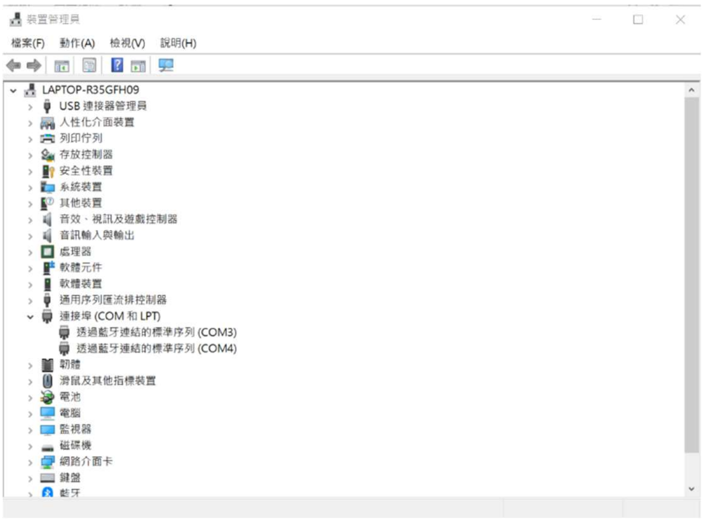
第二組_深度學習與類神經網路實作

成員: 周家禾(113368507)、謝昀燊(113368517)、周振強(113368519)、謝逸婷(114368533)

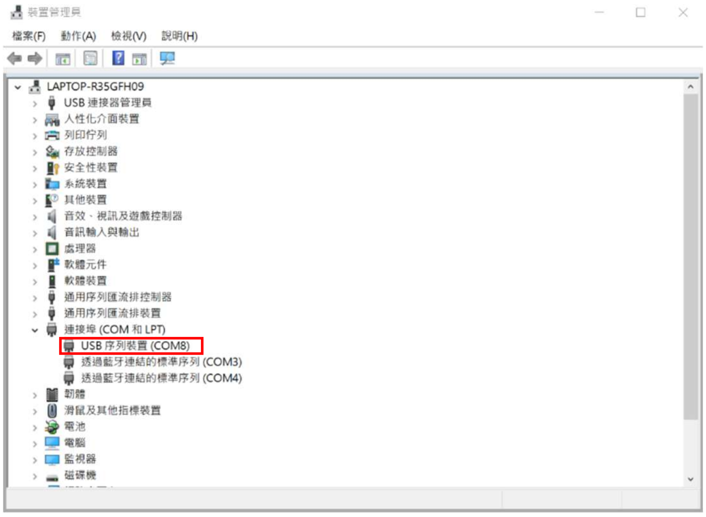
ESP32-P4-EYE 視覺開發板燒錄方法(windows)

1、確認 ESP32-P4-EYE 連接埠編號方式：對「開始」點擊右鍵，點選「裝置管理員」，點開「連接埠(COM 和 LPT)」，接著插拔裝置，就會顯示出現在這個 port 的代號。

2、下載「putty」程式來接收 ESP32-P4-EYE 的訊息，下載網址：

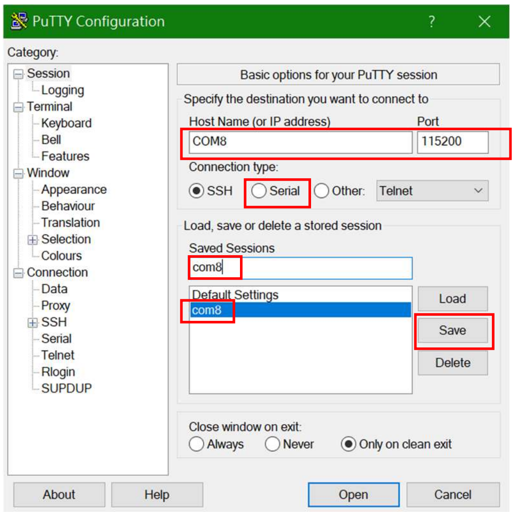


LAPTOP-R35GFH09
- USB 連接器管理員
- 人性化介面裝置
- 列印佇列
- 存放控制器
- 安全性裝置
- 系統裝置
- 其他裝置
- 音效、視訊及遊戲控制器
- 音訊輸入與輸出
- 處理器
- 軟體元件
- 軟體裝置
- 通用序列匯流排控制器
- 連接埠 (COM 和 LPT)
  - 透過藍牙連結的標準序列 (COM3)
  - 透過藍牙連結的標準序列 (COM4)
- 韌體
- 滑鼠及其他指標裝置
- 電池
- 電腦
- 監視器
- 磁碟機
- 網路介面卡
- 鍵盤
- 藍牙



LAPTOP-R35GFH09
- USB 連接器管理員
- 人性化介面裝置
- 列印佇列
- 存放控制器
- 安全性裝置
- 系統裝置
- 其他裝置
- 音效、視訊及遊戲控制器
- 音訊輸入與輸出
- 處理器
- 軟體元件
- 軟體裝置
- 通用序列匯流排控制器
- 通用序列匯流排裝置
- 連接埠 (COM 和 LPT)
  - USB 序列裝置 (COM8)
  - 透過藍牙連結的標準序列 (COM3)
  - 透過藍牙連結的標準序列 (COM4)
- 韌體
- 滑鼠及其他指標裝置
- 電池
- 電腦
- 監視器
- 磁碟機

https://putty.org/index.html，安裝完後點開設定，設定完可以按 save 存起來重複使用。



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): COM8

Port: 115200

Connection type:
○ SSH ● Serial ○ Other: Telnet

Load, save or delete a stored session

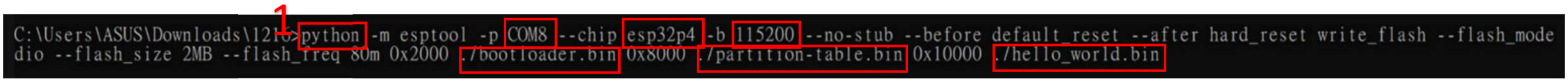
Saved Sessions: com8

Default Settings
com8

Close window on exit:
○ Always ○ Never ● Only on clean exit

3、燒錄指令：python -m esptool -p COM8 --chip esp32p4 -b 115200 --no-stub --before default_reset --after hard_reset write_flash --flash_mode dio --flash_size 2MB --flash_freq 80m 0x2000 ./bootloader.bin 0x8000 ./partition-table.bin 0x10000 ./hello_world.bin

其實 python 在各作業系統的終端機指令幾乎都相同，所以老師寫的 macOS 指令在 windows 中也是同樣的指令，如下圖。

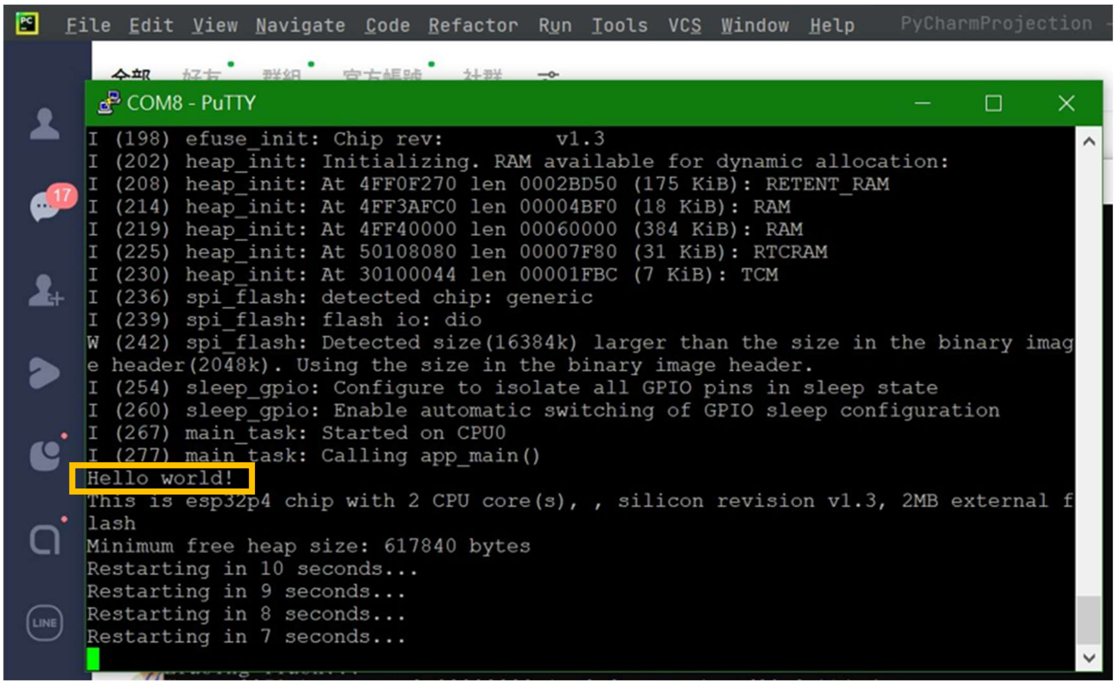


```
C:\Users\ASUS\Downloads\1216>python -m esptool -p COM8 --chip esp32p4 -b 115200 --no-stub --before default_reset --after hard_reset write_flash --flash_mode
dio --flash_size 2MB --flash_freq 80m 0x2000 ./bootloader.bin 0x8000 ./partition-table.bin 0x10000 ./hello_world.bin
```

