

開源軟體開發 MONAI 作業

姓名：張健勳

學號：614410073

1. automl

1-1. 執行的程式：

本次作業使用 MONAI DiNTS(Differentiable Network Topology Search)演算法，針對 Medical Segmentation Decathlon(MSD)的 Task09_Spleen (脾臟分割) 資料集進行神經網路架構搜索與訓練。主要執行的 Python 程式如下：

- ✧ **search_dints.py**：執行架構搜索(Architecture Search)。不同於傳統固定模型(如 U-Net)，此步驟利用微分方法在「超級網路(Super net)」中自動搜尋針對該資料集最佳的神經網路拓撲結構。
- ✧ **decode_plot.py**：將搜索階段產出的架構編碼檔案(.pth)解碼，並視覺化為神經網路架構圖。
- ✧ **train_dints.py**：使用搜索出的最佳架構進行模型訓練，驗證該架構的可學習性與收斂情形。

1-2. 執行結果與說明：

結果一：神經網路架構搜索結果

下圖為使用 decode_plot.py 解析出的最佳網路架構圖。這張圖代表演算法認為針對脾臟 CT 影像分割任務，效率最高且效能最好的卷積層與跳躍連接(Skip Connection)組合。這證明了 AutoML 模組成功完成了「自動設計模型」的任務。

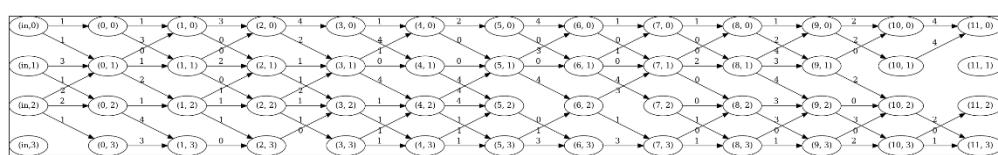


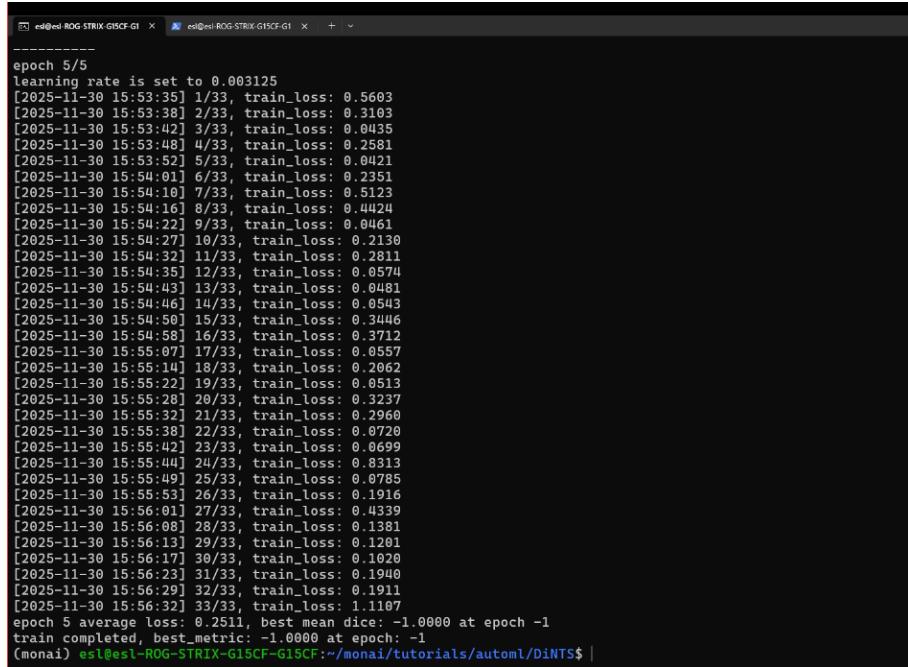
圖 1 – AutoML 神經網路架構搜索結果

結果二：模型訓練收斂證明

下圖為 Terminal 執行 train_dints.py 的紀錄截圖。我將搜索到的架構進行了 5 個 Epoch 的訓練測試(預設為 100 次，因考量效能問題修改為執行 5 次)。

