Project Title: Tomographic Medical Image Reconstruction using Deep Learning

Group Members: Asher Burrell, Christopher Hinton, Ty Mercer

Faculty Advisor/Client: Dr. Debasis Mitra

Introduction

This document outlines our test plan for our Senior Design project. We will conduct two types of tests for this project: Static tests (reviewing programs, files, etc to check for correctness) and dynamic tests (testing the running software). These tests are described below.

Our test cases will follow the following formula, in order to consistently convey the relevant information about the testing process:

Test Case Title

Requirements: This is a list of requirements to be tested, listed as the requirement numbers from the requirements document. If a requirement listed here has sub-requirements (for example, 1.1.1 has 1.1.1.1 as a sub-requirement), all sub-requirements are being tested in this test case.

Steps: A list of steps to be performed for the test case

Exceptions: A list of expected exceptions that may occur during the test case execution, and what each exception means. Note that this list is not comprehensive, and just includes the examples we are specifically looking for.

Similar Tests: A list of similar tests to be done, following the same steps with slight deviations. Include for each test the deviation and expected outcome. Dynamic testing only.

Notes: Any notes we have on this particular test case.

Static Tests

Some of our requirements require static testing in order to verify that certain files are correctly formulated. These tests will still be represented as test cases, but they involve actions other than running code and messing with inputs/outputs. Many of these tests need to be conducted before some of the dynamic test cases. We will be conducting the following static tests during our testing process:

Verify Blank XCAT Phantom

Requirements

- 1.1

Steps

- Open the folder containing all of the blank XCAT phantoms (that is, the phantoms only containing the default values)
- For each XCAT phantom:
    - Open the phantom in ImageJ
    - For each organ in the phantom, move the cursor over the area representing that organ
    - Mark the values for heart, liver, left ventricle, and right ventricle
    - Investigate the other organs in the phantom to see if any of them match the value of one of these organs
- Compare the values for heart, liver, left ventricle, and right ventricle across all the XCAT phantoms, to ensure that they are the same for each phantom

Exceptions

- One phantom has the same value for heart/liver/left ventricle/right ventricle and another organ
- Two phantoms have different values for heart, liver, left ventricle, or right ventricle

Notes

- We have used these phantoms in Dr. Mitra's lab before, and they have already been verified. In practice, we will primarily be making sure the phantoms are consistent with one another, as any issues with duplicate values should be more noticeable during later tests.

Inspect Training Data

Requirements

- 3.1

Steps

- Open the folder containing training input data (sinograms) for the neural net
- Verify that all files in the folder are, in fact, sinograms
    - This includes making sure that they are dicom files

- Mark the names of the files in the folder
- Open the folder containing the training output data (reconstructed images) for the neural net
- Verify that these images are all raw binary files of the expected size (128x128x128x32)
    - .Vol file extension is expected, this is raw binary
- Verify that, for each input sinogram, there is a corresponding output image that can be identified based on the conventions used by the neural network training program
    - This should be guaranteed by the reconstruction process
- Verify that the heart is visible in a randomly selected sample of the training output images

Exceptions
- Input data is not a dicom file
- Output data is not a raw binary file
- Input data is not a sinogram
- Output data is the wrong size
- Output data does not include an image of the heart

Notes
- None

Dynamic Testing

These tests will be conducted by running some part of the system.