紅框內為不同環境或設定，而會有改變的部份。特別解釋 1 的紅框，因為我的環境變數有設定

python 路徑，所以在 CMD 上直接打 python，它會去找當時設定路徑時的 python.bin 檔來執行，要注意這個 python 環境是否有安裝到 esptool 套件，不然也是沒辦法燒錄的。

4、結果：



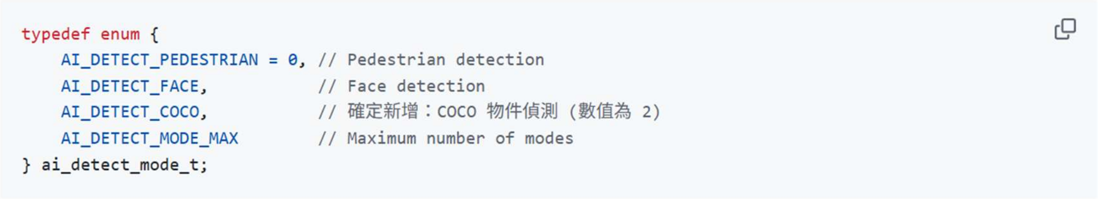
```
I (198) efuse_init: Chip rev:         v1.3
I (202) heap_init: Initializing. RAM available for dynamic allocation:
I (208) heap_init: At 4FF0F270 len 0002BD50 (175 KiB): RETENT_RAM
I (214) heap_init: At 4FF3AFC0 len 00004BF0 (18 KiB): RAM
I (219) heap_init: At 4FF40000 len 00060000 (384 KiB): RAM
I (225) heap_init: At 50108080 len 00007F80 (31 KiB): RTCRAM
I (230) heap_init: At 30100044 len 00001FBC (7 KiB): TCM
I (236) spi_flash: detected chip: generic
I (239) spi_flash: flash io: dio
W (242) spi_flash: Detected size(16384k) larger than the size in the binary imag
e header(2048k). Using the size in the binary image header.
I (254) sleep_gpio: Configure to isolate all GPIO pins in sleep state
I (260) sleep_gpio: Enable automatic switching of GPIO sleep configuration
I (267) main_task: Started on CPU0
I (277) main_task: Calling app_main()
Hello world!
This is esp32p4 chip with 2 CPU core(s), , silicon revision v1.3, 2MB external f
lash
Minimum free heap size: 617840 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```

COCO / YOLO / esp-dl 與模型部署

實作 COCO 偵測功能的 UI 觸發，必須完成以下四個確定步驟：

P.S.以下路徑為 github 專案說明寫的路徑，此次 clone 下來，在 jupyter 上實際的檔案絕對路徑為 esp-dev-kits/examples/esp32-p4-eye/examples/factory_demo/main/< token >

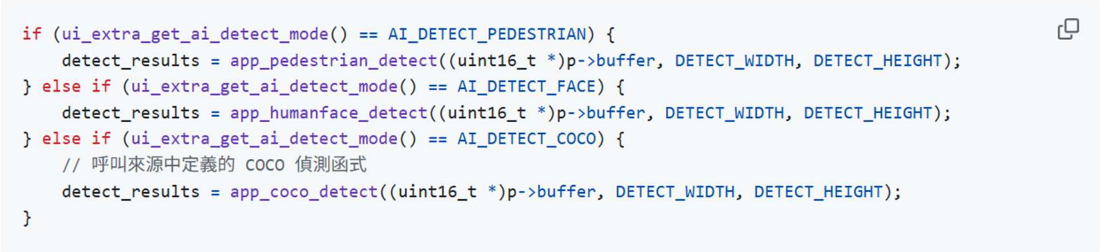
1、 修改模式定義 (./main/ui/ui_extra.h)，在現有的枚舉中新增 AI_DETECT_COCO。根據原始碼，PEDESTRIAN 為 0，FACE 為 1。



```
typedef enum {
    AI_DETECT_PEDESTRIAN = 0, // Pedestrian detection
    AI_DETECT_FACE,           // Face detection
    AI_DETECT_COCO,           // 確定新增：COCO 物件偵測 (數值為 2)
    AI_DETECT_MODE_MAX        // Maximum number of modes
} ai_detect_mode_t;
```

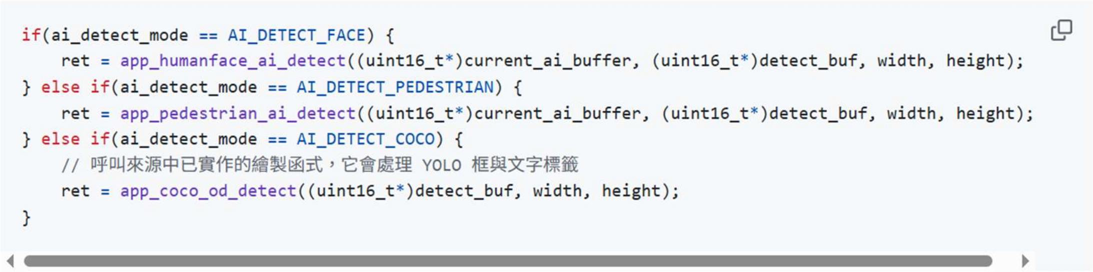
2、 更新 AI 處理邏輯 (./main/app/AI/app_ai_detect.cpp)，您必須在後端的任務與影格處理函式中加入 COCO 的路徑。

修改偵測任務：在 camera_dectect_task 函式中加入 COCO 偵測呼叫。



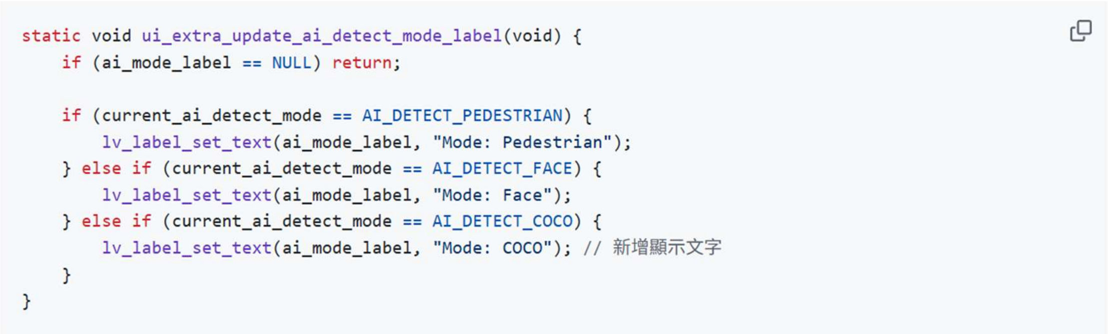
```
if (ui_extra_get_ai_detect_mode() == AI_DETECT_PEDESTRIAN) {
    detect_results = app_pedestrian_detect((uint16_t *)p->buffer, DETECT_WIDTH, DETECT_HEIGHT);
} else if (ui_extra_get_ai_detect_mode() == AI_DETECT_FACE) {
    detect_results = app_humanface_detect((uint16_t *)p->buffer, DETECT_WIDTH, DETECT_HEIGHT);
} else if (ui_extra_get_ai_detect_mode() == AI_DETECT_COCO) {
    // 呼叫來源中定義的 COCO 偵測函式
    detect_results = app_coco_detect((uint16_t *)p->buffer, DETECT_WIDTH, DETECT_HEIGHT);
}
```

修改影格處理：在 app_ai_detection_process_frame 中串聯繪圖邏輯。



```
if(ai_detect_mode == AI_DETECT_FACE) {
    ret = app_humanface_ai_detect((uint16_t*)current_ai_buffer, (uint16_t*)detect_buf, width, height);
} else if(ai_detect_mode == AI_DETECT_PEDESTRIAN) {
    ret = app_pedestrian_ai_detect((uint16_t*)current_ai_buffer, (uint16_t*)detect_buf, width, height);
} else if(ai_detect_mode == AI_DETECT_COCO) {
    // 呼叫來源中已實作的繪製函式，它會處理 YOLO 框與文字標籤
    ret = app_coco_od_detect((uint16_t*)detect_buf, width, height);
}
```

3、 更新 UI 標籤顯示 (./main/ui/ui_extra.c)，修改 ui_extra_update_ai_detect_mode_label 函式，讓 UI 能顯示「Mode: COCO」