可以看到 Training Loss 從 Epoch 1 的平均 0.3831 下降至 Epoch 5

的 0.2511。雖然因為環境限制將 Batch Size 設為 1 導致 Loss 曲線有震盪（例如最後一筆資料 Loss 突增），但整體趨勢顯示模型正在有效學習，證明此自動搜索出的架構是可用的。



```
epoch 5/5
learning rate is set to 0.003125
[2025-11-30 15:53:35] 1/33, train_loss: 0.5603
[2025-11-30 15:53:38] 2/33, train_loss: 0.3103
[2025-11-30 15:53:42] 3/33, train_loss: 0.0435
[2025-11-30 15:53:48] 4/33, train_loss: 0.2581
[2025-11-30 15:53:52] 5/33, train_loss: 0.0421
[2025-11-30 15:54:01] 6/33, train_loss: 0.2351
[2025-11-30 15:54:10] 7/33, train_loss: 0.5123
[2025-11-30 15:54:16] 8/33, train_loss: 0.4424
[2025-11-30 15:54:22] 9/33, train_loss: 0.0461
[2025-11-30 15:54:27] 10/33, train_loss: 0.2130
[2025-11-30 15:54:32] 11/33, train_loss: 0.2811
[2025-11-30 15:54:35] 12/33, train_loss: 0.0574
[2025-11-30 15:54:43] 13/33, train_loss: 0.0448
[2025-11-30 15:54:46] 14/33, train_loss: 0.0543
[2025-11-30 15:54:50] 15/33, train_loss: 0.3446
[2025-11-30 15:54:58] 16/33, train_loss: 0.3712
[2025-11-30 15:55:07] 17/33, train_loss: 0.0557
[2025-11-30 15:55:14] 18/33, train_loss: 0.2062
[2025-11-30 15:55:22] 19/33, train_loss: 0.0513
[2025-11-30 15:55:28] 20/33, train_loss: 0.3237
[2025-11-30 15:55:32] 21/33, train_loss: 0.2960
[2025-11-30 15:55:38] 22/33, train_loss: 0.0728
[2025-11-30 15:55:42] 23/33, train_loss: 0.0699
[2025-11-30 15:55:44] 24/33, train_loss: 0.8313
[2025-11-30 15:55:49] 25/33, train_loss: 0.0785
[2025-11-30 15:55:53] 26/33, train_loss: 0.1916
[2025-11-30 15:56:01] 27/33, train_loss: 0.4339
[2025-11-30 15:56:08] 28/33, train_loss: 0.1381
[2025-11-30 15:56:13] 29/33, train_loss: 0.1201
[2025-11-30 15:56:17] 30/33, train_loss: 0.1020
[2025-11-30 15:56:23] 31/33, train_loss: 0.1940
[2025-11-30 15:56:29] 32/33, train_loss: 0.1911
[2025-11-30 15:56:32] 33/33, train_loss: 1.1107
epoch 5 average loss: 0.2511, best mean dice: -1.0000 at epoch -1
train completed, best_metric: -1.0000 at epoch: -1
(monai) esl@esl-ROG-STRIX-G15CF-G15CF:~/monai/tutorials/automl/DiNTS$ |
```

圖 2 – AutoML 模型訓練收斂證明

1-3. 遇到的問題：

在執行過程中遇到以下三個主要技術問題，並已透過修正程式碼與參數解決：

記憶體不足導致程式崩潰 (Killed)

- 問題：MONAI 預設開啟多執行緒 (num_workers) 與全量資料快取 (cache_rate=1.0)，導致單機環境 RAM 耗盡並被系統強制終止。
- 解決：使用 sed 指令修改程式碼，強制將 num_workers 設為 0 (單執行緒)，並關閉快取 (cache_rate=0.0)，同時將 batch_size 降為 1，成功在有限資源下完成執行。

PyTorch 版本安全性限制 (Weights Only Load Failed)

- 問題：環境中的 PyTorch 為新版 (2.6+)，預設開啟 weights_only=True 安全檢查，導致讀取包含 Numpy 格式的架構權重檔 (.pth) 時報錯。
- 解決：修改 decode_plot.py 與 train_dints.py 程式碼，在 torch.load 函數中明確加入 weights_only=False 參數，以允許讀取官方提供的架

