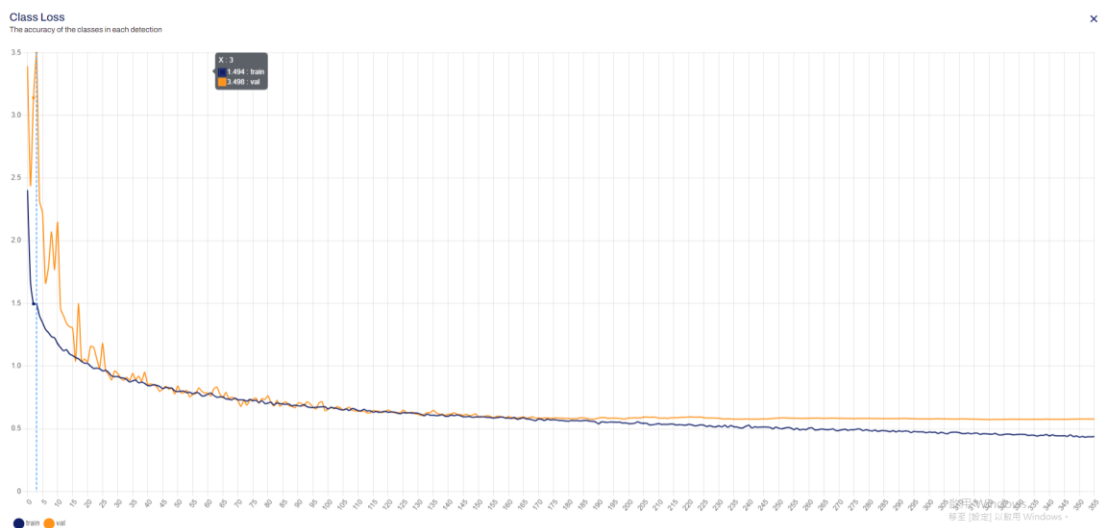
這張圖顯示了模型在驗證期間的損失變化，與訓練損失類似，展示了模型在驗證數據上的表現。

Box Loss:



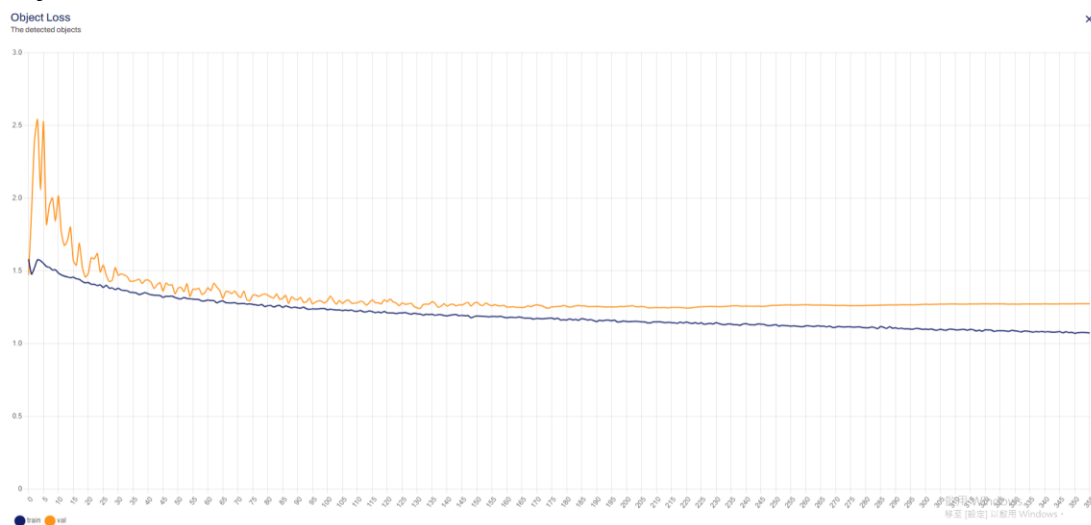
以上這張圖詳細顯示了框損失在訓練和驗證期間的變化，訓練損失和驗證損失均顯示出逐漸下降並趨於穩定的趨勢。

Class Loss:



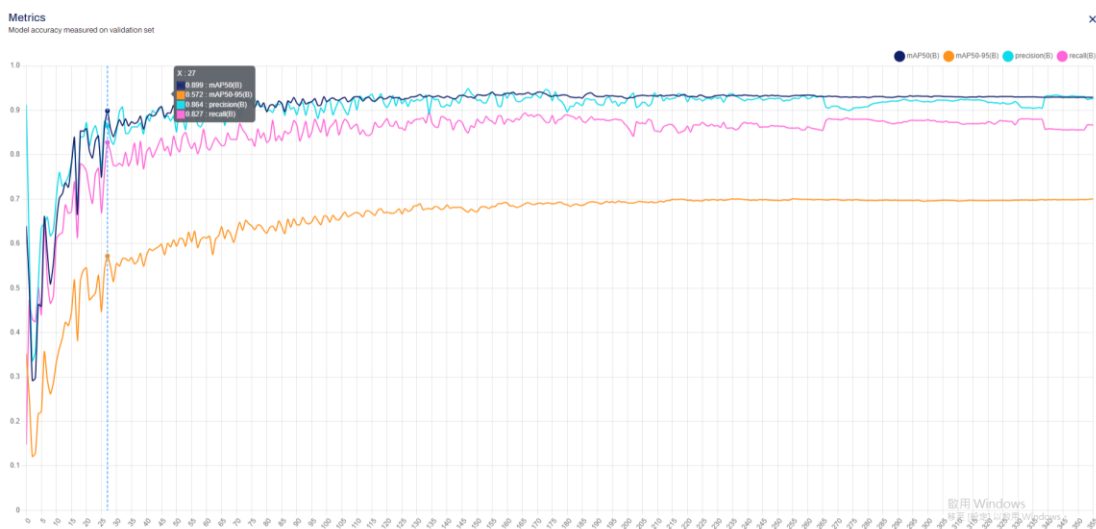
以上這張圖詳細顯示了分類損失在訓練和驗證期間的變化，同樣顯示出損失逐漸減少並趨於穩定的趨勢。

Object Loss:



以上這張圖詳細顯示了物體損失在訓練和驗證期間的變化，顯示出隨著訓練的進行損失逐漸減少並最終穩定。

Metrics (第二張):