```
static void ui_extra_update_ai_detect_mode_label(void) {
    if (ai_mode_label == NULL) return;

    if (current_ai_detect_mode == AI_DETECT_PEDESTRIAN) {
        lv_label_set_text(ai_mode_label, "Mode: Pedestrian");
    } else if (current_ai_detect_mode == AI_DETECT_FACE) {
        lv_label_set_text(ai_mode_label, "Mode: Face");
    } else if (current_ai_detect_mode == AI_DETECT_COCO) {
        lv_label_set_text(ai_mode_label, "Mode: COCO"); // 新增顯示文字
    }
}
```

4、 實作 UI 按鈕切換邏輯 (./main/ui/ui_extra.c)。目前的 UI 透過上下按鈕來切換模式。您需要修改 ui_extra_btn_up 與 ui_extra_btn_down 的 switch-case 邏輯，使其支援三個模式的循環切換。

向下按鈕 (Next Mode)：

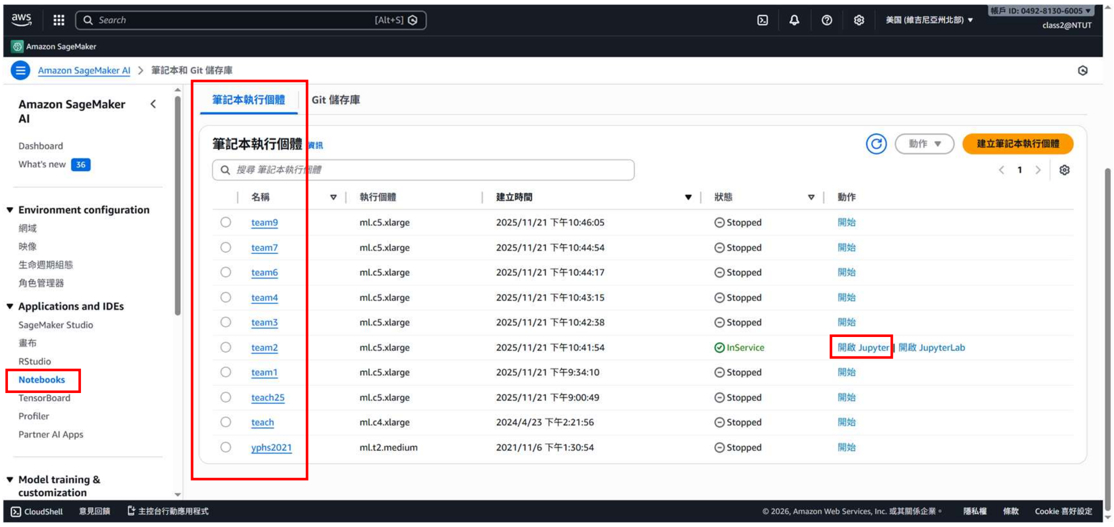
```
case UI_PAGE_AI_DETECT:
    if (current_ai_detect_mode == AI_DETECT_PEDESTRIAN) {
        ui_extra_change_ai_detect_mode(AI_DETECT_FACE);
    } else if (current_ai_detect_mode == AI_DETECT_FACE) {
        ui_extra_change_ai_detect_mode(AI_DETECT_COCO); // 切換至 COCO
    } else {
        ui_extra_change_ai_detect_mode(AI_DETECT_PEDESTRIAN); // 循環回第一個
    }
    break;
```

向上按鈕（Prev Mode）：

```
case UI_PAGE_AI_DETECT:
    if (current_ai_detect_mode == AI_DETECT_PEDESTRIAN) {
        ui_extra_change_ai_detect_mode(AI_DETECT_COCO); // 回到最後一個（COCO）
    } else if (current_ai_detect_mode == AI_DETECT_COCO) {
        ui_extra_change_ai_detect_mode(AI_DETECT_FACE);
    } else {
        ui_extra_change_ai_detect_mode(AI_DETECT_PEDESTRIAN);
    }
    break;
```

Amazon SageMaker AI 訓練實作

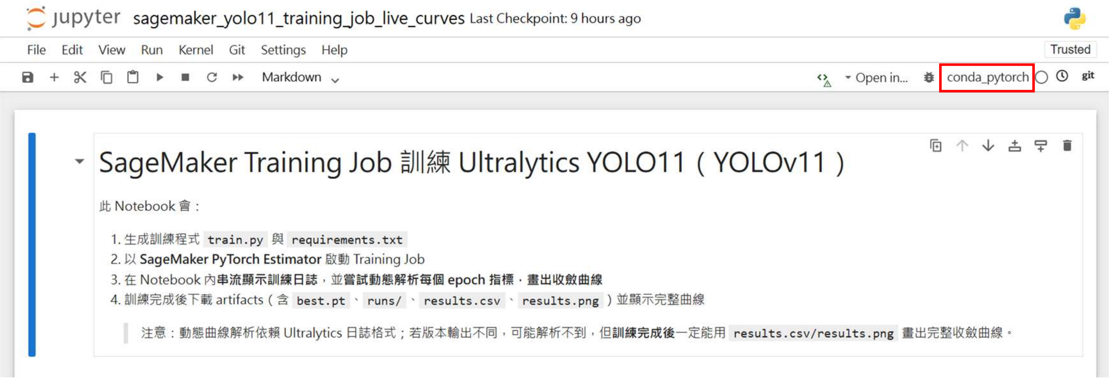
1、 登入 AWS console，search bar 輸入 Amazon SageMaker AI，左側選單點選 Applications and IDEs -> Notebooks，選取自己組別的個體，按「開始」，完全開啟後點選「開啟 Jupyter」 -> 2025_AIoT_3.4 -> sagemaker_yolo11_training_job_live_curves.ipynb。



筆記本執行個體

	名稱	執行個體	建立時間	狀態	動作
○	team9	ml.c5.xlarge	2025/11/21 下午10:46:05	Stopped	開始
○	team7	ml.c5.xlarge	2025/11/21 下午10:44:54	Stopped	開始
○	team6	ml.c5.xlarge	2025/11/21 下午10:44:17	Stopped	開始
○	team4	ml.c5.xlarge	2025/11/21 下午10:43:15	Stopped	開始
○	team3	ml.c5.xlarge	2025/11/21 下午10:42:38	Stopped	開始
○	team2	ml.c5.xlarge	2025/11/21 下午10:41:54	InService	開啟 Jupyter | 開啟 JupyterLab
○	team1	ml.c5.xlarge	2025/11/21 下午9:34:10	Stopped	開始
○	teach25	ml.c5.xlarge	2025/11/21 下午9:00:49	Stopped	開始
○	teach	ml.c4.xlarge	2024/4/23 下午2:21:56	Stopped	開始
○	yphs2021	ml.t2.medium	2021/11/6 下午1:30:54	Stopped	開始

2、 此次使用 pytorch 做訓練，確認 kernal 是否為「conda_pytorch」。


jupyter
sagemaker_yolo11_training_job_live_curves Last Checkpoint: 9 hours ago
File Edit View Run Kernel Git Settings Help
Trusted
Markdown
Open in...
conda_pytorch
git

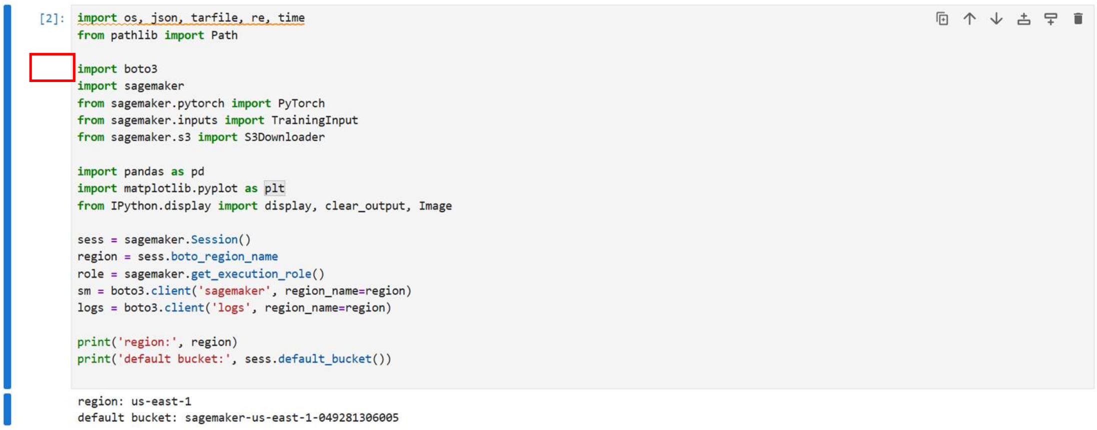
SageMaker Training Job 訓練 Ultralytics YOLO11（YOLOv11）

此 Notebook 會：

1. 生成訓練程式 `train.py` 與 `requirements.txt`
2. 以 **SageMaker PyTorch Estimator** 啟動 Training Job
3. 在 Notebook 內**串流顯示訓練日誌**，並**嘗試動態解析每個 epoch 指標，畫出收斂曲線**
4. 訓練完成後下載 artifacts（含 `best.pt`、`runs/`、`results.csv`、`results.png`）並顯示完整曲線

> 注意：動態曲線解析依賴 Ultralytics 日誌格式；若版本輸出不同，可能解析不到，但**訓練完成後**一定能用 `results.csv/results.png` 畫出完整收斂曲線。

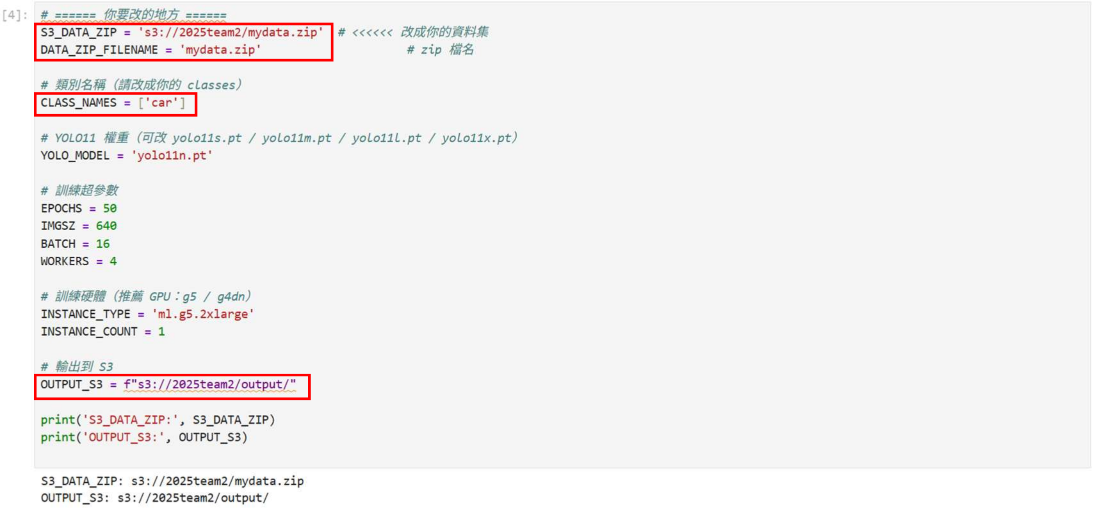
3、 確認無誤之後，按著程式碼框依序執行，點選方框後，shift + enter 就會開始執行。左上角[*]表示正在執行中，*號變成數字表示執行完畢，結果也會呈現於下方。



