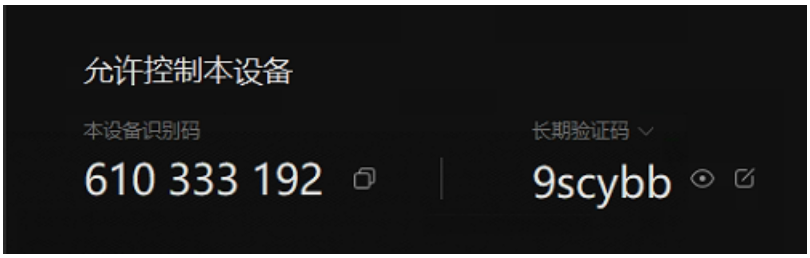
CMR数据处理全流程

一、将原始dicom文件处理成nifti

处理的数据和代码全部在台式机。

台式机登录方式：向日葵远程



台式机锁屏密码：ai-medicine123

处理数据的代码路径：C:\Users\Administrator\PycharmProjects\data_preprocess.py

数据路径：I:\CMR-China-new

接下来几个步骤可以分步运行，每一步没问题之后再进行下一步。

1、对原始数据预处理

```
## 1. process raw dicom
rename_dicom(root_path)

resample_dataset(root_path, new_path)

fix_seriesnum(new_path)
```

rename_dicom：下载来的源数据可能是dcm格式，也可能是无后缀的文件。如果无后缀，就运行rename_dicom，如果已经是dcm文件就跳过。

resample_dataset：将dcm文件按照格式存放到另一个文件夹new_path，**必要步骤**。

fix_seriesnum：固定dcm文件的slice数、frame数等，根据数据情况运行。

2、从dcm文件中提取三个模态的nifti

```
## 2. save dicom to nifti
save_sax_cine(new_path, save_path)
# depends on overlay type
# save_cine_with_overlay(new_path, save_path)
save_4ch_cine(new_path, save_path)
save_lge(new_path, save_path)
```

每个模态是根据dcm文件中的file tag、len(files)等信息筛选的，不同源的cmr数据格式不统一，**这一步需要根据dcm的保存格式调整代码**。（比如sax cine的文件名一般包含‘sax’、‘sa’等，4ch cine包含‘4ch’、‘4c’等，LGE包含‘psir’等字符，各模态的文件名互斥。）

建议先运行一遍，再看提取不成功的病例是什么情况，调整代码的筛选条件，再重新提取。

分割标签：有些数据可能有分割标签，可能包含在原图的dcm文件中（以overlay的tag存储，参考save_cine_with_overlay这个函数），也可能是用另外的文件存储（参考save_lge），需要根据情况修改代码。如果没有标签直接保存图就可以。

这一步尽量将能提取出nifti的病例全部提取出来，可能需要人工反复检查和修改代码。

3、检查nifti数据

```
# 3. check and fix (manual)
check_abnormal_lge(root=save_path)

check_abnormal_4ch(root=save_path)

error = check_size(dataroot=save_path)
error_process(error_map=error)
```

检查所有模态的数据，每个模态的一般标准如下：

SAX CINE：每个病例包含多个slice，每个slice都是一个nii文件，包含25-30帧。

4CH CINE：每个病例只有一个4CH nii文件，文件帧数在25-30帧。

SAX LGE：每个病例只有一个LGE nii文件，帧数一般在5-10帧（后面会统一到9）。

需要修改代码检查对应的项，有些多余数据要删除。

check_size和err_process是对sax cine的每个slice检查帧数和文件尺寸，并将某个尺寸不一致的文件直接移除。

4、整理nifti到统一格式

```
# 4. fix frames, rename SAX cine, normalize LGE frames
fix_fps(nifti_path = save_path)

rename_mid(save_path)

batch_reslice(root=save_path)
```

fix_fps是将sax cine和4ch cine的帧数统一，函数里设置要统一到25还是30。

rename_mid是对sax cine标记出我们需要的up、mid和down，运行之后文件名会被修改。（up的心脏更大，down的心脏更小）

batch_reslice是对SAX LGE的帧数统一到9。

5、resample到需要的分辨率

```
# 5. final step, resample, only when all problems were fixed
error = resample_dataset2(datapath=save_path, spacing=[1.826])
```

