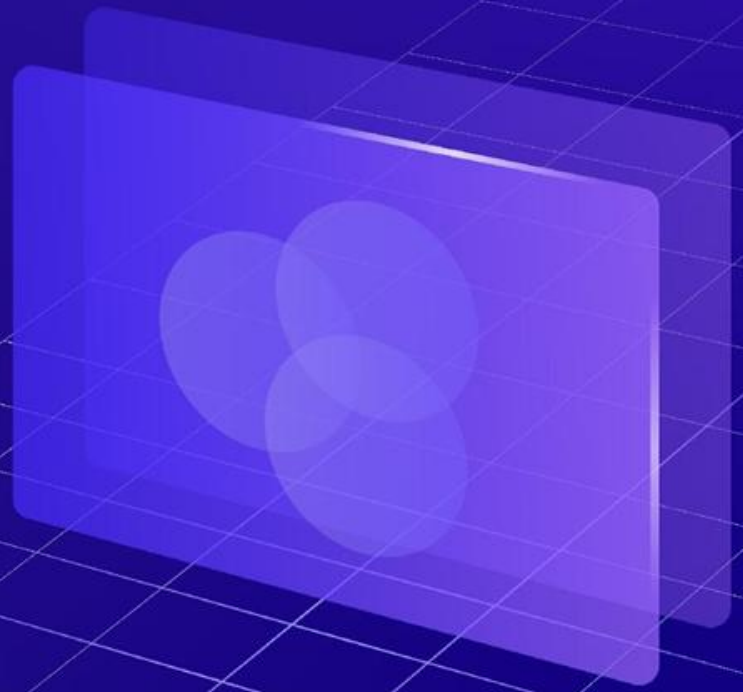
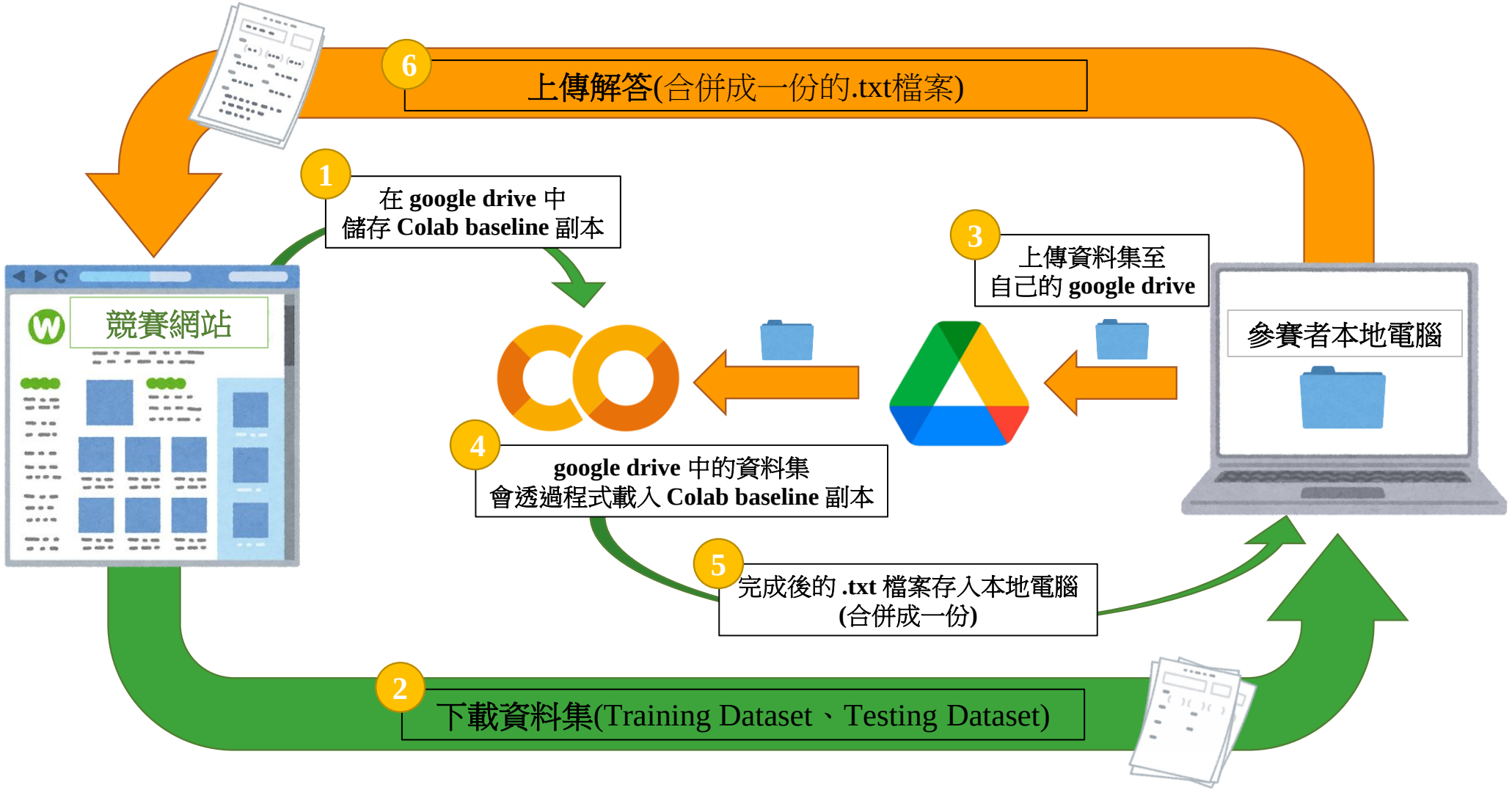
AI CUP 2025 秋季賽

電腦斷層心臟肌肉影像分割競賽 II
主動脈瓣物件偵測



目錄

使用 Baseline 程式參加競賽流程圖



01

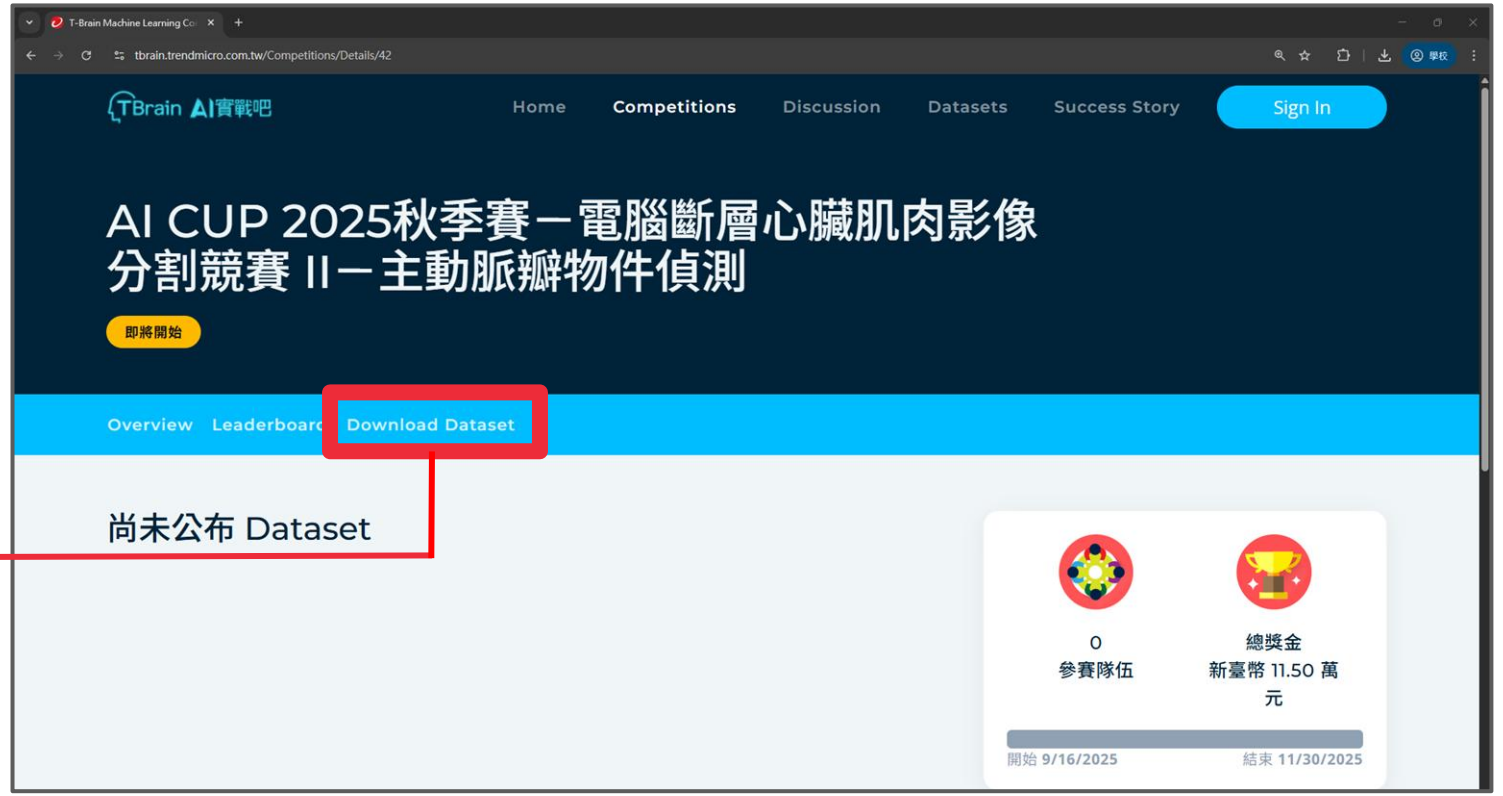
資料集說明

下載競賽資料集

新興智慧顯示科技教育聯盟

- 報名完成後可至趨勢科技 Tbrain 競賽網站下載資料集。
- 下載網址 : https://tbrain.trendmicro.com.tw/Competitions/Details/42 。

點選可以看到資料集



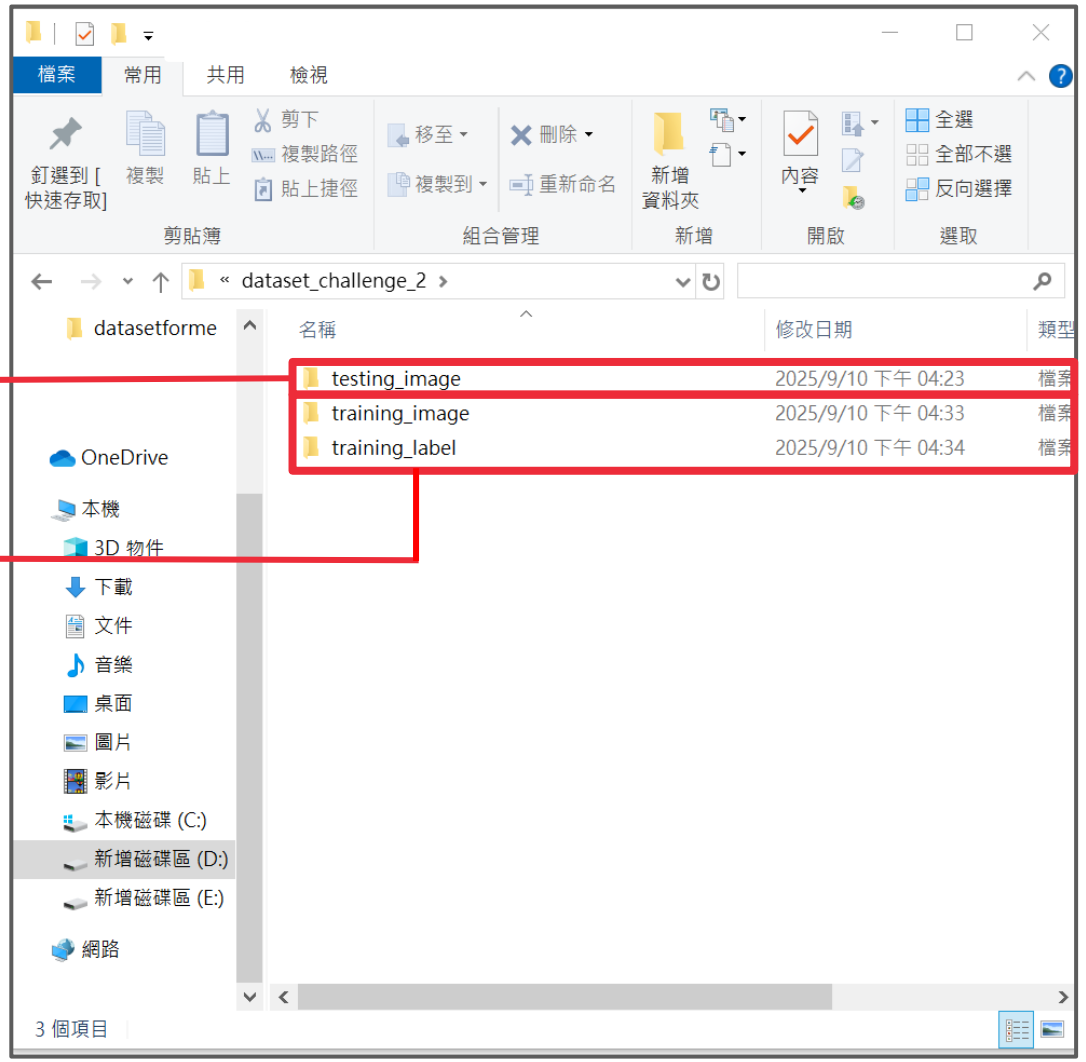
訓練集 & 測試集

- **資料集**包含兩部分：
  - Training Dataset
  - Testing Dataset

本次競賽試題！

Testing Dataset

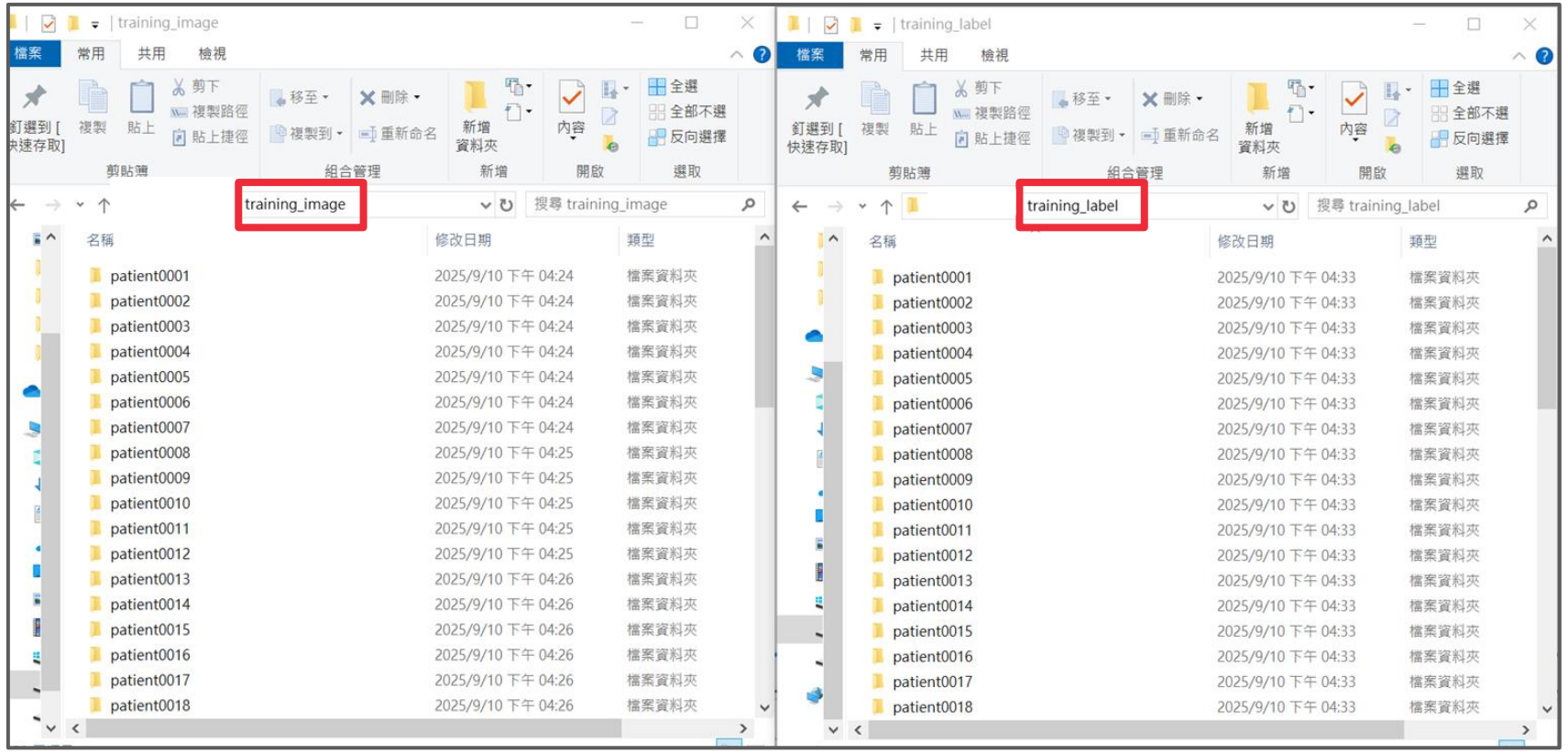
Training Dataset



▶ 資料集

訓練集組成(50筆電腦斷層掃描)

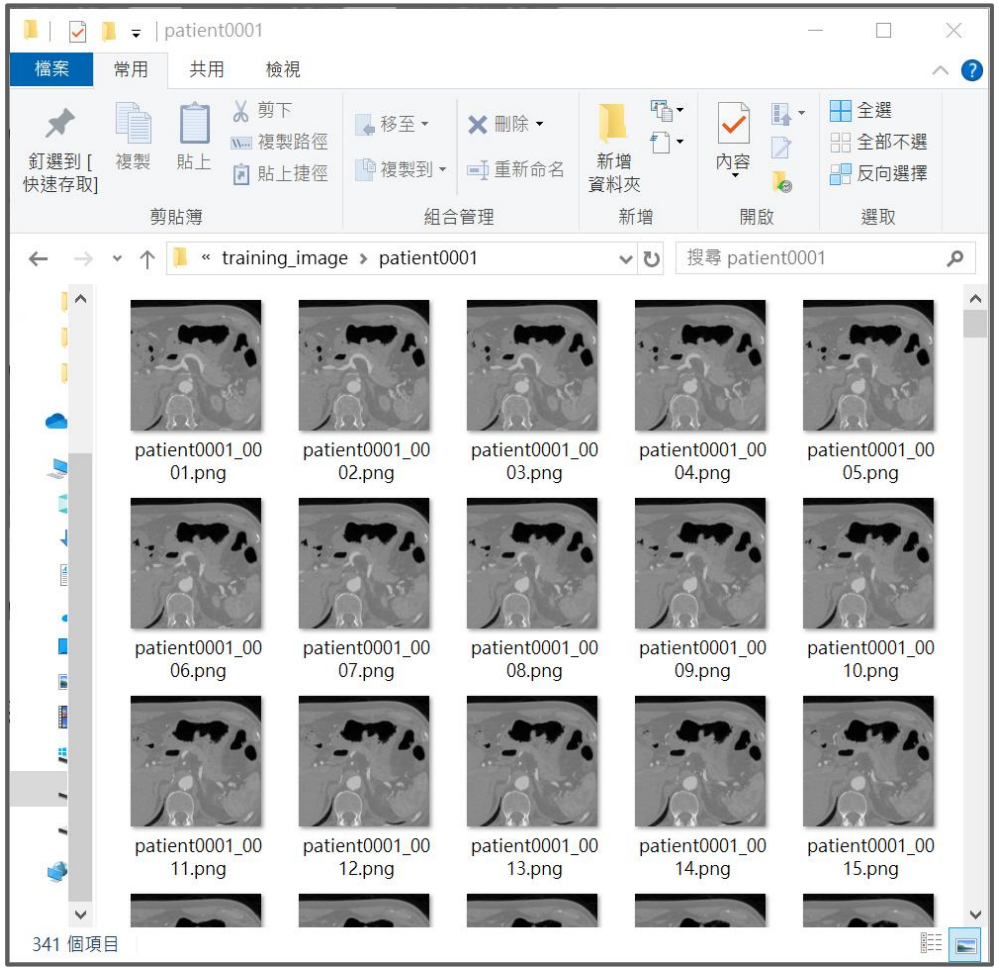
- Training Dataset 分成 training_image 和 training_label 兩個**檔案夾**。
- 各自內含50個**檔案夾**(patient0001 ~ patient0050)。



