AI CUP 2025 秋季賽

電腦斷層主動脈瓣物件偵測競賽報告

隊伍 : TEAM_9104
隊員 : 王丞頤
Private leaderboard : 0.973565 / Rank 11

1、 資料處理與程式環境

1、資料前處理:

訓練:

使用的環境:因為我有購買colab pro所以有A100可以用，故我選擇訓練了多模型去想要透過ensemble predict去做預測結果，故為了讓不同的模型能夠看到不同的資料避免因為部分資料拿去驗證而少看了驗證相關的資料特性，故我分兩種策略:

1. 訓練集: (1~40) 驗證(41~50)
2. 訓練: (1~10 , 21~50) 驗證: (11~20)
   其如果是用第二種切割方法的模型命名會出現 _a ex: 12n_a

```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.copy2(img_path, f"./datasets/{split}/images/")
            shutil.copy2(label_path, f"./datasets/{split}/labels/")
```

```
# #法1:
# move_patients(1, 40, "train")
# move_patients(41, 50, "val")

#法2:
# patient0001~0030 → train
move_patients(1, 10, "train")
move_patients(21, 50, "train")
# patient0031~0050 → val
move_patients(11, 20, "val")

print("完成移動！")
```

2、原始程式碼概述

訓練:

使用 Ultralytics 套件內的 YOLOv12m,12n,11m,11n,10m,9c 去當我的預訓練模型進行訓練，並且因為考量到數據的數量並不算非常多，如果epoch設定太大會導致overfit，所以設定epoch=85，並且為了讓訓練loss優化更平滑，所以batch=32，在訓練的過程中有調整過不同的imgsz當時有設定過 512 640 768 960 這幾個，因為考量到圖片都是512*512，所以最後基於結果出來可以發現640是比較好的解析度，避免在提升解析度時因為過度插植導致失真，最後我想讓倒數幾圈可以關掉圖片的augument，故在最後3圈時關掉mosaic，
最後設定優化器為AdamW，每個模型的訓練策略均同上(差異為: 資料集上分割的不同)，最終將結果打包成zip

```
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
```

```
!pip install ultralytics
import ultralytics
ultralytics.checks()
```

```
Ultralytics 8.3.233 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (NVIDIA A100-SXM4-80GB, 81222MiB)
Setup complete ✅ (12 CPUs, 167.1 GB RAM, 38.1/235.7 GB disk)
```

```
from ultralytics import YOLO
model = YOLO("yolo12n.pt")

model.train(
    data="./aortic_valve_colab.yaml",
    epochs=85,
    batch=32,
    imgsz=640,
    patience=40,
    device=0,
    close_mosaic=3,
    optimizer="AdamW",
    amp=True,
)
```

```
# from ultralytics import YOLO     Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this
# model = YOLO("yolo12m.pt")

# model.train(
#     data="./aortic_valve_colab.yaml",
#     epochs=85,
#     batch=32,
#     imgsz=640,
#     patience=40,
#     device=0,
#     close_mosaic=3,
#     optimizer="AdamW",
#     amp=True,
# )
```

```
# from ultralytics import YOLO     Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this
# model = YOLO("yolo11n.pt")

# model.train(
#     data="./aortic_valve_colab.yaml",
#     epochs=85,
#     batch=32,
#     imgsz=640,
#     patience=40,
#     device=0,
#     close_mosaic=3,
#     optimizer="AdamW",
```

推論:

一樣透過 Ultralytics 套件去完成預測，並且因為我是將權重放在雲端，讓其可以透過gdown下載:

最後採用了: [12m95,11m95,10m_a,9c,12n,11n]，並選擇透過ensemble-boxes去融合，並且因為透過給予不同的模型不同的權重占比，以及在前期的conf 和 Iou我有相對的給比較寬鬆的門檻，在最後融合時有再判斷一遍其min_conf，等於兩層過濾的概念，因為Ultralytics並沒有提供相關的融合API調用，最終因為有事先將預測圖片分成兩堆，故最終將兩個txt預測檔案合併成最終的txt

```
#12m
gdown.download("https://drive.google.com/uc?export=download&id=1kudnbcTwc_RlsNRQRNWJWSpRblWac6Bp","/content/yolo12m.pt")
#12m95
gdown.download("https://drive.google.com/uc?export=download&id=1CXDxIVdue-KeQVTdk8K68CqJ1xzYF1F6","/content/yolo12m95.pt")
#11m
gdown.download("https://drive.google.com/uc?export=download&id=1wdiGZPjCXAeQfj-VDxvN9Medsi3d3foK","/content/yolo11m.pt")
#11m95
gdown.download("https://drive.google.com/uc?export=download&id=1tegM1-Gkdfge3hK08_zYozPwTyEcSFkE","/content/yolo11m95.pt")
#9c
gdown.download("https://drive.google.com/uc?export=download&id=1bCISAs4SoJnVGaY1MQ_YDwJWD3mt3UcX","/content/yolo9c.pt")
#12a
gdown.download("https://drive.google.com/uc?export=download&id=1r4ElyxgNKSgCrIY1_gxddpkhbpFnKiwL","/content/yolo12a.pt")
#11a
gdown.download("https://drive.google.com/uc?export=download&id=1OXkGibEnnjFS1S8JmXPs51IPGsKhsP_b","/content/yolo11a.pt")
#12l_a
gdown.download("https://drive.google.com/uc?export=download&id=13zdIY1NUSBgfU7ZSzQYiWUqgx5OKoQMJ","/content/yolo12l_a.pt")
#10m_a
gdown.download("https://drive.google.com/uc?export=download&id=1drohNx_7Jal96J3EsRpZACUI30dMYvHv","/content/yolo10m_a.pt")
#11n
gdown.download("https://drive.google.com/uc?export=download&id=1EHjYYRqfQXmWaRtnVKkZbOBMKDuiD4Eu","/content/yolo11n.pt")
#12n
gdown.download("https://drive.google.com/uc?export=download&id=1KYvGQlpfPKUsTg4N7euazi1w5vQT9DH1","/content/yolo12n.pt")
```

```python
import os
import shutil

base_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)

# 自動找到第一個「直屬子資料夾含 patient*」的目錄
patient_root = base_root
for dirpath, dirnames, _ in os.walk(base_root):
    if any(d.lower().startswith("patient") for d in dirnames):
        patient_root = dirpath
        break

# 收集所有圖片路徑（只看直屬的 patient 資料夾）
all_files = []
for patient_folder in os.listdir(patient_root):
    patient_path = os.path.join(patient_root, patient_folder)
    if os.path.isdir(patient_path) and patient_folder.lower().startswith("patient"):
        for fname in os.listdir(patient_path):
            if fname.lower().endswith(".png"):
                all_files.append(os.path.join(patient_path, fname))

# 按名稱排序並對半移動
all_files.sort()
half = len(all_files) // 2

for f in all_files[:half]:
    shutil.move(f, os.path.join(dst_root1, os.path.basename(f)))

for f in all_files[half:]:
    shutil.move(f, os.path.join(dst_root2, os.path.basename(f)))

print(f"來源根目錄: {patient_root}")
```