```
[2]: import os, json, tarfile, re, time
     from pathlib import Path

     import boto3
     import sagemaker
     from sagemaker.pytorch import PyTorch
     from sagemaker.inputs import TrainingInput
     from sagemaker.s3 import S3Downloader

     import pandas as pd
     import matplotlib.pyplot as plt
     from IPython.display import display, clear_output, Image

     sess = sagemaker.Session()
     region = sess.boto_region_name
     role = sagemaker.get_execution_role()
     sm = boto3.client('sagemaker', region_name=region)
     logs = boto3.client('logs', region_name=region)

     print('region:', region)
     print('default bucket:', sess.default_bucket())
```

```
region: us-east-1
default bucket: sagemaker-us-east-1-049281306005
```

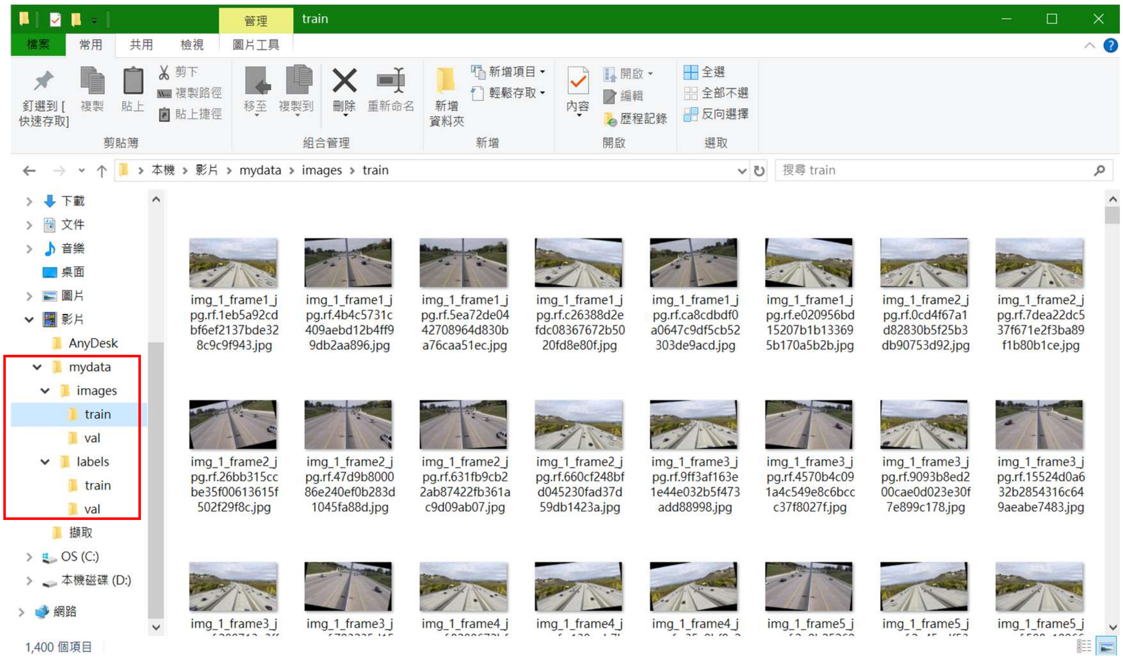
4、 進行部份參數修改，此次將資料集放在 aws S3，因此路徑名稱與檔名需要修改。資料 Zip 檔內部有建議結構，如下圖，請參照。



```
[4]: # ====== 你要改的地方 ======
     S3_DATA_ZIP = 's3://2025team2/mydata.zip'  # <<<<<< 改成你的資料集
     DATA_ZIP_FILENAME = 'mydata.zip'                     # zip 檔名

     # 類別名稱（請改成你的 classes）
     CLASS_NAMES = ['car']

     # YOLO11 權重（可改 yolo11s.pt / yolo11m.pt / yolo11l.pt / yolo11x.pt）
     YOLO_MODEL = 'yolo11n.pt'

     # 訓練超參數
     EPOCHS = 50
     IMGSZ = 640
     BATCH = 16
     WORKERS = 4

     # 訓練硬體（推薦 GPU：g5 / g4dn）
     INSTANCE_TYPE = 'ml.g5.2xlarge'
     INSTANCE_COUNT = 1

     # 輸出到 S3
     OUTPUT_S3 = f"s3://2025team2/output/"

     print('S3_DATA_ZIP:', S3_DATA_ZIP)
     print('OUTPUT_S3:', OUTPUT_S3)

     S3_DATA_ZIP: s3://2025team2/mydata.zip
     OUTPUT_S3: s3://2025team2/output/
```

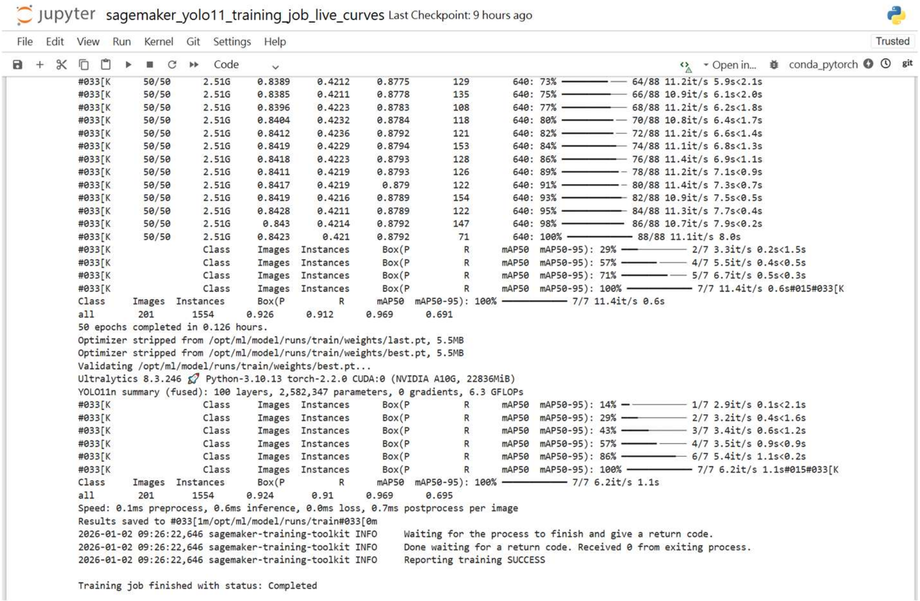
資料 zip 內部建議結構：