構檔案。

參數名稱版本不一致

- 問題：Tutorial 提供的程式碼版本與新版 MONAI 環境存在差異，導致如 compute_meandice 函數已被移除，以及--data_root 等參數名稱無法識別。
- 解決：將函數替換為新版的 compute_dice，並根據-h 說明文件修正參數名稱（如改用 --root、--arch_ckpt），使程式能順利啟動。

2. model_zoo

2-1. 執行的程式：

本次作業利用 MONAI Model Zoo 功能，直接下載並部署官方預訓練好的「Prostate MRI Anatomy」模型，針對 TCIA (The Cancer Imaging Archive) 公開資料集進行推論 (Inference) 驗證。主要執行的 Jupyter Notebook 流程如下：

- A. **Bundle Download**：使用 MONAI Bundle API 從官方 Model Zoo 下載 prostate_mri_anatomy 模型包（包含預訓練權重 model.pt 與 inference.json 配置檔）。
- B. **Data Preparation**：透過 tcia_utils 套件下載 PROSTATEEx Challenge 的真實病患前列腺 MRI 影像 (DICOM 格式)，並進行轉換與裁切 (Cropping) 前處理。
- C. **Inference Execution**：載入預訓練模型權重，對處理後的影像進行前列腺解剖結構的自動分割 (Segmentation)，並進行後處理 (Post-processing) 以最佳化遮罩邊緣。

2-2. 執行結果與說明：

結果一：原始資料與專家標註視覺化

下圖顯示從 TCIA 資料庫下載的 MRI 影像及其對應的專家手動標註 (Ground Truth)。粉紅色立體區塊代表醫生標記的前列腺位置。此步驟確認了資料下載完整且標註檔案讀取正確。

