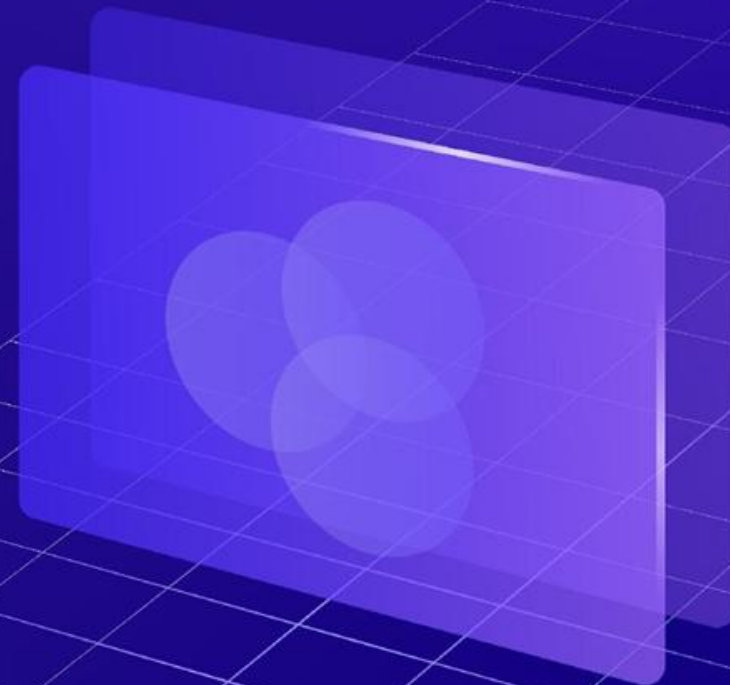


AI CUP 2025 秋季賽

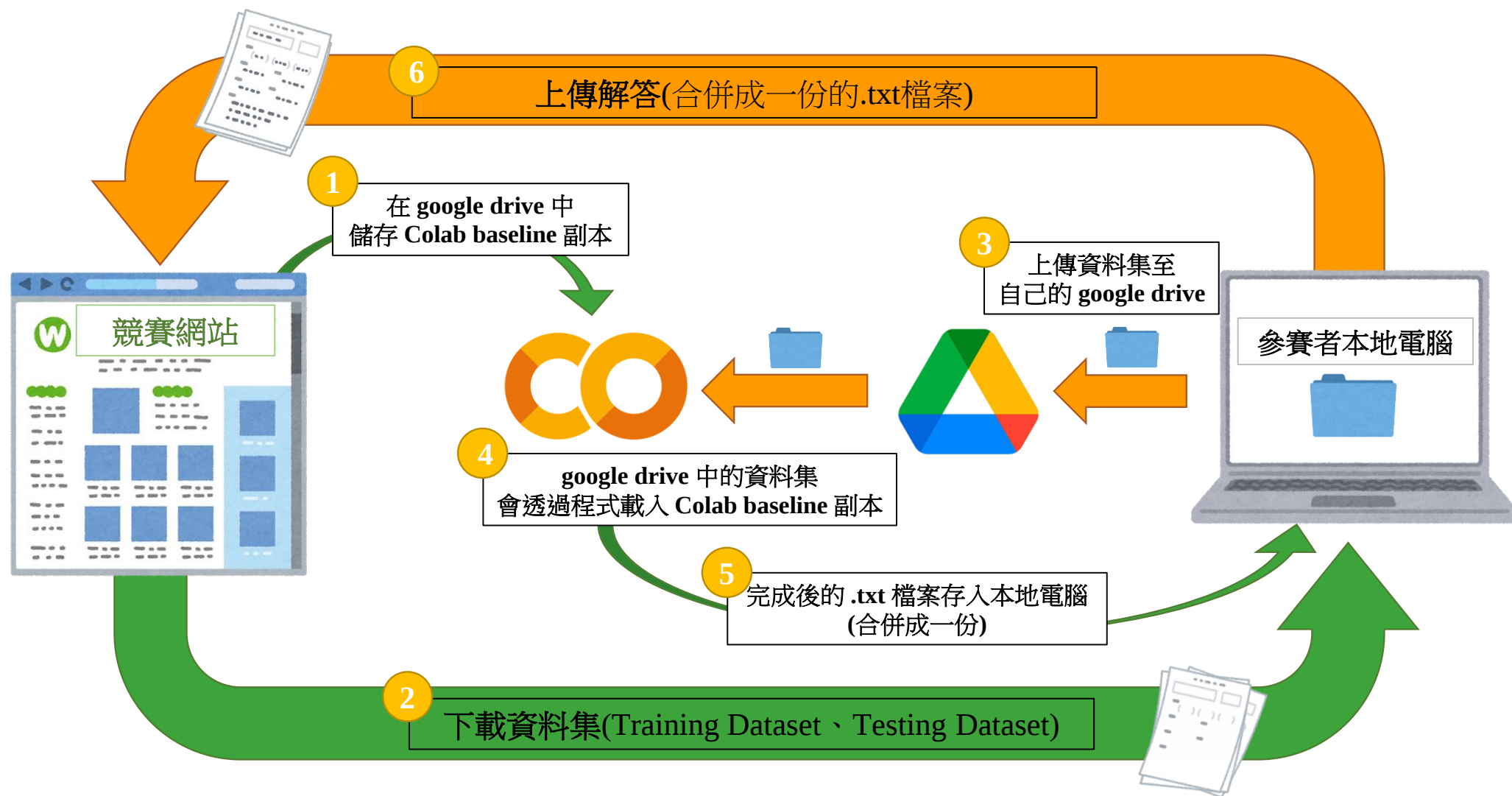
電腦斷層心臟肌肉影像分割競賽 II
主動脈瓣物件偵測



目錄

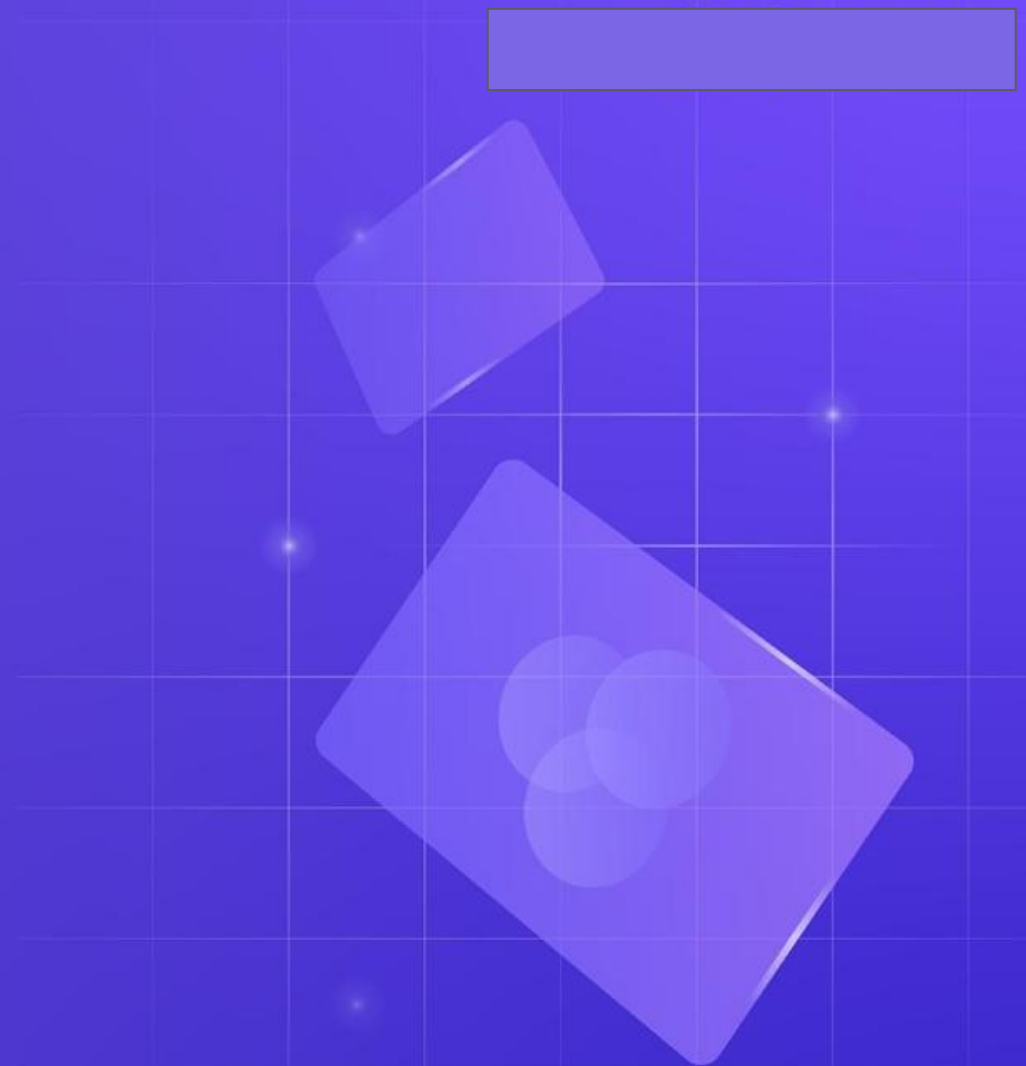
1. 資料集說明
2. 繳交檔案格式說明
3. Colab Baseline 程式說明 (train)
4. Colab Baseline 程式說明 (predict)
5. 預測成果上傳至競賽網頁