▶ training_image 和 training_label 檔案夾

training_image 為電腦斷層掃描

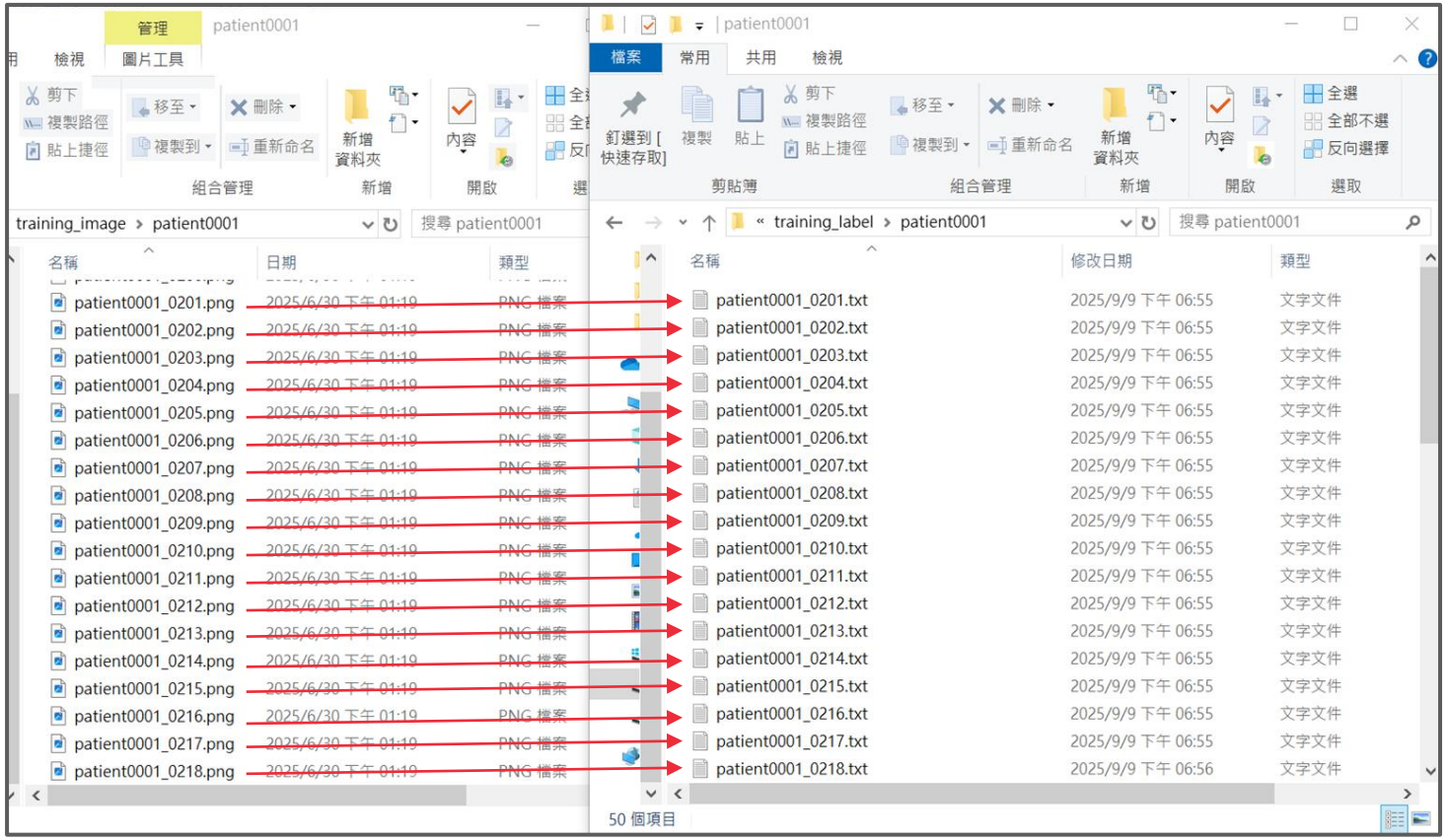
- training_image 檔案夾中
  - 每個 **patient 檔案夾**包含有 250~450 張不等之 PNG 圖檔。
  - 每張 PNG 圖檔尺寸為 512 * 512 pixel。



▶ training_image 檔案夾中包含 PNG 圖檔

training_label 為標註

- training_label 中每個 patient 檔案夾中，大概包含有 30~80 個對應 training_image 的標註檔(副檔名為 .txt)。
- 例如 patient0001_0201.png 對應的標註檔為 patient0001_0201.txt 。
- 若某張 PNG 圖檔沒有對應的標註檔，則表示該圖中沒有主動脈瓣，故無標註檔。

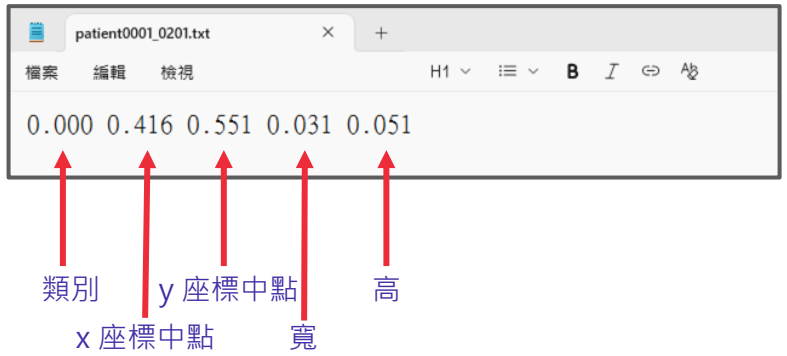


▲ training_label 檔案夾中包含 .txt 標註檔

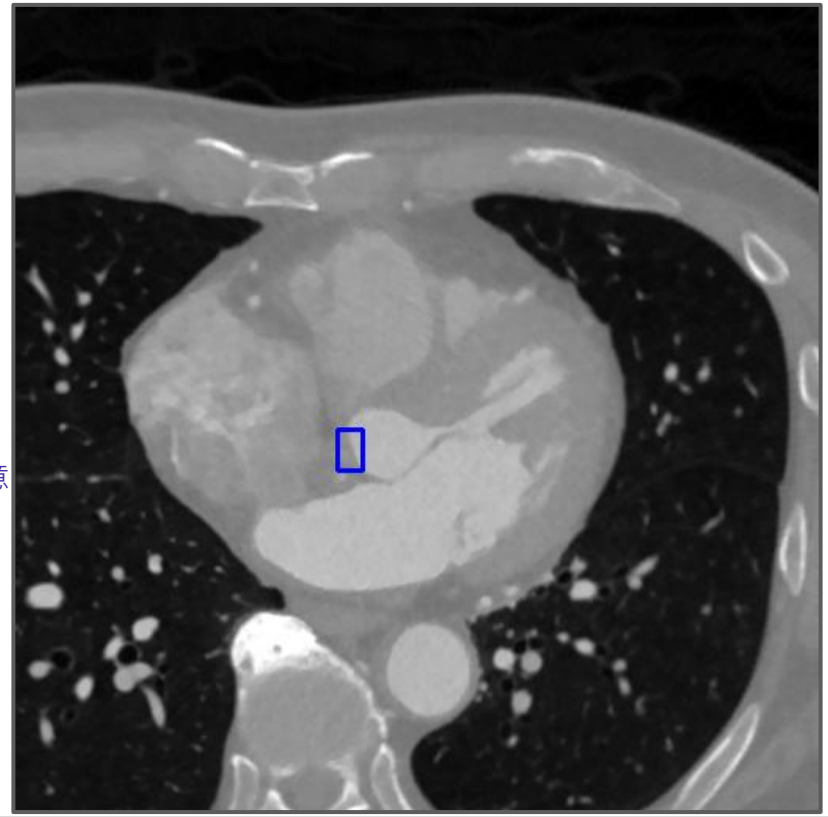
標註檔 .txt 中的各項數值(正規化座標)

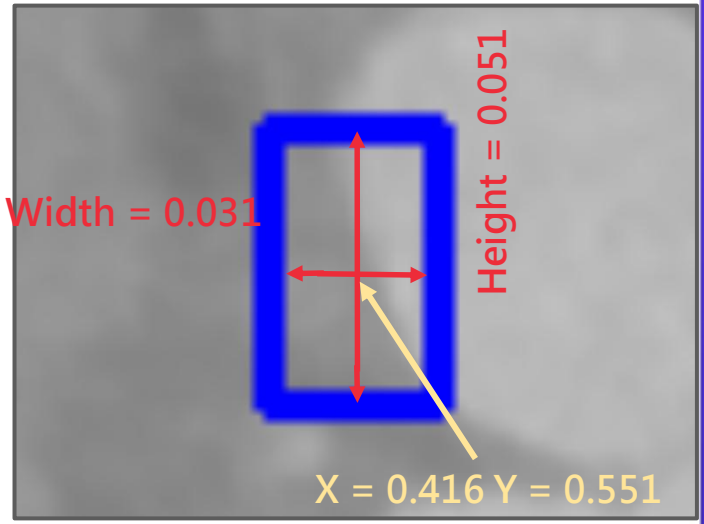
- 標註檔 .txt 採用 YOLO 模型訓練格式，檔案內的各數值依序分別為

類別-空格-**正規化後的預測框 x 座標中點**-空格-**正規化後的預測框 y 座標中點**-空格-**正規化後的預測框寬度**-空格-**正規化後的預測框高度**



training_label 檔案夾中 .txt 標註檔內容以及其含意





標註資料產出過程

- 使用 LabelMe 進行標註後會產生 .json 檔。
- 因為 YOLO 訓練需要正規化後的座標，所以使用 python程式將原始座標轉為正規化座標。
- .json 檔 透過 python 程式正規化後會產生 training_label 檔案夾中的標註檔。

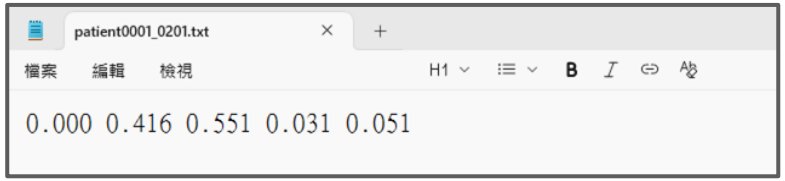
▶使用 LabelMe 進行標註

```
{
  "version": "5.4.0",
  "flags": {},
  "shapes": [
    {
      "label": "ac",
      "points": [
        [
          205.21465968586386,
          269.35078534031413
        ],
        [
          221.18324607329842,
          295.0052356020943
        ]
      ],
      "group_id": null,
      "description": "",
      "shape_type": "rectangle",
      "flags": {},
      "mask": null
    }
  ],
  "imagePath": "IMG0201.png",
  "imageData": "iVBORw0KGgoAAAANSUhEUgAAAgA
  "imageHeight": 512,
  "imageWidth": 512
}
```

▲LabelMe 產生的 .json檔

```
# 轉成 YOLO 格式
label = label_dict[label_name]
xc = ((pt1[0]+pt2[0])/2) / x_len
yc = ((pt1[1]+pt2[1])/2) / y_len
w = abs(pt2[0]-pt1[0]) / x_len
h = abs(pt2[1]-pt1[1]) / y_len
to_txt.append([label, xc, yc, w, h])
```

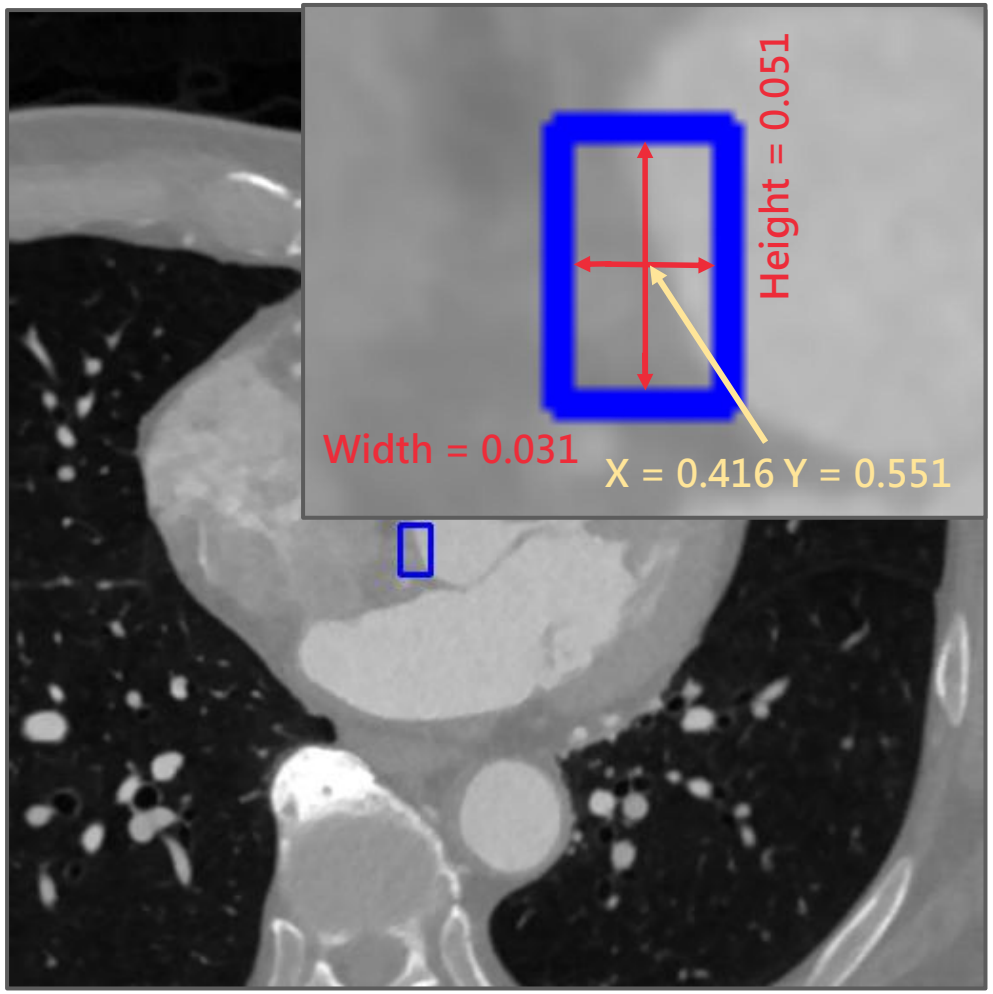
▲透過 python 程式轉換 YOLO 格式



▲training_label中的標註檔

計算正規化座標方法 (從原始座標)

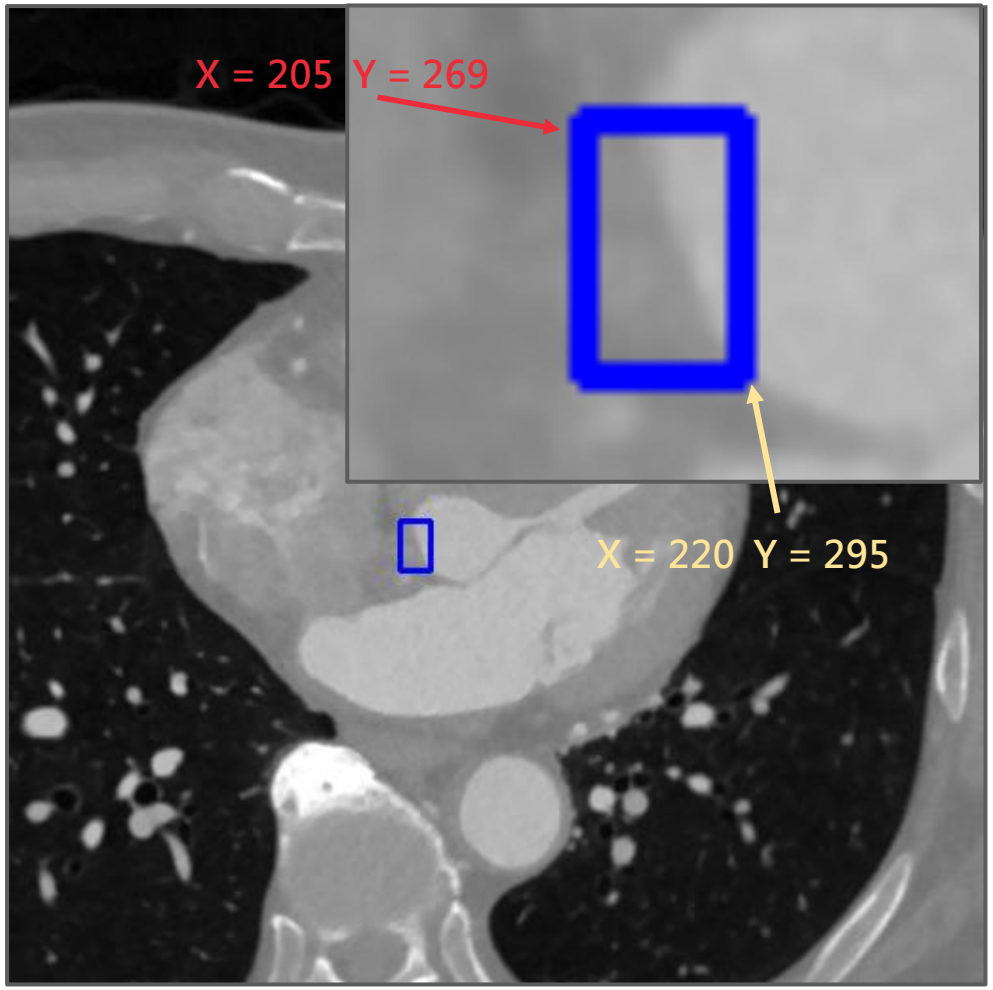
- 預測框正規化數值可以透過以下公式計算。
  xc = ((x_min + x_max) / 2) / width
  yc = ((y_min + y_max) / 2) / height
  w = (x_max - x_min) / width
  h = (y_max - y_min) / height
  xc : 正規化後的預測框x座標中點 w : 正規化後的預測框寬度
  yc : 正規化後的預測框y座標中點 h : 正規化後的預測框高度
  width : 圖片寬度 (本競賽使用的圖片寬度皆為 512)
  height : 圖片高度 (本競賽使用的圖片高度皆為 512)
- Example:
  若 x_min = 205 y_min = 269 x_max = 221 y_max = 295，則原始座標為
  xc = ((205 + 221) / 2) / 512 = 0.416
  yc = ((269 + 295) / 2) / 512 = 0.551
  w = (221 - 205) / 512 = 0.031
  h = (295- 269) / 512 = 0.051