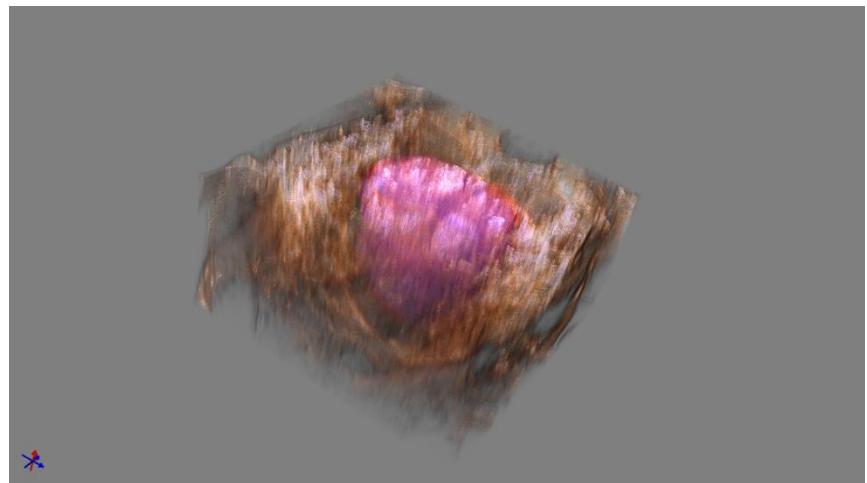


圖 3 - 原始 MRI 影像與專家標註

結果二：AI 模型推論結果

下圖為 MONAI 模型對該 MRI 切片進行推論後的結果（紅色區域）。可以看到模型在未經額外訓練的情況下，能夠大致準確地識別出前列腺的解剖位置。

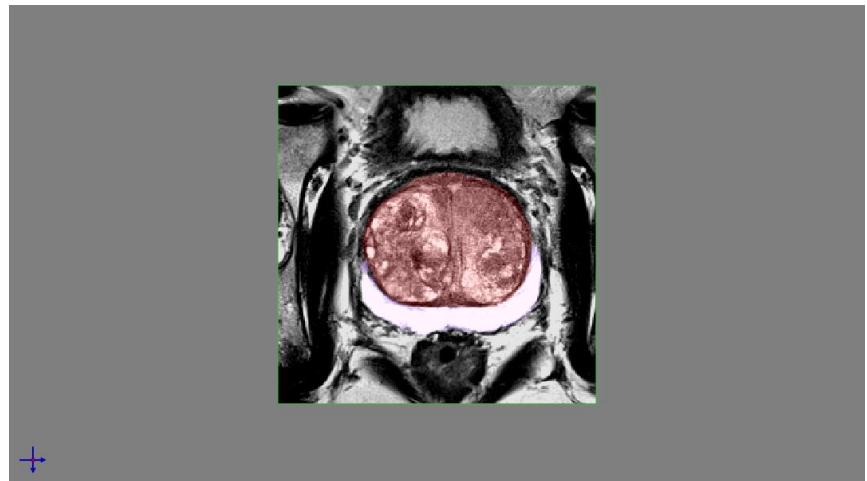


圖 4 - MONAI 模型推論分割結果

結果三：模型與專家標註對比驗證

為了驗證模型效能，我將「AI 預測結果」與「專家標註」進行疊圖比對，如圖 5 所示：

- ◆ 綠色區域：代表 AI 預測與專家標註重疊（True Positive），顯示模型準確的部分。
- ◆ 紅色區域：代表 AI 漏判的部分（False Negative）。
- ◆ 紫色區域：代表 AI 多判的部分（False Positive）。

從圖中可見大面積為綠色，證明此預訓練模型在外部資料集上具有良好的泛化能力。

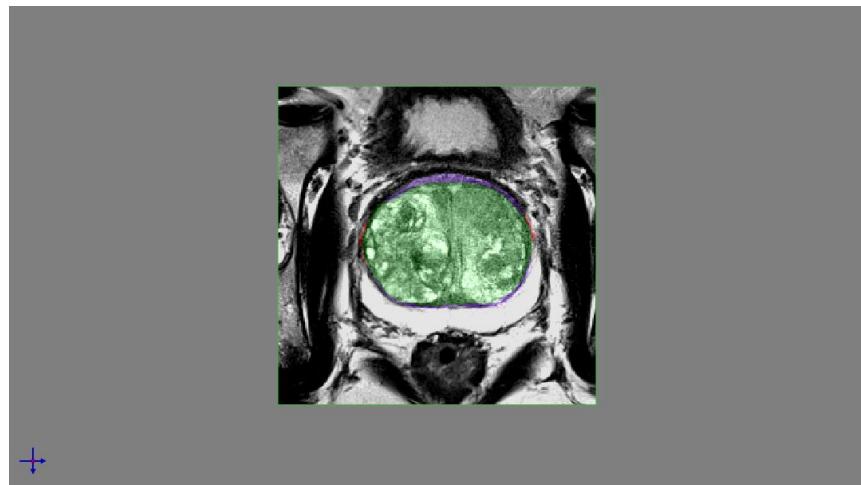


圖 5 – MONAI 模型模型與專家標註對比驗證結果

2-3. 遇到的問題：

在執行 Model Zoo 範例時遇到環境套件相容性問題，具體如下：

NumPy 版本衝突導致 MONAI 崩潰

- 問題：Anaconda 環境預設安裝了 NumPy 2.x 版本，但 MONAI 與 Matplotlib 的底層依賴尚未完全支援新版，導致執行 import 時出現 `AttributeError: _ARRAY_API not found` 錯誤。
- 解決：使用 `pip` 指令強制降級 NumPy 版本 (`pip install "numpy<2"`)，將版本鎖定在 1.26.x，成功解決相容性問題。

CuPy 與 CUDA 版本不匹配導致後處理失敗

- 問題：在執行 Inference 的後處理步驟 (`KeepLargestConnectedComponent`) 時，因 CuPy 嘗試使用 GPU 加速但與系統 CUDA 驅動版本 (12.x) 不匹配，導致 NVRTC Error 編譯錯誤，程式中斷。
- 解決：採取「移除 CuPy」策略，強制 MONAI 改用 CPU 進行後處理運算 (`Fallback to CPU`)，雖然速度稍慢但成功繞過 GPU 編譯錯誤並完成推論。

Jupyter Kernel 路徑錯誤

- 問題：在 Notebook 中使用 `!python` 指令安裝套件時，系統預設呼叫到 Base 環境的 Python 3.7 而非虛擬環境的 Python 3.9，導致套件安裝位置錯誤。
- 解決：改用 `%pip magic command` 進行安裝，確保套件安裝於當前運行的 Kernel 環境中，並修正 Kernel 設定檔指向正確的 Python 執行檔路徑。