```
mydata/
  images/train
  images/val
  labels/train
  labels/val
```

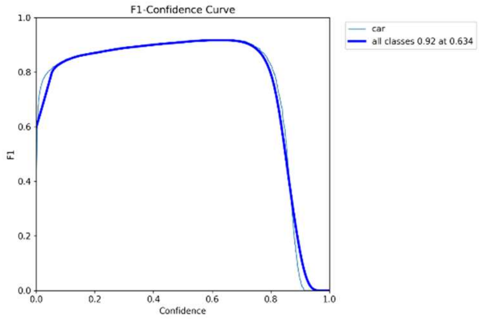


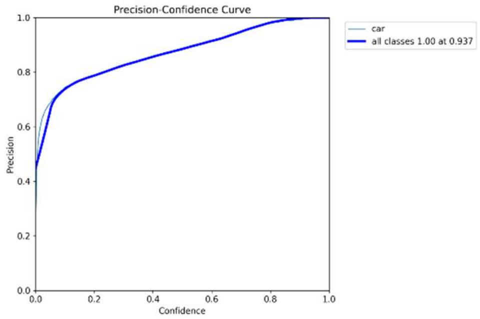
管理
train
檔案
常用
共用
檢視
圖片工具
釘選到 [快速存取]
複製
貼上
剪下
複製路徑
貼上捷徑
剪貼簿
移至
複製到
刪除
重新命名
組合管理
新增資料夾
新增項目
輕鬆存取
新增
內容
開啟
編輯
歷程記錄
開啟
全選
全部不選
反向選擇
選取
本機 › 影片 › mydata › images › train
搜尋 train
下載
文件
音樂
桌面
圖片
影片
AnyDesk
mydata
images
train
val
labels
train
val
擷取
OS (C:)
本機磁碟 (D:)
網路
img_1_frame1_jpg.rf.1eb5a92cdbf6ef2137bde328c9c9f943.jpg
img_1_frame1_jpg.rf.4b4c5731c409aebd12b4ff99db2aa896.jpg
img_1_frame1_jpg.rf.5ea72de0442708964d830ba76caa51ec.jpg
img_1_frame1_jpg.rf.c26388d2efdc08367672b5020fd8e80f.jpg
img_1_frame1_jpg.rf.ca8cdbdf0a0647c9df5cb52303de9acd.jpg
img_1_frame1_jpg.rf.e020956bd15207b1b133695b170a5b2b.jpg
img_1_frame2_jpg.rf.0cd4f67a1d82830b5f25b3db90753d92.jpg
img_1_frame2_jpg.rf.7dea22dc537f671e2f3ba89f1b80b1ce.jpg
img_1_frame2_jpg.rf.26bb315ccbe35f00613615f502f29f8c.jpg
img_1_frame2_jpg.rf.47d9b800086e240ef0b283d1045fa88d.jpg
img_1_frame2_jpg.rf.631fb9cb22ab87422fb361ac9d09ab07.jpg
img_1_frame2_jpg.rf.660cf248bfd045230fad37d59db1423a.jpg
img_1_frame3_jpg.rf.9ff3af163e1e44e032b5f473add88998.jpg
img_1_frame3_jpg.rf.4570b4c091a4c549e8c6bccc37f8027f.jpg
img_1_frame3_jpg.rf.9093b8ed200cae0d023e30f7e899c178.jpg
img_1_frame3_jpg.rf.15524d0a632b2854316c649aeabe7483.jpg
img_1_frame3_j
img_1_frame3_j
img_1_frame4_j
img_1_frame4_j
img_1_frame4_j
img_1_frame5_j
img_1_frame5_j
img_1_frame5_j
1,400 個項目

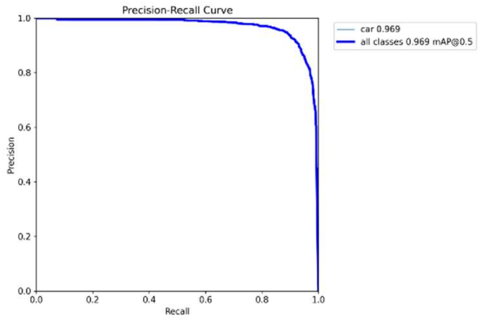
5、　　訓練完畢後結果，如下圖。

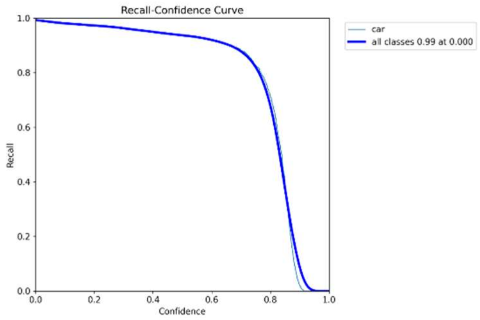


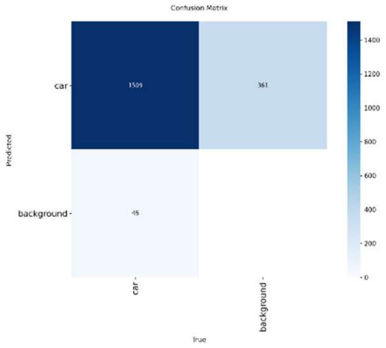
```
#033[K      50/50      2.51G     0.8389     0.4212     0.8775        129        640: 73% ━━━━━━━━━━━━━━───── 64/88 11.2it/s 5.9s<2.1s
#033[K      50/50      2.51G     0.8385     0.4211     0.8778        135        640: 75% ━━━━━━━━━━━━━━───── 66/88 10.9it/s 6.1s<2.0s
#033[K      50/50      2.51G     0.8396     0.4223     0.8783        108        640: 77% ━━━━━━━━━━━━━━───── 68/88 11.2it/s 6.2s<1.8s
#033[K      50/50      2.51G     0.8404     0.4232     0.8784        118        640: 80% ━━━━━━━━━━━━━━━──── 70/88 10.8it/s 6.4s<1.7s
#033[K      50/50      2.51G     0.8412     0.4236     0.8792        121        640: 82% ━━━━━━━━━━━━━━━──── 72/88 11.2it/s 6.6s<1.4s
#033[K      50/50      2.51G     0.8419     0.4229     0.8794        153        640: 84% ━━━━━━━━━━━━━━━━─── 74/88 11.1it/s 6.8s<1.3s
#033[K      50/50      2.51G     0.8418     0.4223     0.8793        128        640: 86% ━━━━━━━━━━━━━━━━─── 76/88 11.4it/s 6.9s<1.1s
#033[K      50/50      2.51G     0.8411     0.4219     0.8793        126        640: 89% ━━━━━━━━━━━━━━━━━── 78/88 11.2it/s 7.1s<0.9s
#033[K      50/50      2.51G     0.8417     0.4219      0.879        122        640: 91% ━━━━━━━━━━━━━━━━━── 80/88 11.4it/s 7.3s<0.7s
#033[K      50/50      2.51G     0.8419     0.4216     0.8789        154        640: 93% ━━━━━━━━━━━━━━━━━━─ 82/88 10.9it/s 7.5s<0.5s
#033[K      50/50      2.51G     0.8428     0.4211     0.8789        122        640: 95% ━━━━━━━━━━━━━━━━━━─ 84/88 11.3it/s 7.7s<0.4s
#033[K      50/50      2.51G      0.843     0.4214     0.8792        147        640: 98% ━━━━━━━━━━━━━━━━━━━ 86/88 10.7it/s 7.9s<0.2s
#033[K      50/50      2.51G     0.8423      0.421     0.8792         71        640: 100% ━━━━━━━━━━━━━━━━━━━━ 88/88 11.1it/s 8.0s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 29% ━━━━─────────────── 2/7 3.3it/s 0.2s<1.5s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 57% ━━━━━━━━━━────────── 4/7 5.5it/s 0.4s<0.5s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 71% ━━━━━━━━━━━━━─────── 5/7 6.7it/s 0.5s<0.3s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━━━━━━━━━ 7/7 11.4it/s 0.6s#015#033[K
Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━━━━━━━━━ 7/7 11.4it/s 0.6s
all        201       1554      0.926      0.912      0.969      0.691
50 epochs completed in 0.126 hours.
Optimizer stripped from /opt/ml/model/runs/train/weights/last.pt, 5.5MB
Optimizer stripped from /opt/ml/model/runs/train/weights/best.pt, 5.5MB
Validating /opt/ml/model/runs/train/weights/best.pt...
Ultralytics 8.3.246 🚀 Python-3.10.13 torch-2.2.0 CUDA:0 (NVIDIA A10G, 22836MiB)
YOLO11n summary (fused): 100 layers, 2,582,347 parameters, 0 gradients, 6.3 GFLOPs
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 14% ━━───────────────── 1/7 2.9it/s 0.1s<2.1s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 29% ━━━━─────────────── 2/7 3.2it/s 0.4s<1.6s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 43% ━━━━━━━──────────── 3/7 3.4it/s 0.6s<1.2s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 57% ━━━━━━━━━━────────── 4/7 3.5it/s 0.9s<0.9s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 86% ━━━━━━━━━━━━━━━━━─── 6/7 5.4it/s 1.1s<0.2s
#033[K                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━━━━━━━━━ 7/7 6.2it/s 1.1s#015#033[K
Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━━━━━━━━━ 7/7 6.2it/s 1.1s
all        201       1554      0.924       0.91      0.969      0.695
Speed: 0.1ms preprocess, 0.6ms inference, 0.0ms loss, 0.7ms postprocess per image
Results saved to #033[1m/opt/ml/model/runs/train#033[0m
2026-01-02 09:26:22,646 sagemaker-training-toolkit INFO     Waiting for the process to finish and give a return code.
2026-01-02 09:26:22,646 sagemaker-training-toolkit INFO     Done waiting for a return code. Received 0 from exiting process.
2026-01-02 09:26:22,646 sagemaker-training-toolkit INFO     Reporting training SUCCESS