原图的分辨率不统一，需要resample到两种分辨率上，0.994和1.826，这一步会生成两种分辨率的nii文件

**、打印信息

```
# **. check data and print info csv
info = check_data_loss(nifti_path=save_path, dcm_path=new_path)
# print(info)
chart = pd.DataFrame(info, columns=['ID', 'Name', 'SAX_data', '4CH_dat
chart.to_csv('I:/CMR-China-new/RJ_data/nifti_data/check_loss.csv', ind
chart.head(10)
# error chart
chart = find_error(root_path=save_path, out_name='G:/NM2023-FW-additio
# info chart
all_info=slice_stats(save_path)
print(len(all_info))
chart = pd.DataFrame(all_info, columns=['Class', 'Name', '4CH_data', '
chart.to_csv('G:/NM2023-FW-additional-testing/2-5-data/info.csv', inde
chart.head(10)
```

根据数据情况打印出对应的信息，可以在运行前几步时配合使用，方便检查数据异常。

数据预处理完成，会生成类似下面格式的数据集：

nii_data_spacing_0.994/SAX_data/000001/slice_up.nii.gz, slice_mid.nii.gz, slice_down.nii.gz ...

nii_data_spacing_0.994/4CH_data/000001/slice_4ch.nii.gz

nii_data_spacing_0.994/SAX_LGE_data/000001/slice_fps_9.nii.gz

nii_data_spacing_1.826（一样）

二、将台式机的数据传到bitahub

可以建立bitahub的ssh连接传，也可以借助睿客网盘中转（速度可以达到2~3MB/s），下面的步骤是第二种。

1、将需要传输的数据集压缩成zip文件

2、将zip文件传到睿客网盘的合适位置

睿客网地址：https://rec.ustc.edu.cn/

帐号：T2159

密码：Joyceustc85526377!#@

3、获取睿客网盘上数据集的下载链接，选中zip文件点击下载再暂停，将链接复制下来。

4、在bitahub上开一个调试，在命令行中下载文件。

先cd到目标路径，再运行下面的代码，curl后面替换成网盘的下载链接，最后-o后面设置传输文件的名字。等待下载完成即可。

```
# if curl not found
# apt-get install curl unzip

curl 'https://recstore.ustc.edu.cn/file/20240319_b1a35ea6db868323d4f26db8b469455
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
-H 'Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7' \
-H 'Connection: keep-alive' \
-H 'Cookie: sduuid=2ec405317be78ac5fda4415396c80969; _ga=GA1.3.1436782938.166245
-H 'Referer: https://rec.ustc.edu.cn/' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-site' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
-H 'sec-ch-ua: "Chromium";v="110", "Not A(Brand";v="24", "Google Chrome";v="110'
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Windows"' \
--compressed -o all_mid_lvef_e33.pth
```

三、提取和检查数据ROI

先使用nnUnet模型自动提取各个模态的ROI，再人工检查和修复异常ROI。

1、将nii文件处理成nnUnet模型需要的格式，运行**/code/data_process/nnunet_process.py**中的代码，**需要根据数据路径、nnUnet task ID、以及nnUnet中的存放路径修改**。SAX cine模态仅用up的slice分割。

2、运行nnUnet，生成ROI（分割结果）。

这里附上王玉峰整理的nnUnet教程（Brief usage methods of nnUNet），对于CMR数据，可以直接使用之前训练好的模型，在新数据上运行test即可（参考For VST里面的指令）。

3、分割结果后处理，运行**/code/data_process/process_ROI.ipynb**中的代码，将nnUnet输出的结果处理成annotation，保存在之前的pkl文件中。