▲ 正規化後的點座標

還原原始座標方法 (從YOLO格式)

- 預測框原始座標可以透過以下公式計算。
  x_min = round ((xc - w / 2) * width)
  x_max = round ((xc + w / 2) * width)
  y_min = round ((yc - h / 2) * height)
  y_max = round ((yc + h / 2) * height)
  xc : 正規化後的預測框x座標中點 w : 正規化後的預測框寬度
  yc : 正規化後的預測框y座標中點 h : 正規化後的預測框高度
  width : 圖片寬度 (本競賽使用的圖片寬度皆為 512)
  height : 圖片高度 (本競賽使用的圖片高度皆為 512)
- Example:
  若 .txt 的後四個數值為 0.416 0.551 0.031 0.051，
  則原始座標為
  x_min = round ((0.416 – 0.031 / 2) * 512) = 205
  x_max = round ((0.416 + 0.031 / 2) * 512) = 221
  y_min = round ((0.551 – 0.051 / 2) * 512) = 269
  y_max = round ((0.551 + 0.051 / 2) * 512) = 295

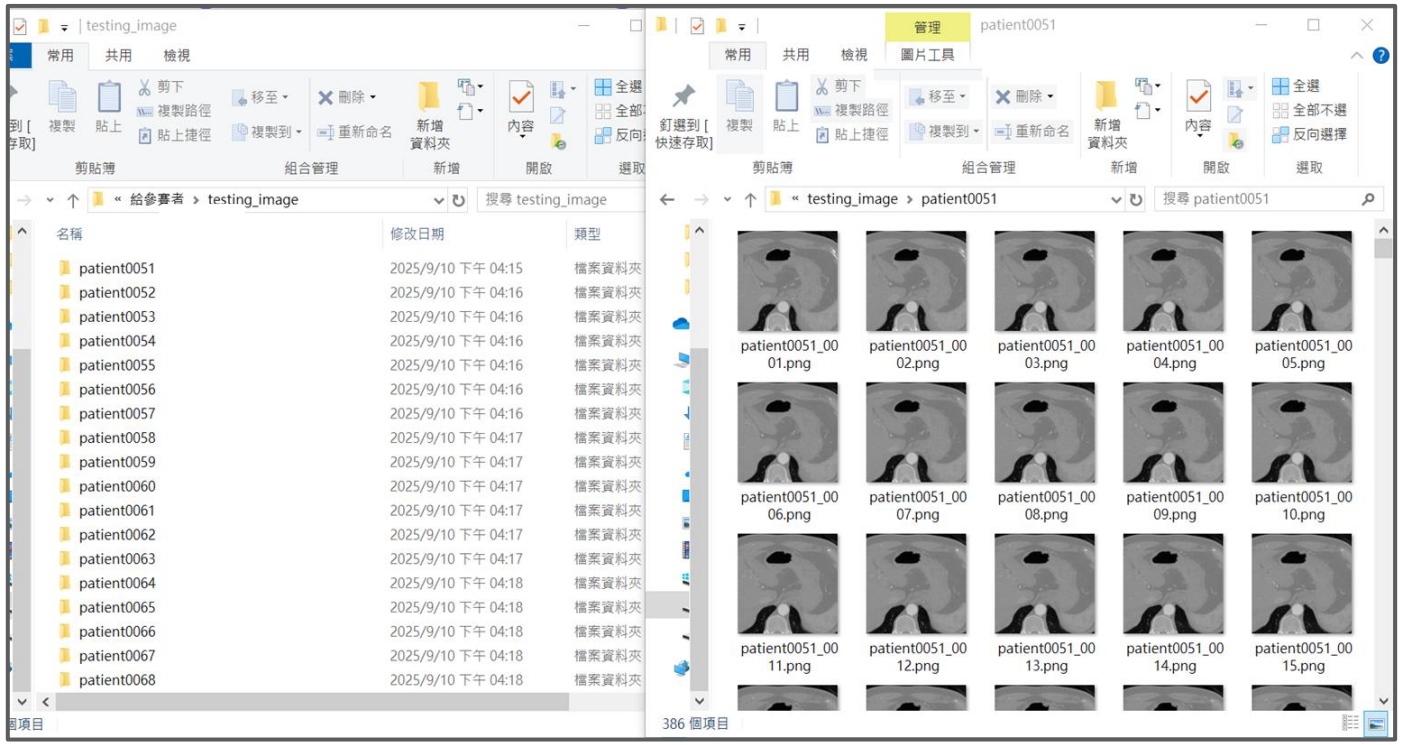


▲ 還原後的點座標

測試集組成(50筆電腦斷層掃描)

- Testing Dataset
  - 只有testing_image 資料夾
  - patient0051 ~ patient0100 為測試集

- testing_image 檔案夾中
  - 每個 **patient 檔案夾**包含有 250~450 張不等之 PNG 圖檔
  - 每張 PNG 圖檔尺寸為 512 * 512 pixel



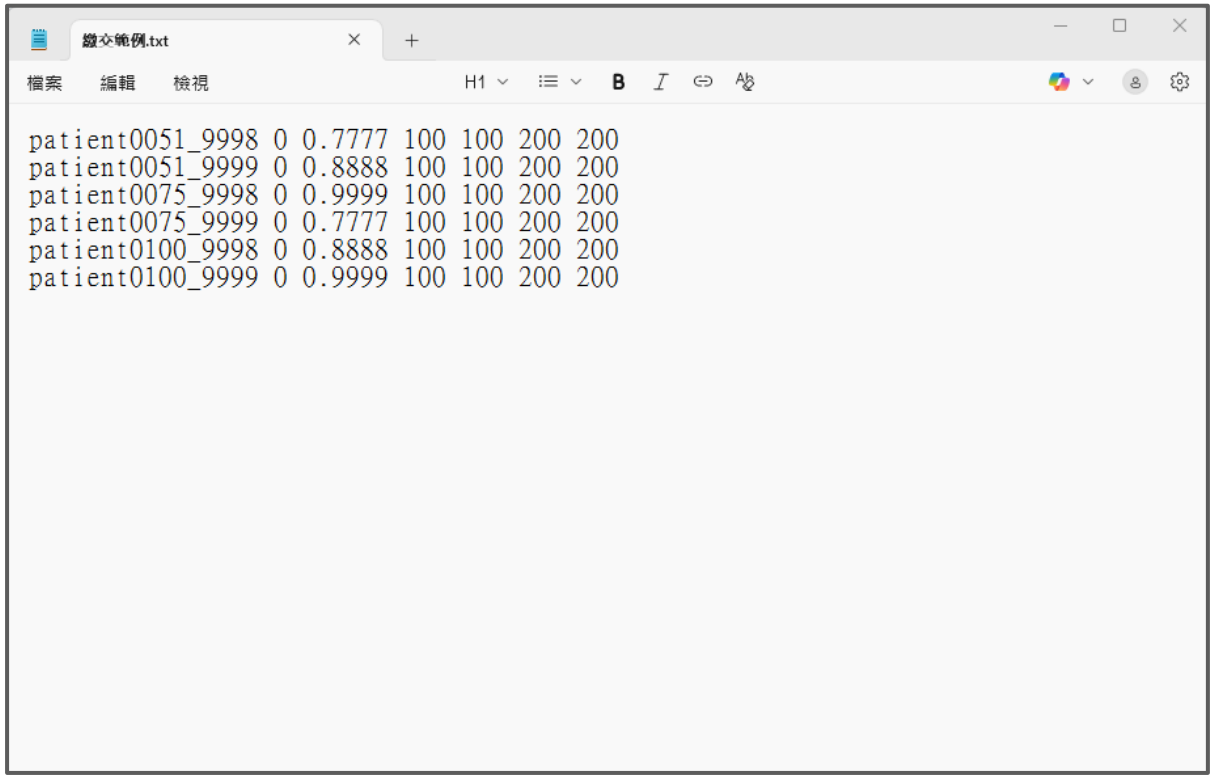
▲ testing_image 檔案夾

02

繳交檔案格式說明

繳交合併後的檔案

- 參賽者只需要繳交**共一份**合併後包含patient0051~patient0100預測結果的.txt (可參考" 繳交範例.txt " https://drive.google.com/file/d/1LNg_jEYqgNvNbH-1aZRFsOZX_4YJd3T0/view?usp=sharing)。

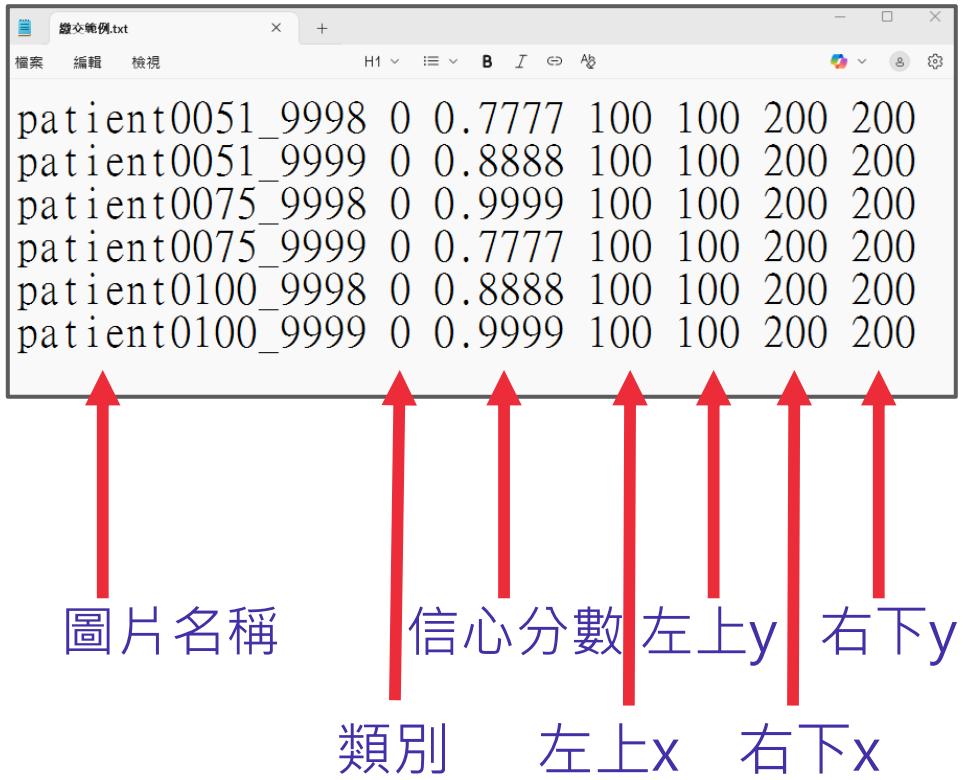


▲繳交範例

繳交內容格式說明

- 每一行為一個預測框格式如下。

圖片名稱(不包含.png) -空格-類別-空格-信心分數(0~1之間的小數)-空格-預測框左上角原始 x 座標(整數)-空格-預測框左上角原始 y 座標(整數)-空格-預測框右下角原始 x 座標(整數)-空格-預測框右下角原始 y 座標(整數)



▲繳交範例內各項數值

信心分數&沒有預測到主動脈瓣的圖片

- 信心分數只影響預測框與Ground Truth的配對順序。
- 信心分數數值高低不會直接影響AP分數。
- 若使用模型的預測結果無信心分數可全部都填0或1。
- 沒有預測到主動脈瓣的圖片不需要寫入任何數據在.txt中。

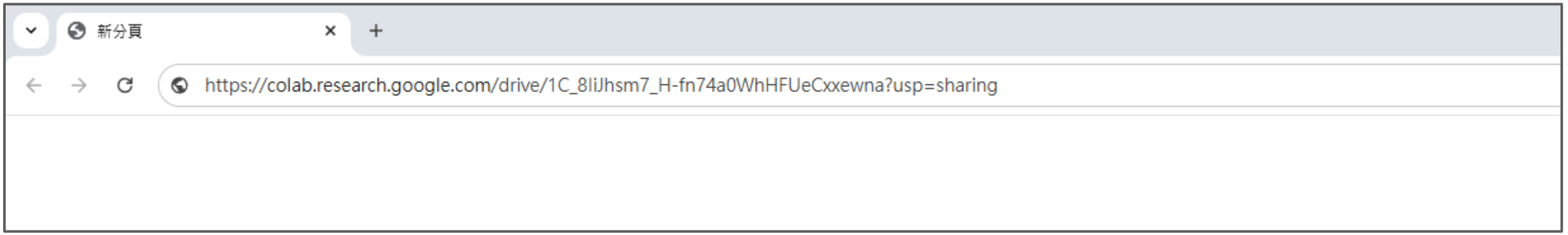
03

Colab Baseline 程式說明 (train)

打開瀏覽器

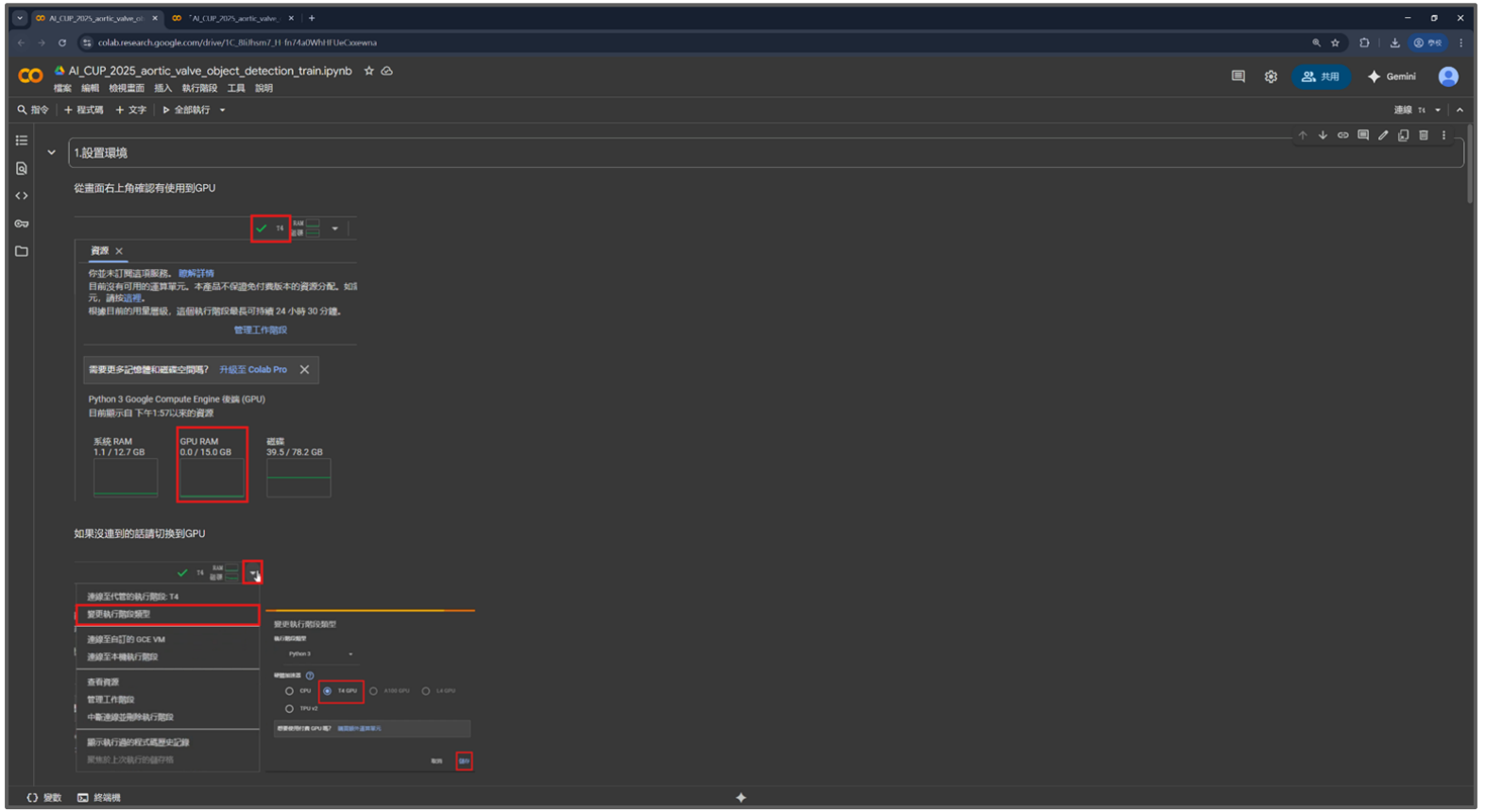
新興智慧顯示科技
教育聯盟

Baseline 會透過瀏覽器打開 Colab 完成模型訓練。



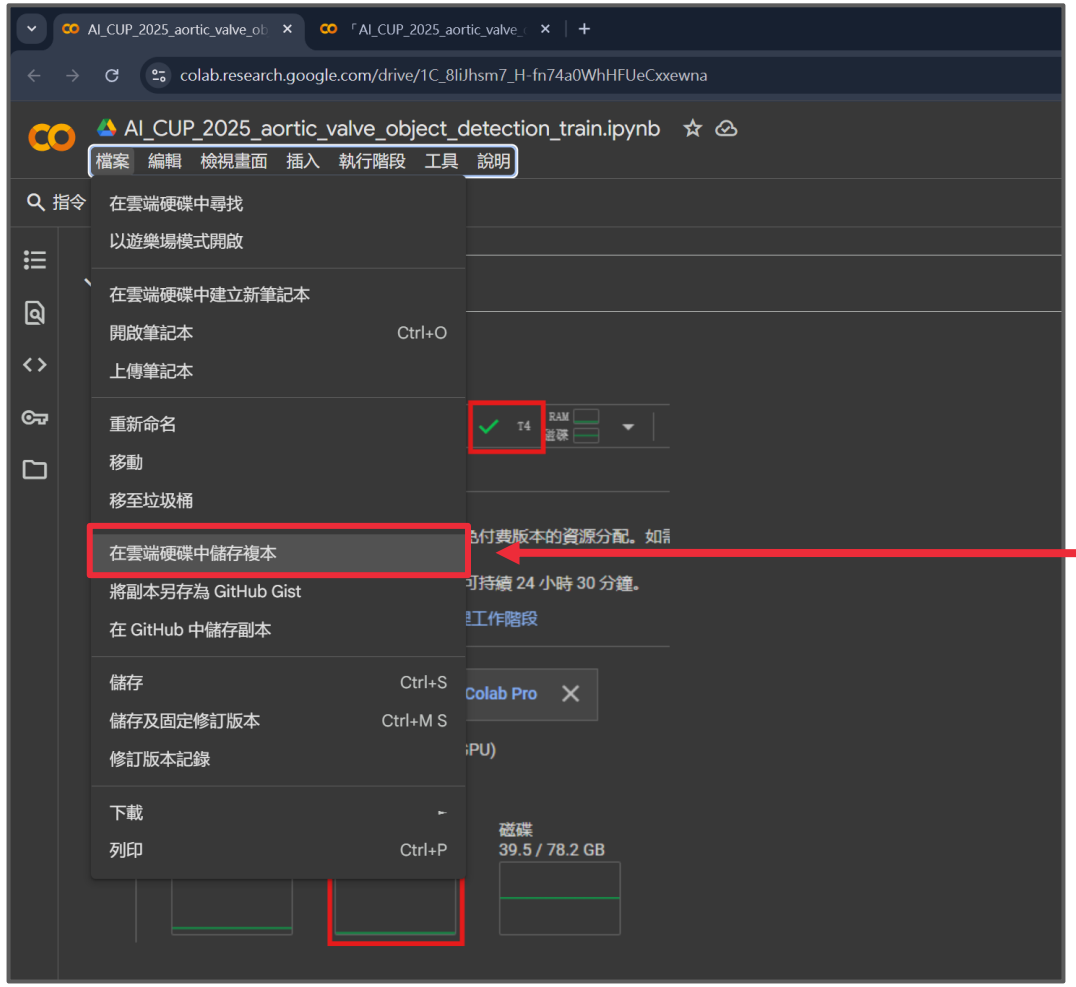
開啟 Colab (train)