程式碼儲存格 (Ctrl+M B)

```python
alytics import YOLO
from ensemble_boxes import weighted_boxes_fusion
import os
import numpy as np

# ──────────────────────────────
# 設定模型與資料夾
# ──────────────────────────────
model_paths = [
    "/content/yolo12m95.pt",          # 模型1: 12m
    "/content/yolo11m95.pt",          # 模型3: 11m
    "/content/yolo9c.pt",           # 模型3: 11m
    "/content/yolo12n.pt",           # 模型3: 11m
    "/content/yolo11n.pt",           # 模型3: 11m
    "/content/yolo10m_a.pt",          # 模型3: 11m
]

models = [YOLO(p) for p in model_paths]

source_dir = "./datasets/test/images1/"      # <— 你要跑的資料夾
output_txt = "./predict_txt/images1_wbf.txt"
os.makedirs("./predict_txt", exist_ok=True)

# ──────────────────────────────
# WBF 參數
# ──────────────────────────────
IOU_THR = 0.45
CONF_THR = 0.08

# ──────────────────────────────
# 開啟輸出檔案
# ──────────────────────────────
fout = open(output_txt, "w")
```

```
# ──────────────────────────
# 開啟輸出檔案
# ──────────────────────────
fout = open(output_txt, "w")

# ──────────────────────────
# 處理資料夾下的每張圖片
# ──────────────────────────
img_list = sorted([f for f in os.listdir(source_dir) if f.lower().endswith(".png")])

print(f"Total images = {len(img_list)}")

for img_name in img_list:
    img_path = os.path.join(source_dir, img_name)
    filename_noext = img_name.replace(".png", "")

    # ──── 多模型預測 (3 個) ────
    results = [
        m.predict(
            img_path,
            imgsz=640,
            conf=CONF_THR,
            iou=IOU_THR,
            verbose=False
        )[0]
        for m in models
    ]

    boxes_list = []
    scores_list = []
    labels_list = []

    # 假設所有模型輸入同一張圖，orig_shape 一樣
    H, W = results[0].orig_shape

    for r in results:
        if len(r.boxes) == 0:
```

```
    # 假設所有模型輸入同一張圖，orig_shape 一樣
    H, W = results[0].orig_shape

    for r in results:
        if len(r.boxes) == 0:
            boxes_list.append([])
            scores_list.append([])
            labels_list.append([])
            continue

        xyxy = r.boxes.xyxy.cpu().numpy()
        scores = r.boxes.conf.cpu().numpy()
        labels = r.boxes.cls.cpu().numpy()

        # 轉成 normalized xyxy
        norm_boxes = []
        for (x1, y1, x2, y2) in xyxy:
            norm_boxes.append([x1 / W, y1 / H, x2 / W, y2 / H])

        boxes_list.append(norm_boxes)
        scores_list.append(scores)
        labels_list.append(labels)

    # ──── WBF ────
    boxes_wbf, scores_wbf, labels_wbf = weighted_boxes_fusion(
        boxes_list,
        scores_list,
        labels_list,
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.2,1.2,1.2,1.2],
        iou_thr=0.5,
        skip_box_thr=0.01,
    )
    MIN_SCORE = 0.088
    # ──── 寫回你原本格式 ────
    for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
```

```
        )
        MIN_SCORE = 0.088
        # ——— 寫回你原本格式 ———
        for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
            if score < MIN_SCORE:
                continue
            # 還原 pixel xyxy
            x1 = int(box[0] * W)
            y1 = int(box[1] * H)
            x2 = int(box[2] * W)
            y2 = int(box[3] * H)

            line = f"{filename_noext} {int(label)} {score:.4f} {x1} {y1} {x2} {y2}\n"
            fout.write(line)

fout.close()
print(f"Done! Ensemble result saved to: {output_txt}")
```

```
Total images = 8310
Done! Ensemble result saved to: ./predict_txt/images1_wbf.txt
```

```
from ultralytics import YOLO
from ensemble_boxes import weighted_boxes_fusion
import os
import numpy as np

# ------------------------------
# 設定模型與資料夾
# ------------------------------
model_paths = [
    "/content/yolo12m95.pt",          # 模型1: 12m
    "/content/yolo11m95.pt",          # 模型3: 11m
    "/content/yolo9c.pt",           # 模型3: 11m
    "/content/yolo10m_a.pt",          # 模型3: 11m
    "/content/yolo12n.pt",          # 模型3: 11m
    "/content/yolo11n.pt",          # 模型3: 11m
]

models = [YOLO(p) for p in model_paths]

source_dir = "./datasets/test/images2/"      # <-- 你要跑的資料夾
output_txt = "./predict_txt/images2_wbf.txt"
os.makedirs("./predict_txt", exist_ok=True)

# ------------------------------
# WBF 參數
# ------------------------------
IOU_THR = 0.45
CONF_THR = 0.08

# ------------------------------
# 開啟輸出檔案
# ------------------------------
fout = open(output_txt, "w")

# ------------------------------
# 處理資料夾下的每張圖片
# ------------------------------
img_list = sorted([f for f in os.listdir(source_dir) if f.lower().endswith(".png")])

print(f"Total images = {len(img_list)}")
```

```
式碼儲存格 (Ctrl+M B)
    for img_name in img_list:
        img_path = os.path.join(source_dir, img_name)
        filename_noext = img_name.replace(".png", "")

        # ------ 多模型預測 (3 個) ------
        results = [
            m.predict(
                img_path,
                imgsz=640,
                conf=CONF_THR,
                iou=IOU_THR,
                verbose=False
            )[0]
            for m in models
        ]

        boxes_list = []
        scores_list = []
        labels_list = []

        # 假設所有模型輸入同一張圖，orig_shape 一樣
        H, W = results[0].orig_shape

        for r in results:
            if len(r.boxes) == 0:
                boxes_list.append([])
                scores_list.append([])
                labels_list.append([])
                continue

            xyxy = r.boxes.xyxy.cpu().numpy()
            scores = r.boxes.conf.cpu().numpy()
            labels = r.boxes.cls.cpu().numpy()

            # 轉成 normalized xyxy
            norm_boxes = []
            for (x1, y1, x2, y2) in xyxy:
                norm_boxes.append([x1 / W, y1 / H, x2 / W, y2 / H])

            boxes_list.append(norm_boxes)
            scores_list.append(scores)
            labels_list.append(labels)
```

```
        # ------ WBF ------
        boxes_wbf, scores_wbf, labels_wbf = weighted_boxes_fusion(
            boxes_list,
            scores_list,
            labels_list,
            # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
            weights=[1.4, 1.3, 1.1, 1.3, 1.2, 1.2],
            iou_thr=0.6,
            skip_box_thr=0.01,
        )
        MIN_SCORE = 0.1
        # ------ 寫回你原本格式 ------
        for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
            if score < MIN_SCORE:
                continue
            # 還原 pixel xyxy
            x1 = int(box[0] * W)
            y1 = int(box[1] * H)
            x2 = int(box[2] * W)
            y2 = int(box[3] * H)

            line = f"{filename_noext} {int(label)} {score:.4f} {x1} {y1} {x2} {y2}\n"
            fout.write(line)

fout.close()
print(f"Done! Ensemble result saved to: {output_txt}")
```

```
Total images = 8310
Done! Ensemble result saved to: ./predict_txt/images2_wbf.txt
```