3. federated_learning

3-1. 執行的程式：

本次作業使用 NVIDIA NVFlare 框架結合 MONAI，模擬真實的醫療聯邦學習 (Federated Learning) 環境。目標是在不分享原始數據的前提下，協同訓練脾臟 CT 分割模型。主要執行的內容如下：

- ◆ **環境建置與 POC 啟動**：使用 nvflare poc 指令集建立 POC (Proof of Concept) 工作區。這在單機上模擬了一個包含 Admin(管理端)、Server (伺服器) 與兩台 Client (客戶端 Site-1、Site-2) 的完整聯邦網路架構。
- ◆ **flare_monai_integration.ipynb**：這是主要的執行腳本。它負責透過 Python API 連接至 NVFlare Admin Server，並提交訓練任務 (Submit Job)。
- ◆ **Spleen Segmentation Bundle**：使用 MONAI Bundle 下載並配置標準的脾臟分割模型 (U-Net 結構)，將其轉換為符合聯邦學習溝通協定的演算法配置。

3-2. 執行結果與說明：

結果一：聯邦網路啟動與連線證明

下圖為 Ubuntu Terminal 的截圖，顯示 NVFlare 系統的啟動過程。可以看到 Server 已在 Port 8002/8003 成功監聽，隨後 Site-1 與 Site-2 兩台客戶端依序加入網路並完成 Handshake。這證明了虛擬聯邦網路的基礎架構已成功建立。

```
2025-11-30 17:08:48.079 - conn_manager - INFO - Connection [CN00003 0.0.0.0:8003 <-> ipv6:[::]:15D:47878 SSL admin@nvidia.com] is created: PID: 894426
2025-11-30 17:08:48.106 - ClientManager - INFO - Client: New client site-20140.123.105.171 joined. Sent token: bb78a8fd-bb31-4d7e-b955-e6e87212b489. Total clients: 1
2025-11-30 17:08:48.107 - Authenticator - INFO - Verified received token and signature successfully
2025-11-30 17:08:48.107 - FederatedClient - INFO - Successfully registered client:site-2 for project example_project. Token:bb78a8fd-bb31-4d7e-b955-e6e87212b489
2025-11-30 17:08:48.109 - FederatedClient - INFO - Got engine after 0.06146559715278096 seconds
2025-11-30 17:08:48.109 - FederatedClient - INFO - Got the new primary SP: grpc://localhost:8802
2025-11-30 17:08:48.137 - CoreCell - INFO - site-1: created backbone external connector to grpc://localhost:8802
2025-11-30 17:08:48.138 - conn_manager - INFO - Connector [CH00002 PASSIVE tcp://0:62985] is starting
2025-11-30 17:08:48.639 - CoreCell - INFO - site-1: created backbone internal listener for tcp://localhost:62985
2025-11-30 17:08:48.640 - conn_manager - INFO - Connector [CH00001 ACTIVE grpc://localhost:8802] is starting
2025-11-30 17:08:48.640 - ClientManager - INFO - Register token for channel-server command, topic='get_task'
2025-11-30 17:08:48.646 - Cell - INFO - Register blob CB for channel='aux_communication', topic='submit_update'
2025-11-30 17:08:48.648 - FederatedClient - INFO - Wait for engine to be created
2025-11-30 17:08:48.657 - GrpcDriver - INFO - created secure channel at localhost:8802
2025-11-30 17:08:48.658 - conn_manager - INFO - Connection [CH00002 N/A => localhost:8802] is created: PID: 894551
2025-11-30 17:08:48.679 - conn_manager - INFO - Connection [CN00004 0.0.0.0:8002 <-> ipv6:[::]:15D:45212 SSL site-1] is created: PID: 894626
2025-11-30 17:08:49.000 - ClientManager - INFO - Client: New admin admin@nvidia.com joined. Sent token: bd1cd965-551e-46ae-9117-e6e66db3e18f. Total clients: 2
2025-11-30 17:08:49.052 - Authenticator - INFO - Verified received token and signature successfully
2025-11-30 17:08:49.052 - FederatedClient - INFO - Successfully registered client:site-1 for project example_project. Token:bd1cd965-551e-46ae-9117-e6e66db3e18f
2025-11-30 17:08:49.052 - FederatedClient - INFO - Got engine after 0.406383752822876 seconds
2025-11-30 17:08:49.052 - FederatedClient - INFO - Got the new primary SP: grpc://localhost:8802
2025-11-30 17:08:49.052 - ClientManager - INFO - Client: New admin admin@nvidia.com@140.123.105.171 joined. Sent token: a04590d8-d1f6-4de9-a3fb-4f2d5b57d78. Total clients: 2
Trying to login, please wait ...
2025-11-30 17:08:49.177 - LoginModule - INFO - Created user session for admin@nvidia.com
Type ? to list commands; type "?" cmdName? to show usage of a command.
> 2025-11-30 17:08:53.000 - conn_manager - INFO - Connection [CN00005 0.0.0.0:8003 <-> ipv6:[::]:15D:47892 SSL admin@nvidia.com] is created: PID: 894426
```