Colab連結 : https://colab.research.google.com/drive/1C_8IiJhsm7_H-fn74a0WhHFUeCxxewna?usp=sharing



儲存副本至個人雲端

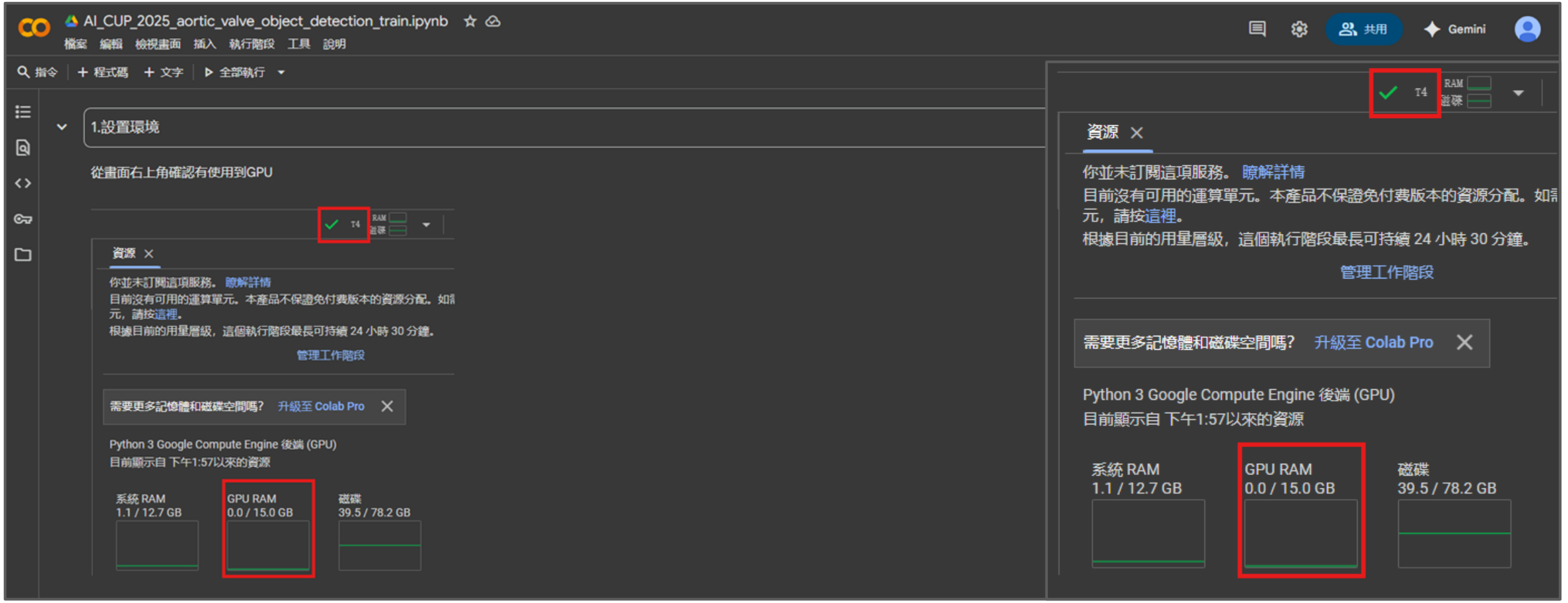
如果想保留運行過程，請先登入Google帳號並儲存副本再開始運行。



儲存副本

連線至GPU

點選 Colab 右上角進行連線，連線有綠色勾後可以再次點開查看資源是否有 GPU RAM。



切換連線至GPU

如果沒有連線至 GPU 可以點選右上倒三角形 -> 變更執行階段類型 -> 選取 T4 GPU -> 儲存，切換連線至 GPU 。

Colab運行情況(成功)

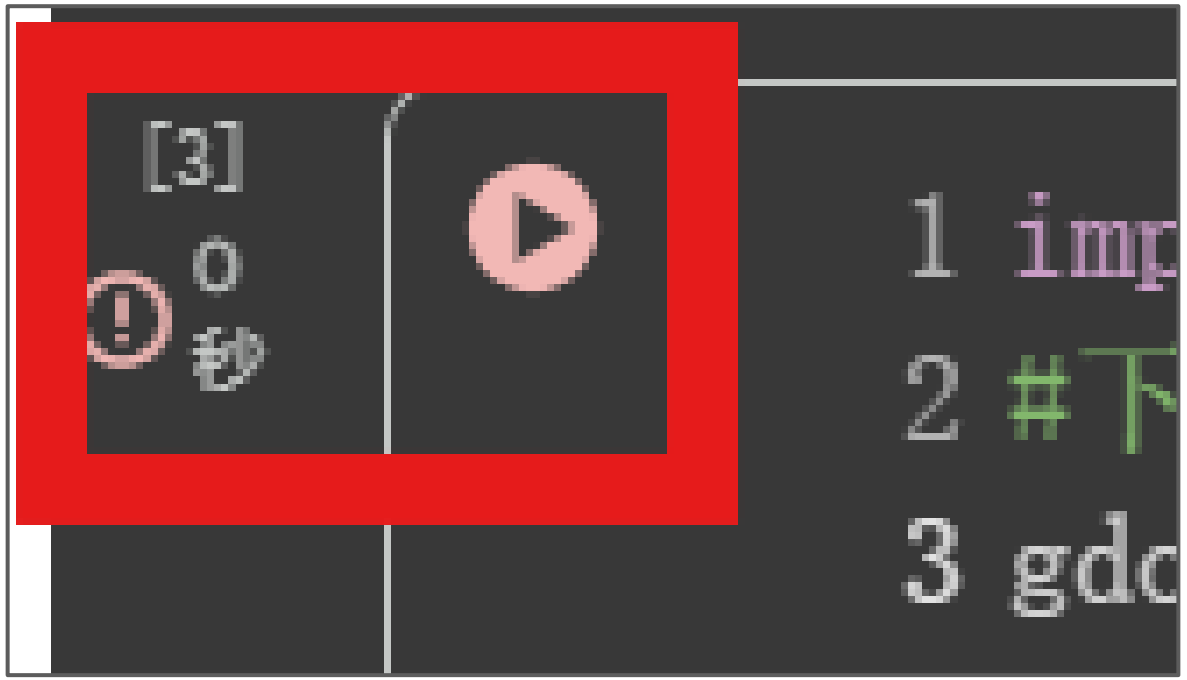
Colab區塊左方有綠色勾代表執行完成且成功

Colab運行情況(執行中)

區塊左方有圈在轉代表執行中

Colab運行情況(運行錯誤)

區塊左方有紅色驚嘆號代表運行錯誤



設置環境

確保不會出現編碼錯誤和下載YOLOv12套件。

查看GPU資訊(運行時間<1秒)

```
[ ]
1 !nvidia-smi
```

```
Tue Sep  9 14:10:10 2025
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   38C    P8              9W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

確保不會出現編碼錯誤(運行時間<1秒)

```
[1]
0
秒
1 import  locale
2 def  getpreferredencoding(do_setlocale  =  True):
3         return  "UTF-8"
4 locale.getpreferredencoding  =  getpreferredencoding
```

下載YOLOv12套件(運行時間12秒)

```
[2]
12
秒
1 !pip  install  ultralytics
2 import  ultralytics
3 ultralytics.checks()
```

```
Ultralytics 8.3.198 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 39.0/112.6 GB disk)
```

上傳訓練資料集和.yaml檔

因為要從雲端下載資料集到 Colab 所以先將 **training_image.zip**、**training_label.zip** 和 **aortic_valve_colab.yaml** 上傳至個人的Google雲端硬碟。

.yaml 為 YOLO 訓練時讀取**資料路徑**和**預測類別**的檔案

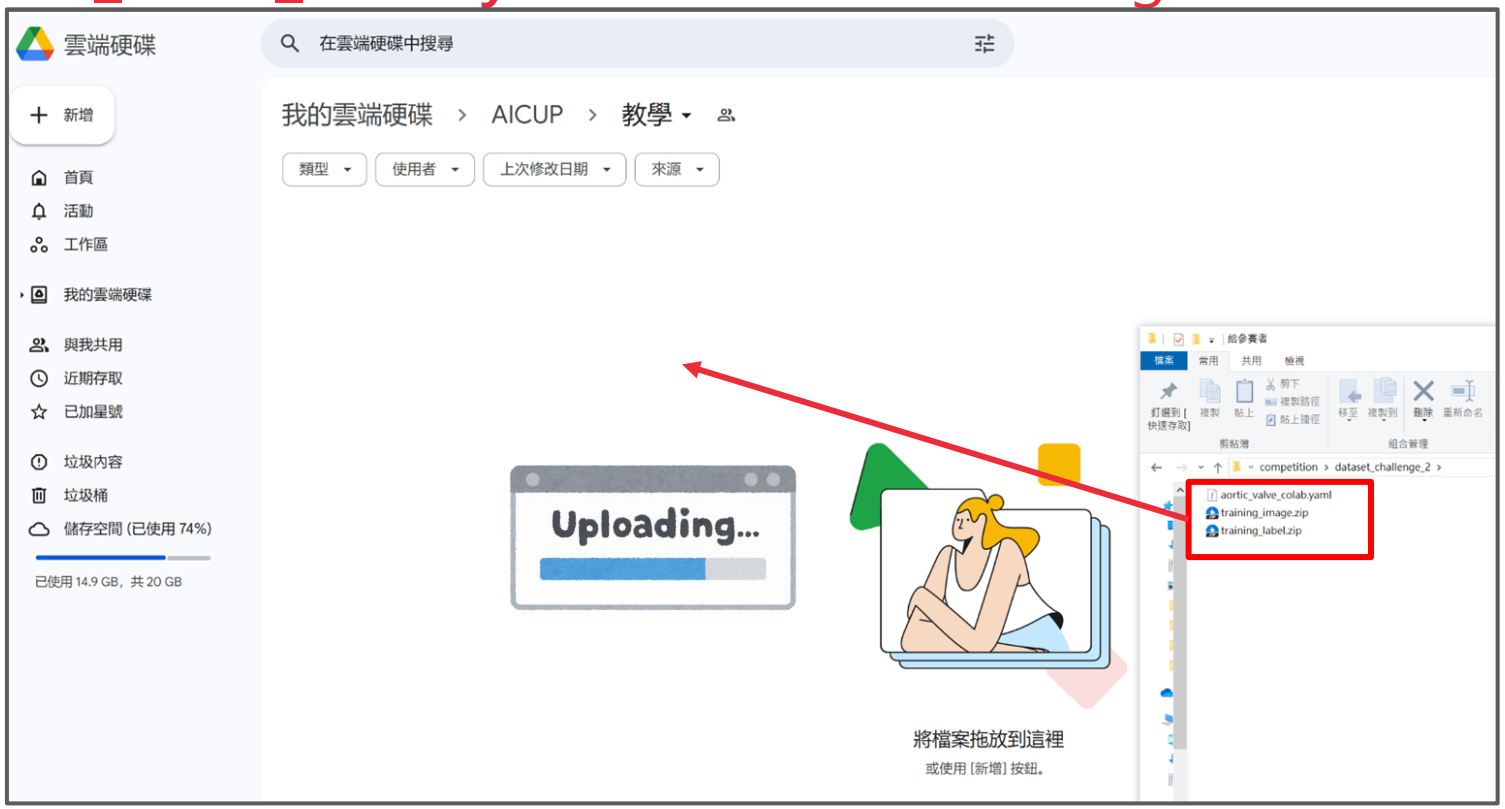
.yaml 範例:https://drive.google.com/file/d/13hSr3sa2wOZqlvY1RAwr2msCfTdkfjMe/view?usp=sharing

aortic_valve_colab .yaml

competition > aortic_valve_colab .yaml

```yaml
train: "./datasets/train/images"
val: "./datasets/val/images"

names:
  0: aortic_valve
```

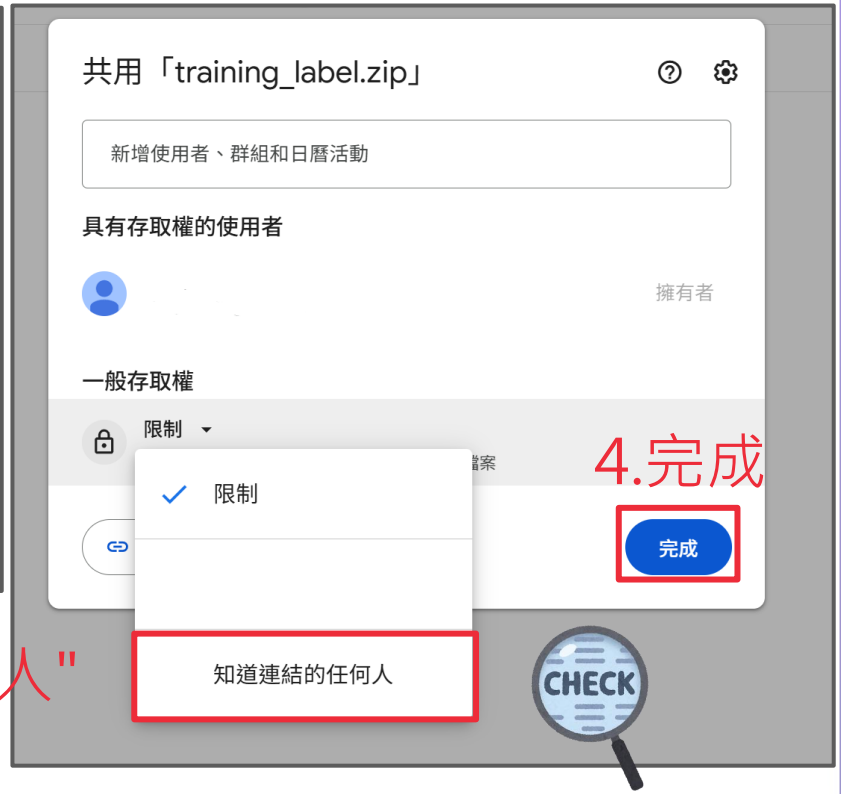
▲ aortic_valve_colab.yaml 內容



▲ 將檔案上傳至雲端

更改檔案權限

檔案權限從**限制**更改為**知道連結的任何人**。
(training_image.zip、training_label.zip 和 aortic_valve_colab.yaml **三個檔案權限都要更改**)



取得檔案連結方法

3.複製連結

轉換連結為可直接下載格式

將剛剛取得的網址透過下方網站轉換成直接下載的格式。
轉換網站：
https://sites.google.com/view/twdrivefromdownload/%E7%B9%81%E9%AB%94%E4%B8%AD%E6%96%87traditional-chinese?authuser=0

替換下載連結

使用轉換後的三個連結將 Colab 下載資料集的網址替換掉。

透過上面方法更改下方程式的三個網址

*建議自行上傳雲端替換成自己的連結，此資料集雖然與最初競賽資料集相同，但不保證會更新

```
[ ]
1 #下載資料集
2 import gdown
3 import os
4 import shutil
5
6 #下載training_image.zip
7 gdown.download("https://drive.google.com/uc?export=download&id=1FfrsOgeJKSHtRM3Q649HJvSF1l4blxpl","/content/training_image.zip")
8 #下載training_label.zip
9 gdown.download("https://drive.google.com/uc?export=download&id=1KmYS2JhAexpWWlzlo_qwoX1TifA1COWW","/content/training_label.zip")
10 #下載訓練aortic_valve_colab.yaml
11 gdown.download("https://drive.google.com/u/0/uc?id=1OyGJ7mfVLutjkE55U9wfs4bmHCtIN1cv&export=download","/content/aortic_valve_colab.yaml")
```

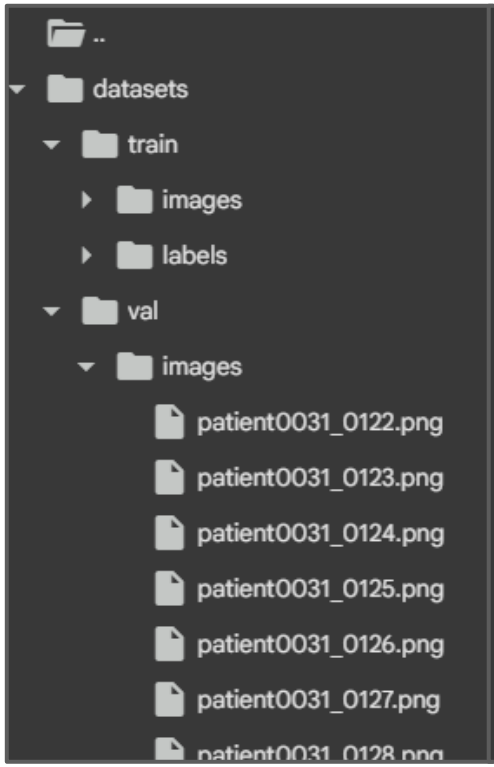
training_image.zip

training_label.zip

training_label.zip

移動檔案(透過程式)

baseline 將**前30筆**資料用於**訓練**，**後20筆**資料用於**驗證**，
可在程式中**53行**及**56行**更改比例
因為Colab運算資源有限此處只採用有標註檔的圖片，其餘圖片可以自行修改程式利用
將前30筆資料的圖片全部移動至 ./datasets/train/images，
前30筆資料的標註全部移動至 ./datasets/train/labels，
後20筆資料的圖片全部移動至 ./datasets/val/images，
後20筆資料的標註全部移動至 ./datasets/val/labels。




```python
#移動檔案
def find_patient_root(root):
    """往下找，直到找到含有 patientXXXX 的資料夾"""
    for dirpath, dirnames, filenames in os.walk(root):
        if any(d.startswith("patient") for d in dirnames):
            return dirpath
    return root  # fallback

# 解壓縮到固定資料夾
if not os.path.isdir("./training_image") and os.path.exists("training_image.zip"):
    os.makedirs("./training_image", exist_ok=True)
    !unzip -q training_image.zip -d ./training_image

if not os.path.isdir("./training_label") and os.path.exists("training_label.zip"):
    os.makedirs("./training_label", exist_ok=True)
    !unzip -q training_label.zip -d ./training_label