4、人工检查新提取的ROI。

这一步需要可视化打印下来检查，异常情况大致有两种：原图错误和ROI标注错误。

对于原图错误，可以根据病例ID检查提取出的nii是否有误，nii错误的话需要重新提取，即重走一遍数据处理的步骤。

对于ROI标注错误，可以使用手动标注（用3D Slicer软件标注出segmentation文件，再根据segmentation更新pkl），或者直接修改pkl文件（读出该nii文件的annotation，手动修改几个坐标值再保存），两种方式需要

的代码全部在**/code/data_process/process_ROI.ipynb**中。

5、异常全部修复之后数据便可以使用。

Brief usage methods of nnUNet

adapt to CMR-AI Date: 02/26/2023

Training Pipeline

Dataset format

- imagesTr
- imagesTs(Optional)
- labelTr
- dataset.json

How to prepare dataset and experiment

Please read the reference of nnUNet

dataset.json

Please use the code provided within the file `nnUNET\\nnUNet\\nnunet\\dataset_conversion\\utils.py` to customize your `dataset.json`

Start experiments

- Install nnUNet from source:

```
cd nnUNET
pip install -v -e ./
```

- Set workdirs of nnUNet:

```
export nnUNet_raw_data_base="nnunet_workdir/nnunet_dataset"
export nnUNet_preprocessed="nnunet_workdir/nnunet_preprocessed"
export RESULTS_FOLDER="nnunet_workdir/nnunet_trained_models"
```

- Start five-fold training (3D model: 3d_fullres; 2D model: 2d)

```
CUDA_VISIBLE_DEVICES=0 nnUNet_train [3d_fullres/2d] nnUNetTrainerV2 TaskXX
X_name 0 --npz
CUDA_VISIBLE_DEVICES=1 nnUNet_train [3d_fullres/2d] nnUNetTrainerV2 TaskXX
X_name 1 --npz
```

```
CUDA_VISIBLE_DEVICES=2 nnUNet_train [3d_fullres/2d] nnUNetTrainerV2 TaskXX
X_name 2 --npz
CUDA_VISIBLE_DEVICES=3 nnUNet_train [3d_fullres/2d] nnUNetTrainerV2 TaskXX
X_name 3 --npz
CUDA_VISIBLE_DEVICES=4 nnUNet_train [3d_fullres/2d] nnUNetTrainerV2 TaskXX
X_name 4 --npz
```

- Find best configuration of five-fold cross training (Optional)

```
nnUNet_find_best_configuration -m 2d -t XXX –strict
```

The default number of training epoch of nnUNet is **500**, if want to change it please go to `nnUNET\\nnUNet\\nnunet\\training\\network_training\\nnUNetTrainerV2.py` and change the property at line **48**: `self.max_num_epochs`

Testing Pipeline

Dataset format

Prepare your nifti data just as training pipeline do, and there is no restrictions of folders' name, you also don't have to make `dataset.json`, just provide the dataset folder containing data to be tested.

Start experiments

- Install nnUNet from source:

```
cd /data/Joyce/nnUnet/nnUNet
pip install -v -e ./ -i https://pypi.tuna.tsinghua.edu.cn/simple/
```

- Set workdirs of nnUNet:

```
export nnUNet_raw_data_base="/data/Joyce/nnUnet/nnUNet_workdir/nnunet_data
set"
export nnUNet_preprocessed="/data/Joyce/nnUnet/nnUNet_workdir/nnunet_prepr
ocessed"
export RESULTS_FOLDER="/data/Joyce/nnUnet/nnUNet_workdir/nnunet_trained_mo
dels"
```

- Start testing (3D model: 3d_fullres; 2D model: 2d)

```
CUDA_VISIBLE_DEVICES=0 nnUNet_predict -i [PATH OF TEST DATASET] -o [OUTPUT
PATH] -tr nnUNetTrainerV2 -ctr nnUNetTrainerV2CascadeFullRes -m [3d_fullre
s/2d] -p nnUNetPlansv2.1 -t TaskXXX_name
```

  The parameter `TaskXXX_name` is the experiment ID that you want to use, it should be one of your completed training experiments.