使用 Baseline 程式參加競賽流程圖



01

資料集説明



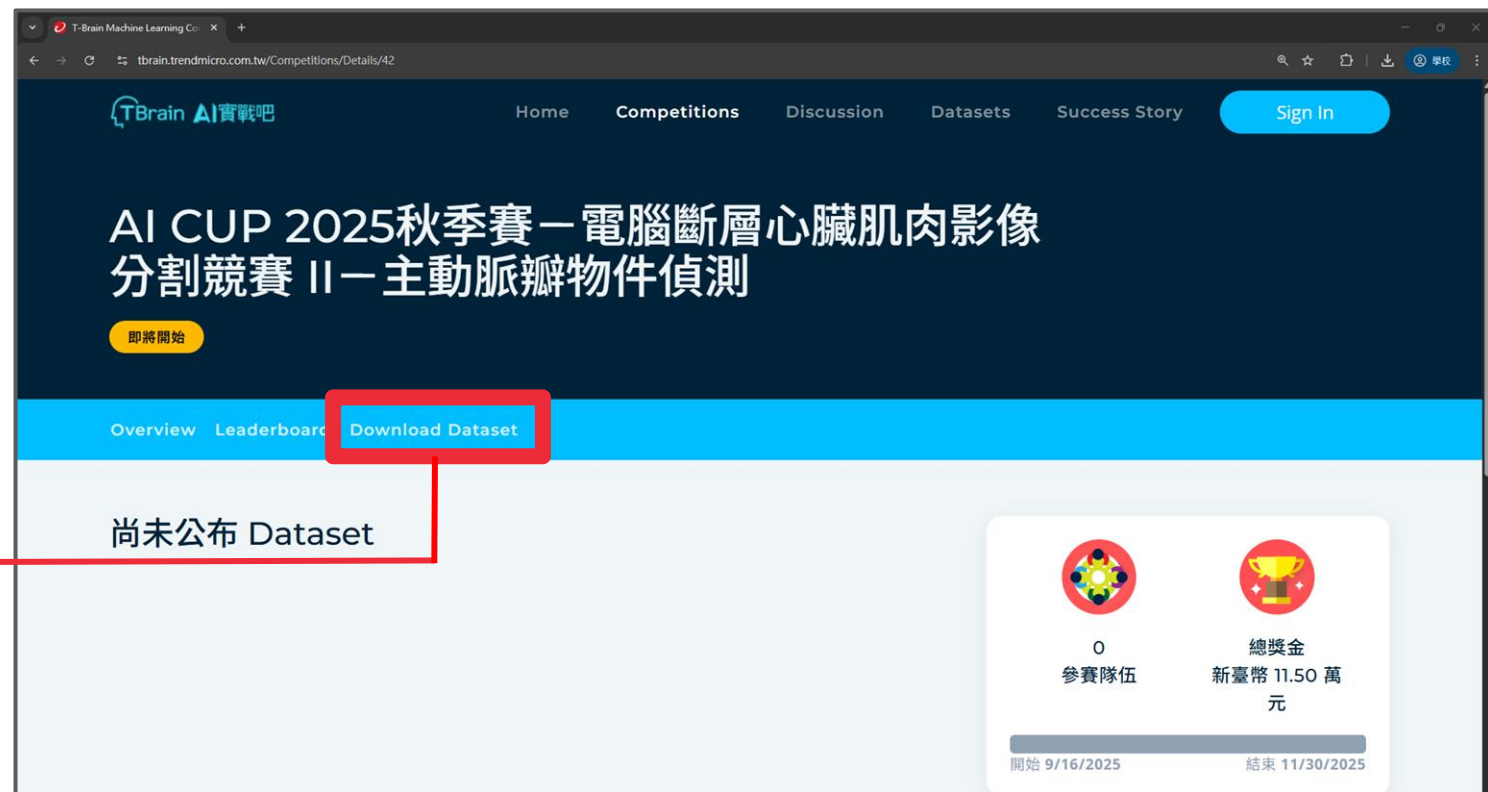
下載競賽資料集



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

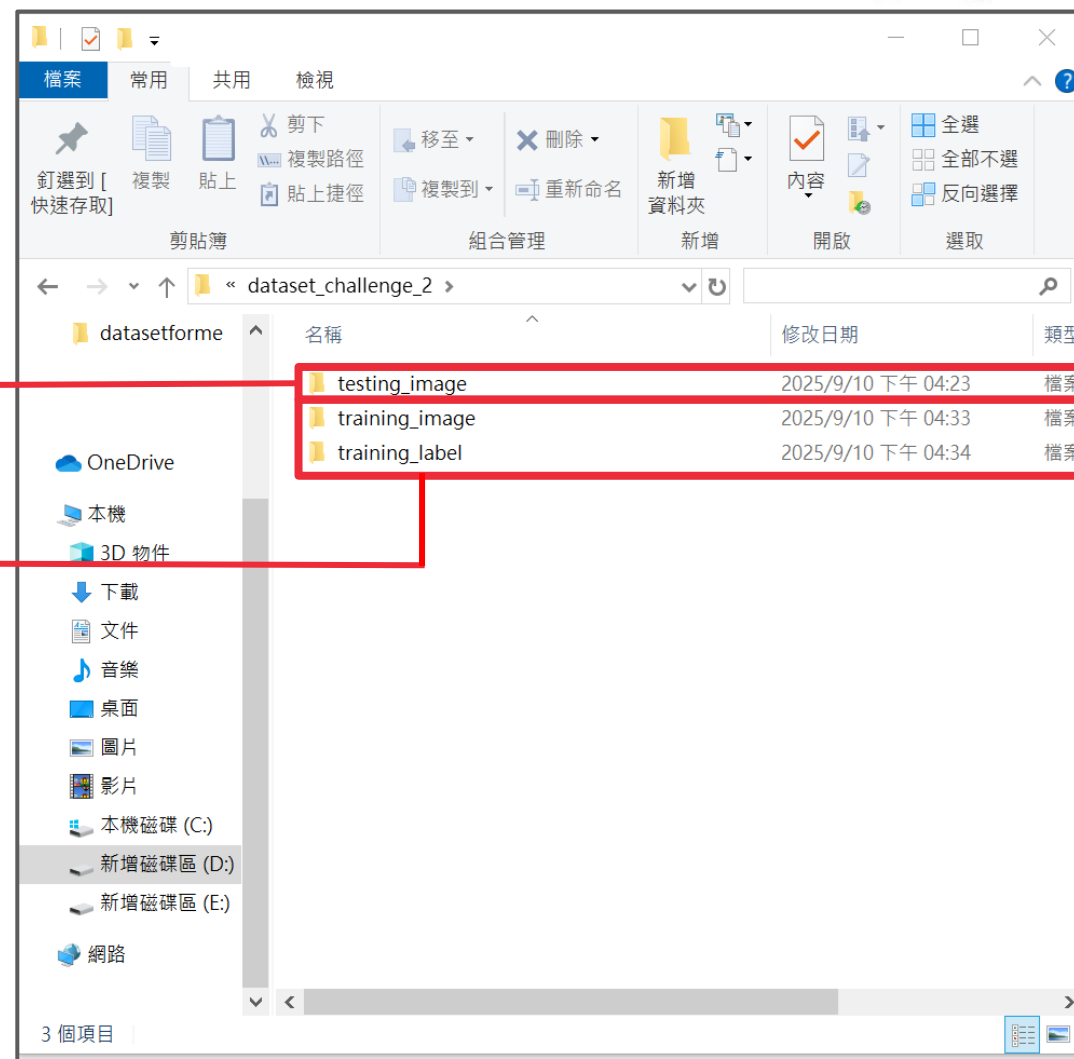


本次競賽試題！

Testing Dataset

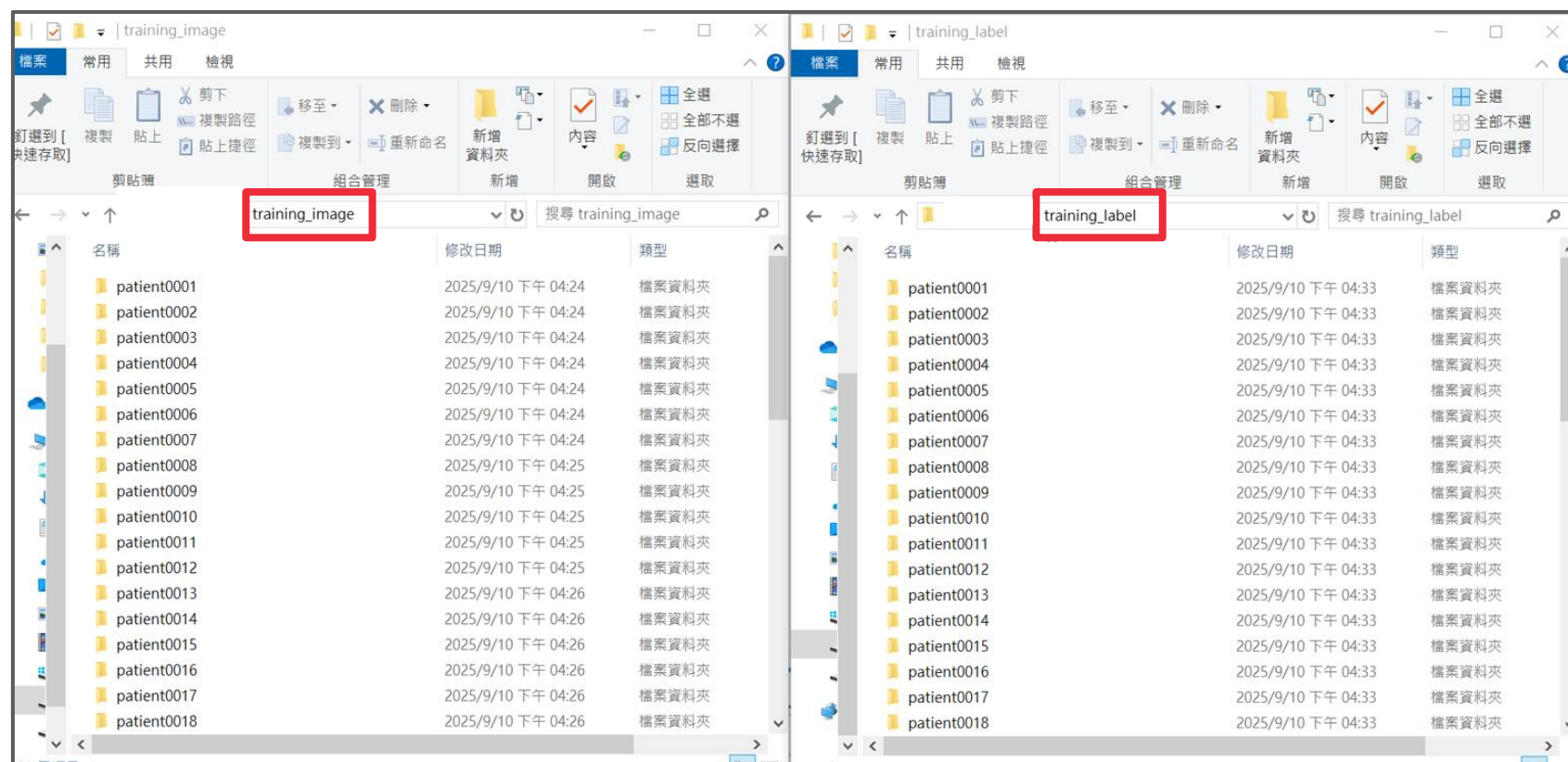
Training Dataset

▶ 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)。

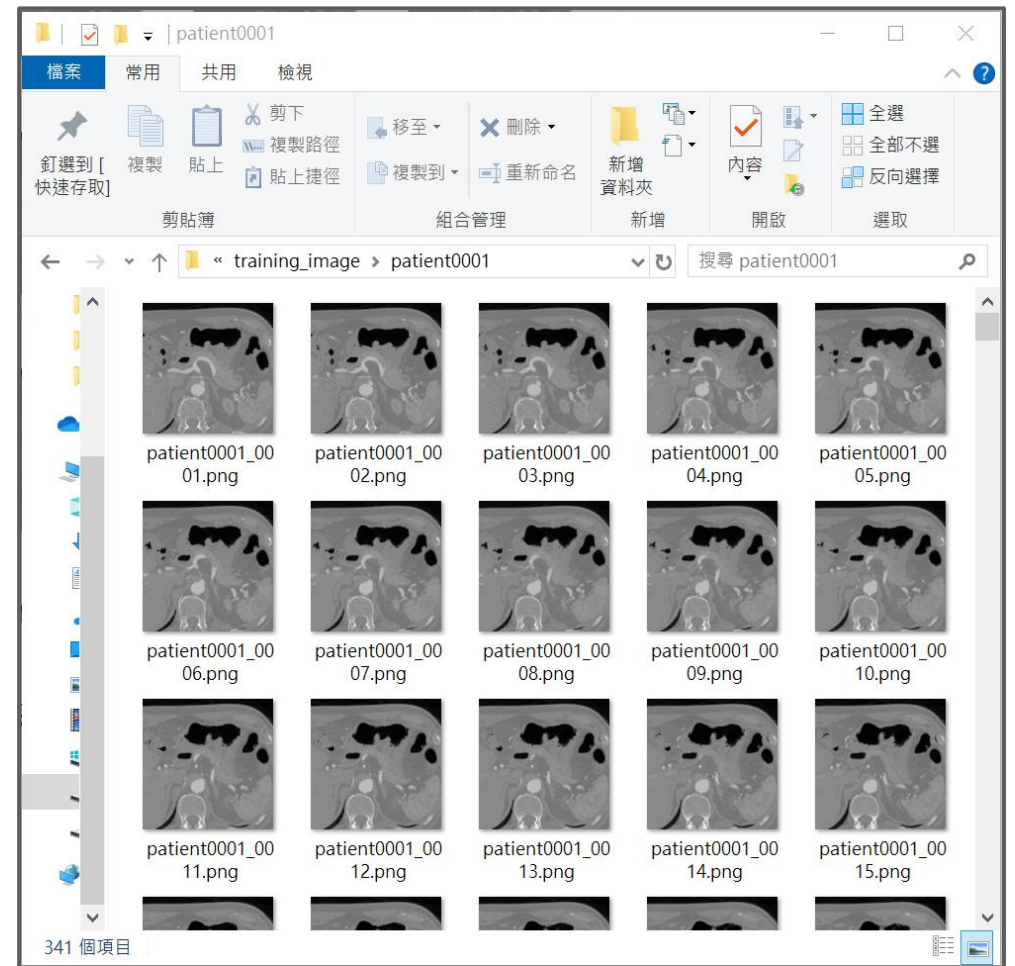


► training_image 和 training_label 檔案夾

training_image 為電腦斷層掃描

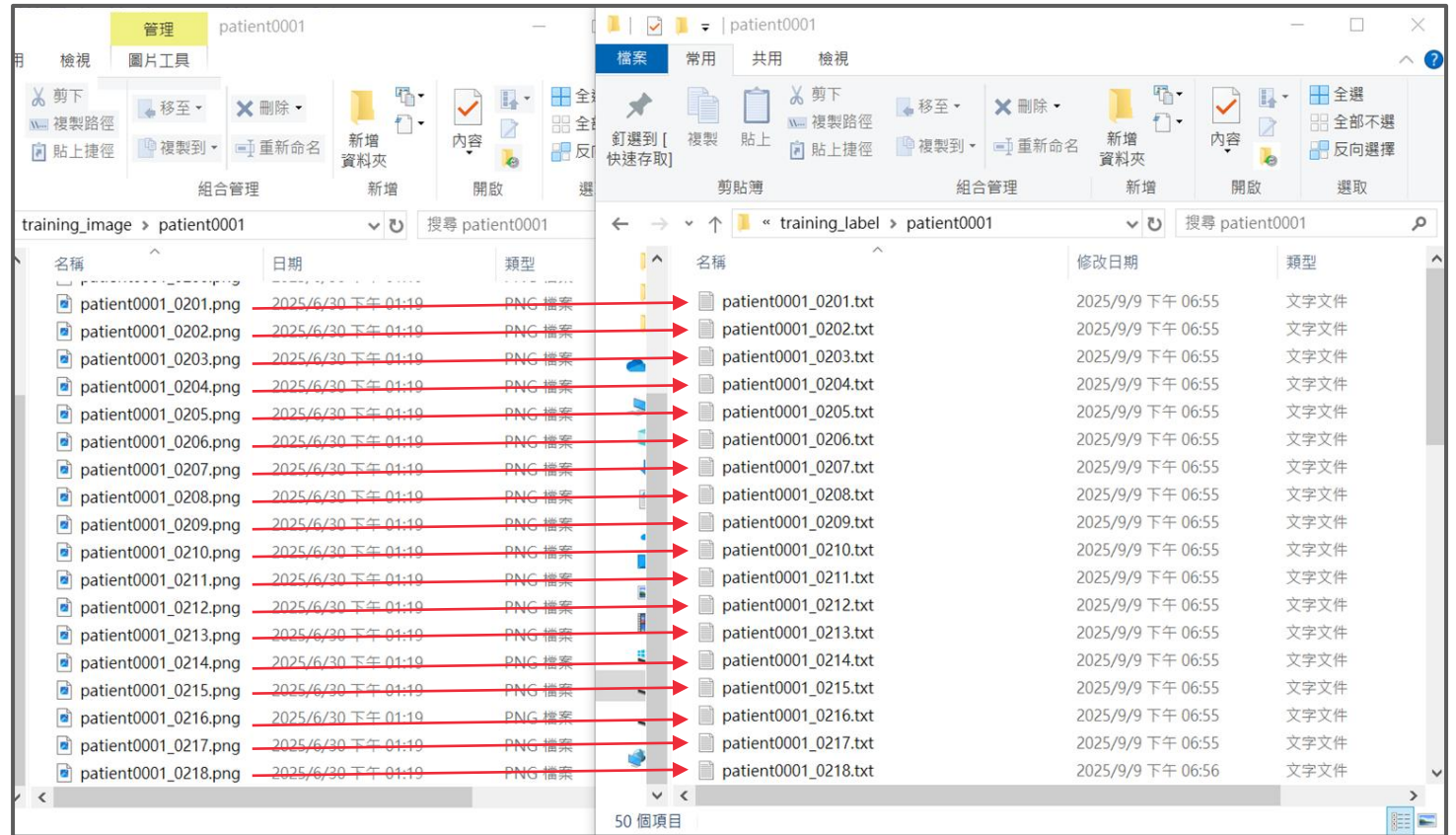
- training_image 檔案夾中
 - 每個 patient 檔案夾包含有 250~450 張不等之 PNG 圖檔。
 - 每張 PNG 圖檔尺寸為 512 * 512 pixel。

▶ 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。

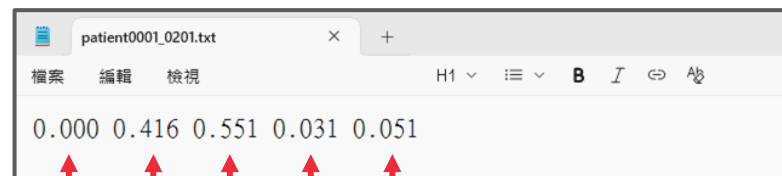


▲ training_label 檔案夾中包含 .txt 標註檔

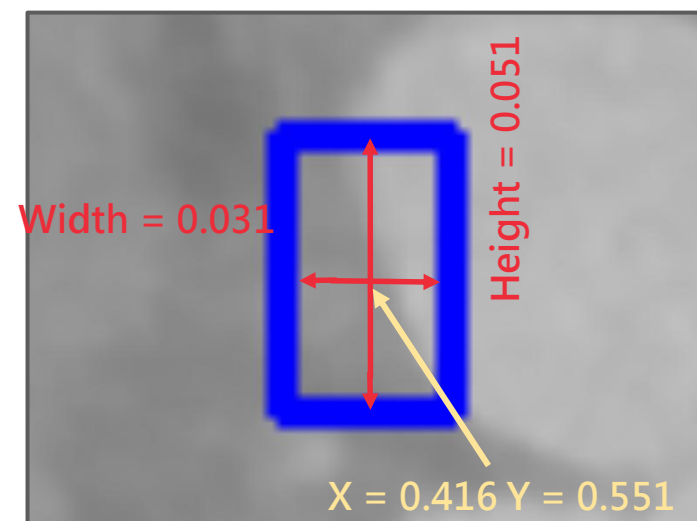
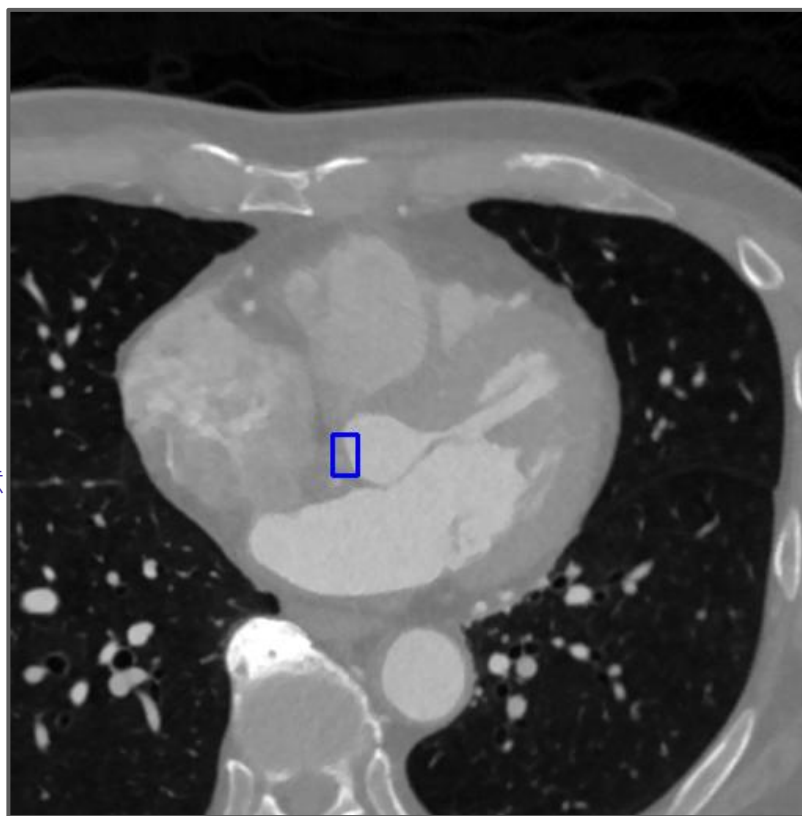
標註檔 .txt 中的各項數值(正規化座標)