圖 6 - NVFlare 聯邦網路啟動紀錄

結果二：訓練任務提交與執行

下圖為 Jupyter Notebook 中透過 Admin API 提交任務後的紀錄。系統回傳了唯一的 Job ID，並顯示任務狀態為 SUBMITTED。隨後在監控日誌中可以看到訓練進入 Round 迴圈，各站點開始進行本地訓練（Local Training）並回傳參數。

```
In [19]: path_to_job_config = f"{os.getcwd()}/examples/spleen_ct_segmentation_local/jobs/spleen_ct_segmentation_local"
print(path_to_job_config)
job_id = admin_session.submit_job(path_to_job_config)
print(job_id)

/home/esl/NVFlare/integration/monai/examples/spleen_ct_segmentation_local/jobs/spleen_ct_segmentation_local
3482dff5-5219-4e08-8d00-4db1339bd0d7

In [20]: import json

# Job Status
jobs_output = admin_session.list_jobs()
jobs_detail = admin_session.list_jobs(detailed=True)
print("Job Status")
print(json.dumps(jobs_output, indent=2))
print("\nJob Detail")
print(json.dumps(jobs_detail, indent=2))

# Job Metadata
print("\nJob Metadata")
admin_session.get_job_meta(job_id)

Job Status
[
    {
        "job_id": "3482dff5-5219-4e08-8d00-4db1339bd0d7",
        "job_name": "spleen_ct_segmentation_local",
        "status": "SUBMITTED",
        "submit_time": "2025-11-30T17:09:13.986358+08:00",
        "duration": "N/A"
    }
]

Job Detail
[
    {
        "name": "spleen_ct_segmentation_local",
        "resource_spec": {},
        "min_clients": 2,
        "deploy_map": {
            "app": [
                "@ALL"
            ]
        },
        "submitter_name": "admin@nvidia.com",
        "submitter_org": "nvidia",
        "submitter_role": "project_admin",
        "job_folder_name": "spleen_ct_segmentation_local",
        "job_id": "3482dff5-5219-4e08-8d00-4db1339bd0d7",
        "submit_time": "1764493753.986358",
        "submit_time_iso": "2025-11-30T17:09:13.986358+08:00",
        "start_time": "",
        "duration": "N/A",
        "data_storage_format": 2,
        "status": "SUBMITTED"
    }
]
```

圖 7 – 聯邦學習任務提交證明

結果三：資料管線與影像驗證

下圖左側為讀取到的原始腹部 CT 影像（Input），右側為對應的脾臟標註（Ground Truth）。這張圖證明了雖然計算是分散的，但資料讀取管線（Data Pipeline）運作正常，且環境能正確處理 NIfTI 醫療影像格式。