More details of nnUNet please refer to the official website of nnUNet

For VST

We have prepared some easy-using scripts to help you start testing your CMR dataset with our pre-trained segmentation task models.

The available pre-trained models are listed here:

CMR modality			3d_fullres	2d
Cinema	Short axis		531	
	Long axis	Four chamber	530	
		Three chamber	535	
LGE	Short axis		532	534
	Long axis four chamber		533	

Table of available checkpoints for CMR data segmentation tasks. There are two types of pre-trained models: 3d_fullres and 2d, and some modalities of CMR data have both types of checkpoints. The output of pre-trained nnUNet is a near-rectangle ROI, we need more post-processing steps to convert it to usable ROI data. The available checkpoints are labeled as their task ID in the table.

Testing Demo

We provided five demo data samples for every task, they are stored within the folder `test_demo` under the path of every task. To help you understand the performance of our pre-trained model, we introduce this testing demo to you.

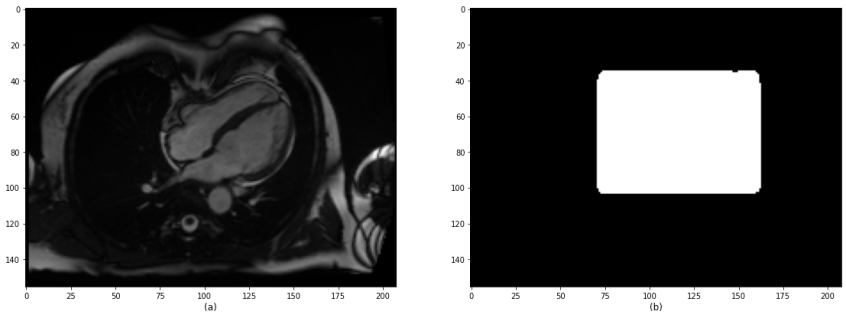
Start testing (Let 530 be an example)

```
CUDA_VISIBLE_DEVICES=0 nnUNet_predict -i /data/Joyce/nnUnet/nnUNet_workdir/nnunet_dataset/nnUNet_raw_data/Task530_4CH_SEG/4ch_ubk_1k6 -o /data/Joyce/nnUnet/nnUNet_workdir/nnunet_dataset/nnUNet_raw_data/Task530_4CH_SEG/4ch_ubk_1k6_output -tr nnUNetTrainerV2 -ctr nnUNetTrainerV2CascadeFullRes -m 3d_fullres -p nnUNetPlansv2.1 -t Task530_4CH_SEG

CUDA_VISIBLE_DEVICES=0 nnUNet_predict -i /data/Joyce/nnUnet/nnUNet_workdir/nnunet_dataset/nnUNet_raw_data/Task531_SAX_SEG/sax_ubk_377 -o /data/Joyce/nnUnet/nnUNet_workdir/nnunet_dataset/nnUNet_raw_data/Task531_SAX_SEG/sax_ubk_377_output -tr nnUNetTrainerV2 -ctr nnUNetTrainerV2CascadeFullRes -m 3d_fullres -p nnUNetPlansv2.1 -t Task531_SAX_SEG
```

```
CUDA_VISIBLE_DEVICES=0 nnUNet_predict -i /data/Joyce/nnUnet/nnUNet_workdir/nn
unet_dataset/nnUNet_raw_data/Task532_SAXLGE_SEG/lge_biobank_p1 -o /data/Joyc
e/nnUnet/nnUNet_workdir/nnunet_dataset/nnUNet_raw_data/Task532_SAXLGE_SEG/lge
_biobank_p1_output -tr nnUNetTrainerV2 -ctr nnUNetTrainerV2CascadeFullRes -m
3d_fullres -p nnUNetPlansv2.1 -t Task532_SAXLGE_SEG
```

After testing the segmentation files will be generated to the output path `nnunet_workdir/nnunet_dataset/nnUNet_raw_data/Task530_4CH_SEG/test_demo_output`.