- 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。



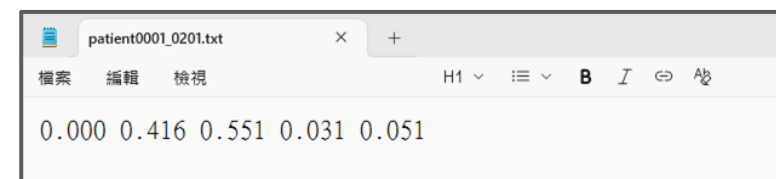
► 使用 LabelMe 進行標註

```
1 {
2   "version": "5.4.0",
3   "flags": {},
4   "shapes": [
5     {
6       "label": "ac",
7       "points": [
8         [
9           205.21465968586386,
10          269.35078534031413
11        ],
12        [
13          221.18324607329842,
14          295.0052356020943
15        ]
16      ],
17      "group_id": null,
18      "description": "",
19      "shape_type": "rectangle",
20      "flags": {},
21      "mask": null
22    }
23  ],
24  "imagePath": "IMG0201.png",
25  "imageData": "iVBORw0KGgoAAAANSU...EUGAAAgA...",
26  "imageHeight": 512,
27  "imageWidth": 512
28 }
```

▲ 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:

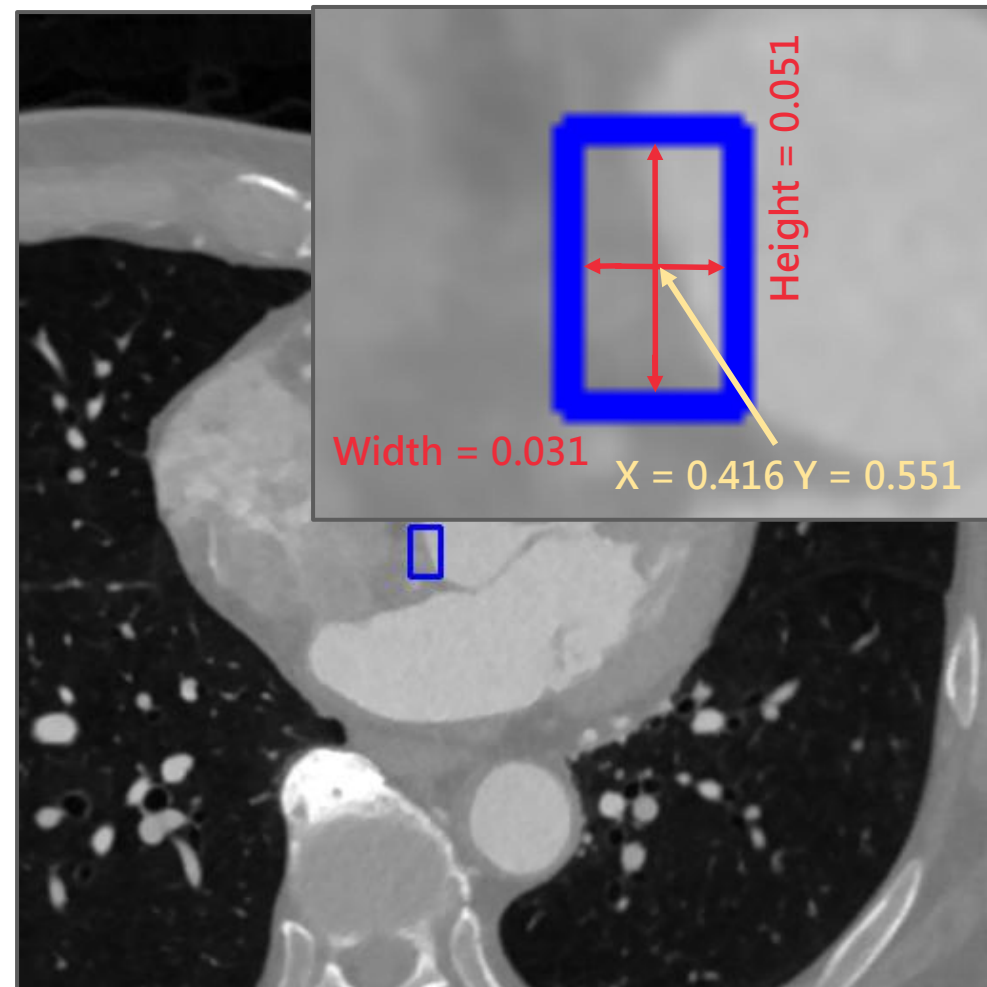
若 $x_min = 205$ $y_min = 269$ $x_max = 221$ $y_max = 295$,
則原始座標為

$$xc = ((205 + 221) / 2) / 512 = 0.416$$

$$yc = ((269 + 295) / 2) / 512 = 0.551$$

$$w = (221 - 205) / 512 = 0.031$$

$$h = (295 - 269) / 512 = 0.051$$



▲ 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。

$x_min = \text{round}((xc - w / 2) * width)$

$x_max = \text{round}((xc + w / 2) * width)$

$y_min = \text{round}((yc - h / 2) * height)$

$y_max = \text{round}((yc + h / 2) * height)$

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 ,

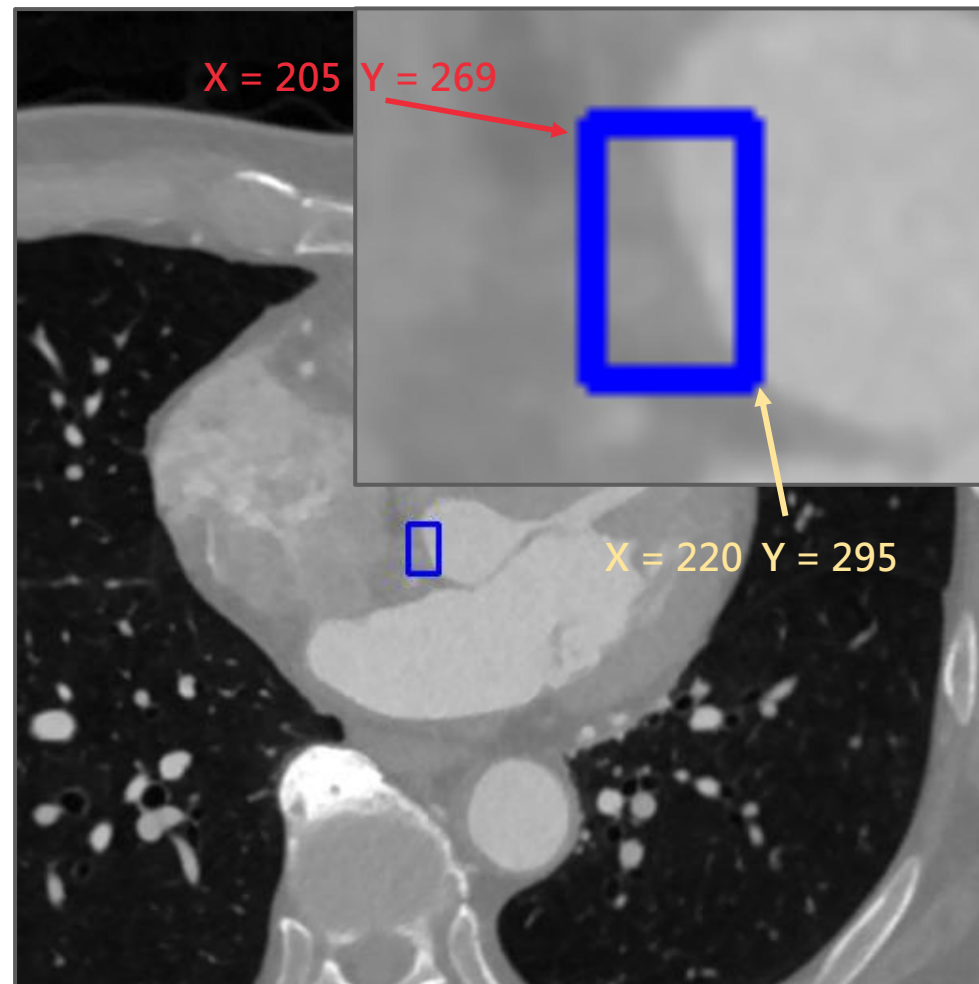
則原始座標為

$x_min = \text{round}((0.416 - 0.031 / 2) * 512) = 205$

$x_max = \text{round}((0.416 + 0.031 / 2) * 512) = 221$

$y_min = \text{round}((0.551 - 0.051 / 2) * 512) = 269$

$y_max = \text{round}((0.551 + 0.051 / 2) * 512) = 295$



▲ 還原後的點座標

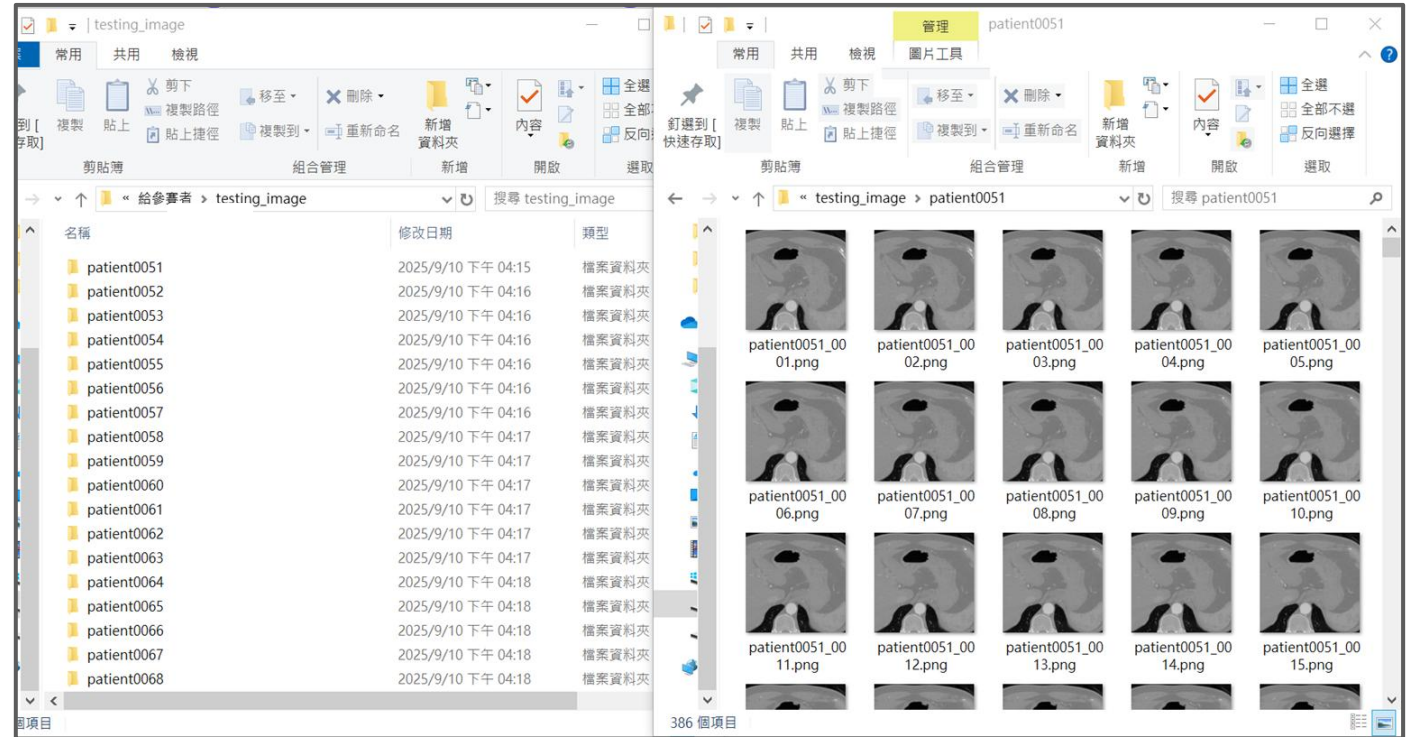
測試集組成(50筆電腦斷層掃描)

- 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

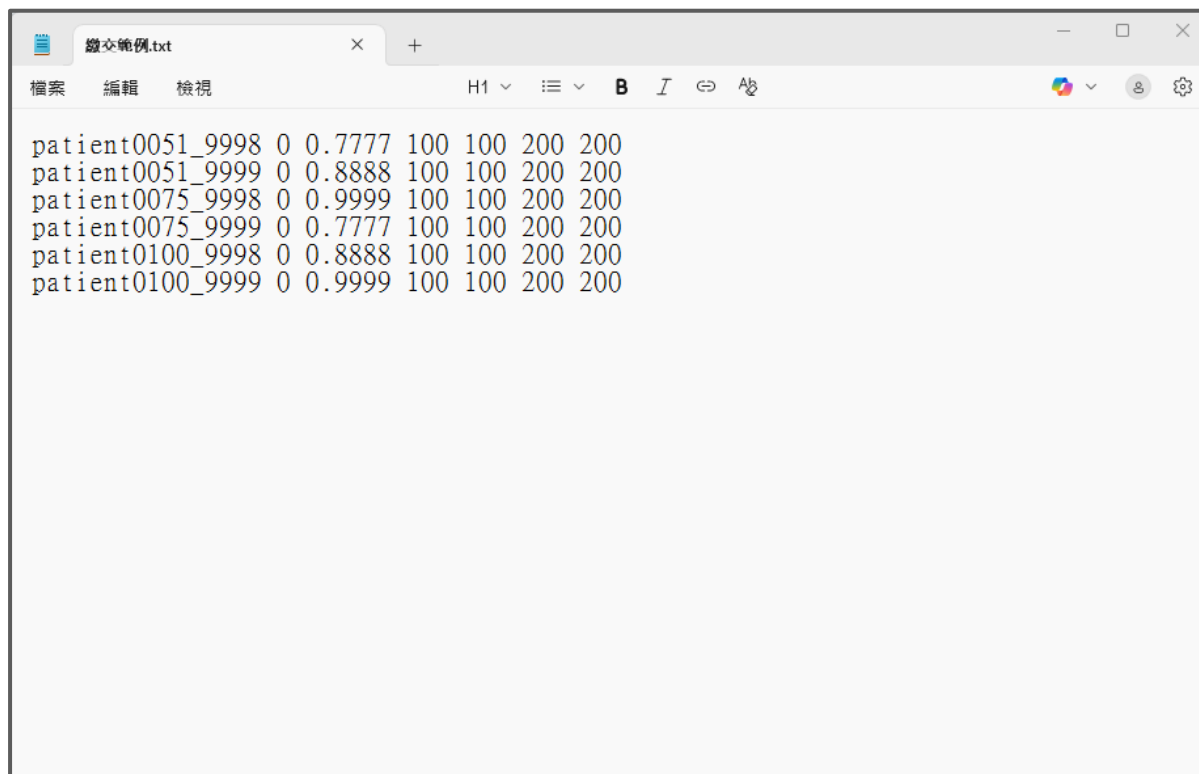
02

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/1LNq_jEYqgNvNbH-1aZRFsOZX_4YJd3T0/view?usp=sharing)。



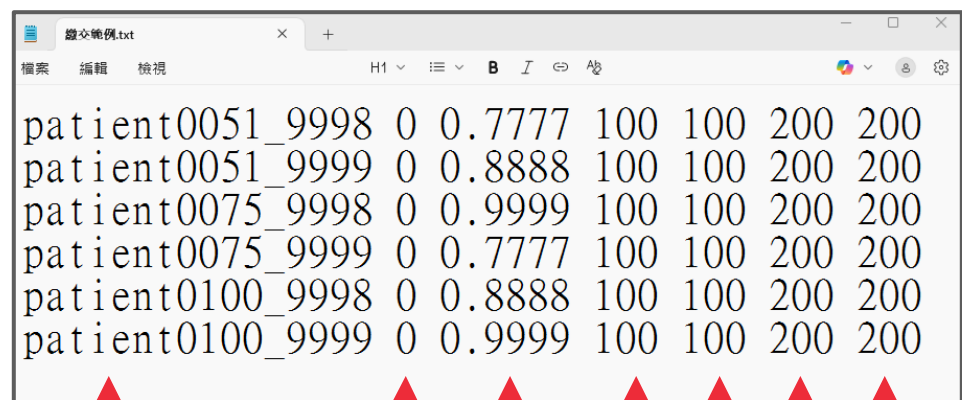
```
patient0051_9998 0 0.7777 100 100 200 200
patient0051_9999 0 0.8888 100 100 200 200
patient0075_9998 0 0.9999 100 100 200 200
patient0075_9999 0 0.7777 100 100 200 200
patient0100_9998 0 0.8888 100 100 200 200
patient0100_9999 0 0.9999 100 100 200 200
```