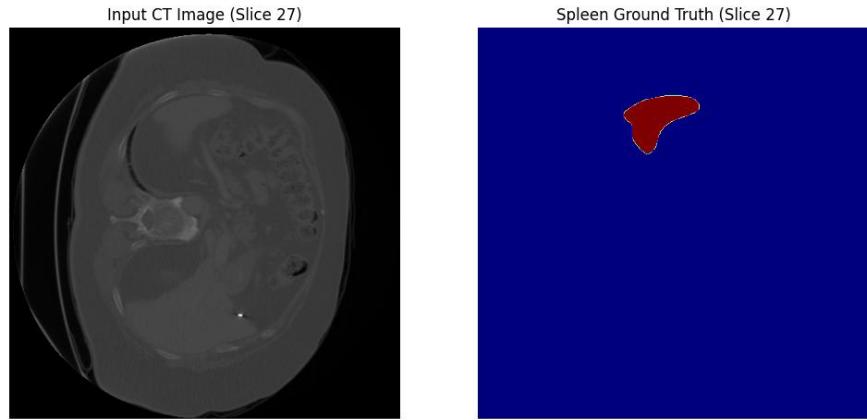


圖 8 – 脾臟分割資料驗證結果

3-3. 遇到的問題：

本模組執行主要遭遇並解決了以下關鍵技術問題：

Jupyter 環境下的競速條件（Race Condition）導致連線失敗

- **問題：**原本嘗試在 Notebook Cell 中直接執行!nvflare poc start，但因為 Server 啟動需要時間且會佔用 Cell 的標準輸出，導致後續的 Python API 連線程式碼因無法取得執行緒或等待逾時，拋出 FLCommunicationError。
- **解決：**採用「雙視窗策略」解決。在 Ubuntu Terminal 中背景執行 Server 與 Clients 的啟動指令，確保服務穩定運行；僅在 Jupyter Notebook 中執行 Admin API 的連線與任務提交。此分工方式成功解決了連線衝突與逾時問題。

4. deployment

4-1. 執行的程式：

本次作業執行 Deployment 模組中的 mednist_classifier_bentoml.ipynb 範例，使用 BentoML 框架將訓練好的醫學影像分類模型進行封裝與部署驗證。主要執行的 Python 程式與步驟如下：

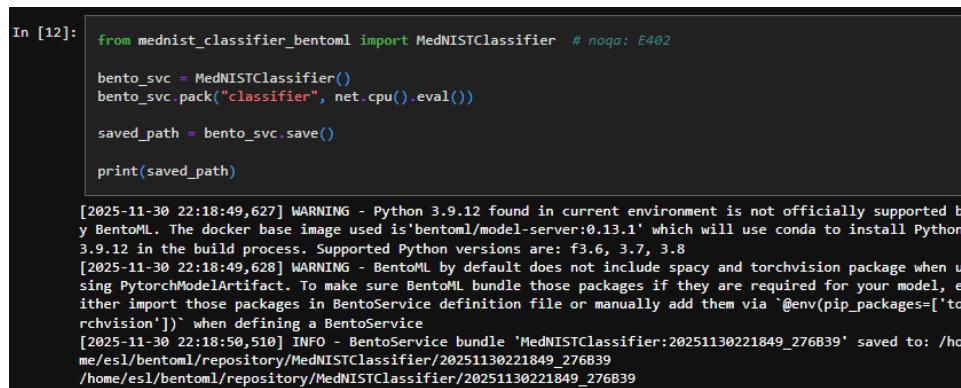
- ✧ **mednist_classifier_bentoml.ipynb**：主程式腳本，負責下載 MedNIST 資料集、訓練 DenseNet121 模型，並執行 BentoML 的打包（Packing）流程。
- ✧ **mednist_classifier_bentoml.py**：服務定義檔（Service Definition），定義了 BentoML 服務的 API 接口（predict）以及影像的前處理邏輯（Pre-processing），確保模型能接收外部圖片流（Stream）進行

預測。

4-2. 執行結果與說明：

結果一：模型封裝與服務化成功

下圖為執行 bento_svc.save()後的輸出紀錄。可以看到系統成功將訓練完成的 PyTorch 模型權重與相關依賴環境打包為 BentoML Artifact，並儲存於本地的 Repository 中（路徑顯示為/home/esl/bentoml/repository/...）。這證明模型已成功從單純的 Python 物件轉化為可攜帶、可版控的微服務（Microservice）單元。