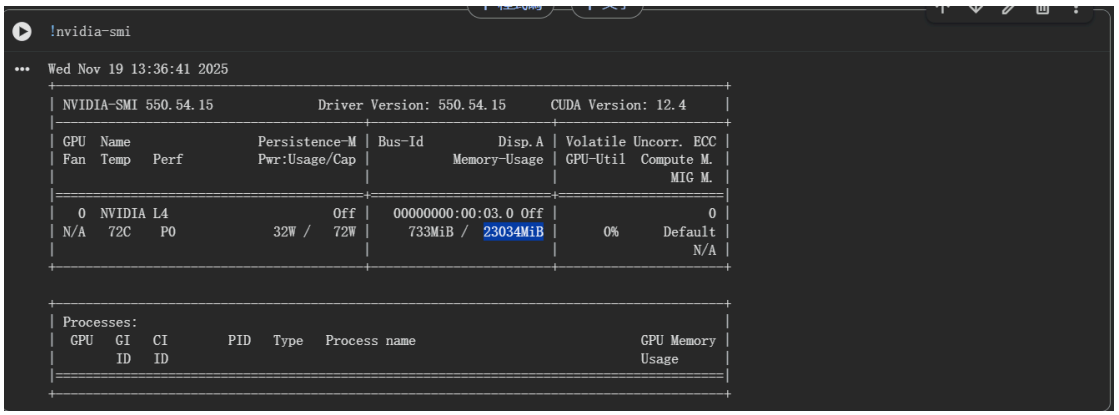
```
file1 = "./predict_txt/images1_wbf.txt"
file2 = "./predict_txt/images2_wbf.txt"
output = "./predict_txt/merged_wbf.txt"

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}")

from google.colab import files
files.download('/content/predict_txt/merged_wbf.txt')
```

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

3、執行環境

作業系統與硬體

- 平台：Linux-6.6.105+-x86_64-with-glibc2.35
- 作業系統：Linux 6.6.105+
- Python：3.12.12
- GPU 與驅動（nvidia-smi）：L4 , 23034MiB, 訓練: A100 GPU (colab)



- NVCC：Build cuda_12.5.r12.5/compiler.34385749_0

PyTorch / CUDA / cuDNN
- torch：2.8.0+cu124
- torchvision：0.23.0+cu124
- CUDA 可用：True
- CUDA 版本（torch）：12.4
- L4 (Colab GPU

Ultralytics / YOLO
- ultralytics：8.3.227
  ensemble-boxes : 1.0.9
主要套件（精簡清單）

套件	版本
ultralytics	8.3.227
ensemble-boxes : 1.0.9
torch	2.8.0+cu126
torchvision	0.23.0+cu126
opencv-python	4.12.0.88
pillow	11.3.0
numpy	2.0.2
pandas	2.2.2
albumentations	2.0.8
matplotlib	3.10.0
seaborn	0.13.2
scikit-learn	1.6.1