▲ 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)



```
patient0051_9998 0 0.7777 100 100 200 200
patient0051_9999 0 0.8888 100 100 200 200
patient0075_9998 0 0.9999 100 100 200 200
patient0075_9999 0 0.7777 100 100 200 200
patient0100_9998 0 0.8888 100 100 200 200
patient0100_9999 0 0.9999 100 100 200 200
```

圖片名稱

信心分數 左上y 右下y

類別 左上x 右下x

▲ 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

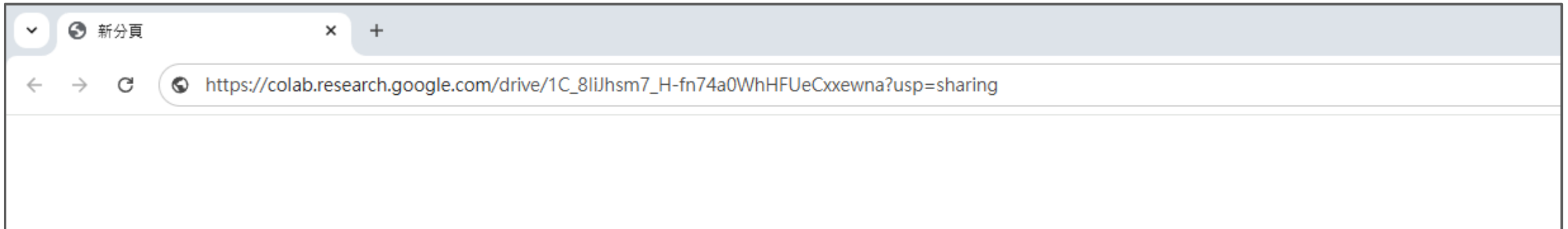
Colab Baseline 程式說明 (train)

打開瀏覽器



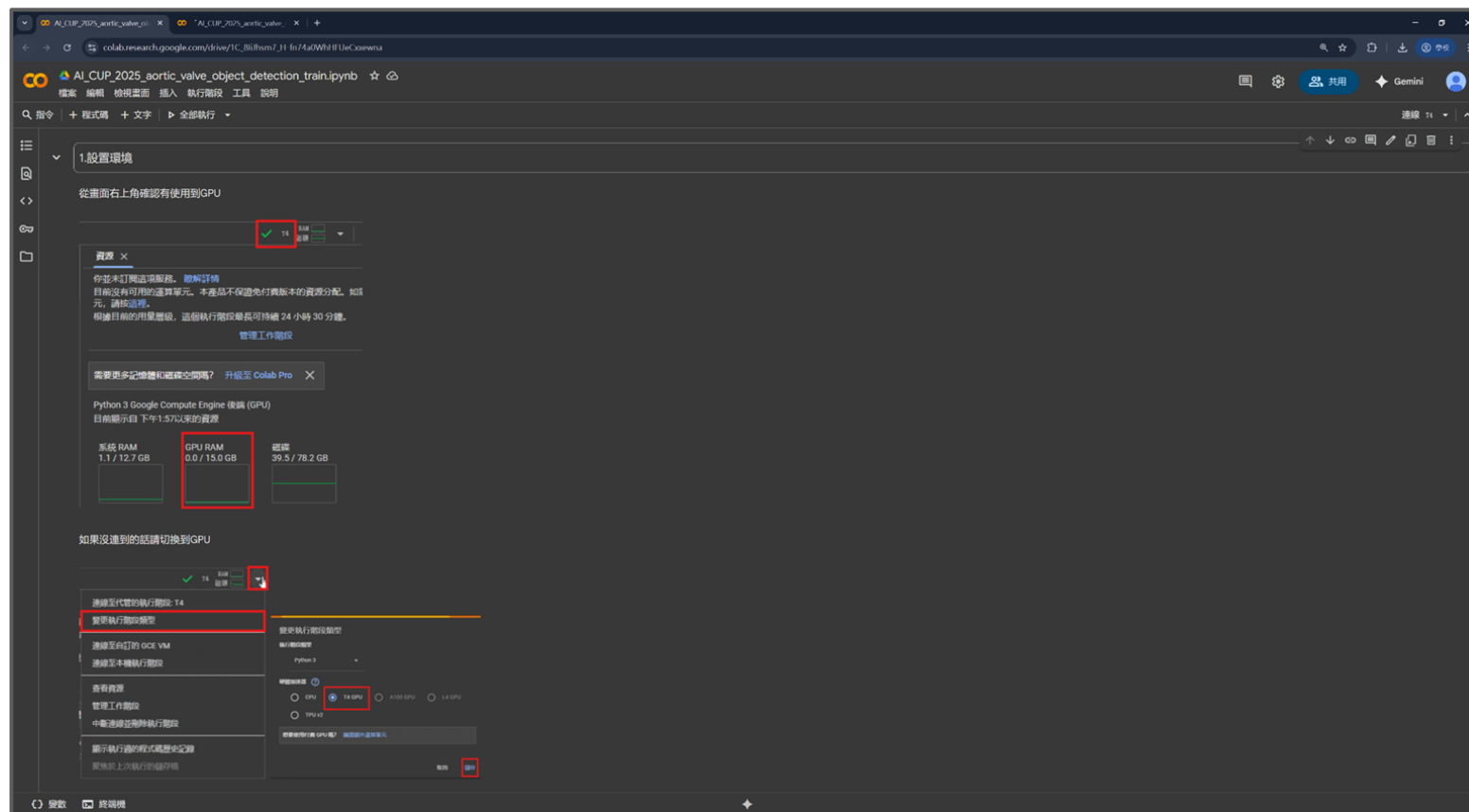
新興智慧顯示科技
教育聯盟

Baseline 會透過瀏覽器打開 Colab 完成模型訓練。



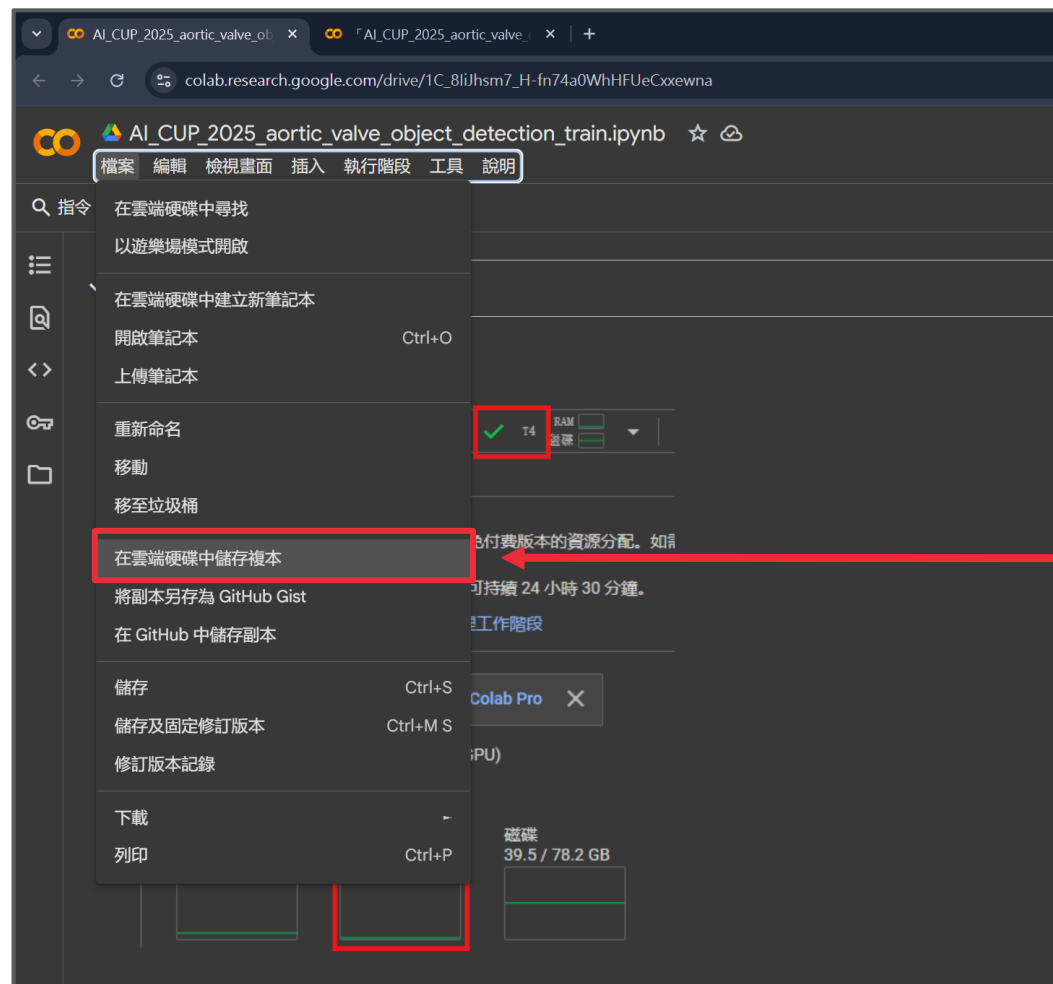
開啟 Colab (train)

Colab連結 : https://colab.research.google.com/drive/1C_8liJhsm7_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選查看資源是否有 GPU RAM。

The screenshot displays the Google Colab interface for a notebook titled "AI_CUP_2025_aortic_valve_object_detection_train.ipynb". The top navigation bar includes icons for chat, settings, sharing, Gemini, and a user profile. Below the bar, a toolbar shows "指令" (Commands), "+ 程式碼" (+ Code), "+ 文字" (+ Text), and "全部執行" (Run All). The left sidebar contains icons for file explorer, search, and other functions.

The main content area is divided into sections. The first section, "1.設置環境" (1. Setup Environment), includes a prompt "從畫面右上角確認有使用到GPU" (Confirm GPU usage from the top right corner of the screen). Below this, a resource selection dropdown is shown with a green checkmark and "T4" selected, indicating GPU availability. A "資源" (Resources) panel is open, displaying a message: "你並未訂閱這項服務。瞭解詳情" (You are not subscribed to this service. Learn more details). It states that no compute units are currently available and that the execution stage can last up to 24 hours and 30 minutes. A button "管理工作階段" (Manage execution stage) is visible.

A second "資源" panel is also open, showing a similar message. Below it, a banner asks "需要更多記憶體和磁碟空間嗎？升級至 Colab Pro" (Need more memory and disk space? Upgrade to Colab Pro). The bottom section, "Python 3 Google Compute Engine 後端 (GPU)", shows resource usage for the current session starting at 1:57 PM. It includes three metrics: "系統 RAM" (System RAM) at 1.1 / 12.7 GB, "GPU RAM" at 0.0 / 15.0 GB (highlighted with a red box), and "磁碟" (Disk) at 39.5 / 78.2 GB. Each metric is accompanied by a progress bar.

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



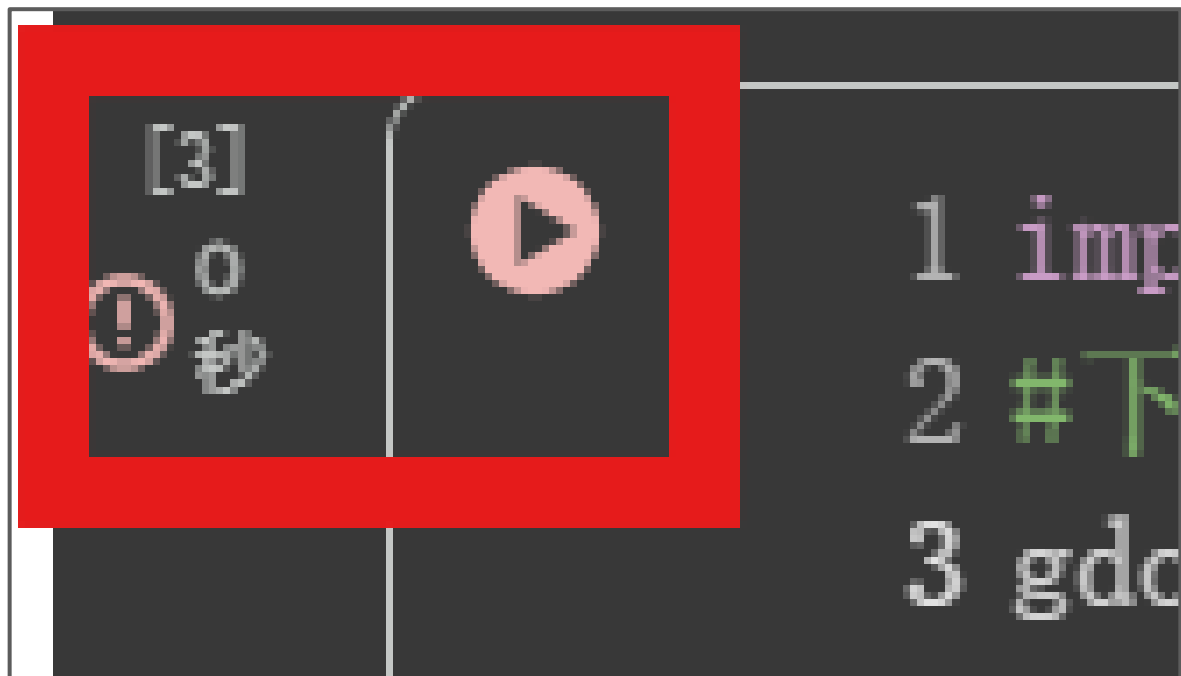
Colab運行情況(執行中)