Training job finished with status: Completed
```

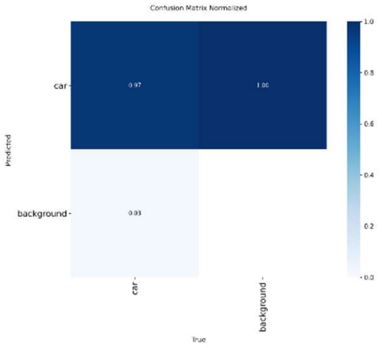


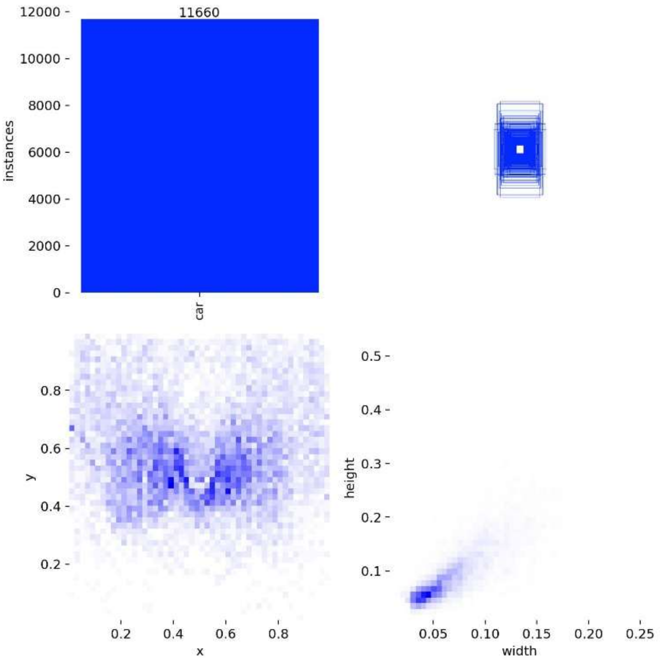


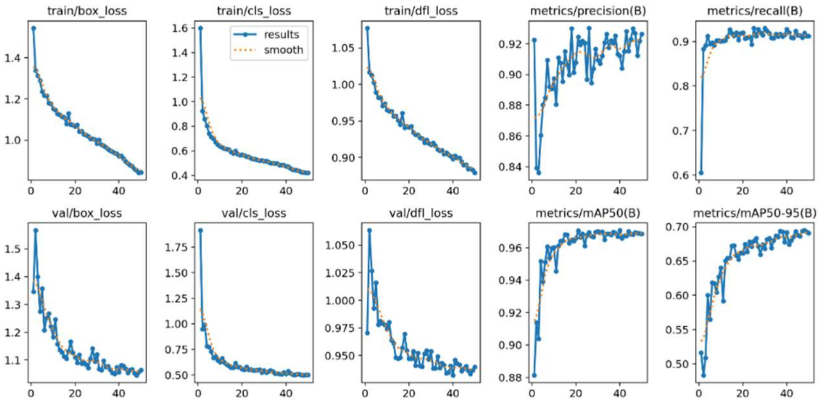


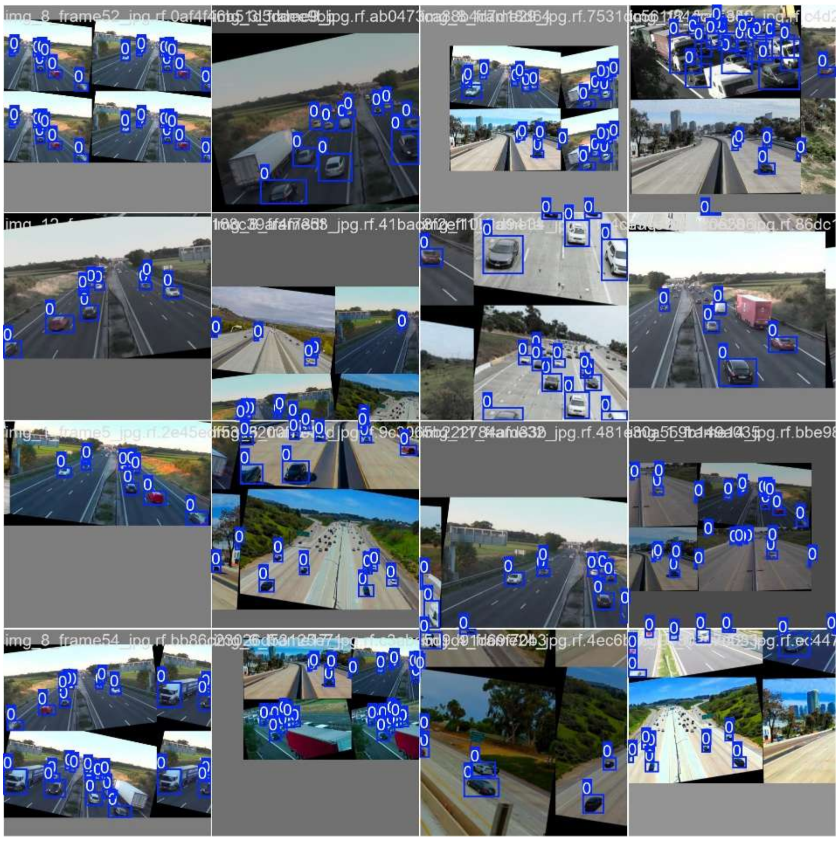


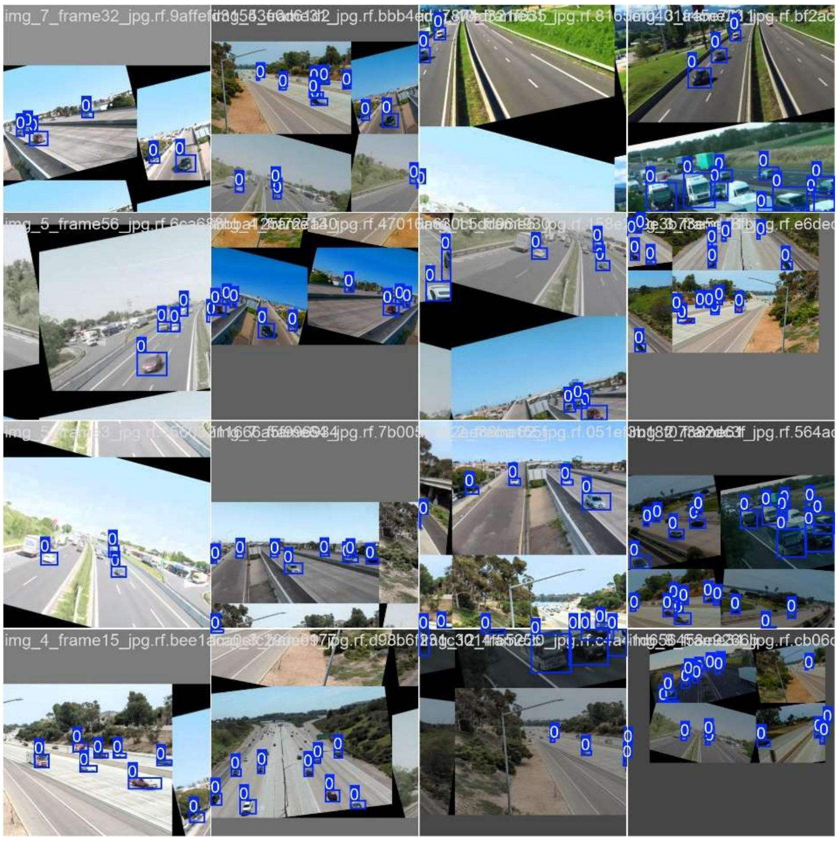


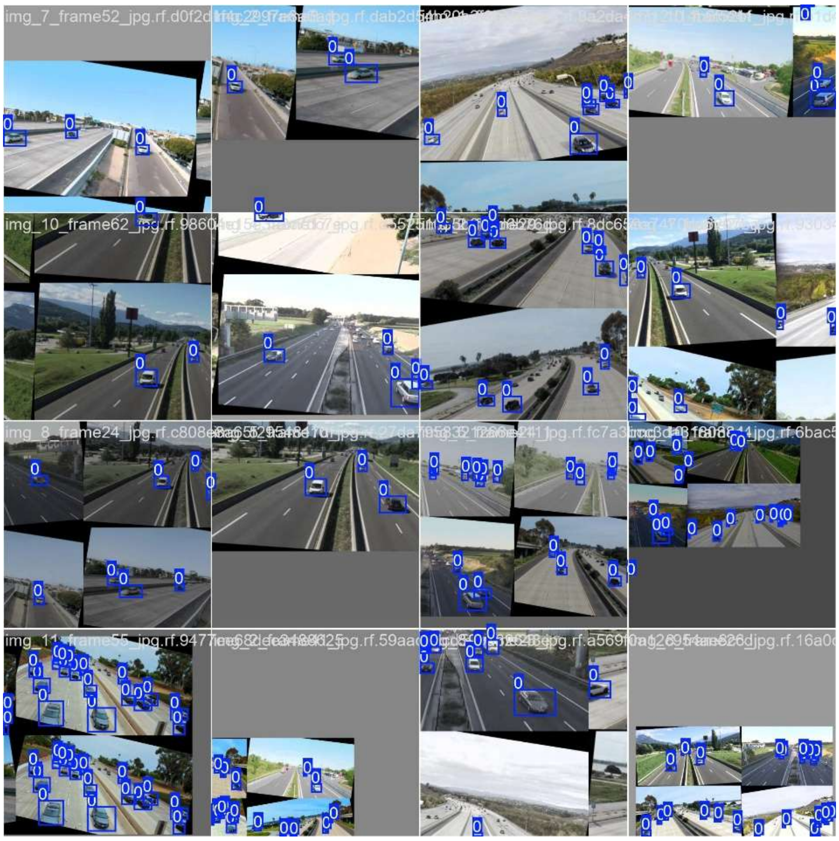


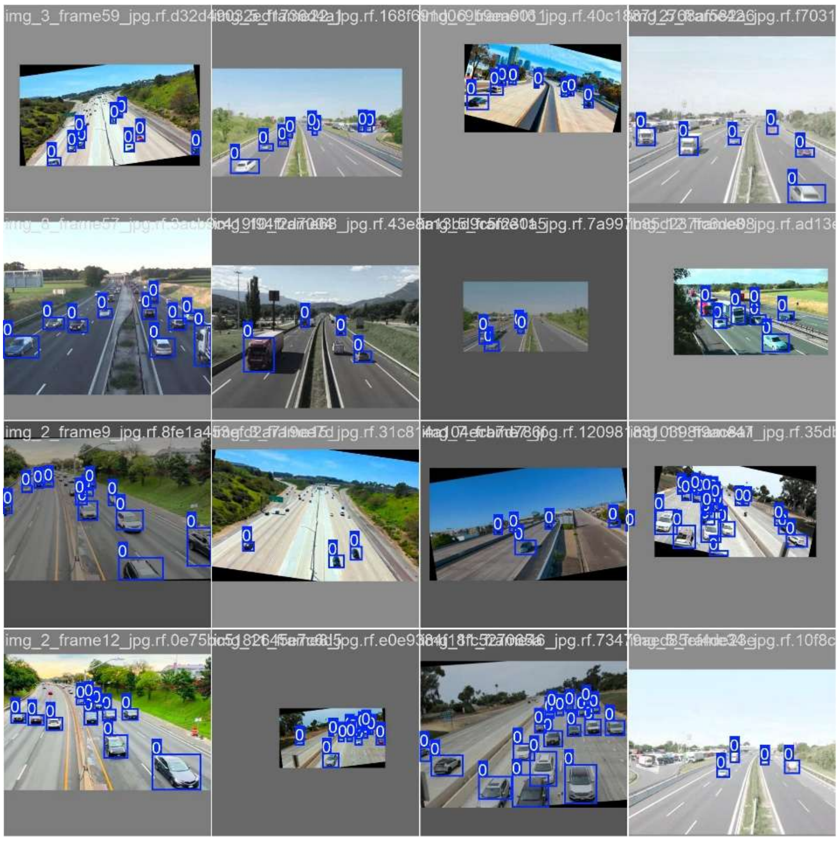

11660
12000
10000
8000
6000
4000
2000
0
instances
car
0.8
0.6
0.4
0.2
y
0.2
0.4
0.6
0.8
x
0.5
0.4
0.3
0.2
0.1
height
0.05
0.10
0.15
0.20
0.25
width

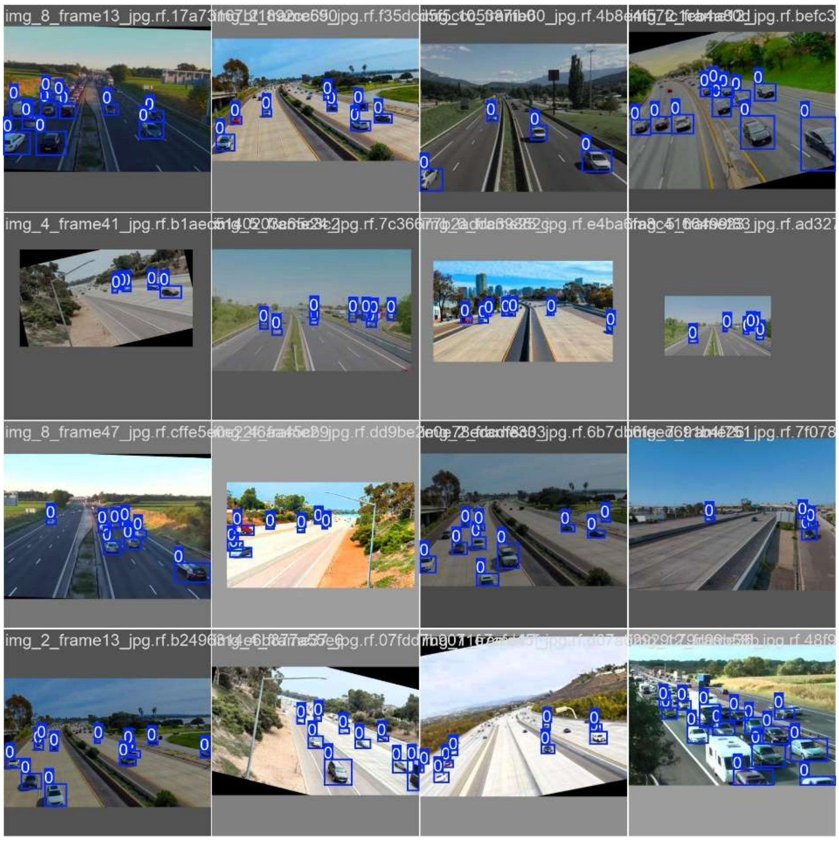

train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)
0
20
40

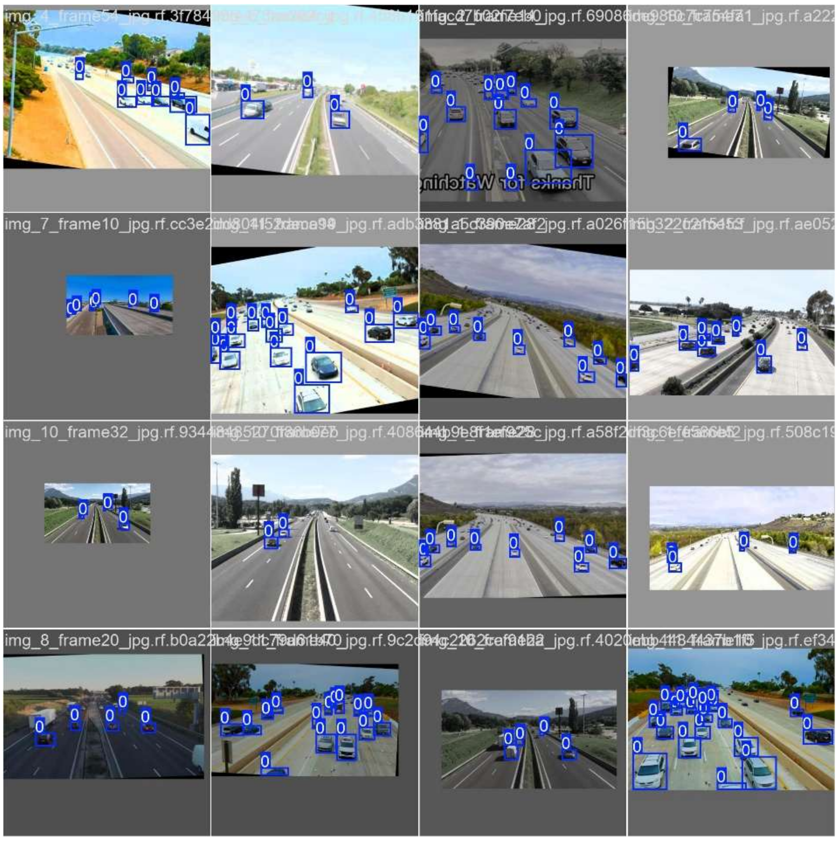

img_8_frame52_jpg.rf.0af4f
img_8_frame54_jpg.rf.bb86
0

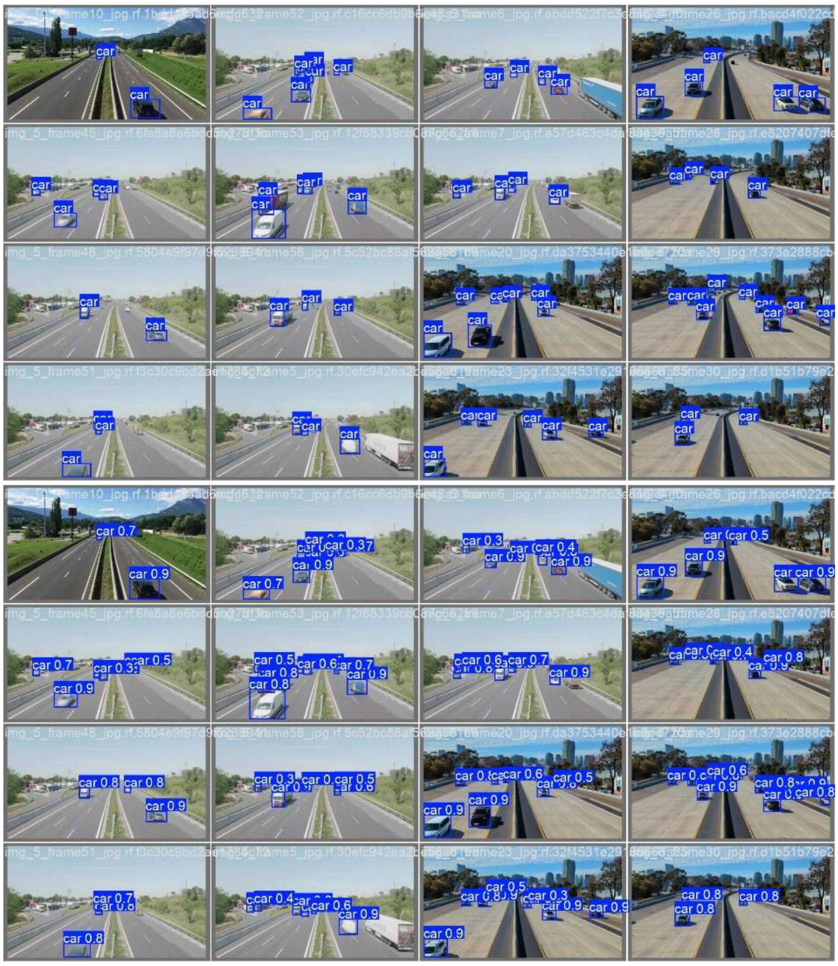

img_7_frame32_jpg.rf.9affe
img_5_frame56_jpg.rf.6ca
img_4_frame15_jpg.rf.bee1
jpg.rf.bbb4
jpg.rf.bf2ac
jpg.rf.e6dec
jpg.rf.564ac
jpg.rf.cb06c

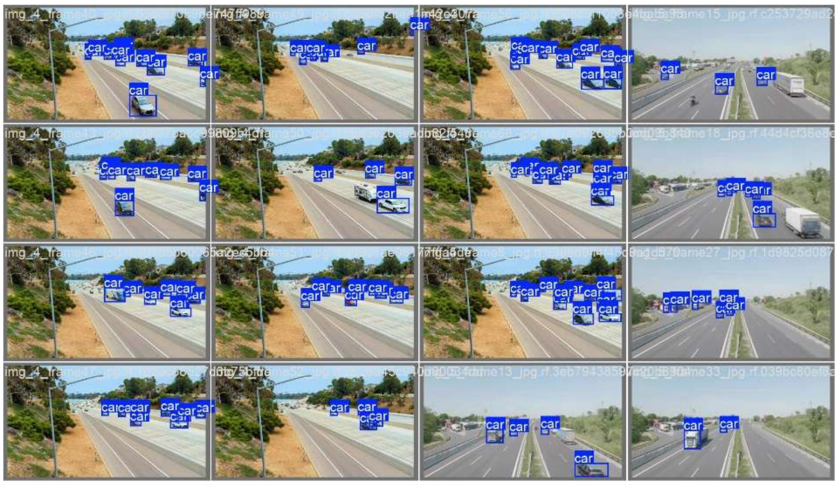