以上這張圖與第一張圖類似，顯示了模型在訓練期間的各種性能指標，展示了 mAP、精度和召回率等指標的變化趨勢。

6. 方法實現成果與 demo

6.1 Line Notify 通知

攝影機偵測到猴子時，系統會自動拍攝猴子的影像，並發送通知提醒使用者避開該區域。

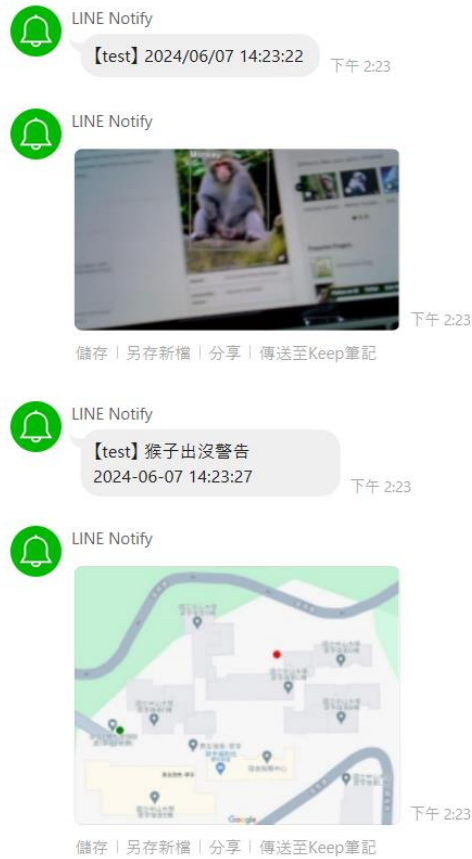


圖 6-1 Line Notify 偵測通知

若一段時間沒偵測到猴子，也會發送通知告知使用者已能安全通過

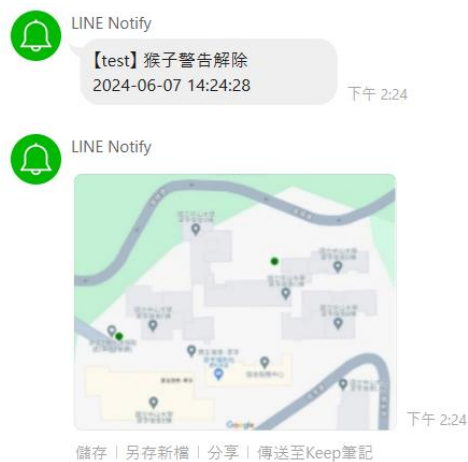


圖 6-2 Line Notify 解除通知

6.2 猴子分佈地圖

猴子分布地圖可以讓使用者快速知道哪裡有猴子，另外如有更多攝影機的設備要擴增也可以方便部屬，偵測到猴子的位置會顯示為紅色，而未偵測的位置則會顯示為綠色。



圖 6-3 猴子分佈地圖

6.3 上鎖功能與影片演示

https://www.youtube.com/watch?v=yn_bBYU0wsU

7. 心得與討論

黃啟桓：

我是負責 Server 的建置，主要的麻煩在於“Server 需要可以被外網連接”才可以實現遠端的傳輸。這部分是使用 ngrok 的服務，蠻方便的，並且可以固定 domain，使得邊緣裝置(樹莓派)不用因為 Server 換 IP，每次都改成程式碼。除此之外，Server 對於個邊緣裝置的定位是使用 mac address，這也使得 Server 可以辨別各邊緣裝置，進而用於地圖圖片定位，這個方法也可以用於多個裝置並且各裝置不需更改程式碼。

了解 Roboflow、Ultralytics 這兩個工具。Roboflow 是一個專門針對電腦視覺應用的工具平台，提供了多種功能來幫助用戶處理、標註、增強圖像數據、

數據集管理、模型訓練、預訓練模型、部署與推理。Ultralytics 是一個強大的數據分析平台。數據整合、數據可視化、數據分析、自動報告生成、機器學習、協作與共享。

訓練資料的獲取來自於網路上影片或圖片，並手動進行標註。對於精準度的提升，是漸進式的修正。一開始，人也會被辨識成猴子。所以加入一定數量的人的圖片，並且不進行標註，使模型訓練時知道人不是屬於猴子。後來，對於深色皮膚的圖片，也會被辨識成猴子，所以加入一定數量的深色皮膚的圖片，並且不進行標註，使模型訓練時知道此不是屬於猴子。到後面精確度已經很高了，每次 demo 的時候都可以成功辨識到猴子，並且沒誤判。但沒想到最後一次 demo 拍影片時，竟然將我的手辨識成猴子，超乎我的意料，看來模型還是有不足的地方，所以就保留這部分的影片。

徐睿鍾：

這次我的工作負責標記猴子的照片，這是我第一次從事標記工作。這次讓我深刻感受到 Robotflow 這個平台的便利性和高效性。Robotflow 的共同標記功能非常強大，允許多個人同時參與標記工作，這大大提高了我們的工作效率。我和我的組員可以實時共享進度，一起完成大量的標記任務，這對於大規模數據集的處理非常有幫助。其次，Robotflow 支持上傳包含猴子的影片，並自動切割成所需數量的照片。這個功能極大地簡化了我們數據準備的過程，讓我們可以更專注於標記本身，而不必花費大量時間在預處理上。標記完成後，這些照片可以方便地用於模型訓練。Robotflow 的整合性設計使得數據從標記到訓練的流程變得非常順暢。這讓我們需要快速迭代和測試模型的情況來說超有幫助。關於我們做的宿舍猴子偵測系統，其優點也顯而易見。這個系統可以有效地偵測和識別猴子的存在，這對於保護宿舍學生的安全非常重要。隨著猴子頻繁出現在宿舍區域，它們可能對學生的生活和財產造成威脅。這套系統能夠及時發出警報，提醒帶食物回去的學生提前採取措施，從而降低潛在的食物被搶的危險。食物被搶可能造成心情不好；心情不好書就讀不好；書讀不好就無法順利地完成學校考試；最後無法畢業。所以我們的東西非常有貢獻。最後感謝這堂課以及我的組員一起完成了期末報告。