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  |
+-----+-----+

Processes:
+-----+
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   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 X
competition > ! aortic_valve_colab.yaml
1 train: "./datasets/train/images"
2 val: "./datasets/val/images"
3
4 names:
5 0: aortic_valve
```

▲ aortic_valve_colab.yaml 內容

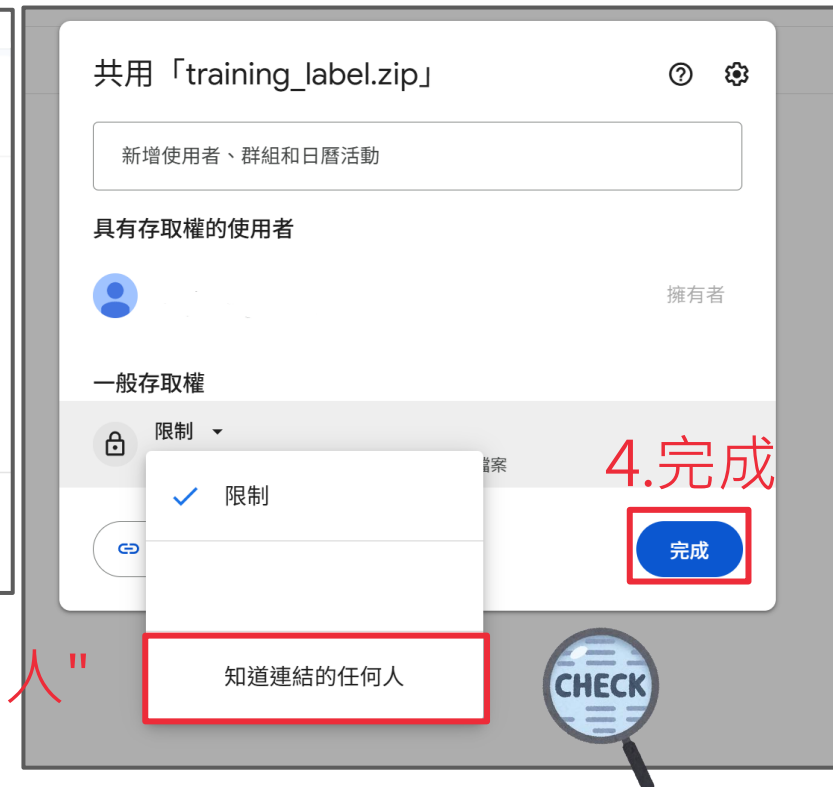


▲ 將檔案上傳至雲端

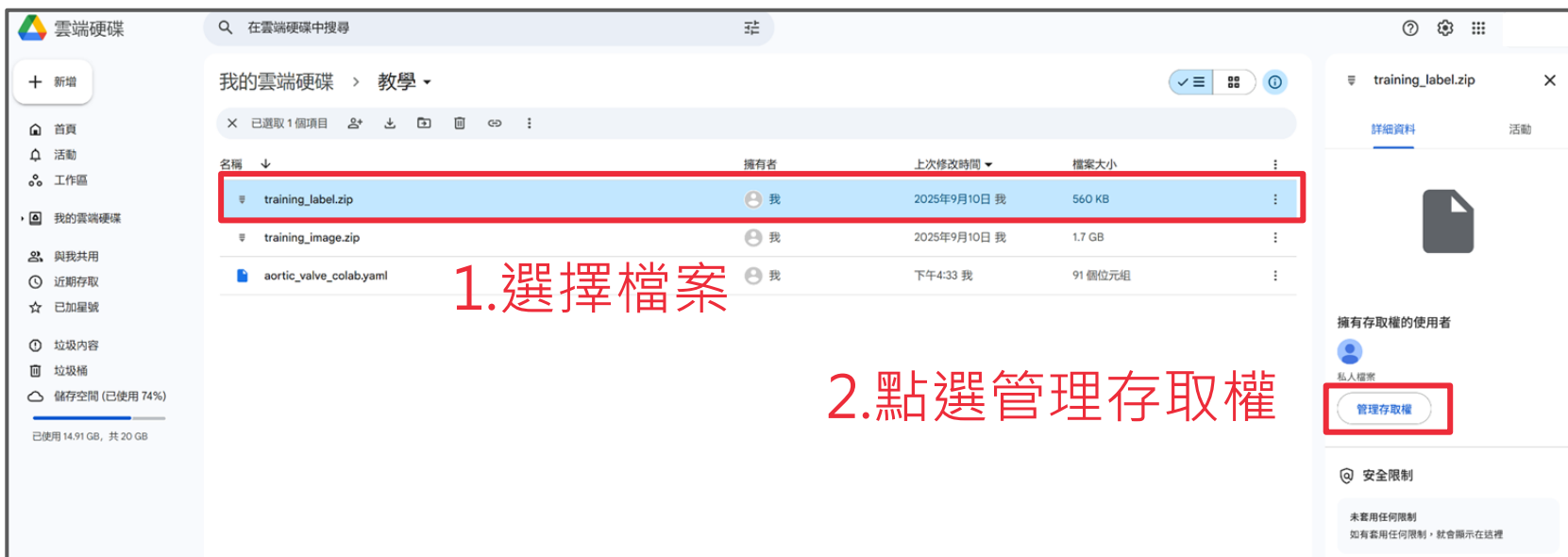
更改檔案權限

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>

The screenshot shows a web browser window with the URL `sites.google.com/view/twdrivefromdownload/繁體中文traditional-chinese?authuser=0`. The page title is "Google Drive Share Link to Di...". The main heading is "注意事項" (Notice). Below it are three numbered instructions in Chinese. The main content area is titled "Google 雲端硬碟直接下載連結轉換器" (Google Drive Direct Download Link Converter). It contains a text input field with the URL `https://drive.google.com/file/d/1fEEG5UIGlvv68a84UhR6K3Zl_0Gi1JCP/view?usp=sharing`, a button labeled "產生直接下載連結" (Generate Direct Download Link), and a section for the direct download link starting with `https://drive.google.com/uc?export=download&id=1fEEG5UIGlvv68a84UhR6K3Zl_0Gi1JCP`. A large "Ctrl C" icon is overlaid on the right side of the direct download link section. To the left of the interface, three red numbers indicate the steps: 1. 貼上網址 (Paste the URL), 2. 點選 (Click), and 3. 複製 (Copy).

注意事項

1. 請確認您的 Google 雲端硬碟檔案權限已設定為「知道連結的人皆可查看」。如果設定為「受限」，則只有登入 Google 且擁有檔案存取權的人，才能開啟該直接下載連結。
2. 本工具僅支援已上傳的檔案，不適用於 Google 文件、試算表或簡報等直接在雲端建立的內容。
3. 如果檔案非常大，點擊直接下載連結時，Google 可能會先顯示一個頁面，提示無法掃描病毒，該頁面會提供一個按鈕讓您繼續下載。

Google 雲端硬碟直接下載連結轉換器

請貼上您的 Google 雲端硬碟連結：

`https://drive.google.com/file/d/1fEEG5UIGlvv68a84UhR6K3Zl_0Gi1JCP/view?usp=sharing`

產生直接下載連結

直接下載連結：

`https://drive.google.com/uc?export=download&id=1fEEG5UIGlvv68a84UhR6K3Zl_0Gi1JCP`

Ctrl C

1. 貼上網址
2. 點選
3. 複製

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=1Ffrs0geJKSHtRM3Q649HJvSF1l4blxpl", "/content/training_image.zip")  
8 #下載training_label.zip  
9 gdown.download("https://drive.google.com/uc?export=download&id=1KmYS2JhAexpWWlzl_o_qwoX1TifA1COWW", "/content/training_label.zip")  
10 #下載訓練aortic_valve_colab.vaml  
11 gdown.download("https://drive.google.com/u/0/uc?id=10yGJ7mfVLutjkeE55U9wfs4bmHCtINlcv&export=download", "/content/aortic_valve_colab.vaml")
```

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。



The screenshot displays a Jupyter Notebook interface with three main components: a file explorer on the left, a code editor in the center, and an output area on the right.

File Explorer (Left): Shows a directory structure with 'datasets' containing 'train' and 'val' subdirectories. Each subdirectory has 'images' and 'labels' folders. Under 'train/images', several PNG files are listed, including 'patient0031_0122.png' through 'patient0031_0128.png'.

Code Editor (Center): Contains Python code for moving files. The code defines a function `find_patient_root` to locate training data, sets up directories for training and validation, and uses `shutil.move` to relocate image and label files based on patient IDs. Comments in the code specify moving files for patients 0001-0030 to the training set and 0031-0050 to the validation set.

```
1 #移動檔案
2 def find_patient_root(root):
3     """往下找，直到找到含有 patientXXXX 的資料夾"""
4     for dirpath, dirnames, filenames in os.walk(root):
5         if any(d.startswith("patient") for d in dirnames):
6             return dirpath
7     return root # fallback
8
9 # 解壓縮到固定資料夾
10 if not os.path.isdir("./training_image") and os.path.exists("training_image.zip"):
11     os.makedirs("./training_image", exist_ok=True)
12     !unzip -q training_image.zip -d ./training_image
13
14 if not os.path.isdir("./training_label") and os.path.exists("training_label.zip"):
15     os.makedirs("./training_label", exist_ok=True)
16     !unzip -q training_label.zip -d ./training_label
17
18 IMG_ROOT = find_patient_root("./training_image")
19 LBL_ROOT = find_patient_root("./training_label")
20
21 print("IMG_ROOT =", IMG_ROOT)
22 print("LBL_ROOT =", LBL_ROOT)
23
24 # 建立輸出資料夾
25 os.makedirs("./datasets/train/images", exist_ok=True)
26 os.makedirs("./datasets/train/labels", exist_ok=True)
27 os.makedirs("./datasets/val/images", exist_ok=True)
28 os.makedirs("./datasets/val/labels", exist_ok=True)
29
30 def move_patients(start, end, split):
31     for i in range(start, end + 1):
32         patient = f"patient{i:04d}"
33         img_dir = os.path.join(IMG_ROOT, patient)
34         lbl_dir = os.path.join(LBL_ROOT, patient)
35         if not os.path.isdir(lbl_dir):
36             continue
37
38         for fname in os.listdir(lbl_dir):
39             if not fname.endswith(".txt"):
40                 continue
41
42             label_path = os.path.join(lbl_dir, fname)
43             base, _ = os.path.splitext(fname) # 取出檔名不含副檔名
44             img_path = os.path.join(img_dir, base + ".png")
45             if not os.path.exists(img_path):
46                 print(f"找不到對應圖片: {img_path}")
47                 continue
48
49             shutil.move(img_path, f"./datasets/{split}/images/")
50             shutil.move(label_path, f"./datasets/{split}/labels/")
51
52 # patient0001~0030 -> train
53 move_patients(1, 30, "train")
54
55 # patient0031~0050 -> val
56 move_patients(31, 50, "val")
57
58 print("完成移動!")
```

Output Area (Right): Shows the execution of the code, with the final output being "完成移動!" (Move completed!).

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)

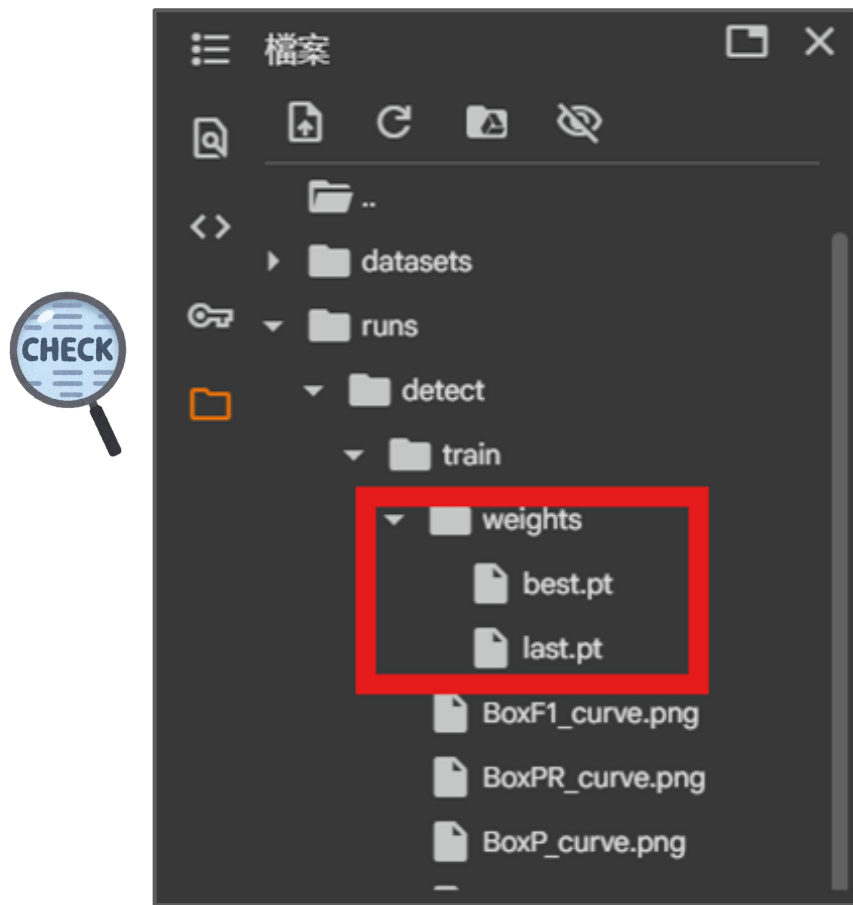
執行後依序有Epoch在跑代表成功

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%
	all	1156	1156	0.0687	0.344	0.0596	0.0254 37/37 3.1it/s 12.0s
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%
	all	1156	1156	0.645	0.765	0.73	0.397 37/37 3.6it/s 10.2s
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%
	all	1156	1156	0.672	0.589	0.645	0.3 37/37 3.4it/s 10.9s
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%
	all	1156	1156	0.754	0.735	0.815	0.497 37/37 3.5it/s 10.5s
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%
	all	1156	1156	0.725	0.754	0.807	0.487 37/37 3.5it/s 10.6s

```
[ ] 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。

The screenshot displays a Google Colab environment. The top part shows a terminal window with the following commands and output:

```
[9] 0 秒
1 !zip -r '/content/train.zip' '/content/runs/detect/train' #打包訓練模型和結果
2 from google.colab import files
3 files.download('/content/train.zip')
```

The terminal output lists various files being added to the zip archive, including weights, confusion matrix, results, and training/validation images. At the bottom of the terminal, a red box highlights the message: "Downloading 'train.zip': [progress bar]".

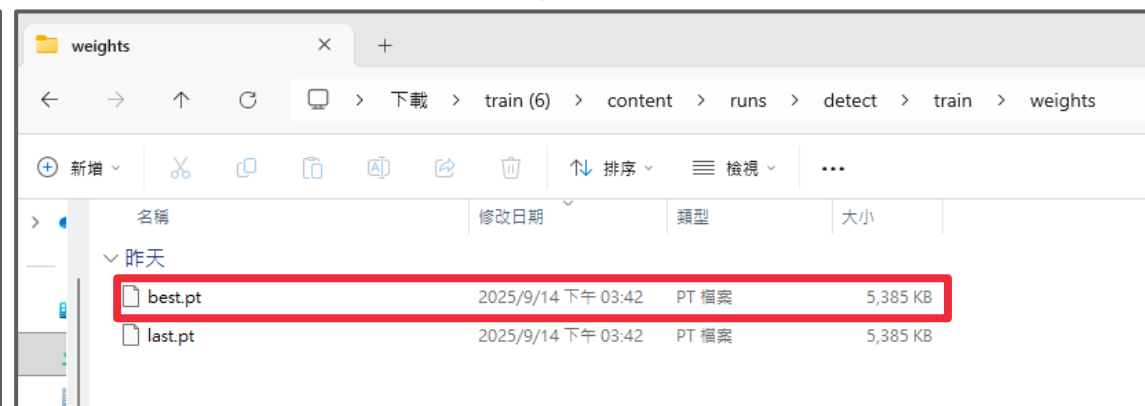
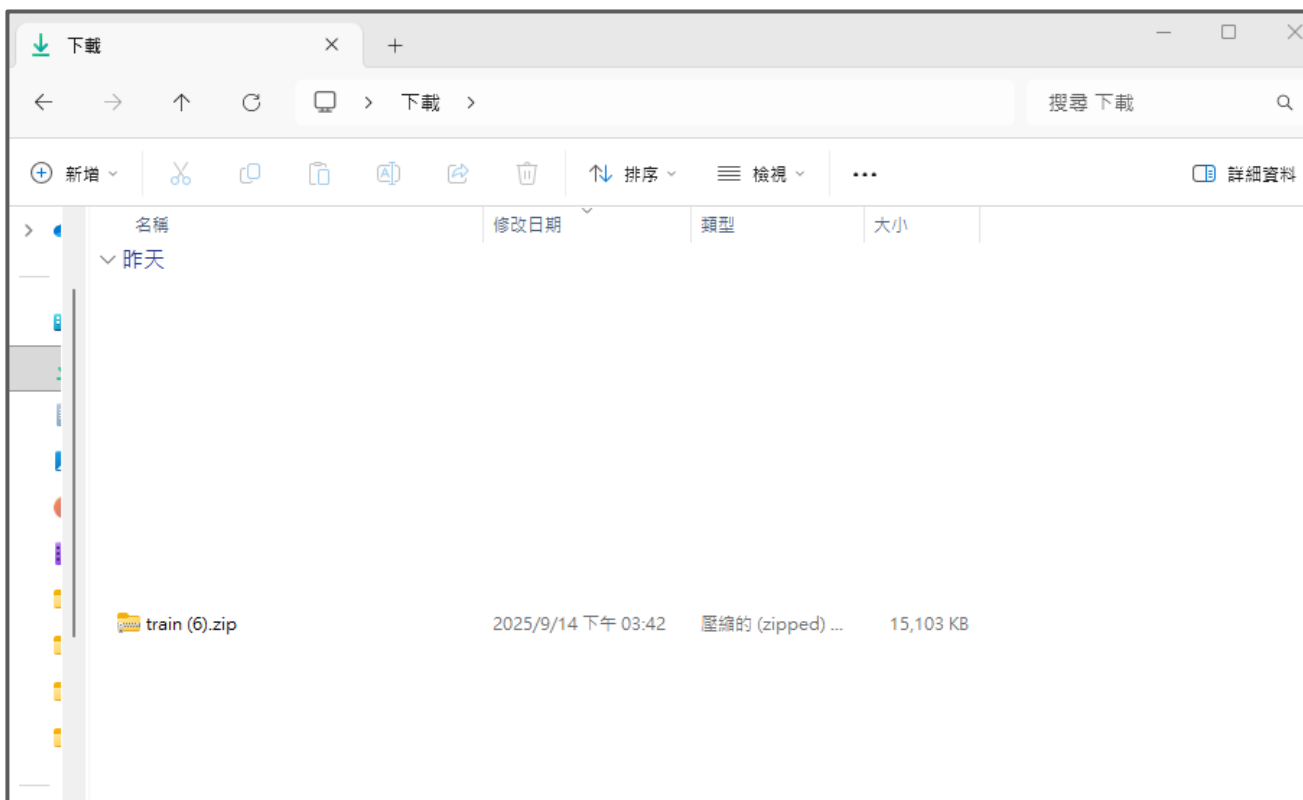
Overlaid on the bottom left is a "Downloading..." dialog box with a blue progress bar.

Overlaid on the right is a file explorer window titled "下載" (Download). It shows a table of downloaded files:

名稱	修改日期	類型	大小
train (6).zip	2025/9/14 下午 03:42	壓縮的 (zipped) ...	15,103 KB

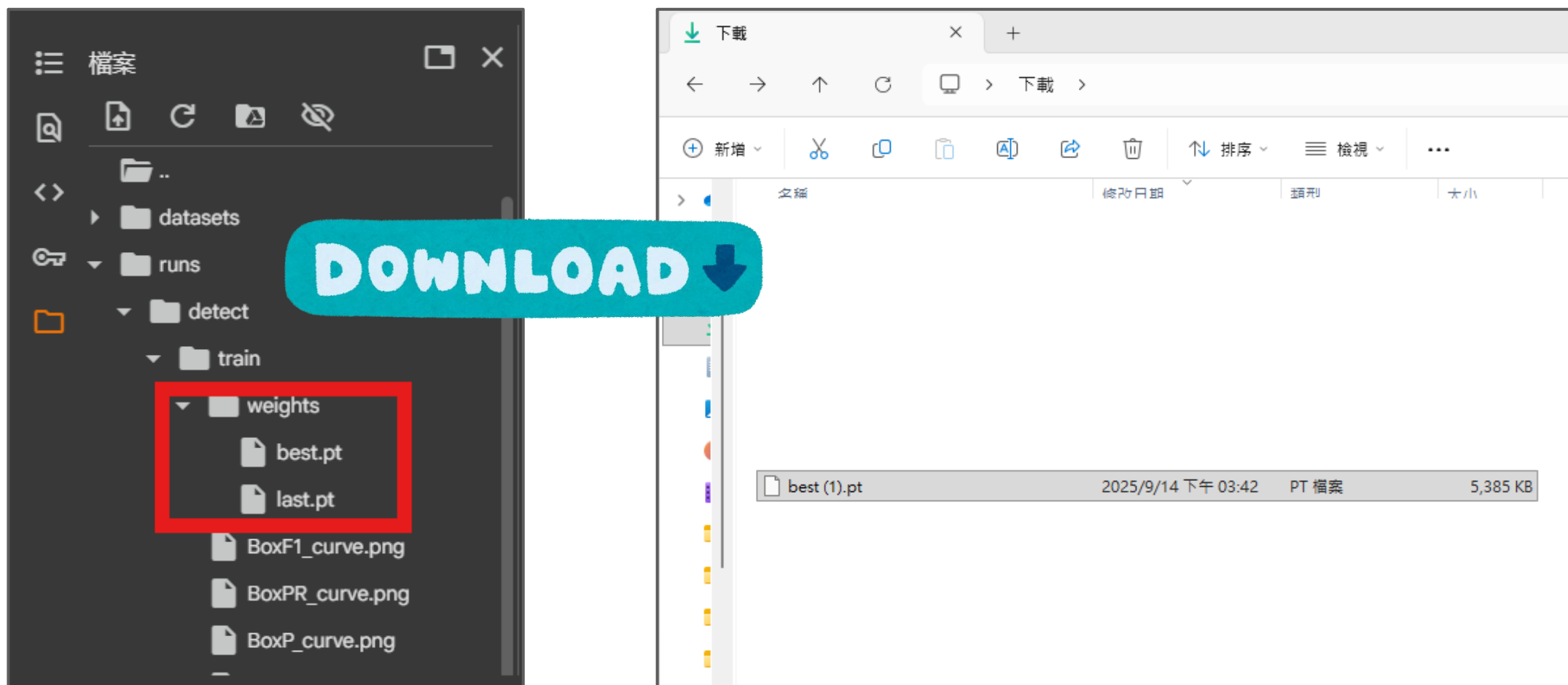
從本地下載確認檔案

- 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

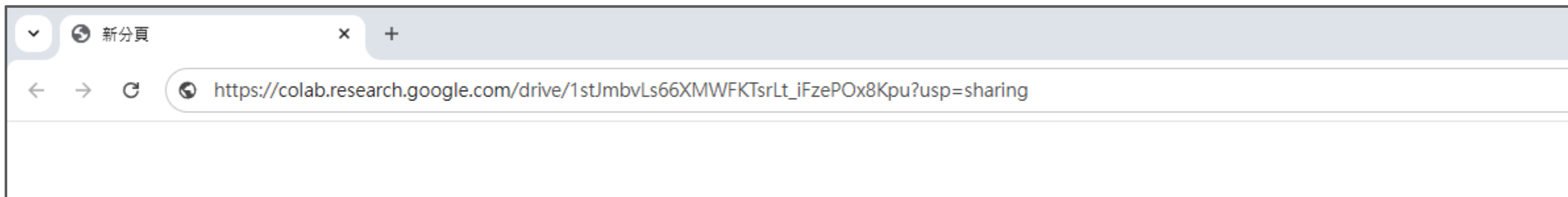
Colab Baseline 程式說明 (predict)

打開瀏覽器



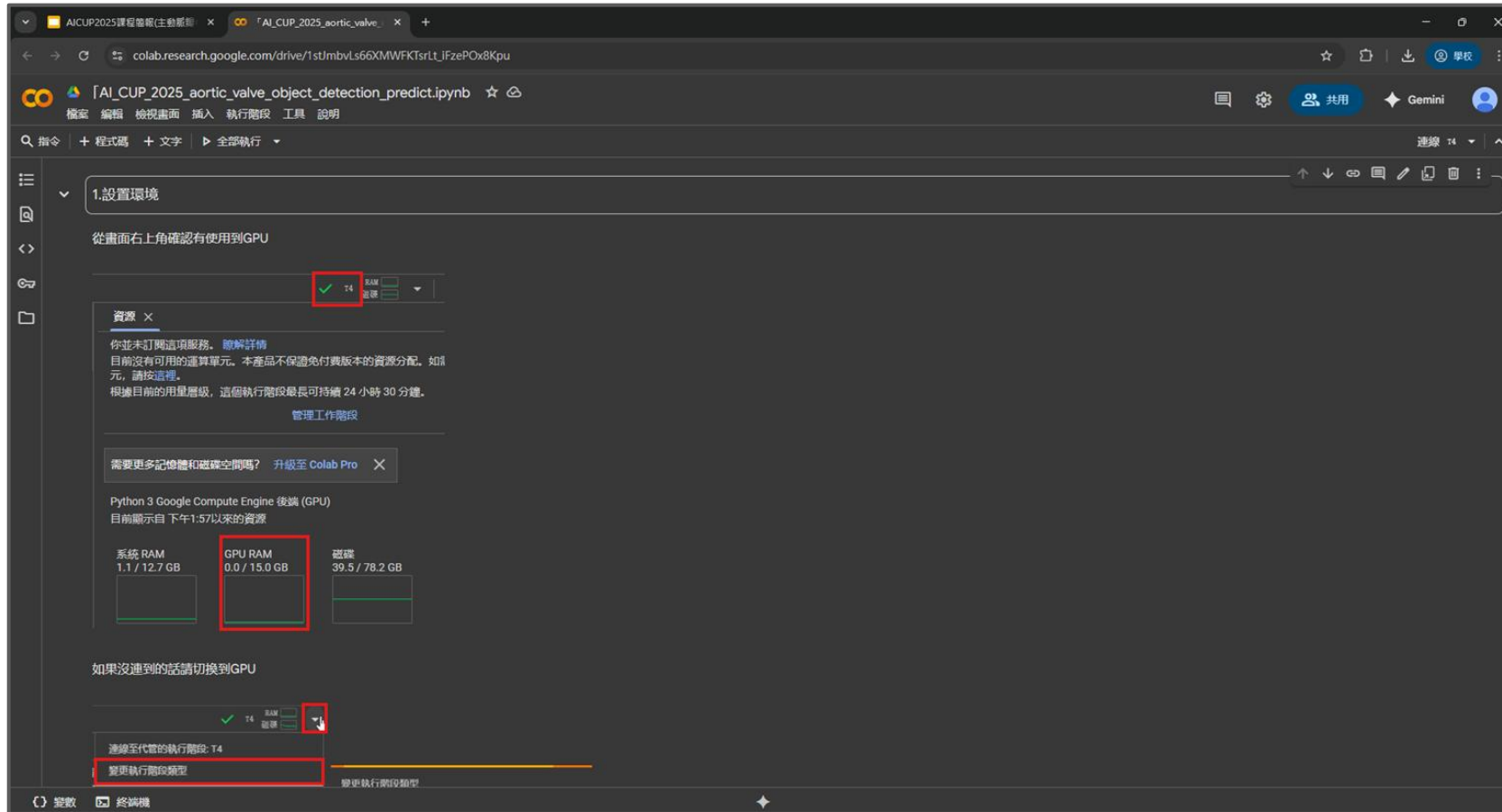
新興智慧顯示科技
教育聯盟

Baseline 會透過瀏覽器打開 Colab 完成預測。



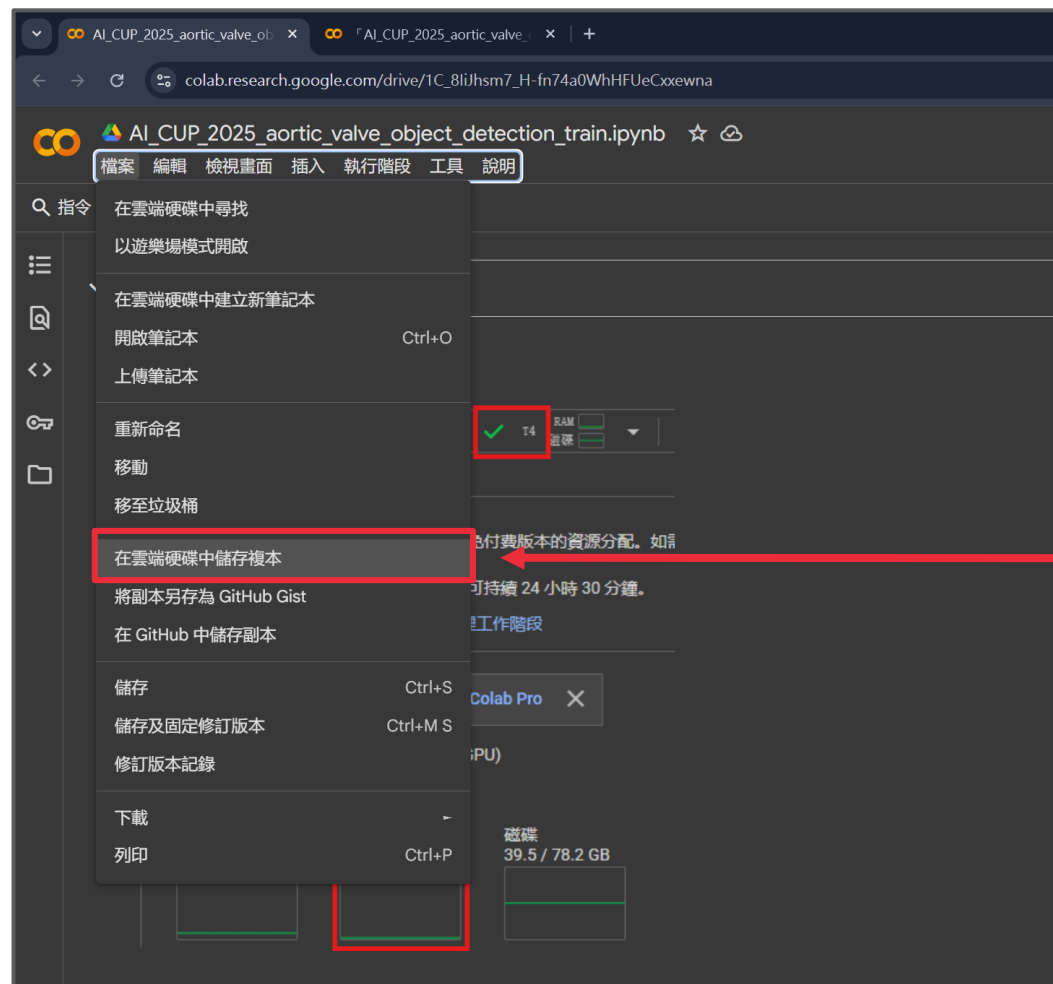
開啟Colab(predict)

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選查看資源是否有 GPU RAM。

The screenshot displays the Google Colab interface for a notebook titled "AI_CUP_2025_aortic_valve_object_detection_train.ipynb". The interface is in Chinese. In the top right corner, there is a "共用" (Share) button and a "Gemini" logo. Below the notebook title, there are tabs for "檔案" (Files), "編輯" (Edit), "檢視畫面" (View), "插入" (Insert), "執行階段" (Runtime), "工具" (Tools), and "說明" (Help). The "執行階段" (Runtime) tab is selected, showing a dropdown menu with "全部執行" (All) and a green checkmark next to "T4".

On the left sidebar, there is a "1.設置環境" (1. Setup environment) section. It contains a message: "從畫面右上角確認有使用到GPU" (Confirm GPU usage from the top right corner of the screen). Below this, there is a "資源" (Resources) section with a green checkmark and "T4" label. It also includes a message: "你並未訂閱這項服務。瞭解詳情" (You are not subscribed to this service. Learn more) and "目前沒有可用的運算單元。本產品不保證免付費版本的資源分配。如需要，請按這裡。" (No available compute units. This product does not guarantee resource allocation for the free version. If needed, click here). A button "管理工作階段" (Manage runtime) is visible.

At the bottom of the sidebar, there is a section titled "Python 3 Google Compute Engine 後端 (GPU)" (Python 3 Google Compute Engine backend (GPU)). It shows resource usage: "系統 RAM 1.1 / 12.7 GB", "GPU RAM 0.0 / 15.0 GB" (highlighted with a red box), and "磁碟 39.5 / 78.2 GB".

On the right side, there is a "資源" (Resources) section. It contains a message: "你並未訂閱這項服務。瞭解詳情" (You are not subscribed to this service. Learn more) and "目前沒有可用的運算單元。本產品不保證免付費版本的資源分配。如需要，請按這裡。" (No available compute units. This product does not guarantee resource allocation for the free version. If needed, click here). A button "管理工作階段" (Manage runtime) is visible. Below this, there is a section titled "Python 3 Google Compute Engine 後端 (GPU)" (Python 3 Google Compute Engine backend (GPU)). It shows resource usage: "系統 RAM 1.1 / 12.7 GB", "GPU RAM 0.0 / 15.0 GB" (highlighted with a red box), and "磁碟 39.5 / 78.2 GB".

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