IMG_ROOT = find_patient_root("./training_image")
LBL_ROOT = find_patient_root("./training_label")

print("IMG_ROOT =", IMG_ROOT)
print("LBL_ROOT =", LBL_ROOT)

# 建立輸出資料夾
os.makedirs("./datasets/train/images", exist_ok=True)
os.makedirs("./datasets/train/labels", exist_ok=True)
os.makedirs("./datasets/val/images", exist_ok=True)
os.makedirs("./datasets/val/labels", exist_ok=True)
```

```python
def move_patients(start, end, split):
    for i in range(start, end + 1):
        patient = f"patient{i:04d}"
        img_dir = os.path.join(IMG_ROOT, patient)
        lbl_dir = os.path.join(LBL_ROOT, patient)
        if not os.path.isdir(lbl_dir):
            continue

        for fname in os.listdir(lbl_dir):
            if not fname.endswith(".txt"):
                continue

            label_path = os.path.join(lbl_dir, fname)
            base, _ = os.path.splitext(fname)  # 取出檔名不含副檔名
            img_path = os.path.join(img_dir, base + ".png")
            if not os.path.exists(img_path):
                print(f"找不到對應圖片: {img_path}")
                continue

            shutil.move(img_path, f"./datasets/{split}/images/")
            shutil.move(label_path, f"./datasets/{split}/labels/")

# patient0001~0030 → train
move_patients(1, 30, "train")

# patient0031~0050 → val
move_patients(31, 50, "val")

print("完成移動！")
```

確認檔案

確認檔案是否成功移動，如果執行完數量和下圖相同代表成功。

確認檔案是否成功移動

如果資料數量符合下圖數量代表移動成功

```
訓練集圖片數量 :  1631
訓練集標記數量 :  1631
驗證集圖片數量 :  1156
驗證集標記數量 :  1156
```

```
[ ]
1 !ls
```

```
aortic_valve_colab.yaml  sample_data     training_image.zip  training_label.zip
datasets                 training_image  training_label
```

```
[ ]
1 print('訓練集圖片數量  :  ',len(os.listdir("./datasets/train/images")))
2 print('訓練集標記數量  :  ',len(os.listdir("./datasets/train/labels")))
3 print('驗證集圖片數量  :  ',len(os.listdir("./datasets/val/images")))
4 print('驗證集標記數量  :  ',len(os.listdir("./datasets/val/labels")))
```

```
訓練集圖片數量 :  1631
訓練集標記數量 :  1631
驗證集圖片數量 :  1156
驗證集標記數量 :  1156
```

訓練模型

執行後依序有 Epoch 在跑代表成功

3. 訓練模型(運行時間約15分鐘)

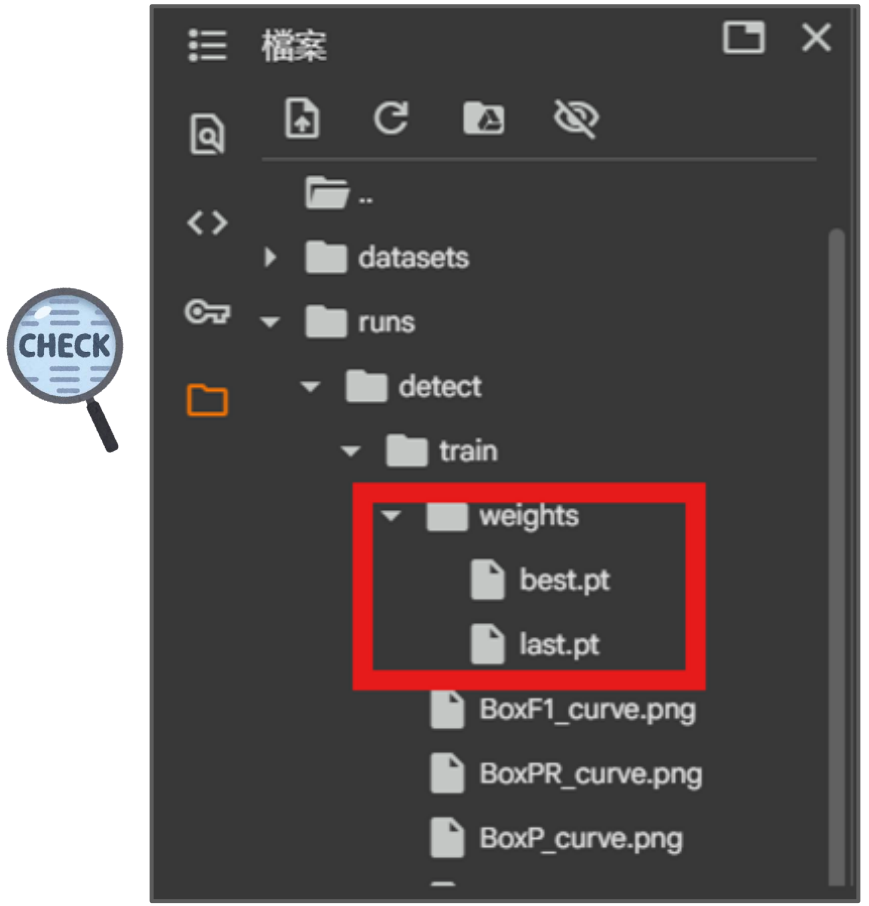
執行後依序有Epoch在跑代表成功

```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/20      3.73G      1.938      3.491      1.567         23        640: 100% ━━━━━━━━━━━━ 102/102 2.7it/s 37.6s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 37/37 3.1it/s 12.0s
             all       1156       1156     0.0687      0.344     0.0596     0.0254

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/20      3.75G      1.49      1.877      1.279         31        640: 100% ━━━━━━━━━━━━ 102/102 2.9it/s 34.9s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 37/37 3.6it/s 10.2s
             all       1156       1156      0.645      0.765       0.73      0.397

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/20      3.76G      1.45      1.466      1.271         16        640: 100% ━━━━━━━━━━━━ 102/102 3.0it/s 33.6s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 37/37 3.4it/s 10.9s
             all       1156       1156      0.672      0.589      0.645        0.3

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/20      3.77G      1.404      1.225      1.252         25        640: 100% ━━━━━━━━━━━━ 102/102 2.9it/s 34.6s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 37/37 3.5it/s 10.5s
             all       1156       1156      0.754      0.735      0.815      0.497

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/20      3.77G      1.331      1.047      1.206         18        640: 100% ━━━━━━━━━━━━ 102/102 3.1it/s 33.2s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 37/37 3.5it/s 10.6s
             all       1156       1156      0.725      0.754      0.807      0.487
```