2、 模型設計與訓練策略
1、模型訓練原始程式碼與說明
github或Colab連結：

https://github.com/WCY91/AICUP_contest2_9104

2、模型訓練與預測的流程

透過採用Ultralytics 官方提供的 YOLOv12m 11m, 10m ,11n ,12n ,9c 為我的預訓練模型列表，選擇其原因是為了在預測中可以透過不同的模型架構，讓模型可以透過不同的角度去偵測，如:12系列有加入一些注意力相關的機制，11比較多為CNN based的。在訓練上也採用了小mosaic、以及採用了amp=True等。透過增強資料的隨機性跟變化使其模型能對不同角度與亮度有更高的精準度，並透過比較不同的epoch可以知道，因為資料集大小的問題，所以雖然在epoch設定為200時，在訓練可以到達0.98，但實際已經overfit了，最後經過不斷的finetune，選擇了epoch=85，當我的最終超參數，並設定close_mosaic因為想讓最後的結果可以比較穩定，並且避免因為過度插植導致圖片失真所以設定imgsz=640 (原本的圖片都是512*512)。

```
from ultralytics import YOLO
model = YOLO("yolo12n.pt")

model.train(
    data="./aortic_valve_colab.yaml",
    epochs=85,
    batch=32,
    imgsz=640,
    patience=40,
    device=0,
    close_mosaic=3,
    optimizer="AdamW",
    amp=True,
)
```

預測時:

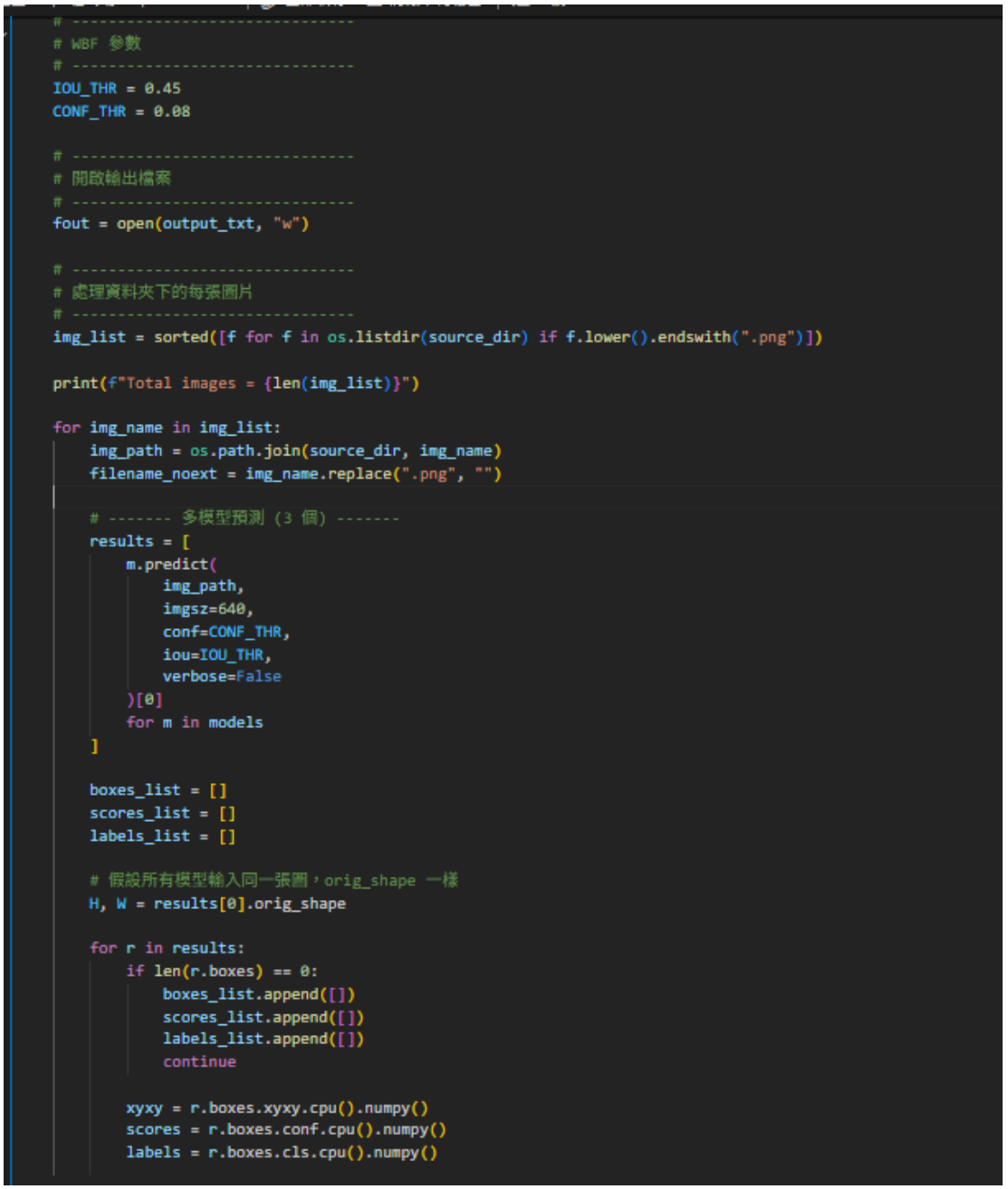
先將整個資料集分成兩堆:

```python
import os
import shutil

base_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)

# 自動找到第一個「直屬子資料夾含 patient*」的目錄
patient_root = base_root
for dirpath, dirnames, _ in os.walk(base_root):
    if any(d.lower().startswith("patient") for d in dirnames):
        patient_root = dirpath
        break

# 收集所有圖片路徑（只看直屬的 patient 資料夾）
all_files = []
for patient_folder in os.listdir(patient_root):
    patient_path = os.path.join(patient_root, patient_folder)
    if os.path.isdir(patient_path) and patient_folder.lower().startswith("patient"):
        for fname in os.listdir(patient_path):
            if fname.lower().endswith(".png"):
                all_files.append(os.path.join(patient_path, fname))

# 按名稱排序並對半移動
all_files.sort()
half = len(all_files) // 2

for f in all_files[:half]:
    shutil.move(f, os.path.join(dst_root1, os.path.basename(f)))

for f in all_files[half:]:
    shutil.move(f, os.path.join(dst_root2, os.path.basename(f)))

print(f"來源根目錄: {patient_root}")
print(f"完成移動! 總共 {len(all_files)} 張，前半 {half} 張到 images1，後半 {len(all_files)-half} 張到 images2")
```

```
來源根目錄: /content/datasets/test/tmp
完成移動! 總共 16620 張，前半 8310 張到 images1，後半 8310 張到 images2
```