```
查看GPU資訊(運行時間<1秒)
```

```
[1] 0 秒
```

```
1 !nvidia-smi
```

10 02.00.00 2025

Colab運行情況(執行中)

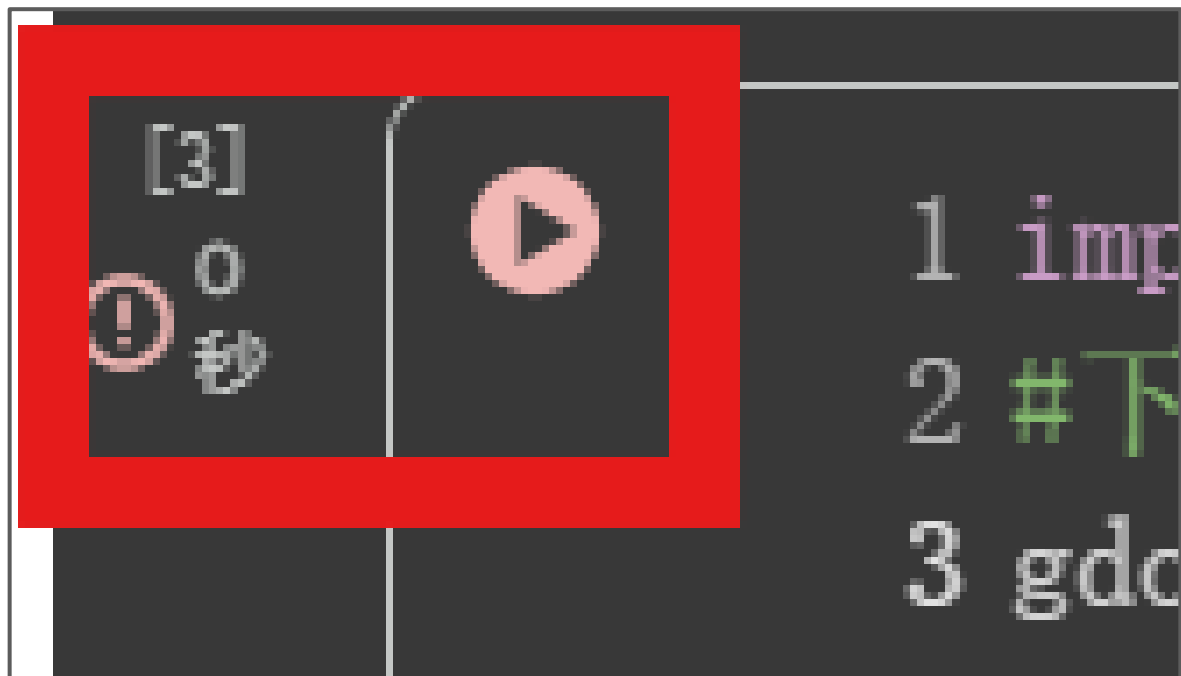
區塊左方有圈在轉代表執行中



```
[8] 1 from ult
    2
    3 model =
    4 vocu1 + c
```

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| 0x-            0x    | GPU-Util  Compute M. |
|===========================================|=====================|========================|
| 0     Tesla T4            Off          | 00000000:00:04:0 Off |                    0 |
| N/A   38C    P8           9W /  70W   | 0MiB / 15360MiB | 0%      Default  |
|===========================================|=====================|========================|

Processes:
+-----+-----+
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   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

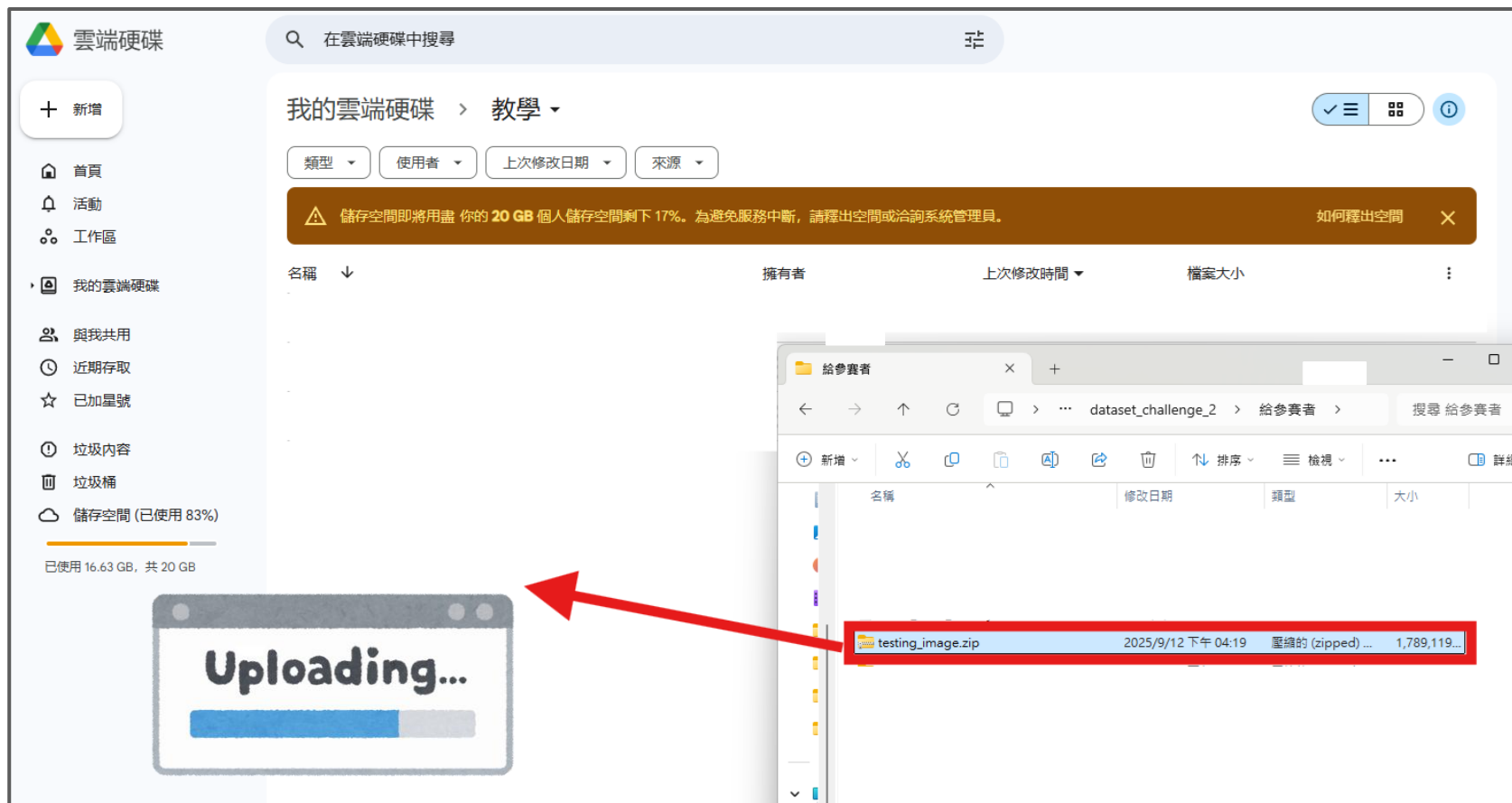
下載YOLOv12套件(運行時間12秒)

[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。

雲端硬碟

在雲端硬碟中搜尋

我的雲端硬碟 > 教學

已選取 1 個項目

名稱	擁有者	上次修改時間	檔案大小
testing_image.zip	我	2025年9月10日 我	1.71 GB

1. 選擇檔案

2. 點選管理存取權

管理存取權

共用「testing_image.zip」

新增使用者、群組和日曆活動

具有存取權的使用者

一般存取權

限制

知道連結的任何人

4. 完成

完成

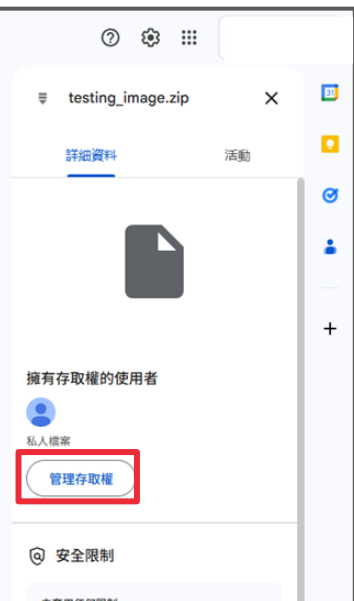
CHECK

取得檔案連結方法

取得檔案下載連結



2.點選管理存取權



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>

注意事項

1. 請確認您的 Google 雲端硬碟檔案權限已設定為「知道連結的人皆可查看」。如果設定為「受限」，則只有登入 Google 且擁有檔案存取權的人，才能開啟該直接下載連結。
2. 本工具僅支援已上傳的檔案，**不適用於 Google 文件、試算表或簡報等直接在雲端建立的內容。**
3. 如果檔案非常大，點擊直接下載連結時，Google 可能會先顯示一個頁面，提示無法掃描病毒，該頁面會提供一個按鈕讓您繼續下載。

Google 雲端硬碟直接下載連結轉換器

請貼上您的 Google 雲端硬碟連結：

☒ 直接下載連結：

Ctrl C

1. 貼上網址
2. 點選
3. 複製

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

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



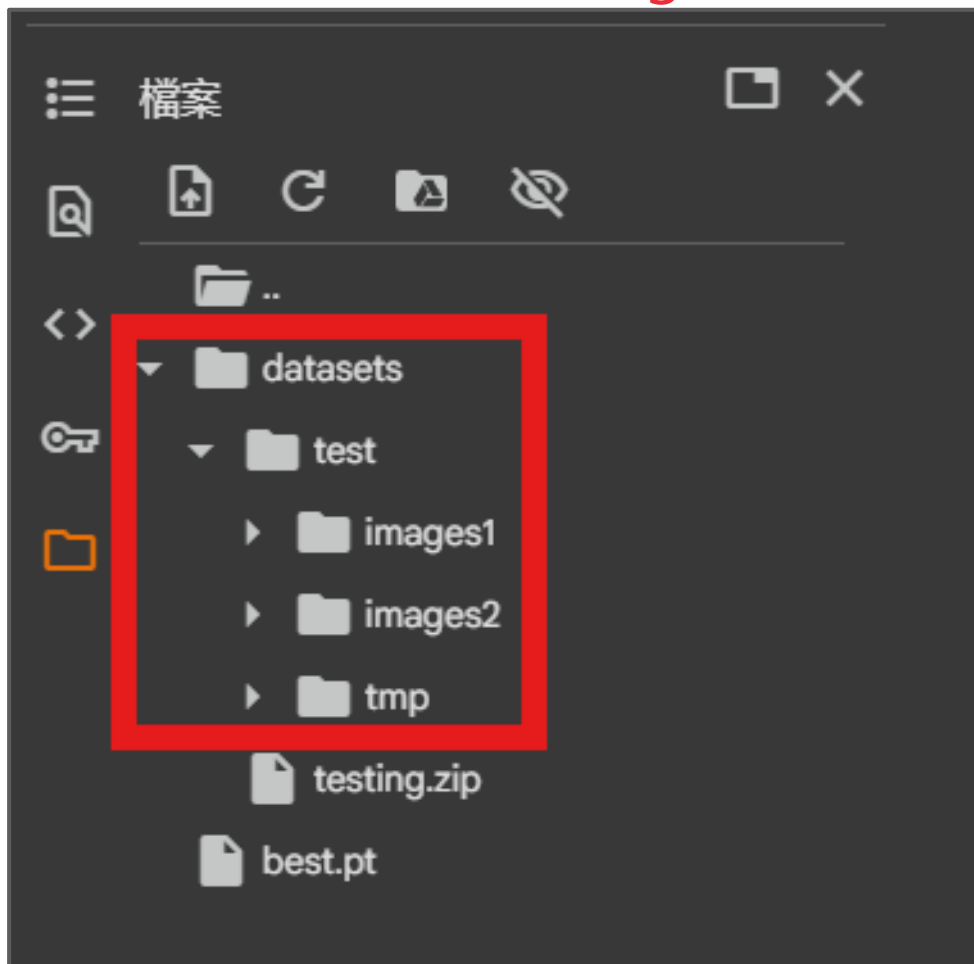
Colab 運算資源限制

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 **images1** 檔案夾，另一半移至 **images2** 檔案夾。



```
[ ]
1 import os
2 import shutil
3
4 src_root = "/content/datasets/test/tmp"
5 dst_root1 = "/content/datasets/test/images1"
6 dst_root2 = "/content/datasets/test/images2"
7
8 os.makedirs(dst_root1, exist_ok=True)
9 os.makedirs(dst_root2, exist_ok=True)
10
11 # 收集所有圖片路徑
12 all_files = []
13 for patient_folder in os.listdir(src_root):
14     patient_path = os.path.join(src_root, patient_folder)
15     if os.path.isdir(patient_path) and patient_folder.startswith("patient"):
16         for fname in os.listdir(patient_path):
17             if fname.endswith(".png"):
18                 all_files.append(os.path.join(patient_path, fname))
19
20 # 按照檔名排序，方便重現結果
21 all_files.sort()
22
23 # 計算一半
24 half = len(all_files) // 2
25
26 # 前半移到 images1/
27 for f in all_files[:half]:
28     dst_file = os.path.join(dst_root1, os.path.basename(f))
29     shutil.move(f, dst_file)
30
31 # 後半移到 images2/
32 for f in all_files[half:]:
33     dst_file = os.path.join(dst_root2, os.path.basename(f))
34     shutil.move(f, dst_file)
35
36 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。

3. Predict

下載模型權重檔(此權重檔為baseline提供的模型，參賽者可替換成自行訓練的模型檔)

使用自行訓練模型可透過拖移方式上傳至Colab

命令 | 程式碼 | 文字 | 全部執行

檔案 | 樣本數據

weights

名稱	修改日期	類型	大小
best.pt	2023/9/13 下午 06:36	PT 檔案	5,385 KB
test.pt	2023/9/13 下午 06:36	PT 檔案	5,385 KB

```
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'

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'
```