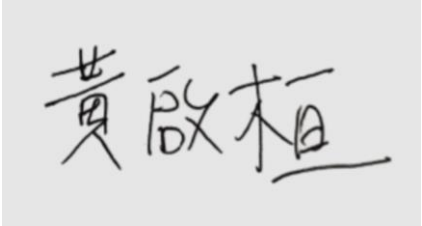

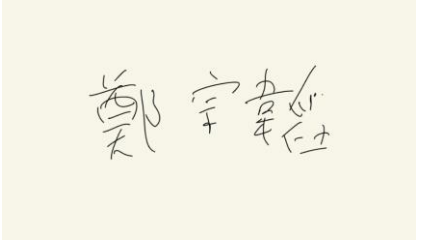
鄭宗韜：

我是負責 GUI,模型訓練,收集資料,標記資料&整體 project 的架構建設與構想。開發一個偵測動物的系統,目的是防止猴子破壞物品/偷食物相信都在中山讀書大家都少都有這個走在路上害怕猴子的這心情。所以我就想說利用這堂課所學到的 ML 方法來解決這個問題。這個系統主要用 NXP 的 GuiGuider 設計控制面板的 GUI 部分,而其他部分,比如 Raspberry Pi 4 和 5 以及鏡頭的應用,都是用 Python 在 Raspberry Pi 上完成的。利用 NXP 的 GuiGuider 設計控制面板的 GUI,不僅提升了系統的互動性,還讓操作變得更簡單直觀。GuiGuider 提供了豐富的元件和靈活的設計選項,讓我們可以用最簡單的方式展示複雜的功能,用戶可以輕鬆進行各種操作和設置。在硬體選擇上,我用了 Raspberry Pi 4 和 5。這些小型電腦處理能力強、擴展性好,非常適合這類偵測系統。連接高解析度鏡頭後,我們可以精確偵測環境中的動物。在模型訓練方面,我們收集了大量猴子的影像資料與照片,主要來源是中山猴猴日記,詳細標記後,利用深度學習技術訓練模型。這些模型能夠有效識別猴子,並及時發出警報,防止猴子破壞設施或物品。雖然資料收集和標記過程繁瑣,但高質量的資料為模型的高準確度提供了堅實基礎。每個環節都經過反覆測試和優化,保證系統的穩定性和高效性,並提供了一個用戶友好、操作簡便的解決方案。總的來說,這個偵測動物的系統結合了先進的硬體設備和優秀的軟體設計,通過 NXP GuiGuider 設計的控制面板讓操作更直觀簡單,利用 Raspberry Pi 和 Python 的強大功能實現高效動物偵測,為防止猴子破壞物品提供了可靠的解決方案。這次的開發經歷,不僅提升了我的專業技能,也讓我對未來更多科技應用充滿了期待。這次是一個突發異想所設計出來的一個系統整體上蠻有趣的雖然標記資料的時候很浪費時間就對了。

邱柏翔：

我的工作是負責標記猴子照片與書面資料的整理,也是在這次工作中認識到 Roboflow 等平台。由於這次的資料集需要手動準備,這些工具對於資料集的產生提供了相當的幫助,本身影片分割的功能能透過影片直接快速得到多個不同角度的照片,讓我們能直接從 Facebook 或 YouTube 等平台取得影片資料,相較於需要單獨上傳的照片大幅提高了便利性,也節省了大量的時間,協助我們快速準備資料集,雖然最終仍須要手動逐一標記資料,但已經提供了很大的便利,除此之外 Roboflow 帶有快速選擇資料預處理與資料增強項目的功能,而不必手動對於各個項目進行手動處理,大大節省了這部分需要花的心思,也讓我們能更容易的完成資料集的準備。這次專題是我第一次接觸機器學習相關內容,對於需要較高技術面能力的工作較勝任,但過程中學長也幫忙解決了許多困難的問題,在整理資料時也看了許多學長提供的相關資料,大幅增加了對於機器學習相關的了解,也在協助的過程中接觸到各種實用的工具與平台,對於以後要深入了解機器學習相關的內容也提供了不小的幫助,學長在專案後期提出的 NXP 面板與 server 的使用也讓我大開眼界,先前多半是只接觸到單純對於圖像識別相關的內容,這次也了解了機器學習與其他技術結合後能達成的更多便利功能,也感謝學長在過程中的指點與協助,讓原先還不太了解機器學習的我也能參與到這個系統的製作。

8. 分工比例&簽名

	分工	比例	簽名
B083022053 黃啟桓	Server 建置、樹莓派的程式微調，使其能與 Server 互動。	25%	
B093040038 徐睿鍾	幫忙部分猴子照片的標記、期末報告 ppt 製作	25%	
B093040056 鄭宗韜	GUI,模型訓練，收集資料，標記資料&整體 project 的架構建設與構想	25%	
B123040042 邱柏翔	幫忙部分猴子照片的標記、書面資料整理	25%	