```
[ ]
1 from ultralytics import YOLO
2
3 model = YOLO('yolo12n.pt') #初次訓練使用YOLO官方的預訓練模型，如要使用自己的模型訓練可以將'yolo12n.pt'替換掉
4 results = model.train(data="./aortic_valve_colab.yaml",
5                       epochs=20,  #跑幾個epoch
6                       batch=16,  #batch_size
7                       imgsz=640,  #圖片大小640*640
8                       device=0  #使用GPU進行訓練
9                       )
```

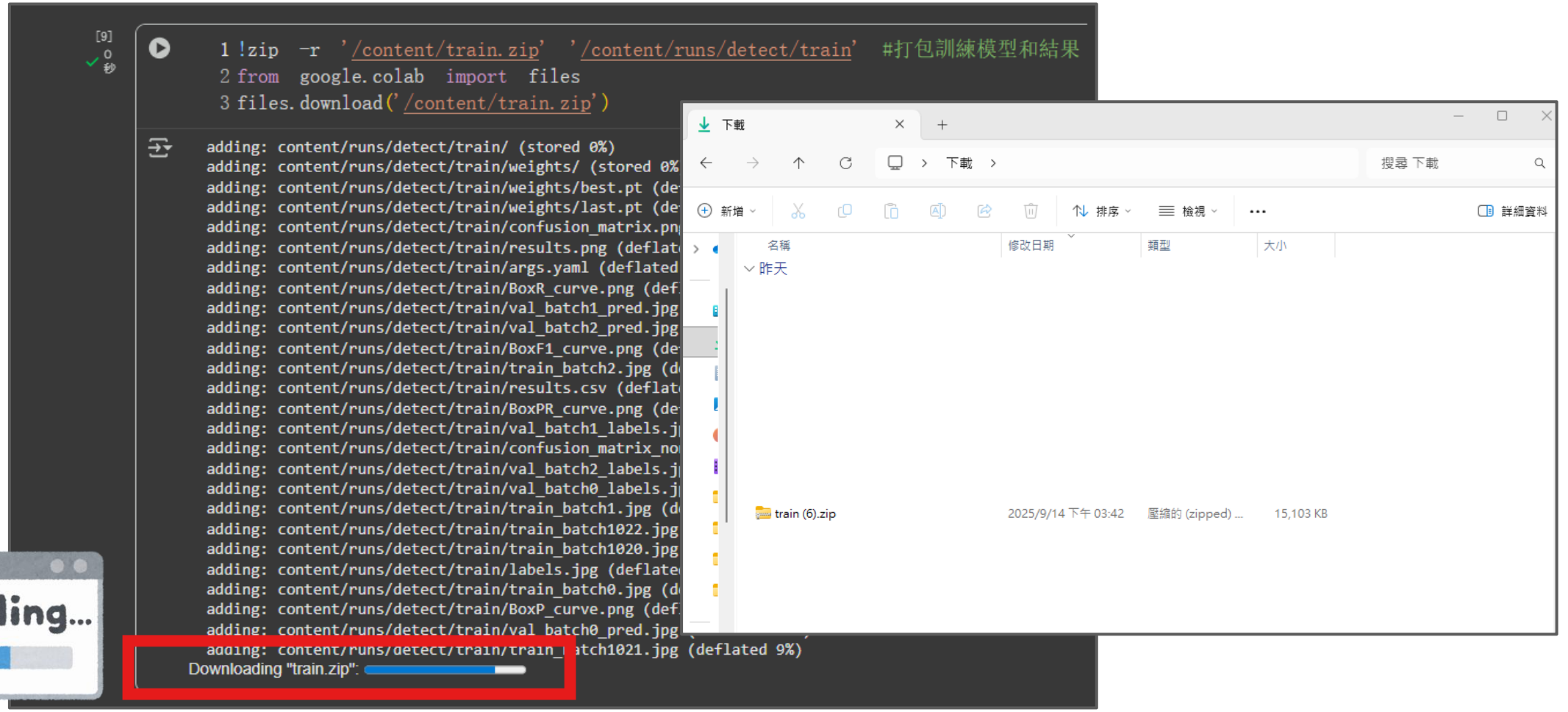
訓練模型

訓練完左方檔案會有 run 資料夾，此份 Colab 主要目標是得到 best.pt 用於下一份 Colab 進行預測(!!如果有重複訓練**請下載最後成功**的 train 編號資料夾)



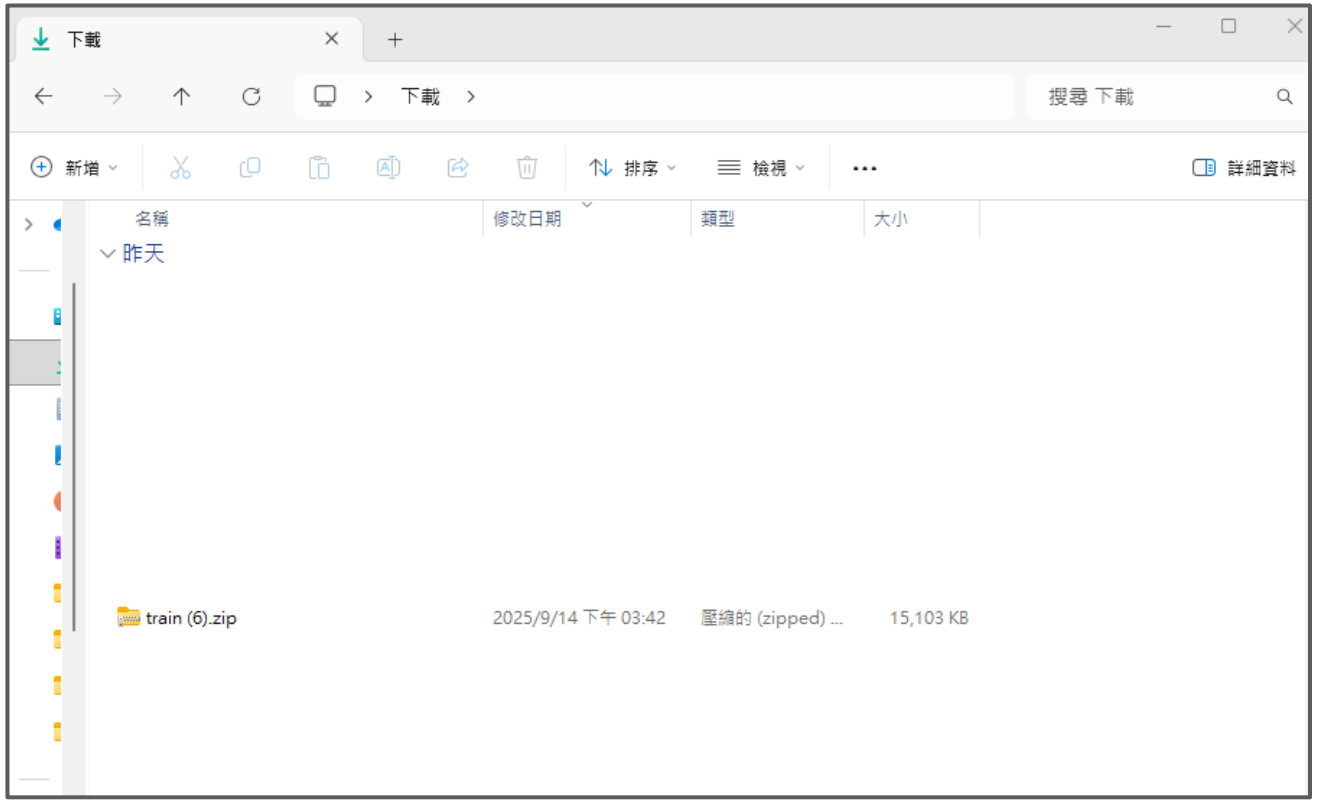
壓縮並下載訓練完的模型

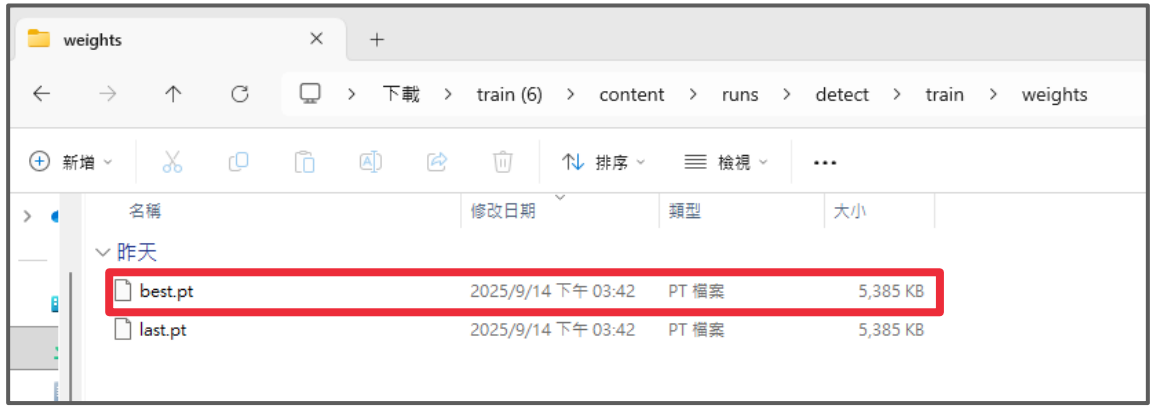
確認下方進度條消失才有下載成功。



從本地下載確認檔案

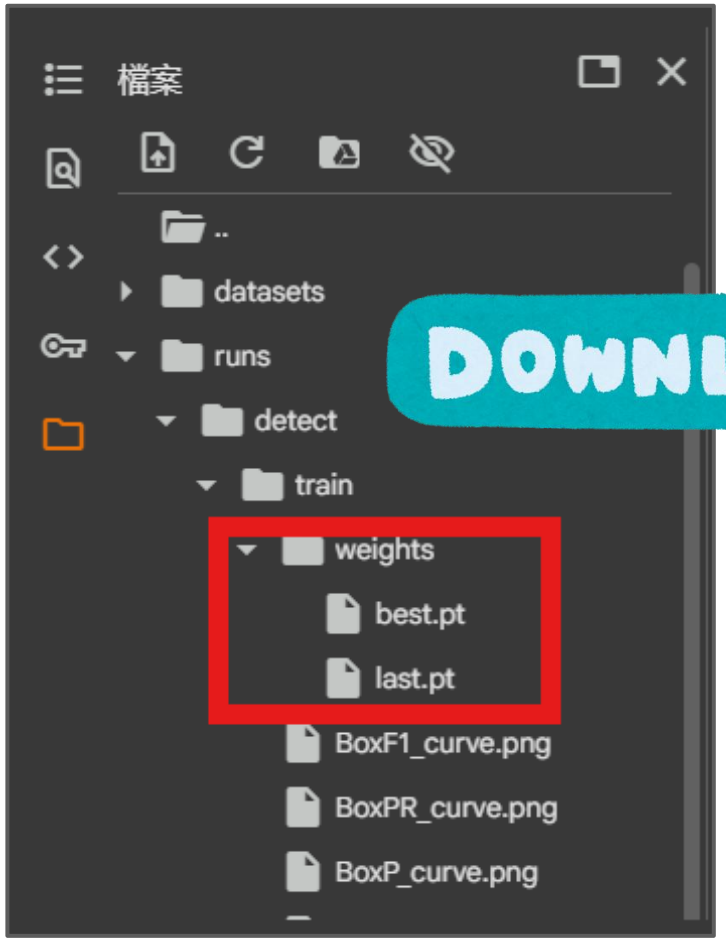
- 有 train.zip (檔案大小約15103 KB) 代表下載成功，預測 Colab 會使用 best.pt 模型權重檔。
- 解壓縮後可在下載\train\content\runs\detect\train\weights找到 best.pt。

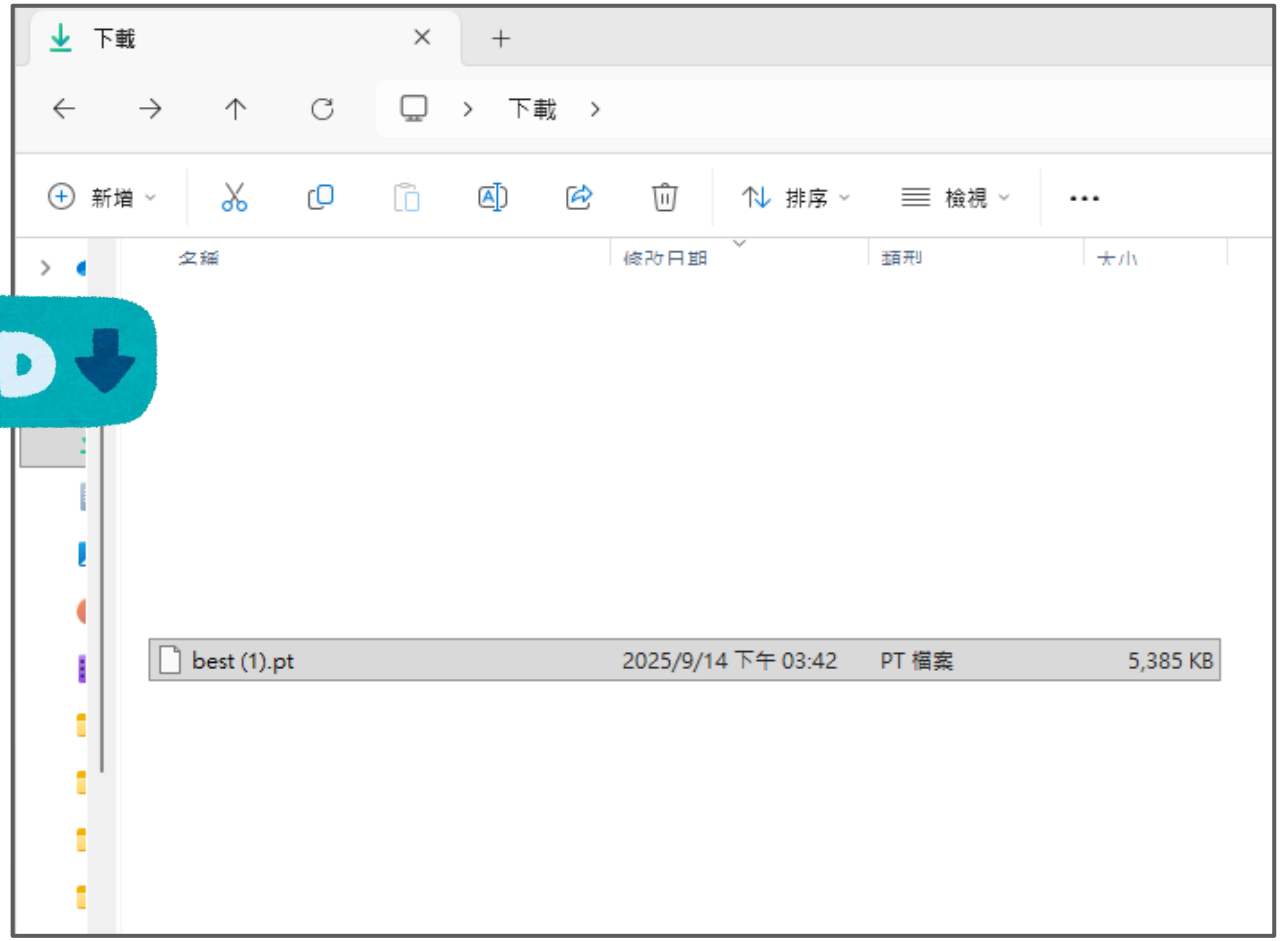




第二種下載方式

也可以右鍵直接下載 best.pt (檔案大小約5385 KB)。





04

Colab Baseline 程式說明 (predict)

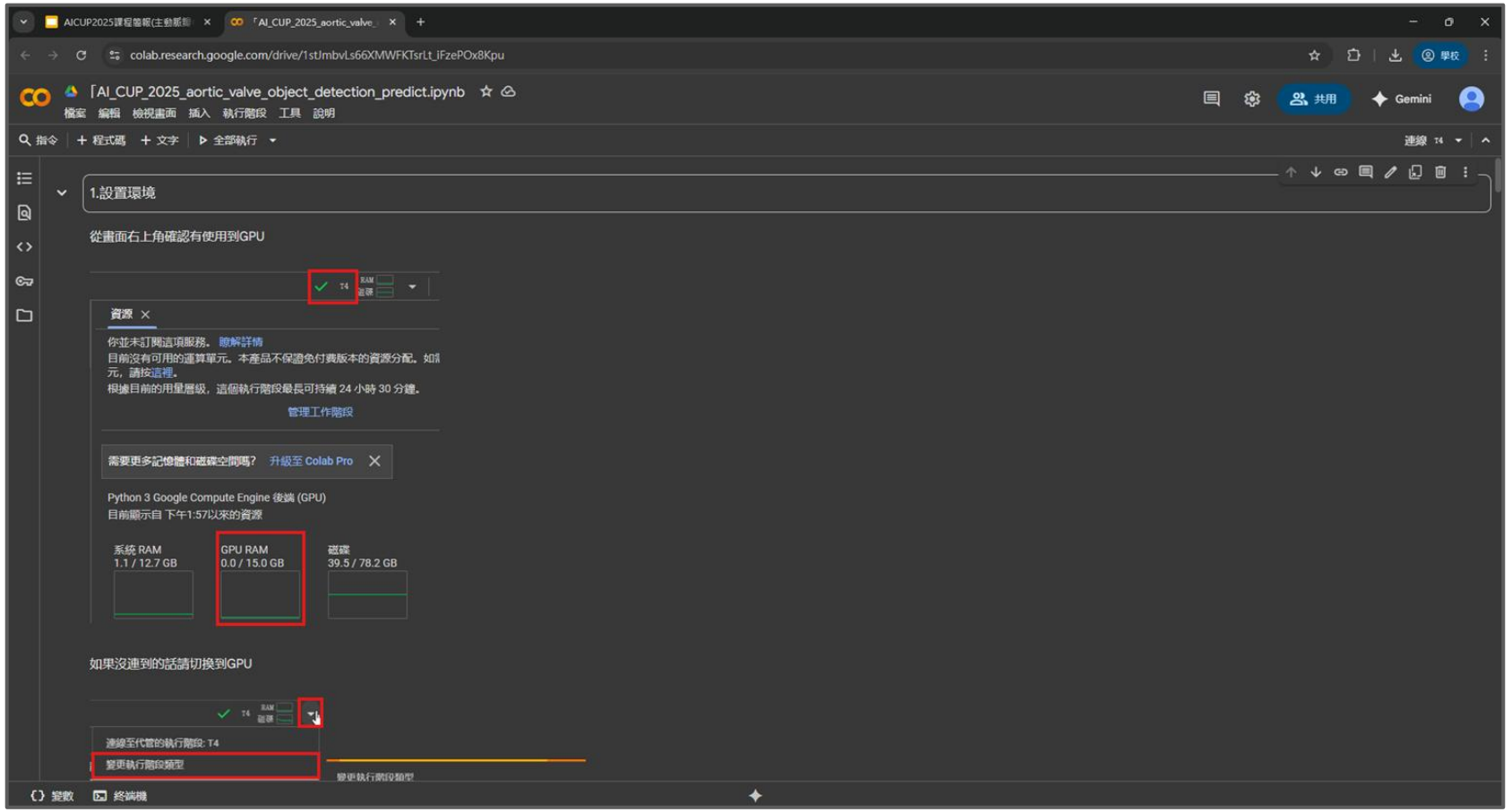
打開瀏覽器

新興智慧顯示科技
教育聯盟

Baseline 會透過瀏覽器打開 Colab 完成預測。

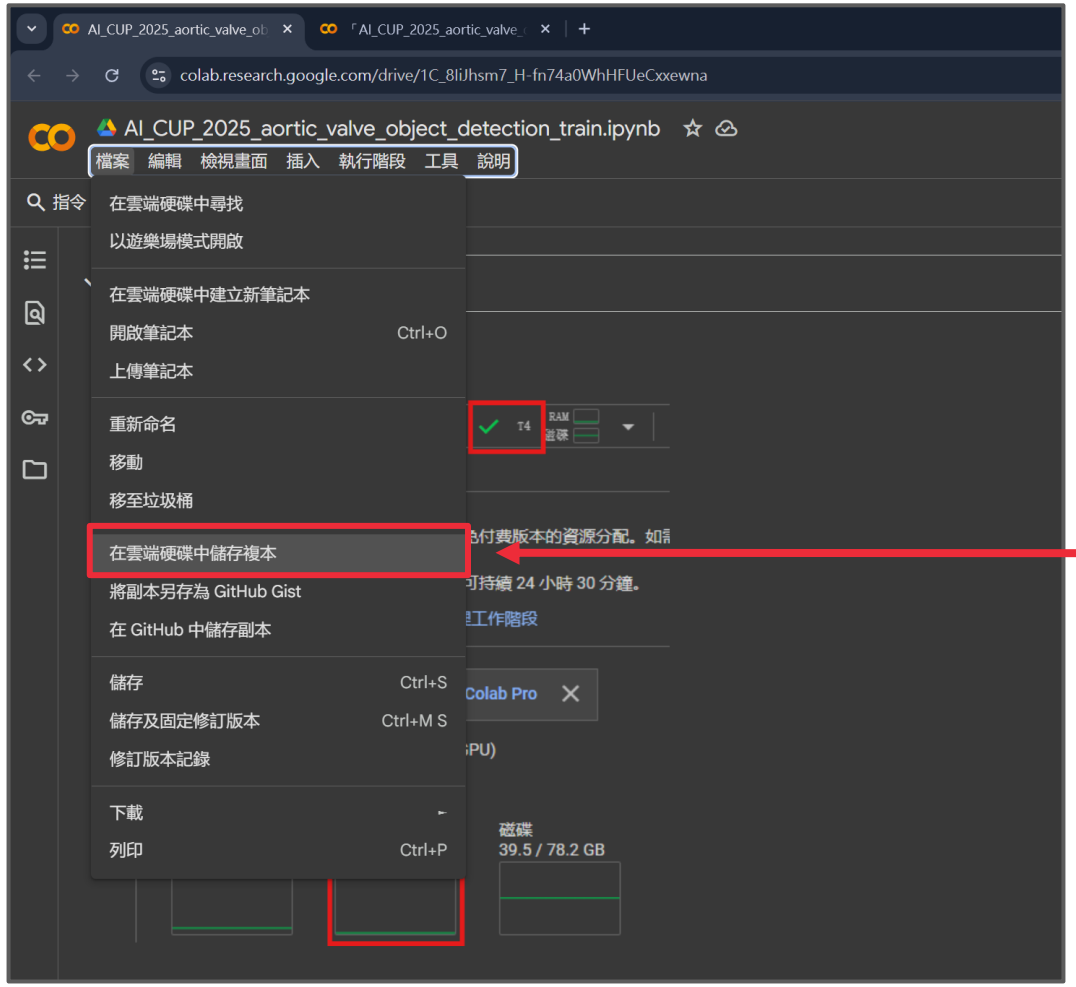
開啟Colab(predict)

https://colab.research.google.com/drive/1stJmbvLs66XMWFKTsrLt_iFzePOx8Kpu?usp=sharing



儲存副本至個人雲端

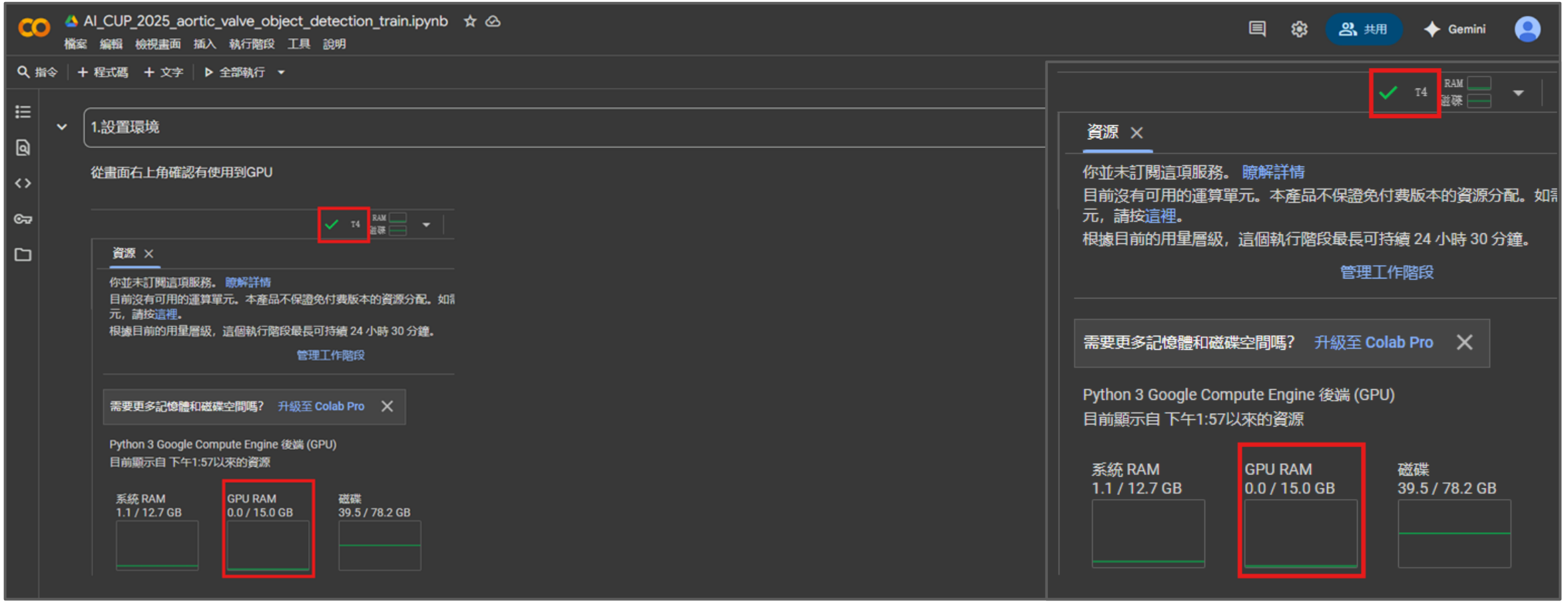
如果想保留運行過程，請先登入Google帳號並儲存副本再開始運行。



儲存副本

連線至GPU

點選 Colab 右上角進行連線，連線有綠色勾後可以再次點開查看資源是否有 GPU RAM。



切換連線至GPU

如果沒有連線至 GPU 可以點選右上倒三角形 -> 變更執行階段類型 -> 選取 T4 GPU -> 儲存，切換連線至 GPU 。

Colab運行情況(成功)

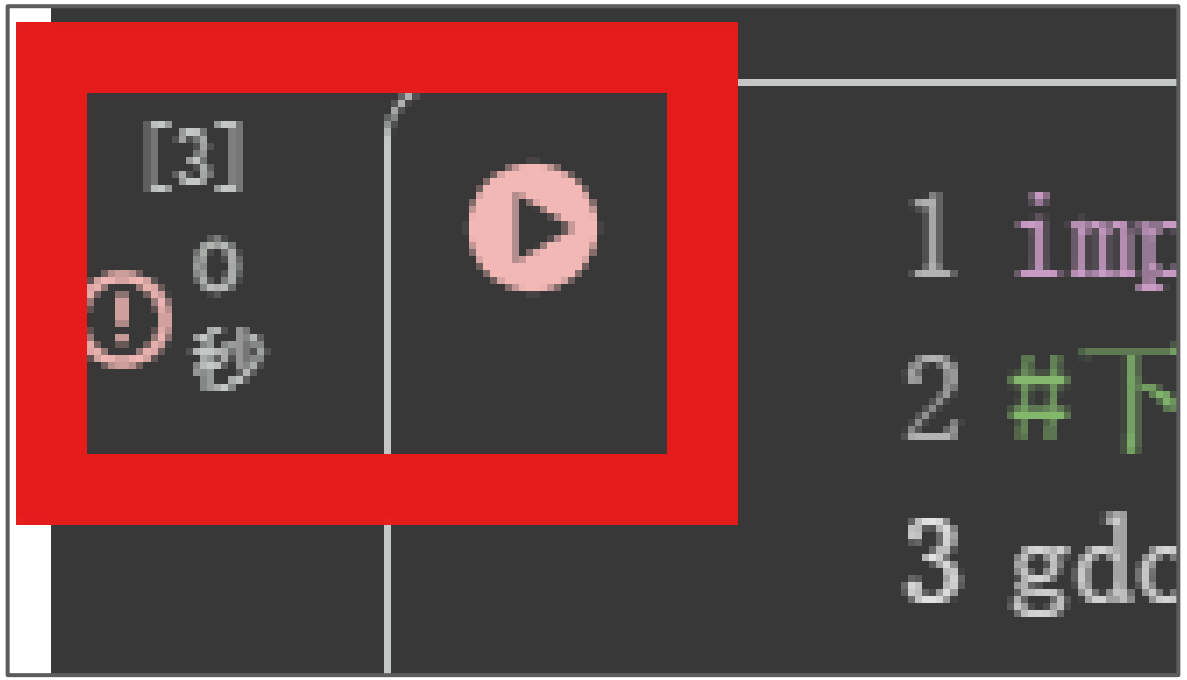
Colab區塊左方有綠色勾代表執行完成且成功

Colab運行情況(執行中)

區塊左方有圈在轉代表執行中

Colab運行情況(運行錯誤)

區塊左方有紅色驚嘆號代表運行錯誤



設置環境

確保不會出現編碼錯誤和下載YOLOv12套件。

查看GPU資訊(運行時間<1秒)

```
[ ]
1 !nvidia-smi
```

```
Tue Sep  9 14:10:10 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   38C    P8              9W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

確保不會出現編碼錯誤(運行時間<1秒)

```
[1]
0
秒
1 import  locale
2 def  getpreferredencoding(do_setlocale  =  True):
3         return  "UTF-8"
4 locale.getpreferredencoding  =  getpreferredencoding
```