然後透過ensemble-boxes這個套件，此套件可以用在輸出的結果融合，因為在Ultralytics其實在以前的時候，有提供給定List的模型列表，讓其在model predict可以自動融合，但因為現在這個方法並不work，所以採用了WBF去手動融合:

透過先讓不同的模型各自預測出結果並將其放在一個list:



```
# ------------------------------
# WBF 參數
# ------------------------------
IOU_THR = 0.45
CONF_THR = 0.08

# ------------------------------
# 開啟輸出檔案
# ------------------------------
fout = open(output_txt, "w")

# ------------------------------
# 處理資料夾下的每張圖片
# ------------------------------
img_list = sorted([f for f in os.listdir(source_dir) if f.lower().endswith(".png")])

print(f"Total images = {len(img_list)}")

for img_name in img_list:
    img_path = os.path.join(source_dir, img_name)
    filename_noext = img_name.replace(".png", "")

    # ------- 多模型預測 (3 個) -------
    results = [
        m.predict(
            img_path,
            imgsz=640,
            conf=CONF_THR,
            iou=IOU_THR,
            verbose=False
        )[0]
        for m in models
    ]

    boxes_list = []
    scores_list = []
    labels_list = []

    # 假設所有模型輸入同一張圖，orig_shape 一樣
    H, W = results[0].orig_shape

    for r in results:
        if len(r.boxes) == 0:
            boxes_list.append([])
            scores_list.append([])
            labels_list.append([])
            continue

        xyxy = r.boxes.xyxy.cpu().numpy()
        scores = r.boxes.conf.cpu().numpy()
        labels = r.boxes.cls.cpu().numpy()
```