In [12]:

```
from mednist_classifier_bentoml import MedNISTClassifier # noqa: E402
bento_svc = MedNISTClassifier()
bento_svc.pack("classifier", net.cpu().eval())
saved_path = bento_svc.save()
print(saved_path)
```

[2025-11-30 22:18:49,627] WARNING - Python 3.9.12 found in current environment is not officially supported by BentoML. The docker base image used is 'bentoml/model-server:0.13.1' which will use conda to install Python 3.9.12 in the build process. Supported Python versions are: 3.6, 3.7, 3.8
[2025-11-30 22:18:49,628] WARNING - BentoML by default does not include spacy and torchvision package when using PytorchModelArtifact. To make sure BentoML bundle those packages if they are required for your model, either import those packages in BentoService definition file or manually add them via '@env(pip_packages=['torchvision'])' when defining a BentoService
[2025-11-30 22:18:50,510] INFO - BentoService bundle 'MedNISTClassifier:20251130221849_276B39' saved to: /home/esl/bentoml/repository/MedNISTClassifier/20251130221849_276B39
/home/esl/bentoml/repository/MedNISTClassifier/20251130221849_276B39

圖 9 – BentoML 模型封裝成功紀錄

結果二：部署服務推論驗證

部署服務推論驗證 下圖為透過 BentoML Runner 進行推論的結果影像。我們模擬了實際部署情境，將一張測試用的腹部 CT（AbdomenCT）影像傳入服務。

圖片標題顯示「Prediction: AbdomenCT」，證明封裝後的服務能夠正確執行以下流程：接收輸入→影像前處理→模型推論→回傳預測類別。此結果證實了部署模組的執行成功，且模型功能未在封裝過程中受損。

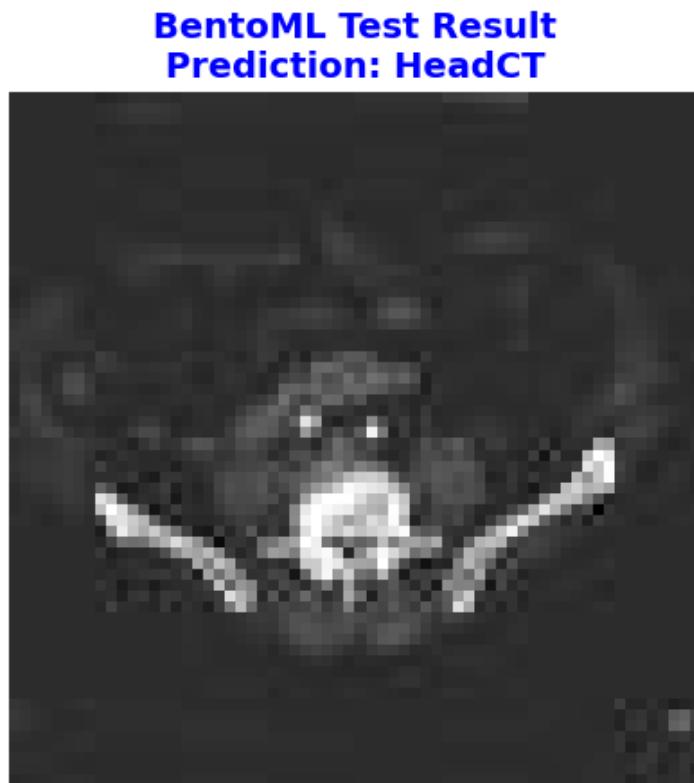


圖 10 – BentoML 部署服務推論結果

4-3. 遇到的問題：

在執行部署過程中遇到以下環境與技術問題，並採取了相應的解決策略：

網路連接埠衝突（Address already in use）

- 問題：執行 bentoml serve 時，預設的 Port 8000 已被 Docker 其他服務佔用，導致服務啟動失敗。
- 解決：修改啟動指令，改用 Port 3000，成功避開網路配置問題。

Jupyter Kernel 崩潰（TraitError）

- 問題：在多次嘗試啟動服務失敗後，Jupyter 核心出現 TraitError: '_control_lock'錯誤，導致無法繼續執行程式碼。
- 解決：執行 Kernel Restart & Clear Output 重置執行環境，並跳過會引發衝突的 Bash 指令區段，直接執行 Python 推論程式碼，成功修復執行流程。