Verify XCAT Population

Requirements
- 1.2

Steps
- Locate the file that populates XCAT phantoms and ensure it is in Python
- Update the parameter file to include the heart, liver, left ventricle, and right ventricle, and to specify a valid SciPy distribution for each one.
    - This SciPy distribution will be based on our real data from Dr. Mitra's lab
- Run the program from the command line
- Open the output folder where the populated XCAT phantom has been generated
- Open the generated phantom in ImageJ
- Using the cursor, inspect the heart, liver, and ventricles

- Compare the values for each organ to the mean and standard deviation given by the SciPy distribution

Exceptions
- XCAT population file is not in Python
- Program encounters an exception
- No XCAT phantom is generated
- Program injects data directly into existing phantom instead of making a copy
- One or more organs was not populated with statistical data
- One or more organs was populated with the wrong statistical data

Similar Tests
- Invalid SciPy distribution
  - Program should throw an exception from the SciPy Stats package
- No organs provided
  - Program should exit without changing any files
- Organ does not have statistical data
  - Program should skip the organ that does not have data provided
- Statistical data is provided for a nonexistent organ
  - Program should ignore the extra data

Notes
- Dependant on test "Verify blank XCAT phantom"

Verify OpenGATE Sinogram Generation

Requirements
- 2

Steps
- Open the OpenGATE simulation
- Input an XCAT phantom with statistical data for the heart
- Specify a directory for the sinogram data to be placed in
- Run the OpenGATE simulation on the phantom
- Run iterative reconstruction on the generated sinogram
- Verify that the reconstructed image has a recognizable heart
- Verify that the values in the reconstructed heart are similar to the values in the XCAT phantom

Exceptions

- OpenGATE simulation has an exception
- OpenGATE simulation does not output a sinogram
- OpenGATE simulation does not output a realistic sinogram

Similar Tests
- Input a sinogram with distributions for heart and other organs (ventricles, liver)
    - Resulting sinogram should have realistic values for those organs as well

Notes
- OpenGATE simulation has been previously verified in Dr. Mitra's lab, test is primarily to ensure it has not broken.

Sanity Check for Neural Network

Requirements
- 3.2
- 3.3
- 3.4

Steps
- Place some subset of the validation set in the input folder
- Run the Python program that calls the neural network
    - Mark the time the program takes to run
- Open the output folder
- Verify that all outputs are raw binary files of the expected size (128x128x128x32)
- Verify that there is an output for every input

Exceptions
- Program encounters an exception
- No outputs
- Some outputs are missing
- Some outputs are the wrong size
- Program takes more than one second per sinogram

Similar Tests
- Place some non-sinogram dicom files in the input
    - These files should be reconstructed as though they were sinograms, and should have corresponding images
- Place some non-dicom files in the input
    - These files should be ignored

- Modify the program to have an exception while reconstructing one or more sinograms (keep track of which sinograms should throw the exception)
    - The sinograms that throw exceptions should not be reconstructed
    - An error message should be printed for every sinogram that throws an exception
- Have files in the output folder with the same name as the expected outputs
    - These duplicate files should be overwritten
- Delete the output folder
    - A new output folder should be created
- Delete the input folder
    - The program should exit without modifying any files
- Have an input folder with no dicom files
    - The program should exit without modifying any files

Notes
- Testing the neural network's basic functions, not accuracy

Check Neural Network for Accuracy

Requirements
- 3.5

Steps
- Place some subset of the validation set sinograms in the input folder
- Run the neural network program
- For each outputted image:
    - Open the image in ImageJ
    - Visually inspect the image for a recognizable heart
    - Open the ground-truth image in ImageJ
    - Visually compare the images to ensure that the heart is (approximately) in the correct place
    - Perform dice-coefficient analysis between the ground truth image and the AI reconstructed image to check for accuracy

Exceptions
- Reconstructed image has no heart
- Reconstructed image has the heart in the wrong place
- Reconstructed image has significantly different values for the heart

Notes
- Assumes the test case "Sanity Check for Neural Network" is passed.

Verify Lack of External Connectivity

Requirements
- 4

Steps
- Run some sample of previous test cases, in an environment with no internet and no external interfaces
- Verify that no files were created outside of the directories where the programs being run were located

Exceptions
- Program encounters an error due to lack of internet connectivity
- Files are created in an inappropriate location

Notes
- Requirements tested can easily be met by not using interfaces that require the internet, and by using caution when choosing file paths.