且這邊會設定IOU_THR = 0.45，以及CONF=0.08，會設定比較低是因為後面融合時可以有比較多的輸入，因為後續還會再濾一遍，故希望這邊可以有比較多的可能的結果避免在前期因為標準訂太高，導致在融合時輸入不夠，之後透過ensemble-boxes API 去透過weight作結果的加總，並判斷是否超過iou_thr，並且在最終的時候還有再判斷一

次score，判斷其是否有大於min_score，最終將結果去合併

```
    # ------- WBF -------
    boxes_wbf, scores_wbf, labels_wbf = weighted_boxes_fusion(
        boxes_list,
        scores_list,
        labels_list,
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.1,1.3,1.2,1.2],
        iou_thr=0.6,
        skip_box_thr=0.01,
    )
    MIN_SCORE = 0.1
    # ------- 寫回你原本格式 -------
    for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
        if score < MIN_SCORE:
            continue
        # 還原 pixel xyxy
        x1 = int(box[0] * W)
        y1 = int(box[1] * H)
        x2 = int(box[2] * W)
        y2 = int(box[3] * H)

        line = f"{filename_noext} {int(label)} {score:.4f} {x1} {y1} {x2} {y2}\n"
        fout.write(line)

fout.close()
print(f"Done! Ensemble result saved to: {output_txt}")
```

到了第二堆其實概念一樣的只是當時的模型列表順序有變以及weight也有給不同，以及在第二堆有特別給不同的min_score如下圖:

```
# --------------------------------
# 設定模型與資料夾
# --------------------------------
model_paths = [
    "/content/yolo12m95.pt",          # 模型1：12m
    "/content/yolo11m95.pt",          # 模型3：11m
    "/content/yolo9c.pt",           # 模型3：11m
    "/content/yolo12n.pt",           # 模型3：11m
    "/content/yolo11n.pt",           # 模型3：11m
    "/content/yolo10m_a.pt",          # 模型3：11m
]
```

```
    for r in results:
        if len(r.boxes) == 0:
            boxes_list.append([])
            scores_list.append([])
            labels_list.append([])
            continue

        xyxy = r.boxes.xyxy.cpu().numpy()
        scores = r.boxes.conf.cpu().numpy()
        labels = r.boxes.cls.cpu().numpy()

        # 轉成 normalized xyxy
        norm_boxes = []
        for (x1, y1, x2, y2) in xyxy:
            norm_boxes.append([x1 / W, y1 / H, x2 / W, y2 / H])

        boxes_list.append(norm_boxes)
        scores_list.append(scores)
        labels_list.append(labels)

    # ------- WBF -------
    boxes_wbf, scores_wbf, labels_wbf = weighted_boxes_fusion(
        boxes_list,
        scores_list,
        labels_list,
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.2,1.2,1.2,1.2],
        iou_thr=0.5,
        skip_box_thr=0.01,
    )
    MIN_SCORE = 0.088
    # ------- 寫回你原本格式 -------
    for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
        if score < MIN_SCORE:
            continue
        # 還原 pixel xyxy
        x1 = int(box[0] * W)
        y1 = int(box[1] * H)
        x2 = int(box[2] * W)
        y2 = int(box[3] * H)

        line = f"{filename_noext} {int(label)} {score:.4f} {x1} {y1} {x2} {y2}\n"
        fout.write(line)

fout.close()
print(f"Done! Ensemble result saved to: {output_txt}")
```

3、參數設定

訓練參數 Epochs=85, batch=32, imgsz=640,cose_mosaic = 3 ,amp=True , optimizer = AdamW。

predict:
第一堆的MIN_SCORE 為0.088 第二堆為0.1
第一堆:

```
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.2,1.2,1.2,1.2],
        iou_thr=0.5,
        skip_box_thr=0.01,
```