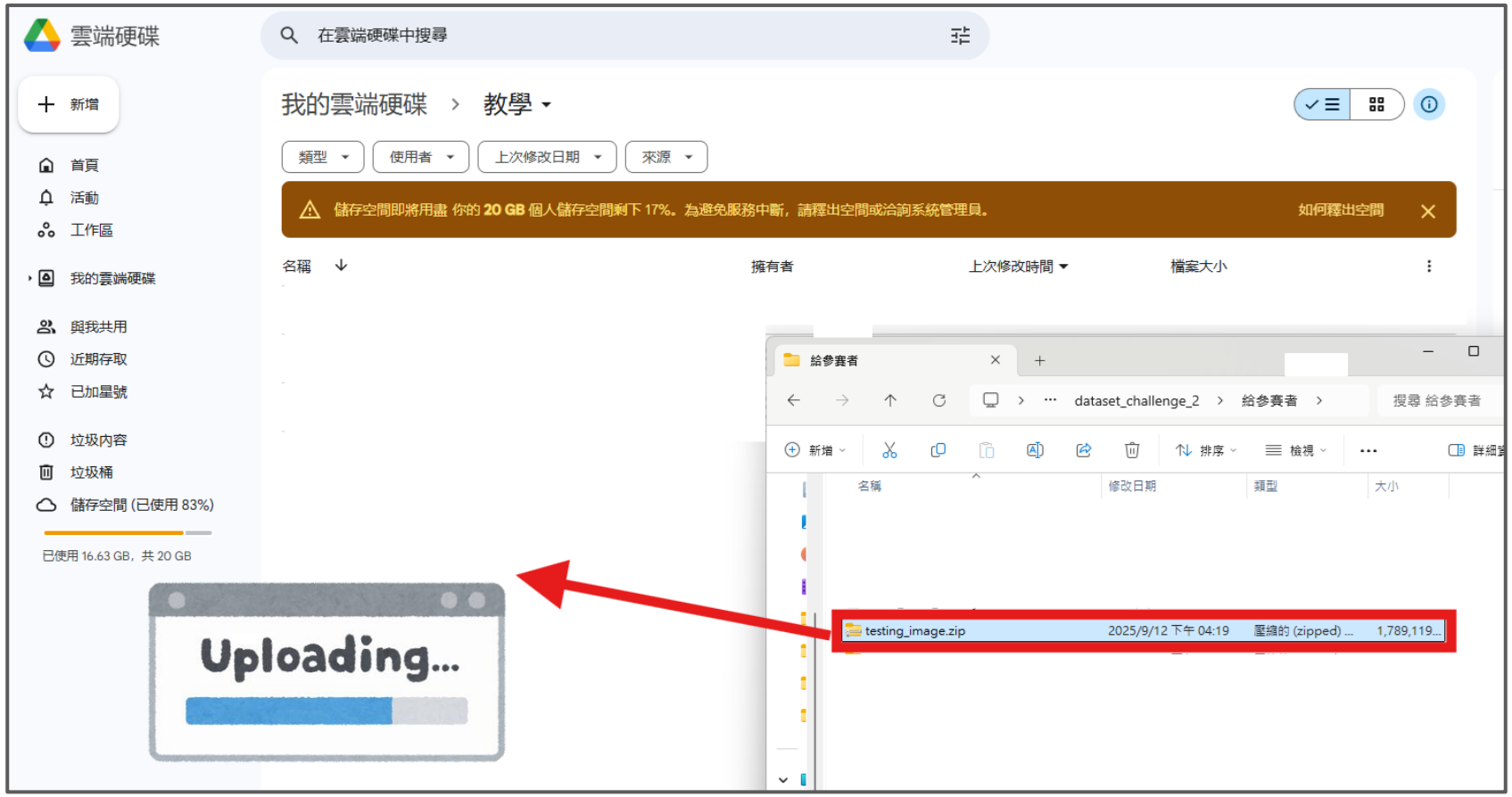
下載YOLOv12套件(運行時間12秒)

```
[2]
12
秒
1 !pip  install  ultralytics
2 import  ultralytics
3 ultralytics.checks()
```

```
Ultralytics 8.3.198 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 39.0/112.6 GB disk)
```

上傳測試資料集

因為要從雲端下載資料集到 Colab 所以先將 testing_image.zip 上傳至個人的 Google雲端硬碟。



更改檔案權限

更改 testing_image.zip 檔案權限從**限制**更改為**知道連結的任何人**。

3. 選取"知道連結的任何人"

取得檔案連結方法

取得檔案下載連結

轉換連結為可直接下載格式

將剛剛取得的網址透過下方網站轉換成直接下載的格式

轉換網站：

https://sites.google.com/view/twdrivefromdownload/%E7%B9%81%E9%AB%94%E4%B8%AD%E6%96%87traditional-chinese?authuser=0

替換下載連結

使用轉換後連結將 Colab 下載資料集的網址替換掉。

透過上面方法更改下方程式的網址(執行時間<1分鐘)

*建議自行上傳雲端替換成自己的連結，此範例連結雖然與最初競賽資料集相同，但不保證會更新

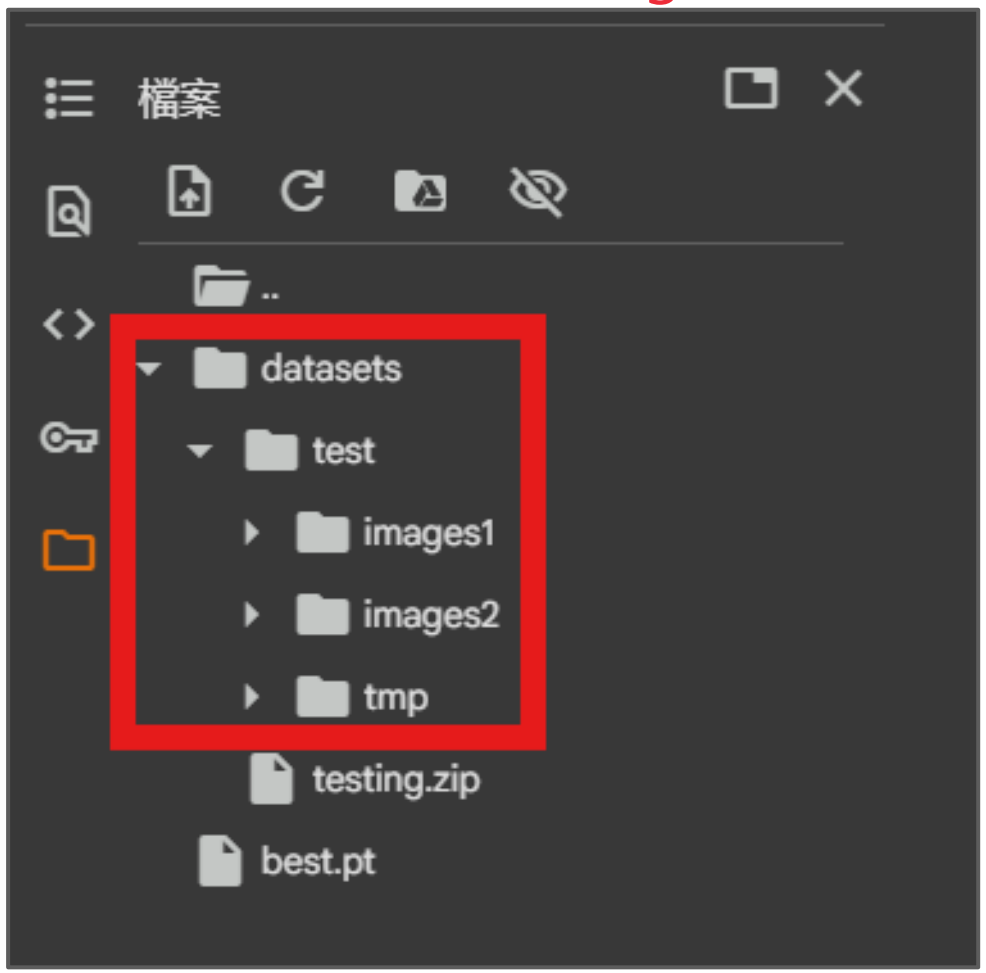
```
#下載資料集
import  gdown
import  os
!mkdir  ./datasets
!mkdir  ./datasets/test
gdown.download("https://drive.google.com/uc?export=download&id=1B-CMYW4MDPTc9Yl3p5trDMdyA7KdVNXd","/content/datasets/testing.zip")
!unzip  '/content/datasets/testing'  -d  '/content/datasets/test/tmp'
```

Colab 運算資源限制

因為本競賽 Testing 資料集有16620張圖片，如果全部一起預測Colab的RAM會不足，所以將Testing圖片分成兩次做預測。

將圖片分到兩個檔案夾

一半的圖片移至 imges1 檔案夾，另一半移至 imges2 檔案夾。

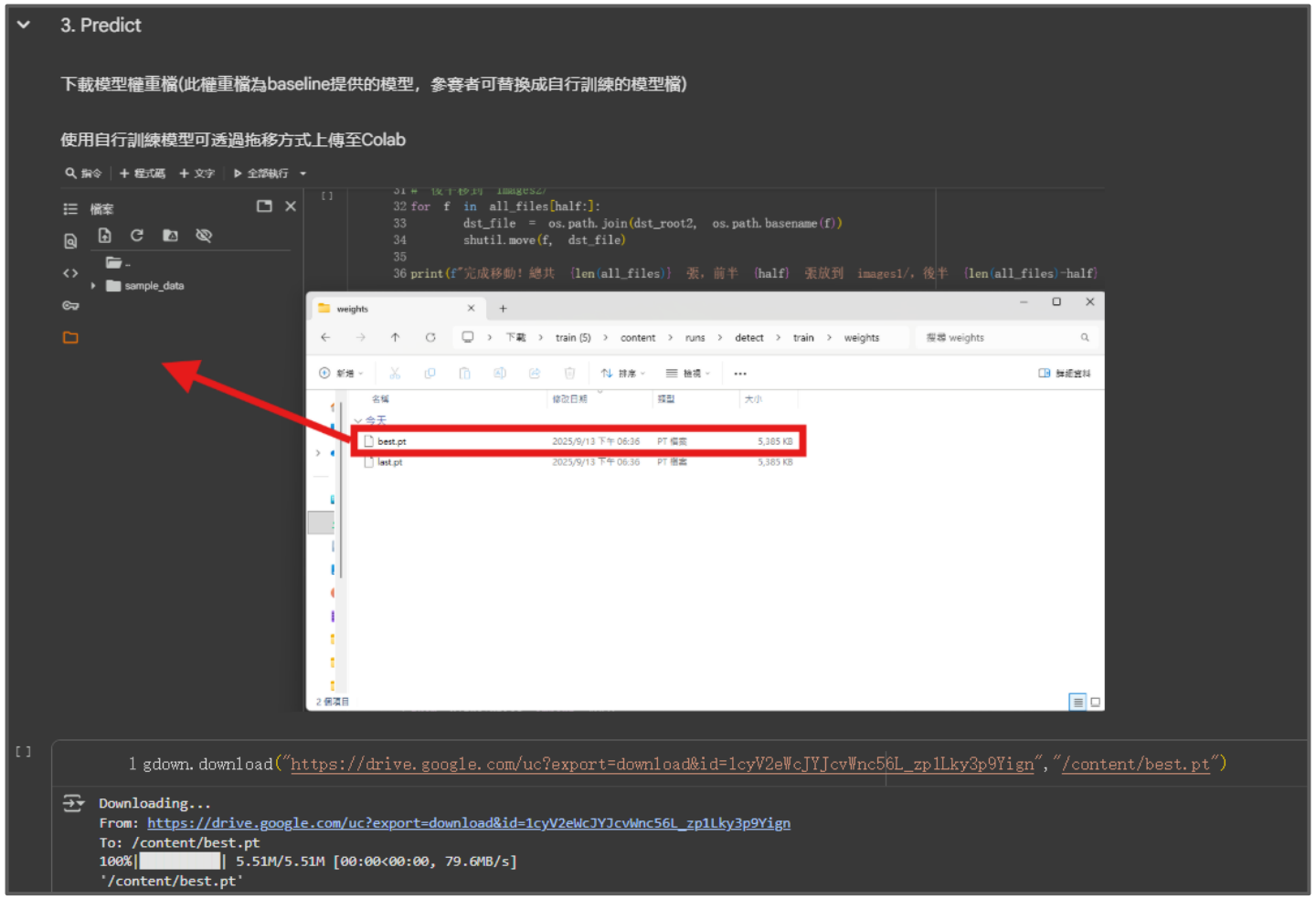



```python
import os
import shutil

src_root = "/content/datasets/test/tmp"
dst_root1 = "/content/datasets/test/images1"
dst_root2 = "/content/datasets/test/images2"

os.makedirs(dst_root1, exist_ok=True)
os.makedirs(dst_root2, exist_ok=True)

# 收集所有圖片路徑
all_files = []
for patient_folder in os.listdir(src_root):
    patient_path = os.path.join(src_root, patient_folder)
    if os.path.isdir(patient_path) and patient_folder.startswith("patient"):
        for fname in os.listdir(patient_path):
            if fname.endswith(".png"):
                all_files.append(os.path.join(patient_path, fname))

# 按照檔名排序，方便重現結果
all_files.sort()

# 計算一半
half = len(all_files) // 2

# 前半移到 images1/
for f in all_files[:half]:
    dst_file = os.path.join(dst_root1, os.path.basename(f))
    shutil.move(f, dst_file)

# 後半移到 images2/
for f in all_files[half:]:
    dst_file = os.path.join(dst_root2, os.path.basename(f))
    shutil.move(f, dst_file)

print(f"完成移動！總共 {len(all_files)} 張，前半 {half} 張放到 images1/，後半 {len(all_files)-half} 張放到 images2/")
```

```
完成移動！總共 16620 張，前半 8310 張放到 images1/，後半 8310 張放到 images2/
```

上傳模型權重

將訓練得到的 best.pt 用拖拉的方式上傳至 Colab。



下載模型權重

如果沒有訓練得到的 best.pt，可以透過程式下載 baseline 提供的模型權重檔。

```
[ ]  1 gdown.download("https://drive.google.com/uc?export=download&id=1cyV2eWcJYJcvWnc56L_zp1Lky3p9Yign","/content/best.pt")

Downloading...
From: https://drive.google.com/uc?export=download&id=1cyV2eWcJYJcvWnc56L_zp1Lky3p9Yign
To: /content/best.pt
100%|██████████| 5.51M/5.51M [00:00<00:00, 79.6MB/s]
'/content/best.pt'
```

進行預測

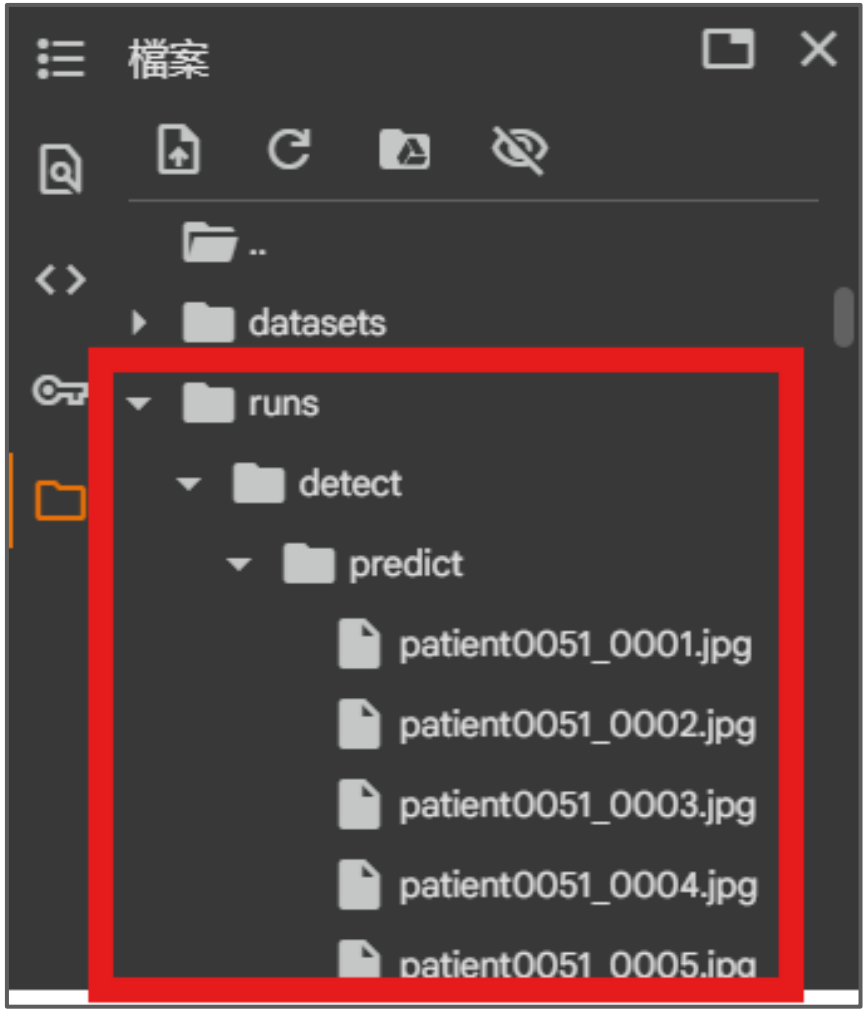
predict 檔案夾內有圖片代表預測完成

使用訓練過模型進行預測(執行時間3分鐘)

[9] 3分鐘

```
from ultralytics import YOLO

model = YOLO('/content/best.pt')
results = model.predict(source="./datasets/test/images1/",
                        save=True,
                        imgsz=640,
                        device=0
                        )
```

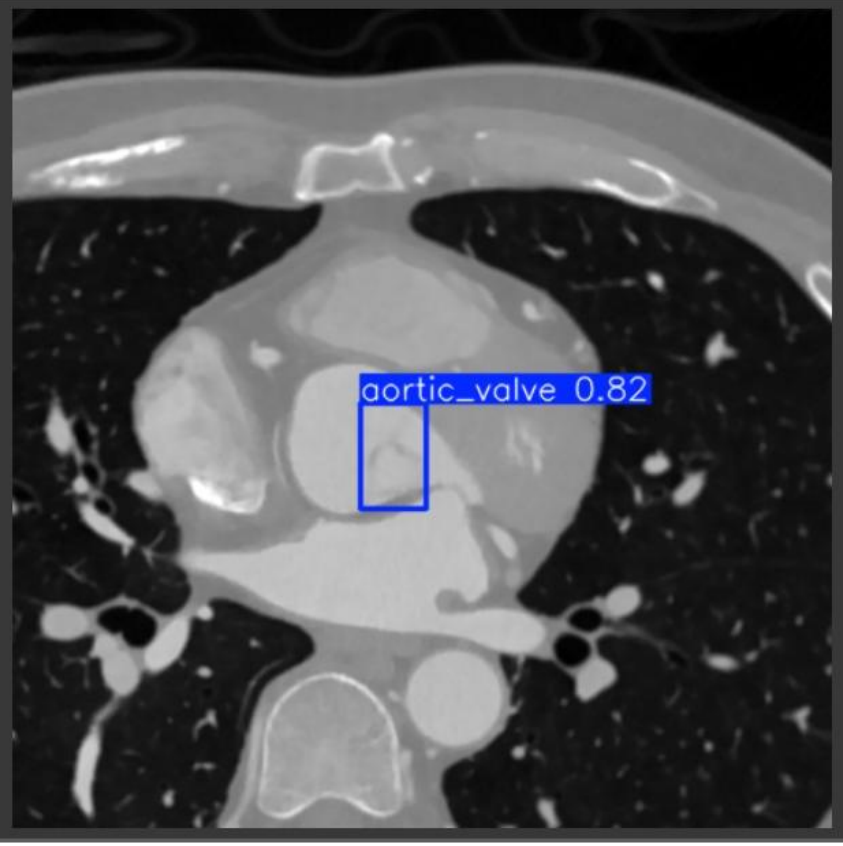
```
image 8215/8310 /content/datasets/test/images1/patient0076_0091.png: 640x640 (no detections), 21.9ms
image 8216/8310 /content/datasets/test/images1/patient0076_0092.png: 640x640 (no detections), 23.6ms
image 8217/8310 /content/datasets/test/images1/patient0076_0093.png: 640x640 (no detections), 19.9ms
image 8218/8310 /content/datasets/test/images1/patient0076_0094.png: 640x640 (no detections), 16.5ms
image 8219/8310 /content/datasets/test/images1/patient0076_0095.png: 640x640 (no detections), 16.9ms
image 8220/8310 /content/datasets/test/images1/patient0076_0096.png: 640x640 (no detections), 17.2ms
image 8221/8310 /content/datasets/test/images1/patient0076_0097.png: 640x640 (no detections), 16.8ms
image 8222/8310 /content/datasets/test/images1/patient0076_0098.png: 640x640 (no detections), 16.2ms
image 8223/8310 /content/datasets/test/images1/patient0076_0099.png: 640x640 (no detections), 16.2ms
image 8224/8310 /content/datasets/test/images1/patient0076_0100.png: 640x640 (no detections), 16.1ms
```