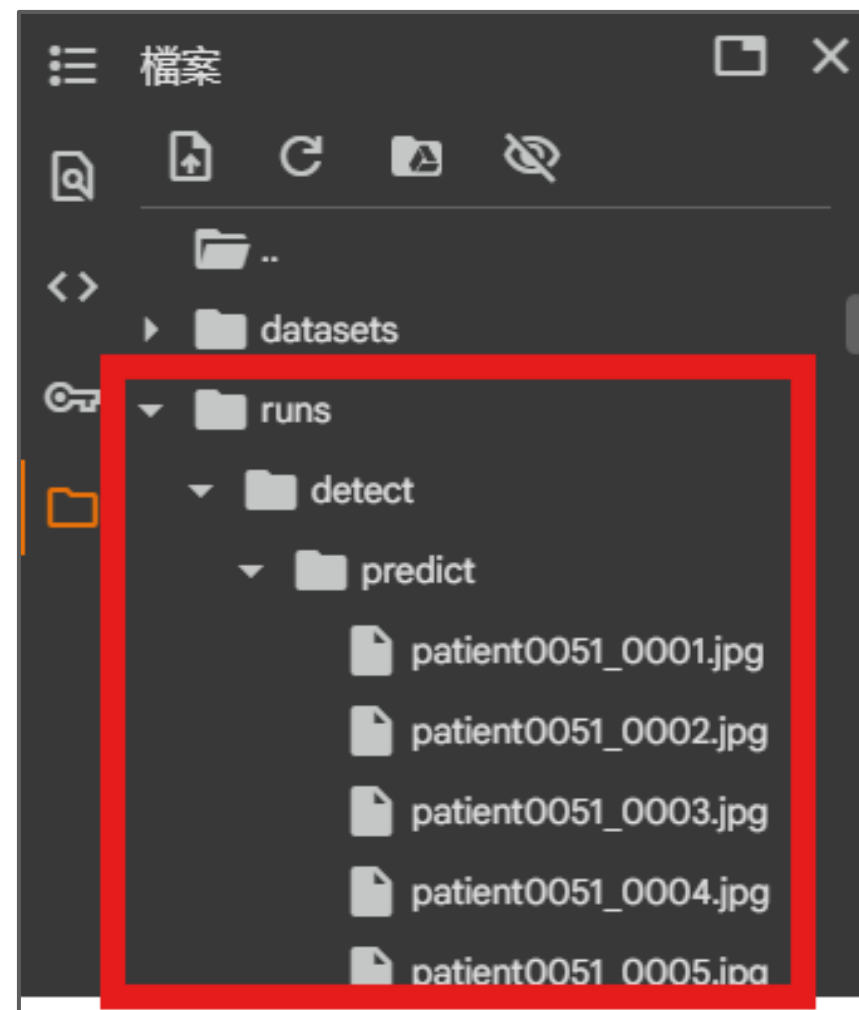
進行預測

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)
```

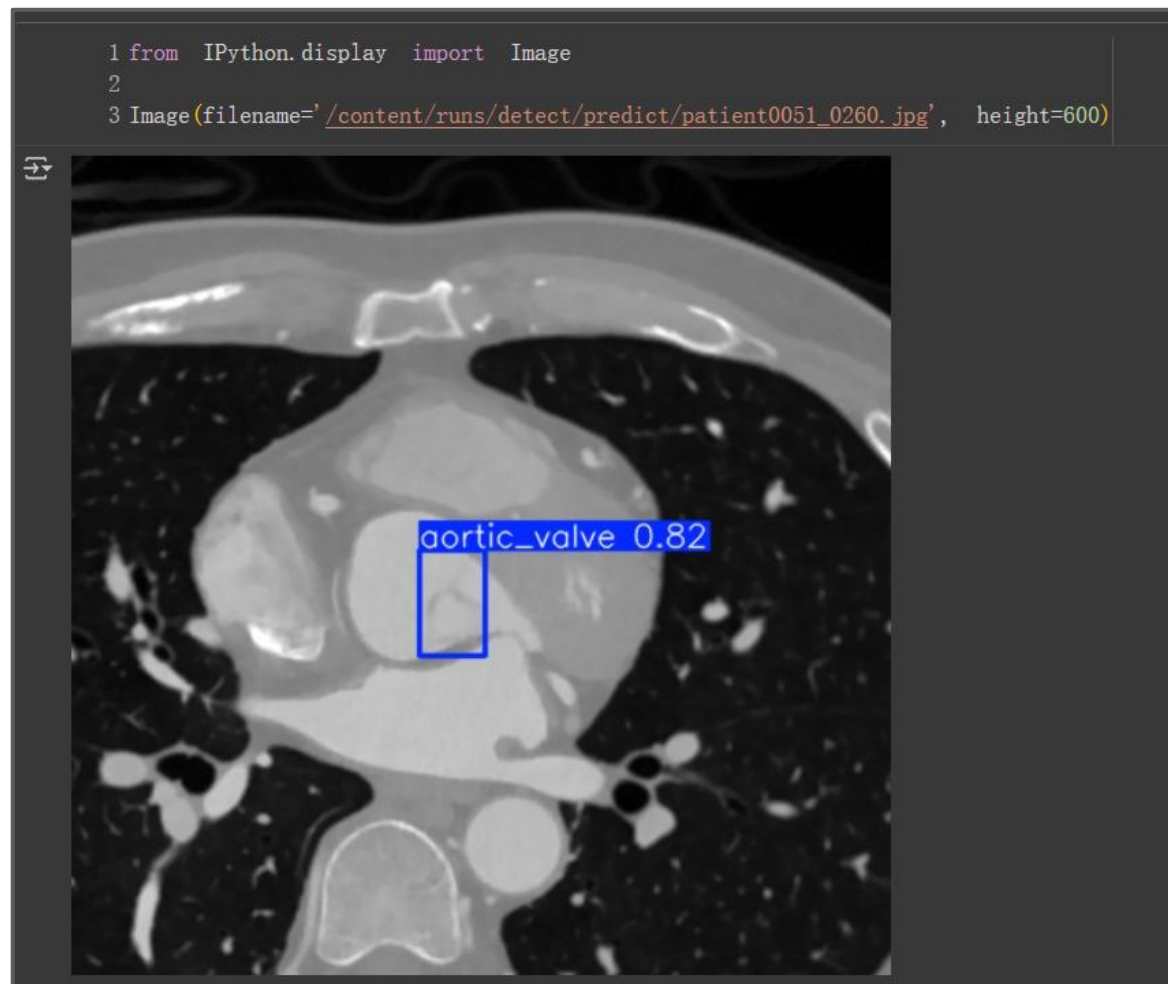
```
[9] 3分鐘  
✓ 3分鐘  
1 from ultralytics import YOLO  
2  
3 model = YOLO('/content/best.pt')  
4 results = model.predict(source="/content/datasets/test/images1/",  
5                           save=True,  
6                           imgsiz=640,  
7                           device=0  
8                           )
```

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



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]

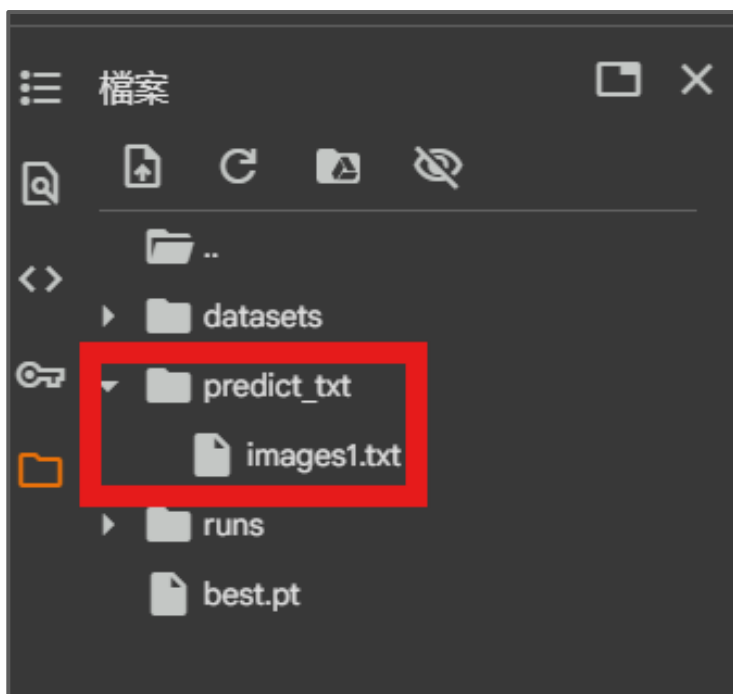
✓ 0 秒

```
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
1 !mkdir ./predict_txt/
2 output_file = open('./predict_txt/images1.txt', 'w')
3 for i in range(len(results)):
4     # 取得圖片檔名 (不含副檔名)
5     filename = results[i].path.split('/')[-1].split('.png')[0]
6
7     # 取得預測框數量
8     boxes = results[i].boxes
9     box_num = len(boxes.cls.tolist())
10
11     # 如果有預測框
12     if box_num > 0:
13         for j in range(box_num):
14             # 提取資訊
15             label = int(boxes.cls[j].item()) # 類別
16             conf = boxes.conf[j].item() # 信心度
17             x1, y1, x2, y2 = boxes.xyxy[j].tolist() # 邊界框座標
18
19             # 建立一行資料
20             line = f"{filename} {label} {conf:.4f} {int(x1)} {int(y1)} {int(x2)} {int(y2)}\n"
21             output_file.write(line)
22
23 # 關閉輸出檔案
24 output_file.close()
25
```

釋放RAM

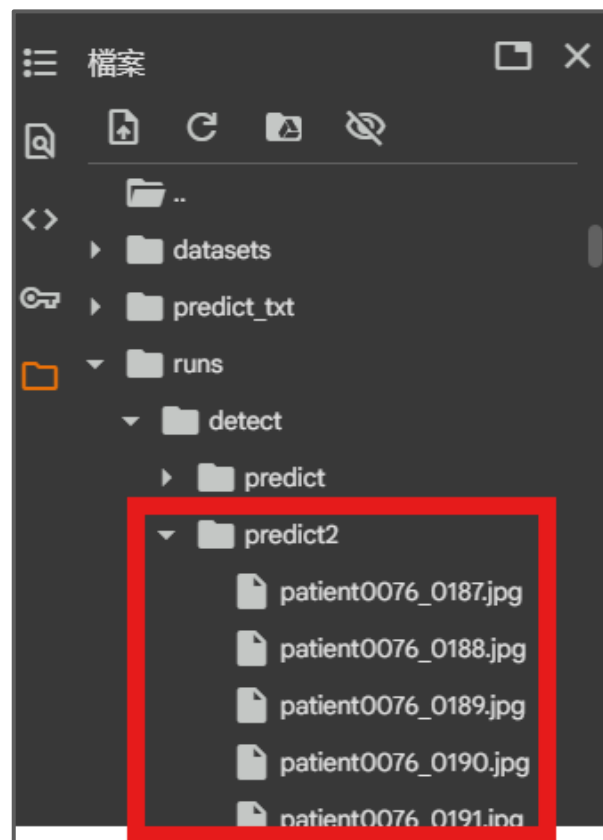
因為Colab RAM不會自動釋放，所以透過程式釋放。

```
[ ]
```

```
1 import torch , gc
2
3 # 刪除大型變數
4 del boxes, all_files, results
5 gc.collect()
6 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)

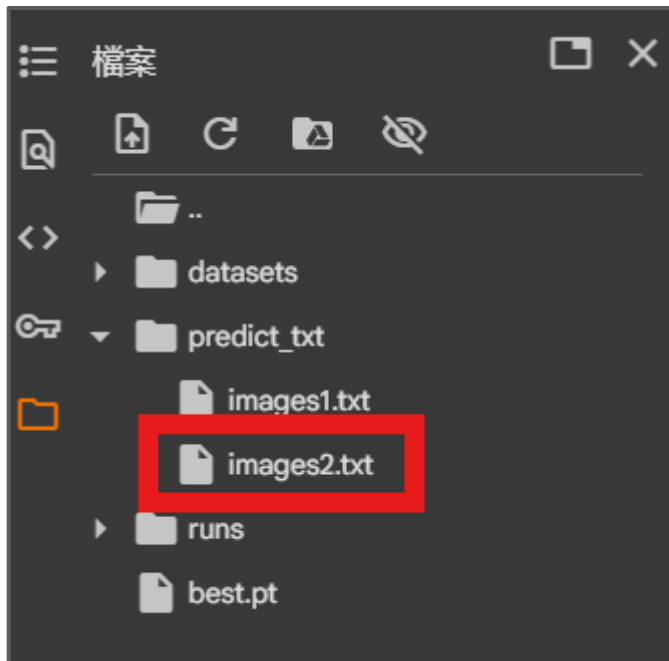
[15]
✓ 3分鐘

1 from ultralytics import YOLO
2
3 model = YOLO('/content/best.pt')
4 results = model.predict(source="/content/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
image 8262/8310 /content/datasets/test/images2/patient0100_0308.png: 640x640 (no detections), 18.7ms
```

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` 。

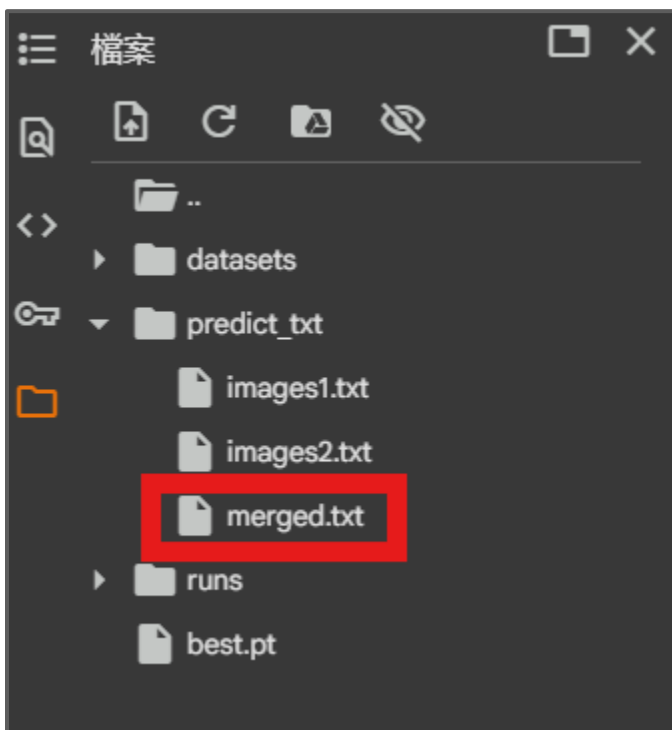


```
[16]
✓ 0 秒

1 output_file = open('./predict_txt/images2.txt', 'w')
2 for i in range(len(results)):
3     # 取得圖片檔名 (不含副檔名)
4     filename = results[i].path.split('/')[1].split('.')[0]
5
6     # 取得預測框數量
7     boxes = results[i].boxes
8     box_num = len(boxes.cls.tolist())
9
10    # 如果有預測框
11    if box_num > 0:
12        for j in range(box_num):
13            # 提取資訊
14            label = int(boxes.cls[j].item()) # 類別
15            conf = boxes.conf[j].item() # 信心度
16            x1, y1, x2, y2 = boxes.xyxy[j].tolist() # 邊界框座標
17
18            # 建立一行資料
19            line = f"{filename} {label} {conf:.4f} {int(x1)} {int(y1)} {int(x2)} {int(y2)}\n"
20            output_file.write(line)
21
22 # 關閉輸出檔案
23 output_file.close()
24
```

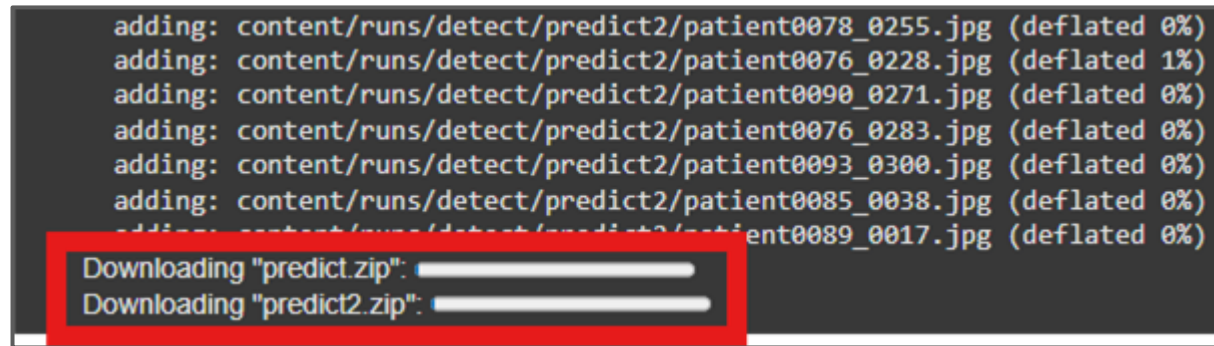
合併兩個.txt

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



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



新興智慧顯示科技
教育聯盟

將 merged.txt 檔案提交至趨勢科技 Tbrain 競賽網站。

The screenshot shows the T-Brain competition website for the "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". The website has a dark blue header with navigation links: Home, Competitions, Discussion, Datasets, Success Story, and a Sign In button. Below the header, the competition title is displayed in large white text. A yellow button labeled "即將開始" (Starting Soon) is visible. The main content area has a light blue navigation bar with links: Overview, Leaderboard, and Download Dataset. The "Overview" section contains the "競賽說明" (Competition Description) and "報名規範" (Registration Rules). The "Download Dataset" link is highlighted with a red arrow. To the right, a file explorer window is open, showing a file named "merged(2).txt" with a size of 205 KB, which is highlighted with a red box.

Download Dataset

競賽說明

本競賽旨在進行電腦斷層心臟肌肉影像分割，分割目標有二：全心臟肌肉以及主動脈瓣。透過深度學習方法訓練模型，所預測出的分割結果能夠大幅度地減少人工標記所需的人員時間。主動脈瓣的功能為幫助心臟全身的器官。經導管主動脈瓣置換手術能夠解決主動脈瓣狹窄問題，手術後主動脈瓣位置的定位。為此，我們舉辦了電腦斷層心臟肌肉影像分割與主動脈瓣物件偵測的影像分割技術，在獲得心臟肌肉分割結果後，能夠建立病患的專屬心臟模型與模擬，避免手術進行中的相關風險。

報名規範

1. 具中華民國學籍之在學學生（大學生、研究生、及其他在校學生皆可參加，包括國內及國際社會人士），凡年滿18歲皆可報名參加。若為未滿18歲之人同意後可報名參加。（趨勢科技公司員工除外）。
2. 參與學生組排名賽競賽的學生參賽者，必須於報名系統登記其就讀學校的資訊。

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