weights =[1.4,1.3,1.2,1.2,1.2,1.2]
iou_thr = 0.5,
min_score = 0.088

```
model_paths = [
    "/content/yolo12m95.pt",          # 模型1：12m
    "/content/yolo11m95.pt",          # 模型3：11m
    "/content/yolo9c.pt",          # 模型3：11m
```

```
    "/content/yolo10m_a.pt",          # 模型3：11m
    "/content/yolo12n.pt",          # 模型3：11m
    "/content/yolo11n.pt",          # 模型3：11m
]

# ------------------------------
# WBF 參數
# ------------------------------
IOU_THR = 0.45
CONF_THR = 0.08

    # ------- WBF -------
    boxes_wbf, scores_wbf, labels_wbf =
weighted_boxes_fusion(
        boxes_list,
        scores_list,
        labels_list,
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.2,1.2,1.2,1.2],
        iou_thr=0.5,
        skip_box_thr=0.01,
    )
    MIN_SCORE = 0.088
    # ------- 寫回你原本格式 -------
    for box, score, label in zip(boxes_wbf, scores_wbf,
labels_wbf):
        if score < MIN_SCORE:
            continue
        # 還原 pixel xyxy
        x1 = int(box[0] * W)
        y1 = int(box[1] * H)
        x2 = int(box[2] * W)
        y2 = int(box[3] * H)

        line = f"{filename_noext} {int(label)}
{score:.4f} {x1} {y1} {x2} {y2}\n"
        fout.write(line)
```

第二堆:
weights = [1.4,1.3,1.1,1.3,1.2,1.2]
iou_thr = 0.6 ,
min_score = 0.1

```python
# ------------------------------
# 設定模型與資料夾
# ------------------------------
model_paths = [
    "/content/yolo12m95.pt",          # 模型1：12m
    "/content/yolo11m95.pt",          # 模型3：11m
    "/content/yolo9c.pt",           # 模型3：11m
    "/content/yolo10m_a.pt",          # 模型3：11m
    "/content/yolo12n.pt",          # 模型3：11m
    "/content/yolo11n.pt",          # 模型3：11m
]
```

```python
# ------------------------------
# WBF 參數
# ------------------------------
IOU_THR = 0.45
CONF_THR = 0.08
```

```python
    # ------- WBF -------
    boxes_wbf, scores_wbf, labels_wbf = weighted_boxes_fusion(
        boxes_list,
        scores_list,
        labels_list,
        # 這裡可以調權重，例如 [2, 2, 1] 讓兩個 12m 比較重
        weights=[1.4,1.3,1.1,1.3,1.2,1.2],
        iou_thr=0.6,
        skip_box_thr=0.01,
    )
    MIN_SCORE = 0.1
    # ------- 寫回你原本格式 -------
    for box, score, label in zip(boxes_wbf, scores_wbf, labels_wbf):
        if score < MIN_SCORE:
            continue
        # 還原 pixel xyxy
        x1 = int(box[0] * W)
        y1 = int(box[1] * H)
        x2 = int(box[2] * W)
        y2 = int(box[3] * H)

        line = f"{filename_noext} {int(label)} {score:.4f} {x1} {y1} {x2} {y2}\n"
        fout.write(line)
```

最後都是imgsz=640與訓練時用一樣的。
其餘使用 ultralytics 預設參數
https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/default.yaml。

3、 分析與結論

在這次比賽中，我最大的感受是 YOLO系列 在醫學影像上的表現真的超乎我預期。剛開始只用預訓練模型測試時，成績就已經不錯，讓我蠻意外的，後來開始跑訓練後發現模型收斂速度其實很快，大概在 60～70 epoch 結果就已經穩定，後面再訓練比較像是在微調參數而不是大幅提升，而且如果epoch過大反而會因為太偏向訓練集導致在測試上不是很好。

另外這次我覺得資料真的比模型還重要，尤其是醫學影像這種任務，因為我一開始沒有分成兩個階段去拆分訓練跟驗證，導致所有的模型都沒有看過41~50的資料，在ensemble上也很難再往上提升。此外在推論階段我也發現 conf、iou 的設定對結果影響很大，調太嚴格會少抓東西，太寬鬆又會變得很亂，所以要花時間慢慢試，最後用多模型加ensemble-boxes 融合後分數有明顯提升，也證明 ensemble 對這種影像偵測任務是有效的，只是需要確定資料的預測輸出一致性。

整體來說，這次比賽讓我更熟悉 YOLO 的訓練流程，也學到從資料處理、模型訓練到後處理融合，每一步都會影響最終結果。雖然還有很多地方可以做得更好，但在這次比賽中拿到還不錯的排名，我蠻開心的，是很棒一次的經驗。

4、 使用的外部資源與參考文獻

1、”Yolov12: Attention-centric real-time object detectors”Tian, Y., Ye, Q., & Doermann, D. (2025). Yolov12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*.

2、”ultralytics 套件”https://github.com/ultralytics/ultralytics

3. “ensemble boxes”套件
https://pypi.org/project/ensemble-boxes/