預測完成

其中一張預測結果

```
1 from IPython.display import Image
2
3 Image(filename='/content/runs/detect/predict/patient0051_0260.jpg', height=600)
```



預測數量及資訊

取得預測數量、類別、信心分數和座標，寫出提交的 .txt 檔會用到。
想取得更多資訊可參考官方說明：https://docs.ultralytics.com/zh/modes/predict/#boxes

預測數量

```
[11]
✓ 0 秒
1 print(len(results))
```

```
8310
```

取得預測結果的資訊

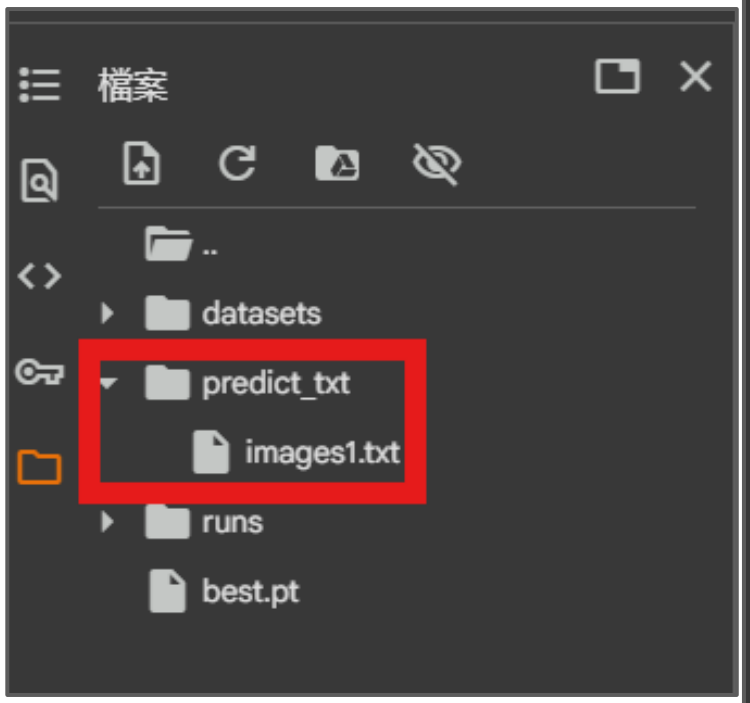
想取得更多資訊可參考官方說明 https://docs.ultralytics.com/zh/modes/predict/#boxes

```
[12]
✓ 0 秒
1 print('預測類別  :  ',results[260].boxes.cls[0].item())
2 print('預測信心分數  :  ',results[260].boxes.conf[0].item())
3 print('預測框座標  :  ',results[260].boxes.xyxy[0].tolist())
```

```
預測類別 :  0.0
預測信心分數 :  0.8218923211097717
預測框座標 :  [216.84521484375, 246.9698486328125, 259.8663635253906, 312.4429931640625]
```

將偵測框數值寫入.txt檔

執行成功會在左側的檔案看到 predict_txt 檔案夾內有 images1.txt。




```
[13]
0
秒
```

```python
!mkdir ./predict_txt/
output_file = open('./predict_txt/images1.txt', 'w')
for i in range(len(results)):
    # 取得圖片檔名（不含副檔名）
    filename = results[i].path.split('/')[-1].split('.png')[0]

    # 取得預測框數量
    boxes = results[i].boxes
    box_num = len(boxes.cls.tolist())

    # 如果有預測框
    if box_num > 0:
        for j in range(box_num):
            # 提取資訊
            label = int(boxes.cls[j].item())    # 類別
            conf = boxes.conf[j].item()         # 信心度
            x1, y1, x2, y2 = boxes.xyxy[j].tolist()    # 邊界框座標

            # 建立一行資料
            line = f"{filename} {label} {conf:.4f} {int(x1)} {int(y1)} {int(x2)} {int(y2)}\n"
            output_file.write(line)

# 關閉輸出檔案
output_file.close()

```

釋放RAM

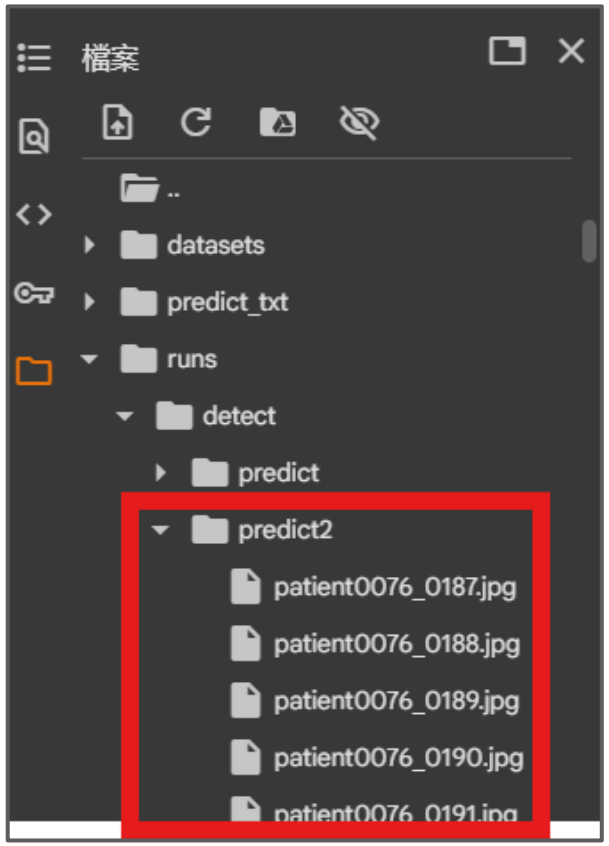
因為Colab RAM不會自動釋放，所以透過程式釋放。

```
[ ]
```

```python
import  torch  ,gc

#  刪除大型變數
del  boxes,all_files,results
gc.collect()
torch.cuda.empty_cache()
```

預測後半圖片

predict2 檔案夾內有圖片代表預測完成。



預測後半圖片(執行時間3分鐘)

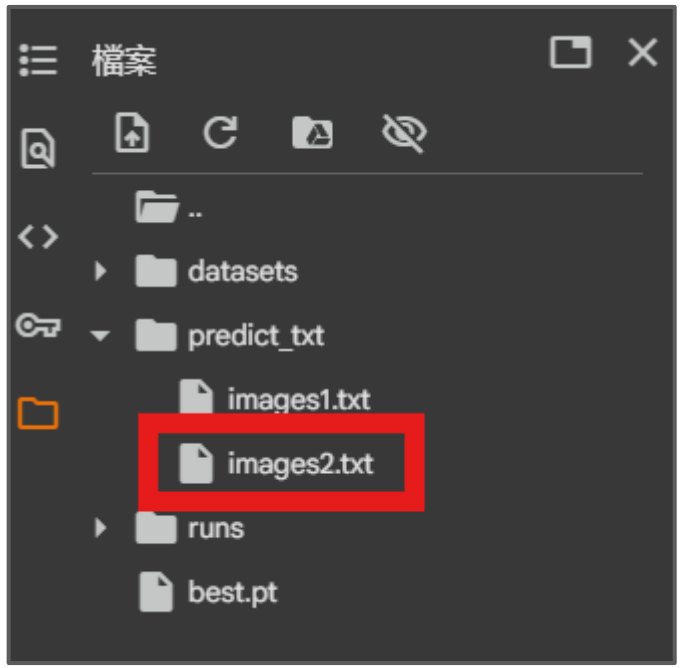
[15] ✓ 3分鐘

```
1 from ultralytics import YOLO
2
3 model = YOLO('/content/best.pt')
4 results = model.predict(source="./datasets/test/images2/",
5                         save=True,
6                         imgsz=640,
7                         device=0
8                         )
```

```
image 8255/8310 /content/datasets/test/images2/patient0100_0301.png: 640x640 (no detections), 16.9ms
image 8256/8310 /content/datasets/test/images2/patient0100_0302.png: 640x640 (no detections), 15.8ms
image 8257/8310 /content/datasets/test/images2/patient0100_0303.png: 640x640 (no detections), 15.7ms
image 8258/8310 /content/datasets/test/images2/patient0100_0304.png: 640x640 (no detections), 18.2ms
image 8259/8310 /content/datasets/test/images2/patient0100_0305.png: 640x640 (no detections), 21.7ms
image 8260/8310 /content/datasets/test/images2/patient0100_0306.png: 640x640 (no detections), 16.5ms
image 8261/8310 /content/datasets/test/images2/patient0100_0307.png: 640x640 (no detections), 15.8ms
```

將後半偵測框數值寫進.txt檔

執行成功會在左側的檔案看到 predict_txt 檔案夾內有 images2.txt。

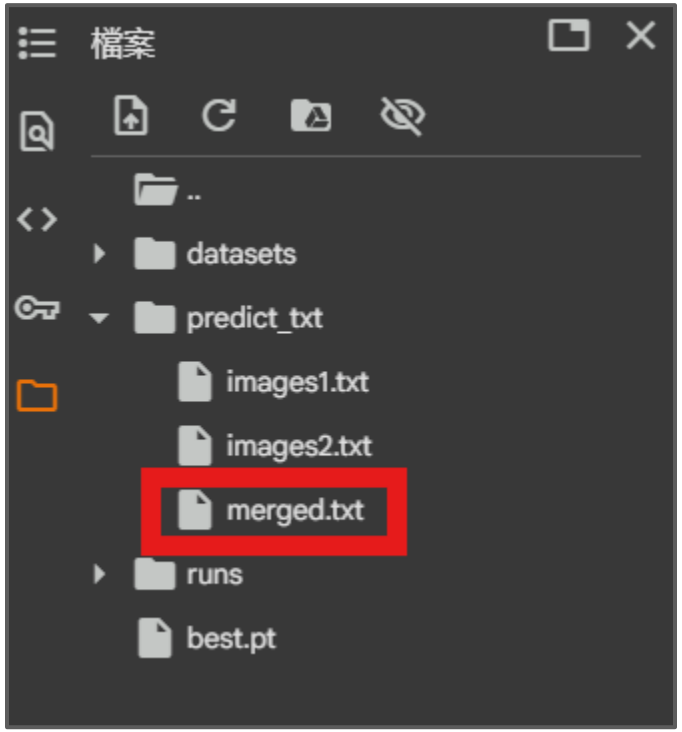


[16] 0 秒

```python
output_file = open('./predict_txt/images2.txt', 'w')
for i in range(len(results)):
    # 取得圖片檔名（不含副檔名）
    filename = results[i].path.split('/')[-1].split('.png')[0]

    # 取得預測框數量
    boxes = results[i].boxes
    box_num = len(boxes.cls.tolist())

    # 如果有預測框
    if box_num > 0:
        for j in range(box_num):
            # 提取資訊
            label = int(boxes.cls[j].item())    # 類別
            conf = boxes.conf[j].item()         # 信心度
            x1, y1, x2, y2 = boxes.xyxy[j].tolist()    # 邊界框座標

            # 建立一行資料
            line = f"{filename} {label} {conf:.4f} {int(x1)} {int(y1)} {int(x2)} {int(y2)}\n"
            output_file.write(line)

# 關閉輸出檔案
output_file.close()

```

合併兩個.txt

執行成功會在左側的檔案看到 predict_txt 檔案夾內有 merged.txt。



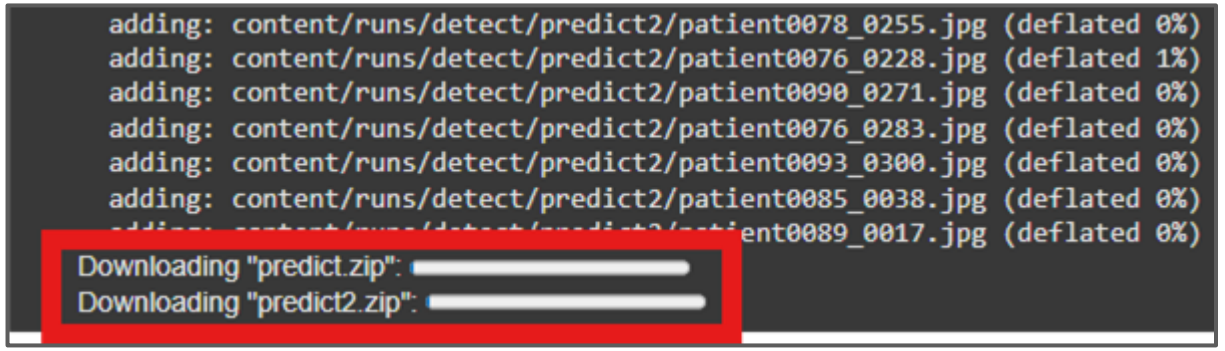
[17]
0
秒

```
file1 = "./predict_txt/images1.txt"
file2 = "./predict_txt/images2.txt"
output = "./predict_txt/merged.txt"

with open(output, "w", encoding="utf-8") as fout:
    for f in [file1, file2]:
        if os.path.exists(f):
            with open(f, "r", encoding="utf-8") as fin:
                fout.writelines(fin.readlines())

print(f"合併完成 -> {output}")

```

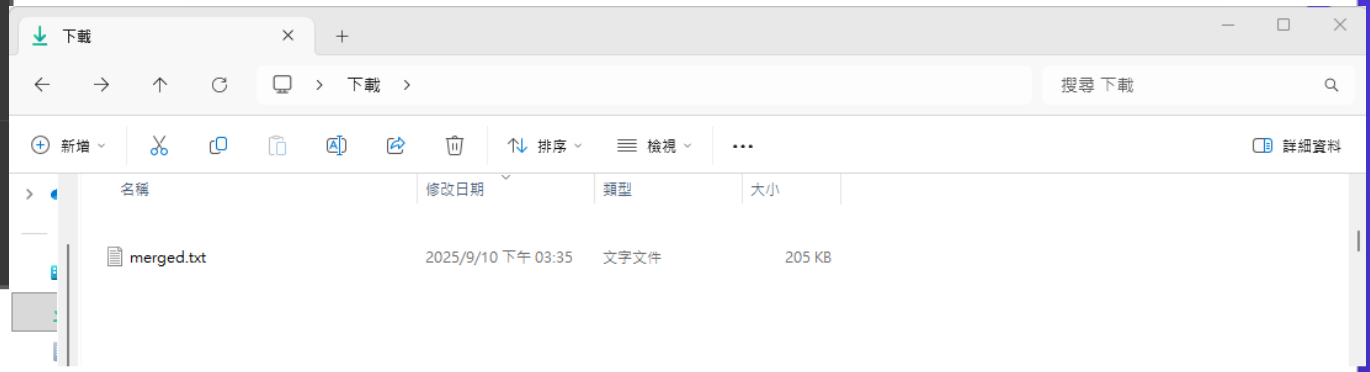
```
合併完成 -> ./predict_txt/merged.txt
```

下載競賽提交檔案

- 下載競賽提交 .txt檔及供參賽者參考預測圖片(**圖片不須上傳**至競賽網頁)。
- 下載完成會在本地端有 merged.txt 檔案。



▲ 等待進度條消失才會完成下載

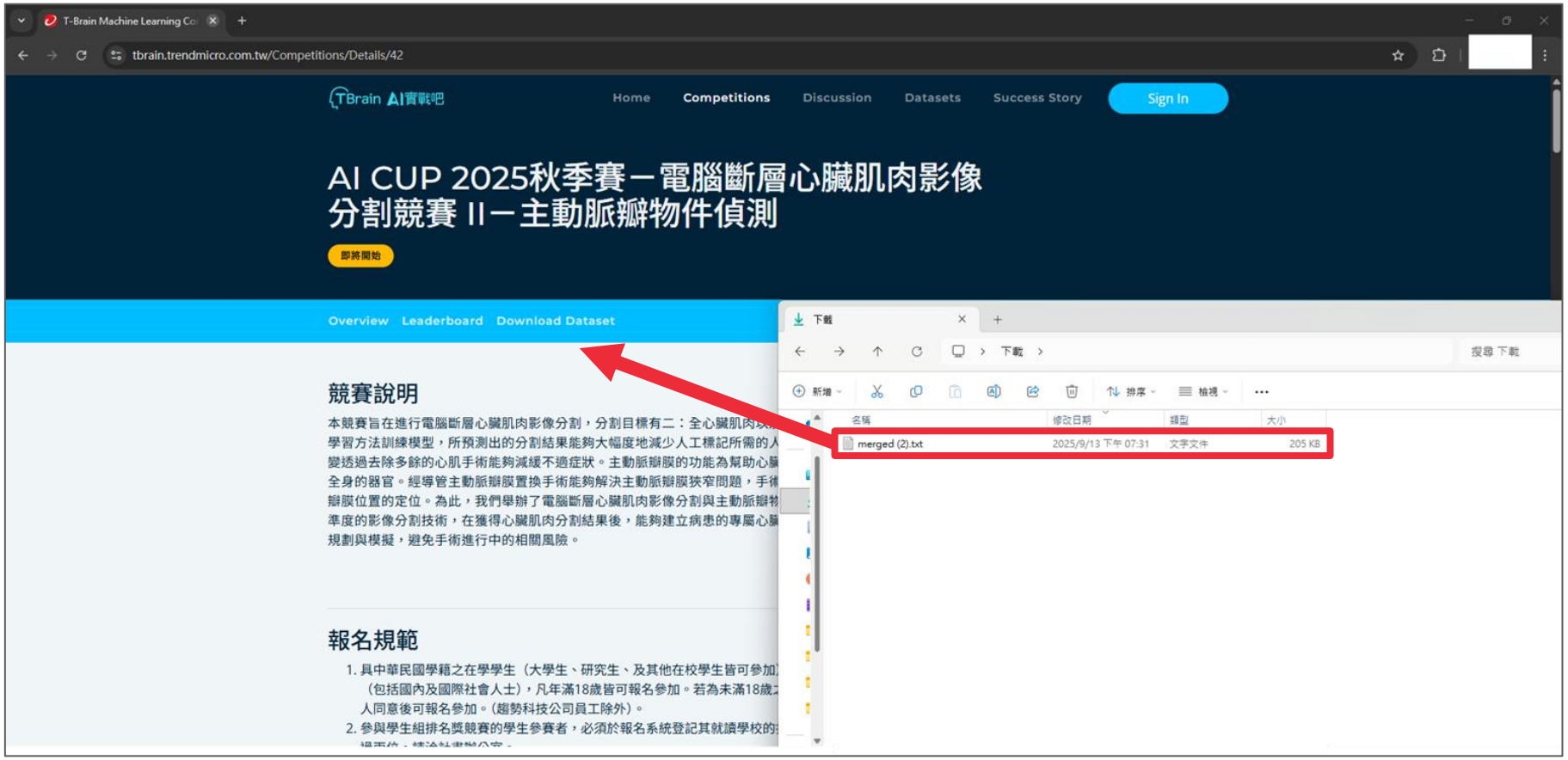


05

預測成果上傳至競賽網頁

提交檔案

將 merged.txt 檔案提交至趨勢科技 Tbrain 競賽網站。



相關連結

1. AICUP網站 : https://www.aicup.tw/

1. 趨勢科技競賽網站 : https://tbrain.trendmicro.com.tw/

1. 提問表單 : https://forms.gle/fw3Kcc3W3Ct7WJWM6

END