An example of four chamber input data and ROI produced by nnUNet. a. The original four chamber data. b.The ROI produced by nnUNet.

服务器原始运行指令

1. first run LGE_fornnUnet.py or cine_fornnUnet.py to make dataset
2. creat a file like Task_502_LGE_Seg.py or Task503_4chCine_Seg.py, then run
   'python3 nnunet/dataset_conversion/Task504_3dLGE_Seg.py'
3. process dataset, run "nnUNet_plan_and_preprocess -t 502"
4. start training, run "nnUNet_train 2d nnUNetTrainerV2 505 0", ... "nnUNet_train 2d nnUNetTrainerV2 502 4"
   if been killed, run "nnUNet_train 2d nnUNetTrainerV2 502 0 -c" to continue training
5. run "nnUNet_find_best_configuration -m 2d -t 505", then open the "prediction_commands.txt" file
   in path "nnUNet_trained_models/nnUNet/ensembles/Taskxxx"
6. change the data path in command, like
   "nnUNet_predict -i FOLDER_WITH_TEST_CASES -o OUTPUT_FOLDER_MODEL1 -tr nnUNetTrainerV2
   -ctr nnUNetTrainerV2CascadeFullRes -m 2d -p nnUNetPlansv2.1 -t Task503_4chCine_Seg"
   to
   "CUDA_VISIBLE_DEVICES=5 nnUNet_predict -i
/media/ssd2/wenyi/nnUNet/dataset/nnUNet_raw/nnUNet_raw_data/Task505_saxCineSeg/imagesPresa
abnorm-f50-down/ -o
/media/ssd2/wenyi/nnUNet/dataset/nnUNet_raw/nnUNet_raw_data/Task505_saxCineSeg/imagesPresa
abnorm-f50-downRes/ -tr nnUNetTrainerV2 -ctr nnUNetTrainerV2CascadeFullRes -m 2d -p

nnUNetPlansv2.1 -t Task505_saxCineSeg"
and run it
7. finally, you can find the prediction results in output path:

"/home/wenyi/nnUNet/dataset/nnUNet_raw/nnUNet_raw_data/Task503_4chCine_Seg/imagesTsPred/"

bitahub脑肿瘤运行指令

export nnUNet_raw_data_base="/data/JoyceW/nnUNetData" &&

export nnUNet_preprocessed="/data/JoyceW/nnUNetData/nnUNet_preprocessed" &&

export RESULTS_FOLDER="/data/JoyceW/nnUNetData/RESULTS_FOLDER" &&

nnUNet_predict

-i '/data/JoyceW/nnUNetData/nnUNet_raw_data/Task504_TumorNative/imagesTsStanford2023'

-o
'/data/JoyceW/nnUNetData/RESULTS_FOLDER/nnUNet/3d_fullres/Task504_TumorNative/nnUNetTraine

-t 504 -m 3d_fullres

export nnUNet_raw_data_base="/data/JoyceW/nnUNetData" &&

export nnUNet_preprocessed="/data/JoyceW/nnUNetData/nnUNet_preprocessed" &&

export RESULTS_FOLDER="/data/JoyceW/nnUNetData/RESULTS_FOLDER" &&

nnUNet_predict

-i '/data/JoyceW/nnUNetData/nnUNet_raw_data/Task504_TumorNative/imagesTsTiantanNew'

-o
'/data/JoyceW/nnUNetData/RESULTS_FOLDER/nnUNet/3d_fullres/Task504_TumorNative/nnUNetTraine

-t 504 -m 3d_fullres