img_7_frame52_jpg.rf.d0f2d
img_10_frame62_jpg.rf.9860
img_8_frame24_jpg.rf.c808e
img_11_frame55_jpg.rf.9477
jpg.rf.dab2d
jpg.rf.8dc6
jpg.rf.fc7a
jpg.rf.6bac
jpg.rf.59aa
jpg.rf.a569
jpg.rf.16a0


img_3_frame59_jpg.rf.d32d4
img_2_frame9_jpg.rf.8fe1a4
img_2_frame12_jpg.rf.0e75
0

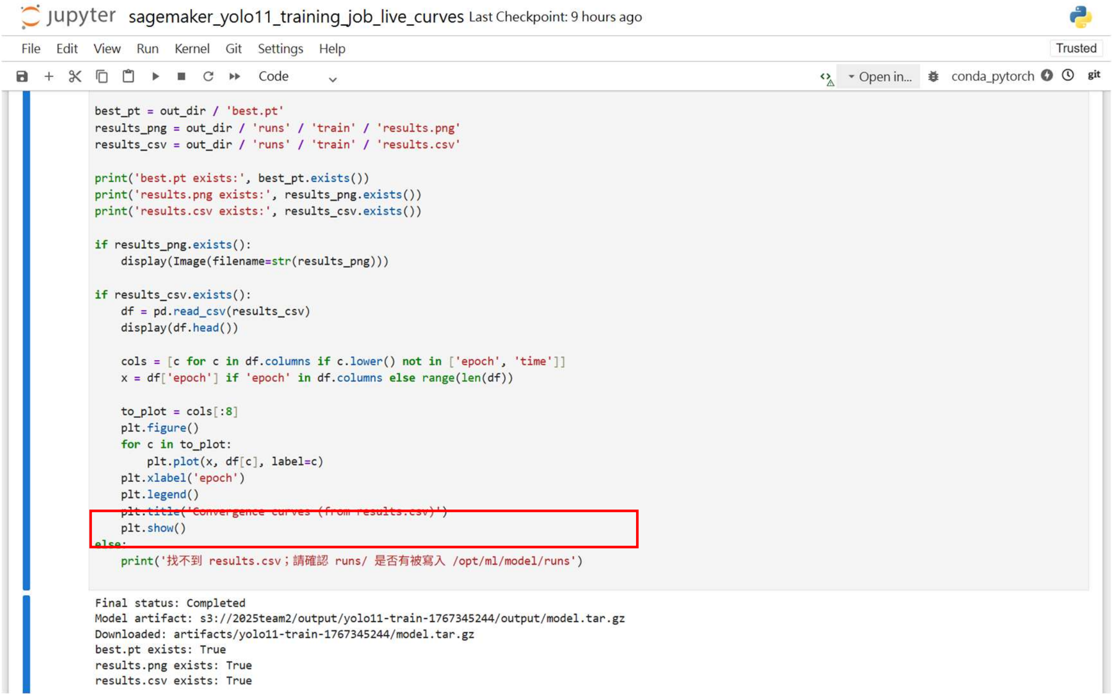

img_8_frame13_jpg.rf.17a7
img_4_frame41_jpg.rf.b1ae
img_8_frame47_jpg.rf.cffe5
img_2_frame13_jpg.rf.b249
0


img_7_frame10_jpg.rf.cc3e
img_10_frame32_jpg.rf.934
img_8_frame20_jpg.rf.b0a2
0


img_5_frame45_jpg.rf.6fe8a8e6
img_5_frame48_jpg.rf.5804e9
img_5_frame51_jpg.rf.f3c30c9bd2
car
car 0.9
car 0.8
car 0.7



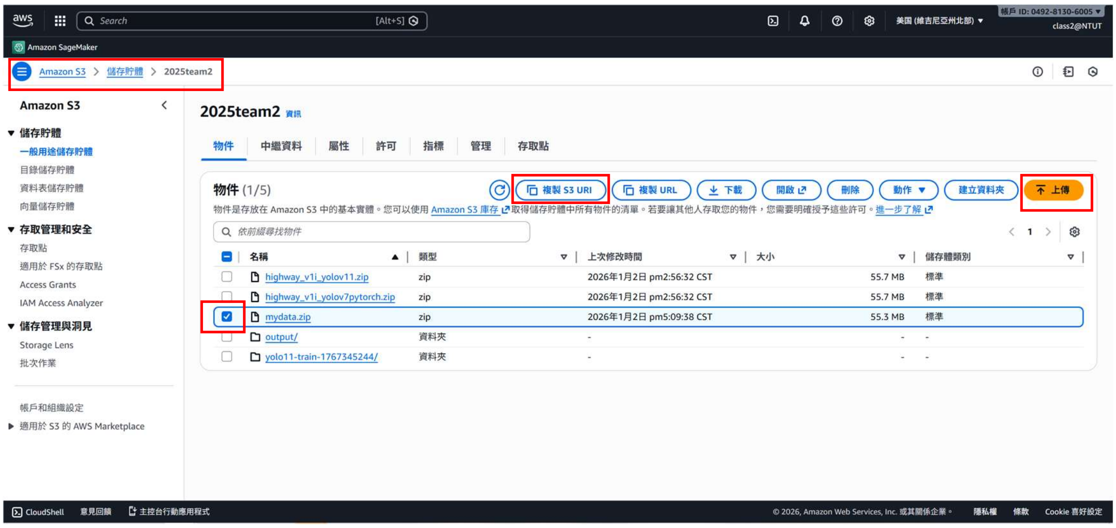
6、 結果儲存需執行「5) 下載 artifacts 並畫出完整收斂曲線 (results.csv / results.png)」這個段落的程式碼，結果與儲存路徑，如下圖。


jupyter sagemaker_yolo11_training_job_live_curves Last Checkpoint: 9 hours ago
File Edit View Run Kernel Git Settings Help
Trusted
Code
Open in...
conda_pytorch
git

