Test Plan

Federated Learning of Medical Image Reconstruction

Yash Jani, Tanuj Kancharla, Izzy MacDonald, and Joshua Sheldon

1. Introduction

1.1. Purpose

The purpose of this document is to outline the test cases that will be administered to make sure all our requirements are working as they should. Outline both the inputs and outputs of each test case.

1.2. Objective

The objective of this document is to give reference to what each feature should be able to do. Outlining test cases we need to complete throughout the project to ensure that the features are running smoothly.

2. Data Generation and Reconstruction Test Cases

2.1. XCAT Phantom and XCAT+ Generation

2.1.1. Generating an XCAT

Steps:

1) Check the Organ_Params file and make sure all the parameters are as they should be
2) Run the python code with Organ_Params as the input
3) Once the code is finished, double check the new XCAT was placed in the correct folder
4) Open the XCAT with ImageJ to double check the image looks like a SPECT image

2.1.2. Automated Randomized Parameters

Steps:

1) Run the make_xcats.py file
2) Ensure that the Organ_Params file that gets made is placed in the right folder
3) Open Orgam_Param file to ensure there was randomization to the 6 parameters
4) Open XCAT with ImageJ to double check they look like SPECT images

2.1.3. XCAT to XCAT+

Steps:

1) Run the xcat_plus.py file, having the following inputs: <XCAT file> <control/diseased> <rest/stress> (our goal is to be able to generate diseased XCAT+ images but for the moment they are generally control)

2) Once code is finished, ensure that the new XCAT+ image is in the correct file.
3) Open the XCAT+ file with ImageJ to double check the tracer distribution is correct for the heart, liver, left ventricle, and right ventricle

2.1.4. Introducing Lesions

Steps:

1) Update Organ_Params file to allow for lesions to be in XCAT
2) Run make_xcat.py
3) Ensure updated output Organ_Params file has changes to the lesion parameter
4) Use ImageJ to open the XCAT
5) Inspect image for lesion

2.2. OpenGATE Simulation

2.2.1. Simulate SPECT and Generate Sinograms

Steps:

1) The XCAT+ file is placed into a place where another Python script is run to take the XCAT+ file and runs OpenGATE
2) The sinogram is outputted in another directory
3) Steps 1 and 2 are repeated, but the angle of rotation is changed by 1, so that there is 120 sinograms
4) As of now, step 3 is repeated 5 times for each XCAT+

2.2.2. Validate Dimensions

Steps:

1) I don't think it makes sense for there to be a validate dimensions section as it's already kind of incorporated within the OpenGATE cuz it outputs a specific type of resolution, which I'm trying to change or I guess we can also say that here?

2.2.3. Configuration for Lesions

Steps:

1) The steps required to simulate SPECT scans and generate Sinograms are the same

2.3. Data Augmentation

2.3.1. Squashing

Steps:

1) The output of the OpenGATE simulation is a 128x128x240 image, so we squash the information into a 128x128x120 image using a script

2.3.2. Gaussian Blur

Step:

1) Run a script to apply a Gaussian Blur to both components of every input-output pair
2) Compare the pairs without Gaussian Blur to those with, of the same input pairs

2.3.3. Z-Shift

Steps:

1) Run a script to apply a Z-shift, a clockwise rotation to both components of every input-output pair anywhere from -15 to 15 degrees 5 times for each input-output pair (5 times to each XCAT+)

2.3.4. Filtered Back Projection

Steps:

1) Run a script to apply filtered back projection (inverse radon transformation) to both components of every input-output pair on the "XY plane" and apply 10 random shifts to each input-output pair, creating 10 more pairs of data (in a total of 11, including the unshifted)
2) Then, apply forward projection to the input and return it to a series of sinograms

3. Federated Learning Framework Test Cases

3.1. Federated Learning Framework

3.1.1. Orchestrator Application

3.1.1.1. Global Model Upkeep (Req 3.2.1.1-3.2.1.3)

Steps:

1) Have contributor account upload data to the system
2) Have the system train based on the data uploaded
3) Ensure that the updated global model was sent back to the contributor account
4) Repeat step 1-3 as many times as you want, checking if the global model was sent back after each training period

3.1.1.2. Graphical Interface

Steps:

1) Create learning manager account
2) Create contributor account
3) Switch back to learning manager account, approve of the contributor account (ensure this is possible)
4) Switch to contributor account and upload data
5) Switch to learning manager account, ensure there is a way to start training
6) Ensure there is a way to monitor training progress
7) Ensure there is a way to pause the training
8) Ensure there is a way to then begin the training where it left off
9) Ensure that there is a way to check approved contributors

3.1.1.3. Logging and Error Handling

Steps:

1) Have contributor account upload incorrect data to the system
2) Ensure that the systems notices this and tells the contributor account there is an error
3) Have contributor account upload correct data
4) When model finished training, ensure that there is a log stating when the contributor account uploaded data and when the system trained on it
5) Ensure that it is logged when the contributor account attempted to upload incorrect data

3.1.2. Contributor Application

3.1.2.1. Training (Req 3.2.2.1-3.2.2.5)

Steps:

1) Have contributor account upload data
2) Ensure that only parameter updates get sent to the model, not private information
3) Ensure data being uploaded is real SPECT data and synthetic data
4) Have the model train on the data
5) Ensure that the global model is sent to the contributor account once training is finished
6) Ensure there is a log that lists when and who uploaded data, when the model trained on the data, when the model was finished training, then the global model was then shared with contributor applications

4. Performance Test Cases

4.1. Image Reconstruction

Steps:

1) Run the Python code that call the neural network - include a timer (gets outputted)
2) Once code completed, check if the outputted time is under 5 secs per instance
3) Validate the outputs files against a synthetic sinogram
4) Validate the outputs files against a real sinogram

4.2. Federated Learning Cycle

4.2.1. Communication

Steps:

1) Have contributor accounts from more than one machine upload data
2) Ensure that the system receives all data from the different machines
3) Start the training on the data
4) Have a couple of the machines log out of the system

5) Ensure that the training continues even with those machines logging out
6) Ensure that the each contributor account on the different machines get the updated global model

4.2.2. Duration

Steps:

1) Have the contributor account upload training data
2) Set up a timer for the training
3) Start the training for the system
4) When training done, have the timer output time
5) Ensure that the training time is efficient

4.3. Data Augmentation

Steps:

1) Add a running timer to the pipeline code
2) Run the pipeline code
3) Ensure the code is running in a timely manner
4) Ensure that all output are as they should be

5. Security and Privacy Test Cases

5.1. Data Privacy

Steps:

1) Have contributor account upload training data
2) Ensure that the system only gets parameter updates from the data, nothing private
3) Ensure that the information being transmitted is encrypted

5.2. Access Control

5.2.1. Role-based Access

Steps:

1) Create an account as a learning manager.
2) Ensure that the functions that are given access, are for a learning manager.
3) Create an account as a contributor.
4) Ensure that the functions that are given access, are for a contributor.
5) Ensure that the contributor account can not access any features of a learning manager.

5.2.2. Authentication of Contributor Applications

Steps:

1) Create an account as a contributor
2) Ensure contributors need to submit credentials that the learning manager requires
3) Have contributor account pending until learning manager account grants access
4) Switch to learning manager account
5) Have a pop-up that the contributor account is asking for permission

6) Ensure that the learning manager account can check the credentials of the contributor account
7) The learning manager account can now grant access to the contributor account
8) Switch back to the contributor account
9) Ensure the contributor account now has the correct access since being approved

5.3. Audit and Logging

Steps:

1) Log into contributor account
2) Upload training data
3) Switch to learning account
4) Access data and train the model
5) Access the model with the trained data
6) Repeat step 1-5 as seen fit
7) Check audit log and ensure that everything done has been accounted for