```python
best_pt = out_dir / 'best.pt'
results_png = out_dir / 'runs' / 'train' / 'results.png'
results_csv = out_dir / 'runs' / 'train' / 'results.csv'

print('best.pt exists:', best_pt.exists())
print('results.png exists:', results_png.exists())
print('results.csv exists:', results_csv.exists())

if results_png.exists():
    display(Image(filename=str(results_png)))

if results_csv.exists():
    df = pd.read_csv(results_csv)
    display(df.head())

    cols = [c for c in df.columns if c.lower() not in ['epoch', 'time']]
    x = df['epoch'] if 'epoch' in df.columns else range(len(df))

    to_plot = cols[:8]
    plt.figure()
    for c in to_plot:
        plt.plot(x, df[c], label=c)
    plt.xlabel('epoch')
    plt.legend()
    plt.title('Convergence curves (from results.csv)')
    plt.show()
else:
    print('找不到 results.csv；請確認 runs/ 是否有被寫入 /opt/ml/model/runs')
```

```
Final status: Completed
Model artifact: s3://2025team2/output/yolo11-train-1767345244/output/model.tar.gz
Downloaded: artifacts/yolo11-train-1767345244/model.tar.gz
best.pt exists: True
results.png exists: True
results.csv exists: True
```

7、 補充 aws S3 操作：search bar 輸入 s3，選取自己組別儲存體後，點選右上角「上傳」。上傳完畢之後，點選想要使用的資料集，接著按上方的「複製 S3 URI」，貼到先前參數路徑上。詳見下圖：



物件是存放在 Amazon S3 中的基本實體。您可以使用 Amazon S3 庫存 取得儲存貯體中所有物件的清單。若要讓其他人存取您的物件，您需要明確授予這些許可。進一步了解

	名稱	類型	上次修改時間	大小	儲存體類別
□	highway_v1i_yolov11.zip	zip	2026年1月2日 pm2:56:32 CST	55.7 MB	標準
□	highway_v1i_yolov7pytorch.zip	zip	2026年1月2日 pm2:56:32 CST	55.7 MB	標準
☑	mydata.zip	zip	2026年1月2日 pm5:09:38 CST	55.3 MB	標準
□	output/	資料夾	-	-	-
□	yolo11-train-1767345244/	資料夾	-	-	-